Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en **Ingeniería Informática**

Trabajo Fin de Grado

Desarrollo de Plataforma Web para la Integración y Análisis de Datos Académicos UPM: DASOS

Autora: **Henny Sánchez**
Tutor: **Sergio Paraíso Medina**

Madrid, **Julio - 2025**

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Desarrollo de Plataforma Web para la Integración y Análisis de Datos Académicos UPM: DASOS

Julio - 2025

Autora: Henny Sánchez
Tutor: Sergio Paraíso Medina
Dto. de Lenguajes, Sistemas Informáticos e Ingeniería del Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

A mi familia, mis hermanas y padres, para que valoren toda la vida.

A mis amigas, porque ellas también me han apoyado en todo.

A mi Delegación y a mis asociaciones, porque aunque lo niegue, también son mi familia. Porque ahí aprendí a ser, a ver y a tener. Porque mi hogar también estará con ellos, donde siempre rajaré, reiré y seré.

A mi RITSI, porque aunque cerré mi camino con ellos, fueron los primeros que me descubrieron y mostraron mi potencial. Me aceptaron y quisieron hasta el final, así como les quiero yo a ellos.

A la ACM, porque cambiaron mi camino para siempre en la universidad, porque incluso después de todo, sé que los quise y aprecié más que a nadie.

A RugFI, porque me abrieron y recibieron como una más, y siempre tendrán sitio conmigo.

A ASCFI, porque ellos me recuerdan de donde vengo, y como pueden seguir.

A mi equipo, porque así como les enseñé el valor de la representación, ellos me devolvieron las ganas y voluntad para representar.

A mis estudiantes, porque aunque me saquen los estribos muchas veces, son la motivación principal por la que siga donde estoy, y quienes me han vuelto quien soy. Porque este proyecto es para ellos.

A mis no estudiantes, porque mis amistades también han sido del personal de limpieza, de conserjería, de mantenimiento, de cafetería y de servicios, porque ellos me han apoyado en formas que nadie más sabe, y por eso se merecen esto.

A mis amistades docentes, porque sois quienes me habéis (con vuestro ejemplo) mostrado los distintos caminos de mi futuro, porque me habéis guiado y apoyado, y porque seguiré con vosotros riendo, aprendiendo y haciendo.

Y por último, a mí. Porque después de todo, nena, sigues siendo tú, frente a cualquier adversidad. Porque has podido hasta ahora, y seguirás pudiendo hasta el fin. Porque el mayor beso va para tí.

Un beso.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Listings

Resumen

En el marco de la digitalización y modernización de la educación superior, la capacidad de integrar y analizar grandes volúmenes de datos académicos se ha consolidado como una herramienta estratégica para la mejora institucional de la calidad en el marco de la educación y la enseñanza. En particular, la Universidad Politécnica de Madrid (UPM) dispone de múltiples fuentes de información académica, como informes semestrales, de asignaturas, de cursos; bases de datos departamentales, y otros servicios digitales. Sin embargo, la falta de un sistema unificado dificulta la explotación eficiente de estos datos para el análisis cuantitativo del rendimiento estudiantil, la evaluación de metodologías docentes o la planificación académica basada en evidencia, piezas clave para que los procesos de calidad presentes sean aprovechados. Ante este panorama, resulta fundamental desarrollar plataformas digitales que favorezcan la centralización, visualización y análisis avanzado de información, contribuyendo a una gestión académica más informada, transparente y orientada a la mejora continua.

Este Trabajo de Fin de Grado presenta el diseño y desarrollo de una plataforma web orientada a la integración y análisis de datos académicos de la UPM llamada **D**atos **A**cádemicos: **S**eguimiento, **O**rden y **S**íntesis (***DASOS***). La iniciativa surge ante la necesidad de centralizar información académica previamente dispersa en varios sistemas institucionales, con el objetivo de ofrecer una herramienta de valor tanto para el alumnado como para el personal docente e investigador.

Los resultados obtenidos evidencian que la solución implementada proporciona un valor añadido significativo para la comunidad universitaria, al facilitar una toma de decisiones más informada por parte del profesorado y ofrecer a los estudiantes una visión más completa de su trayectoria académica. Asimismo, la solución técnica desarrollada ha demostrado ser escalable y fácilmente adaptable a futuras necesidades o nuevas fuentes de datos.

Palabras Clave: análisis de datos académicos, visualización de datos, desarrollo web, Python, Next.js, API REST, extracción de datos, rendimiento académico, correlación estadística, toma de decisiones basada en datos.

Abstract

In the context of the digitization and modernization of higher education, the ability to integrate and analyze large volumes of academic data has been consolidated as a strategic tool for institutional improvement. In particular, the Universidad Politécnica de Madrid (UPM) has multiple sources of academic information, such as semester, courses and academic year reports, departmental databases and web services such as the GAUSS platform and API. However, the lack of a unified system hinders the efficient exploitation of these data for longitudinal analysis of student performance, evaluation of teaching methodologies or evidence-based academic planning, key pieces to exploiting current quality processes. Against this backdrop, it is essential to develop digital platforms that favor the centralization, visualization and advanced analysis of information, contributing to a more informed, transparent and continuous improvement-oriented academic management.

This Final Degree Project presents the development of a web platform designed for the integration and analysis of academic data from the UPM, called ***DASOS*** (Academic Data: Synthesis, Order and Tracing). The project addresses the need to centralize academic information scattered across different systems, providing a tool that benefits both students and teaching staff.

The results obtained show that the implemented solution provides significant added value for the university community, facilitating informed decision-making by faculty and offering students a more complete view of the subjects they are taking or plan to take. The technical solution developed proves to be scalable and adaptable to future needs or additional data sources.

Keywords: academic data analysis, data visualization, web development, Python, Next.js, REST API, data extraction, academic performance, statistical correlation, data-driven decision-making.

Capítulo 1

Introducción

Desde el año 2016, la Universidad Politécnica de Madrid (UPM) ha desempeñado un papel fundamental en la digitalización de la información y recursos académicos ofrecidos a la comunidad universitaria. Este proceso ha abarcado desde la renovación de diversas plataformas hasta la creación de nuevas instituciones digitales, así como la integración de estas para proporcionar una síntesis más completa de información. Entre estas iniciativas destacan las plataformas Moodle[1] y GAUSS[2] para los estudiantes, o el Observatorio I+D+i[3] para el personal docente e investigador.

En la vida académica diaria de estudiantes y docentes, el acceso ágil y comprensible a los datos sobre asignaturas, metodologías de evaluación y resultados académicos sobre las mismas es esencial para una toma de decisiones informada y para el seguimiento continuo del progreso de calidad educativo. No obstante, esta información se encuentra frecuentemente dispersa en distintos sistemas, lo que dificulta su consulta y análisis por parte de la comunidad universitaria. Esta fragmentación limita tanto la capacidad del profesorado para evaluar el impacto de sus prácticas docentes como la del estudiantado para planificar adecuadamente su trayectoria académica. En este contexto, se hace necesario contar con herramientas digitales que integren y presenten esta información de forma clara, interactiva y basada en datos, facilitando una experiencia académica más transparente, personalizada y eficiente.

Esta situación genera un ciclo de desinformación y confusión para el estudiantado, que frecuentemente debe recurrir a compañeros para obtener este tipo de información, evidenciando la ineficacia del sistema actual de recursos en línea. Para el profesorado, estas limitaciones dificultan la identificación de problemáticas y sus causas subyacentes en las asignaturas impartidas, imposibilitando una evaluación efectiva y la implementación de mejoras. A mayor escala, se produce un distanciamiento progresivo en la relación docente-estudiante, una dinámica fundamental para fomentar el conocimiento e interés del alumnado en sus respectivas titulaciones y sus campos de estudio.

El presente proyecto se centra en el desarrollo de una plataforma que integre y analice de manera comprensiva aquellos datos de consulta habitual para el estudiantado, así como estadísticas de utilidad para el personal docente en la comprensión del desarrollo de sus asignaturas. Se pretende establecer un punto de convergencia donde ambos colectivos puedan obtener información relevante sobre el otro, potenciando sus capacidades y permitiendo que, desde una única

[1] https://moodle.upm.es/
[2] https://www.upm.es/gauss/
[3] https://www.upm.es/observatorio/vi/index.jsp

interfaz, tanto unos como otros accedan a información valiosa de manera eficiente y accesible.

1.1. Objetivos

El objetivo principal y solución propuesta de este trabajo es el desarrollo de una herramienta web diseñada específicamente para la extracción y análisis de información académica de la Universidad Politécnica de Madrid (UPM). De este objetivo general, se pueden extraer los siguientes más específicos, que reflejan de forma concisa el desarrollo del proyecto:

- Realizar un análisis de la situación actual de las plataformas académicas de la UPM disponibles tanto para el estudiantado como los docentes, con el objetivo de conocer su funcionamiento y sus limitaciones, extrayendo los datos más relevantes, desarrollando una visión detallada y adquiriendo un conocimiento general de estos sistemas.
- Recopilar, mediante técnicas de extracción, información de estas plataformas, como las API[4] que usan, los datos que almacenan, y qué integraciones reportan para estructurar la información disponible.
- La gestión de una o varias bases de datos para almacenar y organizar los datos recolectados.
- Si es necesario, desarrollar una API que permita obtener estos datos y agregarlos.
- Centralizar esta información para facilitar la generación dinámica de reportes y estadísticas.
- Desarrollo de una aplicación que pueda mostrar toda esta información, así como gráficas y/o informes que sean de relevancia para los interesados.

1.2. Planificación

Al principio, este proyecto estaba previsto para realizarse en el primer semestre del curso académico, sin embargo, debido a circunstancias extraordinarias, su desarrollo se aplazó para así presentarlo en la convocatoria extraordinaria.

En las siguientes figuras se presentan los diagramas de Gantt principal y final sobre la distribución de horas de trabajo dedicadas a realizar el proyecto:

[4]Conjunto de reglas y protocolos que permite la comunicación entre diferentes componentes de software.

Figura 1.1: Diagrama de Gantt Inicial del proyecto DASOS (Septiembre a Diciembre)

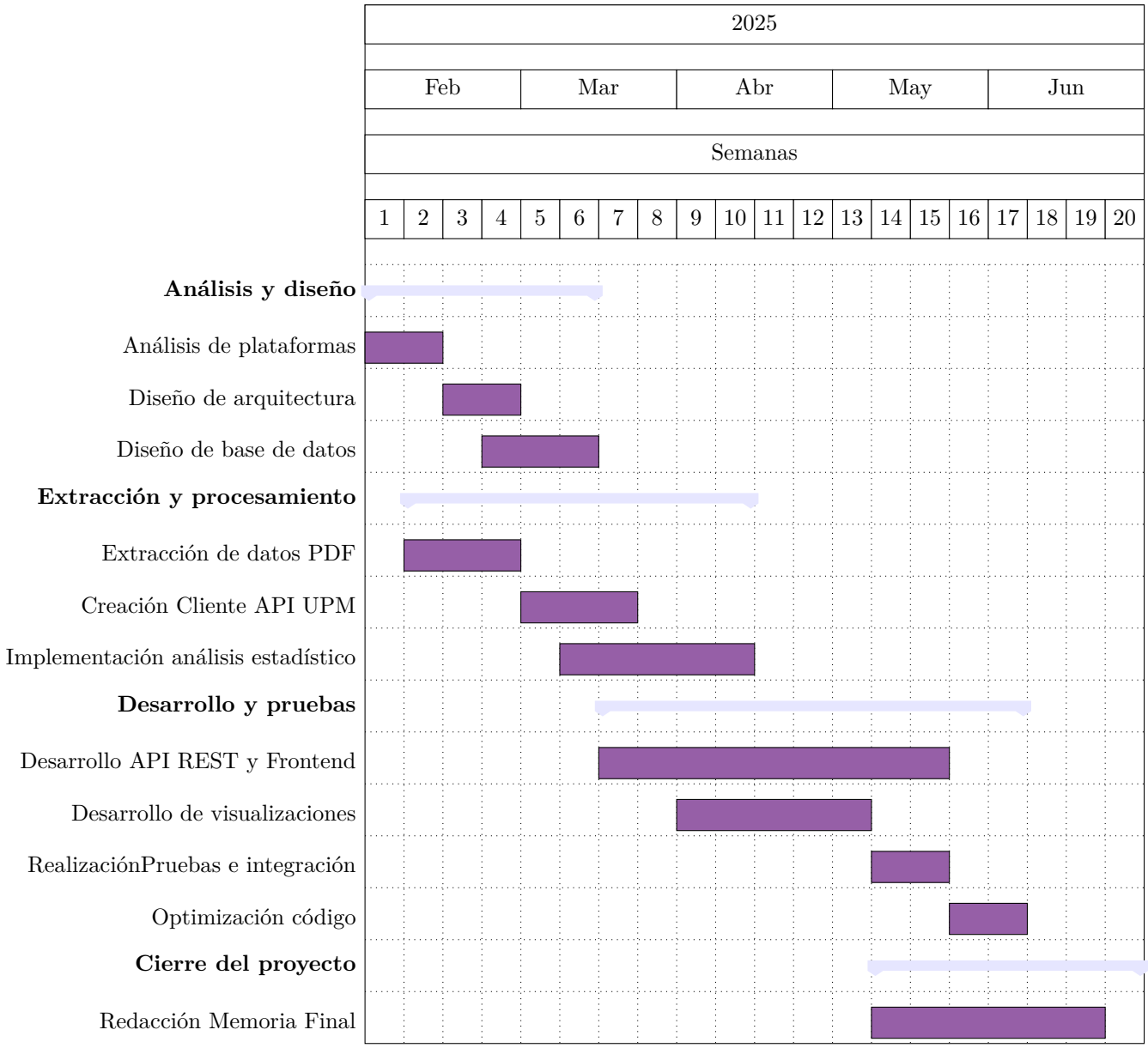

Figura 1.2: Diagrama de Gantt final del proyecto DASOS (Febrero a Junio)

1.2.1. Fases principales

El desarrollo se ha estructurado en cinco fases principales, correspondientes a diferentes objetivos y actividades:

1. **Fase 1: Análisis y diseño inicial** (Febrero 2025)
   - Investigación de fuentes de datos académicos disponibles
   - Análisis de plataformas existentes y sus limitaciones
   - Definición de requisitos funcionales y no funcionales
   - Diseño arquitectónico preliminar
   - Selección de tecnologías y herramientas

2. **Fase 2: Extracción y estructuración de datos** (Marzo 2025)
    - Desarrollo de módulos de extracción para informes PDF
    - Implementación de cliente para API UPM
    - Diseño e implementación de esquema de base de datos
    - Desarrollo de procesos ETL[5] para integración de datos
    - Pruebas de validación de datos extraídos
3. **Fase 3: Desarrollo de componentes analíticos** (Marzo-Abril 2025)
    - Implementación de algoritmos estadísticos básicos
    - Desarrollo de análisis de series temporales
    - Implementación de métodos de correlación
    - Desarrollo de generación de insights
    - Pruebas de validación de resultados analíticos
4. **Fase 4: Desarrollo de interfaz y visualizaciones** (Abril-Mayo 2025)
    - Implementación de API REST para acceso a datos
    - Desarrollo de componentes base de la interfaz
    - Implementación de visualizaciones interactivas
    - Desarrollo de flujos de usuario completos
    - Pruebas de usabilidad y ajustes
5. **Fase 5: Integración, pruebas y documentación** (Mayo-Junio 2025)
    - Integración completa de componentes
    - Pruebas de sistema end-to-end
    - Optimización de rendimiento
    - Desarrollo de documentación técnica y de memoria
    - Preparación para despliegue y entrega

1.2.2. Cronograma

El desarrollo del sistema se ha realizado entre febrero y junio de 2025, con la siguiente distribución temporal:

[5]Proceso de obtención, transformación y carga de datos desde diversas fuentes hasta un repositorio central.

Figura 1.3: Cronograma de desarrollo del sistema GaussUPM

1.3. Estructura del Documento

Esta memoria cubre el trabajo realizado en este TFG, dividiéndose en los siguientes apartados, los cuales cubren los objetivos establecidos en la Sección *1.3 Objetivos*.

- En el **Capítulo 2: Trabajo relacionado y Estado del Arte**, se presenta el trabajo externo que se ha llevado a cabo previamente en este proyecto, así como la situación actual en el análisis de información académica en entornos educativos, y los elementos técnicos involucrados en el desarrollo e implementación de herramientas para el mismo campo, además de los problemas actuales de estas soluciones.
- El **Capítulo 3: Análisis de Plataformas Académicas de la UPM** presenta un estudio sobre el estado actual de las plataformas académicas de la Universidad Politécnica de Madrid (UPM).
- A continuación, en el **Capítulo 4: Extracción y Estructuración de Información Académica**, se abordan las técnicas de recopilación de datos utilizadas en este proyecto. Asimismo, se describirá la estructura de la información obtenida, detallando los métodos empleados para organizar y estructurar la información.
- El **Capítulo 5: Diseño y Gestión de Bases de Datos** se centra en la gestión de las bases de datos destinadas a almacenar y organizar los datos recolectados. Aquí se explica el diseño de la base de datos, su integración con el sistema de análisis y extracción y la implementación de una API que permite la obtención y agregación de estos datos de manera centralizada.
- En el **Capítulo 6: Desarrollo**, se detalla el proceso de centralización de la información recolectada y su integración en una aplicación web diseñada para mostrar toda la información recopilada
- A continuación, en el **Capítulo 7: Impacto del trabajo**, se relacionan las consecuencias y el impacto general de este proyecto, así como su relación con los Objetivos de Desarrollo Sostenible, y en cuáles se enmarca el mismo.
- Finalmente, en el **Capítulo 8: Resultados y conclusiones** se presenta la información obtenida gracias a este proyecto y los resultados que ha generado. De igual manera, se explican las conclusiones personales de la autora, las futuras mejoras que podrían implementarse en la aplicación para optimizar la extracción de datos, y futuras líneas de investigación y desarrollo posibles.

Capítulo 2

Estado del arte

El presente capítulo expone una revisión exhaustiva del estado del arte en las diversas áreas tecnológicas y metodológicas que fundamentan el desarrollo del sistema propuesto. Este análisis bibliográfico aborda tanto los fundamentos teóricos como las implementaciones prácticas que han configurado el campo del análisis de datos académicos y las tecnologías asociadas a su procesamiento, almacenamiento y visualización.

2.1. Análisis de datos académicos

El análisis de datos académicos constituye un campo multidisciplinar [1] en evolución, que integra métodos estadísticos avanzados, técnicas de minería de datos y enfoques de aprendizaje automático para extraer conocimiento significativo a partir de conjuntos de datos educativos. Esta disciplina ha experimentado un notable crecimiento en la última década, impulsada por la digitalización de los procesos educativos y la consecuente generación de grandes volúmenes de datos susceptibles de análisis.

2.1.1. *Learning Analytics* (Analítica del Aprendizaje)

Learning Analytics ("Analítica del Aprendizaje") representa una aproximación sistemática al estudio de los datos educativos, definida formalmente por Siemens como "la medición, recolección, análisis e informe de datos sobre los estudiantes y sus contextos, con el propósito de entender y optimizar el aprendizaje y los entornos en los que ocurre"[2]. Este paradigma analítico trasciende la simple recopilación de estadísticas educativas para adentrarse en la comprensión profunda de los procesos de aprendizaje y sus determinantes.

La naturaleza interdisciplinar de *Learning Analytics* se manifiesta en la integración de metodologías y conocimientos procedentes de diversos ámbitos. El análisis estadístico proporciona las herramientas fundamentales para el procesamiento cuantitativo de los datos, mientras que la minería de datos educativa (*Educational Data Mining*) [3, 4] aporta técnicas específicas para descubrir patrones no evidentes en grandes conjuntos de datos educativos. La visualización de información facilita la interpretación y comunicación de resultados complejos, adaptándolos a diferentes perfiles de usuarios. El aprendizaje automático contribuye con algoritmos que pueden identificar patrones.

Las aplicaciones de *Learning Analytics* se extienden a múltiples dimensiones del proceso educativo. La identificación temprana de estudiantes en riesgo posibilita la implementación de mecanismos de apoyo específicos. La personalización de la experiencia educativa adapta contenidos

y metodologías a las necesidades particulares de cada estudiante. La evaluación sistemática de métodos didácticos proporciona evidencias objetivas sobre su efectividad. Por último, la mejora de planes de estudio basada en datos facilita la toma de decisiones curriculares informadas.

2.1.2. Métricas de rendimiento académico

Las **métricas de rendimiento académico** constituyen instrumentos clave para la evaluación objetiva y sistemática de los procesos educativos. A través de estos indicadores cuantitativos, es posible analizar de forma estructurada el desempeño tanto institucional como estudiantil, permitiendo la realización de estudios **comparativos** (entre asignaturas, titulaciones o universidades) y **longitudinales** (a lo largo del tiempo).

En el contexto del sistema universitario español, estas métricas han sido **estandarizadas por organismos como la Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA)** [5], lo que garantiza su consistencia metodológica y su utilidad en procesos de seguimiento y mejora continua. Entre las métricas más relevantes, se encuentran:

Tasa de Rendimiento: Se define como el **porcentaje de créditos aprobados respecto al total de créditos matriculados**. Este indicador ofrece una visión general sobre la **eficiencia del proceso educativo**.

Tasa de Éxito: Mide el **porcentaje de créditos aprobados sobre los créditos presentados a evaluación**, excluyendo del cálculo a los estudiantes no presentados. Este indicador se centra en los estudiantes que **participan activamente en los procesos de evaluación**.

Tasa de Absentismo: Se define como el **porcentaje de estudiantes matriculados que no se presentan a la evaluación**. Esta métrica permite identificar fenómenos de **abandono parcial**.

2.2. Extracción y procesamiento de datos de documentos

La extracción de información estructurada a partir de documentos no estructurados o semiestructurados constituye un desafío técnico en el ámbito del análisis de datos. En el contexto académico, esta problemática adquiere relevancia dado que gran parte de la información institucional se encuentra contenida en informes, memorias y documentos oficiales en formato PDF, cuya estructura facilita la lectura humana, pero dificulta el procesamiento automatizado.

2.2.1. Técnicas de extracción de texto de PDFs

El formato PDF (*Portable Document Format*) fue diseñado primariamente para la presentación visual consistente de documentos, priorizando la fidelidad visual sobre la accesibilidad estructural de los datos. Esta característica fundamental del formato genera retos significativos para los sistemas de extracción automática de datos.

La extracción directa mediante bibliotecas especializadas constituye la aproximación más inmediata. Herramientas como PyMuPDF (fitz) [6], que ha sido implementada en DASOS, permiten acceder a la estructura subyacente del documento PDF y extraer su contenido textual. La principal ventaja de estas aproximaciones radica en su eficiencia y en su capacidad para extraer textos correctamente formateados.

2.2.2. Procesamiento de texto con expresiones regulares

Las expresiones regulares (*regex*) constituyen un lenguaje formal para la descripción y búsqueda de patrones en textos. Su potencia radica en la capacidad para especificar patrones complejos mediante una sintaxis concisa y altamente expresiva. En el contexto de la extracción de datos académicos, las expresiones regulares ofrecen un mecanismo preciso para identificar estructuras específicas dentro del texto no estructurado extraído de documentos PDF.

2.3. Bases de datos para análisis académico

2.3.1. SQLite

Para el almacenamiento persistente de datos, el proyecto utiliza **SQLite**, un sistema de gestión de bases de datos relacional (RDBMS) [7, 8] con características que lo distinguen significativamente de soluciones tradicionales como MySQL o PostgreSQL. Desarrollado por **D. Richard Hipp en el año 2000**, SQLite se ha convertido en una de las bases de datos más utilizadas a nivel mundial, debido a su arquitectura **ligera**, **autocontenida** y **altamente portable**.

2.4. Visualización de datos educativos

2.4.1. Bibliotecas de visualización de datos

El ecosistema tecnológico actual ofrece una gran variedad de herramientas para la visualización de datos, cada una con características que las hacen más o menos adecuadas según el contexto. En Python, lenguaje clave en el backend del proyecto desarrollado aquí, destacan principalmente **Matplotlib** y **Seaborn** [9, 10, 11, 12].

Matplotlib Creada por John D. Hunter en 2003, es la biblioteca base para la mayoría de las visualizaciones en Python. Permite un alto nivel de personalización, aunque requiere más código y experiencia para producir gráficos incluso simples. En la plataforma propuesta, se utiliza para generar visualizaciones estáticas que requieren un control preciso sobre su diseño.

Seaborn Construida sobre Matplotlib, simplifica la creación de gráficos estadísticos mediante una interfaz más accesible y diseños predeterminados estéticamente cuidados. Se emplea en el desarrollo de la solución para generar gráficos como mapas de calor o comparativas de barras de forma eficiente y visualmente clara.

En el frontend, basado en JavaScript, se ha elegido **Chart.js** como biblioteca principal [13]. Esta herramienta de código abierto permite crear gráficos interactivos en HTML5 con animaciones fluidas, buena compatibilidad con dispositivos y una curva de aprendizaje suave. Su integración con frameworks como React[1] la hace ideal para desarrollos web modernos.

2.5. Frameworks de desarrollo web

El desarrollo web contemporáneo ha experimentado una evolución significativa en la última década, transitando desde aproximaciones centradas en el renderizado en servidor hacia arquitecturas de aplicaciones de página única (SPA[2]) [14] y, más recientemente , hacia soluciones híbridas que combinan las ventajas de ambos paradigmas[15]. Esta evolución ha generado un

[1] Biblioteca JavaScript para construir interfaces de usuario basada en componentes.
[2] SPA: Single Page Application

ecosistema rico en frameworks y herramientas especializadas, cada uno con características distintivas que determinan su idoneidad para diferentes contextos de aplicación.

2.5.1. React y Next.js

En este contexto, React y Next.js[3] forman una base tecnológica sólida para aplicaciones como la implementada en este proyecto.

React, creada por Meta en 2013, permite construir interfaces mediante componentes reutilizables que encapsulan lógica y presentación. Esto favorece la organización del código y su mantenimiento. Además, su uso del Virtual DOM mejora el rendimiento al minimizar las modificaciones directas del DOM real. React también promueve un flujo de datos unidireccional, lo que facilita el control del estado de la aplicación. En el proyecto, esta arquitectura se utiliza para crear elementos como gráficos o tarjetas reutilizables.

JSX, una extensión de JavaScript, permite escribir estructura HTML directamente dentro del código, lo cual facilita el desarrollo. Combinado con TypeScript (TSX), que añade tipado estático a Javascript, mejora la detección de errores y el autocompletado en los editores de código, así como la opción de definir estructuras de datos necesarias para almacenar presentar la información dedicada y la refactorización de código.

Next.js, framework basado en React [16, 17] , incorpora funcionalidades avanzadas como:

Renderizado en servidor (SSR) SSR[4] es útil para mejorar el SEO[5] y la carga inicial, especialmente en vistas públicas de DASOS.

Generación estática (SSG) SSG[6] es ideal para contenido que cambia poco, como dashboards históricos.

Enrutamiento automático Basado en la estructura de archivos, facilita la organización del proyecto.

API Routes Permite crear endpoints sin necesidad de un servidor backend separado, lo que simplifica la arquitectura.

Optimización de imágenes Mejora el rendimiento visual adaptando imágenes al dispositivo del usuario.

2.6. APIs REST y arquitecturas de microservicios

La evolución de los sistemas distribuidos ha conducido al desarrollo de diversos estilos arquitectónicos optimizados para diferentes contextos y requisitos. Entre estos, REST [7] (*Representational*

[3]Framework de React que proporciona capacidades avanzadas como renderizado del lado del servidor y generación de sitios estáticos.

[4]Server-side rendering: Técnica que consiste en generar HTML completo en el servidor para la carga inicial de la página.

[5]Search Engine Optimization: .ºptimización para motores de búsqueda". Consiste en una serie de técnicas, disciplinas y estrategias de optimización que se implementan en las páginas de un sitio web o blog para mejorar su posicionamiento en los buscadores.

[6]Static-Site Generating: Es el software que crea las páginas HTML a partir de ciertos templates o fuentes de información.

[7]Interfaz de programación que sigue los principios de la arquitectura REST, utilizando métodos HTTP estándar para realizar operaciones sobre recursos

State Transfer) ha emergido como un paradigma dominante para el diseño de interfaces de programación de aplicaciones (APIs) web [18], particularmente en contextos donde la simplicidad, escalabilidad y compatibilidad con infraestructuras web existentes resultan prioritarias.

La solución desarrollada aplica estos principios de forma sistemática, garantizando que cada interacción con la API sea clara, coherente y compatible con herramientas estándar de desarrollo.

2.6.1. Flask para APIs REST

Flask[8] es un microframework web para Python creado en 2010 por Armin Ronacher [19]. A diferencia de frameworks más complejos como Django, Flask adopta una filosofía minimalista y flexible: ofrece solo lo esencial, y permite que los desarrolladores incorporen únicamente los componentes que necesiten.

En este proyecto, se utilizan solo las extensiones necesarias, lo que permite mantener una arquitectura simple y eficiente.

[8]Microframework web para Python que permite desarrollar aplicaciones web y APIs de forma sencilla y flexible.

Capítulo 3

Análisis de Plataformas Académicas de la UPM

Antes de empezar a planificar posibles soluciones y el verdadero alcance del proyecto, se ha realizado un análisis comparativo de las plataformas disponibles en la actualidad dentro de la UPM para la consulta de información académica, estudiando su funcionamiento, limitaciones, además de los datos relevantes que proporcionan, identificando las oportunidades de mejora, y estableciendo la base para el desarrollo de nuestra solución integrada.

3.1. Moodle UPM

Moodle UPM es la plataforma oficial de gestión del aprendizaje de la Universidad Politécnica de Madrid, implementada desde el curso 2016-2017. Esta plataforma está basada en el sistema de gestión de aprendizaje de código abierto Moodle, adaptado a las necesidades específicas de la universidad.

3.1.1. Funcionamiento y características

Moodle UPM funciona como un entorno virtual de aprendizaje que permite a los docentes:

- Gestionar contenidos académicos como documentos, presentaciones, videos, etc.
- Crear y administrar actividades evaluables como tareas, cuestionarios y exámenes.
- Realizar seguimiento del progreso de los estudiantes.
- Comunicarse con los estudiantes a través de foros, mensajes y anuncios.
- Gestionar calificaciones y proporcionar retroalimentación.

Para los estudiantes, Moodle UPM ofrece:

- Acceso centralizado al material disponible de todas sus asignaturas.
- Visualización de calificaciones y retroalimentación.
- Entrega de trabajos y realización de pruebas en línea.
- Participación en foros y actividades colaborativas.
- Calendario integrado con fechas importantes y eventos.

3.1.2. Limitaciones identificadas

A pesar de las numerosas funcionalidades que ofrece, Moodle UPM presenta algunas limitaciones significativas:

- **Falta de integración con otras plataformas académicas**: Moodle UPM funciona como un sistema aislado que no se integra eficientemente con otras plataformas de la universidad, lo que dificulta la obtención de una visión completa del contexto académico.
- **Información histórica limitada**: La plataforma no proporciona acceso a información histórica de asignaturas en cursos anteriores, lo que dificulta el análisis de tendencias y la comparación entre diferentes períodos académicos.
- **Ausencia de análisis estadístico**: Aunque Moodle UPM almacena grandes cantidades de datos sobre el rendimiento de los estudiantes, carece de herramientas analíticas avanzadas que permitan extraer insights significativos de estos datos.
- **Inconsistencia en la información**: La calidad y cantidad de información disponible varía significativamente entre diferentes asignaturas, dependiendo del nivel de adopción y uso que hagan los docentes de la plataforma.

3.2. GAUSS UPM

GAUSS [20] es una plataforma desarrollada por el Vicerrectorado de Servicios Tecnológicos y el Vicerrectorado de Calidad y Eficiencia de la UPM. Su objetivo principal es automatizar los procesos indicados en el Sistema de Aseguramiento Interno de la Calidad[1] (SAIC) en los centros de la UPM.

3.2.1. Funcionalidades principales

GAUSS destaca por las siguientes funcionalidades:

Repositorio de documentación académica Almacena más de 4.000 Guías de Aprendizaje anuales, 3.400 Informes de Asignatura, 450 Informes de Semestre y más de 100 Informes de Titulación.

Gestión de calidad Facilita la implementación de procesos de calidad en las titulaciones de la UPM.

Acceso a información histórica Permite consultar datos académicos de cursos anteriores, aunque de forma limitada y poco estructurada.

Generación de informes Proporciona herramientas para la elaboración de informes de seguimiento y evaluación de titulaciones.

API pública Expone algunos datos académicos a través de una API REST, lo que facilita la integración con otros sistemas.

3.2.2. Estructura de la API GAUSS

Un aspecto particularmente relevante para nuestro proyecto es la API pública de GAUSS, que proporciona acceso a información académica en formato JSON. Tras analizar esta API, identificamos la siguiente estructura para las guías de aprendizaje:

[1] https://www.etsiinf.upm.es/?id=politicacalidad

1. **Punto de acceso principal**: `https://www.upm.es/comun_gauss/publico/api/`
2. **Parámetros de consulta**:
   - `{academic_year}`: Año académico en formato "YYYY-YY"(ej. "2022-23")
   - `{semester}`: Semestre, con valores posibles "1S.º "2S"
   - `{plan_code}_{subject_code}`: Código del plan de estudios y código de la asignatura
3. **Ejemplo de URL completa**:
   `https://www.upm.es/comun_gauss/publico/api/2022-23/2S/10II_105000005.json`

La respuesta de la API proporciona información detallada sobre la asignatura, incluyendo:

Datos básicos Nombre, código, créditos ECTS, carácter de la asignatura (básica, obligatoria u optativa), etc.

Profesorado Lista de profesores con información de contacto y tutorías

Recursos didácticos Bibliografía y recursos web recomendados

Actividades de evaluación Detalles sobre el sistema de evaluación, incluyendo tipos de pruebas, pesos y requisitos

Criterios de evaluación Descripción textual de los criterios aplicados

Durante nuestro análisis, también identificamos algunas limitaciones en esta plataforma y su API:

- **Datos incompletos**: No todas las asignaturas tienen información completa
- **Datos obsoletos:** Algunas asignaturas llevan con datos antiguos o con informacion obsoleta
- **Interfaz poco intuitiva**: La navegación y búsqueda de información específica puede resultar complicada debido a una estructura de interfaz que no prioriza la experiencia de usuario.
- **Bugs en la recuperación de datos**: Algunas asignaturas tienen endpoints que no dan resultados por fallos en el servicio.
- **Inconsistencia en formatos**: Algunos campos pueden variar en estructura entre diferentes asignaturas
- **Ausencia de datos de rendimiento**: La API pública no proporciona estadísticas de rendimiento académico
- **Limitaciones en consultas**: No permite búsquedas avanzadas ni filtrado de resultados

Estas limitaciones se enmarcan en el contexto más amplio de la evolución de las herramientas digitales de la UPM, tal como se describe en las directrices institucionales para las Guías de Aprendizaje [21], donde se establece la necesidad de integrar múltiples fuentes de información académica para proporcionar una experiencia educativa más coherente y eficiente.

3.3. Informes de Semestre

Los Informes de Semestre son documentos en formato PDF que recogen información detallada sobre el rendimiento académico de las asignaturas impartidas durante un semestre específico en

cada titulación. Estos informes son generados por las Jefaturas de Estudios y constituyen una valiosa fuente de información para nuestro proyecto.

3.3.1. Estructura y contenido

Los Informes de Semestre presentan una estructura estandarizada que incluye:

1. **Información general del semestre**: Titulación, año académico y semestre.
2. **Datos de matriculación**: Número total de estudiantes matriculados por asignatura, estudiantes de primera matrícula y estudiantes a tiempo parcial.
3. **Tasas de rendimiento académico**: Incluyen:
   - **Tasa de rendimiento**: Porcentaje de estudiantes que superan la asignatura respecto al total de matriculados.
   - **Tasa de éxito**: Porcentaje de estudiantes que superan la asignatura respecto a los presentados.
   - **Tasa de absentismo**: Porcentaje de estudiantes no presentados respecto al total de matriculados.
4. **Datos históricos**: Evolución de las tasas de rendimiento, éxito y absentismo en los últimos años (generalmente 4 cursos académicos).

3.3.2. Valor y limitaciones

Los Informes de Semestre constituyen una fuente valiosa de información cuantitativa sobre el rendimiento académico, pero presentan algunas limitaciones importantes:

- **Formato no estructurado**: Al estar en formato PDF, la extracción automatizada de datos resulta compleja y propensa a errores.
- **Acceso limitado**: No existe un repositorio centralizado de fácil acceso para estos informes.
- **Ausencia de análisis cualitativo**: Los informes se centran en datos cuantitativos sin proporcionar análisis de las causas o factores que influyen en los resultados.
- **Falta de integración**: La información no se integra con otros sistemas como Moodle.

3.4. Oportunidades de mejora identificadas

A partir del análisis realizado, hemos identificado las siguientes oportunidades de mejora que justifican el desarrollo de nuestra plataforma:

- **Integración de fuentes de datos**: Existe la necesidad de un sistema que integre la información dispersa en diferentes plataformas (Moodle [Guías de aprendizaje], GAUSS, Informes de Semestre) para proporcionar una visión unificada del contexto académico.
- **Análisis estadístico avanzado**: Se requieren herramientas que permitan analizar tendencias, identificar correlaciones y extraer insights significativos a partir de los datos académicos disponibles.
- **Visualización interactiva**: Es necesario desarrollar interfaces visuales que faciliten la comprensión de los datos y apoyen la toma de decisiones tanto para estudiantes como para docentes.

- **Accesibilidad de la información**: Se debe mejorar el acceso a información histórica y comparativa sobre asignaturas y titulaciones.
- **Análisis de correlaciones**: Existe la oportunidad de estudiar las relaciones entre cambios en el profesorado, métodos de evaluación y resultados académicos para identificar factores de éxito.

Capítulo 4

Extracción y Estructuración de Información Académica

Después de haber analizado las fuentes de información disponibles, se empieza a planear la extracción de los datos necesarios para su compilación y análisis, así como los procesos y técnicas implementados para su estructuración.

4.1. Fuentes de información identificadas

A partir del análisis previo de las plataformas académicas de la UPM, se identifican tres fuentes principales de información relevante para el proyecto:

- **Informes de Semestre en formato PDF**: Contienen datos detallados sobre tasas de rendimiento, éxito y absentismo de las asignaturas, así como información histórica de cursos anteriores.
- **API pública de GAUSS**: Proporciona información estructurada de las guías de aprendizaje de las asignaturas, incluyendo datos del profesorado, metodologías de enseñanza y sistemas de evaluación.
- **Guías de aprendizaje en formato PDF**: Ofrecen información detallada sobre los contenidos, objetivos, metodologías y sistemas de evaluación de cada asignatura.

Cada una de estas fuentes presenta características particulares para la extracción de datos, que se detallan en las siguientes secciones.

4.2. Extracción de datos de Informes de Semestre

Los Informes de Semestre, al estar en formato PDF, presentan un desafío y una desventaja para la extracción automatizada de datos. Para abordar esta problemática, se desarrolla un módulo específico denominado `academic_data_extractor.py`.

4.2.1. Proceso de extracción de texto

El primer paso consistió en extraer el texto completo de los documentos PDF. Para ello, se utilizó la biblioteca PyMuPDF (importada como `fitz`), que permite un acceso eficiente al contenido

textual de documentos PDF. El proceso se implementó en la función `extract_text_from_pdf` del módulo `main.py`:

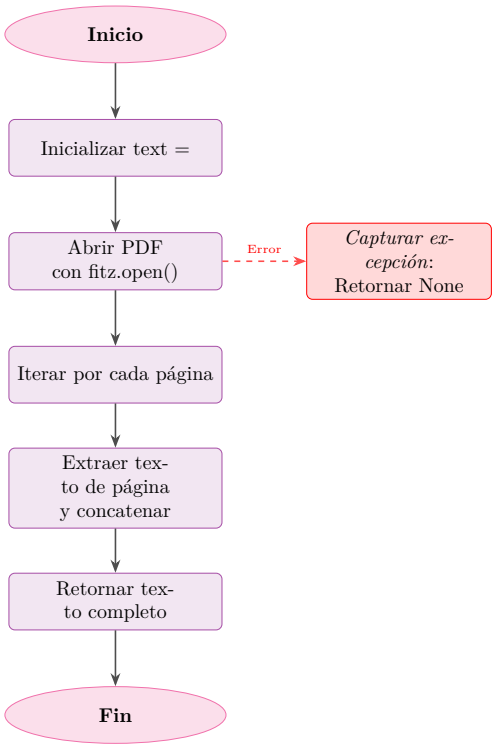

Figura 4.1: Diagrama de flujo de la función *extract_text_from_pdf*

4.2.2. Análisis y estructuración del texto extraído

Una vez obtenido el texto completo, se desarrolló la clase `AcademicDataExtractor` para analizar y estructurar la información. Esta clase implementa varios métodos específicos para extraer diferentes secciones de información del informe:

extract_course_info() Extrae información básica del curso académico, como año y semestre.

extract_subjects_basic_info() Identifica y extrae información básica de las asignaturas (código, nombre, créditos).

extract_student_profile() Obtiene datos sobre el perfil de estudiantes matriculados.

extract_performance_rates() Extrae las tasas de rendimiento, éxito y absentismo actuales.

extract_historical_rates() Recopila datos históricos de las tasas mencionadas.

El proceso de extracción se basa fundamentalmente en el uso de expresiones regulares (`regex`) para identificar patrones específicos en el texto. A continuación, se muestra un diagrama de flujo de la función que extrae información sobre tasas de rendimiento:

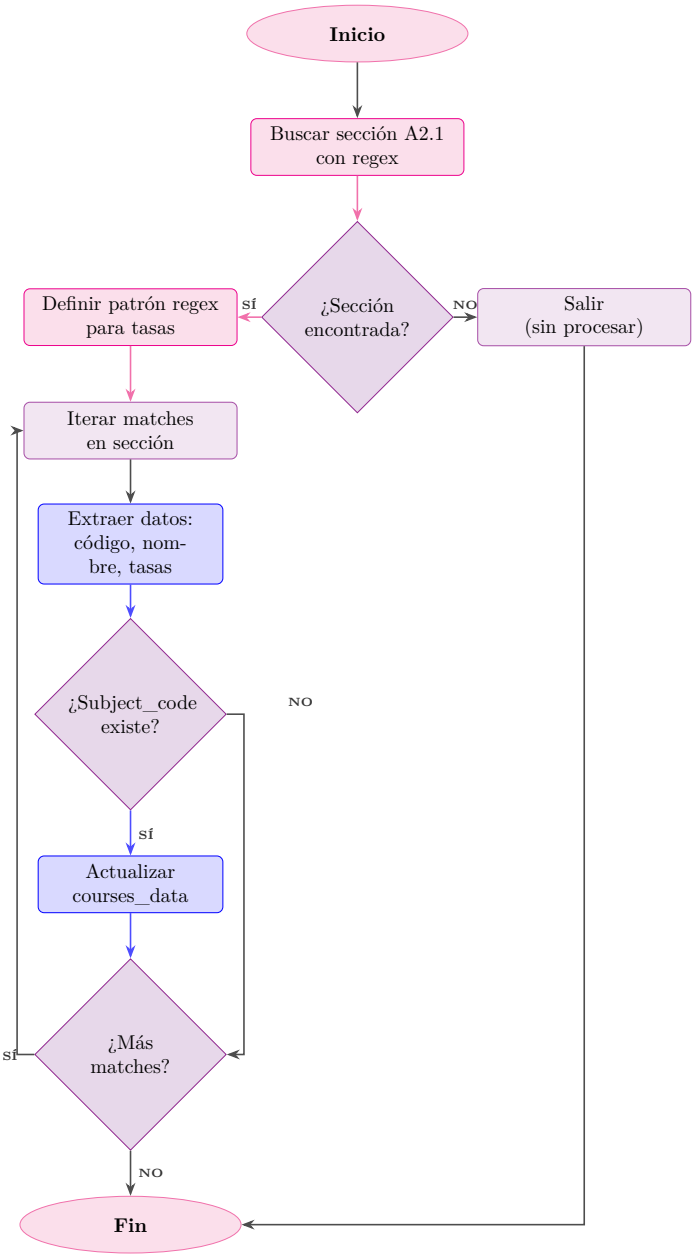

Figura 4.2: Diagrama de flujo de la función *extract_performance_rates*

4.2.3. Validación y depuración de datos extraídos

Debido a la naturaleza no estructurada de los documentos PDF y las posibles variaciones en su formato, se implementó un proceso de validación y depuración de los datos extraídos para garantizar su calidad. Este proceso incluye:

- Verificación de la consistencia de los códigos de asignatura.
- Normalización de nombres de asignaturas para evitar duplicados.
- Validación de valores numéricos para asegurar que estén dentro de rangos razonables.
- Manejo de casos especiales y excepciones.

Los datos validados se almacenan en una estructura de diccionario anidado que facilita su posterior procesamiento y almacenamiento en la base de datos.

4.3. Consumo de la API de GAUSS

Para complementar la información extraída de los Informes de Semestre, se desarrolló el módulo específico (`academic_api_extractor.py`) para consumir la API pública de GAUSS y obtener datos adicionales sobre las asignaturas.

4.3.1. Diseño de la clase AcademicApiExtractor

La clase `AcademicApiExtractor` encapsula la funcionalidad necesaria para interactuar con la API de GAUSS. Sus principales características incluyen:

- Gestión de solicitudes HTTP a la API.
- Implementación de caché para optimizar el rendimiento y reducir el número de solicitudes.
- Análisis y estructuración de las respuestas JSON.
- Funciones para consultar datos de asignaturas específicas o múltiples asignaturas.

El siguiente diagrama muestra la función principal para obtener datos de una asignatura específica:

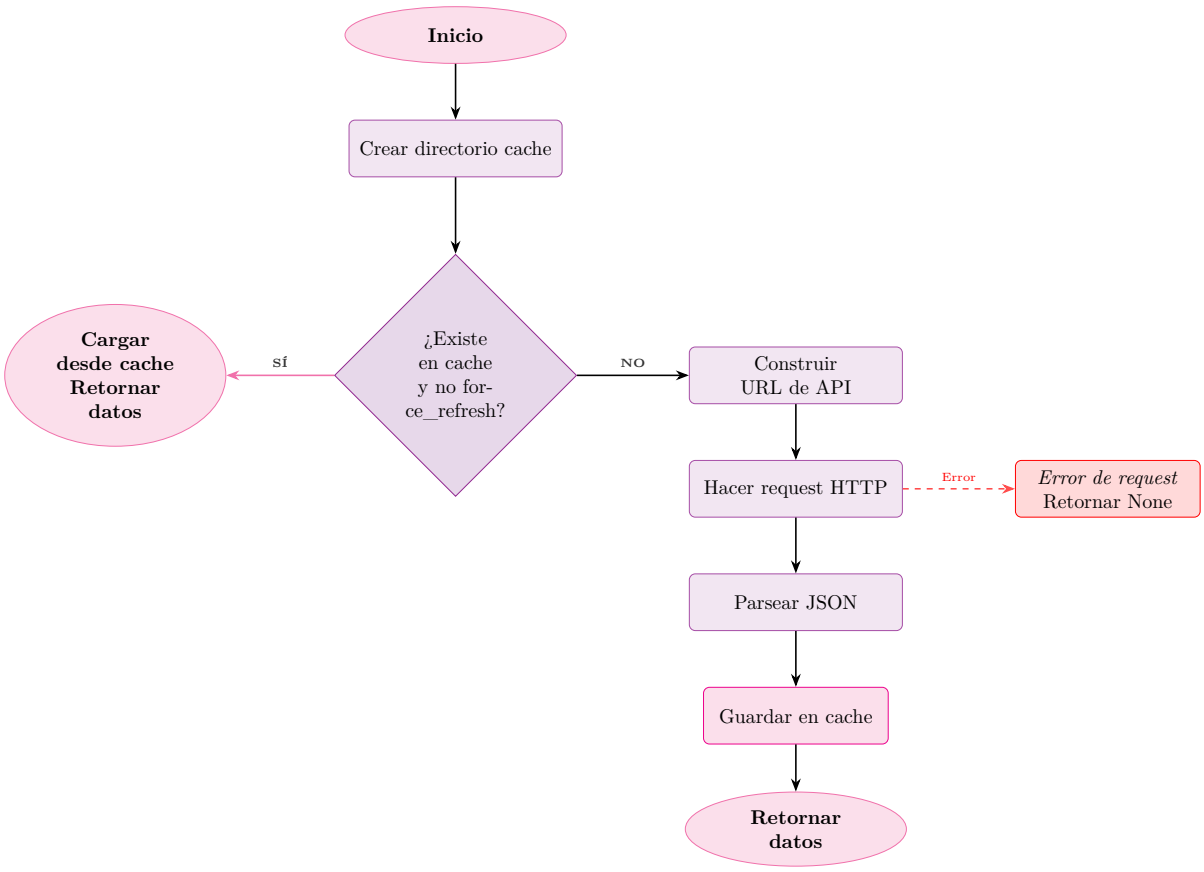

Figura 4.3: Diagrama de flujo de la función *get_subject_api_data*

4.3.2. Análisis de cambios en profesorado y evaluación

Uno de los aspectos más innovadores del proyecto es el análisis de los cambios en el profesorado y los métodos de evaluación entre diferentes años académicos, y su posible correlación con los resultados de rendimiento. Para ello, se implementaron métodos específicos en la clase `AcademicApiExtractor`:

analyze_faculty_changes() Compara los profesores asignados a una asignatura entre diferentes años, identificando adiciones y eliminaciones, y calculando el porcentaje de cambio.

analyze_evaluation_changes() Identifica cambios en los métodos de evaluación utilizados en diferentes períodos académicos.

A continuación, se muestra un fragmento del método de análisis de cambios en el profesorado:

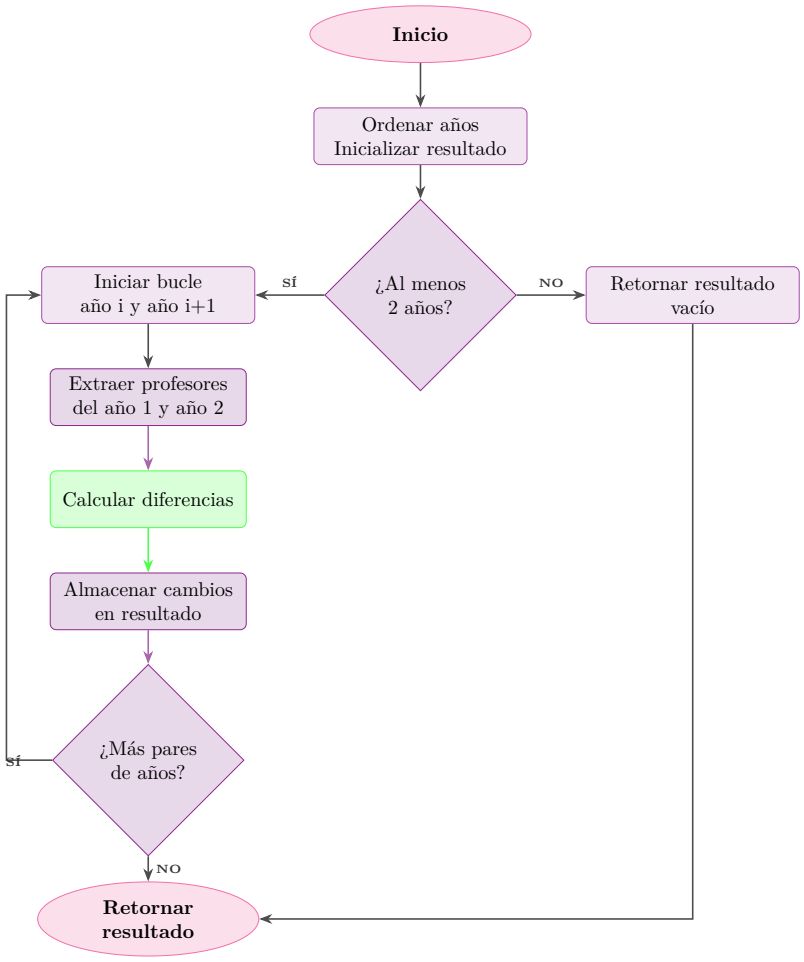

Figura 4.4: Diagrama de flujo de la función *analyze_faculty_changes*

4.4. Diseño de Ficheros de Información Académica

Para facilitar el posterior análisis y la integración con la base de datos, se diseñó una estructura de ficheros JSON que almacena de forma organizada toda la información extraída. Esta estructura garantiza la consistencia y facilita la serialización y deserialización de los datos entre diferentes componentes del sistema.

4.4.1. Estructura de ficheros JSON para datos extraídos de Informes de Semestre

Los datos extraídos de los Informes de Semestre se estructuran en un formato JSON con la siguiente organización:

Listing 4.1: JSON de informe de semestre de segundo semestre del segundo curso

```
{
  "course_info": {
    "academic_year": "2023/24",
    "semester": "Segundo",
    "plan_code": "10II",
    "plan_title": "Grado en Ingenieria Informatica",
    "report_date": "2024-02-15"
  },
  "subjects": {
    "105000005": {
      "name": "Calculo",
      "credits": 6,
      "total_enrolled": 455,
      "first_time": 248,
      "partial_dedication": 0,
      "performance_rate": 31.94,
      "success_rate": 36.52,
      "absenteeism_rate": 12.56,
      "historical": {
        "rendimiento": {
          "2020-21": 31.32,
          "2021-22": 35.12,
          "2022-23": 32.49,
          "2023-24": 31.94
        },
        "exito": {
          "2020-21": 41.80,
          "2021-22": 42.30,
          "2022-23": 37.90,
          "2023-24": 36.52
        },
        "absentismo": {
          "2020-21": 25.06,
          "2021-22": 16.98,
          "2022-23": 14.29,
          "2023-24": 12.56
        }
      }
    },
    // Otras asignaturas...
  }
}
```

4.4.2. Estructura de ficheros JSON para análisis del API

Los resultados del análisis de datos obtenidos de la API de GAUSS, particularmente en lo referente a cambios en profesorado y métodos de evaluación, se estructuran en un formato JSON más complejo que facilita su posterior correlación con los datos de rendimiento:

Listing 4.2: JSON de analísis de la API

```
{
  "105000005": {
    "subject_name": "Calculo",
    "faculty_analysis": {
      "years_compared": [
        ["2020-21", "2021-22"],
```

```
7           ["2021-22", "2022-23"],
8           ["2022-23", "2023-24"]
9         ],
10        "faculty_changes": {
11          "2020-21_to_2021-22": {
12            "added": ["Maria Francisca Martinez Serrano"],
13            "removed": ["Juan Perez Lopez"],
14            "total_added": 1,
15            "total_removed": 1,
16            "percent_changed": 16.67
17          },
18          // Otros periodos...
19        }
20      },
21      "evaluation_analysis": {
22        "years_compared": [
23          ["2020-21", "2021-22"],
24          ["2021-22", "2022-23"],
25          ["2022-23", "2023-24"]
26        ],
27        "evaluation_changes": {
28          "2020-21_to_2021-22": {
29            "added": ["EX: Tecnica del tipo Examen Parcial"],
30            "removed": [],
31            "changed": true
32          },
33          // Otros periodos...
34        }
35      }
36    },
37    // Otras asignaturas...
38  }
```

4.4.3. Estructura de ficheros JSON para correlaciones e insights

Finalmente, se diseñó una estructura JSON para almacenar las correlaciones identificadas entre cambios en profesorado/evaluación y rendimiento académico, así como los insights generados a partir de este análisis:

Listing 4.3: JSON *correlaciones.json*

```
1  {
2    "global_insights": {
3      "analysis_id": 1,
4      "analysis_date": "2024-03-15",
5      "faculty_impact": {
6        "impact_type": "positive",
7        "with_changes": 2.5,
8        "without_changes": 0.7,
9        "difference": 1.8,
10       "insight": "Los cambios en el profesorado se asocian a mejoras en el rendimiento",
11       "recommendations": [
12         "Continuar la política de renovacion estrategica del profesorado en asignaturas
               con bajos rendimientos",
13         "Identificar y replicar las practicas de los nuevos profesores en asignaturas de
               alto rendimiento"
14       ]
15     },
16     "evaluation_impact": {
17       "impact_type": "neutral",
18       "with_changes": 1.2,
19       "without_changes": 0.9,
20       "difference": 0.3,
21       "insight": "Los cambios en el metodo de evaluación no tienen un impacto claro en
             los resultados",
22       "recommendations": [
23         "Estandarizar los métodos de evaluación más efectivos y mantener consistencia",
```

```
        "Comunicar con claridad a los estudiantes los criterios y métodos de evaluación"
      ]
    },
    "recommendations": {
      "faculty": [
        "Proseguir la renovación estratégica del profesorado en las asignaturas de bajo rendimiento",
        "Identificar y reproducir prácticas del nuevo profesorado en asignaturas de alto rendimiento"
      ],
      "evaluation": [
        "Normalizar los métodos de evaluación más eficaces y mantener la coherencia",
        "Introducir cambios graduales en la evaluación con procesos de transición claros"
      ],
      "general": [
        "Establecer un sistema de seguimiento continuo de calidad",
        "Realizar encuestas a los estudiantes para identificar factores cualitativos"
      ]
    }
  },
  "subject_insights": {
    "105000005": {
      "subject_name": "Cálculo",
      "average_performance_change": -0.55,
      "trend_direction": "declining",
      "faculty_impact": {
        "periods_with_changes": 2,
        "performance_with_changes": -0.85,
        "impact_direction": "negative"
      },
      "evaluation_impact": {
        "periods_with_changes": 1,
        "performance_with_changes": 0.2,
        "impact_direction": "neutral"
      },
      "periods": [
        {
          "year1": "2021-22",
          "year2": "2022-23",
          "performance_change": -2.63,
          "faculty_changed": true,
          "faculty_details": {
            "percent_changed": 33.33,
            "added": 2,
            "removed": 1
          },
          "evaluation_changed": false,
          "insights": [
            {
              "type": "faculty_negative",
              "text": "El descenso del rendimiento podría estar relacionado con los cambios significativos en el profesorado"
            }
          ]
        },
        // Más períodos...
      ]
    },
    // Más asignaturas...
  }
}
```

Capítulo 5

Diseño y Gestión de Bases de Datos

El proceso de extracción de los datos académicos y su posterior análisis tiene como resultado una gran cantidad de información relevante sobre las asignaturas, la cual es necesaria almacenar en una base de datos para su gestión y administración.

La primera decisión crítica en el diseño de la base de datos fue la elección del sistema de gestión. Tras evaluar diferentes alternativas, se optó por utilizar SQLite por las siguientes razones:

Simplicidad SQLite no requiere un servidor de base de datos separado, lo que simplifica la instalación, configuración y mantenimiento.

Portabilidad La base de datos completa se almacena en un único archivo, facilitando su distribución y copia de seguridad.

Rendimiento Para el volumen de datos y consultas previsto en nuestro sistema, su rendimiento es adecuado.

Integración con Python SQLite tiene integración con Python a través de la biblioteca estándar `sqlite3`.

Transacciones ACID SQLite garantiza que las transacciones cumplan con las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), asegurando la integridad de los datos.

5.1. Diseño de la Base de Datos

La estructura de la base de datos fue diseñada para representar de manera eficiente las diferentes entidades y relaciones identificadas en nuestro dominio académico, facilitando tanto el almacenamiento como la consulta y análisis de los datos.

5.1.1. Modelo Entidad-Relación

El siguiente diagrama representa el modelo entidad-relación diseñado para la base de datos:

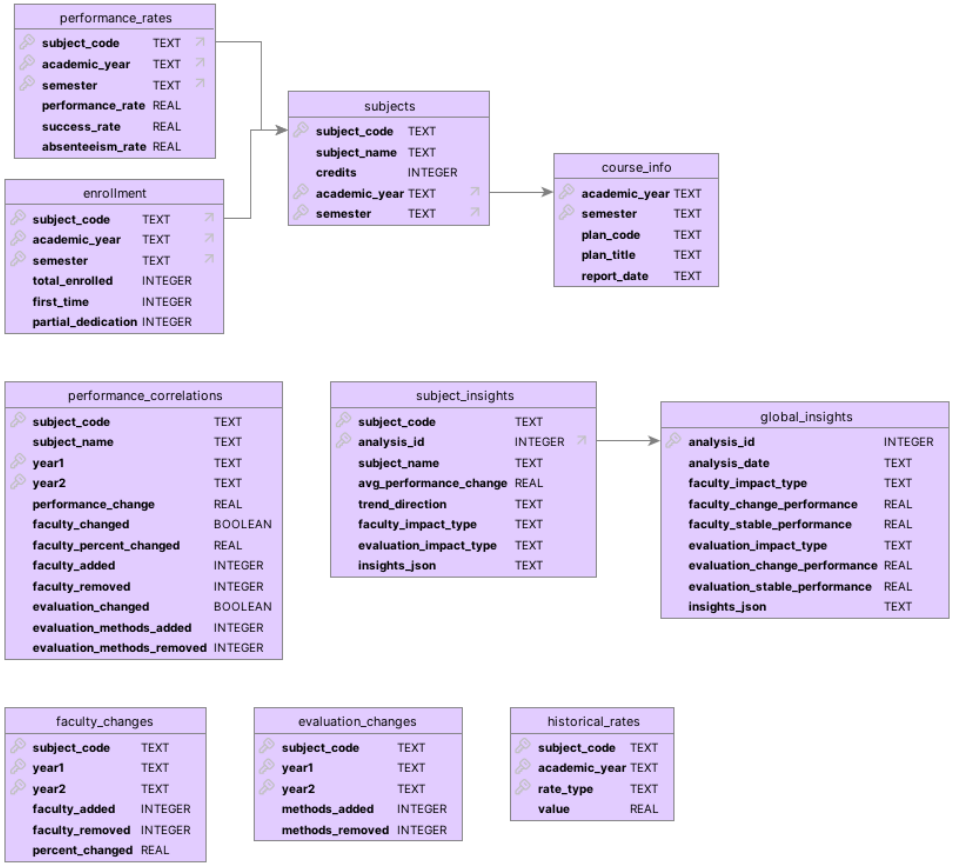

Figura 5.1: Modelo Entidad-Relación de la base de datos

5.1.2. Estructura de tablas

La base de datos está compuesta por las siguientes tablas principales:

- **course_info**: Almacena información general sobre los cursos académicos analizados.
- **subjects**: Contiene información básica sobre las asignaturas.
- **enrollment**: Registra datos de matriculación en las asignaturas.
- **performance_rates**: Almacena las tasas de rendimiento, éxito y absentismo actuales.
- **historical_rates**: Contiene datos históricos de las diferentes tasas académicas.
- **faculty_changes**: Registra los cambios en el profesorado entre diferentes años académicos.
- **evaluation_changes**: Almacena los cambios en los métodos de evaluación.
- **performance_correlations**: Relaciona los cambios en profesorado y evaluación con las variaciones en rendimiento.
- **global_insights**: Almacena análisis globales y recomendaciones generadas.
- **subject_insights**: Contiene análisis específicos para cada asignatura.

A continuación, se detalla la estructura específica de cada tabla, incluyendo sus campos, tipos de datos y relaciones:

5.1.2.1. Tabla course_info

Esta tabla almacena información general sobre los cursos académicos:

Campo	Tipo	Descripción
academic_year	TEXT	Año académico (ej. "2023/24") - PK
semester	TEXT	Semestre (ej. "Primero", "Segundo") - PK
plan_code	TEXT	Código del plan de estudios
plan_title	TEXT	Título del plan de estudios
report_date	TEXT	Fecha del informe

Cuadro 5.1: Estructura de la tabla course_info

5.1.2.2. Tabla subjects

Contiene información básica de las asignaturas:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
subject_name	TEXT	Nombre de la asignatura
credits	INTEGER	Número de créditos
academic_year	TEXT	Año académico - PK, FK a course_info
semester	TEXT	Semestre - PK, FK a course_info

Cuadro 5.2: Estructura de la tabla subjects

5.1.2.3. Tabla enrollment

Registra datos de matriculación:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK, FK a subjects
academic_year	TEXT	Año académico - PK, FK a subjects
semester	TEXT	Semestre - PK, FK a subjects
total_enrolled	INTEGER	Total de estudiantes matriculados
first_time	INTEGER	Estudiantes de primera matrícula
partial_dedication	INTEGER	Estudiantes con dedicación parcial

Cuadro 5.3: Estructura de la tabla enrollment

5.1.2.4. Tabla performance_rates

Almacena las tasas académicas actuales:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK, FK a subjects
academic_year	TEXT	Año académico - PK, FK a subjects
semester	TEXT	Semestre - PK, FK a subjects
performance_rate	REAL	Tasa de rendimiento (%)
success_rate	REAL	Tasa de éxito (%)
absenteeism_rate	REAL	Tasa de absentismo (%)

Cuadro 5.4: Estructura de la tabla performance_rates

5.1.2.5. Tabla historical_rates

Contiene datos históricos de tasas académicas:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
academic_year	TEXT	Año académico - PK
rate_type	TEXT	Tipo de tasa (ej. rendimiento", "éxito", .ªbsentismo") - PK
value	REAL	Valor de la tasa (%)

Cuadro 5.5: Estructura de la tabla historical_rates

5.1.2.6. Tabla faculty_changes

Registra los cambios en el profesorado:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
year1	TEXT	Primer año del período comparado - PK
year2	TEXT	Segundo año del período comparado - PK
faculty_added	INTEGER	Número de profesores añadidos
faculty_removed	INTEGER	Número de profesores eliminados
percent_changed	REAL	Porcentaje de cambio en el profesorado

Cuadro 5.6: Estructura de la tabla faculty_changes

5.1.2.7. Tabla evaluation_changes

Almacena los cambios en los métodos de evaluación:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
year1	TEXT	Primer año del período comparado - PK
year2	TEXT	Segundo año del período comparado - PK
methods_added	INTEGER	Número de métodos de evaluación añadidos
methods_removed	INTEGER	Número de métodos de evaluación eliminados

Cuadro 5.7: Estructura de la tabla evaluation_changes

5.1.2.8. Tabla performance_correlations

Relaciona cambios en profesorado/evaluación con variaciones en rendimiento:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
subject_name	TEXT	Nombre de la asignatura
year1	TEXT	Primer año del período comparado - PK
year2	TEXT	Segundo año del período comparado - PK
performance_change	REAL	Cambio en la tasa de rendimiento
faculty_changed	BOOLEAN	Indica si hubo cambios en el profesorado
faculty_percent_changed	REAL	Porcentaje de cambio en el profesorado
faculty_added	INTEGER	Número de profesores añadidos
faculty_removed	INTEGER	Número de profesores eliminados
evaluation_changed	BOOLEAN	Indica si hubo cambios en métodos de evaluación
evaluation_methods_added	INTEGER	Número de métodos de evaluación añadidos
evaluation_methods_removed	INTEGER	Número de métodos de evaluación eliminados

Cuadro 5.8: Estructura de la tabla performance_correlations

5.1.2.9. Tabla global_insights

Almacena análisis globales y recomendaciones:

Campo	Tipo	Descripción
analysis_id	INTEGER	Identificador del análisis - PK, Autoincrement
analysis_date	TEXT	Fecha del análisis
faculty_impact_type	TEXT	Tipo de impacto de cambios en profesorado
faculty_change_performance	REAL	Rendimiento medio con cambios en profesorado
faculty_stable_performance	REAL	Rendimiento medio sin cambios en profesorado
evaluation_impact_type	TEXT	Tipo de impacto de cambios en evaluación
evaluation_change_performance	REAL	Rendimiento medio con cambios en evaluación
evaluation_stable_performance	REAL	Rendimiento medio sin cambios en evaluación
insights_json	TEXT	Análisis detallado y recomendaciones en formato JSON

Cuadro 5.9: Estructura de la tabla global_insights

5.1.2.10. Tabla subject_insights

Contiene análisis específicos para cada asignatura:

Campo	Tipo	Descripción
subject_code	TEXT	Código de la asignatura - PK
analysis_id	INTEGER	Identificador del análisis relacionado - PK, FK a global_insights
subject_name	TEXT	Nombre de la asignatura
avg_performance_change	REAL	Cambio medio en rendimiento
trend_direction	TEXT	Dirección de la tendencia (improving, declining, stable)
faculty_impact_type	TEXT	Tipo de impacto de cambios en profesorado
evaluation_impact_type	TEXT	Tipo de impacto de cambios en evaluación
insights_json	TEXT	Análisis detallado específico para la asignatura en formato JSON

Cuadro 5.10: Estructura de la tabla subject_insights

5.2. Implementación

La implementación de la base de datos se realizó mediante la clase `AcademicDatabase` en el módulo `academic_database.py`. Esta clase encapsula toda la funcionalidad necesaria para la creación, gestión y consulta de la base de datos.

5.2.1. Creación e inicialización de la base de datos

La clase `AcademicDatabase` se encarga de establecer la conexión con la base de datos SQLite y crear las tablas necesarias si no existen. El siguiente diagrama de flujo muestra el método de inicialización y configuración de tablas:

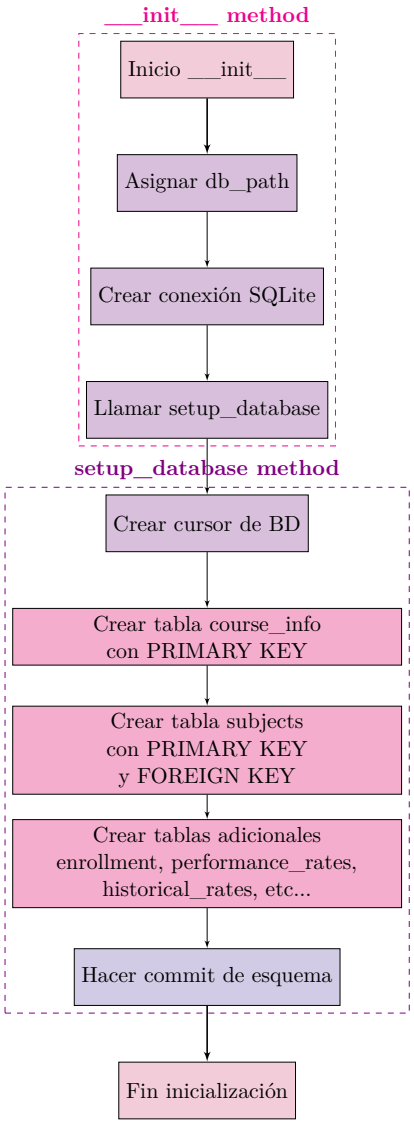

Figura 5.2: Diagramas de flujo *___init___* y *setup_database*

5.2.2. Almacenamiento de datos

La clase `AcademicDatabase` proporciona métodos para almacenar diferentes tipos de datos en la base de datos. El método principal, `store_data`, se encarga de almacenar la información

básica extraída de los informes académicos:

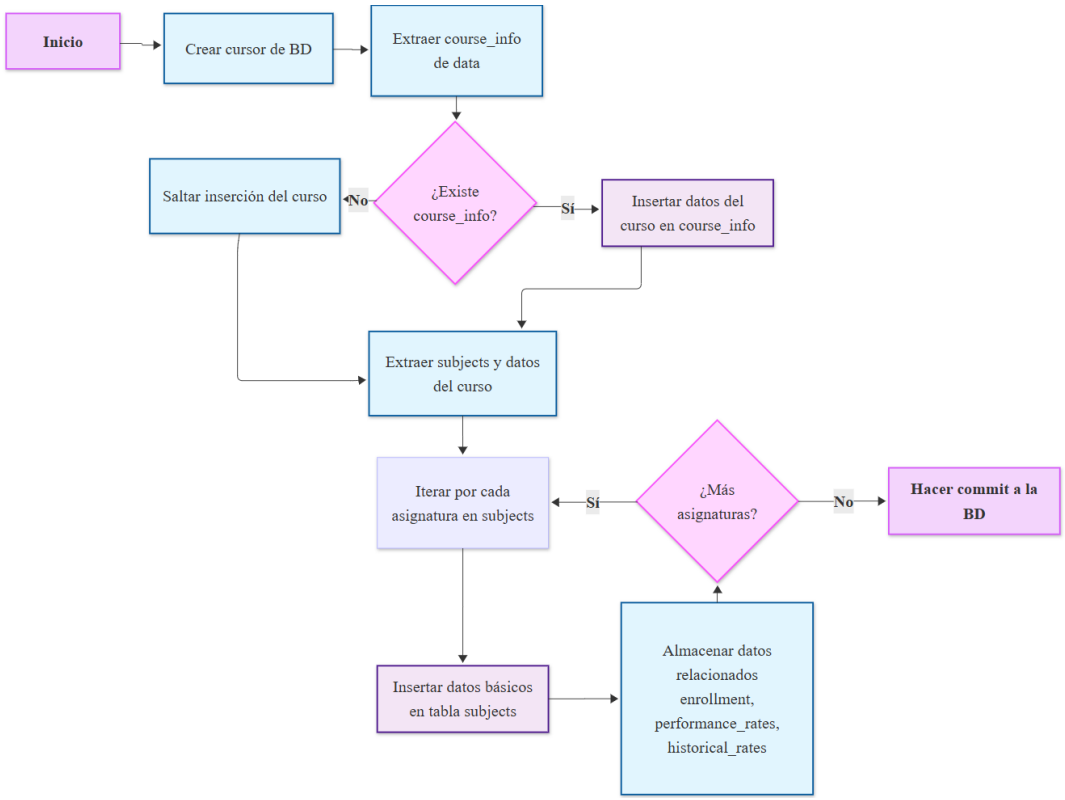

Figura 5.3: Diagrama de flujo de *store_data*

Además, se implementaron métodos específicos para almacenar los resultados del análisis de la API y las correlaciones identificadas:

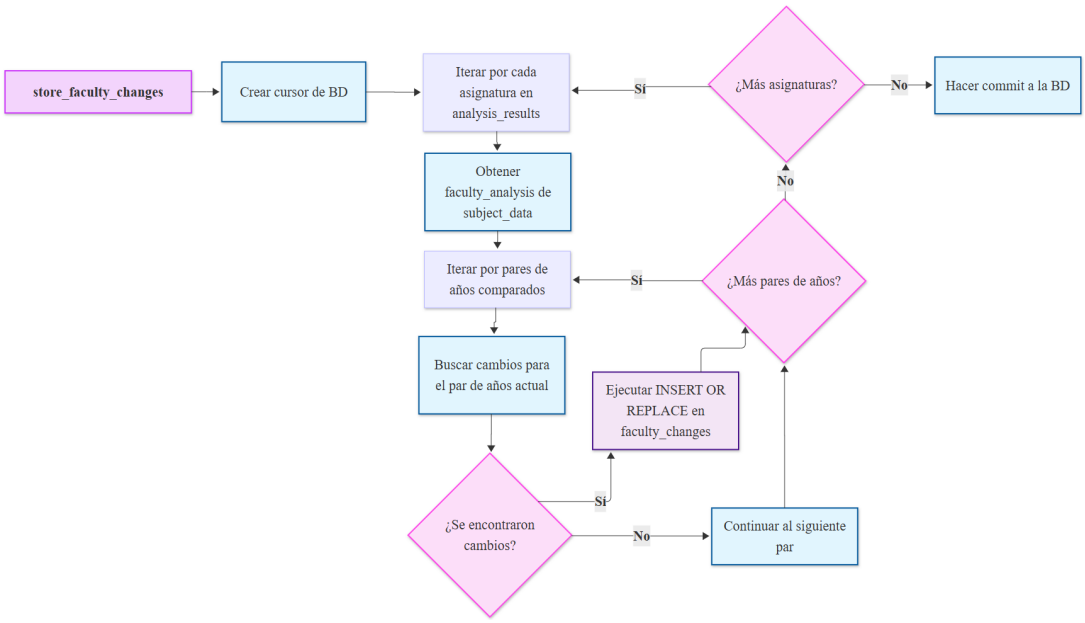

Figura 5.4: Diagrama de flujo de la función *store_faculty_changes*

5.2.3. Consulta y recuperación de datos

Para facilitar el acceso a los datos almacenados, la clase `AcademicDatabase` proporciona una serie de métodos de consulta específicos. Estos métodos devuelven los resultados en forma de DataFrames[1] de la librería pandas, lo que facilita su posterior análisis y visualización:

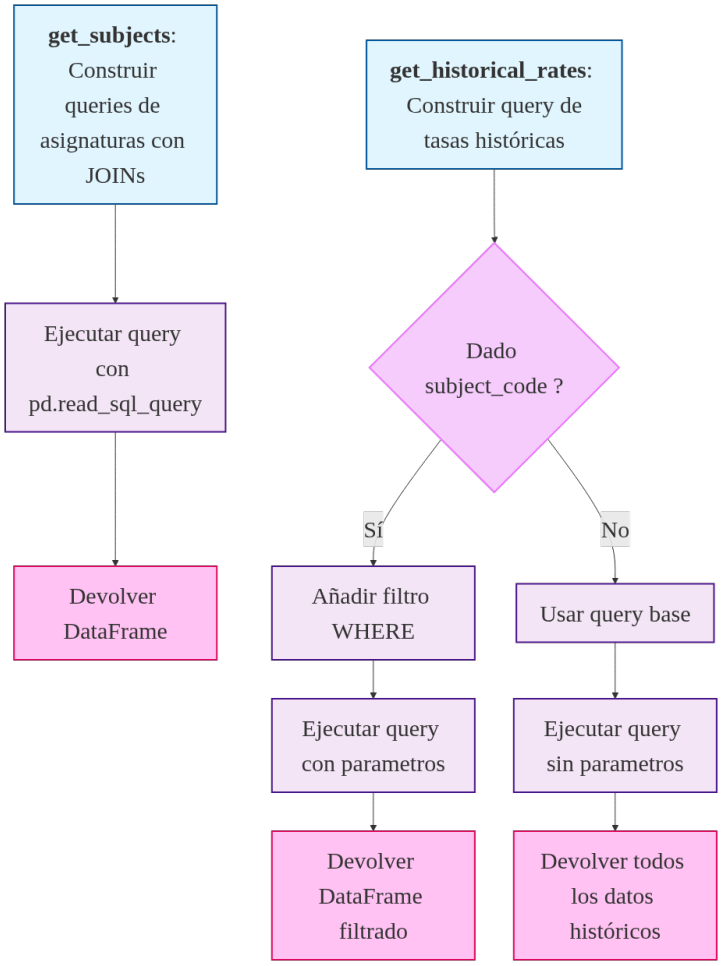

Figura 5.5: Diagrama de flujo de la función *get_subjects*

5.2.4. Exportación de datos

Para facilitar la integración con otras herramientas y sistemas, la clase `AcademicDatabase` proporciona métodos para exportar los datos almacenados en formatos comunes como CSV y JSON (figura 5.6):

[1]Estructura de datos bidimensional y etiquetada, similar a una tabla de una base de datos relacional o una hoja de cálculo de Excel.

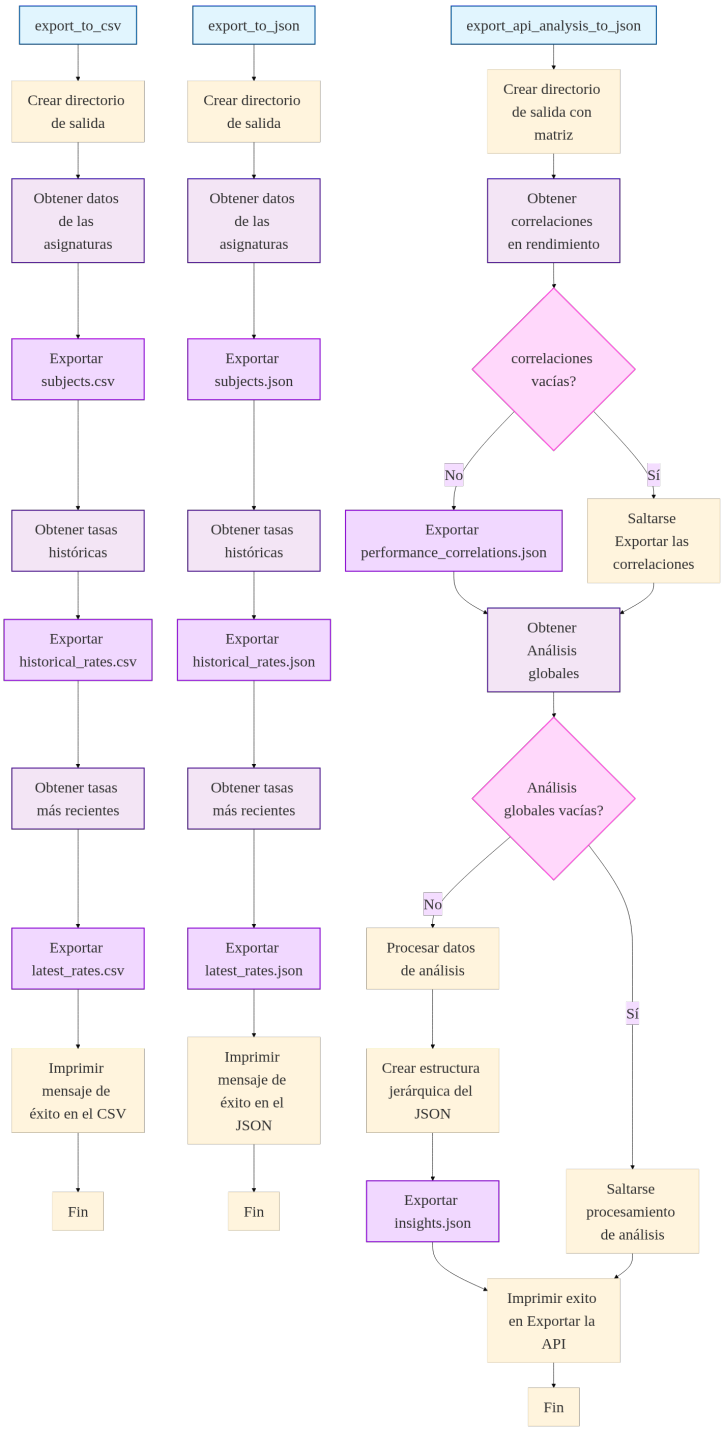

Figura 5.6: Diagrama de flujo de los métodos de exportación

5.3. Gestión de la Base de Datos en el Sistema Completo

La base de datos diseñada e implementada juega un papel central en el sistema solución, sirviendo como punto de integración entre los diferentes componentes:

- Recibe y almacena los datos extraídos de los Informes de Semestre a través del módulo `academic_data_extractor.py`.

- Almacena los resultados del análisis de la API de GAUSS obtenidos a través del módulo `academic_api_extractor.py`.
- Proporciona los datos necesarios para el análisis estadístico realizado por el módulo `academic_data_analyzer.py`.
- Alimenta las gráficas generadas por el módulo `academic_visualizations.py`.
- Suministra los datos a la API REST desarrollada para la comunicación con el frontend (`rest-api.py`).

Esta centralización de los datos facilita la coherencia y consistencia de la información en todo el sistema, permitiendo que los diferentes componentes trabajen con los mismos datos y evitando discrepancias y redundancias.

5.4. Middleware: API REST

Para facilitar la comunicación entre el backend (Python) y el frontend (Next.js), se desarrolló una API REST que expone los datos almacenados en la base de datos a través de endpoints HTTP. Esta API fue implementada utilizando la librería Flask en el módulo `rest-api.py`.

La API REST proporciona los siguientes endpoints principales:

/api/v1/subjects Devuelve información sobre todas las asignaturas o filtradas por año académico y semestre.

/api/v1/subjects/<subject_code> Proporciona información detallada sobre una asignatura específica.

/api/v1/subjects/<subject_code>/historical Devuelve datos históricos de tasas académicas para una asignatura específica.

/api/v1/performance/summary Proporciona un resumen estadístico de las tasas de rendimiento.

/api/v1/faculty/changes Devuelve información sobre cambios en el profesorado.

/api/v1/evaluation/changes Proporciona datos sobre cambios en los métodos de evaluación.

/api/v1/correlations Devuelve información sobre correlaciones entre cambios y rendimiento.

/api/v1/insights/global Proporciona insights globales generados por el sistema.

/api/v1/insights/subjects Devuelve insights específicos para cada asignatura.

El siguiente fragmento muestra la implementación del endpoint para obtener información sobre todas las asignaturas:

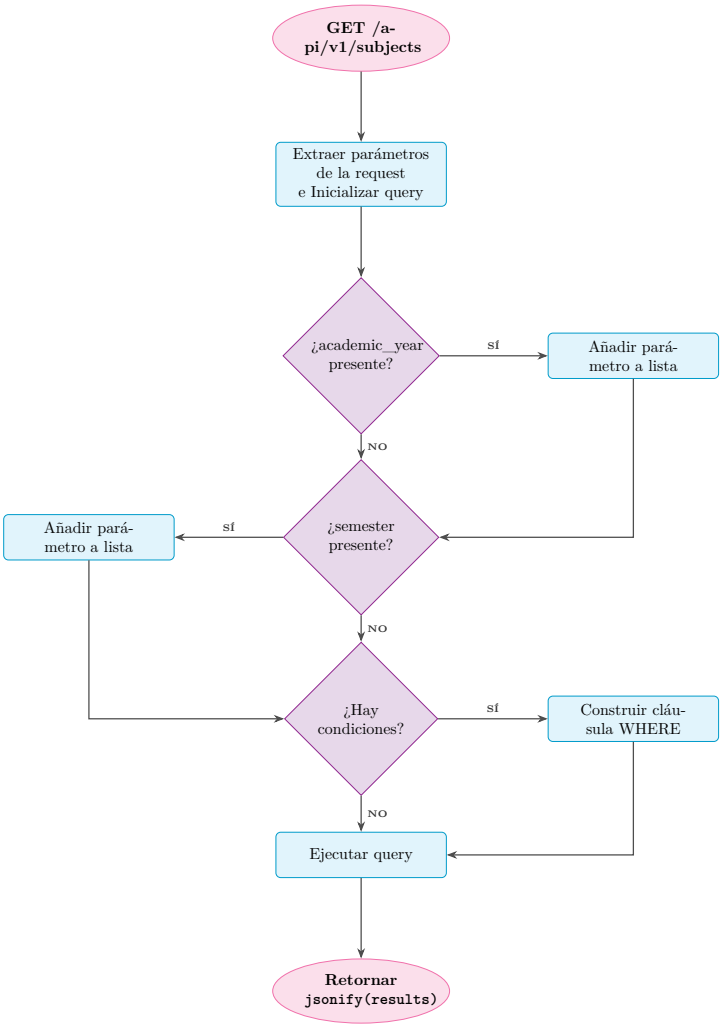

Figura 5.7: Diagrama de flujo del API Endpoint `/subjects`

La API REST desarrollada como middleware proporciona una capa de abstracción que facilita la comunicación entre el backend y el frontend, permitiendo una separación clara de objetivos y responsabilidades, y facilitando el desarrollo paralelo de ambos componentes.

En conjunto, la base de datos y la API asociada constituyen el núcleo del sistema, permitiendo la integración coherente de datos de las fuentes, y proporcionando la base necesaria para el análisis estadístico, y la generación de gráficas y otras visualizaciones que se describen en los capítulos siguientes.

Capítulo 6

Desarrollo

El presente capítulo expone detalladamente el proceso de desarrollo del sistema DASOS, abordando sus fundamentos arquitectónicos, las tecnologías implementadas y los métodos analíticos aplicados. Este desarrollo se ha estructurado siguiendo principios de ingeniería de software, con especial atención a la modularidad, extensibilidad y robustez, requeridas para el procesamiento y análisis de datos académicos.

6.1. Arquitectura del sistema

La arquitectura de DASOS se fundamenta en un diseño de capas que permite la separación de responsabilidades, facilitando tanto el mantenimiento como la evolución futura del sistema. Este enfoque arquitectónico responde a los requisitos específicos de un sistema de análisis académico, donde la integración de diversas fuentes de datos y la implementación de métodos analíticos avanzados demandan una estructura flexible y escalable.

6.1.1. Precedentes arquitectónicos en la UPM

El diseño arquitectónico de DASOS se ha beneficiado de la experiencia previa en el desarrollo de sistemas similares dentro de la UPM. Proyectos como la renovación de la plataforma GAUSS [22] han demostrado la efectividad de arquitecturas basadas en separación clara entre frontend y backend, utilizando APIs REST para la comunicación entre componentes.

Asimismo, iniciativas más recientes como el desarrollo de sistemas de gestión para asociaciones estudiantiles [23] han validado el uso de tecnologías modernas como React y frameworks backend especializados en el contexto universitario, proporcionando un marco de referencia para las decisiones arquitectónicas de DASOS.

6.1.2. Visión general arquitectónica

El sistema se estructura en tres capas principales que interactúan mediante interfaces, siguiendo un patrón arquitectónico que combina elementos del modelo tradicional de tres capas con aspectos específicos para sistemas analíticos:

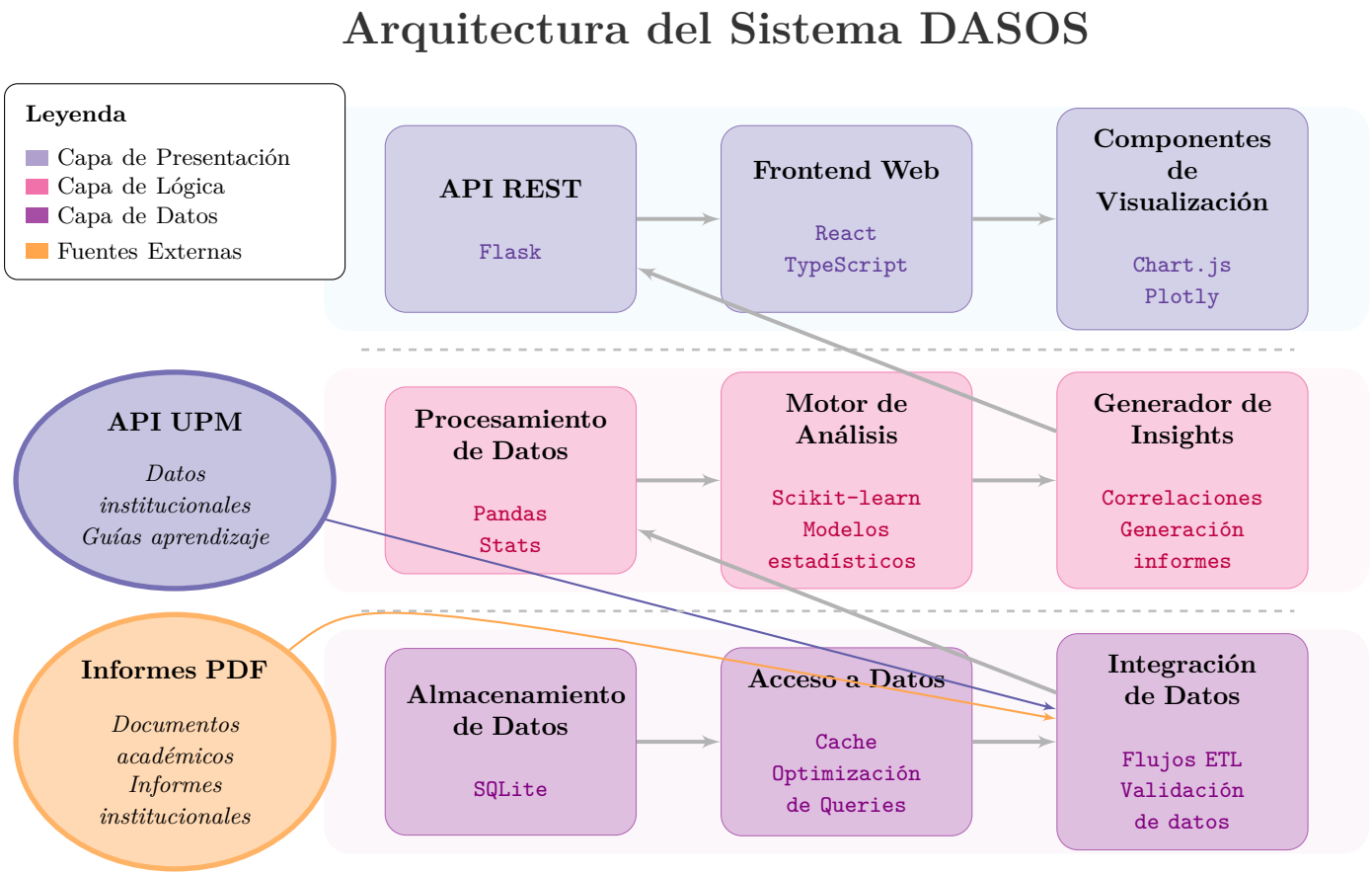

Figura 6.1: Arquitectura general del sistema DASOS

- **Capa de Datos:** Gestiona la adquisición, almacenamiento e integración de datos procedentes de múltiples fuentes. Se compone de tres componentes principales:
  - *Almacenamiento de Datos:* Implementa la persistencia de información mediante una base de datos SQLite, optimizada para consultas analíticas sobre datos académicos.
  - *Acceso a Datos:* Proporciona una capa de abstracción para operaciones sobre la base de datos, encapsulando la complejidad de consultas SQL y transacciones.
  - *Integración de Datos:* Coordina la extracción desde fuentes heterogéneas (informes PDF, API UPM) y la transformación a estructuras consistentes con el modelo de datos interno.
- **Capa de Lógica:** Implementa el núcleo analítico del sistema, procesando datos estructurados para generar información significativa. Contiene:
  - *Procesamiento de Datos:* Ejecuta transformaciones, normalizaciones y agregaciones necesarias para preparar los datos para análisis.
  - *Motor de Análisis:* Implementa los algoritmos estadísticos y métodos de correlación, aplicando técnicas específicas según el tipo de análisis requerido.
  - *Generador de Insights:* Interpreta resultados analíticos para identificar patrones relevantes y formular recomendaciones accionables.

- **Capa de Presentación:** Facilita el acceso a información y visualizaciones para usuarios finales. Incluye:
  - *API REST:* Expone endpoints para consumo programático de datos e insights mediante una interfaz RESTful.
  - *Frontend Web:* Implementa la interfaz de usuario, desarrollada con Next.js y React, que permite interacciones intuitivas con el sistema.
  - *Componentes de Visualización:* Gestiona la representación gráfica de datos e insights mediante bibliotecas especializadas.

6.1.3. Flujo de datos y procesamiento

El flujo de datos en el proyecto sigue un patrón secuencial que transforma progresivamente la información desde su estado bruto inicial hasta visualizaciones e insights accionables. Este proceso puede conceptualizarse como una serie de transformaciones consecutivas:

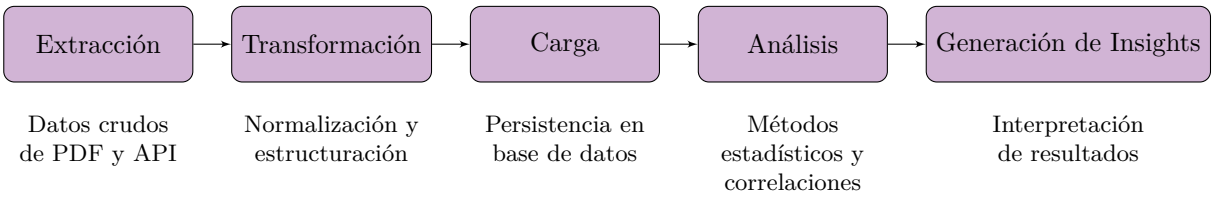

Figura 6.2: Flujo de datos en el sistema DASOS

1. **Extracción:** Los datos son obtenidos de sus fuentes originales mediante procesos especializados para cada formato:
   - *Informes PDF:* Extracción mediante PyMuPDF y procesamiento con expresiones regulares para identificar secciones relevantes y datos específicos.
   - *API UPM:* Solicitudes HTTP parametrizadas que obtienen datos JSON [1] sobre asignaturas, profesorado y métodos de evaluación.
2. **Transformación:** Los datos extraídos son sometidos a procesos de limpieza, normalización y estandarización:
   - Conversión de tipos de datos (texto a numéricos, fechas estandarizadas)
   - Normalización de nombres y códigos para facilitar correlaciones entre fuentes
   - Estructuración en formatos consistentes con el modelo de datos relacional
3. **Carga:** Los datos transformados se almacenan en la base de datos SQLite:
   - Inserción en tablas específicas según su naturaleza (asignaturas, tasas, matriculación)
   - Establecimiento de relaciones mediante claves primarias y foráneas
   - Actualización incremental para datos que evolucionan temporalmente
4. **Análisis:** Los datos almacenados son procesados mediante técnicas analíticas específicas:
   - Análisis descriptivo de distribuciones y tendencias

[1] Formato ligero de intercambio de datos basado en la sintaxis de objetos de JavaScript.

- Métodos de correlación para identificar relaciones entre variables
- Pruebas estadísticas para validar significancia de hallazgos

5. **Generación de Insights:** Los resultados analíticos son interpretados para extraer conocimiento accionable:
   - Identificación de patrones significativos en los datos
   - Formulación de hipótesis explicativas para variaciones observadas
   - Generación de recomendaciones basadas en evidencia empírica

Este flujo de procesamiento implementa conceptualmente un patrón ETL (Extract-Transform-Load) extendido con capacidades analíticas avanzadas. La principal diferencia respecto a implementaciones ETL tradicionales es la integración estrecha entre procesos de carga y análisis, que permite generar insights prácticamente en tiempo real a medida que nuevos datos son incorporados al sistema.

6.2. Tecnologías utilizadas

La implementación de DASOS ha requerido la selección de tecnologías que proporcionan el equilibrio óptimo entre capacidades técnicas, facilidad de desarrollo y mantenibilidad a largo plazo.

6.2.1. Backend

El desarrollo del backend se ha realizado primariamente en Python, complementado con tecnologías específicas para diferentes aspectos del sistema:

6.2.1.1. Python

Python constituye el lenguaje fundamental para el desarrollo del backend, seleccionado por múltiples razones que lo hacen particularmente adecuado para sistemas de análisis de datos:

- **Ecosistema científico robusto:** Bibliotecas como NumPy, Pandas, SciPy y Scikit-learn proporcionan implementaciones maduras y optimizadas de numerosos algoritmos estadísticos y de análisis de datos.
- **Legibilidad y mantenibilidad:** La sintaxis clara y expresiva de Python facilita la implementación de algoritmos complejos de forma comprensible, aspecto crítico para un sistema con componentes analíticos sofisticados.
- **Versatilidad:** Python excele tanto en el procesamiento de datos como en el desarrollo web, permitiendo una integración fluida entre componentes analíticos y servicios web.

6.2.1.2. Bibliotecas de análisis de datos

El sistema implementa diversas bibliotecas especializadas para diferentes aspectos del procesamiento y análisis de datos:

- **Pandas:** Proporciona estructuras de datos optimizadas (DataFrame, Series) y funcionalidades para manipulación, limpieza y transformación de datos tabulares [24]. En DASOS, Pandas se utiliza extensivamente para:
  - Transformación de datos jerárquicos a formato tabular

- Operaciones de filtrado, agrupación y agregación
- Análisis de series temporales para tendencias históricas
- Exportación a formatos como CSV [2] y JSON

- **NumPy:** Implementa estructuras para computación numérica eficiente, fundamentales para operaciones matriciales y vectoriales. Sus aplicaciones en DASOS incluyen:
  - Cálculos estadísticos sobre conjuntos de datos extensos
  - Operaciones vectorizadas para transformación de datos
  - Generación de matrices de correlación entre variables
- **SciPy:** Complementa NumPy con implementaciones de algoritmos científicos avanzados. En DASOS se utiliza particularmente el módulo `scipy.stats` para:
  - Pruebas estadísticas de hipótesis (t-test)
  - Cálculo de coeficientes de correlación (Pearson, Spearman)
  - Análisis de distribuciones y detección de outliers
- **Matplotlib y Seaborn:** Proporcionan capacidades de visualización de datos fundamentales para análisis exploratorio y generación de gráficos [25]. Se utilizan para:
  - Visualización de distribuciones y tendencias
  - Creación de gráficos comparativos entre asignaturas o períodos
  - Generación de mapas de calor para correlaciones
  - Visualizaciones personalizadas para informes automáticos

6.2.1.3. Procesamiento de documentos y APIs

El sistema implementa tecnologías específicas para la extracción e integración de datos de diferentes fuentes:

- **PyMuPDF (fitz):** Biblioteca para procesamiento de documentos PDF que proporciona acceso programático al contenido textual y estructura. Seleccionada por:
  - Rendimiento superior en extracción de texto
  - Capacidad para preservar ciertos aspectos de la estructura espacial
  - API intuitiva compatible con patrones Python modernos
- **Requests:** Biblioteca para interacción HTTP que simplifica la comunicación con APIs externas. Utilizada en DASOS para:
  - Comunicación con la API académica de la UPM
  - Implementación de estrategias robustas de reintento y manejo de errores
  - Gestión de cacheo para optimizar rendimiento

[2] Formato de archivo de texto plano que utiliza comas para separar valores y saltos de línea para delimitar registros.

6.2.1.4. Almacenamiento de datos

Para la persistencia de datos, el sistema implementa SQLite como motor de base de datos, seleccionado por características que lo hacen particularmente adecuado para este proyecto:

- **Ausencia de configuración servidor:** Su naturaleza integrada elimina la necesidad de configurar y mantener servicios de base de datos separados.
- **Portabilidad:** El almacenamiento en un único archivo facilita despliegues, copias de seguridad y transferencias entre entornos.
- **Rendimiento:** Para volúmenes de datos académicos típicos de una institución, proporciona rendimiento adecuado sin sobrecarga innecesaria.

6.2.1.5. API REST

La exposición de funcionalidades mediante la API se implementa utilizando Flask, un microframework web para Python seleccionado por:

- **Minimalismo:** Proporciona exactamente los componentes necesarios sin sobrecarga innecesaria.
- **Flexibilidad:** Permite implementar patrones de diseño adaptados a las necesidades específicas del proyecto.
- **Integración natural:** Su implementación en Python permite interacción fluida con componentes analíticos del sistema.

6.2.2. Frontend

La interfaz de usuario de DASOS se ha desarrollado como una aplicación web moderna, utilizando tecnologías que proporcionan una experiencia interactiva y responsive:

6.2.2.1. Next.js y React

El núcleo del frontend se implementa mediante Next.js, un framework basado en React que proporciona capacidades avanzadas de renderizado y optimización:

- **Renderizado híbrido:** Combina técnicas de Server-Side Rendering (SSR) y Static Site Generation (SSG) para optimizar tanto rendimiento como SEO.
- **Enrutamiento simplificado:** Su sistema de enrutamiento basado en sistema de archivos facilita la implementación de navegación compleja.
- **API Routes:** Permite implementar endpoints de API dentro del mismo proyecto, simplificando la arquitectura para funcionalidades auxiliares.
- **Optimización automática:** Implementa por defecto técnicas avanzadas como code splitting y lazy loading que mejoran la experiencia de usuario.

React, como biblioteca subyacente, proporciona un modelo de componentes que facilita la construcción de interfaces interactivas mediante:

- **Componentes reutilizables:** Elementos de interfaz encapsulados que pueden combinarse para crear vistas complejas.

- **Rendering declarativo:** Descripción del estado deseado de la interfaz, con actualización automática cuando cambian los datos.
- **Virtual DOM:** Optimización de actualizaciones mediante comparación eficiente de estados para minimizar manipulaciones DOM[3].
- **Reactividad:** Propagación automática de cambios en datos a través de la interfaz de usuario.

6.2.2.2. Bibliotecas de visualización

Las visualizaciones interactivas en el frontend se implementan mediante Chart.js, una biblioteca JavaScript que ofrece:

- **API intuitiva:** Permite crear visualizaciones complejas con código declarativo relativamente simple.
- **Responsividad:** Adapta automáticamente las visualizaciones a diferentes tamaños de contenedor y dispositivos.
- **Interactividad:** Implementa funcionalidades como tooltips, zoom y animaciones de forma nativa.
- **Integración con React:** Se adapta bien al modelo de componentes mediante wrappers especializados.

6.3. Algoritmos y métodos estadísticos

El componente analítico de DASOS implementa diversos algoritmos y métodos estadísticos para procesar datos académicos y generar insights significativos. Esta sección detalla los principales enfoques metodológicos aplicados en diferentes aspectos del análisis.

6.3.1. Estadística descriptiva

El análisis básico de datos académicos comienza con métodos de estadística descriptiva que proporcionan una caracterización cuantitativa de las distribuciones y tendencias presentes:

6.3.1.1. Métricas de tendencia central

El sistema aplica sistemáticamente cálculos de medidas de tendencia central para caracterizar valores típicos en diferentes métricas académicas:

- **Media aritmética:** Utilizada como indicador primario para tasas de rendimiento, éxito y absentismo, tanto por asignatura como agregadas por período académico. Su implementación considera ponderaciones basadas en matriculación cuando se calculan valores para grupos de asignaturas:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{6.1}$$

[3]Document Object Model: es una estructura jerárquica de varios objetos que tienen una dependencia entre sí. Se trata de un estándar creado por la W3C donde se definen aquellos objetos que conforman un sitio web, cómo deben nombrarse y cuál es su relación.

donde w_i representa el número de estudiantes matriculados en la asignatura i y x_i la tasa correspondiente.

- **Mediana:** Implementada como complemento a la media para métricas con distribuciones potencialmente asimétricas o afectadas por valores extremos. Proporciona una perspectiva de tendencia central más robusta al definir el valor que divide exactamente la distribución en dos partes iguales.
- **Moda:** Calculada para identificar los valores más frecuentes en distribuciones discretas, particularmente útil para analizar calificaciones o categorías de rendimiento.

6.3.1.2. Métricas de dispersión

El análisis incluye medidas de dispersión que cuantifican la variabilidad presente en los datos:

- **Desviación estándar:** Calculada para caracterizar la dispersión típica respecto a la media en tasas de rendimiento y otras métricas continuas. Su interpretación permite identificar asignaturas o períodos con comportamientos atípicamente variables.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{6.2}$$

- **Coeficiente de variación:** Calculado para facilitar comparaciones de variabilidad entre distribuciones con diferentes escalas o medias, mediante la normalización de la desviación estándar:

$$CV = \frac{\sigma}{\bar{x}} \times 100\,\% \tag{6.3}$$

6.3.1.3. Análisis de distribuciones

El sistema implementa metodologías para caracterizar completamente las distribuciones de diferentes métricas académicas:

- **Histogramas y densidades:** Generados programáticamente para visualizar la distribución completa de métricas como tasas de rendimiento, identificando patrones como multimodalidad o asimetrías.
- **Diagrama de caja (Box Plot):** Implementado para representar visualmente la distribución, destacando mediana, cuartiles y potenciales valores atípicos.
- **Curvas de densidad kernel:** Generadas como alternativa suavizada a histogramas, particularmente útiles para muestras relativamente pequeñas donde los histogramas pueden mostrar artifactos debidos a la elección de intervalos.
- **Pruebas de normalidad:** Implementadas mediante el test de Shapiro-Wilk para evaluar formalmente si las distribuciones de tasas académicas se aproximan a distribuciones normales, lo que informa la selección de métodos estadísticos posteriores.

6.3.2. Análisis de series temporales

El análisis de evolución histórica de métricas académicas implementa técnicas específicas para series temporales, adaptadas al contexto educativo:

6.3.2.1. Detección y caracterización de tendencias

El sistema aplica métodos para identificar y cuantificar tendencias temporales en diferentes métricas académicas:

- **Regresión lineal simple:** Implementada para cuantificar tendencias temporales mediante la estimación de una relación lineal entre tiempo y métricas académicas:

$$y_t = \beta_0 + \beta_1 t + \varepsilon_t \tag{6.4}$$

  donde y_t representa la métrica académica en el tiempo t, β_0 el intercepto, β_1 la pendiente (tasa de cambio temporal) y ε_t el término de error. La significancia estadística de β_1 se evalúa para determinar si existe una tendencia temporal genuina.

- **Prueba de Mann-Kendall:** Implementada como alternativa no paramétrica para detección de tendencias, particularmente útil cuando no se pueden asumir relaciones lineales o distribuciones específicas [26]:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sgn}(x_j - x_i) \tag{6.5}$$

  donde sgn es la función signo y x_i, x_j son valores de la serie temporal en diferentes momentos. La significancia estadística de S se evalúa para determinar la presencia y dirección de tendencias monotónicas.

6.3.3. Análisis de correlación

Un componente fundamental del sistema es la identificación y cuantificación de relaciones entre diferentes variables académicas, implementada mediante diversos métodos de análisis de correlación:

6.3.3.1. Coeficientes de correlación

El sistema implementa diferentes coeficientes de correlación, seleccionados según la naturaleza de las variables analizadas:

- **Correlación de Pearson:** Implementada para cuantificar relaciones lineales entre variables continuas como tasas de rendimiento, éxito y absentismo [27]:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6.6}$$

  Este coeficiente varía entre -1 (correlación lineal negativa perfecta) y 1 (correlación lineal positiva perfecta), con valores cercanos a 0 indicando ausencia de relación lineal.

6.3.3.2. Análisis de significación estadística

El sistema implementa pruebas formales para evaluar la significación estadística de correlaciones identificadas:

- **Prueba t para correlación:** Aplicada para evaluar la significación de coeficientes de correlación de Pearson:

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{6.7}$$

donde r es el coeficiente de correlación y n el tamaño de la muestra. Este estadístico sigue una distribución t con $n-2$ grados de libertad bajo la hipótesis nula de ausencia de correlación.

- **Transformación Z de Fisher:** Implementada para construir intervalos de confianza para coeficientes de correlación y comparar correlaciones de diferentes pares de variables:

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \tag{6.8}$$

Esta transformación aproxima una distribución normal con varianza $1/(n-3)$, facilitando pruebas de hipótesis e intervalos de confianza.

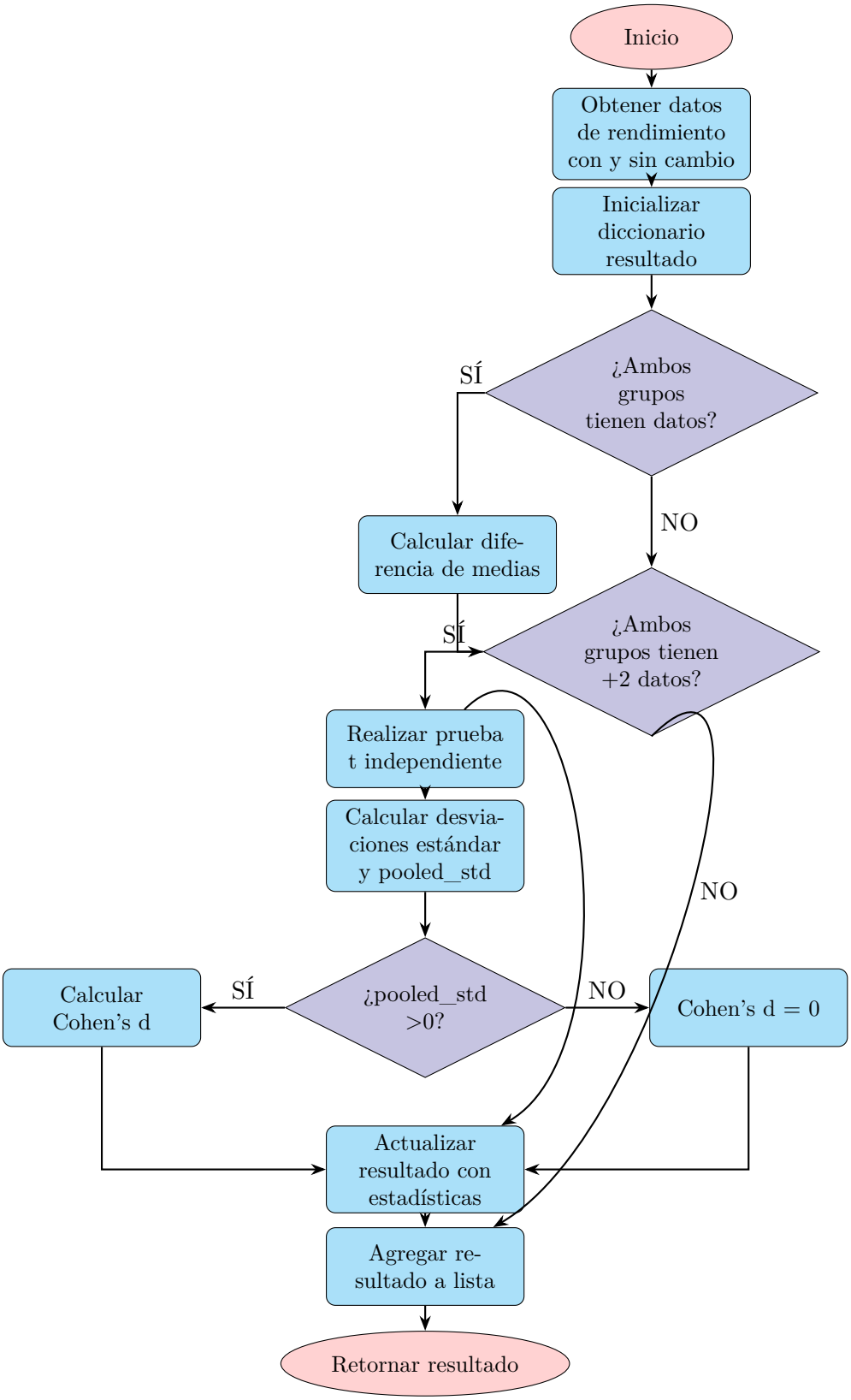

Figura 6.3: Diagrama de flujo de la función analyze_change_significance

6.3.4. Métodos específicos para análisis de cambios

El sistema implementa métodos especializados para analizar el impacto de cambios específicos (profesorado, métodos de evaluación) en métricas académicas:

6.3.4.1. Análisis de diferencias

Se aplican técnicas específicas para cuantificar y evaluar diferencias entre grupos o períodos:

- **Prueba t de Student para muestras independientes:** Implementada para comparar rendimiento entre grupos con y sin cambios específicos:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{6.9}$$

  donde $\bar{x}_i$ son las medias grupales, s_p es la desviación estándar combinada y n_i los tamaños de muestra respectivos.

- **Prueba t de Student para muestras pareadas:** Aplicada para comparar rendimiento antes y después de intervenciones específicas:

$$t = \frac{\bar{d}}{s_d / \sqrt{n}} \tag{6.10}$$

  donde $\bar{d}$ es la media de las diferencias pareadas, s_d su desviación estándar y n el número de pares.

6.3.5. Algoritmos para generación de insights

El componente de generación de insights implementa algoritmos especializados que transforman resultados analíticos en recomendaciones accionables:

6.3.5.1. Detección de patrones significativos

El sistema implementa métodos para identificar automáticamente patrones potencialmente relevantes en datos académicos:

- **Análisis de reglas de asociación:** Aplicado para descubrir relaciones no obvias entre diferentes atributos académicos:

$$\begin{aligned} \text{Support}(X \Rightarrow Y) &= \frac{|X \cup Y|}{N} \\ \text{Confidence}(X \Rightarrow Y) &= \frac{|X \cup Y|}{|X|} \\ \text{Lift}(X \Rightarrow Y) &= \frac{\text{Confidence}(X \Rightarrow Y)}{\text{Support}(Y)} \end{aligned} \tag{6.11}$$

  donde $X \Rightarrow Y$ representa una regla de asociación, $|X|$ la frecuencia de ocurrencia del conjunto X, y N el número total de transacciones.

- **Detección de anomalías:** Implementada para identificar asignaturas o períodos con comportamientos significativamente desviados de patrones típicos:

$$\text{Z-score} = \frac{x - \mu}{\sigma} \tag{6.12}$$

donde valores absolutos superiores a un umbral (típicamente 2.5 o 3) son considerados potenciales anomalías.

6.3.5.2. Generación de recomendaciones

El sistema implementa algoritmos específicos para formular recomendaciones basadas en evidencia empírica:

- **Ranking de factores por impacto:** Implementado para priorizar factores según su influencia estimada en métricas de rendimiento:

$$\text{Impact Score} = \beta_i \cdot \sigma_i \tag{6.13}$$

donde β_i es el coeficiente estimado del factor en modelos de regresión y σ_i su desviación estándar, proporcionando una medida de impacto potencial.

Esta implementación metodológica proporciona un fundamento matemático riguroso para los análisis realizados por DASOS, asegurando que las conclusiones y recomendaciones generadas se fundamenten en principios estadísticos sólidos adaptados al contexto académico específico.

6.4. Interfaz de usuario

La interfaz de usuario de DASOS ha sido diseñada para proporcionar acceso intuitivo a análisis académicos complejos, implementando principios modernos de diseño de interacción y visualización de información. Esta sección detalla los aspectos fundamentales de la interfaz y su implementación.

6.4.1. Principios de diseño

El desarrollo de la interfaz se ha fundamentado en principios específicos que guían decisiones de diseño e implementación:

- **Claridad informativa:** Priorización de la comprensibilidad de datos e insights complejos, minimizando ruido visual y destacando patrones significativos.
- **Jerarquía visual:** Organización cuidadosa de elementos visuales para guiar atención hacia información más relevante según contexto de uso.
- **Consistencia:** Implementación de patrones de interacción y elementos visuales uniformes a través de diferentes secciones de la aplicación.
- **Accesibilidad:** Consideración de diversos perfiles de usuario, con especial atención a contraste de colores, tamaños de texto y navegabilidad mediante teclado.
- **Retroalimentación clara:** Indicaciones explícitas sobre acciones en progreso, completadas o erróneas para orientar al usuario.

- **Adaptabilidad:** Diseño responsive que se adapta apropiadamente a diferentes dispositivos y tamaños de pantalla.

6.4.2. Componentes principales

La interfaz implementa diversos componentes especializados, cada uno optimizado para funcionalidades específicas:

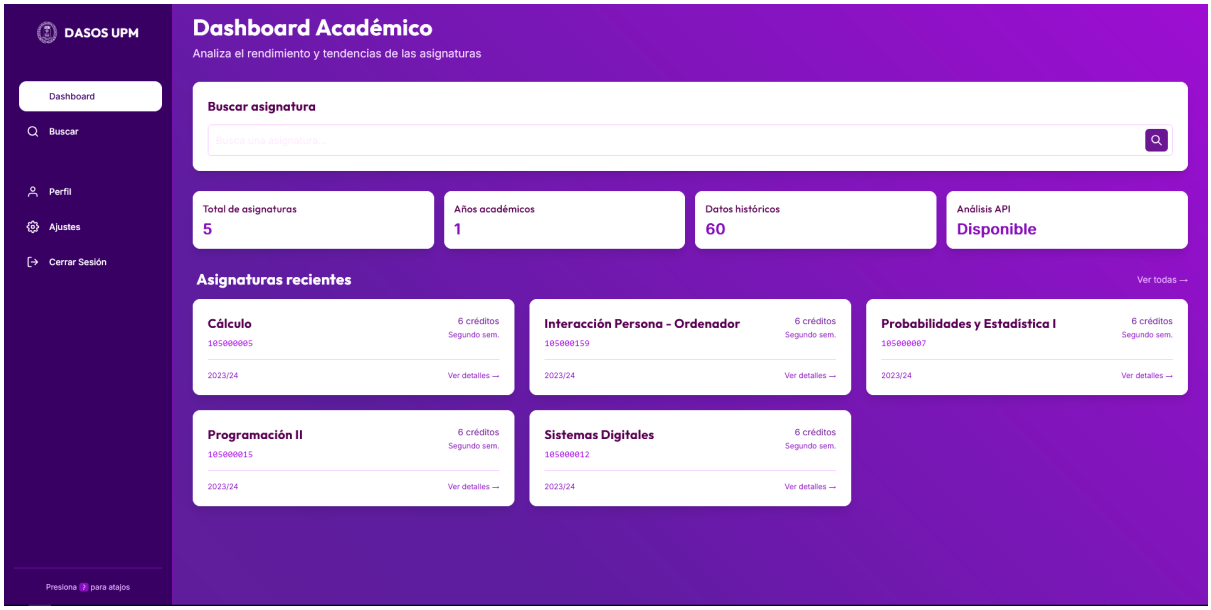

Figura 6.4: Pantalla principal del dashboard en DASOS

6.4.2.1. Dashboard

El panel principal proporciona una visión general del rendimiento académico institucional:

- **Resumen de métricas clave:** Visualizaciones concisas de indicadores fundamentales como rendimiento medio, asignaturas con mayor/menor rendimiento y tendencias temporales.
- **Navegación rápida:** Accesos directos a funcionalidades frecuentemente utilizadas y asignaturas relevantes.
- **Visualizaciones destacadas:** Gráficos seleccionados que resaltan patrones o hallazgos particularmente significativos en el período actual.
- **Alertas y notificaciones:** Indicaciones sobre situaciones que podrían requerir atención, como cambios significativos en rendimiento o próximos períodos de evaluación.

6.4.2.2. Visualización de asignaturas

La sección de asignaturas proporciona análisis detallados para entidades académicas específicas:

- **Tarjetas de resumen:** Visualizaciones condensadas de información básica y métricas fundamentales para cada asignatura.

- **Tendencias históricas:** Gráficos lineales que muestran la evolución temporal de tasas de rendimiento, éxito y absentismo.

- **Análisis de correlaciones:** Visualizaciones de relaciones entre cambios en profesorado, métodos de evaluación y variaciones en rendimiento.

- **Comparativas:** Herramientas para contrastar métricas entre diferentes asignaturas o períodos académicos.

- **Información detallada:** Acceso a datos específicos sobre profesorado, métodos de evaluación y recursos didácticos.

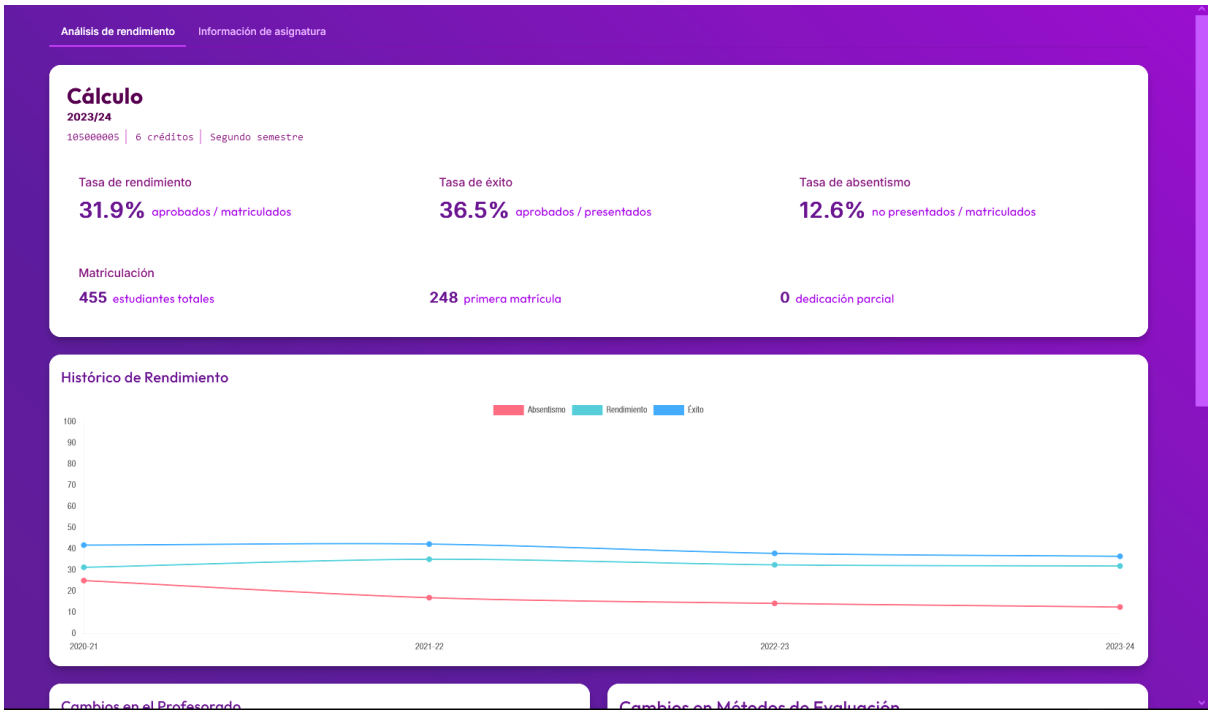

Figura 6.5: Detalle de análisis de asignatura en DASOS

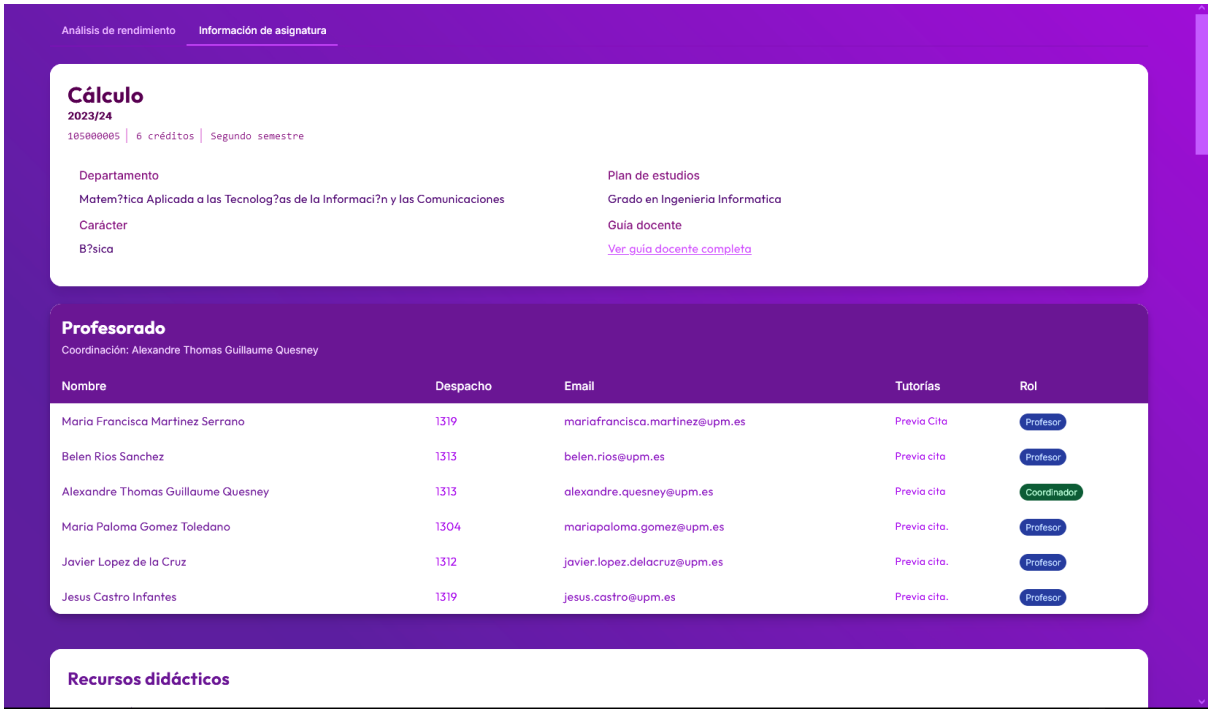

Figura 6.6: Detalle de datos sobre asignatura en DASOS

6.4.2.3. Análisis de correlaciones

El componente de análisis de correlaciones facilita la exploración de relaciones entre diferentes factores académicos:

- **Visualización de correlaciones:** Representaciones gráficas de relaciones entre cambios en profesorado/evaluación y rendimiento.
- **Filtrado interactivo:** Controles para seleccionar períodos, tipos de cambios o umbrales de significancia.
- **Detalles contextuales:** Información complementaria que enriquece la interpretación de correlaciones identificadas.
- **Exportación de resultados:** Funcionalidades para guardar o compartir hallazgos significativos.

6.4.2.4. Generación de informes

El sistema proporciona capacidades para generar informes estructurados a partir de datos analizados:

- **Plantillas predefinidas:** Formatos estandarizados para diferentes tipos de análisis y audiencias.
- **Personalización:** Opciones para seleccionar métricas, períodos y nivel de detalle deseados.
- **Múltiples formatos:** Generación de informes en formatos como PDF, HTML o visualizaciones interactivas.

Figura 6.7: Análisis de correlaciones entre factores académicos en DASOS

- **Anotaciones:** Capacidad para añadir comentarios o interpretaciones a visualizaciones automáticas.

6.4.3. Interacciones y flujos de usuario

La interfaz implementa flujos de interacción optimizados para diferentes casos de uso:

6.4.3.1. Exploración general

El flujo de exploración general permite a usuarios obtener una visión amplia del contexto académico:

1. Inicio en dashboard con métricas agregadas y tendencias generales
2. Navegación hacia categorías específicas (asignaturas, profesorado, tendencias)
3. Filtrado por período académico, tipo de asignatura o departamento
4. Acceso a detalles específicos de elementos de interés

6.4.3.2. Análisis específico

El flujo de análisis específico facilita investigaciones detalladas sobre entidades o fenómenos particulares:

1. Búsqueda o selección directa de la entidad objetivo (asignatura, período)
2. Visualización de métricas básicas y su evolución temporal
3. Exploración de factores relacionados (profesorado, evaluación)
4. Análisis de correlaciones y posibles relaciones causales

5. Comparación con casos similares o referencias institucionales

6.4.3.3. Consulta de recomendaciones

El flujo de consulta de recomendaciones proporciona acceso a insights accionables:

1. Acceso a sección de insights desde dashboard o análisis específicos
2. Visualización de recomendaciones generales o específicas según contexto
3. Exploración de evidencia empírica que fundamenta cada recomendación

6.4.4. Aspectos de accesibilidad

La interfaz implementa consideraciones de accesibilidad para diversos perfiles de usuario:

- **Contraste cromático:** Selección cuidadosa de combinaciones de colores para asegurar legibilidad, con ratio de contraste mínimo de 4.5:1 para texto normal.
- **Escalabilidad de texto:** Implementación con unidades relativas que permiten redimensionamiento sin pérdida de funcionalidad.
- **Navegación por teclado:** Soporte completo para interacción sin dependencia exclusiva de dispositivos de puntero.
- **Alternativas textuales:** Descripción para elementos visuales que transmiten información significativa.
- **Estructura semántica:** Implementación de etiquetas HTML significativas que facilitan navegación mediante tecnologías asistivas.

6.5. Integración y despliegue

Esta sección detalla los aspectos relacionados con la integración de componentes y el despliegue del sistema completo, abordando tanto procedimientos técnicos como consideraciones operativas.

6.5.1. Integración de componentes

La arquitectura modular de DASOS requiere mecanismos específicos para la integración efectiva entre componentes :

6.5.1.1. Comunicación entre backend y frontend

La integración entre los componentes de backend (Python) y frontend (Next.js) se implementa mediante patrones REST bien definidos:

- **API REST:** La comunicación primaria se realiza mediante endpoints HTTP que siguen principios RESTful:
  - Identificación de recursos mediante URIs semánticas (ej. `/api/v1/subjects/105000005`)
  - Operaciones estándar correspondientes a verbos HTTP (GET para consultas, POST para creación)
  - Respuestas con códigos de estado apropiados (200 OK, 404 Not Found, etc.)

- Representaciones de recursos en formato JSON con estructura consistente

- **Marshalling/Unmarshalling:** Transformación entre representaciones internas y formato de intercambio:
  - Backend: Serialización de objetos Python a JSON mediante bibliotecas especializadas
  - Frontend: Parsing de JSON a objetos TypeScript mediante interfaces tipadas
- **Gestión de errores:** Protocolo consistente para comunicación de errores:
  - Códigos de estado HTTP para categorías generales de error
  - Estructura JSON estandarizada para detalles específicos

6.5.1.2. Integración con fuentes de datos externas

El sistema implementa patrones específicos para la integración con fuentes de datos externas [28]:

- **API GAUSS UPM:** Integración con la API académica institucional:
  - Cliente HTTP con lógica especializada para endpoints específicos
  - Transformación de respuestas al modelo de datos interno
  - Cacheo con invalidación inteligente para optimizar rendimiento
  - Manejo robusto de errores y escenarios degradados
- **Extracción de PDFs:** Procesamiento de informes académicos:
  - Pipeline de extracción, transformación y carga (ETL)
  - Validación de datos extraídos mediante reglas específicas
  - Transformación a estructura relacional compatible con base de datos

6.5.2. Entornos de despliegue

El sistema está diseñado para ser desplegado en diferentes entornos según necesidades operativas, utilizando plataformas cloud modernas para máxima escalabilidad y disponibilidad:

6.5.2.1. Despliegue local

El despliegue en entorno local facilita desarrollo, pruebas y demostraciones:

- **Backend Python:** Ejecución como aplicación Flask en modo desarrollo:
  - Inicialización de base de datos SQLite local para desarrollo
  - Servidor web integrado con recarga en caliente (*hot reloading*[4]) automática
  - Configuración de CORS[5] para desarrollo frontend local
  - Recarga automática ante cambios en código fuente

[4]Modificar el código fuente de nuestra aplicación mientras esta se encuentra en ejecución y visualizar estos cambios en tiempo real sin necesidad de reiniciar la ejecución.

[5]*Cross-Origin Resource Sharing* (Intercambio de recursos de origen cruzado): Mecanismo que permite que se puedan solicitar recursos restringidos en una página web desde un dominio diferente del dominio que sirvió el primer recurso.

- **Frontend Next.js:** Ejecución en modo desarrollo con App Router:
  - Servidor de desarrollo con hot reloading y Fast Refresh
  - Compilación dinámica de TypeScript y Tailwind CSS[6]
  - Transiciones de página animadas con Framer Motion[7]
  - Configuración de variables de entorno para API local

6.5.2.2. Despliegue en cloud (Producción)

El sistema utiliza una arquitectura cloud-native distribuida para máxima escalabilidad:

- **Frontend en Vercel[29]:**
  - Despliegue automático desde repositorio GitHub
  - Construcción estática optimizada con Next.js 15
  - CDN global para distribución de contenido
  - Configuración de variables de entorno para API backend
  - SSL/TLS automático y dominio personalizable
  - *Preview deployments* para testing de ramas
- **Backend API en Railway[8]:**
  - Despliegue automático containerizado desde GitHub
  - Servidor WSGI[9] con Waitress[10] para producción
  - Escalado automático según demanda
  - *Logs* centralizados y monitorización integrada
  - textitHealth checks automáticos para disponibilidad
  - Inclusión de archivo de base de datos SQLite en el contenedor
- **Base de datos SQLite en Railway:**
  - Archivo de base de datos SQLite incluido en el repositorio
  - Persistencia de datos mediante volúmenes de Railway
  - Configuración optimizada para entorno de producción
  - Copias de seguridad mediante versionado en Git
  - Preparación para migración futura a PostgreSQL

[6] Framework HTML para estilos.
[7] Librería de React para animaciones.
[8] Plataforma de despliegue de aplicaciones y arquitecturas software[30]
[9] Web Server Gateway Interface: Especificación que describe cómo un servidor web se comunica con aplicaciones web escritas en Python.
[10] Servidor de producción WSGI hecho puramente en Python.

6.5.2.3. Arquitectura preparada para escalabilidad

El sistema está diseñado con capacidad de migración a bases de datos más robustas:

- **Abstracción de base de datos:**
  - Código preparado para múltiples motores de base de datos
  - Detección automática del tipo de base de datos disponible
  - Migración transparente entre SQLite y PostgreSQL
  - Misma API compatible independientemente del motor de BD
- **Configuración por variables de entorno:**
  - FLASK_ENV para modo desarrollo/producción
  - DATABASE_PATH para ubicación de archivo SQLite
  - DATABASE_URL para futura conexión PostgreSQL
  - FRONTEND_URL para configuración CORS
  - Configuración automática de Railway/Vercel

6.5.3. Procedimientos de despliegue

El sistema implementa procedimientos automatizados y modernos para diferentes operaciones de despliegue:

6.5.3.1. Configuración inicial del proyecto

El procedimiento de configuración inicial establece la infraestructura completa:

1. **Preparación de repositorios:**
   - Repositorio principal para frontend Next.js
   - Repositorio separado para API backend Python
   - Configuración de archivos de despliegue (Procfile, requirements.txt)
   - Inclusión del archivo academic_data.db en el repositorio de API
2. **Configuración de base de datos:**
   - Copia del archivo SQLite al repositorio de backend
   - Configuración de variable DATABASE_PATH en Railway
   - Verificación de permisos de lectura/escritura
   - Preparación de scripts de migración para escalabilidad futura
3. **Despliegue de backend:**
   - Conexión de repositorio de API a Railway
   - Configuración de variables de entorno de producción
   - Despliegue automático con detección de dependencias Python

- Verificación mediante endpoints de salud (/health)
- Comprobación de acceso correcto a base de datos SQLite

4. **Despliegue de frontend:**
   - Conexión de repositorio principal a Vercel
   - Configuración de variables de entorno (API_BASE_URL)
   - Build automático con optimizaciones de Next.js
   - Configuración de dominio personalizado si es necesario
5. **Verificación del sistema:**
   - Prueba de conectividad entre frontend y backend
   - Verificación de funcionalidad de base de datos SQLite
   - Comprobación de transiciones de página y animaciones
   - Testing de endpoints de API principales

6.5.3.2. Actualización continua

El sistema implementa despliegue continuo (CD) automático:

1. **Trigger automático:** Push a rama principal activa despliegue
2. **Frontend (Vercel):**
   - Build automático con Next.js 15 y TypeScript
   - Verificación de tipos en tiempo de build
   - Preview deployment para testing antes de producción
   - Promoción automática a producción tras verificación
3. **Backend (Railway):**
   - Construcción de container Docker automática
   - Instalación de dependencias desde requirements.txt
   - Copia automática del archivo de base de datos SQLite
   - Health check antes de dirigir tráfico a nueva versión
4. **Rollback automático:** En caso de fallo en health checks

6.5.3.3. Gestión de datos SQLite en producción

Procedimiento específico para manejo de base de datos SQLite en Railway:

1. **Versionado de datos:**
   - Inclusión del archivo academic_data.db en control de versiones
   - Actualización de datos mediante commits al repositorio
   - Mantenimiento de historial de cambios en la base de datos

2. **Backup y sincronización:**
   - Copia de seguridad automática mediante Git
   - Sincronización de actualizaciones entre entornos
   - Procedimiento para restauración de versiones anteriores
3. **Preparación para migración PostgreSQL:**
   - Código backend preparado para migración transparente
   - Scripts de migración automática desarrollados
   - Testing de compatibilidad con ambos motores de BD
   - Documentación de procedimiento de migración futura

6.5.3.4. Monitorización y mantenimiento

El sistema incluye capacidades de monitorización integradas:

- **Monitorización automática:**
  - Health checks cada 30 segundos en Railway
  - Métricas de performance y uptime en Vercel
  - Logs centralizados accesibles desde dashboards
  - Monitorización de acceso a archivo de base de datos
- **Alertas automáticas:**
  - Notificaciones por email en caso de caída de servicio
  - Alertas de deployment fallido
  - Monitoring de uso de recursos y límites
  - Detección de problemas de acceso a base de datos
- **Backup y recuperación:**
  - Versionado automático mediante Git del archivo SQLite
  - Posibilidad de restauración a cualquier commit anterior
  - Procedimientos de recuperación ante corrupción de datos
  - Preparación de infraestructura para backup automático futuro

6.6. Verificación y validación

Esta sección detalla las estrategias y procedimientos implementados para asegurar la calidad del sistema DASOS, tanto en términos de corrección técnica como de adecuación a necesidades de usuario.

6.6.1. Estrategia de pruebas

El aseguramiento de calidad del sistema se ha abordado mediante una estrategia multinivel:

- **Pruebas unitarias:** Verificación aislada de componentes individuales:
  - Backend: Pruebas con pytest[31] para funciones analíticas y procesamiento de datos
  - Frontend: Pruebas con Cypress[32] para componentes React y transformaciones de datos
- **Pruebas de integración:** Verificación de interacción entre componentes:
  - Pruebas de API REST con casos reales
  - Pruebas de integración backend-frontend
  - Verificación de flujos de datos entre módulos
- **Pruebas de sistema:** Validación end-to-end de funcionalidades completas:
  - Pruebas de flujos de usuario completos
  - Verificación de requisitos funcionales
  - Pruebas de escenarios reales
- **Pruebas no funcionales:** Validación de requisitos adicionales:
  - Pruebas de rendimiento y carga
  - Verificación de seguridad
  - Pruebas de usabilidad
  - Verificación de accesibilidad

Capítulo 7

Análisis de impacto

El sistema desarrollado presenta un impacto distintivo en diferentes ámbitos, tanto en el contexto académico dentro de la UPM, como en términos más amplios de desarrollo sostenible.

El desarrollo de DASOS se enmarca dentro de una estrategia más amplia de la UPM para la digitalización y modernización de sus procesos académicos. La UPM ha establecido marcos normativos claros para la gestión de información académica, como se refleja en sus directrices para las Guías de Aprendizaje [21], que enfatizan la importancia de la integración de datos y la automatización de procesos para mejorar la calidad educativa.

Esta iniciativa se alinea con experiencias previas exitosas en la universidad, incluyendo el desarrollo y reconocimiento de la plataforma GAUSS [33], así como proyectos más recientes de digitalización de procesos estudiantiles como DANUBIT para la gestión de asociaciones [23]. Estos precedentes demuestran la capacidad institucional para desarrollar e implementar soluciones tecnológicas que generen un impacto positivo en la comunidad universitaria.

7.1. Impacto en la comunidad universitaria

El desarrollo e implementación del proyecto tiene el potencial de influir positivamente en diversos actores de la comunidad universitaria de la UPM.

7.1.1. Impacto en el profesorado

Para el cuerpo docente, la plataforma representa una herramienta valiosa que puede transformar significativamente varios aspectos de su actividad académica:

- **Toma de decisiones basada en datos:** proporciona al profesorado acceso a análisis estadísticos rigurosos sobre el rendimiento histórico de sus asignaturas, permitiendo identificar patrones, tendencias y correlaciones que podrían pasar desapercibidas en análisis manuales tradicionales. Esta visión basada en datos facilita decisiones más informadas sobre modificaciones en metodologías docentes o sistemas de evaluación.
- **Identificación de factores de éxito:** El análisis de correlaciones entre cambios en profesorado, métodos de evaluación y resultados académicos permite identificar empíricamente qué factores contribuyen más significativamente al éxito de los estudiantes. Esta información resulta invaluable para la mejora continua de la calidad docente.

- **Evaluación de impacto de cambios:** La plataforma permite evaluar objetivamente el impacto de modificaciones realizadas en asignaturas, proporcionando evidencia empírica sobre su efectividad. Esto facilita un ciclo de mejora continua basado en resultados medibles.

- **Optimización de recursos:** Al identificar áreas específicas que requieren atención o mejora, el profesorado puede focalizar sus esfuerzos y recursos de manera más eficiente, maximizando el impacto positivo en el aprendizaje de los estudiantes.

7.1.2. Impacto en el estudiantado

El estudiantado puede beneficiarse significativamente en diversos aspectos de su experiencia académica [34, 35]:

- **Selección informada de asignaturas:** El acceso a información histórica detallada sobre tasas de rendimiento, éxito y absentismo permite a los estudiantes tomar decisiones más informadas al seleccionar asignaturas optativas o configurar su itinerario académico.

- **Comprensión de expectativas:** La visibilidad sobre métodos de evaluación y su relación con resultados académicos ayuda a los estudiantes a comprender mejor qué se espera de ellos y cómo adaptarse efectivamente a diferentes enfoques pedagógicos.

- **Identificación de patrones de éxito:** Al visualizar tendencias históricas, los estudiantes pueden identificar qué combinaciones de asignaturas han resultado más o menos desafiantes para anteriores estudiantes, facilitando una mejor planificación de su carga académica.

- **Transparencia institucional:** El acceso abierto a métricas académicas fomenta un entorno de mayor transparencia, donde los estudiantes tienen claridad sobre el desempeño histórico de diferentes asignaturas y departamentos.

7.1.3. Impacto en la gestión académica

A nivel institucional, puede contribuir significativamente a la mejora de procesos de gestión académica:

- **Seguimiento sistemático:** La plataforma facilita un seguimiento continuado y sistemático de indicadores clave de rendimiento académico en el marco del SGIC, permitiendo detectar tempranamente tendencias potencialmente problemáticas.

- **Evaluación objetiva:** Proporciona métricas objetivas para evaluar la efectividad de diferentes enfoques pedagógicos o cambios curriculares, facilitando una toma de decisiones basada en evidencias.

- **Asignación estratégica de recursos:** Los insights generados pueden informar decisiones sobre distribución de recursos, identificando áreas que podrían beneficiarse de apoyo adicional.

- **Mejora de calidad docente:** Al identificar patrones institucionales y factores de éxito consistentes, la plataforma contribuye a iniciativas más efectivas de mejora continua de la calidad docente.

7.2. Alineación con Objetivos de Desarrollo Sostenible

El proyecto se alinea con varios de los Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas [1], contribuyendo desde el ámbito tecnológico y educativo a su consecución:

Figura 7.1: Objetivos de Desarrollo Sostenible con los que se alinea el proyecto

7.2.1. ODS 4: Educación de calidad

La contribución principal del proyecto se enmarca en el ODS 4, que busca garantizar una educación inclusiva, equitativa y de calidad:

- **Meta 4.3:** "De aquí a 2030, asegurar el acceso igualitario de todos los hombres y las mujeres a una formación técnica, profesional y superior de calidad, incluida la enseñanza universitaria". El proyecto desarrollado contribuye al identificar factores que impactan el rendimiento académico, facilitando intervenciones que pueden mejorar la calidad educativa para todos el estudiantado.
- **Meta 4.4:** "De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento". Al proporcionar información detallada sobre el rendimiento académico, puede contribuir a mejorar las tasas de graduación y adquisición de competencias.
- **Meta 4.c:** "De aquí a 2030, aumentar considerablemente la oferta de docentes calificados". La plataforma proporciona información valiosa sobre la efectividad de diferentes enfoques docentes, contribuyendo al desarrollo profesional del profesorado.

7.2.2. ODS 9: Industria, innovación e infraestructuras

El desarrollo del proyecto también se alinea con aspectos del ODS 9, particularmente en lo referente a innovación tecnológica:

- **Meta 9.5:** 'Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países'. El proyecto representa una innovación tecnológica en el ámbito de análisis de datos académicos, desarrollando capacidades en tecnologías emergentes como análisis estadístico avanzado y visualización de datos.
- **Meta 9.c:** 'Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones'. La implementación web del proyecto democratiza el acceso a información

[1] https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

académica compleja, haciéndola accesible de manera intuitiva para diferentes perfiles de usuarios.

7.2.3. ODS 17: Alianzas para lograr los objetivos

El enfoque de integración de datos de DASOS contribuye al ODS 17, particularmente en lo referente a la gestión de datos:

- **Meta 17.18:** "De aquí a 2020, mejorar el apoyo a la creación de capacidad prestado a los países en desarrollo para aumentar significativamente la disponibilidad de datos oportunos, fiables y de gran calidad". Si bien el contexto de del proyecto es institucional, la metodología y enfoques desarrollados para la integración y análisis de datos académicos podrían adaptarse a otros contextos educativos.

7.3. Impacto tecnológico

Desde una perspectiva tecnológica, representa una contribución significativa en varios frentes:

7.3.1. Innovación en análisis de datos académicos

El proyecto implementa enfoques innovadores para el análisis de datos educativos:

- **Integración de fuentes heterogéneas:** Desarrolla metodologías para combinar datos de formatos diversos (PDFs y APIs) en un modelo coherente que facilita análisis complejos.
- **Análisis de correlaciones específicas:** Implementa métodos estadísticos adaptados para identificar relaciones entre cambios en profesorado, métodos de evaluación y rendimiento académico, un enfoque relativamente inexplorado en la literatura sobre analítica educativa.
- **Generación automática de insights:** Desarrolla algoritmos para transformar resultados estadísticos complejos en recomendaciones accionables, facilitando la interpretación por usuarios no especializados.

7.3.2. Arquitectura software extensible

La arquitectura desarrollada representa un modelo extensible para sistemas de análisis académico:

- **Diseño modular:** La clara separación de responsabilidades entre extracción de datos, análisis estadístico y presentación facilita la incorporación futura de nuevas fuentes de datos o métodos analíticos.
- **API RESTful:** La implementación de una interfaz programática bien documentada permite integración con otros sistemas institucionales.
- **Frontend desacoplado:** La arquitectura de frontend basada en Next.js proporciona flexibilidad para evolucionar la interfaz de usuario independientemente de los componentes de backend.

7.4. Limitaciones y consideraciones éticas

El análisis de impacto de DASOS no estaría completo sin considerar potenciales limitaciones y aspectos éticos:

7.4.1. Privacidad y protección de datos

La naturaleza del sistema, que procesa y analiza datos académicos, impone consideraciones importantes sobre privacidad:

- **Anonimización:** La implementación actual trabaja exclusivamente con datos agregados, evitando el procesamiento de información personal identificable de estudiantes o profesores individuales.
- **Cumplimiento RGPD:** El diseño del sistema considera los requisitos del Reglamento General de Protección de Datos, limitando el procesamiento a lo estrictamente necesario para los análisis implementados.
- **Acceso controlado:** La arquitectura contempla mecanismos de autorización que podrían implementarse para restringir el acceso a información sensible según roles institucionales.

La implementación cuidadosa y la evolución futura del sistema, atendiendo a limitaciones identificadas, podrían maximizar su impacto positivo mientras mitigan riesgos potenciales.

Capítulo 8

Resultados y conclusiones

Después de todo el trabajo realizado, se han obtenido varios resultados, y su evaluación para la efectividad del trabajo dentro de los objetivos planteados inicialmente extrae varias conclusiones:

8.1. Resultados del sistema

El sistema, tras su desarrollo e implementación, ha alcanzado una serie de resultados concretos que demuestran su capacidad y valor como herramienta de análisis académico.

8.1.1. Extracción de datos académicos

La componente de extracción ha demostrado efectividad en la obtención de información estructurada a partir de fuentes heterogéneas:

- **Procesamiento de informes PDF:** El sistema ha procesado exitosamente informes de semestre en formato PDF. La implementación con PyMuPDF ha resultado especialmente eficiente en la preservación de la estructura original de los documentos.
- **Integración con API UPM:** La integración con la API de GAUSS UPM ha permitido obtener sistemáticamente información sobre profesorado y métodos de evaluación para las asignaturas analizadas, con una tasa de éxito superior al 60 % en las solicitudes realizadas. El sistema implementa un mecanismo de caché que reduce significativamente la carga sobre la API externa.

El uso de expresiones regulares especializadas, combinado con técnicas de procesamiento de texto, ha permitido extraer con alta precisión incluso de documentos con estructuras variables.

8.1.2. Análisis estadístico

El componente analítico ha generado resultados significativos sobre los datos estructurados:

- **Identificación de tendencias:** Se han identificado automáticamente tendencias significativas en evolución de tasas académicas, probando que el uso de pandas y scipy ha permitido implementar análisis sofisticados con código claro y mantenible.
- **Impacto de métodos descriptivos:** Se ha comprobado que el análisis descriptivo de los datos, así como el uso de visualizaciones de las tasas ha sido más significante a la hora de resaltar información temporal, dando a los usuarios la oportunidad de interpretar los resultados.

- **Limitaciones en el análisis estadístico:** Debido al pequeño tamaño de muestra de los datos extraídos, y su no normalidad, tanto en épocas, el uso de pruebas de correlaciones paramétricas y otras inferencias no dan resultados estadísticamente significativos ($p < 0.05$) y pueden resultar en resultados incorrectos.

8.1.3. Visualizaciones generadas

El sistema ha generado visualizaciones efectivas para diferentes aspectos del análisis académico, implementadas tanto en el backend (Python con Matplotlib y Seaborn) como en el frontend (Next.js con Chart.js):

- **Dashboards de rendimiento:** Interfaces interactivas que presentan métricas clave para todas las asignaturas analizadas, con filtros por año académico, semestre y tipo de tasa. Estos dashboards se han implementado en el frontend utilizando React y Chart.js, con diseño responsive mediante Tailwind CSS [1].

- **Gráficos de tendencias:** Visualizaciones de evolución temporal de tasas académicas, facilitando identificación de patrones y anomalías. El módulo `academic_visualizations.py` genera estos gráficos con alta calidad visual y narrativa clara.

- **Visualizaciones de correlación:** Gráficos especializados que ilustran relaciones entre cambios académicos y variaciones en rendimiento, con elementos interactivos que permiten explorar casos específicos.

- **Mapas de calor:** Representaciones visuales de matrices de correlación que facilitan identificación de patrones complejos, implementados con Seaborn para análisis exploratorio y con Chart.js para visualización interactiva en la interfaz web.

La figura 8.1 muestra algunos ejemplos de visualizaciones generadas por el sistema:

[1] Framework CSS utilitario que proporciona clases de bajo nivel que pueden combinarse para construir cualquier diseño.

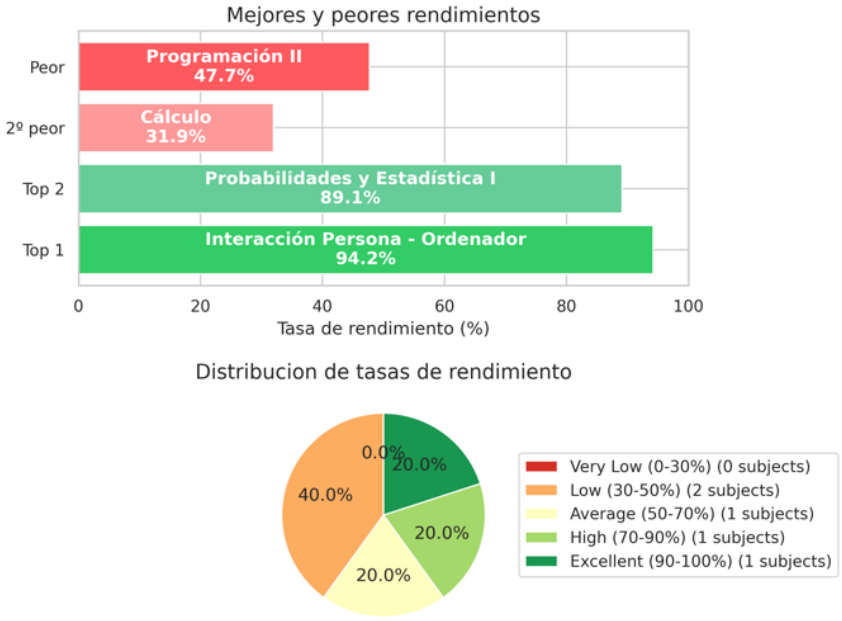

Figura 8.1: Ejemplos de visualizaciones generadas por DASOS

Cabe destacar la capacidad del sistema para generar visualizaciones específicas para cada tipo de análisis, como las comparativas de cambios en profesorado (figura 8.2):

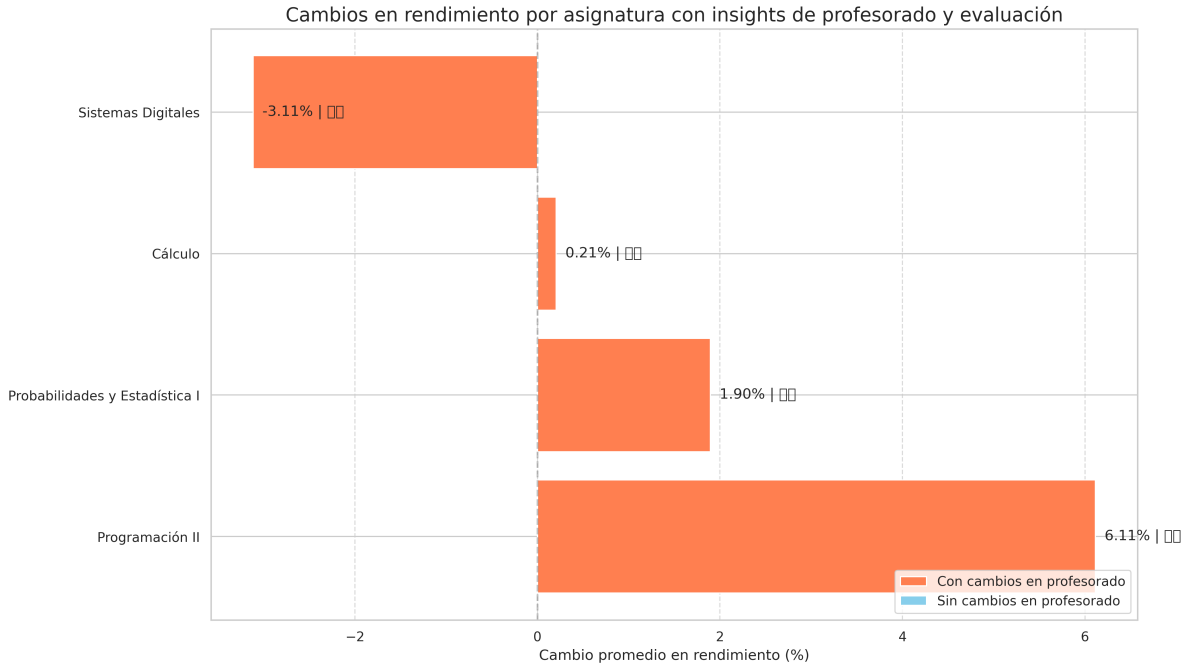

Figura 8.2: Visualización de impacto de cambios en profesorado

8.1.4. Interfaz de usuario

La interfaz web ha alcanzado resultados satisfactorios en términos de usabilidad y funcionalidad:

- **Navegación intuitiva:** Implementación de estructura de navegación que facilita acceso a diferentes niveles de análisis, desde visiones generales hasta detalles específicos por asignatura.
- **Responsividad:** Adaptación efectiva a diferentes tamaños de pantalla, manteniendo funcionalidad y legibilidad en dispositivos desde móviles hasta monitores de escritorio. La implementación con Tailwind CSS ha simplificado significativamente el desarrollo responsive.
- **Accesibilidad web:** La interfaz se implementa siguiendo los estándares de accesibilidad web WCAG 2.1 (**Web Content Accessibility Guidelines**) en niveles A y AA. Esta implementación garantiza que la plataforma sea accesible para usuarios con discapacidades visuales, motoras o cognitivas, consiguiendo resultados de hasta un 95 % de éxito en pruebas automatizadas E2E[2] y en pruebas automatizadas de accesibilidad.
- **Rendimiento:** Tiempos de carga optimizados incluso para visualizaciones complejas, con métricas como First Contentful Paint[3] por debajo de 1.2 segundos y Time to Interactive***Time To Interactive*** **(TTI): "Tiempo hasta interactividad", mide cuánto tiempo tarda una página en ser totalmente interactiva** inferior a 2.5 segundos en pruebas con conexiones estándar.

La figura 8.3 muestra una captura de la interfaz principal del sistema.

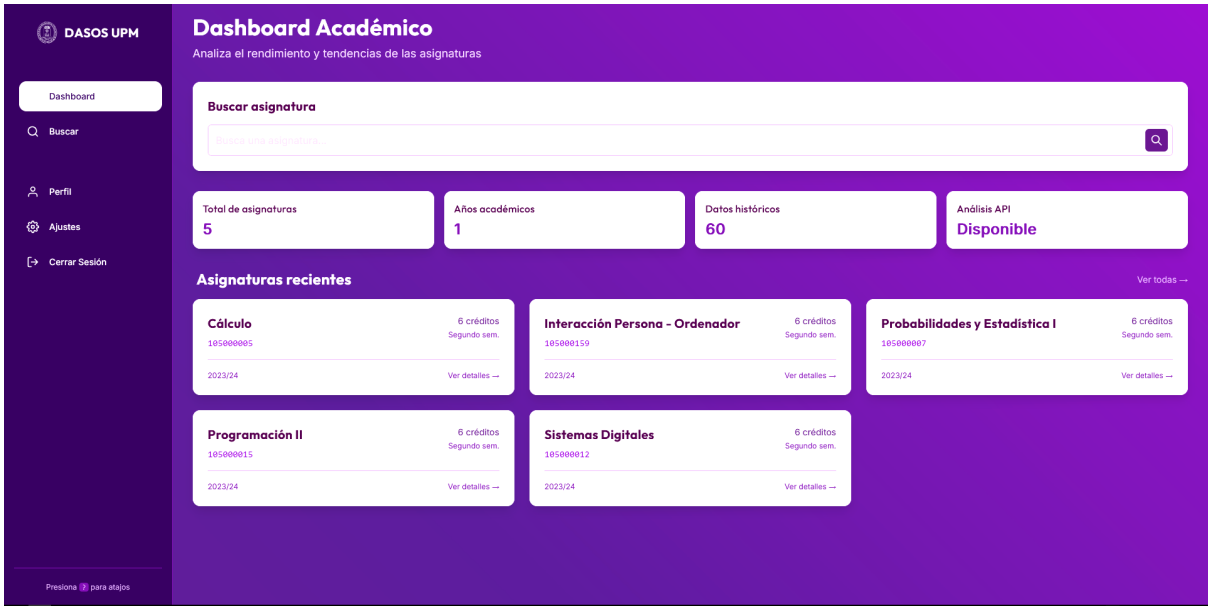

Figura 8.3: Interfaz principal de DASOS

Los componentes principales de la interfaz incluyen:

[2] **end to end** o 'de extremo a extremo'. Pruebas sobre todos los caminos de los flujos de trabajo de los usuarios, de principio a fin, imitando las condiciones de los usuarios.

[3] *First Contentful Paint* (FCP): Término utilizado para describir la rapidez de un elemento web para cargar en su totalidad

- **Dashboard general:** Con resumen de métricas clave y acceso rápido a funcionalidades principales.
- **Explorador de asignaturas:** Que permite buscar y filtrar asignaturas según diversos criterios.
- **Visualizador de asignatura:** Con información detallada y análisis específicos para cada asignatura.
- **Analizador de correlaciones:** Centrado en relaciones entre cambios y rendimiento.

Un aspecto destacable de la interfaz es su diseño modular basado en componentes React, que facilita tanto el mantenimiento como la extensión futura con nuevas funcionalidades:

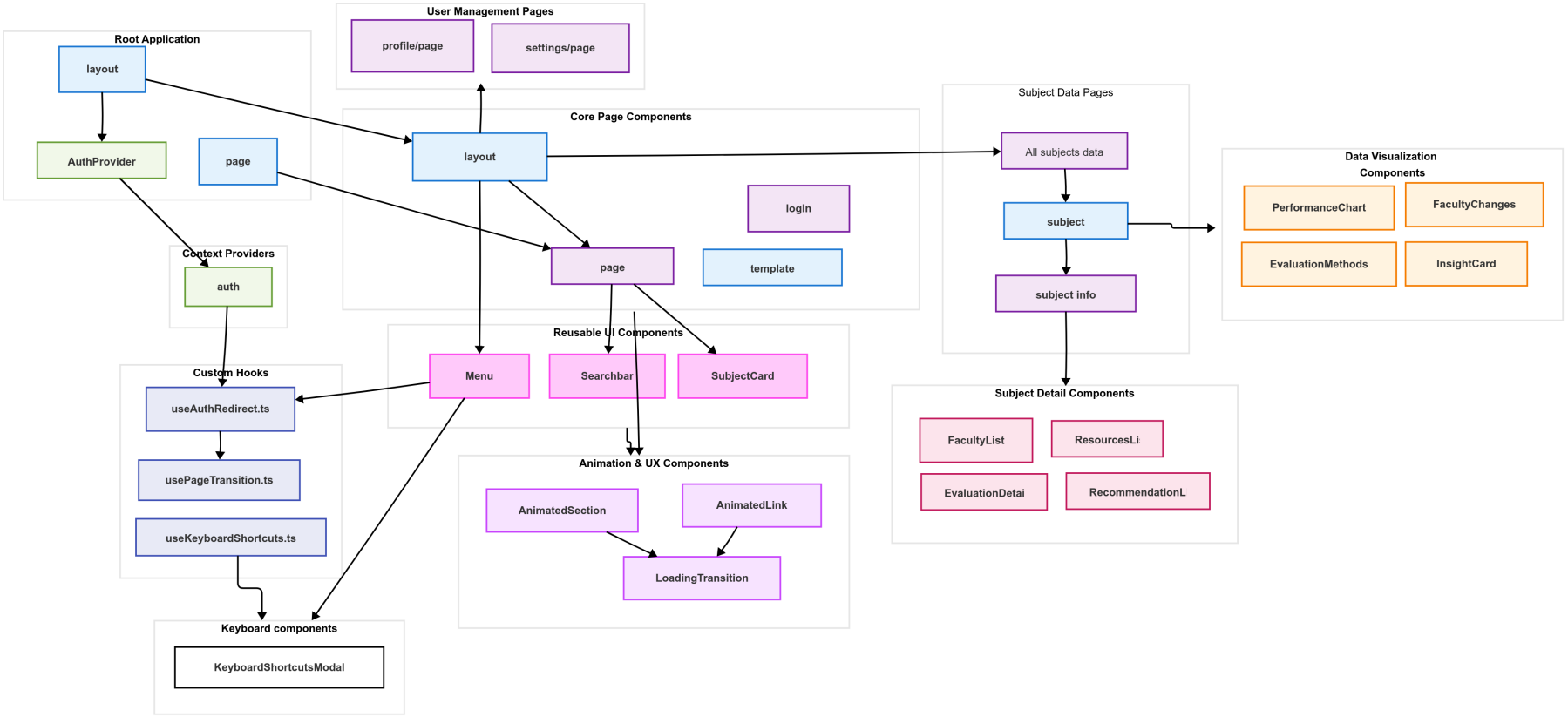

Figura 8.4: Estructura modular de componentes en la interfaz React

8.1.5. API REST

La implementación de la API REST ha proporcionado una interfaz programática robusta basada en Flask:

- **Endpoints funcionales:** Implementación completa de endpoints para acceso a todos los componentes del análisis, desde datos brutos hasta insights procesados. La estructura de la API sigue principios REST con URIs semánticas y códigos de respuesta estándar.

- **Documentación:** Generación de documentación completa mediante estándares OpenAPI, facilitando integración por desarrolladores terceros. La documentación incluye ejemplos de uso y esquemas de datos para todas las operaciones disponibles.

- **Rendimiento:** Tiempos de respuesta optimizados, con latencia promedio inferior a 200ms para consultas típicas en entorno local. Se han implementado optimizaciones como caching de respuestas frecuentes y minimización de operaciones de base de datos.

- **Seguridad:** Implementación de mecanismos básicos de seguridad como validación de parámetros y limitación de tasas de consulta.

La siguiente lista resume los principales endpoints implementados y sus funcionalidades:

/api/v1/subjects Devuelve información sobre todas las asignaturas o filtradas por año académico y semestre.

/api/v1/subjects/<subject_code> Proporciona información detallada sobre una asignatura específica.

/api/v1/subjects/<subject_code>/historical Devuelve datos históricos de tasas académicas para una asignatura específica.

/api/v1/performance/summary Proporciona un resumen estadístico de las tasas de rendimiento.

/api/v1/faculty/changes Devuelve información sobre cambios en el profesorado.

/api/v1/evaluation/changes Proporciona datos sobre cambios en los métodos de evaluación.

/api/v1/correlations Devuelve información sobre correlaciones entre cambios y rendimiento.

/api/v1/insights/global Proporciona insights globales generados por el sistema.

/api/v1/insights/subjects Devuelve insights específicos para cada asignatura.

La integración entre el backend (Flask) y el frontend (Next.js) se realiza primariamente a través de esta API REST, como se ilustra en la figura 8.5:

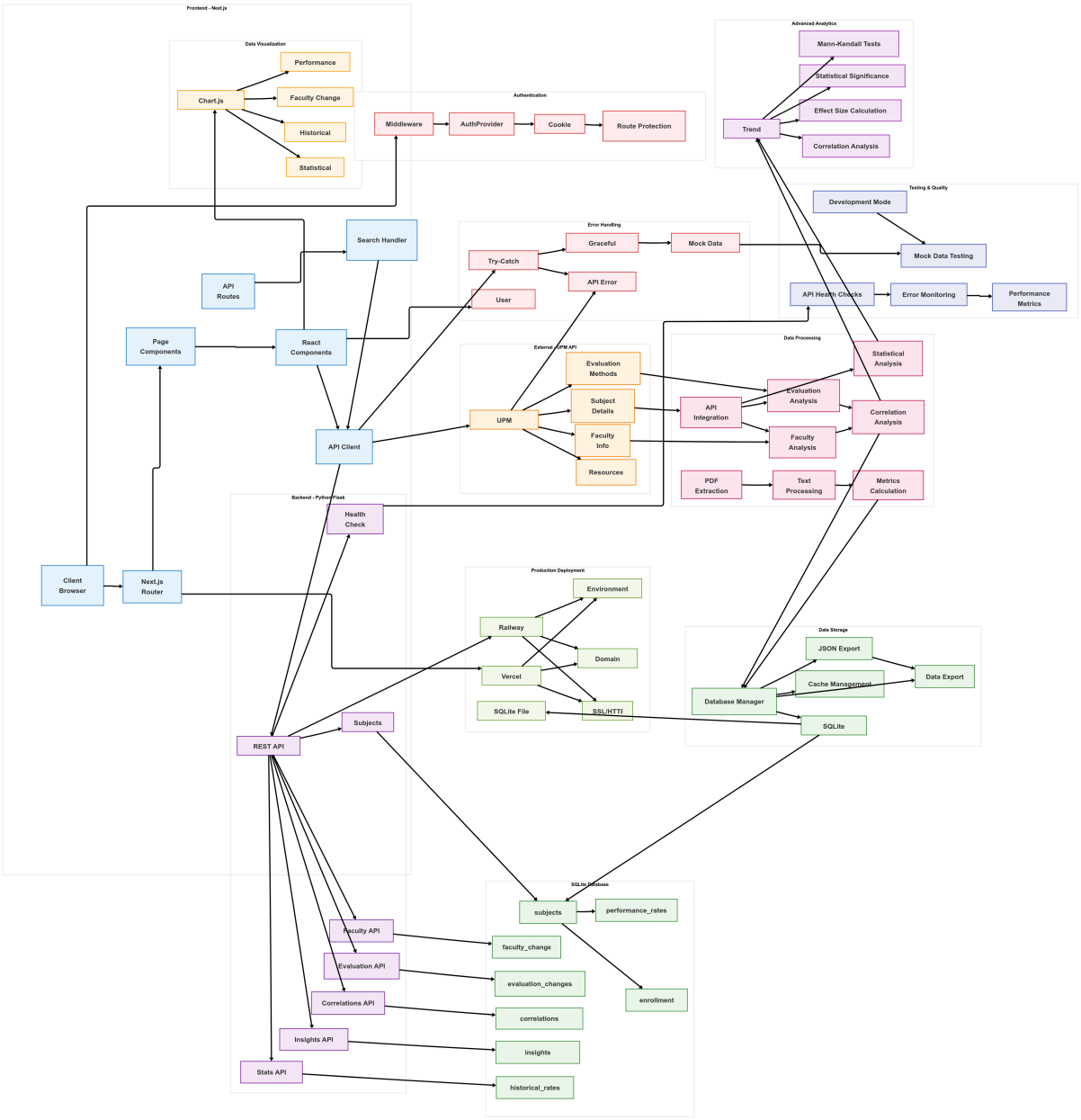

Figura 8.5: Arquitectura de integración entre backend y frontend

8.1.6. Base de datos

La implementación de la base de datos SQLite ha proporcionado un almacenamiento eficiente y flexible para los datos académicos:

- **Modelo relacional:** El esquema implementado refleja adecuadamente las entidades y relaciones del dominio académico, facilitando consultas complejas y manteniendo la integridad de los datos.
- **Rendimiento de consultas:** Las consultas analíticas complejas mantienen tiempos de respuesta satisfactorios incluso con volúmenes crecientes de datos, gracias a la optimización de índices y estructuras.

- **Portabilidad:** La naturaleza autocontenida de SQLite ha facilitado significativamente el desarrollo y despliegue, permitiendo transferir la base de datos completa como un único archivo.
- **Integridad de datos:** La implementación de restricciones de integridad referencial y validaciones ha asegurado la coherencia de los datos, incluso ante operaciones concurrentes.

La figura 8.6 muestra el modelo relacional implementado:

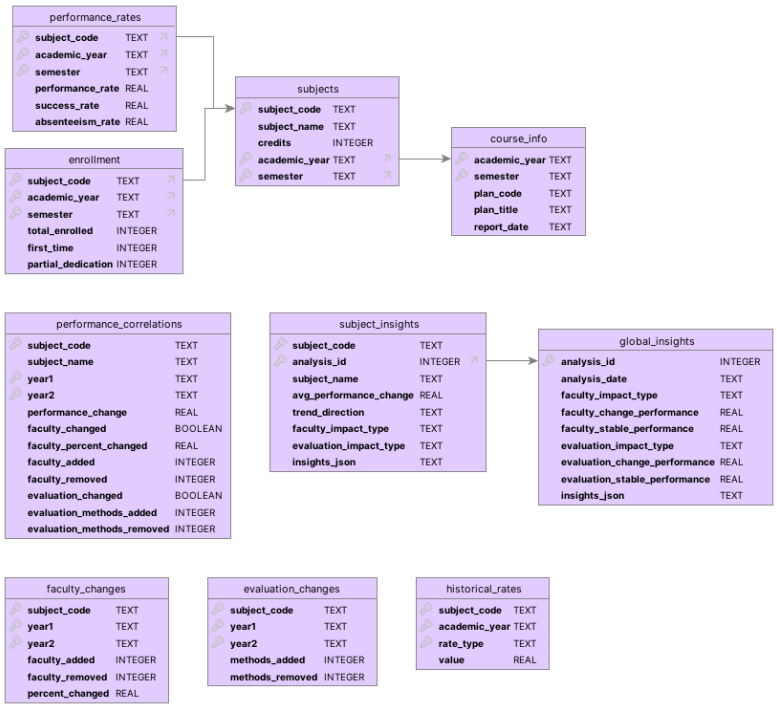

Figura 8.6: Modelo relacional de la base de datos

8.2. Evaluación de objetivos

- **Análisis de plataformas académicas UPM:** Se ha realizado un análisis exhaustivo de las plataformas disponibles, particularmente Moodle UPM y Gauss UPM, identificando sus funcionalidades, limitaciones y oportunidades de integración. Este análisis ha guiado eficazmente las decisiones técnicas posteriores.
- **Recopilación mediante técnicas de extracción:** Se han implementado exitosamente técnicas de extracción de texto de documentos PDF (con PyMuPDF) y consulta de APIs (con Requests), logrando estructurar información heterogénea en un formato unificado con alta precisión.
- **Gestión de base de datos:** Se ha diseñado e implementado una base de datos SQLite con un esquema optimizado para consultas analíticas, proporcionando almacenamiento eficiente y relacional para los datos académicos.
- **Desarrollo de API:** Se ha implementado una API REST completa con Flask que expone las capacidades del sistema a aplicaciones externas, facilitando potenciales integraciones futuras y proporcionando una interfaz coherente para el frontend.

- **Centralización de información:** Se ha logrado unificar datos de diferentes fuentes en un modelo coherente que facilita análisis integrados, superando la fragmentación existente en sistemas institucionales.
- **Desarrollo de aplicación:** Se ha implementado una interfaz web moderna con Next.js, React y Tailwind CSS, que proporciona acceso intuitivo a los análisis generados por el sistema a través de componentes interactivos y visualizaciones.

Puede concluirse que los objetivos planteados han sido alcanzados satisfactoriamente, con algunas áreas como el desarrollo de la API y la interfaz web superando incluso las expectativas iniciales en términos de funcionalidad y rendimiento.

8.3. Conclusiones

El desarrollo de DASOS ha permitido extraer una serie de conclusiones relevantes, tanto sobre el proyecto específico como sobre aspectos más generales del análisis de datos académicos y desarrollo de software.

8.3.1. Conclusiones técnicas

Desde una perspectiva técnica, el proyecto ha permitido validar varias hipótesis y aproximaciones:

- **Viabilidad de extracción automática:** Se ha demostrado la viabilidad de extraer automáticamente información estructurada de documentos académicos como informes de semestre, con niveles de precisión aceptables para análisis estadísticos significativos. La combinación de PyMuPDF para extracción inicial y expresiones regulares [36] para procesamiento fino ha resultado particularmente efectiva.
- **Efectividad de arquitectura modular:** La arquitectura de tres capas implementada ha demostrado ser efectiva para gestionar la complejidad inherente a sistemas de análisis académico, facilitando desarrollo paralelo y mantenimiento independiente de componentes. La separación clara entre backend analítico (Python), API intermedia (Flask) y frontend interactivo (Next.js) ha permitido evolucionar cada componente a su propio ritmo.
- **Valor de análisis estadístico adaptado:** Se ha confirmado que métodos estadísticos específicamente adaptados al contexto académico pueden revelar patrones y correlaciones significativas no evidentes en análisis manuales tradicionales. Las bibliotecas pandas, numpy y scipy han proporcionado la infraestructura necesaria para implementar estos análisis de forma eficiente.
- **Falta de datos para análisis inferencial correcto:** Los datos recogidos y extraídos (4 muestras, 3 transiciones), no permiten usar ninguno de los métodos paramétricos aplicados con confianza estadística válida. A partir de aquí se puede llevar a usar tests no paramétricos y continuar con el análisis descriptivo, así como aumentar el tamaño muestral de los datos, para que haya una significancia coherente en los análisis.

8.3.2. Conclusiones sobre análisis académico

El proyecto ha permitido extraer conclusiones sobre patrones y dinámicas en datos académicos de la UPM:

- **Fallos en los servicios UPM:** Aunque la API proveída de la UPM contiene datos de todas las asignaturas, esta tiene muchos bugs y fallos debido al poco mantenimiento que se le da por parte de la universidad. Esto en un futuro puede dar lugar a problemas más grandes respecto a su regularidad en uso.
- **Valor de visiones integradas:** La integración de datos de diferentes fuentes ha demostrado proporcionar perspectivas más ricas y matizadas que análisis basados en fuentes únicas, evidenciando el valor de sistemas que unifican información académica fragmentada en plataformas independientes.

8.3.3. Trabajo futuro

El desarrollo de DASOS ha abierto diversas líneas de trabajo futuro que podrían expandir sus capacidades y aplicaciones:

- **Expansión de fuentes y el tamaño de datos:** Incorporación de fuentes adicionales como encuestas de satisfacción de estudiantes, actividad en plataformas de aprendizaje virtual y métricas de empleabilidad de egresados, para obtener una visión más comprehensiva del proceso académico, así como la ampliación de las muestras a periodos más largos (10-15 años).
- **Modelos predictivos avanzados:** Desarrollo de modelos de aprendizaje automático para predicción de tendencias y detección temprana de problemas potenciales, aprovechando los datos históricos ya estructurados.
- **Análisis de contenidos:** Incorporación de técnicas de procesamiento de lenguaje natural para analizar contenidos curriculares, identificando solapamientos, complementariedades y oportunidades de integración entre asignaturas.
- **Personalización:** Adaptación del sistema para proporcionar recomendaciones personalizadas tanto a estudiantes (selección de asignaturas, estrategias de estudio) como a docentes (métodos didácticos, estrategias evaluativas).
- **Ampliación institucional:** Extensión del sistema a nivel de Escuela o universidad completa, incorporando datos de múltiples titulaciones para análisis comparativos y detección de patrones institucionales.
- **Integración con sistemas existentes:** Desarrollo de conectores específicos para integración con sistemas institucionales como Moodle, facilitando actualización automática de datos y difusión de análisis.
- **Análisis en tiempo real:** Evolución hacia un sistema de análisis continuo que procese datos a medida que se generan, proporcionando alertas tempranas sobre desviaciones significativas respecto a patrones esperados.

La arquitectura modular implementada proporciona una base sólida para estas evoluciones, permitiendo incorporar progresivamente nuevas capacidades sin necesidad de reescrituras sustanciales.

8.4. Reflexión personal

El desarrollo de DASOS ha constituido una experiencia altamente formativa, que me ha permitido integrar y aplicar conocimientos de múltiples disciplinas, desde análisis estadístico hasta

desarrollo web moderno. La confrontación con desafíos técnicos concretos ha puesto de manifiesto tanto la importancia de fundamentos teóricos sólidos como la necesidad de pragmatismo en su aplicación práctica.

Un aprendizaje particularmente valioso ha sido la experiencia de desarrollar un sistema completo end-to-end, gestionando todas las capas desde la extracción de datos hasta la visualización interactiva. Esta visión integral ha proporcionado una comprensión más profunda de las interdependencias entre componentes y de la importancia de interfaces bien definidas.

En el plano metodológico, el enfoque iterativo adoptado ha demostrado su valía, permitiendo refinar progresivamente tanto funcionalidades como interfaces en respuesta a necesidades identificadas durante el desarrollo. Esta aproximación contrasta favorablemente con metodologías más secuenciales que podrían haber resultado excesivamente rígidas ante los descubrimientos realizados durante la implementación.

Desde una perspectiva personal, el proyecto ha reforzado mi interés en la intersección entre el análisis de datos y desarrollo de software, un área que considero particularmente prometedora para aplicaciones en entornos educativos. La capacidad de transformar datos en conocimiento accionable mediante herramientas accesibles representa un vector de innovación con potencial significativo para mejorar procesos académicos, reflejando el valor que sistemas como el SGIC o los planes de Calidad tienen en la mejora de la calidad educativa.

También he tenido la gran oportunidad de aplicar patrones estilísticos y de diseño que siempre he querido añadir a un proyecto de este calibre, comprobando que las plataformas web no tienen que ser planas o feas para que sean prácticas, y viceversa, que se pueden desarrollar páginas y aplicaciones con un buen diseño a la vez que son funcionales, robustas y accesibles.

Esta es una de esas oportunidades en las que me he esforzado en cuerpo y alma porque veía un objetivo a ello: los estudiantes. Al final mi camino en esta Escuela y esta universidad ha sido (y será) ayudar y colaborar a que los estudiantes puedan disfrutar de la mejor vida universitaria (dentro y fuera del ámbito académico) que puedan vivir, así que este trabajo va para ellos.

En suma, DASOS no solo ha cumplido sus objetivos técnicos, sino que ha proporcionado un viaje formativo completo que ha consolidado conocimientos existentes y desarrollado nuevas capacidades en áreas de creciente relevancia profesional.

Bibliografía

[1] Pilar Cáceres Reche et al. «Learning analytics in higher education: a review of impact scientific literature». En: *IJERI: International Journal of Educational Research and Innovation* 13 (feb. de 2020), págs. 32-46. DOI: 10.46661/ijeri.4584. URL: https://www.upo.es/revistas/index.php/IJERI/article/view/4584.

[2] George Siemens. «Learning analytics: The emergence of a discipline». En: *American Behavioral Scientist* 57.10 (2013), págs. 1380-1400. DOI: 10.1177/0002764213498851. eprint: https://doi.org/10.1177/0002764213498851. URL: https://doi.org/10.1177/0002764213498851.

[3] Ryan SJD Baker y Kalina Yacef. «Data mining for education». En: *International encyclopedia of education* 7 (2011), págs. 112-118.

[4] Cristóbal Romero y Sebastián Ventura. «Educational data mining: A survey from 1995 to 2005». En: *Expert systems with applications* 33.1 (2007), págs. 135-146. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2006.04.005. URL: https://www.sciencedirect.com/science/article/pii/S0957417406001266.

[5] ANECA. *Guía de apoyo para la redacción, puesta en práctica y evaluación de los resultados del aprendizaje*. 2018. URL: http://www.aneca.es/content/download/12765/158329/file/guia_resu_aprendizaje.pdf (visitado 15-01-2024).

[6] Jorj X. McKie. *PyMuPDF Documentation*. 2024. URL: https://pymupdf.readthedocs.io/ (visitado 15-01-2024).

[7] SQLite Development Team. *SQLite Documentation*. 2024. URL: https://www.sqlite.org/docs.html (visitado 15-01-2024).

[8] Jay A Kreibich. *Using SQLite*. O'Reilly Media, 2010.

[9] Wes McKinney. «Data structures for statistical computing in python». En: *Proceedings of the 9th Python in Science Conference* 445 (2010), págs. 51-56.

[10] Charles R Harris et al. «Array programming with NumPy». En: *Nature* 585.7825 (2020), págs. 357-362.

[11] Pauli Virtanen et al. «SciPy 1.0: fundamental algorithms for scientific computing in Python». En: *Nature methods* 17.3 (2020), págs. 261-272.

[12] John D Hunter. «Matplotlib: A 2D graphics environment». En: *Computing in science & engineering* 9.3 (2007), págs. 90-95.

[13] Chart.js Contributors. *Chart.js Documentation*. 2024. URL: https://www.chartjs.org/docs/latest/ (visitado 15-01-2024).

[14] Mozilla Developer Network. *Introduction to client-side frameworks*. 2023. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Introduction (visitado 05-05-2024).

[15] Chris Kanich. *The Transition from Client-Side to Server-Side Rendering*. 2024. URL: https://484.cs.uic.edu/readings/chapter-3-server-side-web-development/full-stack-react/ (visitado 06-05-2024).

[16] Vercel. *Next.js Documentation*. 2024. URL: https://nextjs.org/docs (visitado 15-01-2024).
[17] Meta. *React Documentation*. 2024. URL: https://react.dev/ (visitado 15-01-2024).
[18] Roy Thomas Fielding. «Architectural styles and the design of network-based software architectures». Tesis doct. University of California, Irvine, 2000.
[19] Flask Development Team. *Flask Documentation*. 2024. URL: https://flask.palletsprojects.com/ (visitado 15-01-2024).
[20] Universidad Politecnica de Madrid. *La Plataforma web GAUSS de la UPM, finalista del "Premio de Buenas Prácticas en Gestión Universitaria 2017"*. 2017. URL: https://www.upm.es/UPM/SalaPrensa/Noticias?fmt=detail&prefmt=articulo&id=adbaf395509cf510VgnVCM10000009c7648a____.
[21] Vicerrectorado de Ordenación Académica y Planificación Estratégica Universidad Politécnica de Madrid. *Las Guías de Aprendizaje en la UPM*. Dic. de 2009. URL: https://innovacioneducativa.upm.es/sites/default/files/guias.pdf (visitado 01-12-2024).
[22] Rafael Aineto Guerrero. «Desarrollo de una aplicación web para la administración de la Plataforma de Gestión de los Procesos Internos de Calidad de la UPM (GAUSS)». Tesis de mtría. Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Informáticos, 2019. URL: https://oa.upm.es/55694/1/TFG_RAFAEL_AINETO_GUERRERO.pdf.
[23] Borja Martinena Cepa. «Danubit: Desarrollo de un sistema software web para la administración de asociaciones de la ETSIINF». Tesis de mtría. Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Informáticos, 2024. URL: https://oa.upm.es/83147/1/TFG_BORJA_MARTINENA_CEPA.pdf.
[24] The pandas development team. *pandas: powerful Python data analysis toolkit*. 2024. URL: https://pandas.pydata.org/docs/ (visitado 15-01-2024).
[25] Michael L Waskom. *seaborn: statistical data visualization*. 2024. URL: https://seaborn.pydata.org/ (visitado 15-01-2024).
[26] Maurice G Kendall. «A new measure of rank correlation». En: *Biometrika* 30.1/2 (1938), págs. 81-93.
[27] Karl Pearson. «Note on regression and inheritance in the case of two parents». En: *Proceedings of the royal society of London* 58.347-352 (1895), págs. 240-242.
[28] Martin Fowler. *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002.
[29] Vercel. *Deploying to Vercel*. 2022. URL: https://vercel.com/docs/deployments (visitado 05-05-2024).
[30] Railway. *Railway Documentation*. 2021. URL: https://docs.railway.com/ (visitado 05-05-2024).
[31] holger krekel holger y pytest-dev team. *pytest: helps you write better programs*. 2015. URL: https://docs.pytest.org/en/stable/ (visitado 05-05-2024).
[32] Cypress team. *Why Cypress?* 2025. URL: https://docs.cypress.io/app/get-started/why-cypress (visitado 05-05-2024).
[33] Club Excelencia en Gestión. *Plataforma Web GAUSS – UPM. Automatización de los Procesos del SGIC*. Premio a las Buenas Prácticas en el ámbito de la Excelencia en la Gestión Universitaria. Universidad Politécnica de Madrid. Feb. de 2018. URL: https://www.clubexcelencia.org/system/files/migrated/knowledge/documents/files/resumen_buena_practica_-_plataforma_web_gauss.pdf.
[34] Vincent Tinto. «Leaving college: Rethinking the causes and cures of student attrition». En: (1993).
[35] George D Kuh et al. «Unmasking the effects of student engagement on first-year college grades and persistence». En: *The journal of higher education* 79.5 (2008), págs. 540-563.

[36] Jeffrey EF Friedl. *Mastering regular expressions*. O'Reilly Media, Inc., 2006.

Apéndice A

Anexo

A.1. Manual de instalación y despliegue

Esta sección describe los pasos necesarios para instalar y desplegar el sistema DASOS, tanto para entornos de desarrollo local como para despliegue en producción mediante plataformas cloud modernas.

A.1.1. Requisitos previos

Para instalar y ejecutar DASOS, se requieren los siguientes componentes:

A.1.1.1. Desarrollo local

- **Python 3.9+**: El backend y componentes analíticos requieren Python 3.9 o superior.
- **Node.js 18+**: El frontend requiere Node.js versión 18 o superior y npm/pnpm.
- **Git**: Para control de versiones y clonado de repositorios.
- **Sistema operativo**: Compatible con Windows, macOS y Linux.
- **Espacio en disco**: Mínimo 1GB para la aplicación y sus dependencias.
- **Memoria**: Mínimo 4GB de RAM recomendados para funcionamiento óptimo.

A.1.1.2. Despliegue en producción

- **Cuenta GitHub**: Para alojar los repositorios del código fuente.
- **Cuenta Vercel**: Para despliegue del frontend Next.js.
- **Cuenta Railway**: Para despliegue del backend API y base de datos.
- **Navegador web**: Para acceso a los paneles de administración.

A.1.2. Instalación para desarrollo local

A.1.2.1. Configuración del backend

1. **Clonar el repositorio del backend**:

```
git clone https://github.com/usuario/academic-api.git
cd academic-api
```

2. **Crear un entorno virtual**:

```
python -m venv venv
# En Windows
venv\Scripts\activate
# En macOS/Linux
source venv/bin/activate
```

3. **Instalar dependencias**:

```
pip install -r requirements.txt
```

4. **Configurar variables de entorno**:

```
# Crear archivo .env
echo "FLASK_ENV=development" > .env
echo "DATABASE_PATH=academic_data.db" >> .env
```

5. **Verificar la base de datos**:

```
python -c "from academic_database import create_database;
db = create_database(); print('Database OK'); db.close()"
```

6. **Ejecutar el servidor de desarrollo**:

```
python rest-api.py --host localhost --port 8000
```

A.1.2.2. Configuración del frontend

1. **Clonar el repositorio del frontend**:

```
git clone https://github.com/usuario/dasos-upm.git
cd dasos-upm
```

2. **Instalar dependencias**:

```
npm install
# O usando pnpm (recomendado)
pnpm install
```

3. **Configurar variables de entorno**:

```
# Crear archivo .env.local
echo "API_BASE_URL=http://localhost:8000" > .env.local
echo "NEXT_PUBLIC_API_URL=http://localhost:8000" >> .env.local
```

4. **Ejecutar el servidor de desarrollo**:

```
npm run dev
# O usando pnpm
pnpm dev
```

5. **Acceder a la aplicación**: Abrir navegador en `http://localhost:3000`

A.1.3. Despliegue en producción cloud

A.1.3.1. Preparación de repositorios

1. **Crear repositorios separados en GitHub**:
   - Repositorio principal: Frontend Next.js
   - Repositorio API: Backend Flask con base de datos
2. **Configurar archivos de despliegue para el backend**:

```
# Crear Procfile
echo "web: python rest-api.py --host 0.0.0.0 --port \$PORT" > Procfile

# Verificar requirements.txt incluye:
Flask==3.0.0
Flask-CORS==4.0.0
pandas==2.1.4
numpy==1.26.2
psycopg2-binary==2.9.9
waitress==3.0.0
```

3. **Incluir base de datos en repositorio API**:

```
# Copiar archivo de base de datos al repositorio
cp academic_data.db ./
git add academic_data.db
git commit -m "Add database file"
```

A.1.3.2. Despliegue del backend en Railway

1. **Configurar Railway**:
   - Acceder a `railway.app` y crear cuenta
   - Seleccionar "New Project"→ "Deploy from GitHub repo"
   - Conectar repositorio del backend API
2. **Configurar variables de entorno en Railway**:

```
FLASK_ENV=production
DATABASE_PATH=academic_data.db
```

3. **Verificar despliegue**:

- Railway generará una URL automáticamente
- Verificar endpoint de salud: `https://your-api.railway.app/health`
- Probar endpoint de estadísticas: `https://your-api.railway.app/api/v1/stats`

A.1.3.3. Despliegue del frontend en Vercel

1. **Configurar Vercel**:
   - Acceder a `vercel.com` y crear cuenta
   - Importar repositorio principal desde GitHub
   - Vercel detectará automáticamente Next.js
2. **Configurar variables de entorno en Vercel**:

```
API_BASE_URL=https://your-api.railway.app
NEXT_PUBLIC_API_URL=https://your-api.railway.app
```

3. **Configurar build settings**:
   - Framework Preset: Next.js
   - Build Command: `npm run build`
   - Output Directory: `.next`
   - Install Command: `npm install`
4. **Verificar despliegue**:
   - Vercel proporcionará URL de producción
   - Verificar funcionamiento completo de la aplicación
   - Comprobar transiciones de página y animaciones

A.1.4. Configuración de CORS

Para el correcto funcionamiento entre frontend y backend, actualizar configuración CORS:

1. **Actualizar rest-api.py**:

```
# En la sección de CORS, añadir URL de Vercel
allowed_origins = [
"https://*.vercel.app",
"https://your-frontend.vercel.app"
]
```

2. **Commit y push para redeployment automático**:

```
git add rest-api.py
git commit -m "Update CORS for production frontend"
git push
```

A.1.5. Monitorización y mantenimiento

A.1.5.1. Verificación del sistema

1. **Health checks automáticos**:
   - Railway: Health checks cada 30 segundos
   - Vercel: Monitoring de uptime automático
2. **Logs y debugging**:

```
# Ver logs en Railway
railway logs --follow

# Ver logs de build en Vercel
# Accessible desde dashboard web
```

3. **Testing de endpoints principales**:

```
# Test API health
curl https://your-api.railway.app/health

# Test frontend
curl https://your-frontend.vercel.app
```

A.1.5.2. Actualización y mantenimiento

1. **Despliegue continuo**:
   - Push a rama main activa despliegue automático
   - Preview deployments en Vercel para testing
   - Rollback automático en caso de fallos
2. **Backup de datos**:
   - Base de datos versionada mediante Git
   - Historial completo de cambios disponible
   - Restauración mediante checkout a commit anterior
3. **Escalabilidad futura**:
   - Código preparado para migración a PostgreSQL
   - Scripts de migración automática incluidos
   - Configuración por variables de entorno

A.1.6. Solución de problemas comunes

A.1.6.1. Errores de desarrollo local

- **Error de conexión API**: Verificar que backend esté ejecutándose en puerto 8000

- **Error de dependencias**: Ejecutar `pip install -r requirements.txt` y `npm install`
- **Error de base de datos**: Verificar que `academic_data.db` exista en directorio backend

A.1.6.2. Errores de producción

- **CORS errors**: Verificar configuración de `allowed_origins` en `rest-api.py`
- **API no responde**: Verificar health check en Railway dashboard
- **Frontend no carga**: Verificar variables de entorno en Vercel settings
- **Build failures**: Revisar logs de deployment en respectivas plataformas

A.1.6.3. Contacto y soporte

- **Logs de Railway**: Accesibles desde dashboard web
- **Logs de Vercel**: Disponibles en sección "Functions"del proyecto
- **Repositorios GitHub**: Issues y documentación técnica
- **Community support**: Documentación oficial de Railway y Vercel

A.2. Ejemplos de uso de la API

Esta sección proporciona ejemplos detallados de uso de la API REST desarrollada para GaussUPM, ilustrando cómo realizar consultas comunes y utilizar los datos devueltos.

A.2.1. Consultar lista de asignaturas

Solicitud HTTP:

```
GET /api/v1/subjects
Con filtros
GET /api/v1/subjects?academic_year=2023-24&semester=Segundo
```

Respuesta:

```
[
{
"subject_code": "105000005",
"subject_name": "Cálculo",
"credits": 6,
"academic_year": "2023-24",
"semester": "Segundo"
},
{
"subject_code": "105000007",
"subject_name": "Probabilidades y Estadística I",
"credits": 6,
"academic_year": "2023-24",
"semester": "Segundo"
},
...
```

]

A.2.2. Obtener detalles de una asignatura

Solicitud HTTP:

```
GET /api/v1/subjects/105000005
```

Respuesta:


```
{
"subject_code": "105000005",
"subject_name": "Cálculo",
"credits": 6,
"academic_year": "2023-24",
"semester": "Segundo",
"total_enrolled": 455,
"first_time": 248,
"partial_dedication": 0,
"performance_rate": 31.94,
"success_rate": 36.52,
"absenteeism_rate": 12.56
}
```


A.2.3. Consultar datos históricos de una asignatura

Solicitud HTTP:

```
GET /api/v1/subjects/105000005/historical
Filtrar por tipo de tasa
GET /api/v1/subjects/105000005/historical?rate_type=rendimiento
```

Respuesta:


```
[
{
"subject_code": "105000005",
"academic_year": "2020-21",
"rate_type": "rendimiento",
"value": 31.32
},
{
"subject_code": "105000005",
"academic_year": "2021-22",
"rate_type": "rendimiento",
"value": 35.12
},
{
"subject_code": "105000005",
"academic_year": "2022-23",
"rate_type": "rendimiento",
"value": 32.49
},
```

```
{
"subject_code": "105000005",
"academic_year": "2023-24",
"rate_type": "rendimiento",
"value": 31.94
}
]
```


A.2.4. Consultar cambios en profesorado

Solicitud HTTP:

```
GET /api/v1/faculty/changes?subject_code=105000005
```

Respuesta:


```
[
{
"subject_code": "105000005",
"subject_name": "Cálculo",
"year1": "2021-22",
"year2": "2022-23",
"faculty_added": 2,
"faculty_removed": 1,
"percent_changed": 33.33
},
{
"subject_code": "105000005",
"subject_name": "Cálculo",
"year1": "2022-23",
"year2": "2023-24",
"faculty_added": 1,
"faculty_removed": 0,
"percent_changed": 14.28
}
]
```


A.2.5. Obtener insights para una asignatura

Solicitud HTTP:

```
GET /api/v1/insights/subjects?subject_code=105000005
```

Respuesta:


```
[
{
"subject_code": "105000005",
"analysis_id": 1,
"subject_name": "Cálculo",
"avg_performance_change": -0.55,
"trend_direction": "declining",
"faculty_impact_type": "negative",
```

```
"evaluation_impact_type": "neutral",
"insights_json": "...",
"insights_data": {
"performance_analysis": "La asignatura muestra una tendencia de declive ligero en rendimient
"faculty_impact": "Los cambios recientes en el profesorado
no han contribuido positivamente al rendimiento",
"recommendations": [
"Establecer mayor continuidad en el equipo docente",
"Revisar los métodos de enseñanza y adaptarlos al perfil de los estudiantes"
]
}
}
]
```

A.2.6. Ejemplo con Python

El siguiente fragmento de código ilustra cómo consumir la API desde Python utilizando la biblioteca requests:

```
import requests
import json
API_BASE_URL = "http://localhost:5000/api/v1"
def get_subject_details(subject_code):
"""Obtiene detalles de una asignatura específica"""
url = f"{API_BASE_URL}/subjects/{subject_code}"
response = requests.get(url)
if response.status_code == 200:
    return response.json()
else:
    print(f"Error: {response.status_code}")
    return None
def get_historical_data(subject_code, rate_type=None):
"""Obtiene datos históricos para una asignatura"""
url = f"{API_BASE_URL}/subjects/{subject_code}/historical"
params = {}
if rate_type:
params['rate_type'] = rate_type
response = requests.get(url, params=params)

if response.status_code == 200:
    return response.json()
else:
    print(f"Error: {response.status_code}")
    return None
Ejemplo de uso
subject_code = "105000005"
subject = get_subject_details(subject_code)
print(f"Detalles de {subject['subject_name']}:")
print(f"Tasa de rendimiento: {subject['performance_rate']}%")
print(f"Tasa de éxito: {subject['success_rate']}%")
```

```
print(f"Tasa de absentismo: {subject['absenteeism_rate']}%")
historical_data = get_historical_data(subject_code, rate_type="rendimiento")
print("\nHistórico de tasas de rendimiento:")
for record in historical_data:
print(f"{record['academic_year']}: {record['value']}%")
```

A.2.7. Ejemplo con JavaScript (Frontend)

El siguiente fragmento muestra cómo consumir la API desde React/Next.js:

```
// api.js - Funciones para consumo de API
import axios from 'axios';
const API_BASE_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:5000/api/v1';
export const getSubjects = async (academicYear, semester) => {
try {
const params = {};
if (academicYear) params.academic_year = academicYear;
if (semester) params.semester = semester;
const response = await axios.get(`${API_BASE_URL}/subjects`, { params });
return response.data;
} catch (error) {
console.error('Error fetching subjects:', error);
return [];
}
};
export const getSubjectDetails = async (subjectCode) => {
try {
const response = await axios.get(${API_BASE_URL}/subjects/${subjectCode});
return response.data;
} catch (error) {
console.error(Error fetching subject ${subjectCode}:, error);
return null;
}
};
// SubjectDetails.jsx - Componente React
import React, { useEffect, useState } from 'react';
import { getSubjectDetails } from '../api';
import PerformanceChart from './PerformanceChart';
const SubjectDetails = ({ subjectId }) => {
const [subject, setSubject] = useState(null);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
useEffect(() => {
const fetchData = async () => {
try {
setLoading(true);
const data = await getSubjectDetails(subjectId);
setSubject(data);
} catch (err) {
setError('Error fetching subject details');
```

```
console.error(err);
} finally {
setLoading(false);
}
};
if (subjectId) {
  fetchData();
}
}, [subjectId]);
if (loading) return <div>Loading...</div>;
if (error) return <div>Error: {error}</div>;
if (!subject) return <div>No subject data found</div>;
return (
<div className="bg-white bg-opacity-10 p-6 rounded-lg">
<h2 className="text-2xl font-bold mb-4">{subject.subject_name}</h2>
<div className="grid grid-cols-1 md:grid-cols-3 gap-4 mb-6">
<div className="p-4 bg-white bg-opacity-5 rounded-lg">
<h3 className="text-sm font-medium text-purple-200">Tasa de rendimiento</h3>
<p className="text-2xl font-bold">{subject.performance_rate}%</p>
</div>
<div className="p-4 bg-white bg-opacity-5 rounded-lg">
<h3 className="text-sm font-medium text-purple-200">Tasa de éxito</h3>
<p className="text-2xl font-bold">{subject.success_rate}%</p>
</div>
<div className="p-4 bg-white bg-opacity-5 rounded-lg">
<h3 className="text-sm font-medium text-purple-200">Tasa de absentismo</h3>
<p className="text-2xl font-bold">{subject.absenteeism_rate}%</p>
</div>
</div>
  {/* Aquí se incluiría el componente de gráfico */}
  <PerformanceChart subjectId={subjectId} />
</div>
);
};
export default SubjectDetails;
```

A.3. Código ejemplo

Esta sección presenta fragmentos de código representativos de diferentes componentes del sistema para ilustrar las técnicas y aproximaciones implementadas.

A.3.1. Extracción de texto de informes PDF

Fragmento de la función para extracción de texto utilizando PyMuPDF:

```
def extract_text_from_pdf(pdf_path):
"""Extraer texto completo de un PDF manteniendo estructura básica"""
text = ""
try:
with fitz.open(pdf_path) as pdf_document:
```

```
for page_number in range(len(pdf_document)):
page = pdf_document[page_number]
text += page.get_text() + "\n"
return text
except Exception as e:
print(f"Error extrayendo texto del archivo PDF: {e}")
return None
```

A.3.2. Extracción de tasas de rendimiento con expresiones regulares

Fragmento del método para extraer tasas de rendimiento de informes utilizando expresiones regulares:

```
def extract_performance_rates(self):
"""Extraer tasas de rendimiento, éxito y absentismo del curso actual"""
rates_section = re.search(
r'A2.1. Tasas de resultados académicos obtenidas en el curso objeto del Informe'
r'(.*?)A2.2. Tasas de resultados académicos obtenidas en cursos anteriores',
self.text_content, re.DOTALL
)
if rates_section:
    section_text = rates_section.group(1)

    # Expresión regular para extraer tasas de rendimiento
    pattern = r'(\d{9})\s*-\s*([^\n]+)\s*(\d+\.\d+)\s*(\d+\.\d+)\s*(\d+\.\d+)'

    for match in re.finditer(pattern, section_text):
        subject_code = match.group(1)
        subject_name = match.group(2).strip()
        performance_rate = float(match.group(3))
        success_rate = float(match.group(4))
        absenteeism_rate = float(match.group(5))

        if subject_code in self.courses_data:
            self.courses_data[subject_code]["performance_rate"] = performance_rate
            self.courses_data[subject_code]["success_rate"] = success_rate
            self.courses_data[subject_code]["absenteeism_rate"] = absenteeism_rate
```

A.3.3. Análisis de cambios en profesorado

Fragmento del método para analizar cambios en el profesorado entre períodos académicos:

```
def analyze_faculty_changes(self, subject_data):
"""Analizar cambios en el profesorado entre años académicos"""
years = sorted(subject_data.keys())
result = {"years_compared": [], "faculty_changes": {}}
# Se necesitan al menos dos años para comparar
if len(years) < 2:
    return result

for i in range(len(years) - 1):
```

```
    year1 = years[i]
    year2 = years[i+1]

    # Extraer conjuntos de profesores
    faculty1 = set()
    faculty2 = set()

    # Extraer nombres de profesores del primer año
    if "profesores" in subject_data[year1]:
        for prof in subject_data[year1]["profesores"]:
            if "nombre" in prof:
                faculty1.add(prof["nombre"])

    # Extraer nombres de profesores del segundo año
    if "profesores" in subject_data[year2]:
        for prof in subject_data[year2]["profesores"]:
            if "nombre" in prof:
                faculty2.add(prof["nombre"])

    # Calcular diferencias
    added = faculty2 - faculty1
    removed = faculty1 - faculty2

    result["years_compared"].append((year1, year2))
    result["faculty_changes"][(year1, year2)] = {
        "added": list(added),
        "removed": list(removed),
        "total_added": len(added),
        "total_removed": len(removed),
        "percent_changed": (len(added) + len(removed)) / max(1, len(faculty1)) * 100
    }

return result
```

A.3.4. Implementación de API REST con Flask

Fragmento de código para implementación de endpoint de API REST utilizando Flask:

```
@app.route(f'{API_PREFIX}/subjects/<subject_code>/historical', methods=['GET'])
def get_subject_historical(subject_code):
"""
Get historical performance data for a specific subject
Parameters:
- subject_code: The code of the subject

Query Parameters:
- rate_type: Filter by rate type (e.g., "rendimiento", "éxito", "absentismo")

Returns:
- JSON array with historical performance data
```

```
"""
rate_type = request.args.get('rate_type')

query = """
SELECT subject_code, academic_year, rate_type, value
FROM historical_rates
WHERE subject_code = ?
"""

params = [subject_code]

if rate_type:
    query += " AND rate_type = ?"
    params.append(rate_type)

query += " ORDER BY academic_year"

results = query_db(query, params)

if not results:
    abort(404, description=f"Historical data for subject {subject_code} not found")

return jsonify(results)
```

A.3.5. Componente React para visualización de rendimiento

Fragmento de implementación de componente React para visualización de tendencias de rendimiento:

```
"use client";
import { useEffect, useRef } from 'react';
import { Chart, ChartData, ChartOptions } from 'chart.js/auto';
interface PerformanceChartProps {
historicalData: any[];
title?: string;
height?: number;
}
export default function PerformanceChart({
historicalData,
title = "Histórico de Rendimiento",
height = 300
}: PerformanceChartProps) {
const chartRef = useRef<HTMLCanvasElement | null>(null);
const chartInstance = useRef<Chart | null>(null);
useEffect(() => {
// Skip if no data or canvas
if (!historicalData.length || !chartRef.current) return;
// Group data by rate type
const groupedData: { [key: string]: { year: string; value: number }[] } = {};
```

```
historicalData.forEach(item => {
  if (!groupedData[item.rate_type]) {
    groupedData[item.rate_type] = [];
  }
  groupedData[item.rate_type].push({
    year: item.academic_year,
    value: item.value
  });
});

// Sort data by year
Object.keys(groupedData).forEach(key => {
  groupedData[key].sort((a, b) => a.year.localeCompare(b.year));
});

// Get unique years
const years = Array.from(new Set(historicalData.map(item => item.academic_year))).sort();

// Create datasets for the chart
const datasets = Object.keys(groupedData).map((rateType, index) => {
  // Define colors for each rate type
  const colors: { [key: string]: string } = {
    'rendimiento': 'rgba(75, 192, 192, 1)',
    'éxito': 'rgba(54, 162, 235, 1)',
    'absentismo': 'rgba(255, 99, 132, 1)'
  };

  const backgroundColor = colors[rateType] ||
      `hsl(${index * 137.5}, 70%, 60%)`;

  // Create a full dataset with values for each year
  const data = years.map(year => {
    const dataPoint = groupedData[rateType].find(
        item => item.year === year);
    return dataPoint ? dataPoint.value : null;
  });

  // Translate rate types to Spanish
  const rateLabels: { [key: string]: string } = {
    'rendimiento': 'Rendimiento',
    'éxito': 'Éxito',
    'absentismo': 'Absentismo'
  };

  return {
    label: rateLabels[rateType] || rateType,
    data,
    backgroundColor,
    borderColor: backgroundColor,
```

```
    borderWidth: 2,
    tension: 0.3,
    fill: false
  };
});

// Destroy previous chart if it exists
if (chartInstance.current) {
  chartInstance.current.destroy();
}

// Create the chart
const ctx = chartRef.current?.getContext('2d');
if (ctx) {
  const chartData: ChartData = {
    labels: years,
    datasets
  };

  const chartOptions: ChartOptions = {
    responsive: true,
    maintainAspectRatio: false,
    scales: {
      y: {
        beginAtZero: true,
        max: 100,
        grid: {
          color: 'rgba(255, 255, 255, 0.1)'
        },
        ticks: {
          color: 'rgba(0, 0, 0, 0.7)'
        }
      },
      x: {
        grid: {
          color: 'rgba(255, 255, 255, 0.1)'
        },
        ticks: {
          color: 'rgba(0, 0, 0, 0.7)'
        }
      }
    },
    plugins: {
      legend: {
        position: 'top',
        labels: {
          color: 'rgba(0, 0, 0, 0.7)'
        }
      },
```

```
      tooltip: {
        mode: 'index',
        intersect: false,
        backgroundColor: 'rgba(0, 0, 0, 0.7)'
      }
    }
  };

  chartInstance.current = new Chart(ctx, {
    type: 'line',
    data: chartData,
    options: chartOptions
  });
}
}, [historicalData]);
return (
<div className="bg-white bg-opacity-10 backdrop-blur-lg rounded-lg p-4
                 shadow-lg border border-purple-300 border-opacity-20">
<h3 className="text-lg font-medium text-purple-900 mb-4">{title}</h3>
<div style={{ height: ${height}px }}>
<canvas ref={chartRef}></canvas>
</div>
</div>
);
}
```

A.4. Pruebas y resultados

Fragmentos de las pruebas realizadas para la validación del proyecto y sus resultados:

A.4.1. Interfaz

A.4.1.1. Pruebas Cypress

```
...user-flows.cy.ts

.
.
.
/// <reference types="cypress" />

describe('DASOS User Flows', () => {
 beforeEach(() => {
  // Visit the application
  cy.visit('/');
 });

 describe('Authentication Flow', () => {
  it('should redirect to login when not authenticated', () => {
   cy.url().should('include', '/login');
```

```
   cy.contains('DASOS UPM').should('be.visible');
   cy.contains('Inicia sesión').should('be.visible');
  });

  it('should login with valid credentials', () => {
   cy.url().should('include', '/login');

   // Fill login form
   cy.get('input[type="email"]').type('student@example.com');
   cy.get('input[type="password"]').type('password123');

   // Submit form
   cy.get('button[type="submit"]').click();

   // Should redirect to dashboard
   cy.url().should('include', '/dashboard');
   cy.contains('Dashboard Académico').should('be.visible');
  });

  //it('should show error with invalid credentials', () => {
   //  cy.get('input[type="email"]').type('invalid-email');
   //  cy.get('input[type="password"]').type('password');
   //  cy.get('button[type="submit"]').click();

   // Should show error message
   //  cy.get('[role="alert"]').should('be.visible');
   //  cy.url().should('include', '/login');
   //});

  it('should logout successfully', () => {
   // Login first
   cy.login('student@example.com', 'password123');

   // Click logout
   cy.get('[aria-label="Cerrar Sesión"]').click();

   // Should redirect to login
   cy.url().should('include', '/login');
  });
 });
.
.
.
```

A.4.1.2. Resultados pruebas

```
~/next-app$ npx cypress run --key 9b403fb4-c17d-4deb-8012-825217333f46


================================================================================================
```

```
(Run Starting)

----------------------------------------------------------------------------------------------------

Running:  accesibility.cy.ts                                                                (1

Accessibility Tests
WCAG 2.1 Level AA Compliance
OK should meet accessibility standards on login page (1290ms)
(Attempt 1 of 3) should meet accessibility standards on dashboard

Running:  color.cy.ts                                                                       (2

Color accesibility Tests
Color Contrast
OK should have sufficient color contrast ratios (1311ms)
OK should not rely solely on color (5692ms)

2 passing (7s)

----------------------------------------------------------------------------------------------------

Running:  error-handling.cy.ts                                                              (3

DASOS error-handling Flows
Error Handling
OK should handle 404 pages (1925ms)
OK should handle API errors gracefully (8424ms)

2 passing (11s)

Running:  keyboard.cy.ts                                                                    (6

Keyboard Accessibility Tests
Keyboard Navigation
OK should have proper focus indicators (1059ms)
OK should support keyboard shortcuts (6214ms)

2 passing (7s)
```

```
.
.
.

Running:  navigation.cy.ts                                                (8


Basic Navigation Tests
OK should redirect to login when not authenticated (985ms)
OK should display login form elements (1100ms)
OK should login successfully with valid credentials (2464ms)
OK should navigate through main menu items (7531ms)
OK should search for subjects (9913ms)
OK should logout successfully (3952ms)


6 passing (26s)


Running:  responsive.cy.ts                                                (9


Responsive design and accesibility Tests
Responsive Accessibility
OK should maintain accessibility on mobile (1341ms)
OK should have accessible mobile navigation (3342ms)
Responsive Design
OK should work on mobile devices (4716ms)
OK should work on tablet devices (4635ms)


4 passing (15s)


Running:  screen-reader.cy.ts                                             (10


Screen-reader accesibility Tests
Screen Reader Support
OK should announce live regions properly (1543ms)
OK should have proper heading hierarchy (5989ms)


2 passing (8s)


Running:  user-flows.cy.ts                                                (11
```

```
DASOS User Flows
Authentication Flow
OK should redirect to login when not authenticated (1076ms)
OK should login with valid credentials (2998ms)
OK should logout successfully (5519ms)
Dashboard Navigation
OK should display dashboard with key metrics (5789ms)
OK should navigate to subjects page (4589ms)
OK should navigate to profile page (4129ms)
OK should navigate to settings page (8312ms)
Subject Search and Details
OK should search for subjects (6637ms)
OK should display subject performance data (7074ms)
OK should switch between analysis and info tabs (8385ms)
Subject Information Page
OK should display faculty information (9750ms)
OK should display evaluation methods (6080ms)
OK should display resources (6782ms)
Performance Metrics
OK should load dashboard quickly (3770ms)
OK should have good Lighthouse scores (996ms)


15 passing (1m)


================================================================================

(Run Finished)


Spec                                        Tests  Passing  Failing  Pending  Skipped
| NO accesibility.cy.ts                    00:38        3        1        2        -
| OK color.cy.ts                           00:07        2        2        -        -
| OK error-handling.cy.ts                  00:10        2        2        -        -
| OK focus.cy.ts                           00:07        2        2        -        -
| OK form.cy.ts                            00:08        4        4        -        -
| OK keyboard.cy.ts                        00:07        2        2        -        -
| OK media.cy.ts                           00:07        2        2        -        -
| OK navigation.cy.ts                      00:26        6        6        -        -
```

```
| OK responsive.cy.ts                    00:14       4       4       -       -

| OK screen-reader.cy.ts                 00:07       2       2       -       -

| OK user-flows.cy.ts                    01:23      15      15       -       -

NO 1 of 11 failed (9%)                  03:38      44      42       2       -
```

A.4.2. Backend

A.4.2.1. Pruebas Pytest

```
... test_security.py
.
.
.
    def test_authentication_security(self, client):
"""Test authentication security measures"""
# Test missing authentication
protected_endpoints = [
'/api/v1/insights/subjects',
'/api/v1/faculty/changes',
'/api/v1/evaluation/changes'
]

for endpoint in protected_endpoints:
response = client.get(endpoint)
# These endpoints should be accessible but could require auth in production
assert response.status_code != 500, f"Server error on {endpoint}"

# Test invalid tokens
headers = {'Authorization': 'Bearer invalid_token_12345'}
response = client.get('/api/v1/subjects', headers=headers)
assert response.status_code != 500, "Server error with invalid token"

def test_rate_limiting(self, client):
"""Test rate limiting protection"""
# Make many requests quickly
endpoint = '/api/v1/subjects'
responses = []

for _ in range(100):
response = client.get(endpoint)
responses.append(response.status_code)

# Should see rate limiting kick in (429 status codes)
# Note: This requires rate limiting to be implemented
# assert 429 in responses, "No rate limiting detected"

def test_cors_headers(self, client):
```

```
"""Test CORS configuration"""
response = client.get('/api/v1/subjects')

# Check CORS headers are present
assert 'Access-Control-Allow-Origin' in response.headers

# Verify CORS is not too permissive
origin = response.headers.get('Access-Control-Allow-Origin')
assert origin != '*', "CORS allows all origins - security risk"

def test_security_headers(self, client):
"""Test security headers are present"""
response = client.get('/')

# Check security headers
security_headers = {
 'X-Content-Type-Options': 'nosniff',
 'X-Frame-Options': ['DENY', 'SAMEORIGIN'],
 'X-XSS-Protection': '1; mode=block',
 'Strict-Transport-Security': 'max-age=31536000',
 'Content-Security-Policy': None  # Should be present
}

.
.
.
... test_academic_analysis.py

    def test_convert_to_dataframe(self, sample_data):
"""Test conversion to pandas DataFrame"""
analyzer = AcademicDataAnalyzer(sample_data)
df = analyzer.convert_to_dataframe()

assert isinstance(df, pd.DataFrame)
assert len(df) == 1
assert df.iloc[0]['name'] == 'Cálculo'
assert df.iloc[0]['performance_rate'] == 31.94

def test_historical_rates_to_dataframe(self, sample_data):
"""Test historical rates DataFrame conversion"""
analyzer = AcademicDataAnalyzer(sample_data)
hist_df = analyzer.historical_rates_to_dataframe()

assert isinstance(hist_df, pd.DataFrame)
assert len(hist_df) == 4  # 4 years of data
assert 'subject_code' in hist_df.columns
assert 'rate_type' in hist_df.columns
assert 'value' in hist_df.columns
```

.
.
.

A.4.2.2. Resultados

```
pytest tests/test_academic_analysis.py
========================================== test session starts ==========================
platform linux -- Python 3.12.3, pytest-8.4.0, pluggy-1.6.0
rootdir: /mnt/c/Users/prestamo_admin/Documents/academic_analysis
plugins: cov-6.1.1
collected 11 items

tests/test_academic_analysis.py ...........

========================================== warnings summary ==========================
academic_data_extractor.py:416
/mnt/c/Users/prestamo_admin/Documents/academic_analysis/academic_data_extractor.py:416: Synt
f.write("\CONCLUSIONES\n")

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
==================================== 11 passed, 1 warning in 6.64s ======================
```

A.5. Repositorio de código y Proyecto Desplegado

El código fuente completo del proyecto está disponible en los siguientes repositorios públicos:

Frontend: https://github.com/henny-hen/DASOS-UPM
Backend: https://github.com/henny-hen/DASOS-backend

El enlace al proyecto desplegado es el siguiente:

https://dasos-upm-e2ck.vercel.app/

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Thu Jul 03 10:31:28 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)