INTERFAZ WEB PARA MAUDE
WEB INTERFACE FOR MAUDE

TRABAJO FIN DE GRADO
CURSO 2023-2024

AUTOR
PEDRO JOSÉ PÉREZ VASCO

DIRECTOR
ADRIÁN RIESCO RODRÍGUEZ

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

INTERFAZ WEB PARA MAUDE
WEB INTERFACE FOR MAUDE

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTOR
PEDRO JOSÉ PÉREZ VASCO

DIRECTOR
ADRIÁN RIESCO RODRÍGUEZ

CONVOCATORIA: SEPTIEMBRE 2024

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

3 DE SEPTIEMBRE DE 2024

RESUMEN

Interfaz web para Maude

Este proyecto desarrolla una aplicación web con el objetivo de simplificar el uso de la biblioteca Maude, una herramienta poderosa pero compleja que se ejecuta exclusivamente en sistemas basados en Linux. Mediante el uso del *framework* Django y una biblioteca de bindings de Maude en Python, se ha creado una interfaz accesible para interactuar con esta tecnología, facilitando su uso en un entorno educativo. La aplicación permite analizar y adaptar los comandos de Maude para que puedan ser utilizados de forma intuitiva, apoyando la docencia del lenguaje. El sistema utiliza SQLite3 como base de datos para almacenar la información y se despliega en la plataforma PythonAnywhere, aprovechando su facilidad de configuración para ofrecer una solución práctica y eficiente. Este enfoque mejora la usabilidad para los usuarios, además de mejorar la accesibilidad y utilidad de Maude en el ámbito educativo.

Palabras sugeridas: Aplicación Web, Maude, Django, Python, Bindings de Maude, SQLite3, PythonAnywhere, Interfaz Educativa, Accesibilidad, Linux.

ABSTRACT

Web interface for Maude

This project develops a web application aimed at simplifying the use of the Maude library, a powerful but complex tool that runs exclusively on Linux-based systems. By utilizing the Django *framework* and a Maude bindings library in Python, an accessible interface has been created to interact with this technology, facilitating its use in an educational environment. The application allows for parsing and adapting Maude commands to be used intuitively, supporting the teaching of the language. The system uses SQLite3 as a database to store information and is deployed on the PythonAnywhere platform, taking advantage of its ease of configuration to offer a practical and efficient solution. This approach enhances usability for users, as well as improving the accessibility and usefulness of Maude in the educational field.

Suggested Keywords: Web Application, Maude, Django, Python, Maude Bindings, SQLite3, PythonAnywhere, Educational Interface, Accessibility, Linux

ÍNDICE DE CONTENIDOS

Capítulo 1 - Introducción

La planificación del proyecto se estructuró en torno a tres aspectos clave: la motivación detrás de su desarrollo, los objetivos que guiaron cada fase de implementación, y el plan de trabajo que se siguió para alcanzar dichos objetivos. A continuación, se detallan estos aspectos, proporcionando un marco claro y comprensivo para entender cómo se abordó el proyecto desde su concepción hasta su ejecución.

1.1 Motivación

Maude es una herramienta basada en lógica de reescritura, ampliamente utilizada para modelar sistemas complejos y realizar verificaciones formales. Sin embargo, su uso está predominantemente restringido a entornos Linux, lo que representa una barrera significativa para su adopción en contextos educativos. Los estudiantes que no están familiarizados con Linux enfrentan desafíos considerables para instalar, configurar y utilizar Maude, lo que limita su accesibilidad y su utilidad a entornos académicos donde la diversidad de plataformas es común.

Además de la restricción a un sistema operativo específico, Maude presenta una curva de aprendizaje pronunciada debido a la complejidad de su interfaz de línea de comandos. Esta complejidad exige un nivel avanzado de conocimientos técnicos, tanto en el manejo de terminales como en la comprensión de los fundamentos de la lógica de reescritura y la programación algebraica. Como resultado, muchos potenciales usuarios, especialmente aquellos con menos experiencia en estos campos, se sienten desmotivados. Esta situación no solo dificulta el aprendizaje autodidacta, sino que también complica la integración de Maude en planes de estudios formales, donde se prefieren herramientas más accesibles.

Dada la relevancia de Maude y las barreras mencionadas, se hace evidente la necesidad de desarrollar una solución que haga esta herramienta más accesible y fácil de usar. Un enfoque web, basado en una aplicación que funcione en múltiples plataformas, simplificaría considerablemente la interacción con Maude. Al ofrecer una interfaz gráfica intuitiva y amigable, esta solución permitiría a los usuarios interactuar con

Maude sin necesidad de dominar la línea de comandos ni preocuparse por las limitaciones del sistema operativo. Así, se eliminarían las barreras técnicas que actualmente limitan su adopción y se ampliaría su uso a un público más amplio.

La implementación de una herramienta que facilite el acceso y uso de Maude en un entorno educativo tendría un impacto significativo. Al permitir a estudiantes y profesores interactuar de manera más directa y sencilla con la herramienta, se podría fomentar un mayor interés en los conceptos avanzados que Maude permite explorar. Este nuevo enfoque no solo facilitaría la enseñanza y el aprendizaje de la lógica formal y la reescritura, sino que también integraría Maude en una gama más amplia de disciplinas académicas. Esto, a su vez, podría aumentar la comprensión y el dominio de estos conceptos entre los estudiantes, preparándolos mejor para enfrentar desafíos complejos en sus futuras carreras.

Este proyecto refleja mi interés en la docencia y en la tecnología, así como mi compromiso con la mejora de las herramientas educativas disponibles en el campo de los lenguajes formales y la lógica. Al desarrollar una solución que haga Maude más accesible y utilizable, no solo estoy contribuyendo a la comunidad académica, sino que también estoy avanzando en mi objetivo de simplificar el aprendizaje de conceptos complejos mediante el uso de tecnologías innovadoras. Este proyecto representa un paso significativo hacia la creación de recursos educativos que sean tanto eficaces como inclusivos, facilitando el aprendizaje para una nueva generación de estudiantes.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una interfaz web accesible y fácil de usar que facilite la interacción con Maude. Esta aplicación está diseñada para simplificar significativamente el uso de Maude en entornos educativos, permitiendo a los usuarios ejecutar comandos y gestionar módulos de manera intuitiva, sin la necesidad de conocimientos avanzados en la línea de comandos ni la instalación de la biblioteca en un entorno Linux. Al proporcionar una solución web completa, este proyecto busca hacer que el lenguaje Maude sea más accesible tanto para estudiantes como para profesores, reduciendo las barreras técnicas y promoviendo su adopción en la enseñanza y el aprendizaje.

Para lograr el objetivo principal, se han planteado los siguientes subobjetivos:

1. **Análisis sintáctico de comandos Maude:** Implementar un sistema de análisis que traduzca los comandos de Maude a un formato compatible con la biblioteca de bindings en Python, facilitando así la ejecución de comandos desde la interfaz web.
2. **Creación de chats para interacción modular:** Desarrollar un sistema de chats que permita a los usuarios guardar módulos de Maude, modificarlos, y probar comandos, manteniendo un registro de las interacciones y respuestas recibidas.
3. **Sistema de autenticación y gestión de usuarios:** Implementar un sistema de registro y *login* que requiera un correo electrónico UCM y una confirmación vía enlace. Además, incluir un proceso de recuperación de contraseña a través de un enlace enviado por correo.
4. **Market de módulos:** Crear un *market* donde los administradores puedan gestionar (crear, modificar, eliminar y activar) módulos disponibles para que los usuarios los descarguen o seleccionen como activos en sus chats.
5. **Gestión avanzada de módulos en chats:** Introducir la funcionalidad de versiones para módulos dentro de los chats, permitiendo guardar, seleccionar y comparar diferentes versiones de un módulo, similar a un control de versiones como GitHub.
6. **Sistema de entrega de comandos y módulos:** Añadir una funcionalidad que permita seleccionar comandos utilizados y enviarlos como una entrega junto con el módulo correspondiente. Esto facilita la evaluación de los trabajos por parte de los administradores.
7. **Gestión de entregas por usuarios:** Desarrollar una vista en la que los usuarios puedan ver sus entregas realizadas, filtrarlas por estado (corregida o no), y visualizar los comentarios de los administradores sobre las entregas corregidas.
8. **Sistema de corrección para administradores:** Crear una interfaz donde los administradores puedan ver y corregir las entregas de los usuarios, enviar notas y comentarios, y gestionar las entregas ya corregidas con opción de editar las correcciones.

9. **Funcionalidades avanzadas de búsqueda y filtrado:** Implementar buscadores y filtros en las vistas del *market*, tareas de alumnos, y entregas, permitiendo buscar por nombre, fecha de creación, y otros criterios relevantes.

1.3 Plan de trabajo

Una planificación cuidadosa es esencial para el éxito de cualquier proyecto de desarrollo de software. A continuación, se detalla la planificación seguida durante el desarrollo del proyecto, como podemos ver en la figura 1, desde la fase inicial de investigación hasta la implementación final y las pruebas.

Figura 1 - Cronograma de la planificación

1.3.1 Fase de Investigación Preliminar

La fase inicial del proyecto consistió en una investigación exhaustiva para entender los fundamentos y el contexto técnico del mismo. Dado que Maude es un lenguaje formal especializado, fue esencial adquirir un conocimiento sólido sobre sus características y comandos antes de proceder con la implementación.

- **Investigación sobre Maude**: Se comenzó por explorar qué es Maude, cuál es su propósito, y qué tipos de comandos son los más relevantes para implementar en este proyecto. Esta investigación incluyó la revisión de la documentación oficial, así como la consulta de recursos académicos y ejemplos de uso práctico.
- **Exploración de los *bindings* de Maude en Python**: Una vez comprendido el entorno de Maude, se pasó a investigar cómo este se puede integrar con Python. Se evaluaron los *bindings* disponibles para Python, identificando cómo traducir los comandos de Maude a un formato compatible que pueda ser ejecutado desde Python.

- **Estudio de Django**: Dado que el desarrollo del proyecto se realizaría utilizando Django, fue necesario adquirir conocimientos básicos y avanzados sobre este *framework*. A pesar de no haber trabajado previamente con Django, se invirtió tiempo en comprender su estructura, patrones de diseño y componentes clave, lo que sentó las bases para una implementación eficiente.

Duración estimada: 2 semanas.

1.3.2 Implementación Inicial

Con una base teórica sólida, la siguiente etapa fue iniciar el desarrollo del proyecto. Esta fase se centró en crear las primeras vistas y funcionalidades básicas de la aplicación.

- **Desarrollo de la Vista de Home**: La vista principal de la aplicación, que permite la ejecución de comandos de Maude y la visualización de sus resultados, fue la primera en ser implementada. Se diseñó y desarrolló esta vista para que los usuarios pudieran interactuar directamente con el sistema desde el inicio.
- **Pruebas de la Vista de Home**: Una vez completada la implementación inicial, se realizaron pruebas para asegurar que la funcionalidad básica estuviera operativa. Las pruebas consistieron en verificar que los comandos de Maude se ejecutaran correctamente y que los resultados se mostraran como se esperaba.

Duración estimada: 1.5 semanas.

1.3.3 Desarrollo de Funcionalidades Adicionales

Con la vista principal en funcionamiento, la atención se centró en expandir la funcionalidad de la aplicación mediante el desarrollo de otras vistas esenciales y características adicionales.

- **Desarrollo de la Vista de Registro y Login**: Se implementaron las funcionalidades de autenticación de usuarios, incluyendo el registro y el inicio de sesión. Esta fase fue crucial para gestionar la seguridad y personalización del acceso a la plataforma.
- **Desarrollo de la Vista de Recuperación de Contraseña**: Se añadió la funcionalidad que permite a los usuarios recuperar sus contraseñas en caso de olvido, mejorando la experiencia de usuario y la seguridad de la plataforma.
- **Desarrollo de la Vista de Market de Módulos**: Esta vista permite a los usuarios gestionar módulos de Maude. Se incluyó la capacidad de crear, modificar, activar/desactivar y eliminar módulos, ofreciendo un control detallado sobre los recursos disponibles en la plataforma.
- **Desarrollo de la Vista de Mis Tareas**: Se creó una interfaz donde los usuarios pueden visualizar y gestionar sus entregas de tareas, accediendo a detalles específicos de cada una.
- **Desarrollo de la Vista de Entregas Pendientes e Historial de Entregas Corregidas**: Estas vistas se diseñaron para facilitar el proceso de corrección de tareas por parte de los administradores, permitiéndoles gestionar entregas pendientes y revisar el historial de entregas corregidas.

Duración estimada: 4 semanas.

1.3.4 Pruebas y Validación

Una vez completada la implementación de las vistas y funcionalidades, se procedió a una fase exhaustiva de pruebas y validación para garantizar la calidad del software.

- **Pruebas Unitarias y de Integración**: Se ejecutaron pruebas unitarias en cada componente desarrollado, así como pruebas de integración para asegurar que todos los elementos de la aplicación funcionaran de manera conjunta y sin conflictos.

- **Corrección de Bugs**: Durante la fase de pruebas, se identificaron y corrigieron varios errores, asegurando la estabilidad y fiabilidad del sistema antes de su despliegue.
- **Ajustes Finales**: Basándose en los resultados de las pruebas y el *feedback* recibido, se realizaron ajustes finales en la aplicación para optimizar su funcionamiento y mejorar la experiencia de usuario.

Duración estimada: 2 semanas.

1.3.5 Gestión de Cambios e Imprevistos

Durante el desarrollo del proyecto, surgieron imprevistos que requirieron ajustes en la planificación original. Esta fase se centró en gestionar estos cambios de manera efectiva para minimizar su impacto en el cronograma general.

- **Implementación de la Funcionalidad de Modificación de Módulos**: Una de las modificaciones significativas fue la implementación de una nueva estructura de módulos con tres modos, como se detalló previamente. Esta funcionalidad se incorporó tras identificar la necesidad de ofrecer mayor flexibilidad en la gestión de los módulos.
- **Gestión de Cambios**: Se documentaron todos los cambios realizados, ajustando el cronograma y el alcance del proyecto según las nuevas necesidades o descubrimientos realizados durante el desarrollo.

Duración estimada: 1 semana.

1.4 Repositorio del Proyecto

Con el fin de asegurar la accesibilidad y transparencia del código fuente desarrollado en este proyecto, se ha utilizado GitHub para alojar el repositorio de código. En este repositorio[1] se encuentran todas las versiones del software, documentadas de manera sistemática a través de *commits* que reflejan cada etapa

[1] Repositorio de github del proyecto: https://github.com/Tarusito/web_maude

del desarrollo. El repositorio está disponible públicamente y contiene instrucciones para la instalación y uso de la aplicación, facilitando así su adopción y futura extensión por parte de otros desarrolladores e investigadores. Además, la aplicación se puede encontrar desplegada en esta página[2].

[2] Página web de la aplicación: https://maudeweb.pythonanywhere.com

Introduction

The project planning was structured around three key aspects: the motivation behind its development, the objectives that guided each implementation phase, and the work plan that was followed to achieve these objectives. These aspects are detailed below, providing a clear and comprehensive framework to understand how the project was approached from its conception to execution.

Motivation

Maude is a tool based on rewrite logic, widely used for modeling complex systems and performing formal verifications. However, its use is predominantly restricted to Linux environments, which represents a significant barrier to its adoption in educational contexts. Students who are not familiar with Linux face considerable challenges in installing, configuring, and using Maude, limiting its accessibility and usefulness in academic settings where platform diversity is common.

In addition to being limited to a specific operating system, Maude presents a steep learning curve due to the complexity of its command-line interface. This complexity requires an advanced level of technical knowledge, both in handling terminals and in understanding the fundamentals of rewrite logic and algebraic programming. As a result, many potential users, especially those with less experience in these fields, feel demotivated. This situation not only hinders self-directed learning but also complicates the integration of Maude into formal curricula, where more accessible tools are preferred.

Given the relevance of Maude and the aforementioned barriers, the need arises to develop a solution that makes this tool more accessible and easier to use. A web-based approach, utilizing an application that operates on multiple platforms, would considerably simplify interaction with Maude. By offering an intuitive and user-friendly graphical interface, this solution would allow users to interact with Maude without needing to master the command line or worry about operating system limitations. Thus, the technical barriers that currently limit its adoption would be removed, and its use would be expanded to a broader audience.

Implementing a tool that facilitates access to and use of Maude in an educational environment would have a significant impact. By allowing students and teachers to interact more directly and easily with the tool, greater interest could be fostered in the advanced concepts that Maude enables them to explore. This new approach would not only facilitate the teaching and learning of formal logic and rewriting but also integrate Maude into a wider range of academic disciplines. This, in turn, could increase understanding and mastery of these concepts among students, better preparing them to face complex challenges in their future careers.

This project reflects my interest in teaching and technology, as well as my commitment to improving the educational tools available in the field of formal languages and logic. By developing a solution that makes Maude more accessible and usable, I am not only contributing to the academic community but also advancing my goal of simplifying the learning of complex concepts through the use of innovative technologies. This project represents a significant step towards creating educational resources that are both effective and inclusive, facilitating learning for a new generation of students.

Objectives

The main objective of this project is to develop an accessible and user-friendly web interface that facilitates interaction with Maude. This application is designed to significantly simplify the use of Maude in educational environments, allowing users to execute commands and manage modules intuitively, without the need for advanced command-line knowledge or the installation of the library in a Linux environment. By providing a complete web solution, this project aims to make the Maude language more accessible to both students and teachers, reducing technical barriers and promoting its adoption in teaching and learning.

To achieve the main objective, the following sub-objectives have been outlined:

1- **Syntactic analysis of Maude commands**: Implement a system that translates Maude commands into a format compatible with the Python bindings library, thus facilitating the execution of commands from the web interface.

2- **Creation of chats for modular interaction**: Develop a chat system that allows users to save Maude modules, modify them, and test commands, while keeping a record of interactions and received responses.

3- **Authentication and user management system**: Implement a registration and login system that requires a UCM email and link confirmation. Additionally, include a password recovery process through a link sent via email.

4- **Module marketplace**: Create a marketplace where administrators can manage (create, modify, delete, and activate) available modules for users to download or select as active in their chats.

5- **Advanced module management in chats**: Introduce the functionality of version control for modules within chats, allowing users to save, select, and compare different versions of a module, similar to version control systems like GitHub.

6- **Command and module submission system**: Add functionality that allows users to select the commands they have used and submit them along with the corresponding module. This will facilitate the evaluation of work by administrators.

7- **User submission management**: Develop a view where users can see their submitted work, filter it by status (corrected or not), and view the administrators' comments on the corrected submissions.

8- **Administrator correction system**: Create an interface where administrators can view and correct user submissions, send grades and comments, and manage already corrected submissions with the option to edit corrections.

9- **Advanced search and filtering functionalities**: Implement search and filtering tools in the views for the marketplace, student tasks, and submissions, allowing searches by name, creation date, and other relevant criteria.

Work Plan

Careful planning is essential for the success of any software development project. Below, the planning followed throughout the project development is detailed, as seen in Figure 1, from the initial research phase to the final implementation and testing stages.

Figure 1 - Planning Schedule.

Preliminary Research Phase

The initial phase of the project consisted of extensive research to understand its foundations and technical context. Given that Maude is a specialized formal language, it was essential to acquire a solid understanding of its characteristics and commands before proceeding with the implementation.

- **Research on Maude**: The first step was to explore what Maude is, its purpose, and the types of commands most relevant for this project. This research included reviewing the official documentation, as well as consulting academic resources and practical usage examples.
- **Exploration of Maude bindings in Python**: Once familiar with Maude's environment, the next step was to investigate how it could be integrated with Python. Available bindings for Python were evaluated, identifying how to translate Maude commands into a compatible format that could be executed from Python.
- **Study of Django**: Since the project development was to be carried out using Django, it was necessary to acquire both basic and advanced knowledge of this framework. Despite not having worked with Django before, time was invested in understanding its structure, design patterns, and key components, which laid the groundwork for efficient implementation.

  Estimated duration: 2 weeks.

Initial Implementation

The initial phase of the project consisted of extensive research to understand its foundations and technical context. Given that Maude is a specialized formal language, it was essential to acquire a solid understanding of its characteristics and commands before proceeding with the implementation.

- **Research on Maude**: The first step was to explore what Maude is, its purpose, and the types of commands most relevant for this project. This research included reviewing the official documentation, as well as consulting academic resources and practical usage examples.
- **Exploration of Maude bindings in Python**: Once familiar with Maude's environment, the next step was to investigate how it could be integrated with Python. Available bindings for Python were evaluated, identifying how to translate Maude commands into a compatible format that could be executed from Python.
- **Study of Django**: Since the project development was to be carried out using Django, it was necessary to acquire both basic and advanced knowledge of this framework. Despite not having worked with Django before, time was invested in understanding its structure, design patterns, and key components, which laid the groundwork for efficient implementation.

  Estimated duration: 2 weeks.

Development of Additional Features

With the main view functioning, the focus shifted to expanding the application's functionality by developing other essential views and additional features.

- **Development of the Registration and Login View**: User authentication functionalities, including registration and login, were implemented. This phase was crucial for managing security and personalizing access to the platform.

- **Development of the Password Recovery View**: A feature was added that allows users to recover their passwords in case of loss, improving both user experience and platform security.
- **Development of the Module Market View**: This view allows users to manage Maude modules. It included the ability to create, modify, activate/deactivate, and delete modules, providing detailed control over the resources available on the platform.
- **Development of the My Tasks View**: An interface was created where users can view and manage their task submissions, accessing specific details of each.
- **Development of the Pending Deliveries and Corrected Deliveries History View**: These views were designed to facilitate the task correction process for administrators, allowing them to manage pending submissions and review the history of corrected ones.

  Estimated duration: 4 weeks.

Testing and Validation

Once the implementation of the views and functionalities was completed, an exhaustive testing and validation phase was carried out to ensure the software's quality.

- **Unit and Integration Testing**: Unit tests were performed on each developed component, as well as integration tests to ensure that all elements of the application worked together smoothly and without conflicts.
- **Bug Fixing**: During the testing phase, several errors were identified and corrected, ensuring the system's stability and reliability before deployment.
- **Final Adjustments**: Based on the test results and received feedback, final adjustments were made to the application to optimize its performance and enhance the user experience.

  Estimated duration: 2 weeks.

Change Management and Unexpected Issues

During the project's development, unexpected issues arose that required adjustments to the original plan. This phase focused on managing these changes effectively to minimize their impact on the overall schedule.

- **Implementation of Module Modification Functionality**: One of the significant changes was the introduction of a new module structure with three modes, as previously detailed. This functionality was incorporated after identifying the need to provide greater flexibility in module management.
- **Change Management**: All changes made were documented, and the project's timeline and scope were adjusted according to the new requirements or discoveries made during development.

  Estimated duration: 1 week.

Project Repository

In order to ensure the accessibility and transparency of the source code developed in this project, GitHub was used to host the code repository[3]. This repository contains all versions of the software, systematically documented through commits that reflect each stage of development. The repository is publicly available and includes instructions for installing and using the application, thereby facilitating its adoption and future extension by other developers and researchers. Additionally, the application can be found deployed at this page[4].

[3] Repositorio de github del proyecto: https://github.com/Tarusito/web_maude

[4] Página web de la aplicación: https://maudeweb.pythonanywhere.com

Capítulo 2 - Estado de la cuestión

En este capítulo se presentarán las tecnologías fundamentales que han sido empleadas a lo largo del desarrollo del proyecto. Estas herramientas, desde el lenguaje Maude hasta plataformas de despliegue en la nube como PythonAnywhere, han permitido la creación de una aplicación web robusta y accesible. Además, se explorará cómo estas tecnologías han sido integradas y utilizadas para alcanzar los objetivos del proyecto, así como su papel en la mejora de la interacción con Maude en entornos educativos. Finalmente, se realizará un análisis de aplicaciones similares, proporcionando un contexto más amplio sobre el posicionamiento y las innovaciones del proyecto.

2.1 Tecnologías

A continuación, se hablará de las tecnologías que han sido utilizadas a lo largo del desarrollo del proyecto.

2.1.1 Maude

Maude[5] es un lenguaje de programación especializado que utiliza lógica ecuacional y de reescritura para describir y simular sistemas complejos. Este lenguaje permite modelar el comportamiento de sistemas concurrentes, es decir, que realizan múltiples operaciones al mismo tiempo.

Maude destaca por su flexibilidad, ya que no solo permite crear modelos sino también verificar su comportamiento, algo muy útil para áreas como la computación orientada a objetos y la simulación de sistemas. Además, Maude tiene la capacidad de modificarse a sí mismo, lo que lo convierte en una herramienta muy poderosa y adaptable.

[5] Maude: https://maude.cs.illinois.edu/wiki/Maude_Overview

2.1.2 Python3

Python[6] es un lenguaje de programación muy popular debido a su facilidad de uso y su sintaxis clara, lo que permite escribir y entender código de manera más simple. Se utiliza en una gran variedad de proyectos, desde scripts pequeños hasta aplicaciones web complejas.

En este proyecto, Python fue el lenguaje elegido para gestionar la parte del backend, es decir, las operaciones que se ejecutan en el servidor, ya que permite manejar tareas de manera rápida y eficiente.

2.1.3 Django

Django[7] es un framework web basado en Python que facilita la creación de aplicaciones web de forma rápida y estructurada. Gracias a Django, se puede desarrollar una aplicación con menos esfuerzo, ya que provee herramientas para gestionar usuarios, bases de datos, y muchas otras funciones necesarias para crear sitios web dinámicos y seguros.

Durante el proyecto, Django fue utilizado para crear la interfaz web que permite a los usuarios interactuar con Maude de manera más intuitiva, sin necesidad de usar la línea de comandos.

2.1.4 Bindings Maude en Python

Los *bindings*[8] son herramientas que permiten que dos lenguajes de programación diferentes trabajen juntos. En este caso, los bindings de Maude en Python permiten que las funcionalidades de Maude se utilicen dentro de una aplicación hecha en Python.

---

[6] Python: https://docs.python.org/es/3/tutorial/

[7] Django: https://www.djangoproject.com/start/overview/

[8] *Bindings* de maude en Python: https://github.com/fadoss/maude-bindings?tab=readme-ov-file y https://fadoss.github.io/maude-bindings/

Esto es posible gracias a una herramienta llamada SWIG, que genera estos "puentes" entre ambos lenguajes.

Gracias a los *bindings*, fue posible conectar Maude con la aplicación web desarrollada en Django, lo que permitió que los usuarios pudieran ejecutar comandos de Maude directamente desde la web.

2.1.5 SQLite3

SQLite[9] es una base de datos ligera y fácil de usar. A diferencia de otras bases de datos, no necesita un servidor para funcionar, ya que todos los datos se guardan en un único archivo en el disco.

En el proyecto, SQLite se usó para almacenar toda la información necesaria, como los usuarios, módulos y tareas. Su integración con Django fue sencilla y permitió gestionar los datos de manera eficiente sin complicaciones adicionales.

2.1.6 Pythonanywhere

PythonAnywhere[10] es una plataforma en la nube que permite a los desarrolladores alojar y ejecutar aplicaciones Python directamente desde el navegador, sin tener que configurar servidores ni preocuparse por la instalación de software.

Esta plataforma fue clave para desplegar la aplicación web y hacerla accesible a los usuarios de manera sencilla, ya que proporciona un entorno listo para usar, con soporte para Django.

---

[9] SQLite: https://www.sqlite.org/about.html

[10] Pythonanywhere: https://www.pythonanywhere.com

2.1.7 Git y Github

Git[11] es una herramienta que permite a los desarrolladores llevar un control de las versiones del código. Esto significa que pueden realizar cambios, revertir errores y colaborar de manera segura en un proyecto sin perder el historial de lo que se ha modificado.

GitHub es una plataforma que facilita la colaboración usando Git, permitiendo a los desarrolladores compartir su código, revisar los cambios y trabajar juntos de manera organizada. En este proyecto, se usaron Git y GitHub para gestionar todo el código desarrollado, permitiendo realizar un seguimiento de los avances, corregir errores y trabajar de forma colaborativa.

2.2 Aplicaciones parecidas

Al analizar las aplicaciones relacionadas con Maude, podemos agruparlas en función de su funcionalidad principal, enfocándonos en la detección de errores, la corrección y optimización de programas, y las herramientas para desarrolladores avanzados.

2.2.1 Herramientas de Detección de Errores y Análisis Dinámico

Estas aplicaciones se centran en la detección y diagnóstico de errores durante la ejecución de programas en Maude.

- **ABET[12]**: Combina la comprobación en tiempo de ejecución y la segmentación de trazas para mejorar el diagnóstico de errores en programas de Maude. Proporciona un análisis dinámico que simplifica las trazas para ayudar a los usuarios a identificar la causa de los errores.

---

[11] Git y Github: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

[12] Abet: https://safe-tools.dsic.upv.es/abets/

- **Atame**[13]: Se centra en la corrección automática de programas en Maude al diagnosticar y corregir cálculos que no satisfacen las aserciones establecidas por el sistema. Aunque no requiere una ejecución fallida, Atame asegura que todas las computaciones futuras cumplan con las aserciones definidas.

Ambas herramientas comparten un enfoque en la detección de errores y la corrección automática o asistida de programas. **ABET** es más útil para la depuración dinámica en tiempo de ejecución, mientras que **Atame** es más adecuada para la corrección estática.

2.2.2 Optimización de Programas Mediante Evaluación Parcial y Especialización

Estas herramientas buscan mejorar la eficiencia y precisión de los análisis a través de la evaluación parcial y la especialización de programas en Maude.

- **Presto**[14]: Un evaluador parcial simbólico para las teorías de reescritura de Maude que optimiza la teoría equacional subyacente para mejorar el análisis de protocolos y sistemas complejos. Ofrece un aumento significativo en el rendimiento al eliminar axiomas algebraicos costosos.
- **Victoria**: Similar a Presto, esta herramienta se enfoca en la evaluación parcial de programas de reescritura en Maude. Victoria aplica técnicas de evaluación parcial para optimizar los programas, lo que resulta en mejoras de rendimiento significativas.

Ambas herramientas buscan optimizar programas mediante la evaluación parcial, especializando teorías equacionales complejas para mejorar la eficiencia de los análisis en sistemas concurrentes.

---

[13] Atame: https://safe-tools.dsic.upv.es/atame/

[14] Presto: https://safe-tools.dsic.upv.es/iPresto/

2.2.3 Herramientas para Desarrolladores Avanzados y Extensiones de Maude

Estas aplicaciones están diseñadas para mejorar la experiencia de los desarrolladores que utilizan Maude, añadiendo capacidades avanzadas de desarrollo y análisis.

- **Mau-Dev**[15]: Una extensión para desarrolladores que añade nuevas operaciones de meta-nivel a Maude. Proporciona operaciones nativas de alto rendimiento que son útiles para gestionar representaciones complejas de términos y trazas de reescritura.
- **Meta-Maudest**[16]: Un marco de transformación basado en el plegado y desplegado (fold/unfold) de teorías de reescritura. Es útil para optimizar y transformar programas utilizando técnicas simbólicas avanzadas.

Estas herramientas están más orientadas a desarrolladores que trabajan a un nivel avanzado en la creación y modificación de teorías de reescritura y desean optimizar o transformar sus programas de manera eficiente.

2.2.4 Análisis de Protocolos Criptográficos

Este grupo de aplicaciones está especializado en el análisis de protocolos criptográficos dentro de Maude.

- **Maude-NPA**[17]: Una herramienta avanzada para el análisis de protocolos criptográficos que toma en cuenta propiedades algebraicas complejas como la cancelación de la encriptación y la desencriptación. Utiliza técnicas de

---

[15] Mau-Dev: https://safe-tools.dsic.upv.es/maudev/

[16] Meta-Maudest: https://elp.webs.upv.es/soft/maudest/

[17] Maude-NPA: https://maude.cs.illinois.edu/w/index.php?title=Maude_Tools:_Maude-NPA

búsqueda hacia atrás y unificación para determinar si un estado final es alcanzable.

Esta herramienta es única en su enfoque en la seguridad criptográfica, proporcionando capacidades de análisis que no están presentes en otras aplicaciones de Maude.

Capítulo 3 - Arquitectura e implementación

3.1 Arquitectura

En este apartado se indagará en la arquitectura del proyecto diviéndolo en dos subsecciones, *backend* y *frontend*.

3.1.1 Frontend

El frontend de la aplicación web está diseñado siguiendo principios de modularidad y reutilización de componentes, con un fuerte énfasis en la organización y la claridad del código. Utiliza tecnologías estándar como HTML, CSS, JavaScript, y está apoyado en Bootstrap para la estilización y el manejo de la interfaz de usuario.

3.1.1.1 Estructura de carpetas y archivos

La estructura del proyecto se organiza en dos carpetas principales: templates y static:

- **Templates**: Contiene todas las plantillas HTML, que están organizadas en carpetas según su funcionalidad. Cada página tiene su propia plantilla HTML que se encarga de la estructura básica y el contenido estático. Esta organización facilita la localización y mantenimiento de las plantillas.
- **Static**: Agrupa todos los archivos estáticos, incluyendo CSS y JavaScript, organizados en carpetas específicas para cada tipo de archivo:
    - **Archivos CSS**: Los estilos están divididos en archivos específicos para cada página o sección, lo que permite una mayor modularidad y evita la sobrecarga de un único archivo de estilos. Bootstrap se utiliza como base para la estilización, lo que reduce el tiempo de desarrollo y asegura una apariencia coherente en toda la aplicación.
    - **Archivos JavaScript**: La lógica de interacción del *frontend* está separada en archivos JavaScript por funcionalidad. Cada archivo maneja una parte específica de la aplicación, como la gestión de tareas o la manipulación de entregas. Esto permite una mayor claridad

y facilita la localización de funcionalidades específicas para su desarrollo o depuración.

Esta separación clara entre las plantillas HTML y los archivos estáticos en carpetas dedicadas mejora la organización del proyecto, facilitando el desarrollo, mantenimiento y escalabilidad de la aplicación.

3.1.1.2 Uso de Componentes Reutilizables

Se emplean componentes reutilizables para elementos comunes de la interfaz, como la barra de navegación (navbar) y los modales. Estos componentes se definen una vez y se incluyen en las páginas que los necesitan, lo que reduce la duplicación de código y facilita su mantenimiento.

3.1.1.3 Estilización con Bootstrap

Bootstrap se utiliza como el *framework* CSS principal, proporcionando una base sólida para la creación de una interfaz de usuario moderna y responsiva. Los estilos personalizados se aplican a través de archivos CSS específicos que sobreescriben o complementan las clases de Bootstrap, permitiendo ajustar la apariencia de la aplicación a las necesidades específicas del proyecto.

3.1.1.4 Interactividad y Peticiones AJAX

La interactividad en la aplicación se maneja principalmente mediante JavaScript, haciendo un uso intensivo de peticiones AJAX (Asynchronous JavaScript and XML). Estas peticiones permiten la comunicación asincrónica con el backend, lo que facilita la actualización de partes específicas de la página sin necesidad de recargarla completamente.

Las peticiones AJAX se utilizan para:

- **Cargar y actualizar listas de tareas y entregas**: Se emplean peticiones AJAX para cargar dinámicamente las tareas y entregas en la página, así como para filtrar, ordenar y paginar los resultados en tiempo real.
- **Manipulación de datos**: Acciones como eliminar entregas seleccionadas o corregir tareas se realizan mediante peticiones AJAX, lo que mejora la

velocidad de respuesta de la aplicación y ofrece una mejor experiencia de usuario.

- **Visualización de detalles en modales**: Al hacer clic en elementos específicos, se realizan peticiones AJAX para obtener detalles adicionales desde el servidor y mostrarlos en modales, permitiendo a los usuarios interactuar con la aplicación de manera fluida.

Estas peticiones son manejadas mediante el uso de fetch, que es el método estándar en JavaScript moderno para realizar solicitudes HTTP asincrónicas. El uso de AJAX asegura que la aplicación sea más reactiva y eficiente en el manejo de datos, proporcionando una experiencia de usuario más dinámica.

3.1.1.5 Validación y UX Mejorada

Se ha implementado la validación en tiempo real en formularios críticos, como los de registro y recuperación de contraseñas, para mejorar la experiencia del usuario y reducir la probabilidad de errores. Además, se utilizan modales y alertas para proporcionar feedback inmediato a los usuarios sobre las acciones que realizan, lo que contribuye a una experiencia de usuario más intuitiva y agradable.

3.1.2 Backend

El *backend* de esta aplicación web está construido utilizando el *framework* Django, reconocido por su capacidad para desarrollar aplicaciones web escalables y seguras de manera eficiente. Django sigue el patrón de diseño Modelo-Vista-Controlador (MVC), aunque dentro de su ecosistema es común referirse a este patrón como Modelo-Plantilla-Vista (MTV).

3.1.2.1 Estructura General

- **Modelos**: Los modelos definen la estructura de la base de datos mediante clases en Python. En Django, cada modelo representa una tabla en la base de datos, y cada atributo de una clase se traduce en un campo dentro de esa tabla.

- **Vistas**: Las vistas manejan la lógica de negocio, respondiendo a las solicitudes HTTP y devolviendo las respuestas correspondientes. Las vistas en Django pueden implementarse como funciones o clases.
- **URLs**: El sistema de enrutamiento de Django asigna las solicitudes entrantes a las vistas adecuadas utilizando patrones basados en expresiones regulares. Esto permite dirigir el tráfico de manera precisa a la lógica de negocio correspondiente.
- **Formularios**: Los formularios en Django se emplean para validar los datos y facilitar la interacción entre las vistas y los modelos, asegurando que los datos enviados por los usuarios sean correctos antes de ser procesados.

3.1.2.2 Modelos y Base de Datos

La aplicación utiliza una base de datos relacional SQL, la cual es compatible con múltiples motores de bases de datos soportados por Django, como PostgreSQL, MySQL, SQLite y Oracle. Dado que no se especifica explícitamente el motor de la base de datos en el código, es probable que se esté utilizando SQLite, que es la opción por defecto en Django, o PostgreSQL, dado que son comunes en proyectos Django.

Estos son los modelos principales:

- **Usuario**: Este modelo personalizado extiende de AbstractBaseUser y gestiona la autenticación y la información de los usuarios. Incluye campos como nombre, correo electrónico, verificación de correo, entre otros.
- **Chat**: Representa un chat asociado a un usuario específico mediante una relación de ForeignKey con el modelo Usuario.
- **Mensaje**: Almacena los mensajes enviados dentro de un chat, incluyendo los comandos ejecutados, las respuestas obtenidas y el estado del mensaje.
- **Modulo**: Define un módulo Maude, almacenando información como el nombre, descripción, código Maude, y una imagen opcional que puede ser asociada al módulo.
- **ModuloVersion**: Guarda las diferentes versiones de un módulo específico, vinculadas a un chat.

- **Entrega**: Gestiona las entregas realizadas por los usuarios a los administradores, permitiendo la asociación de múltiples mensajes a través de una relación ManyToMany.

3.1.2.3 Diagrama

El diagrama de la base de datos, como se puede ver en la **figura 2**, ilustra las principales entidades y las relaciones que se gestionan en la aplicación web. Este diagrama muestra la estructura de los modelos utilizados en el *backend*, destacando las siguientes entidades clave:

1. **Usuario**: Este modelo, que extiende *AbstractBaseUser*, representa a los usuarios de la plataforma, tanto normales como administradores. Los campos principales incluyen nombre, email, email_verificado, y is_admin, lo que permite distinguir entre usuarios estándar y administradores. Cada usuario tiene una relación de uno a muchos con los modelos **Chat**, **Modulo**, y **Entrega**.
2. **Chat**: Un chat está vinculado a un usuario mediante una relación de clave foránea (usuario). En cada chat, los usuarios pueden ejecutar comandos que están vinculados a un módulo Maude (modulo). Los chats también contienen una colección de mensajes asociados.
3. **Mensaje**: Los mensajes dentro de un chat incluyen información sobre el comando ejecutado, la respuesta devuelta por el sistema, y un estado (bien, mal, ninguno). Los mensajes están relacionados con un chat y pueden asociarse a entregas realizadas por los usuarios.
4. **Modulo**: Representa los módulos Maude que los usuarios pueden crear y gestionar. Este modelo incluye campos como nombre, descripcion, codigo_maude, e imagen, y está vinculado a un creador (usuario). Los módulos pueden estar activos o inactivos, y también pueden tener múltiples versiones asociadas.
5. **ModuloVersion**: Este modelo guarda las diferentes versiones de un módulo, cada una vinculada a un chat específico. De esta manera, los usuarios pueden trabajar con versiones anteriores de un módulo según sea necesario.

6. **Entrega**: Este modelo gestiona las entregas que los usuarios realizan hacia los administradores. Las entregas contienen una colección de mensajes a través de una relación de muchos a muchos. Además, se registra información como la fecha de entrega, si fue corregida o no, y las notas de los administradores.
7. **Modelos de Django predeterminados**:
    - **Permission** y **Group**: Gestionan los permisos y roles de los usuarios dentro del sistema.
    - **LogEntry** y **ContentType**: Permiten llevar un control de auditoría sobre las acciones realizadas en la plataforma y gestionar los tipos de contenido.

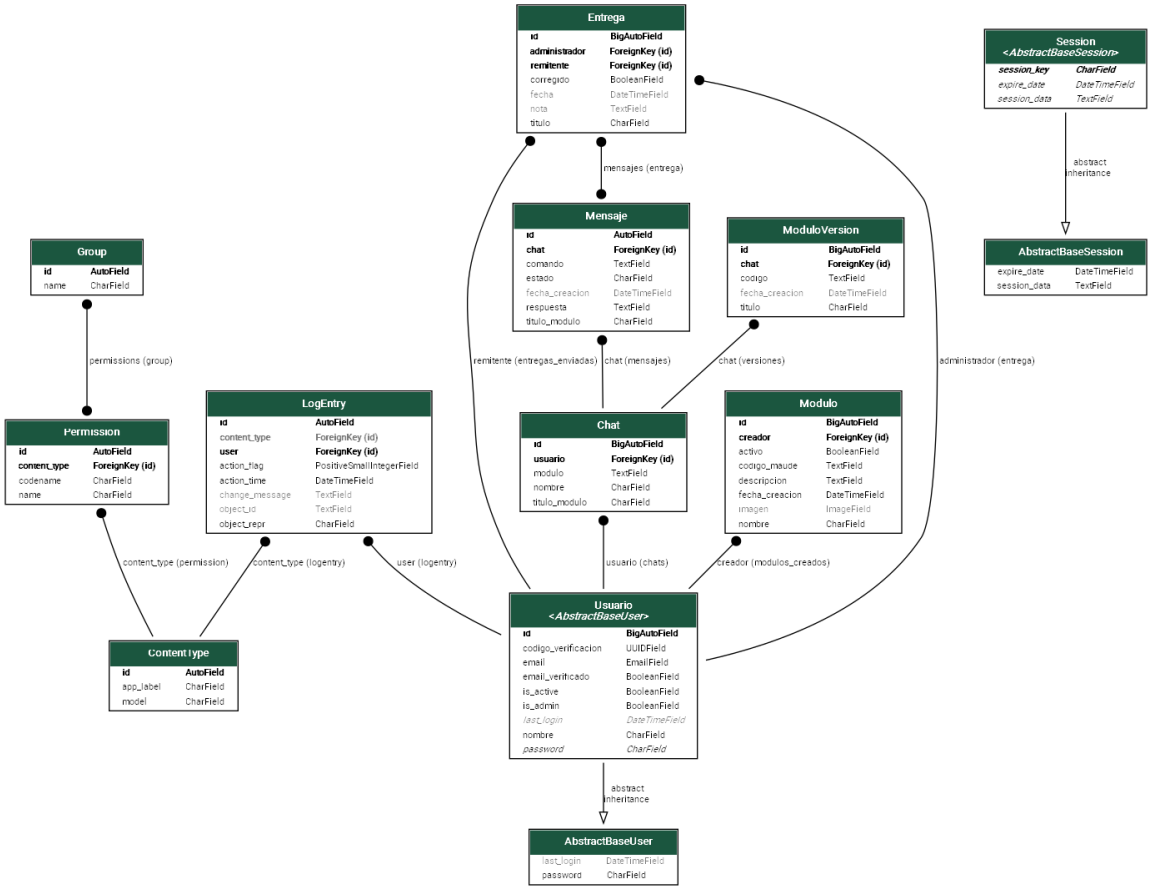

Figura 2 - Diagrama de la base de datos

Este diagrama muestra una arquitectura clara y estructurada para gestionar los diferentes elementos que intervienen en la funcionalidad de la plataforma,

aprovechando las capacidades de Django para manejar relaciones complejas en bases de datos relacionales.

3.1.2.4 Vistas y Lógica de Negocio

Las vistas en este proyecto son responsables de gestionar diversas funcionalidades, desde la autenticación de usuarios hasta la manipulación de chats y módulos Maude.

Estas son las principales vistas:

- **Autenticación**:
  - login, logout_request, register: Estas vistas gestionan la autenticación de usuarios, incluyendo el registro y la verificación de correo electrónico.
  - password_reset_request, password_reset_confirm: Manejan el proceso de recuperación de contraseñas.
- **Gestión de Chats y Mensajes**:
  - home, chat_view, delete_chats: Estas vistas permiten la creación, visualización y eliminación de chats.
  - get_chat_content, saveModule, run_maude_command: Facilitan la interacción con el contenido de los chats y la ejecución de comandos en Maude.
- **Gestión de Módulos**:
  - marketModulos, create_modulo, update_modulo, toggle_modulo: Estas vistas permiten la creación, actualización y gestión de la disponibilidad de los módulos.
  - get_available_modules, get_module_info: Proporcionan información sobre los módulos disponibles y permiten su búsqueda.

- **Gestión de Entregas**:

- entregas_usuario, entrega_detalles, eliminar_entregas: Estas vistas gestionan las entregas realizadas por los usuarios.
- entregas_pendientes,corregir_entrega, historial_entregas_corregidas: Se centran en la gestión de entregas pendientes y corregidas por los administradores.

3.1.2.5 Integración y Manejo de Imágenes

El modelo Modulo incluye la capacidad de manejar imágenes que se suben y almacenan en el servidor. Para optimizar el almacenamiento y la carga de estas imágenes, se emplea la función compress_image, la cual reduce el tamaño de las imágenes antes de guardarlas, asegurando que no excedan un tamaño determinado.

3.1.2.6 Seguridad

La aplicación implementa varias medidas de seguridad para proteger la integridad y confidencialidad de los datos:

- **Protección CSRF**: Se utilizan decoradores como @csrf_protect y @ensure_csrf_cookie para proteger la aplicación contra ataques de tipo Cross-Site Request Forgery (CSRF), garantizando que las solicitudes sean legítimas y realizadas por usuarios autenticados.
- **Gestión Segura de Contraseñas**: Django gestiona las contraseñas de forma segura mediante el uso de algoritmos de hash, lo que asegura que las contraseñas no se almacenen en texto plano.
- **Control de Acceso**: A través de decoradores como @login_required, se asegura que solo los usuarios autenticados puedan acceder a determinadas vistas y funcionalidades. Además, solo los administradores tienen permisos para realizar ciertas acciones críticas dentro de la aplicación.
- **Limitación de Tiempo en Ejecuciones**: En la vista run_maude_command, se ha implementado una limitación de tiempo para la ejecución de comandos Maude. Esto es crucial para evitar que comandos que involucren bucles infinitos o procesos extensos afecten la disponibilidad de la aplicación. Si un comando excede el tiempo permitido de ejecución, este se interrumpe para proteger la estabilidad y el rendimiento del sistema.

Estas medidas garantizan un entorno seguro para los usuarios y administradores, protegiendo tanto los datos como la estabilidad de la aplicación.

3.2 Implentación

En este apartado de la implementación, hemos decidido estructurar la descripción técnica dividiendo las funcionalidades y el desarrollo en distintas vistas de la aplicación. Esta organización permitirá una presentación clara y ordenada de cómo cada componente de la aplicación ha sido construido, qué funcionalidades específicas ofrece y cómo interactúan entre sí. Al segmentar la explicación por vistas, podremos detallar el flujo de trabajo, las decisiones de diseño y los elementos clave que componen cada parte de la aplicación, facilitando así una comprensión más profunda del proyecto en su totalidad.

3.2.1 Vista del home

La vista home es el núcleo central de la interacción del usuario dentro de la aplicación, permitiendo la ejecución de comandos Maude, la gestión de chats y módulos, y la revisión y entrega de resultados. A continuación, se detalla la evolución e implementación de las principales funcionalidades dentro de esta vista:

3.2.1.1 Integración Inicial de Maude en Python

La primera fase de la implementación se centró en la integración de Maude con Python, creando los *bindings* necesarios para ejecutar comandos Maude directamente desde la aplicación. La función run_maude_command fue clave en esta etapa, permitiendo a los usuarios ingresar un módulo Maude y un comando, y visualizar los resultados directamente en la interfaz. Esta funcionalidad básica sentó las bases para la ejecución de operaciones Maude en tiempo real dentro de la aplicación.

3.2.1.2 Implementación del Sistema de Chats

A medida que la aplicación evolucionaba, se introdujo un sistema de chats que permitía a los usuarios crear y gestionar múltiples conversaciones. Cada chat actúa como un contenedor para los comandos Maude y sus respectivas respuestas. Los usuarios pueden crear tantos chats como deseen, y cada uno mantiene un historial completo de los comandos utilizados. Para una mejor organización, el input del módulo Maude se movió a un modal, accesible desde el chat correspondiente, permitiendo así una gestión más limpia y enfocada de los comandos dentro de cada conversación.

3.2.1.3 Integración del Market de Módulos

Con el objetivo de enriquecer la funcionalidad y ofrecer herramientas predefinidas, se añadió un botón que abre un modal para acceder al Market de módulos. Este Market permite a los administradores crear y gestionar módulos Maude que luego pueden ser utilizados por otros usuarios. Desde el modal, los usuarios pueden explorar la lista de módulos disponibles, ver detalles adicionales de cada módulo, o agregar directamente el código del módulo seleccionado a su chat activo. Esta funcionalidad es esencial para facilitar la reutilización y el intercambio de recursos dentro de la comunidad de usuarios.

3.2.1.4 Evaluación de Comandos

Posteriormente, se añadió una funcionalidad de evaluación para los comandos dentro de los chats. Después de ejecutar un comando, los usuarios pueden marcar si el resultado fue satisfactorio o no. Para ello, se implementaron dos botones: un tick verde que indica que el comando funcionó correctamente y una cruz roja que, aunque es meramente decorativa, proporciona un feedback visual. Los comandos marcados con un tick verde pueden ser seleccionados para su entrega posterior a un administrador.

3.2.1.5 Entrega de Comandos

La funcionalidad de entrega es un elemento crucial para la interacción entre estudiantes y profesores. Al seleccionar varios comandos marcados como correctos (tick verde), el usuario puede hacer clic en el botón "Entregar". Esto abre un modal que muestra todos los comandos seleccionados, junto con la versión del módulo Maude que

se utilizó. El usuario puede entonces seleccionar un profesor destinatario y asignar un nombre a la entrega antes de enviarla. Este proceso automatiza y organiza la revisión de ejercicios y comandos, facilitando la evaluación por parte de los administradores.

3.2.1.6 Gestión Avanzada de Módulos

Finalmente, se mejoró la gestión de módulos dentro de la vista home. Al hacer clic en el botón "Cambiar módulo", se abre un modal con tres opciones clave:

- **Modificar el Módulo Actual**: Permite a los usuarios editar el nombre y el código del módulo actual, similar a cómo se maneja el control de versiones en un repositorio Git. Esto asegura que cada cambio quede registrado y puede ser nombrado para facilitar su identificación.
- **Seleccionar una Versión Anterior**: Los usuarios pueden cargar una versión anterior del módulo, permitiendo así la reversión a un estado previo si es necesario.
- **Comparar Versiones**: Esta funcionalidad permite comparar dos versiones diferentes del módulo, destacando las diferencias entre ambas, tal como se hace en herramientas de control de versiones como Git. Esta comparación visual es esencial para entender los cambios realizados y su impacto en el código.

3.2.2 Vista del registro

En la vista de registro, el objetivo principal es permitir a los usuarios crear una cuenta en la aplicación. Esta funcionalidad se implementa mediante un formulario sencillo que solicita al usuario ingresar su nombre, correo electrónico institucional (de la Universidad Complutense de Madrid, UCM), y una contraseña que cumpla con ciertos requisitos de seguridad.

3.2.2.1 Estructura del Formulario

El formulario de registro consta de los siguientes campos:

- **Nombre**: Un campo de texto simple donde el usuario ingresa su nombre.

- **Correo UCM**: Este campo solicita un correo electrónico institucional, validando que el dominio sea "@ucm.es". Esta validación es fundamental para asegurar que solo miembros de la UCM puedan registrarse.
- **Contraseña y Confirmar Contraseña**: Para garantizar la seguridad, la contraseña debe cumplir con varios criterios:
    - Al menos 8 caracteres.
    - Incluir una letra mayúscula.
    - Incluir una letra minúscula.
    - Incluir al menos un número.

Estos requisitos se validan en tiempo real conforme el usuario va ingresando la contraseña, y se proporciona retroalimentación visual para ayudar al usuario a crear una contraseña segura.

3.2.2.2 Lógica de Validación

La lógica de validación se implementa utilizando JavaScript y se realiza en dos niveles:

1. **Validación del Correo**: Asegura que el correo proporcionado por el usuario termina con "@ucm.es". Esta validación es crítica para restringir el acceso a la plataforma a los miembros de la UCM.
2. **Validación de la Contraseña**: Mientras el usuario escribe, el sistema valida si la contraseña cumple con los requisitos mínimos. Además, verifica que la confirmación de la contraseña coincida con la original. Solo cuando todos los requisitos están cumplidos, el botón de registro se habilita.

3.2.2.3 Confirmación y Modal de Verificación

Una vez que el formulario se envía, se muestra un modal que informa al usuario que se ha enviado un correo electrónico para la verificación de su cuenta. Este correo contiene un enlace que, al ser accedido, activará la cuenta del usuario, permitiéndole iniciar sesión en la plataforma.

Este enfoque asegura que solo usuarios con correos electrónicos válidos y cuentas verificadas puedan acceder a las funcionalidades de la aplicación, añadiendo una capa extra de seguridad y control sobre los usuarios registrados.

3.2.3 Vista del login

La vista de login en la aplicación está diseñada para permitir a los usuarios acceder a su cuenta utilizando las credenciales registradas. Esta funcionalidad es crucial para garantizar que solo usuarios autenticados y verificados puedan acceder a las funcionalidades internas de la plataforma.

3.2.3.1 Estructura del Formulario

El formulario de login es sencillo y consta de los siguientes elementos:

- **Correo Electrónico**: Un campo donde el usuario debe ingresar el correo electrónico registrado durante el proceso de registro.
- **Contraseña**: Un campo de entrada de tipo "password" donde el usuario ingresa la contraseña correspondiente a su correo electrónico.

Ambos campos son obligatorios para proceder con el inicio de sesión.

3.2.3.2 Validación y Manejo de Errores

La validación del formulario se realiza de la siguiente manera:

1. **Coincidencia de Credenciales**: Al enviar el formulario, el sistema verifica si el correo electrónico y la contraseña coinciden con los datos registrados en la base de datos. Si alguno de los campos no es correcto, se muestra un mensaje de error indicando que las credenciales son incorrectas.
2. **Verificación del Correo Electrónico**: Incluso si las credenciales son correctas, el sistema también verifica si el correo electrónico ha sido verificado. Si el usuario no ha completado el proceso de verificación de correo electrónico, se muestra un mensaje de advertencia indicando que el usuario debe verificar su correo antes de poder iniciar sesión.

Estos mensajes de error se muestran directamente debajo del formulario, proporcionando retroalimentación inmediata al usuario.

3.2.3.3 Funcionalidades Adicionales

Además del formulario de identificación, la vista también incluye enlaces para registrar una nueva cuenta y para recuperar la contraseña en caso de que el usuario la haya olvidado. Estos enlaces dirigen al usuario a las vistas correspondientes de registro y recuperación de contraseña, facilitando la navegación y el manejo de situaciones comunes relacionadas con el acceso.

3.2.4 Vistas de recuperar contraseña

La funcionalidad de recuperación de contraseña es un componente esencial en cualquier aplicación que gestione cuentas de usuario, ya que permite a los usuarios restablecer su contraseña en caso de que la olviden. Este proceso se divide en dos vistas principales: la solicitud de recuperación y el formulario para establecer una nueva contraseña.

3.2.4.1 Solicitud de Recuperación de Contraseña

La primera parte del proceso de recuperación de contraseña es la solicitud de restablecimiento. En esta vista:

- Formulario de Correo Electrónico: Se presenta un formulario simple en el que el usuario debe ingresar el correo electrónico con el que se registró. Este es un campo obligatorio, ya que es necesario para identificar la cuenta asociada y enviar el enlace de recuperación.
- Envío del Enlace de Recuperación: Una vez que el usuario introduce su correo electrónico y envía el formulario, el sistema genera un enlace único de recuperación y lo envía al correo proporcionado. Este enlace contiene un token de seguridad que asegura que solo el propietario del correo puede restablecer la contraseña.

3.2.4.2 Establecimiento de la Nueva Contraseña

Tras recibir el enlace de recuperación en su correo, el usuario es dirigido a una segunda vista donde puede establecer una nueva contraseña. En esta vista:

- Formulario de Nueva Contraseña: Se presentan dos campos de entrada: uno para la nueva contraseña y otro para confirmar la nueva contraseña. Ambos campos son obligatorios.
- Validación de la Contraseña: La nueva contraseña debe cumplir con ciertos criterios de seguridad previamente establecidos:
    - Debe tener al menos 8 caracteres.
    - Debe incluir al menos una letra mayúscula, una letra minúscula y un número.
    - Ambas contraseñas deben coincidir para poder enviar el formulario.
- Habilitación del Botón de Envío: El botón para cambiar la contraseña permanece deshabilitado hasta que todos los criterios de la contraseña se cumplen y las contraseñas coinciden. Esto ayuda a prevenir errores y asegura que la nueva contraseña sea segura.

3.2.4.3 Flujo Completo

El flujo completo de recuperación de contraseña garantiza que los usuarios puedan restablecer su acceso de manera segura y eficiente. Desde la solicitud inicial hasta la actualización de la contraseña, el proceso está diseñado para ser intuitivo y seguro, protegiendo la integridad de las cuentas de los usuarios.

3.2.5 Vista del market modulos

La vista de Market Módulos es una parte crucial del sistema que permite a los administradores gestionar los módulos disponibles en la plataforma. En esta vista, los módulos pueden ser creados, editados, activados/desactivados, y eliminados, proporcionando un control total sobre los recursos que se ofrecen a los usuarios.

3.2.5.1 Interfaz de Usuario

La vista se presenta con una interfaz intuitiva y funcional que incluye los siguientes componentes:

- **Lista de Módulos**: Los módulos se muestran en tarjetas individuales que incluyen el nombre, una descripción breve, y una imagen (si está disponible). Cada tarjeta tiene opciones para:
    - **Ver y Editar**: Un botón que abre un modal con información detallada del módulo. Desde este modal, se puede modificar la imagen, la descripción y el código del módulo.
    - **Activar/Desactivar**: Un botón que permite activar o desactivar el módulo. Los módulos desactivados no estarán disponibles para los usuarios en la vista de Home.
- **Selección y Eliminación**: Cada módulo tiene un checkbox que permite seleccionarlo. Una vez seleccionado, aparece un botón de eliminación masiva, que permite eliminar varios módulos a la vez.

3.2.5.2 Búsqueda y Filtros

Para facilitar la gestión de un gran número de módulos, la vista incluye herramientas de búsqueda y filtrado:

- **Barra de Búsqueda**: Permite buscar módulos por nombre o descripción, haciendo más fácil encontrar un módulo específico.
- **Filtros de Ordenación**: Los módulos pueden ser ordenados por nombre o fecha de creación, en orden ascendente o descendente. También se puede filtrar la lista para mostrar solo los módulos activos, inactivos, o ambos.
- **Paginación**: La lista de módulos está paginada para mejorar la navegación cuando se manejan muchos módulos.

3.2.5.3 Modales de Interacción

La vista incluye dos modales clave para la interacción con los módulos:

- **Modal de Detalle**: Al hacer clic en "Más información", se abre un modal donde se pueden ver y editar los detalles del módulo seleccionado, incluyendo la imagen, la descripción y el código Maude. Los cambios se pueden guardar desde este modal.
- **Modal de Creación**: Permite a los administradores crear nuevos módulos. El formulario incluye campos para el nombre del módulo, descripción, código Maude, y una imagen opcional.

3.2.5.4 Funcionalidad AJAX

La vista utiliza AJAX para mejorar la experiencia del usuario:

- **Búsqueda Dinámica**: La búsqueda y los filtros se aplican dinámicamente sin necesidad de recargar la página, lo que permite una interacción más fluida.
- **Activación/Desactivación y Eliminación**: Las acciones de activar, desactivar y eliminar módulos también se realizan mediante AJAX, proporcionando feedback inmediato y manteniendo la vista actualizada sin recargar la página completa.

3.2.6 Vista de mis tareas

La vista de "Mis Tareas" es una herramienta esencial para los usuarios, permitiéndoles gestionar y revisar las tareas que han entregado. Esta vista ofrece un acceso detallado a la información relacionada con cada tarea, así como opciones de filtrado y eliminación para facilitar la gestión de múltiples entregas.

3.2.6.1 Interfaz de Usuario

La vista está diseñada para ser intuitiva y eficiente, con los siguientes componentes clave:

- Lista de Entregas: Las tareas entregadas se presentan en tarjetas individuales, cada una mostrando la siguiente información:
    - **Título**: El nombre de la tarea.
    - **Fecha de Entrega**: La fecha en la que se entregó la tarea.
    - **Estado**: Indicador de si la tarea ha sido corregida o no.
    - **Nota del Administrador**: Si la tarea ha sido corregida, se muestra la nota asignada por el administrador.

Cada tarjeta incluye un botón para ver detalles, que abre un modal con información adicional sobre la entrega.

- **Modal de Detalles de la Entrega**: Al hacer clic en "Ver Detalles", se abre un modal que muestra información detallada de la entrega seleccionada:
    - **Título y Administrador**: Nombre de la tarea y del administrador que la recibió.
    - **Fecha de Entrega y Estado**: La fecha en que se entregó y si la tarea ha sido corregida.
    - **Comandos y Respuestas**: Lista de comandos enviados, sus respuestas, y el código del módulo utilizado.
    - **Nota**: Si la tarea ha sido corregida, se muestra la nota.

3.2.6.2 Funcionalidades Adicionales

- **Eliminación de Entregas**: Los usuarios pueden seleccionar varias entregas usando checkboxes y eliminarlas simultáneamente. La eliminación es confirmada a través de un diálogo para evitar acciones accidentales.
- Búsqueda y Filtros:
    - **Barra de Búsqueda**: Permite buscar entregas por título.

- **Filtros**: Los usuarios pueden ordenar las tareas por título o fecha de entrega, en orden ascendente o descendente. También pueden filtrar las entregas por estado (corregidas, no corregidas, o todas).

- **Paginación**: La lista de tareas está paginada para manejar eficientemente un gran número de entregas. Los usuarios pueden navegar entre páginas para encontrar tareas específicas.

3.2.6.3 Funcionalidad AJAX

La vista utiliza AJAX para mejorar la experiencia del usuario al interactuar con la lista de tareas:

- **Cargas Dinámicas**: La lista de tareas, los filtros, la búsqueda, y la paginación se gestionan dinámicamente sin recargar la página completa. Esto permite una interacción más rápida y fluida.

- **Detalles en Tiempo Real**: Los detalles de las entregas se cargan en tiempo real cuando el usuario los solicita, manteniendo la información actualizada y accesible de manera eficiente.

3.2.7 Vistas de entregas

En esta sección, abordamos la implementación de las vistas de "Entregas Pendientes" y "Historial de Entregas Corregidas". Estas vistas están diseñadas para que los administradores puedan gestionar de manera eficiente las entregas realizadas por los usuarios, proporcionando una interfaz clara tanto para la corrección de tareas pendientes como para la revisión del historial de tareas ya corregidas.

3.2.7.1 Interfaz de Usuario

Ambas vistas comparten una estructura similar para facilitar la consistencia y la facilidad de uso:

- **Lista de Entregas**: En cada vista, las entregas se presentan en tarjetas individuales con la siguiente información clave:
    - **Título**: El nombre de la tarea.
    - **Remitente**: El usuario que ha enviado la tarea.
    - **Fecha de Entrega**: La fecha en que la tarea fue entregada.
    - **Nota**: Solo visible en la vista de "Historial de Entregas Corregidas", muestra la calificación o comentario asignado durante la corrección.

Cada tarjeta en ambas vistas incluye un botón para ver detalles o corregir la entrega. En la vista de "Historial de Entregas Corregidas", este botón permite revisar la corrección realizada y, si es necesario, modificar la nota asignada.

3.2.7.2 Funcionalidades Específicas

- **Corrección de Entregas** (Vista de Entregas Pendientes):
    - **Modal de Corrección**: Al seleccionar una entrega pendiente, se abre un modal donde el administrador puede ver los detalles de la entrega, incluyendo los comandos ejecutados, sus respuestas, y el código del módulo utilizado. El administrador puede entonces asignar una nota a la entrega, que se guardará y hará que la entrega se mueva al historial de entregas corregidas.
- **Revisión del Historial** (Vista de Historial de Entregas Corregidas):
    - **Modal de Revisión**: Similar al modal de corrección, pero en este caso permite al administrador revisar y, si es necesario, editar la nota asignada previamente a una entrega corregida.

3.2.7.3 Funcionalidades Comunes

- Búsqueda y Filtros:
    - **Barra de Búsqueda**: Permite buscar entregas por título.

- **Filtros de Ordenación**: Los administradores pueden ordenar las entregas por título o fecha de entrega, en orden ascendente o descendente.

- **Paginación**: Ambas vistas están paginadas, permitiendo a los administradores navegar fácilmente a través de grandes volúmenes de entregas.

3.2.7.4 Funcionalidad AJAX

La implementación de AJAX en ambas vistas mejora la experiencia del usuario al permitir una interacción rápida y fluida:

- **Carga Dinámica**: Las entregas, la búsqueda, los filtros, y la paginación se gestionan sin recargar toda la página. Esto permite a los administradores trabajar de manera más eficiente y sin interrupciones.
- **Actualización en Tiempo Real**: Cualquier cambio realizado en la nota de una entrega se refleja inmediatamente, manteniendo la base de datos y la interfaz sincronizadas.

3.3 Ejemplos de uso

En este apartado se presentarán ejemplos prácticos de uso de la plataforma desarrollada, ilustrando cómo los diferentes tipos de usuarios pueden interactuar con las funcionalidades clave del sistema. A través de escenarios detallados, se mostrará el flujo completo de operaciones desde la perspectiva de un usuario normal y un administrador, cubriendo desde la creación de chats y módulos hasta la gestión de entregas y la corrección de tareas. Cada ejemplo estará acompañado de capturas de pantalla y descripciones detalladas, permitiendo una comprensión clara y visual de cómo utilizar la plataforma de manera efectiva.

3.3.1 Escenario 1- Usuario

En este escenario, exploraremos el flujo completo de un usuario normal interactuando con la plataforma, desde el registro inicial hasta la gestión de tareas y módulos.

3.3.1.1 Registro

El primer paso para un nuevo usuario es registrarse en la plataforma. Para ello, el usuario accede a la página de registro donde se le presenta un formulario, como el que se ve en la figura 3, que debe completar con su nombre, correo electrónico de la

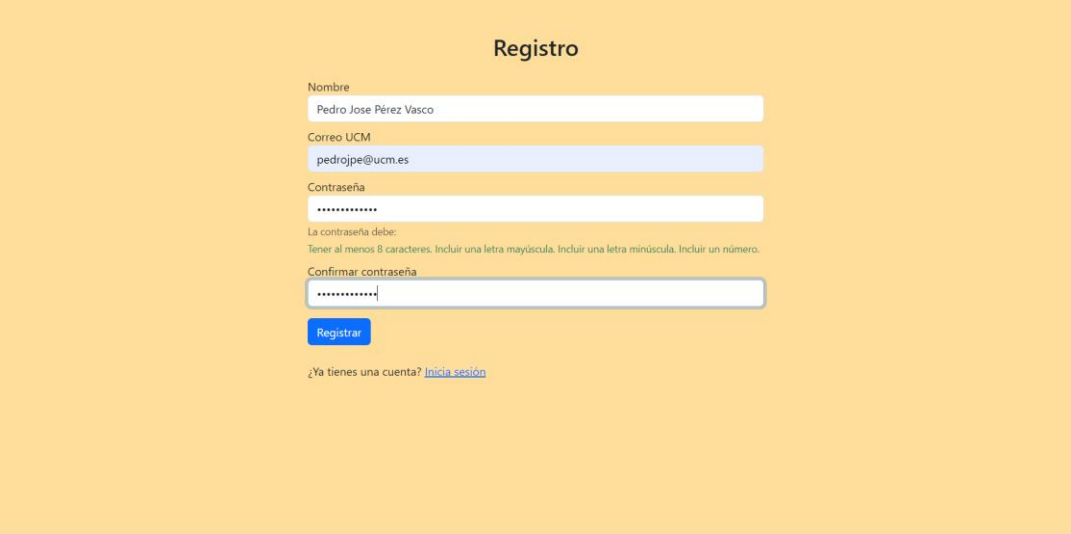

Figura 3 - Formulario de registro

Universidad Complutense de Madrid (UCM), y una contraseña que cumpla con los requisitos de seguridad establecidos, como incluir mayúsculas, minúsculas y números. Una vez completado el formulario, el usuario lo envía, lo que desencadena la aparición de un modal que le indica que debe revisar su correo electrónico para verificar su cuenta como el que podemos ver en la figura 4

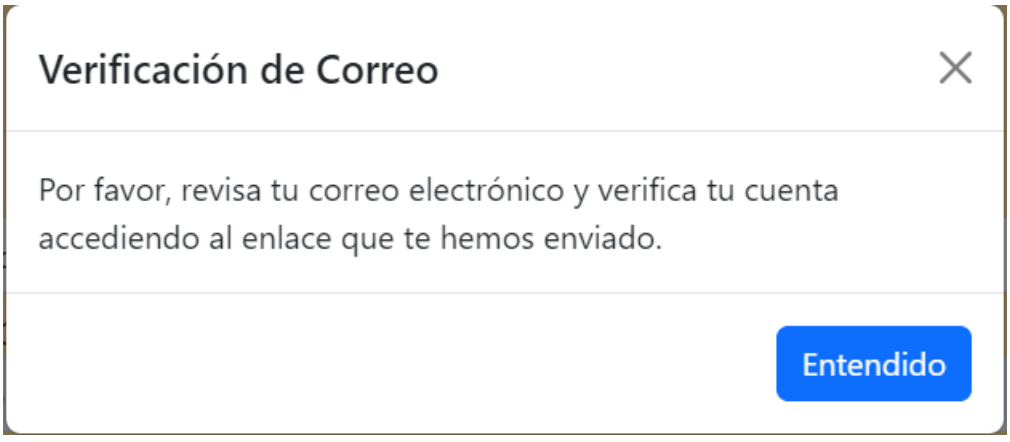

Figura 4 - Modal aviso para mirar correo

El usuario entonces abre su correo y encuentra un mensaje de verificación que contiene un enlace como el que podemos ver en la figura 5. Al hacer clic en este enlace, su cuenta se activa, y ya puede proceder a iniciar sesión en la plataforma.

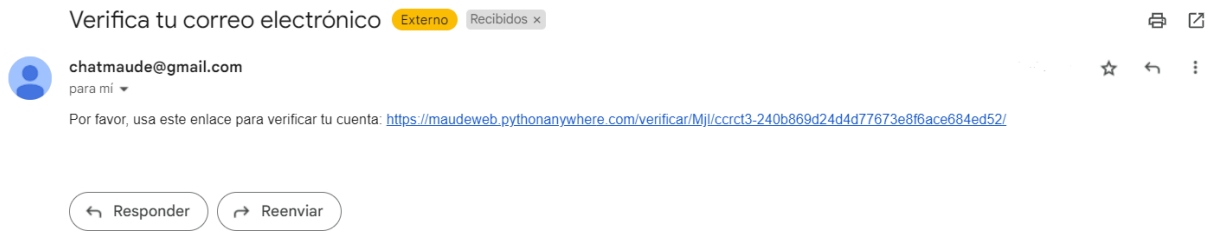

Figura 5 - Enlace enviado al correo electrónico

3.3.1.2 Recuperación de Contraseña

En caso de que el usuario olvide su contraseña, la plataforma ofrece una opción sencilla para recuperarla. El usuario navega a la página de recuperación de contraseña, donde se le solicita ingresar el correo electrónico con el que se registró. Tras enviar este formulario como en el que se puede ver en la figura 6, el sistema le envía un enlace de recuperación a su correo igual al que aparece en la figura 7.

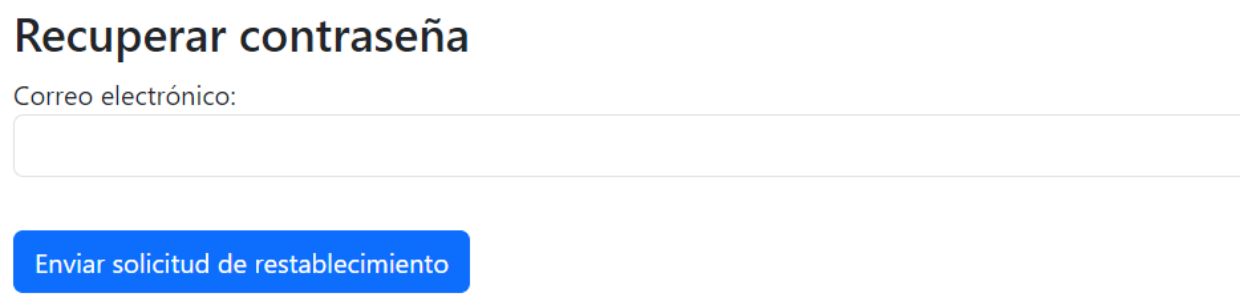

Figura 6 - Formulario recuperación de contraseña

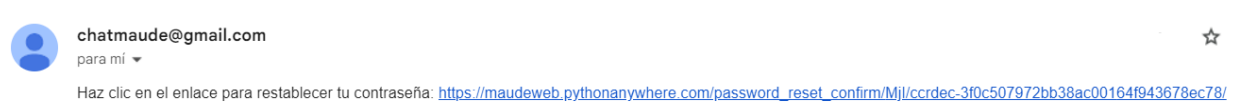

Figura 7 - Correo electrónico con enlace para recuperar contraseña

El usuario accede nuevamente a su correo, encuentra el mensaje de recuperación, y sigue el enlace proporcionado. Este enlace lo dirige a una página donde puede establecer una nueva contraseña igual al que el de la figura 8, asegurándose de que cumple con los requisitos de seguridad. Después de confirmar la nueva contraseña, puede utilizarla para iniciar sesión en la plataforma.

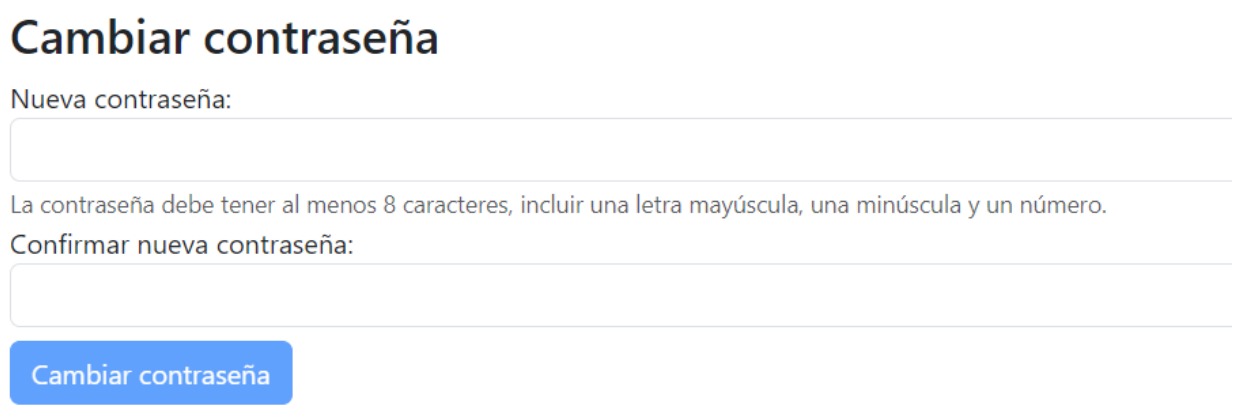

Figura 8 - Formulario para cambiar contraseña

3.3.1.3 Inicio de Sesión

Con su cuenta activada y, si es necesario, su contraseña restablecida, el usuario procede a iniciar sesión. En la pantalla de inicio de sesión, introduce su correo

electrónico y contraseña, accediendo así a la interfaz principal de la plataforma como se puede ver en la figura 9.

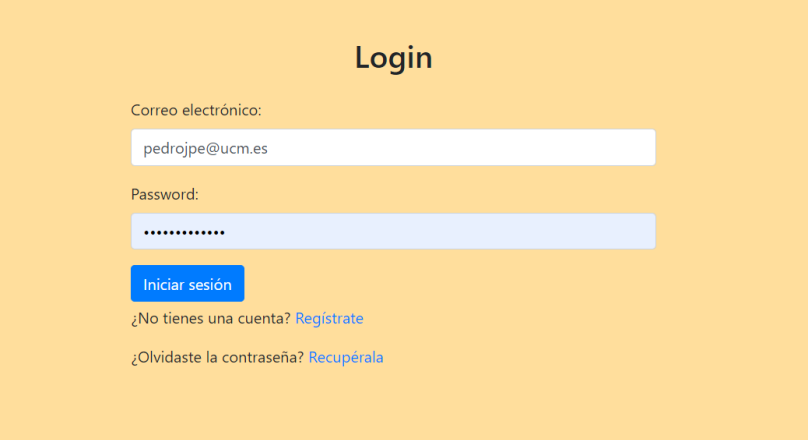

Figura 9 - Formulario para iniciar sesión

3.3.1.4 Creación de un Nuevo Chat

Una vez dentro de la plataforma, el usuario tiene la posibilidad de crear un nuevo chat. Esta funcionalidad le permite interactuar directamente con el sistema, ingresando comandos de Maude a través de un entorno amigable. El proceso es simple: el usuario selecciona la opción para crear un nuevo chat, lo cual abre un modal en el que puede configurar el nombre y otros parámetros del chat (como se ve en la Figura 10).

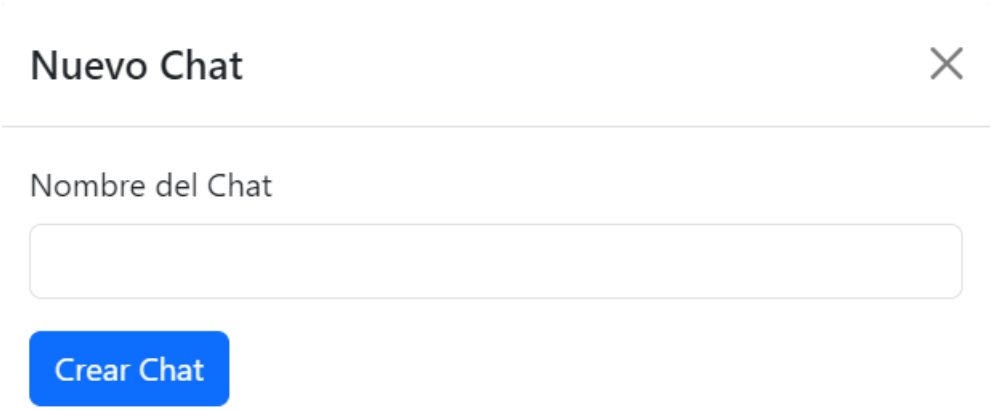

Figura 10 - Modal para crear chat

Después de configurar y crear el chat, el usuario accede a la interfaz principal del chat, donde puede empezar a experimentar con los comandos disponibles. En esta vista, cada comando ingresado se muestra junto a su respuesta, permitiendo al usuario llevar un registro claro de sus interacciones con el sistema (como se muestra en la Figura 11).

Figura 11 - Interfaz del chat

3.3.1.5 Gestión de Módulos

La plataforma también ofrece al usuario la capacidad de gestionar módulos. Desde el menú de módulos como se puede ver en la figura 12, el usuario puede crear un nuevo módulo, especificando su nombre, descripción y el código asociado. Esta funcionalidad es especialmente útil para organizar y reutilizar componentes en diferentes proyectos o tareas. La creación o modificación del módulo se puede ver en la figura 13.

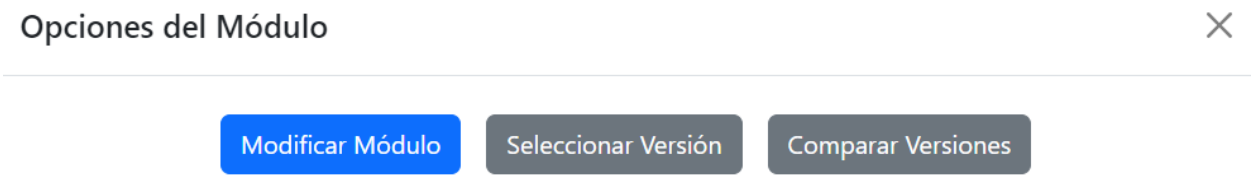

Figura 12 - Modal menú de la gestión del módulo

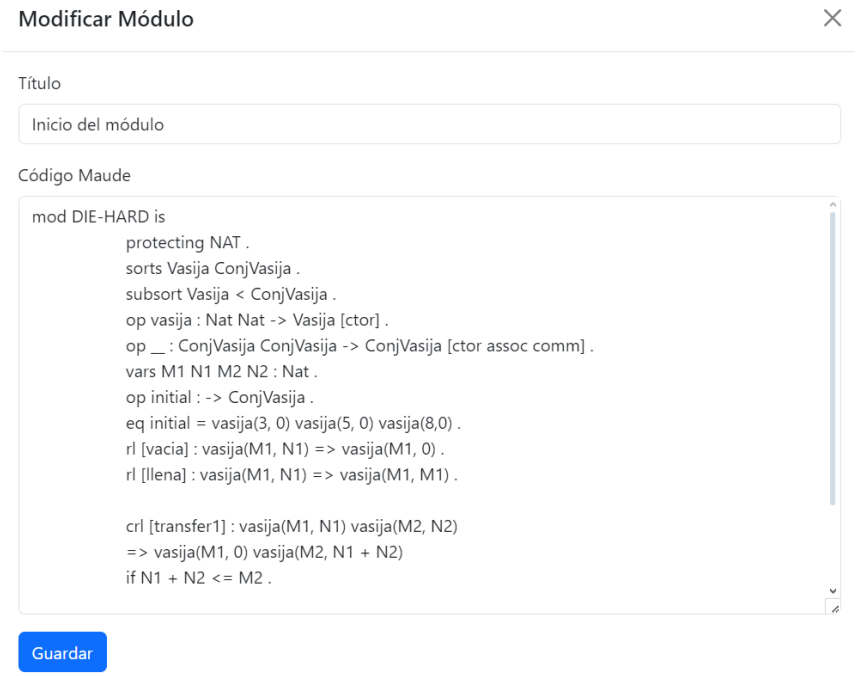

Figura 13 - Modal de creación/modificación de un módulo del chat

Después de crear el módulo, el usuario puede probar varios comandos en el chat como se ve en la figura 14. A medida que los comandos se ejecutan, el usuario tiene la opción de marcar aquellos que considera relevantes o útiles para futuras entregas.

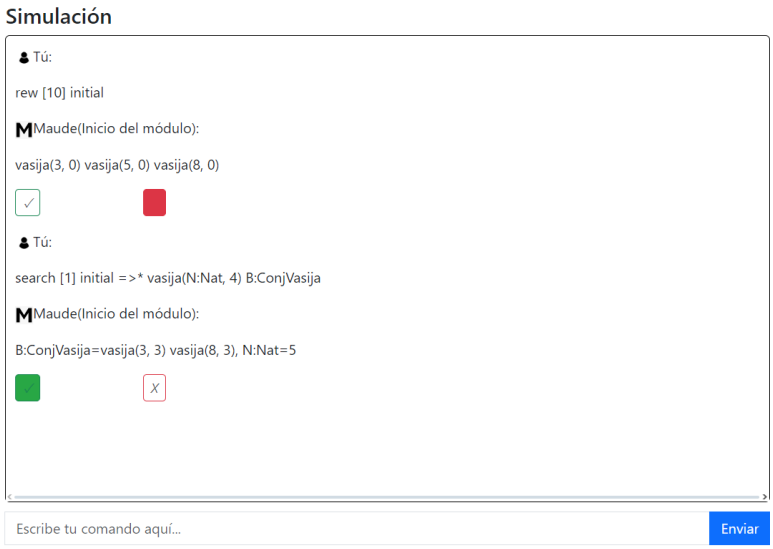

Figura 14 - Chat con comandos creados

3.3.1.6 Realización de una Entrega

Una vez que el usuario ha probado y marcado los comandos de su interés, puede proceder a realizar una entrega como en la figura 15. La plataforma le permite seleccionar los comandos marcados y empaquetarlos como una entrega formal. El usuario puede elegir a qué profesor enviar la entrega y asignarle un nombre, facilitando así la organización y el seguimiento de las tareas enviadas.

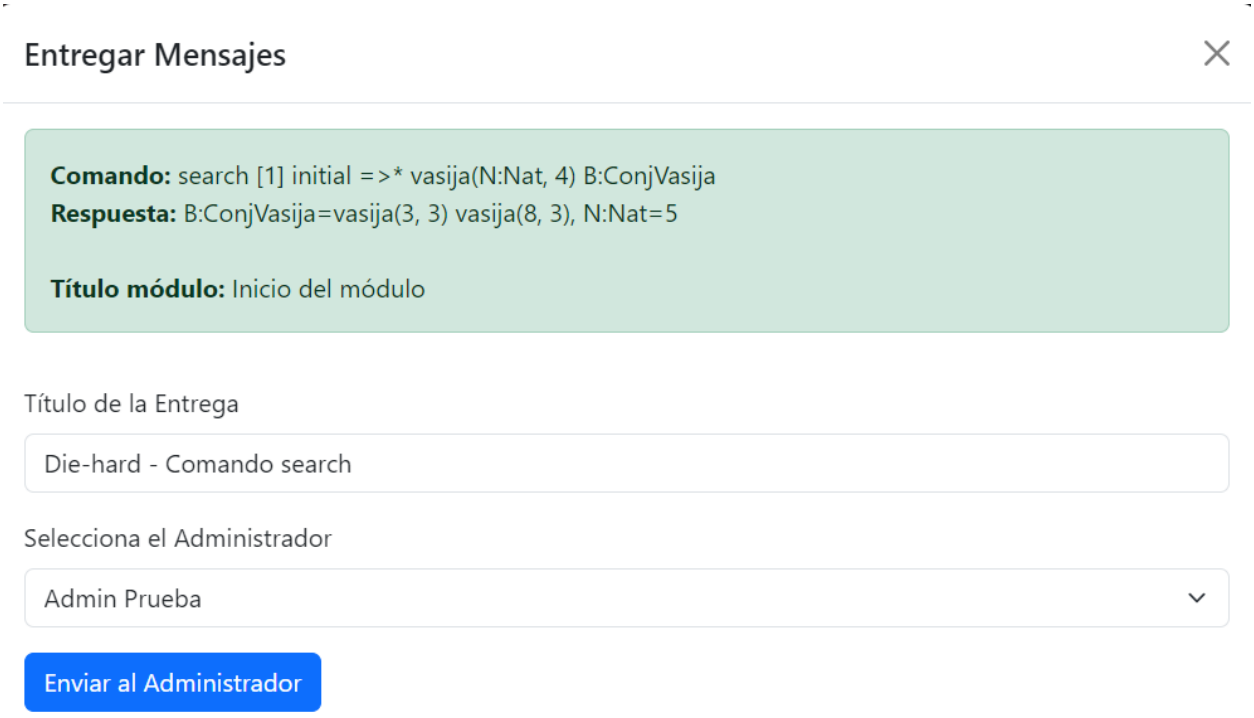

Figura 15 - Modal de la entrega de los comandos

3.3.1.7 Exploración del Market de Módulos

Además de crear sus propios módulos, el usuario tiene acceso al Market de módulos, donde puede explorar y añadir módulos preexistentes a su cuenta, se puede ver en la figura 16. Navegando por el Market, el usuario selecciona un módulo que le interesa y lo incorpora a su espacio de trabajo personal.

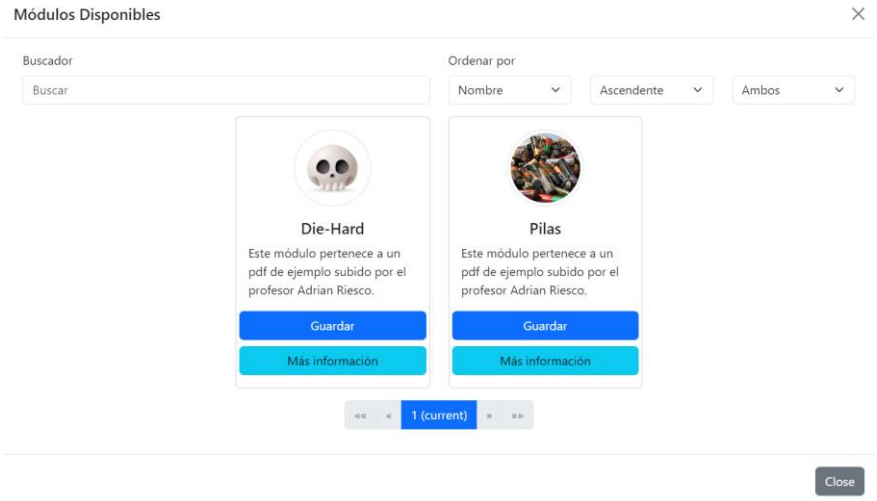

Figura 16 - Modal de los módulos activos en el market

3.3.1.8 Selección y Comparación de Módulos

Una vez añadido un nuevo módulo desde el Market, el usuario puede acceder a la función de selección de módulos y activar el nuevo módulo para su uso en el chat como se puede ver en la figura 17. Posteriormente, la plataforma permite comparar diferentes versiones de los módulos, lo que es invaluable para identificar cambios y asegurarse de que se está trabajando con la versión adecuada. Esta funcionalidad se puede ver en la figura 18.

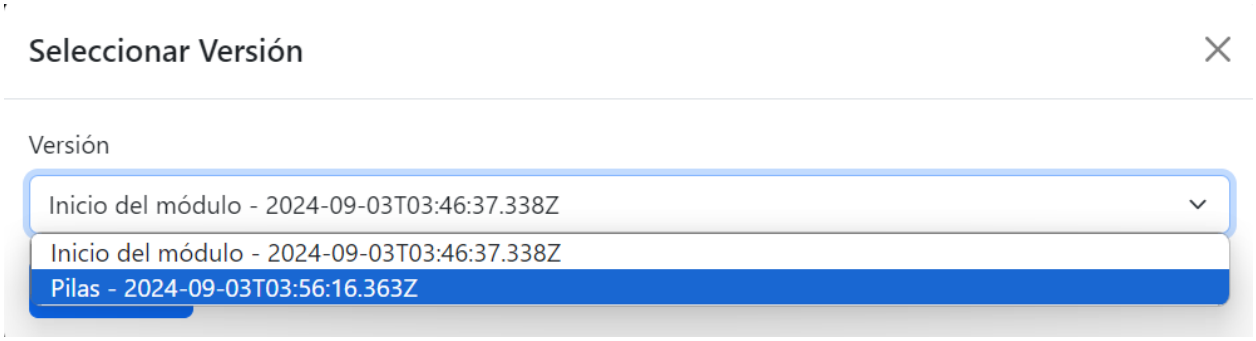

Figura 17 - Modal de selección de la versión de los módulos creados en un chat

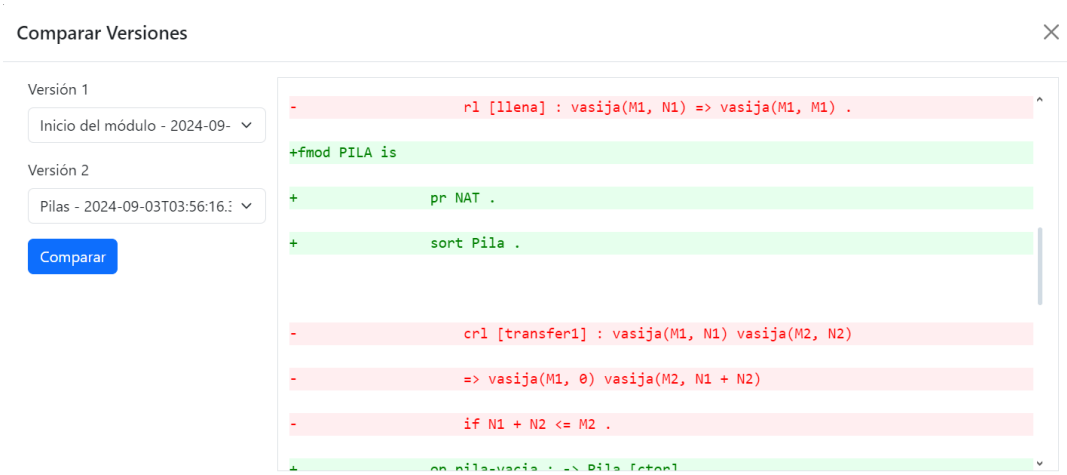

Figura 18 - Modal de comparación de versiones de los módulos de un solo chat

3.3.1.9 Gestión de Tareas

Finalmente, el usuario puede acceder a la sección de "Mis Tareas", donde puede revisar las entregas que ha realizado, como se puede ver en la figura 19. Al seleccionar una tarea, puede visualizar sus detalles en un modal, lo que incluye los comandos ejecutados, el código del módulo, y cualquier comentario o nota del administrador. Si ya no necesita la tarea en su lista, puede optar por eliminarla para mantener su espacio de trabajo ordenado. Se puede visualizar el modal con los detalles de una tarea en la figura 20.

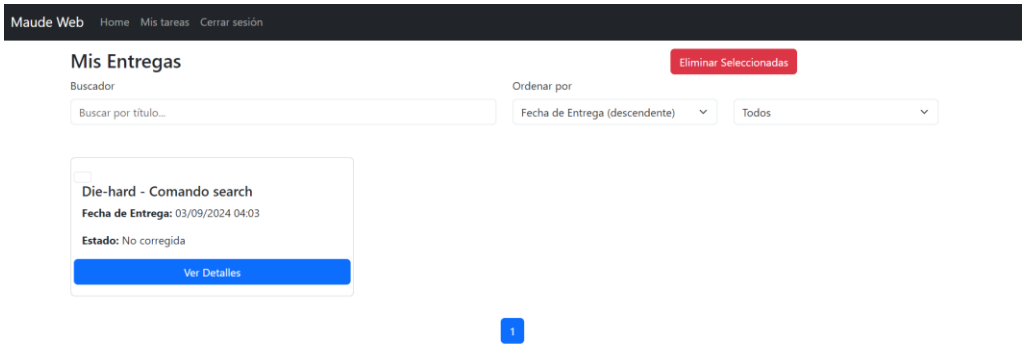

Figura 19 - Vista de mis tareas

Figura 20 - Modal de mostrar detalles de una entrega

3.3.2 Escenario del administrador

La sección dedicada al administrador describe las funcionalidades exclusivas para gestionar el Market de Módulos y las entregas de los usuarios. El administrador puede crear, modificar y eliminar módulos, además de revisar, corregir y gestionar las entregas pendientes y corregidas. Estas herramientas aseguran un control eficiente sobre los recursos disponibles y un seguimiento preciso del progreso de los usuarios.

3.3.2.1 Gestión del Market de Módulos

El administrador tiene acceso a una funcionalidad exclusiva: la gestión del Market de Módulos. Esta sección permite a los administradores crear, modificar y gestionar los módulos disponibles para los usuarios normales. Al acceder al Market, el administrador se encuentra con una lista de todos los módulos disponibles en el sistema, donde puede

ver información básica como el nombre y la descripción de cada módulo, así como opciones para activar, desactivar o eliminar módulos según sea necesario (como se puede ver en la Figura 21).

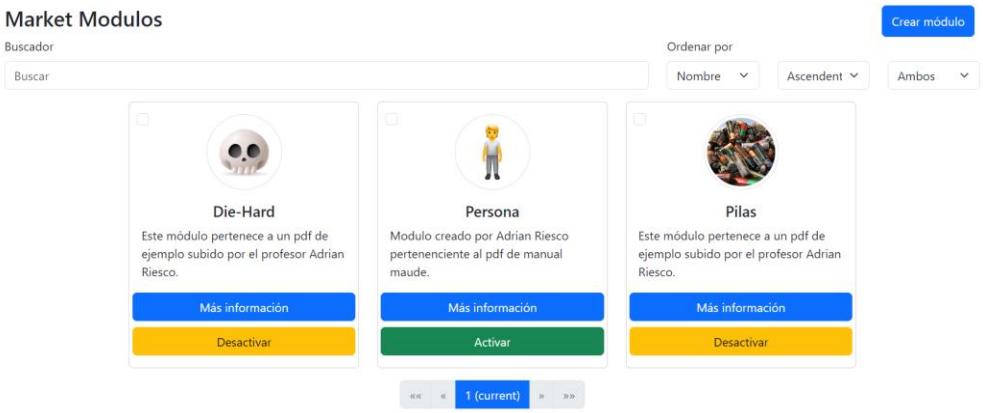

Figura 21 - Vista del market de los módulos

Para crear un nuevo módulo, el administrador selecciona la opción de "Crear módulo", lo que despliega un modal donde se pueden especificar todos los detalles necesarios, como el nombre, la descripción, el código Maude y una imagen representativa del módulo (como se observa en la Figura 22). Este proceso asegura que los módulos estén correctamente configurados antes de ser puestos a disposición de los usuarios.

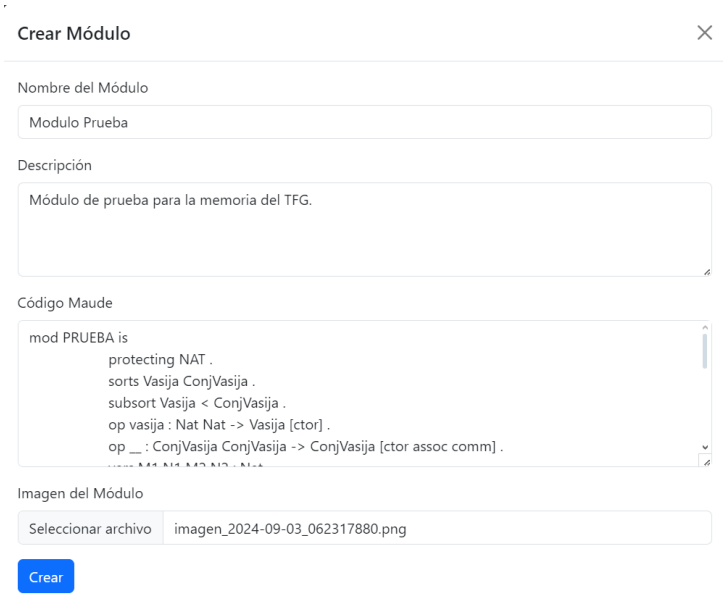

Figura 22 - Modal de creación de un módulo del market

Una vez creado un módulo, el administrador puede revisar y modificar sus detalles en cualquier momento. Al seleccionar la opción "Más información" en la lista de módulos, se abre un modal detallado donde se muestran todos los datos del módulo. Desde este modal, es posible realizar cambios en el código, la descripción, o incluso la imagen del módulo, permitiendo una gestión flexible y eficaz de los recursos disponibles en el sistema (como se ilustra en la Figura 23).

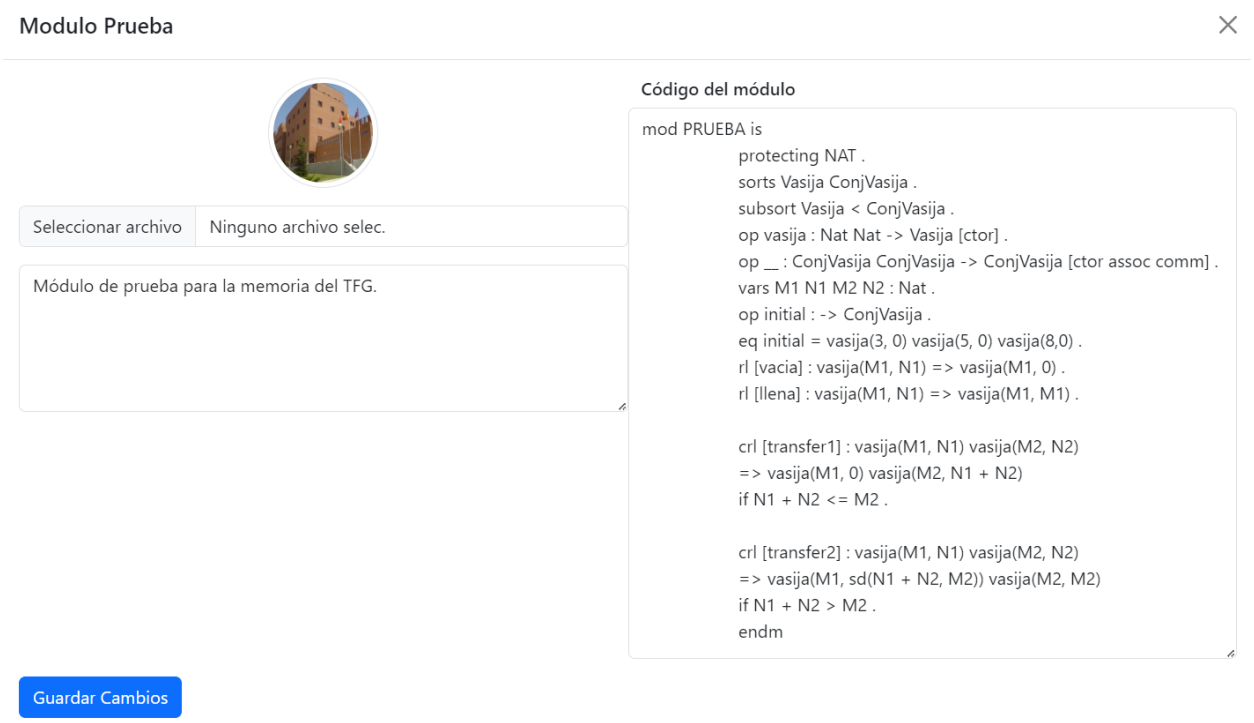

Figura 23 - Modal con los detalles de un módulo del market

3.3.2.2 Gestión de Entregas Pendientes y Corregidas

Además de la gestión de módulos, el administrador también tiene la responsabilidad de corregir las entregas realizadas por los usuarios. En la vista de "Entregas Pendientes", el administrador puede ver una lista de todas las entregas que aún no han sido revisadas. Cada entrega está asociada con un título, el nombre del remitente (usuario que realizó la entrega), y la fecha de entrega, lo que facilita la organización y priorización del trabajo (como se muestra en la Figura 24).

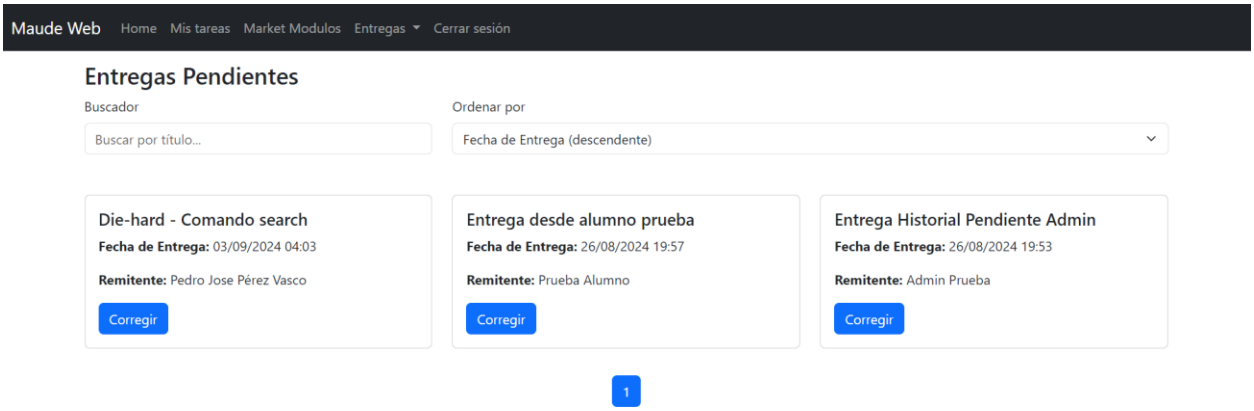

Figura 24 - Vista de las entregas pendientes de corregir por el administrador

Al seleccionar una entrega pendiente, se abre un modal que muestra todos los detalles de la entrega, incluyendo los comandos utilizados por el usuario y las respuestas generadas por el sistema. Este modal también permite al administrador asignar una nota a la entrega, evaluando su precisión y completitud. Una vez corregida, la entrega se mueve automáticamente al historial de entregas corregidas (como se observa en la Figura 25).

Figura 25 - Modal para ver los detalles y corregir una entrega de un usuario

En la vista de "Entregas Corregidas", el administrador puede revisar todas las entregas que ya han sido evaluadas. Cada entrada en esta lista muestra la nota asignada previamente, pero el administrador aún tiene la posibilidad de editar la nota si es necesario, por ejemplo, si se detecta un error en la corrección original. Este proceso asegura que todas las entregas se evalúan con precisión y se mantienen registros claros y actualizados (como se puede ver en la Figura 26).

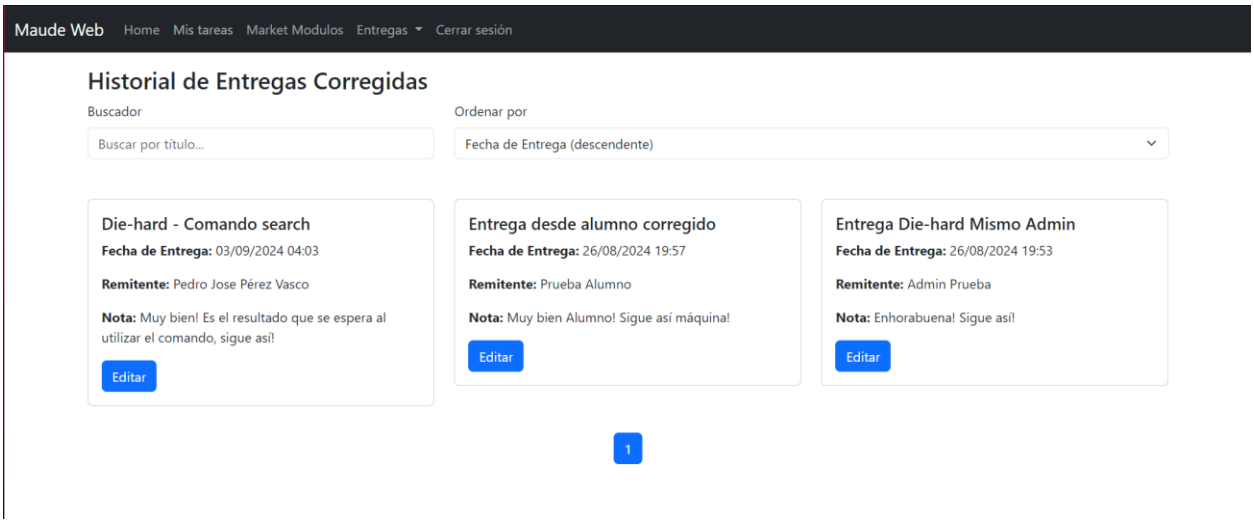

Figura 26 - Vista del historial de las entregas corregidas

Capítulo 4 - Conclusiones y trabajo futuro

4.1 Conclusiones

Al comenzar este proyecto, se establecieron una serie de objetivos que guiaron el desarrollo y aseguraron que la plataforma final cumpliera con las expectativas planteadas. A continuación, se detalla cómo se cumplieron estos objetivos:

1. **Integrar Maude con Python**: El primer y más crucial objetivo fue lograr la integración de Maude con Python a través de bindings. Este paso fue esencial para permitir la ejecución de comandos de Maude desde una interfaz web, facilitando su uso en un entorno más accesible y moderno. Después de una investigación exhaustiva sobre cómo funcionan los bindings de Maude en Python, se desarrolló una funcionalidad que permite traducir comandos de Maude a un formato entendible por Python y ejecutar estos comandos dentro de la plataforma. Esta integración se probó rigurosamente para asegurar que la comunicación entre Maude y la interfaz web fuera fluida y sin errores, cumpliendo así con este objetivo fundamental.
2. **Desarrollar una plataforma web robusta y funcional utilizando Django**: Dado que Django era un framework nuevo para el equipo de desarrollo, se invirtió tiempo en comprender su arquitectura y mejores prácticas. Django fue elegido por su capacidad para manejar aplicaciones web complejas de manera eficiente. La plataforma desarrollada permite a los usuarios interactuar con Maude de manera intuitiva, ejecutar comandos, gestionar módulos y realizar entregas de ejercicios. Además, la interfaz fue diseñada para ser accesible y fácil de usar, incluso para aquellos que no están familiarizados con Maude o con el desarrollo web en general. El resultado es una plataforma robusta, donde todas las funcionalidades clave funcionan como se esperaba, y el rendimiento es consistente bajo diversas cargas de trabajo.
3. **Implementar un sistema de gestión de módulos y versiones**: Uno de los objetivos más ambiciosos del proyecto fue crear un sistema que no solo permitiera a los usuarios gestionar sus propios módulos de Maude, sino también versionarlos,

modificarlos y compararlos de manera similar a un sistema de control de versiones como Git. Esta funcionalidad es crucial para usuarios avanzados que requieren un control granular sobre el desarrollo y evolución de sus módulos. Se implementó una interfaz que permite a los usuarios crear nuevas versiones de sus módulos, realizar cambios y comparar versiones anteriores con las actuales, todo de manera visual y accesible. Esta característica no solo facilita la gestión del código, sino que también promueve mejores prácticas en el desarrollo con Maude.

4. **Crear un sistema de entregas y correcciones**: El sistema de entregas y correcciones fue diseñado para que los usuarios pudieran enviar ejercicios y recibir retroalimentación detallada por parte de los administradores. Este sistema incluye la funcionalidad de marcar comandos como correctos o incorrectos durante las sesiones de chat, y de generar informes que se pueden enviar para revisión. Los administradores, por su parte, tienen la capacidad de revisar estas entregas, añadir notas y devolver correcciones. Este flujo de trabajo garantiza una experiencia educativa completa, donde los usuarios pueden aprender y mejorar a través de un proceso estructurado de entrega y corrección de tareas.
5. **Asegurar una experiencia de usuario fluida**: Desde el inicio, se priorizó la creación de una experiencia de usuario que fuera no solo funcional, sino también intuitiva y agradable. Esto se logró mediante la implementación de un sistema de chat, que permite a los usuarios seguir sus interacciones con Maude en un formato conversacional. Además, se desarrollaron funcionalidades de búsqueda y filtros que permiten a los usuarios localizar rápidamente los comandos y módulos que necesitan, mejorando así la eficiencia de uso de la plataforma. Las pruebas de usuario confirmaron que la plataforma es fácil de navegar y que las funcionalidades son accesibles incluso para aquellos con poca experiencia técnica.

En resumen, los objetivos planteados al inicio del proyecto se han cumplido satisfactoriamente. La plataforma desarrollada no solo logra una integración efectiva entre Maude y Python, sino que también ofrece una interfaz completa y funcional para la gestión de módulos, la entrega de ejercicios, y la corrección de los mismos. Las

pruebas realizadas han demostrado que el sistema es robusto, eficiente y cumple con los requisitos establecidos.

4.2 Trabajo a Futuro

Aunque se han alcanzado los objetivos iniciales, el proyecto presenta diversas oportunidades para su futura evolución. Algunas de las posibles mejoras y funcionalidades adicionales que podrían implementarse en versiones futuras incluyen:

1. **Llevar la cuenta de los errores y mostrar estadísticas**: Una funcionalidad que podría ser de gran valor es el registro de los errores que los usuarios cometen al ejecutar comandos en Maude. Esto podría incluir errores de sintaxis, errores lógicos, y otros problemas comunes. La recopilación de estos datos permitiría generar estadísticas que podrían ser útiles tanto para los usuarios, al ayudarles a identificar patrones en sus errores, como para los administradores, que podrían utilizar esta información para mejorar los materiales de aprendizaje o ajustar las funcionalidades de la plataforma. Las estadísticas podrían mostrar, por ejemplo, cuáles son los errores más comunes o cómo han disminuido los errores de un usuario específico a lo largo del tiempo, promoviendo así un aprendizaje más efectivo.
2. **Poder hacer ejercicios colaborativos**: Actualmente, la plataforma permite a los usuarios trabajar de manera individual en sus ejercicios y módulos. Sin embargo, una expansión lógica sería permitir la colaboración en tiempo real, donde varios usuarios pudieran trabajar juntos en la resolución de un ejercicio o en el desarrollo de un módulo. Esto podría lograrse mediante la creación de sesiones colaborativas, en las que varios usuarios pueden editar y ejecutar comandos simultáneamente, viendo en tiempo real las contribuciones de cada participante. Esta funcionalidad no solo fomentaría el trabajo en equipo, sino que también podría ser de gran utilidad en un entorno educativo, donde los estudiantes pueden aprender unos de otros.
3. **Implementar un sistema de notificaciones**: Un aspecto que podría mejorar la interacción y el flujo de trabajo en la plataforma es la implementación de un sistema de notificaciones. Esto permitiría a los usuarios recibir alertas cuando, por

ejemplo, sus entregas han sido corregidas o cuando se han realizado cambios importantes en los módulos que están utilizando. Las notificaciones podrían ser visuales dentro de la plataforma, enviadas por correo electrónico, o incluso mediante notificaciones push si se decide desarrollar una aplicación móvil en el futuro. Este sistema aseguraría que los usuarios estén siempre al tanto de los eventos importantes, mejorando así la eficiencia y el compromiso con la plataforma.

4. **Ampliar el conjunto de comandos disponibles**: Aunque la plataforma ya soporta una serie de comandos básicos y avanzados de Maude, siempre existe la posibilidad de ampliar este conjunto para cubrir más funcionalidades del lenguaje. Esto podría incluir comandos específicos para ciertas aplicaciones de Maude que no se implementaron en la versión inicial de la plataforma. Ampliar el conjunto de comandos disponibles haría que la plataforma sea aún más útil para una gama más amplia de usuarios, especialmente aquellos que utilizan Maude para aplicaciones avanzadas o especializadas.
5. **Mejorar la documentación y tutoriales**: Finalmente, para asegurar que los nuevos usuarios puedan aprovechar al máximo la plataforma, sería beneficioso desarrollar una documentación más completa y tutoriales interactivos. Estos recursos podrían guiar a los usuarios a través de las funcionalidades de la plataforma, desde los conceptos básicos hasta las características más avanzadas. Además, los tutoriales interactivos permitirían a los usuarios aprender haciendo, con ejercicios prácticos que refuercen los conocimientos adquiridos. Este enfoque no solo ayudaría a los usuarios a familiarizarse rápidamente con la plataforma, sino que también podría reducir la carga de soporte al cliente, ya que los usuarios tendrían acceso a recursos de autoaprendizaje.

Estas mejoras y expansiones no solo incrementarían la funcionalidad y la usabilidad de la plataforma, sino que también abrirían nuevas posibilidades para su uso en diferentes contextos, ya sea en entornos educativos, investigación o aplicaciones industriales. Continuar con el desarrollo de estas áreas asegurará que la plataforma se mantenga relevante y útil para sus usuarios en el futuro.

Conclusions and Future Work

Conclusions

At the beginning of this project, a series of objectives were established to guide the development and ensure that the final platform met the expectations set. Below is a detailed account of how these objectives were achieved:

1. **Integrating Maude with Python**: The first and most crucial objective was to achieve the integration of Maude with Python through bindings. This step was essential to allow the execution of Maude commands from a web interface, making it more accessible and modern. After thorough research into how Maude bindings work in Python, functionality was developed to translate Maude commands into a format understandable by Python and execute these commands within the platform. This integration was rigorously tested to ensure smooth, error-free communication between Maude and the web interface, thus fulfilling this fundamental objective.
2. **Developing a robust and functional web platform using Django**: Since Django was a new framework for the development team, time was invested in understanding its architecture and best practices. Django was chosen for its ability to efficiently handle complex web applications. The developed platform allows users to interact with Maude intuitively, execute commands, manage modules, and submit assignments. Additionally, the interface was designed to be accessible and user-friendly, even for those unfamiliar with Maude or web development. The result is a robust platform where all key functionalities work as expected, and performance remains consistent under various workloads.
3. **Implementing a module and version management system**: One of the most ambitious goals of the project was to create a system that allowed users not only to manage their own Maude modules but also to version, modify, and compare them, similar to a version control system like Git. This functionality is crucial for advanced users who require granular control over the development and evolution of their modules. An interface was implemented that allows users to

create new versions of their modules, make changes, and compare previous versions with current ones in a visual and accessible way. This feature not only facilitates code management but also promotes best practices in Maude development.

4. **Creating a submission and correction system**: The submission and correction system was designed to allow users to submit exercises and receive detailed feedback from administrators. This system includes the ability to mark commands as correct or incorrect during chat sessions and to generate reports that can be submitted for review. Administrators, in turn, have the ability to review these submissions, add notes, and return corrections. This workflow ensures a complete educational experience where users can learn and improve through a structured process of task submission and correction.
5. **Ensuring a smooth user experience**: From the outset, priority was given to creating a user experience that was not only functional but also intuitive and enjoyable. This was achieved through the implementation of a chat system, allowing users to follow their interactions with Maude in a conversational format. Additionally, search and filter functionalities were developed to enable users to quickly locate the commands and modules they need, thus improving the platform's usability. User testing confirmed that the platform is easy to navigate and that functionalities are accessible even to those with limited technical experience.

In summary, the objectives set at the beginning of the project have been successfully met. The platform not only achieves effective integration between Maude and Python but also provides a complete and functional interface for module management, exercise submission, and correction. Testing has shown that the system is robust, efficient, and meets the established requirements.

Future work

Although the initial objectives have been achieved, the project presents several opportunities for future evolution. Some potential improvements and additional features that could be implemented in future versions include:

1. **Tracking errors and displaying statistics**: A valuable feature could be the logging of errors users encounter when executing commands in Maude. This could include syntax errors, logical errors, and other common issues. Collecting this data would allow for the generation of statistics that could be useful both for users, by helping them identify patterns in their errors, and for administrators, who could use this information to improve learning materials or adjust platform functionalities. The statistics could display, for example, the most common errors or how a specific user's errors have decreased over time, thereby promoting more effective learning.
2. **Enabling collaborative exercises**: Currently, the platform allows users to work individually on their exercises and modules. However, a logical expansion would be to enable real-time collaboration, where several users could work together on solving an exercise or developing a module. This could be achieved by creating collaborative sessions, where multiple users can edit and execute commands simultaneously, seeing each participant's contributions in real time. This functionality would not only foster teamwork but could also be of great value in an educational setting, where students can learn from each other.
3. **Implementing a notification system**: One aspect that could enhance the interaction and workflow on the platform is the implementation of a notification system. This would allow users to receive alerts when, for example, their submissions have been graded or when important changes have been made to the modules they are using. Notifications could be visual within the platform, sent via email, or even through push notifications if a mobile application is developed in the future. This system would ensure that users are always aware of important events, thus improving efficiency and engagement with the platform.
4. **Expanding the set of available commands**: Although the platform already supports a range of basic and advanced Maude commands, there is always the possibility to expand this set to cover more language functionalities. This could include specific commands for certain Maude applications that were not implemented in the platform's initial version. Expanding the available commands

would make the platform even more useful for a wider range of users, particularly those using Maude for advanced or specialized applications.

5. **Improving documentation and tutorials**: Lastly, to ensure that new users can make the most of the platform, it would be beneficial to develop more comprehensive documentation and interactive tutorials. These resources could guide users through the platform's functionalities, from basic concepts to more advanced features. Additionally, interactive tutorials would allow users to learn by doing, with practical exercises to reinforce acquired knowledge. This approach would not only help users become familiar with the platform quickly but could also reduce customer support demands, as users would have access to self-learning resources.

These improvements and expansions would not only increase the platform's functionality and usability but also open new possibilities for its use in different contexts, whether in educational settings, research, or industrial applications. Continuing to develop these areas will ensure that the platform remains relevant and useful to its users in the future.

BIBLIOGRAFÍA

[1] Rubio, D. (2017). *Beginning Django : web application development and deployment with Python*. Apress. https://doi.org/10.1007/978-1-4842-2787-9

[2] Rubio, R. (2022). Maude as a Library: An Efficient All-Purpose Programming Interface. *Proceedings of the International Workshop on Rewriting Logic and Its Applications (WRLA)*, 274-294.

[3] Baker, M. (2022). *Secure web application development : a hands-on guide with Python and Django* ([First edition]). Apress. https://doi.org/10.1007/978-1-4842-8596-1