CHATGPT, ¡AYÚDAME CON SQL!
CHATGPT, ¡HELP ME WITH SQL!

Trabajo Fin de Grado
Curso 2024-2025

Autor
MARTA TRUJILLANO ROJANO
Nota:5

DIRECTOR
Fernando Sáenz Pérez

Doble Grado en Ingeniería Informática y Administración de Empresas
Facultad de Informática
Universidad Complutense de Madrid

CHATGPT, ¡AYÚDAME CON SQL!
CHATGPT, ¡HELP ME WITH SQL!

Trabajo de Fin de Grado en Ingeniería Informática

Autor

MARTA TRUJILLANO ROJANO

Nota:5

DIRECTOR

Fernando Sáenz Pérez

Convocatoria: Junio 2025

Doble Grado en Ingeniería Informática y Administración de Empresas

Facultad de Informática

Universidad Complutense de Madrid

26 de MAYO de 2025

AGRADECIMIENTOS

Primeramente, quiero agradecer a mi tutor Fernando por todo el tiempo dedicado durante estos dos cursos académicos en la orientación y ayuda brindadas para la realización de este trabajo de fin de grado, y por todo el aprendizaje que me llevo conmigo.

Además, me gustaría agradecer a mi familia por el apoyo incondicional a lo largo de esta etapa.

Resumen

Este Trabajo de Fin de Grado consiste en la ampliación de la plataforma DESweb utilizada por la Universidad Complutense de Madrid con el objetivo de incorporar herramientas de Inteligencia Artificial que mejoren la experiencia de aprendizaje y colaboración entre estudiantes y docentes.

La principal novedad es la integración de un sistema de mensajería que permite a los miembros de un grupo comunicarse directamente dentro de la propia plataforma, facilitando así la resolución de dudas de forma ágil y centralizada.

Además, se ha incorporado la posibilidad de interactuar con un asistente inteligente que analiza y responde preguntas relacionadas con consultas anteriores, permitiendo al usuario recibir orientación sin necesidad de abandonar la aplicación.

Este proyecto abarca tanto el rediseño de la interfaz para integrar estas nuevas funcionalidades como la implementación de un sistema de comunicación en tiempo real y la conexión con un asistente conversacional especializado en bases de datos.

Palabras clave

SWI Prolog, ChatGPT, DESweb, Universidad Complutense de Madrid

Abstract

This Final Degree Project focuses on enhancing the DESweb platform used by the Complutense University of Madrid by integrating artificial intelligence tools to improve the learning and collaboration experience for both students and faculty.

The main feature added is a messaging system that enables group members to communicate directly within the platform, making it easier to resolve questions efficiently without leaving the environment.

Additionally, an intelligent assistant has been integrated to analyze and respond to user queries related to their recent database activity, offering support without requiring users to switch to external tools.

The project involves both the redesign of the user interface to incorporate these new features and the implementation of real-time communication and AI-driven assistance within the platform.

Keywords

SWI Prolog, Chat GPT, DESweb, Universidad Complutense de Madrid

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 – Introducción

1.1. Motivación

La creciente importancia que la Inteligencia Artificial (IA) ha tomado en los diferentes campos presentes en el día a día presenta un gran cambio en la metodología de trabajo, en la eficiencia y el aprendizaje de los diferentes sectores que conforman la sociedad actual. Es por ello por lo que adaptarse a estos nuevos cambios es imprescindible para no quedarse obsoleto, ya que esta nueva herramienta ha supuesto un antes y un después en la forma de trabajar, aprender y como fuente de información.

En este sentido, la incorporación de un chat tanto entre usuarios como conectado con ChatGPT dentro de aplicaciones educativas responde a la necesidad de ofrecer una asistencia inmediata, personalizada y contextualizada.

Este tipo de integración no solo facilita el acceso a la información y resolución de dudas en tiempo real, sino que también contribuye a una experiencia de usuario más dinámica, interactiva y eficiente.

Adicionalmente, la inclusión no solo de la comunicación con GPT, sino también la posibilidad de comunicarse con otros usuarios da una experiencia más enriquecedora y sencilla a la hora de aprender.

Por todo ello, la motivación principal de este trabajo es hacer una herramienta en la que el aprendizaje sea más accesible, dinámico y personalizado, ofreciendo una ayuda directa dentro de la propia aplicación, sin necesidad de acudir a recursos externos. También me ha brindado la oportunidad de aprender cómo adaptar una herramienta de trabajo a este nuevo movimiento.

1.2. Antecedentes

En los últimos años la inteligencia artificial ha pasado a ser una herramienta del día a día presente en múltiples aspectos de nuestra vida cotidiana. Uno de los avances más significativos en este campo ha sido el desarrollo de modelos de lenguaje como ChatGPT y otras plataformas similares como Copilot o Chat Gemini.

ChatGPT es un sistema capaz de comprender y generar texto de manera coherente, lo que le permite mantener conversaciones, responder preguntas, redactar textos, programar, traducir idiomas y mucho más. Gracias a su utilidad y a la forma de facilitar problemas diarios, se ha integrado en una amplia variedad de sectores: desde el soporte técnico en empresas, la creación de contenido, en forma de asistente o la automatización de tareas administrativas. Sin embargo, uno de los campos donde su impacto está siendo más transformador es el de la educación.

En el ámbito educativo, ChatGPT se está utilizando como asistente virtual para estudiantes y profesores. Varias plataformas de aprendizaje están implementando la asistencia de esta herramienta para diferentes tareas como evaluación, generación y resolución de ejercicios, planificación docente y sistemas de tutoría personalizada. Esto mejora la calidad y accesibilidad de la enseñanza. Su capacidad generar respuestas en tiempo real y explicar conceptos complejos de forma sencilla para adaptarse a los distintos tipos de usuario lo convierte en un recurso valioso tanto dentro como fuera del aula.

Esta tecnología evoluciona cada vez más rápido y crea una nueva visión para aprender y enseñar. Las herramientas de inteligencia artificial serán el futuro de la educación y de otros ámbitos, proporcionando grandes ventajas sin reemplazar el factor humano, sino complementándolo, potenciándolo y creando aprendizajes más eficaces y rápidos.

Algunos ejemplos de la incorporación de ChatGPT a una plataforma educativa:

- **Lessonspace:** es una plataforma de tutoría *online* que actualmente utiliza ChatGPT para ofrecer asistencia automatizada en tiempo real y así permitir a los estudiantes recibir tutorías personalizadas sobre diferentes temas y generar ejercicios basados en el contenido de las lecciones. Además, crea un entorno colaborativo entre los estudiantes mediante chats grupales que permiten la inclusión de la IA[12].
- **iDoceo:** una plataforma de planificación y evaluación para docentes que, en su última versión, permite la asistencia de ChatGPT conectada mediante una clave API. Los docentes hacen uso de los asistentes virtuales para generar contenido como ejercicios y exámenes, todo dentro de la misma aplicación[13].

DESweb es una herramienta de consulta de bases de datos (BBDD) desarrollada por la UCM que permite ejecutar consultas de BBDD a través de una interfaz web. Actualmente, la interfaz permite la creación, modificación y consultas de diferentes tablas en la BBDD. Además, incorpora algunas funciones adicionales como la posibilidad de ver la estructura de la BBDD en formato de tablas, una interfaz gráfica de depuración que cuenta con una representación gráfica del estado de la base de datos, creación de directorios y la posibilidad de modificarlos.

Para el desarrollo de este proyecto se ha hecho uso de la popular herramienta Visual Studio Code (VS Code), que, a día de hoy, es uno de los editores de código más versátiles ya que permite agregar funcionalidades específicas mediante

extensiones. Una de las más destacadas en la actualidad es GitHub Copilot, una herramienta de inteligencia artificial desarrollada por GitHub y OpenAI que sugiere código en tiempo real, ayudando a los desarrolladores a escribir más rápido y con mayor eficiencia.[18]

1.3. Objetivos

El objetivo de este proyecto es realizar una extensión del sistema DESweb actualmente utilizado por la Universidad Complutense de Madrid. Para ello se han añadido nuevas funcionalidades de comunicación entre usuarios y de asistencia automatizada haciendo uso de la principal Inteligencia artificial a día de hoy, ChatGPT.

Esta herramienta es empleada con el fin de mejorar la experiencia de aprendizaje y la resolución de dudas acerca de bases de datos haciendo uso de los lenguajes de SWI-Prolog y JavaScript. Para conseguir esto nos centraremos en los siguientes puntos:

- Modificación de la interfaz de la aplicación actual para incluir una funcionalidad de chat en tiempo real entre los usuarios pertenecientes al mismo grupo.
- Implementación de una posible conversación con ChatGPT dentro del mismo entorno de DESweb, permitiendo, de esta forma, que los usuarios puedan hacer consultas directamente desde la aplicación sin necesidad de acceder a plataformas externas.
- El desarrollo de un sistema que realiza el envío automático a ChatGPT de la última consulta realizada en el sistema SWI-Prolog, para que éste pueda proporcionar asistencia personalizada teniendo en cuenta la estructura de la base de datos.
- Aplicar estilos CSS personalizados que se adapten a la interfaz actual, manteniendo la coherencia visual con el diseño original de DESweb.
- Verificar el correcto funcionamiento de las nuevas funcionalidades mediante pruebas de usuario, validando la utilidad del sistema en entornos académicos.

1.4. Plan de trabajo

La metodología es la adaptación del código actual para poder realizar las nuevas funcionalidades planteadas.

Para ello, se hará uso de GitHub una plataforma de trabajo destinada a código abierto donde se hará investigación sobre el uso de SWI Prolog y su conexión con ChatGPT.

Además, para poder abordar estos objetivos se ha realizado una búsqueda bibliográfica exhaustiva acerca de los diferentes lenguajes utilizados en el desarrollo de la aplicación.

El trabajo se divide en las siguientes etapas

- **Etapa de diseño de especificaciones:** el proceso de diseño se ha basado en una serie de reuniones con el director del trabajo, en las que se han definido y ajustado los objetivos del proyecto. Estas sesiones han permitido establecer un camino claro, así como alinear expectativas y metodologías de trabajo.

- **Etapa de investigación**: durante esta fase, se ha llevado a cabo una investigación exhaustiva sobre los lenguajes de programación a emplear, así como una comprensión profunda del código existente. Se han analizado proyectos similares desarrollados en SWI-Prolog y se han explorado soluciones que integran la interacción cliente-servidor con tecnologías como ChatGPT.
  Su objetivo: responder a la pregunta clave: *¿Qué voy a hacer para alcanzar los objetivos propuestos y cómo voy a hacerlo?*

- **Etapa de desarrollo**: con los recursos definidos y comprendidos, se dio inicio al desarrollo del proyecto, se modificó e implementó código fuente para integrar nuevas funcionalidades, utilizando los conocimientos adquiridos previamente, desde un enfoque práctico y orientado a resultados, incorporando las herramientas y técnicas más adecuadas ya identificadas durante la fase de investigación.

- **Etapa de conclusiones:** primero se realizó un análisis retrospectivo sobre el trabajo realizado, tanto de los objetivos cumplidos como los que no se han podido cumplir. Así mismo se han evaluado posibles funcionalidades a incluir en le futuro.

- **Etapa de documentación:** El trabajo se ha documentado el trabajo detallando los procedimientos implementados, los módulos modificados y las decisiones técnicas adoptadas a lo largo del desarrollo. También se ha verificado el correcto funcionamiento del sistema, asegurando la coherencia entre los objetivos iniciales y los resultados obtenidos.

En la figura 1 se puede ver la distribución temporal de las diferentes etapas del trabajo comentadas anteriormente:

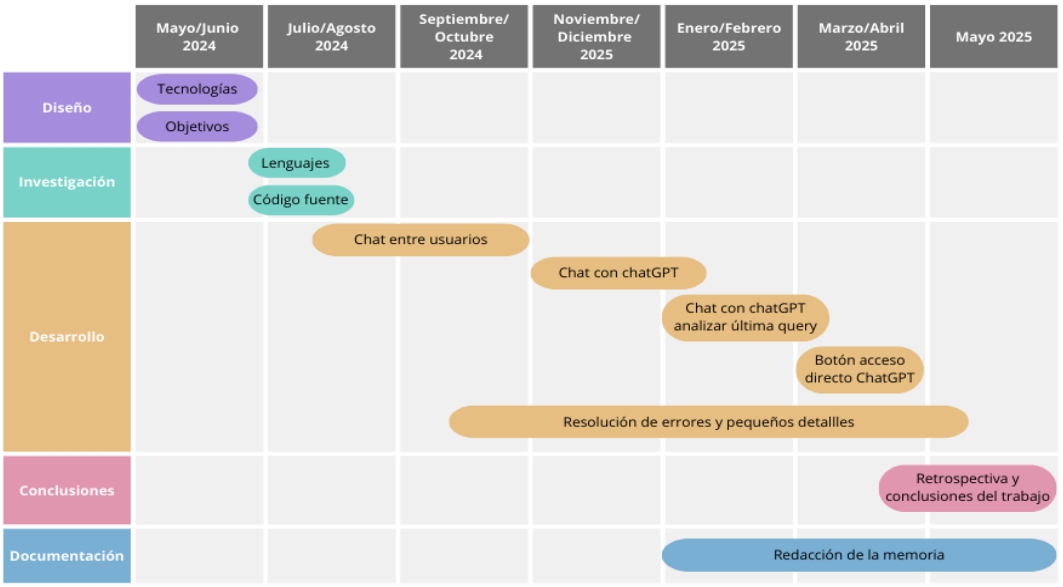

Figura 1 - Diagrama de Gantt Español

1.5. 1.5. Organización de la memoria

Este trabajo se estructura en los siguientes capítulos, cada uno de ellos con un propósito específico para estructurar el desarrollo del proyecto de manera clara y progresiva:

- **Capítulos 1 y 2. Introducción:** se presenta el contexto del proyecto, sus objetivos, la motivación que lo impulsa y la metodología empleada. En el primer capítulo se presentan estos puntos en idioma castellano y en el capítulo dos se exponen los mismos puntos traducidos al inglés.

- **Capítulo 3. Tecnologías utilizadas:** se describen las tecnologías y lenguajes empleados. Se justifica el uso que se le da a cada una de ellas en función de su utilidad y adaptación a las necesidades del proyecto.

- **Capítulo 4. Diseño de la interfaz e Implementación de las nuevas funcionalidades:** se describen los desarrollos realizados, incluyendo los cambios implementados tanto en Prolog, JavaScript y la integración de la API de ChatGPT. También se muestra cómo se ha construido la interfaz de usuario y cómo se gestionan las interacciones.

- **Capítulo 5. Diseño de la interfaz e Implementación de las nuevas funcionalidades**: explica cómo se crea la relación cliente-servidor entre los lenguajes de JavaScript y SWI Prolog y cómo se ha implementado esta relación dentro del código para crear la comunicación entre los distintos usuarios.

- **Capítulos 6 y 7. Conclusiones y trabajo futuro:** se recogen las conclusiones derivadas del trabajo, se evalúa el grado de cumplimiento de los objetivos

planteados y se proponen futuras mejoras. El capítulo 6 muestra estos puntos en castellano y el punto 7 traduce el punto anterior al inglés.

- **Bibliografía:** se listan todas las fuentes consultadas durante el desarrollo del proyecto, incluyendo documentación técnica, libros, artículos y páginas web relevantes.

Capítulo 2 - Introduction

2.1. Motivation

The growing importance that Artificial Intelligence has gained in various fields present in our daily lives represents a major shift in work methodology, efficiency, and learning across different sectors of modern society. For this reason, adapting to these new changes is essential to avoid falling behind or becoming obsolete, as this new tool has marked a before and after in the way we work, learn, and access information.

In this context, the incorporation of a chat connected to ChatGPT within educational applications addresses the need to offer immediate, personalized, and contextualized assistance. This type of integration not only facilitates access to information and real-time doubt resolution, but also contributes to a more dynamic, interactive, and efficient user experience. Additionally, enabling communication not only with GPT but also with other users provides a richer and simpler learning experience.

For all these reasons, the main motivation of this project is to make learning more accessible, dynamic, and personalized, offering direct support within the application itself without needing to resort to external resources. It is also an opportunity to learn how to adapt a work tool to this new technological movement.

2.2. Background

In recent years, Artificial Intelligence has become a daily-use tool present in multiple aspects of our lives. One of the most significant advancements in this field has been the development of language models such as ChatGPT and other similar platforms like Copilot or Chat Gemini. ChatGPT is a system capable of understanding and generating coherent text, which allows it to hold conversations, answer questions, draft content, code, translate languages, and much more.

Thanks to its usefulness and ability to simplify everyday problems, ChatGPT has been integrated into a wide variety of sectors: from technical support in companies, to content creation, as a virtual assistant, and for automating administrative tasks. However, one of the fields where its impact is proving most transformative is education.

In the educational field, ChatGPT is being used as a virtual assistant for students and teachers. Various learning platforms are implementing this tool's assistance for tasks such as assessment, exercise generation and resolution, lesson planning, and personalized tutoring systems. This improves the quality and accessibility of teaching. Its ability to provide real-time answers and explain complex

concepts in a simple way tailored to different user types makes it a valuable resource both inside and outside the classroom.

This technology continues to evolve rapidly and creates a new vision for learning and teaching. These AI tools promise the future of education and other fields by offering an advantage—not replacing the human factor, but complementing and enhancing it, leading to more effective and faster learning.

Some examples of the integration of ChatGPT into educational platforms include:

- **Lessonspace**, an online tutoring platform that currently uses ChatGPT to provide real-time automated assistance, allowing students to receive personalized tutoring on various topics and generate exercises based on lesson content. It also creates a collaborative environment through group chats that include AI interaction.

- **iDoceo**, a planning and assessment platform for teachers, which in its latest version allows assistance from ChatGPT through an API key. Teachers use virtual assistants to generate content such as exercises and exams directly within the app.

DESweb, a database query tool developed by the Complutense University of Madrid (UCM), which allows database queries through a web interface. Currently, it enables the creation, modification, and querying of different tables in the database, along with additional features like interface color changes, password updates, and debugging/editing tabs. However, it has not yet evolved to incorporate AI assistance to ease its use and learning. This is the next step proposed by this final degree project—a leap toward future evolution.

For this project, the popular tool Visual Studio Code (VS Code) has been used. It is currently one of the most versatile code editors, allowing specific functionalities to be added through extensions. One of the most notable today is GitHub Copilot, an AI tool developed by GitHub and OpenAI that suggests code in real-time, helping developers write faster and more efficiently.

2.3. Objectives

The goal of this project is to develop an extension of the DESweb system currently used by the Complutense University of Madrid. To this end, new functionalities for user communication and automated assistance have been added using today's leading AI tool, ChatGPT. This tool is employed to enhance the learning

experience and assist with questions about databases using **SWI-Prolog** and **JavaScript**.

To achieve this, the project focuses on the following points:

- Modifying the current application interface to include real-time chat functionality among users belonging to the same group.

- Implementing a potential conversation feature with ChatGPT within the DESweb environment itself, allowing users to make queries directly from the application without needing to access external platforms.

- Developing a system that automatically sends the last query made in SWI-Prolog to ChatGPT, enabling personalized assistance based on the database structure.

- Applying custom CSS styles to match the current interface, maintaining visual coherence with DESweb's original design.

- Verifying the proper functioning of the new features through user testing, validating the system's usefulness in academic environments.

2.4. Work plan

The methodology in this work involves adapting the current code to implement the new proposed functionalities. For this, GitHub will be used as a collaborative open-source platform to research SWI-Prolog and its integration with ChatGPT. A literature review will also be conducted on the different languages used in the application's development.

The work is divided into the following stages:

- **Specification design stage**: The design process was based on a series of meetings with the project advisor, during which the project objectives were defined and refined. These sessions helped set a clear path and align expectations and methodologies.

- **Research stage**: This phase involved extensive research on the programming languages to be used, as well as a deep understanding of the existing code. Similar projects developed in SWI-Prolog were analyzed, and solutions integrating client-server interaction with technologies like ChatGPT were explored. The goal was to answer the key question: *What will I do to achieve the proposed objectives, and how will I do it?*

- **Development stage**: With the necessary resources defined and understood, the development phase began. This stage consisted of modifying and implementing the source code to integrate new functionalities, applying the knowledge previously acquired. The focus was practical and results-oriented, using the most appropriate tools and techniques identified during the research phase.

- **Conclusion stage**: A retrospective was conducted to assess both achieved and unachieved goals. Future functionality improvements were also envisioned.

- **Documentation stage**: In this final phase, the work carried out was documented, detailing the implemented procedures, modified modules, and technical decisions made throughout development. The system's proper functioning was also verified, ensuring alignment between initial objectives and final results.

Figure 2 shows the timeline of the different stages of the work discussed earlier:

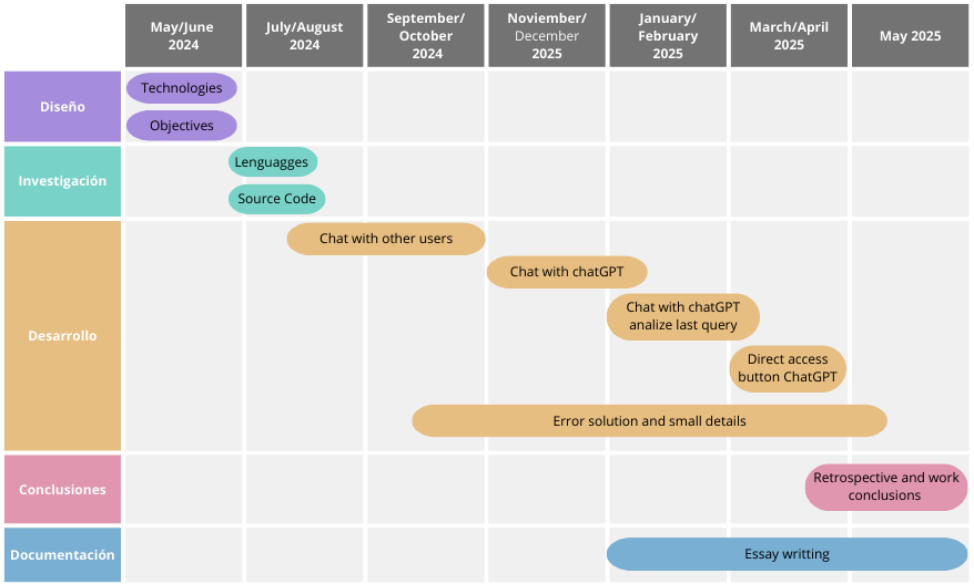

FIGURA 2 - DIAGRAMA DE GANTT INGLÉS

2.5. Structure of the project

This work is organized into the following chapters, each with a specific purpose to clearly and progressively structure the project's development:

- **Chapters 1 and 2. Introduction**
  The project's context, objectives, motivation, and methodology are presented. Chapter 1 is written in Spanish, and Chapter 2 contains the same content translated into English.

- **Chapter 3. Technologies Used**
  This chapter describes the technologies and programming languages used. Their use is justified based on their utility and adaptability to the project's needs.

- **Chapter 4. Interface Design and Implementation of New Features**
  Describes the development work done, including changes made in Prolog, JavaScript, and the integration of the ChatGPT API. It also shows how the user interface was built and how interactions are managed.

- **Chapter 5. Client-Server Interaction**
  This chapter shows how the client-server relationship between JavaScript and SWI-Prolog was created, and how this connection was implemented to enable communication between different users.

- **Chapters 6 and 7. Conclusions and Future Work**
  Presents the conclusions drawn from the project, evaluates the achievement of objectives, and proposes potential improvements.

- **Bibliography**
  Lists all sources consulted during the project's development, including technical documentation, books, articles, and relevant websites.

Capítulo 3 – Herramientas utilizadas

3.1. SWI-Prolog

Prolog es un lenguaje de programación que se basa en la lógica declarativa y su nombre proviene de la expresión "*Programming in Logic*" (Programación en lógica), lo cual quiere decir que este lenguaje se basa en la declaración de condiciones, proposiciones, afirmaciones, etc. El lenguaje fue desarrollado en la década de 1970 por Alain Colmerauer y Philippe Roussel y refleja su enfoque fundamental en la resolución de problemas mediante reglas y hechos.[1]

3.1.1. Hechos

Un hecho es la forma en la que Prolog representa una propiedad o relación entre los objetos a representar. Los hechos determinan si los valores serán verdaderos para un predicado concreto[2]:

```
nombre_predicado(argumentos).
```

Los hechos se pueden dividir en dos formas diferentes:

- Propiedades: las propiedades consisten en un predicado con único argumento que determina la propiedad de un objeto concreto. Ejemplo de sintaxis:

```
prenda(camisa). % Determina que la camisa tiene la propiedad de ser una prenda
```

- Relaciones: en este caso el predicado lleva más de un argumento y por lo tanto expresan la relación entre varios objetos. Ejemplo de sintaxis:

```
color(camisa, verde). % La camisa es de color verde, de esta forma existe una relación entre la camisa y el color verde, expresando una verdad.
```

3.1.1. Reglas

Las reglas determinan la verdad cuando un hecho depende de otro hecho o conjunto de ellos. Es decir, es una combinación de hechos que dan valor a un predicado. Ejemplo de sintaxis[2]:

```
camisa:- prenda
```

De esta forma, camisa es cierta si prenda es cierta.

```
hijo_de(A,B) :- padre_de(B,A). %A es hijo de B si B es padre de A
```

Existen dos tipos de reglas[2]:

- **Conjunciones**: se separan por *comas* y determina en el lenguaje lógico un *AND.* Es decir, una condición se cumple si todos los hechos se cumplen.

- **Disyunciones**: se separan por *punto y coma*. En el lenguaje de la lógica esto hace referencia a un OR, es decir, una condición se cumple si alguno de los hechos se cumple.

SWI-Prolog es una implementación del lenguaje Prolog. Proporciona un mecanismo para la instalación de complementos llamados packs. Los packs pueden instalarse desde un archivo, repositorio GIT o URL usando *pack_install/1*. Los packs se utilizan para compartir código dentro de la comunidad y han desarrollado ecosistemas para tratar con tipos, corrutinas, etc.

SWI-Prolog ofrece una variedad de herramientas de desarrollo que pueden combinarse según sea necesario.

3.2. JavaScript

JavaScript se consolidó como el lenguaje principal para el desarrollo web tras su creación en 1995 por Brendan Eich. Es un lenguaje de alto nivel que se utiliza para hacer interfaces web interactivas para el usuario [3].

Es el intermediario entre el usuario y la página web, la cual se estructura como un DOM (Document Objetc Model) lo que permite la actualización de la página web en tiempo real, además de la creación de experiencias visuales atractivas. Este lenguaje facilita la incorporación de efectos visuales, que hacen que la navegación resulte más atractiva y profesional.

Por último, JavaScript también actúa como intermediario entre el navegador y el servidor, utilizando herramientas que actúan como servidores, como por ejemplo SWI-Prolog, lo que hace posible intercambiar datos de manera asíncrona y la página puede reaccionar a nuevas informaciones en tiempo real, lo que es clave en aplicaciones modernas como chats en tiempo real, sistemas de notificaciones o paneles interactivos.

3.3. CSS y HTML

HTML y CSS son herramientas complementarias e indispensables para el diseño web, HTML crea el contenido que se muestra en la interfaz y CSS aplica los estilos para el diseño de este contenido [16].

3.4. ChatGPT

La versión utilizada para este proyecto es la GPT-3 (Generative Pre-trained Transformer 3), un modelo de lenguaje desarrollado por OpenAI. Es la tercera generación de la familia de modelos GPT, un tipo de red neuronal especialmente eficiente para procesar lenguaje natural. GPT-3 fue, en su momento, el modelo más potente de IA. Su conocimiento viene del autoaprendizaje que realiza a través de internet. Su capacidad para generar texto coherente, responder preguntas, traducir idiomas, escribir código y más, lo hizo revolucionario para la sociedad[15].

Aunque ahora mismo la versión más actual es la versión GPT-4, la versión anterior es lo suficientemente potente para realizar las búsquedas y generar respuestas de gran utilidad.

Para integrar ChatGPT en cualquier aplicación es imprescindible adquirir una API KEY por la que el usuario se pueda conectar para poder interactuar con esta herramienta. El objetivo de esta clave es identificar al cliente y poder detectar qué aplicación está intentando acceder a los recursos, lo que hace que la conexión entre la herramienta y el servidor sea segura.

Cuando se utiliza ChatGPT mediante su API, se está accediendo a una interfaz de programación de aplicaciones ofrecida por OpenAI, la empresa creadora de esta tecnología. A través de esta API, se pueden enviar solicitudes con texto y recibir respuestas generadas por el modelo, lo que permite integrar ChatGPT en diferentes entornos, desde aplicaciones web hasta asistentes virtuales o plataformas educativas[11].

Capítulo 4 – Diseño de la interfaz e Implementación de las nuevas funcionalidades

En este apartado se detalla el diseño de la interfaz web de la plataforma DESweb, así como los desarrollos realizados para integrar las nuevas funcionalidades de mensajería, tanto entre usuarios como con ChatGPT. Se describen las funciones incorporadas, su reflejo en la interfaz gráfica y los archivos modificados o creados para hacerlo posible:

Durante el desarrollo de este proyecto se ha trabajado sobre los siguientes archivos existentes en la aplicación:

- **user_interface.pl:** archivo principal de los usuarios de Prolog encargado de generar su interfaz gráfica enfocada en interaccionar con la base de datos. En él se han añadido los botones y paneles correspondientes al sistema de chat, además de las funciones necesarias para actuar como servidor de mensajería.
- **database_panel.js:** gestiona todas las operaciones relacionadas con la base de datos (consultas, creación de tablas, etc.). Se ha ampliado su funcionalidad para extraer la estructura de la base de datos y enviarla a ChatGPT, lo que permite generar respuestas más precisas y contextualizadas.

También se han añadido los siguientes componentes nuevos:

- **chat.js:** script que gestiona la interacción del usuario con el chat, incluyendo el envío y recepción de mensajes. Contiene la lógica para actuar como cliente de mensajería.
- **prolog2gpt.pl**: archivo reusado que se comunica con ChatGPT mediante la API Key proporcionada por OpenAI, permitiendo recibir respuestas automáticas desde el modelo de lenguaje.
- **chat_style.css**: hoja de estilos dedicada a la apariencia visual del chat, diseñada para que sea intuitivo, atractivo y fácil de utilizar.

Al iniciar DESweb, el usuario accede a una pantalla inicial de selección de usuario, tal como se muestra en la figura 3:

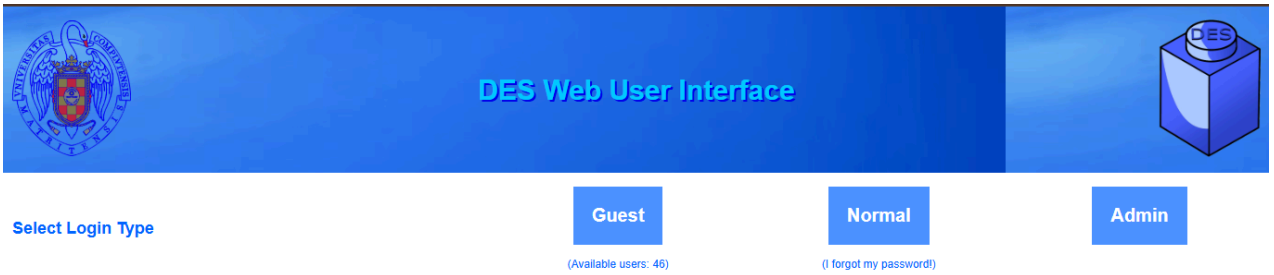

FIGURA 3 - PÁGINA INICIAL DE SELECCIÓN DE USUARIO

Este proyecto centra la atención en el usuario *"Normal"* ya que es el que es el usuario que permite a alumnos y profesores interactuar con la base de datos.

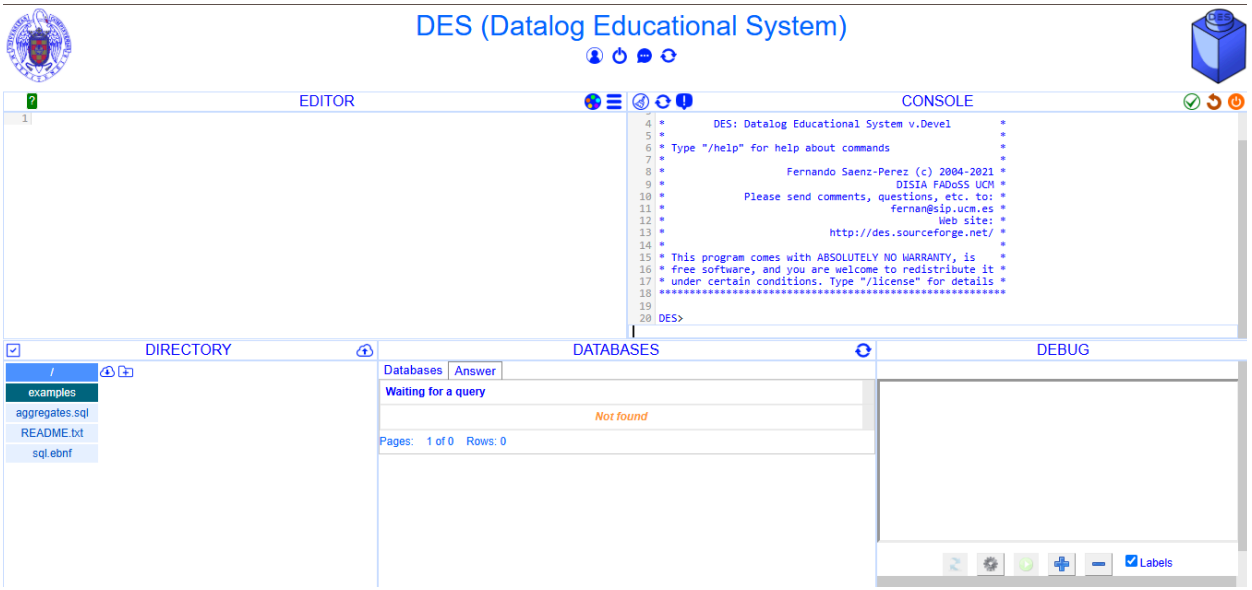

FIGURA 4 - INTERFAZ PRINCIPAL

Una vez introducidos el nombre de usuario y la contraseña, se accede a la interfaz principal (figura 4), que se organiza en seis paneles funcionales:

- **Banner**: ubicado en la parte superior, muestra el título del sistema (DES - Datalog Educational System) y una barra de herramientas con accesos rápidos. Incluye botones para cambiar contraseña, cerrar sesión, refrescar la página y acceder al nuevo sistema de chat.

- **Consola:** situada en el centro de la interfaz, permite ejecutar consultas SQL y realizar operaciones sobre la base de datos (crear/eliminar tablas, insertar datos, etc.).
- **Edición:** adyacente a la consola, permite abrir y modificar archivos del sistema.
- **Directorio**: proporciona una visión jerárquica de los archivos y carpetas del sistema. Incluye opciones para crear nuevos archivos/directorios, insertar archivos y descargarlos.
- **Base de datos:** visualiza las bases de datos disponibles y sus tablas, así como los datos contenidos en cada una.
- **Panel de depuración:** su función principal es realizar una depuración declarativa. El panel muestra una representación visual de la base de datos mediante un grafo compuesto por nodos.

En el panel del banner se muestra la barra de tareas con los siguientes botones de izquierda a derecha respectivamente:

- **Cambio de contraseña**: botón que sirve para hacer un cambio de contraseña (figura 5)

Figura 5 – Botón cambio contraseña

- **Cerrar sesión**: este botón (figura 6) cierra la sesión del usuario actual y vuelve a la página principal (figura 3).

Figura 6 – Botón cerrar sesión

- **Chat**: nuevo botón creado en este proyecto para poder conversar en tiempo real tanto con el resto de los usuarios del mismo grupo, como con ChatGPT. (figura 7)

Figura 7 – Botón chat

- **Refrescar página**: refresca la página y desaparece todo lo realizado en la sesión actual (figura 8).

Figura 8 – Botón refrescar página

Botones en el panel de edición:

- **Cambio de tema de la interfaz**: este botón cambia el tema de la interfaz y tiene la posibilidad de ponerlo en distintos modos como el modo noche, moderno, clásico, etc… (figura 9)

Figura 9 – Botón cambio tema interfaz

- **Menú**: este menú contiene las opciones principales del panel del directorio, como abrir un archivo, guardarlo, etc… (figura 10)

Figura 10 – Botón menú

También podemos encontrar los siguientes botones en el panel de la consola:

- **Limpieza**: este botón limpia todo el contenido enviado a través de la consola. Todas las consultas realizadas. (figura 11)

Figura 11 – Botón limpieza

- **Refrescar panel de consola:** refresca la consola de consultas (figura 12).

Figura 12 – Botón refrescar consola

- **Análisis de ChatGPT**: nuevo botón incorporado en esta nueva versión realizada para este trabajo de fin de grado que envía directamente a ChatGPT

la última consulta realizada y abre un chat con la respuesta de ChatGPT, en la que se podrá continuar la conversación (figura 13).

Figura 13 – Botón análisis chat GPT

Si continuamos a la parte inferior de la interfaz podemos encontrar los paneles mencionados anteriormente, directorio, bases de datos y consola de depuración.

Como hemos visto en la figura 3, se ha añadido un nuevo icono en la barra superior de la interfaz, que contiene el botón para abrir el chat como se puede ver en la figura 14.

Figura 14 - Para que este nuevo icono aparezca en la pantalla de la aplicación, se ha añadido dicha imagen dentro del panel principal: user_interface.pl haciendo uso del predicado: change_password_form(ParamString, User, Realm, _Logout, _Origin, TDs).

Al pulsar este nuevo botón aparecerá el siguiente panel (figura 15) con el listado de todos los usuarios pertenecientes al mismo grupo que el del usuario que ha iniciado sesión. Además de dos chats adicionales en lo alto de la lista que hacen referencia a dos instancias distintas de ChatGPT.

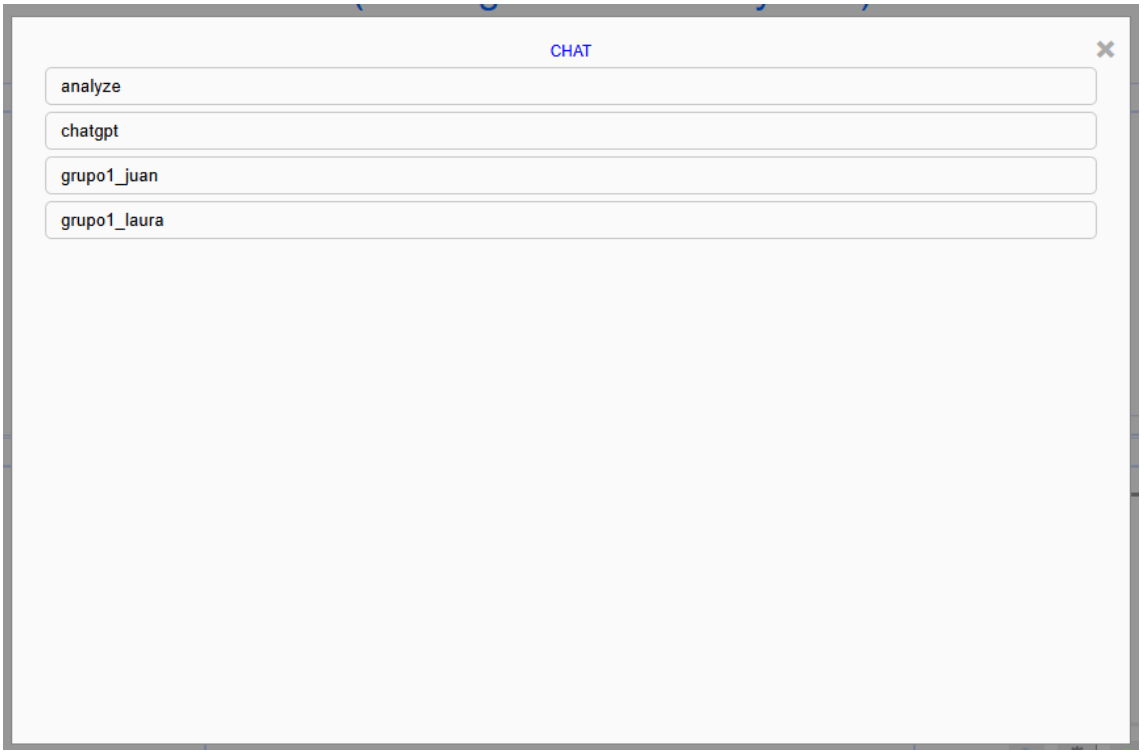

Figura 15 - Chat con lista de usuarios

Para la generación de este *pop-up* se han creado los predicados:

modal_chat(User, Realm) crea el pop-up y llama a la clase chat.js que contiene la funcionalidad del chat. Además, llama al predicado chat_skeleton(User, Realm) encargado de crear el esqueleto del chat.

Este último predicado hace llamada tanto al predicado que lee el archivo con la lista de usuarios (read_passwd_file) como a generate_user_list el cual a su vez llama a generate_user_list_2 que crean la lista de los usuarios con los que se puede comenzar a hablar por el chat, y pone como primeras opciones los dos chats con ChatGPT.

4.1. Chat entre usuarios del mismo grupo

Cuando en la figura 15 se pulsa en alguno de los usuarios del grupo, inmediatamente se cambia la pantalla de la ventana emergente a la que observamos en la figura 16:

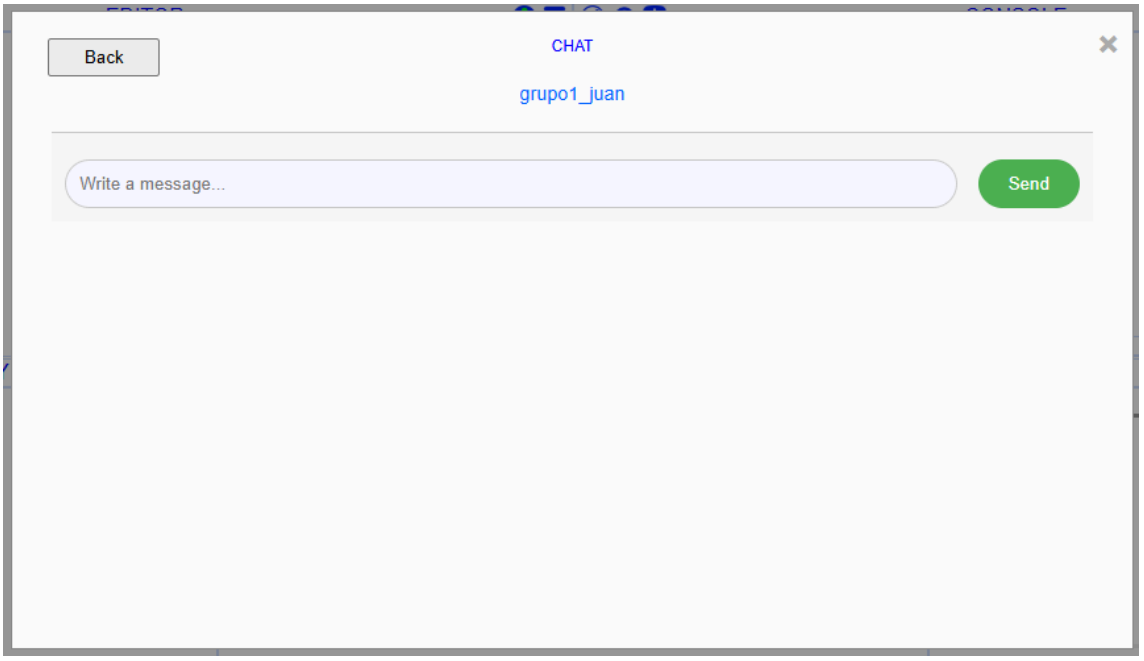

Figura 16 - Chat vacío con un usuario del grupo

Una vez en esta pantalla, podrás escribir lo que quieras al usuario y le llegará en tiempo real (figura 17).

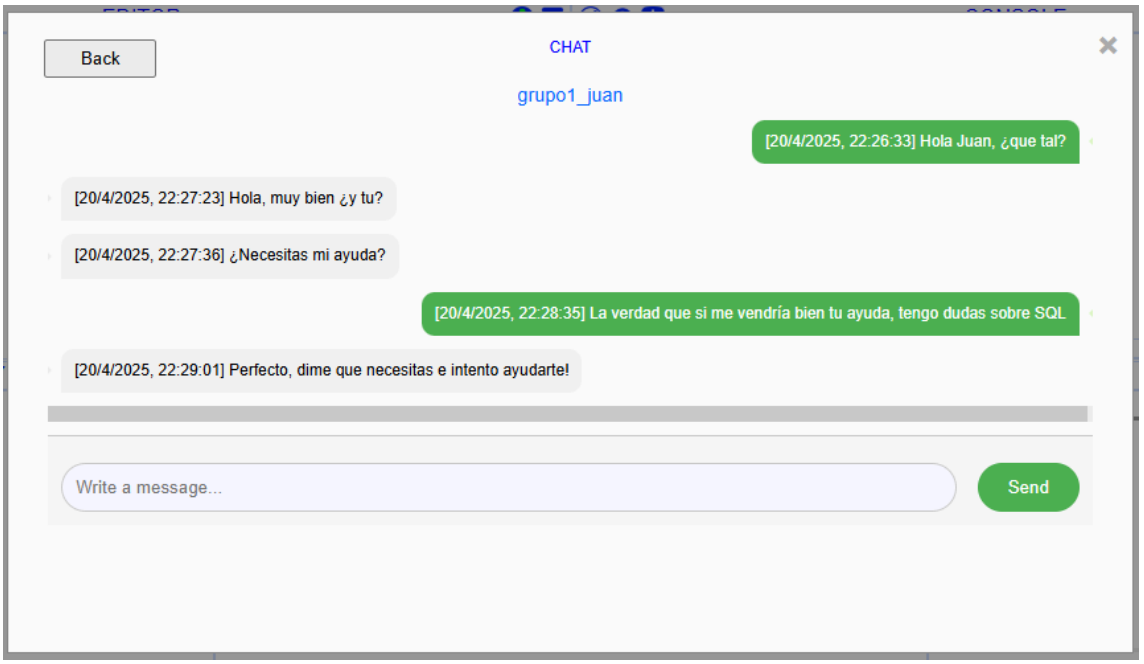

Figura 17 - Imagen del usuario Marta del Grupo 1 hablando con Juan del Grupo 1

Figura 18 - Imagen del usuario Juan del Grupo 1 hablando con Marta del Grupo 1

Como se puede ver en las figuras 17 y 18, los mensajes se actualizan en tiempo real y se muestran a ambos usuarios.

La plataforma muestra los mensajes del usuario principal, a la derecha los enviados en color verde y a la izquierda los recibidos en color gris.

4.2. Chat con ChatGPT

Existen dos modalidades de interacción con ChatGPT integradas en la aplicación. La primera consiste en una conversación convencional, similar a la que se puede mantener en la plataforma oficial de ChatGPT. Esta funcionalidad permite al usuario comunicarse libremente con el modelo sin necesidad de abrir una nueva ventana o abandonar la aplicación actual, se facilita así el acceso inmediato a asistencia, resolución de dudas o generación de contenido directamente desde el entorno de trabajo.

Esta modalidad está representada en el sistema mediante el usuario identificado como "chatgpt" como se puede ver en la siguiente imagen (figura 19):

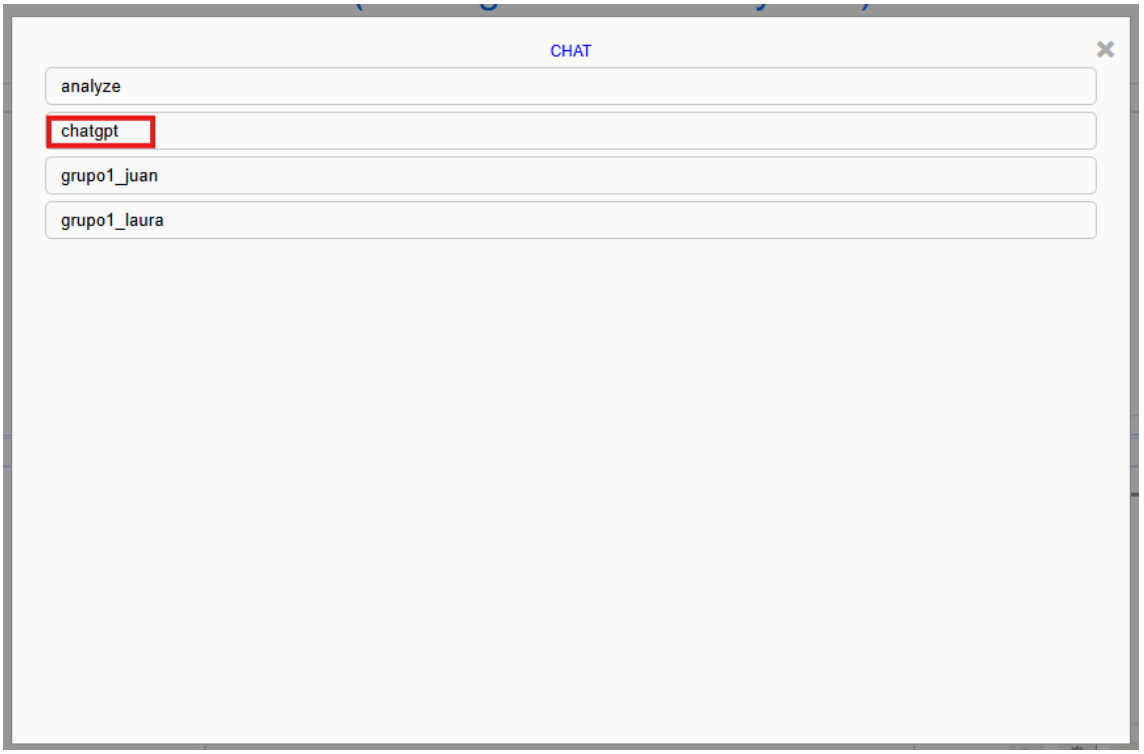

Figura 19 - Lista de usuarios indicando ChatGPT

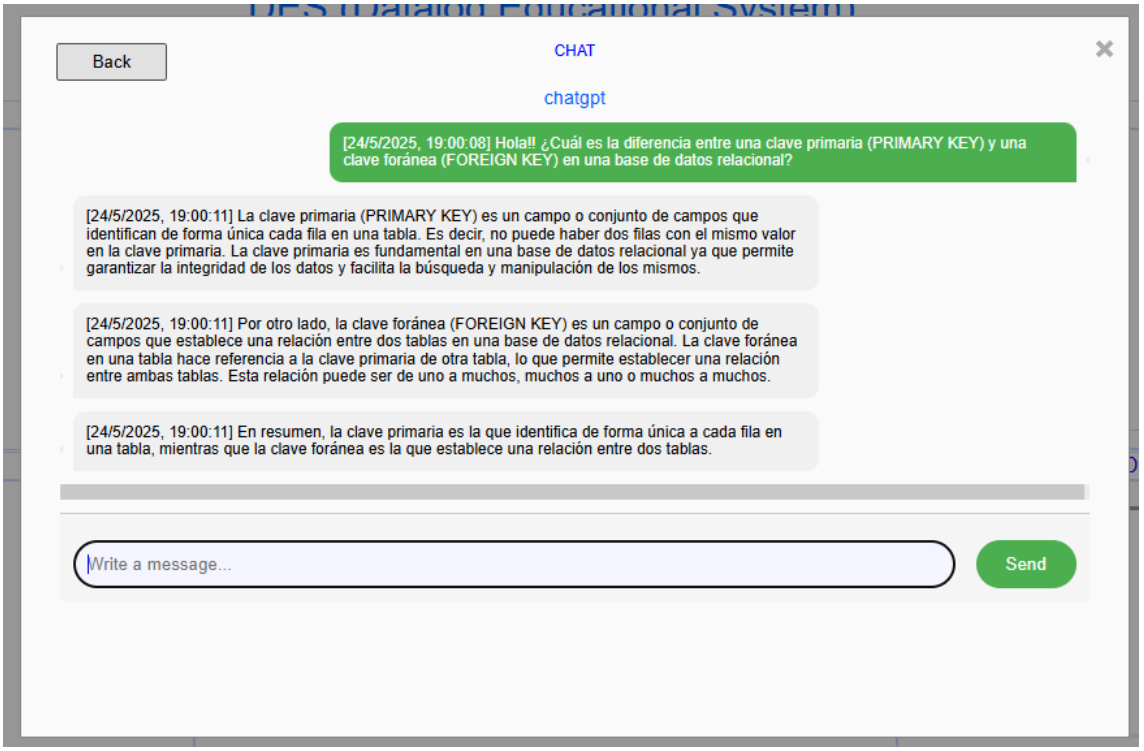

Figura 20 - Conversación con ChatGPT

Como se observa en la figura 20 el comportamiento es igual al de la plataforma oficial, pero a través de la interfaz DESweb.

Por otro lado, se ha incorporado un segundo tipo de interacción representado por el usuario "análisis" (figura 21). En esta modalidad, ChatGPT recibe automáticamente la última consulta realizada por el usuario, junto con la estructura de la base de datos utilizada. Esta información le permite ofrecer una asistencia más precisa y contextualizada, facilitando la identificación de posibles errores o inconsistencias en la consulta y proponiendo soluciones adecuadas (figura 22).

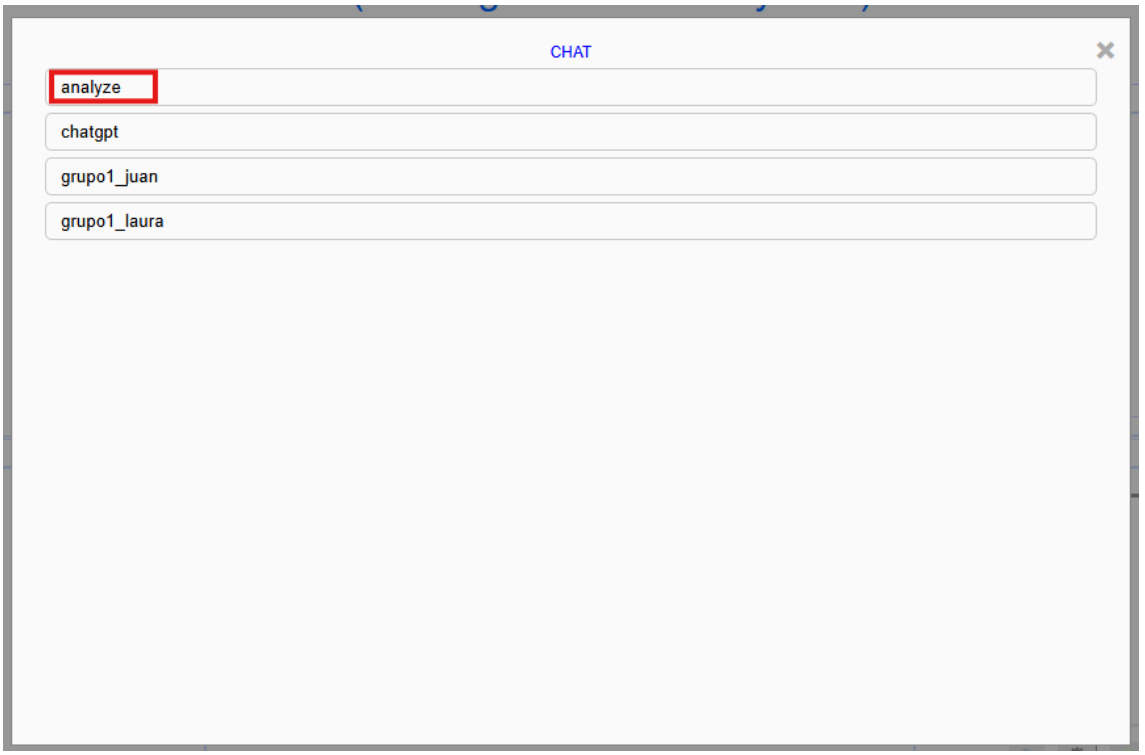

Figura 21 - Lista de usuarios indicando análisis

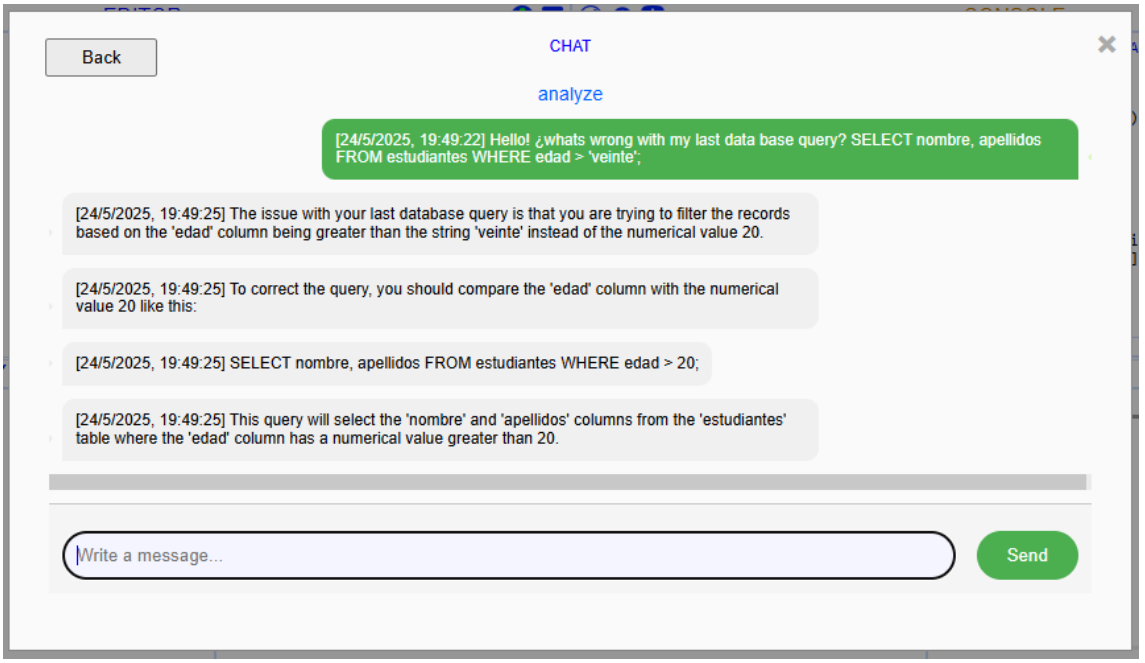

Figura 22 - Conversación con análisis

Figura 23 - Conversación con análisis sin consulta

En la figura 23 se puede ver qué ocurre si al pulsar en el usuario representado en la figura 21 no hay ninguna consulta realizada. Si este caso se da, entonces el mensaje a enviar es: “No hay historial de consultas”, pero se podrá seguir conversando con el asistente independientemente de que no haya historial.

Si no se puede conectar con ChatGPT aparecerá el siguiente mensaje que se muestra en la figura 24 para que el usuario sepa que está ocurriendo un error en la conexión:

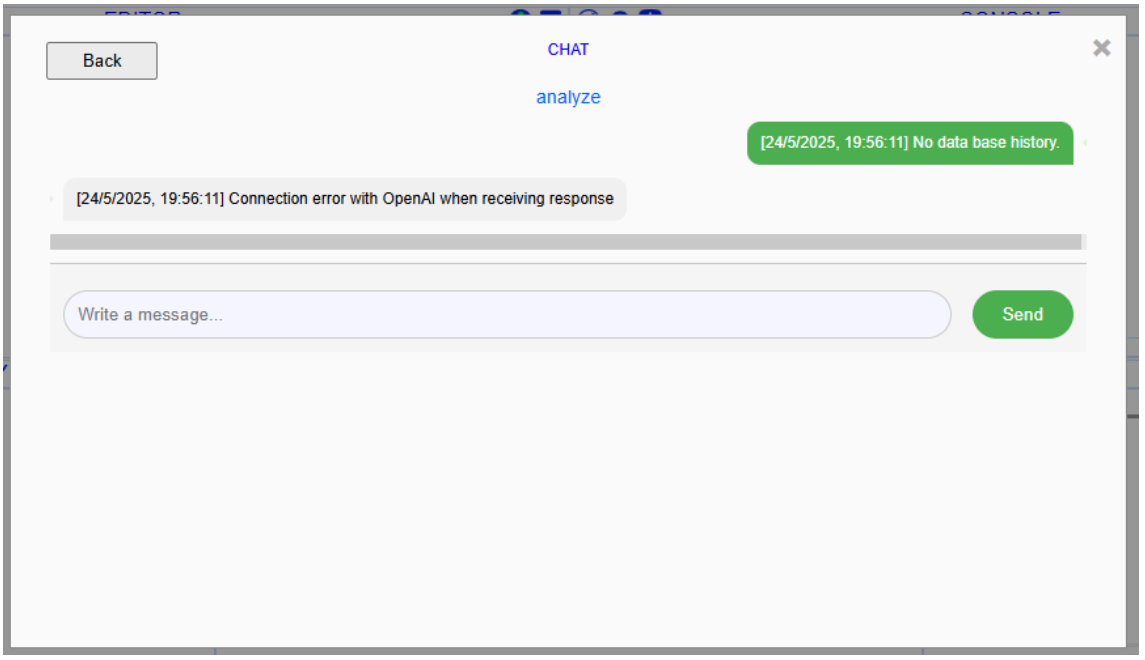

Figura 24 - Conversación con análisis con error al obtener consulta

4.3. Chat directo con “análisis”

Como se puede observar en la figura 13 se ha creado un nuevo botón de acceso directo a ChatGPT. Se puede ver una imagen más detallada del botón en la figura 25.

Figura 25 - Icono de acceso directo a análisis

Este botón es un acceso directo a la conversación con “análisis” sin pasar por la lista de usuarios, de esta forma se podrá acceder a esta conversación y enviarle la última consulta a ChatGPT de una forma más rápida y cómoda para el usuario (figura 26):

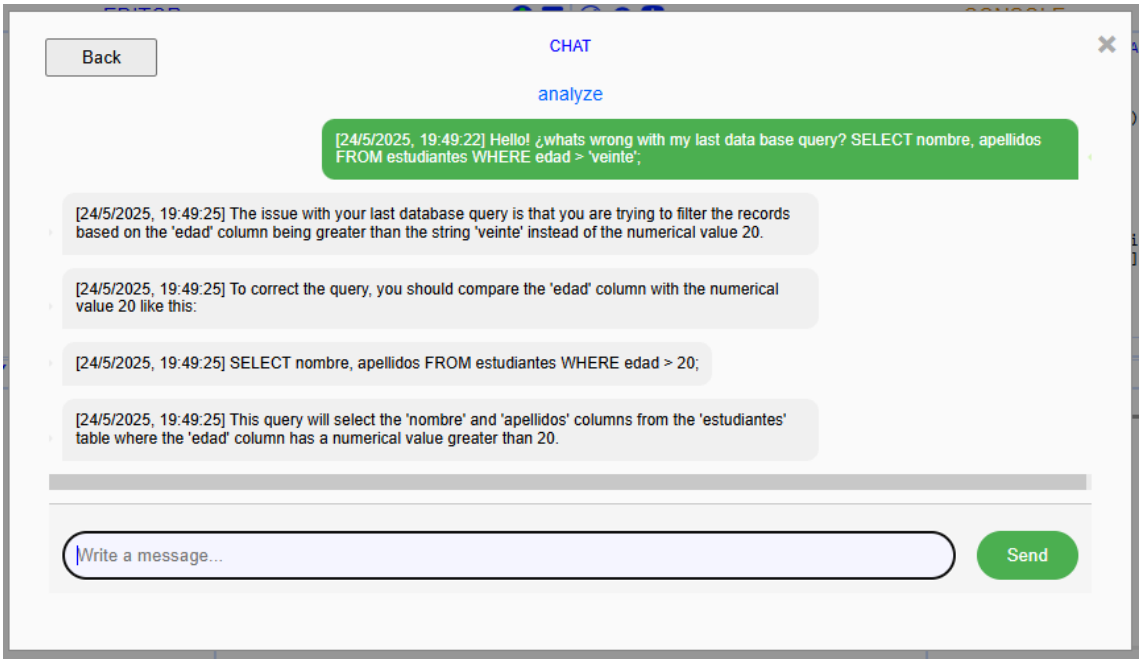

Figura 26 - Conversación acceso directo a análisis

Para el funcionamiento de este nuevo acceso directo (figura 26) al hacer clic en el icono del botón se hace una llamada a la siguiente función de chat.js, launchHelpChatDirectly(), diseñada para abrir el modal del chat y simular un clic en el

botón específico (análisis). También para el correcto funcionamiento se hace uso de la función: openHelpChat().

4.4. Movimiento de ventana de chat

Se ha añadido una nueva funcionalidad que consiste en la posibilidad de mover la ventana emergente del chat para poder ver simultáneamente otras partes de la interfaz como las consultas realizadas u otras ventanas que si no quedarían tapadas por el pop up.

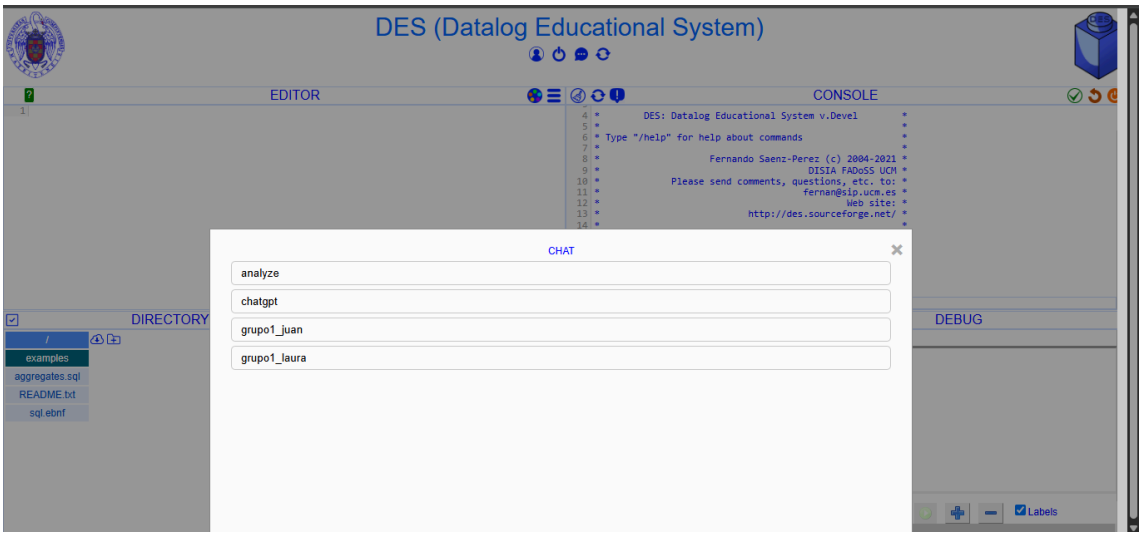

FIGURA 27 - VENTANA MOVIBLE

Como se puede observar en la figura 27, se puede mover por toda la pantalla y así poder ver lo que se encuentra debajo.

Capítulo 5 – Interacción Cliente – Servidor

En el desarrollo actual de aplicaciones web es común seguir una arquitectura basada en la relación entre el cliente y el servidor. En este modelo, el servidor se encarga de procesar la lógica de negocio, gestionar los datos y comunicarse con la base de datos, mientras que el cliente se ocupa de mostrar la interfaz y manejar la interacción del usuario. En este contexto, una configuración, consiste en utilizar Prolog como servidor y JavaScript como cliente.

5.1. Prolog como servidor web

Prolog, es un lenguaje de programación lógico que ha sido históricamente asociado con la inteligencia artificial. Sin embargo, en los últimos años, implementaciones como SWI-Prolog han extendido su funcionalidad para permitir el desarrollo de aplicaciones web completas. Esto ha sido posible gracias a bibliotecas como http, http_server, http_dispatch, http_parameters y http/json, que permiten al servidor Prolog recibir, interpretar y responder a solicitudes HTTP. [1][6][7][8]

Al utilizar Prolog como servidor, se aprovechan sus capacidades únicas para manejar bases de conocimiento lógicas. La lógica del servidor se define en forma de hechos y reglas, como hemos visto anteriormente, lo que permite una representación concisa y semánticamente rica del conocimiento. Esta estructura es especialmente útil en aplicaciones donde el razonamiento, la verificación de restricciones o la toma de decisiones automáticas forman parte central del funcionamiento.

La interacción con el cliente se realiza principalmente a través de puntos de entrada HTTP (endpoints), definidos como predicados que procesan las solicitudes entrantes, extraen parámetros, consultan o modifican el estado interno del servidor, y devuelve respuestas en formatos como JSON o HTML. Gracias a la naturaleza lógica de Prolog, las respuestas se generan de forma dinámica[9][10].

Una característica destacada de SWI-Prolog como servidor es su capacidad para mantener el estado entre sesiones, almacenar hechos dinámicamente en memoria y actualizar el conocimiento a medida que evoluciona la interacción del usuario. Además, puede realizar tareas como autenticación de usuarios o gestión de lo que lo convierte en una plataforma versátil y autosuficiente para aplicaciones web.

5.2. JavaScript como cliente

En el lado del cliente, JavaScript desempeña el papel principal en la construcción de la interfaz dinámica e interactiva. Su ejecución en el navegador permite la realización de eventos, uso del DOM y actualizar el contenido sin recargar la página. Gracias a estas capacidades, JavaScript puede comunicarse constantemente

con el servidor Prolog para solicitar datos, enviar formularios, recibir respuestas en tiempo real y actualizar la interfaz.

Cuando se utiliza JavaScript como cliente con un servidor en Prolog, se establece una arquitectura basada en peticiones HTTP (normalmente del tipo GET o POST), donde el cliente envía datos al servidor o solicita información, y este responde con datos en formato JSON. Esta comunicación asincrónica permite construir aplicaciones funcionales sin comprometer la experiencia de usuario [8].

JavaScript también puede integrarse con bibliotecas como React, Vue.js o jQuery para facilitar la gestión de la interfaz y el uso de su contenido[4][5]. En el caso de una integración con Prolog, el cliente puede estar configurado para enviar automáticamente las entradas del usuario, procesar las respuestas y representar los resultados, ya sea en forma de mensajes, tablas, formularios u otros elementos[9].

5.3. Comunicación cliente-servidor

La interacción entre Prolog y JavaScript se realiza generalmente mediante el protocolo HTTP, en el que JavaScript actúa como cliente que envía peticiones al servidor Prolog, este al mismo tiempo procesa esas peticiones y genera una respuestas. Estas respuestas suelen representarse como objetos JSON[8].

Un flujo típico puede resumirse de la siguiente manera[9]:

1. El usuario realiza una acción en la interfaz (por ejemplo, enviar un mensaje o consultar datos).

2. JavaScript captura esa acción y envía una petición HTTP al servidor Prolog, incluyendo los parámetros necesarios.

3. Prolog recibe la solicitud, ejecuta las reglas o consultas lógicas correspondientes, y genera una respuesta.

4. La respuesta se envía al cliente en formato JSON.

5. JavaScript interpreta la respuesta y actualiza la interfaz de forma dinámica.

Este modelo favorece una separación clara entre la lógica y la presentación, y permite el aprovechamiento de las fortalezas de cada lenguaje: la capacidad declarativa y de razonamiento de Prolog, y el dinamismo de JavaScript en el navegador.

El uso de Prolog como servidor y JavaScript como cliente representa una gran combinación para el desarrollo de aplicaciones web. Esta arquitectura no es la más común en el panorama de desarrollo web actual

La posibilidad de conectar estos dos mundos permite crear soluciones innovadoras, eficientes y altamente adaptables a las necesidades del usuario. Gracias a herramientas como SWI-Prolog, esta integración se vuelve cada vez más sencilla.

5.4. Relación entre chat.js y user_interface.pl

La relación creada entre chat.js y user_interface.pl es una relación cliente-servidor para poder hacer la interacción entre usuarios y con ChatGPT ya que user_interface.pl contiene la lógica del servidor (backend) y chat.js contiene la lógica que configura el cliente (frontend).

Este sistema permite el envío y recepción de mensajes desde la interfaz web. La información se almacena en el servidor y de esta forma se puede recuperar y no desaparece. Este manejo de mensajes se realiza a través de llamadas HTTP tipo *GET* entre Prolog y JavaScript.

Esta llamada GET hace la solicitud de información desde el cliente al servidor, no realiza ninguna modificación en dicha información, solo la extrae.

El flujo entre el frontend y el backend consiste en el envío de un mensaje, guardado de dicho mensaje y finalmente su recuperación para poder mostrar por pantalla.

En el cliente se realizará la siguiente acción, al pulsar en el botón de enviar mensaje dentro del chat:

```
document.getElementById('send-button').addEventListener('click', () => sendMessage());
```

Esta acción llama a la función sendmessage() que construye la URL:

```
function sendMessage(messageText) {
        const message = messageText || input.value.trim();
        const dbStructure = hiddenDbStructureInput.value;
        if (message !== '' && currentChatUser) {
            fetch('/send_message?' + new URLSearchParams({
                user1: currentUser,
                user2: currentChatUser,
                message,
                dbStructure
            }), {
```

```
                method: 'GET'
            }).then(response => response.text())
              .then(() => {
                  input.value = '';
                  getMessages();
              })
      .catch(err => console.error("Error sending message:", err));
        }
    }
```

Esto lanza una petición GET a /send_message con los datos del mensaje. Posteriormente el servidor recibe esta llamada:

```
http_parameters(Request, [user1(User1, []), user2(User2, []),
message(Message, [])]),
    current_timestamp_ms(Timestamp),
    \+ message(User1, User2, Message, Timestamp),
    assertz(message(User1, User2, Message, Timestamp)),
    format('Content-type: text/plain; charset=UTF-8~n~nOK')
```

1. La primera línea extrae de la URL los parámetros user1, user2 y message.
2. Posteriormente se guarda la hora del mensaje en milisegundos con Timestamp (esto se utiliza para ordenar mensajes en la interfaz del usuario).

```
\+ message(User1, User2, Message, Timestamp),
```

3. Este predicado falla si ya existe exactamente el mismo mensaje con ese timestamp (lo que evita mensajes repetidos por error).

```
assertz(message(User1, User2, Message, Timestamp)),
```

4. Con assertz se crea un nuevo hecho dinámico de Prolog. Este hecho queda en memoria, y se puede consultar después con findall/3 (como se hace posteriormente en get_messages).

Por ejemplo:

```
message(grupo1_ana, grupo1_pablo,”Hola, ¿que tal?”, 1715451234567).
```

5. Una vez almacenados los mensajes, se pueden recuperar de la memoria gracias al predicado get_messages(), que posteriormente los transforma en JSON que lee el cliente para mostrar por pantalla.

6. Para comunicarse con ChatGPT, se realizará a través de la siguiente función:

```
auto_response(Input, Output) :-

    catch(gpt_completions('gpt-3.5-turbo', Input, Output, _, []),
_, Output = "Connection error with OpenAI when receiving
response").
```

La llamada a este predicado se realiza en send_message() cuando se da la casuística de que User2 sea o bien “chatgpt” o bien “análisis”.

Como se puede observar se hace una llamada al predicado gpt_completions() que se encuentra en prolog2gpt.pl y se manda la versión de GPT a utilizar, en este caso 'gpt-3.5-turbo'[17].

La siguiente función se encarga de llamar al servidor para que éste le envíe todos los mensajes de un chat entre dos usuarios y mostrarlos en pantalla.

```
function getMessages() {

        if (!currentChatUser) return;

        fetch('/get_messages?' + new URLSearchParams({

            user1: currentUser,

            user2: currentChatUser

        }))

      .then(response => {

        if (!response.ok) {
```

```
            throw new Error("Error in servers response");
        }
    return response.json();  })
.then(data => {messagesContainer.innerHTML = '';

                if (!Array.isArray(data)) {
    console.error("The server did not return a valid array:", data);
     return;
}

const filteredMessages = data.filter(({ content }) => content.trim() !== '');
const parsedMessages = filteredMessages.map(({ sender, content, timestamp }) => ({
            sender,
            content: content.trim(),
            timestamp: new Date(Number(timestamp)) || new Date()
  }));

    parsedMessages.sort((a, b) => a.timestamp - b.timestamp);

    parsedMessages.forEach(({ sender, content, timestamp }) => {
            const li = document.createElement('li');
            const formattedTime = timestamp.toLocaleString();
            li.textContent = `[${formattedTime}] ${content}`;

                    if (sender === currentUser) {
                        li.classList.add('sent');
```

```
                    } else {
                        li.classList.add('received');
                    }

                    messagesContainer.appendChild(li);
                });

    messagesContainer.scrollTop = messagesContainer.scrollHeight;
            })
    .catch(err => console.error("Error obtaining response:", err));
     }
```

En primera instancia, si no hay un chat abierto (currentChatUser), no realiza ninguna acción. Si lo hay, entonces se recuperan los mensajes de los usuarios involucrados desde el servidor:

```
fetch('/get_messages?' + new URLSearchParams({
            user1: currentUser,
            user2: currentChatUser
        }))
```

El servidor recibe esa petición y la procesa de la siguiente manera:

```
get_messages(Request) :-
http_parameters(Request, [user1(User1, []), user2(User2, [])]),
    findall(
        _{sender: Sender, content: Msg, timestamp: Timestamp},
        (
(message(User1, User2, Msg, Timestamp), Msg \= "", Sender = User1);
(message(User2, User1, Msg, Timestamp), Msg \= "", Sender = User2)
```

```
        ),
        Messages
    ),
    reply_json(Messages).
```

1. En la primera línea *http_parameters* se extraen de la URL los dos usuarios User1 y User2.

```
http_parameters(Request, [user1(User1, []), user2(User2, [])]),
```

2. Posteriormente, se realiza una lista de objetos en formato JSON con sender, content y timestamp. Se buscan los mensajes entre los dos usuarios en ambos sentidos, siempre y cuando dicho mensaje no sea vacío.

```
findall(
_{sender: Sender, content: Msg, timestamp: Timestamp},
 (
(message(User1, User2, Msg, Timestamp), Msg \= "", Sender = User1);
(message(User2, User1, Msg, Timestamp), Msg \= "", Sender = User2)
  ),
    Messages
 ),
```

3. Finalmente, se le envía una lista en formato JSON al cliente con la información del mensaje:

```
 reply_json(Messages).
```

El cliente entonces podrá mostrar por pantalla los mensajes recibidos por el servidor de manera que el usuario pueda visualizarlos.

Capítulo 6 – Conclusiones y trabajo futuro

Este trabajo ha sido muy enriquecedor y ha permitido mejorar la ya existente aplicación universitaria DESweb añadiendo una nueva experiencia de uso tanto para estudiantes como para docentes.

Se han incorporado herramientas nuevas como la mensajería instantánea y el acceso directo a inteligencia artificial. Con esta implementación, se ha logrado una plataforma más colaborativa, autónoma y eficiente para la resolución de dudas relacionadas con bases de datos.

Este proyecto permite mejorar la aplicación académica incorporando un chat en tiempo real con otros usuarios además de con ChatGPT, creando una relación cliente-servidor a través del formato de mensajes JSON. De esta forma se incorpora IA a los entornos educativos para mejorar y nutrirse de ella (figura 14):

Esto supone un punto de partida para futuras mejoras que sigan enriqueciendo la docencia y el aprendizaje mediante herramientas digitales avanzadas que son el futuro de la educación.

En cuanto al cumplimiento de los objetivos establecidos al inicio del proyecto, se puede afirmar que la gran mayoría de ellos han sido alcanzados con éxito. Se ha logrado una integración funcional del chat con ChatGPT, una interfaz operativa y una base sólida para futuras expansiones. Sin embargo, uno de los objetivos más ambiciosos —la capacidad de seleccionar texto en cualquier parte de la plataforma y enviarlo automáticamente a ChatGPT para su análisis— no pudo completarse debido a limitaciones de tiempo. Pese a ello, se considera un punto clave a desarrollar en etapas posteriores.

Trabajo futuro

Aunque la aplicación es completamente funcional en su estado actual, se han identificado múltiples áreas susceptibles de mejora o expansión, muchas de ellas centradas en la experiencia de usuario y la ampliación del potencial de la herramienta:

- **Interacción contextual con la IA:** Como ya se ha mencionado, se pretende incorporar una funcionalidad que permita enviar automáticamente a ChatGPT cualquier texto seleccionado por el usuario en la plataforma, similar al envío

actual de la última consulta realizada a la base de datos.

- **Mejoras en la interfaz visual y de notificaciones:**
    - Implementación de notificaciones en tiempo real para informar al usuario de la llegada de nuevos mensajes.
    - Integración de sonidos personalizados para alertas y notificaciones de chat.
    - Posibilidad de incorporar el chat como un panel permanente en la interfaz principal, en lugar de utilizar únicamente una ventana emergente.
- **Chats grupales y multimedia:**
    - Habilitación de chats de grupo, lo que permitiría conversaciones entre múltiples usuarios, ampliando las posibilidades de trabajo colaborativo.
    - Incorporación del envío de mensajes de voz, tanto entre usuarios como hacia ChatGPT, lo que facilitaría la comunicación en contextos donde la escritura no es viable.

Estas mejoras no sólo enriquecerán la funcionalidad de la aplicación, sino que también contribuirán a hacerla más eficiente y cercana a las herramientas digitales utilizadas actualmente en otros contextos educativos y profesionales.

En definitiva, este proyecto sienta las bases para una evolución de la plataforma DESweb, en la que la inteligencia artificial y la interacción en tiempo real se conviertan en pilares fundamentales de la docencia.

Capítulo 7 – Conclusions and future work

This project has been highly enriching and has allowed for the improvement of the existing university application DESweb by adding a new user experience for both students and teachers. New tools such as instant messaging and direct access to artificial intelligence have been incorporated. Thanks to this implementation, a more collaborative, autonomous, and efficient platform for solving database-related queries has been achieved.

This project is not only technically feasible—extending the academic application to incorporate both real-time chat and ChatGPT through a client-server relationship using JSON-formatted messages—but also adds significant value by integrating AI into educational environments. This lays the foundation for future improvements that continue to enhance teaching and learning through advanced digital tools, aligning with the future of education.

In terms of achieving the goals set at the beginning of the project, it can be stated that the vast majority have been successfully accomplished. A functional integration of the chat system with ChatGPT has been achieved, including an operational interface and a solid base for future expansions. However, one of the more ambitious objectives—enabling users to select text from anywhere on the platform and automatically send it to ChatGPT for analysis—could not be completed due to time constraints. Nevertheless, this feature is considered a key point to develop in future stages.

Future work

Although the application is fully functional in its current state, several areas for improvement or expansion have been identified, many of which focus on enhancing user experience and extending the tool's potential:

- **Contextual interaction with AI:** As previously mentioned, the goal is to incorporate a feature that allows any text selected by the user on the platform to be automatically sent to ChatGPT, similar to how the latest database query is currently processed.

- **Visual interface and notification improvements:**

    - Implementation of real-time notifications to inform the user of incoming messages.

    - Integration of custom alert sounds and chat notification tones.

    - Option to embed the chat as a permanent panel within the main interface, instead of relying solely on a pop-up window.

- **Group and multimedia chat:**
    - Enabling group chats, allowing conversations among multiple users and expanding the possibilities for collaborative work.
    - Incorporation of voice message functionality, both between users and towards ChatGPT, facilitating communication in contexts where typing is not practical.

These improvements will not only enhance the functionality of the application but will also make it more efficient and comparable to the digital tools currently used in other educational and professional settings.

In conclusion, this project lays the groundwork for the ongoing evolution of the DESweb platform, where artificial intelligence and real-time interaction become fundamental pillars of academic teaching.

Bibliografía

[1] SWI-Prolog. (n.d.). SWI-Prolog official website. Recuperado de https://www.swi-prolog.org/

[2] SWI-Prolog. (n.d.). Tutorial de Prolog (SWISH). Recuperado de https://swish.swi-prolog.org/p/Tutorial%20de%20prolog.swinb

[3] Flanagan, D. (2006). JavaScript for Web Developers. Universidad Complutense de Madrid. (Material interno).

[4] Packt Publishing. (2020). The JavaScript Workshop: Learn to Develop Interactive Web Applications with Clean and Maintainable JavaScript Code. Packt Publishing.

[5] W3Schools. (n.d.). JavaScript Tutorial. Recuperado de https://www.w3schools.com/js/default.asp

[6] Wielemaker, J., Huang, Z., & van der Meij, L. (2007). *SWI-Prolog and the Web*. arXiv preprint arXiv:0711.0917. Recuperado de https://arxiv.org/abs/0711.0917arXiv+1swi-prolog.org+1

[7] SWI-Prolog. (n.d.). *The HTTP server libraries*. Recuperado de https://www.swi-prolog.org/pldoc/man?section=httpserverswi-prolog.org+8swi-prolog.org+8swi-prolog.org+8

[8] SWI-Prolog. (n.d.). *http_parameters/2 – Get HTTP GET or POST form-data*. Recuperado de https://www.swi-prolog.org/pldoc/doc_for?object=http_parameters%3Ahttp_parameters%2F2swi-prolog.org+1swi-prolog.org+1

[9] Mozilla Developer Network. (n.d.). *Making network requests with JavaScript*. Recuperado de https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data

[10] Maity, S. (2021). *How to communicate effectively between a client and a server using JavaScript*. Dev.to. Recuperado de https://dev.to/codexam/-how-to-communicate-effectively-between-a-client-and-a-server-using-javascript-2bdo

[11] OpenAI. (2023). *API Reference*. OpenAI. https://platform.openai.com/docs/api-reference

[12] Lessonspace. (s.f.). *The Platform for Virtual Classrooms and Online Tutoring*. Recuperado de https://www.thelessonspace.com/LinkedIn+4Lessonspace+4helpdesk.thelessonspace.com+4

[13] iDoceo. (s.f.). *OpenAI / ChatGPT Integration*. Recuperado de https://idoceo.net/index.php/en/instructions/uncategorised/openai-chatgpt-integrationidoceo.net+5idoceo.net+5idoceo.net+5

[14] Additio App. (s.f.). *ChatGPT-4o in the Classroom: Get the Most Out of It*. Recuperado de https://additioapp.com/en/chatgpt-4o-en-el-aula-sacale-el-maximo-provecho/AdditioApp

[15] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language models are few-shot learners*. Advances in Neural Information Processing Systems, 33, 1877–1901. https://arxiv.org/abs/2005.14165

[16] ESDIMA. (s.f.). *¿Qué es HTML y CSS?* Recuperado de https://esdima.com/que-es-html-y-css/

[17] OpenAI. (s.f.). *Modelos: gpt-3.5-turbo*. Recuperado de https://platform.openai.com/docs/models/gpt-3.5-turbo

[18] Microsoft. (s.f.). *Visual Studio con GitHub Copilot: Programación asistida por IA*. Visual Studio. Recuperado el 26 de mayo de 2025, de https://visualstudio.microsoft.com/es/github-copilot/:contentReference[oaicite:3]{index=3}