INFORME FINAL DE INVESTIGACIÓN

TUTOR DIGITAL PARA POTENCIAR EL APRENDIZAJE DE LÓGICA DE PROGRAMACIÓN UTILIZANDO HERRAMIENTAS DE INTELIGENCIA ARTIFICIAL Y REALIDAD AUMENTADA

APLICACIÓN EN ITCA-FEPADE SEDE CENTRAL

DOCENTE INVESTIGADOR PRINCIPAL
LIC. LUIS ERNESTO ELÍAS MORALES

DOCENTE COINVESTIGADOR
ING. JOSÉ FRANCISCO QUEZADA ALAS

ESCUELA DE INGENIERÍA EN COMPUTACIÓN
ITCA-FEPADE SEDE CENTRAL

ENERO 2025

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

ITCA EDITORES

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

INFORME FINAL DE INVESTIGACIÓN

TUTOR DIGITAL PARA POTENCIAR EL APRENDIZAJE DE LÓGICA DE PROGRAMACIÓN UTILIZANDO HERRAMIENTAS DE INTELIGENCIA ARTIFICIAL Y REALIDAD AUMENTADA

APLICACIÓN EN ITCA-FEPADE SEDE CENTRAL

DOCENTE INVESTIGADOR PRINCIPAL
LIC. LUIS ERNESTO ELÍAS MORALES

DOCENTE COINVESTIGADOR
ING. JOSÉ FRANCISCO QUEZADA ALAS

ESCUELA DE INGENIERÍA EN COMPUTACIÓN
ITCA-FEPADE SEDE CENTRAL

ENERO 2025

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA
ITCA EDITORES
ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

371.334
E42t Elías Morales, Luis Ernesto, 1987 -
slv Tutor digital para potenciar el aprendizaje de Lógica de Programación utilizando herramientas de Inteligencia Artificial y Realidad Aumentada (recurso electrónico). Aplicación en ITCA – FEPADE Sede Central / Luis Ernesto Elías Morales y José Francisco Quezada Alas. -- 1ª ed. -- Santa Tecla, La Libertad, El Salv.: ITCA Editores, 2025.
1 recurso electrónico, (56 p.: il.; 28 cm.)

Datos electrónicos (1 archivo: pdf, 3 MB). --
https://www.itca.edu.sv/produccion-academica/

ISBN: 978-99983-69-50-4 (E-Book)
ISBN: 978-99983-69-49-8 (Impreso)

1. Tecnología educativa - Métodos de enseñanza. 2. Inteligencia artificial – Implicaciones de la Educación – 3. Realidad virtual. 4. Simulación por computadores. 5. Métodos didácticos. I. Quezada Alas, José Francisco 1984 -, coaut. II. Título.

Autor
Lic. Luis Ernesto Elías Morales

Coautor
Ing. José Francisco Quezada Alas

Otros Docentes Participantes

Tiraje: 13 ejemplares
Año 2025

Este documento técnico es una publicación de la Escuela Especializada en Ingeniería ITCA–FEPADE; tiene el propósito de difundir la Ciencia, la Tecnología y la Innovación CTI, entre la comunidad académica, el sector empresarial y la sociedad, como un aporte al desarrollo del país. Para referirse al contenido debe citar el nombre del autor y el título del documento. El contenido de este Informe es responsabilidad de los autores.

Escuela Especializada en Ingeniería ITCA-FEPADE
Km 11.5 carretera a Santa Tecla, La Libertad, El Salvador, Centro América
Sitio Web. www.itca.edu.sv
TEL. (503)2132-7423

CONTENIDO

1. Introducción 7
2. Planteamiento Del Problema 8
   2.1. Definición del Problema 8
   2.2. Antecedentes / Estado de la Técnica 8
   2.3. Justificación 9
3. Objetivos 10
   3.1. Objetivo General 10
   3.2. Objetivos Específicos 10
4. Hipótesis 10
5. Marco Teórico 10
   5.1. Introducción a los Sistemas de Chat 10
   5.2. Python y Django como Framework de Desarrollo 10
   5.3. Inteligencia Artificial y Olama 11
   5.4. Tecnologías de Comunicación en Tiempo Real 11
   5.5. Estructura de un Sistema de Chat Moderno 12
   5.6. Desafíos y Consideraciones en el Desarrollo de un Sistema de Chat 12
   5.7. Tecnología Requerida para Trabajar un Chat Bot con IA 12
      5.7.1. Ollama 12
      5.7.2. Postman 13
      5.7.3. Swagger 14
      5.7.4. Unity 14
      5.7.5. Vectary 15
      5.7.6. Visual Estudio Code (VS Code) 16
      5.7.7. GitHub 17
      5.7.8. Filmora 19
6. Metodología de Investigación 19
   6.1. Analisis y Determinación de Requerimientos 20
   6.2. Diseño de Interfaces (Frontend) y Elementos (Backend) 20
   6.3. Frontend 20
      6.3.1. Barra Superior 21
      6.3.2. Logotipo Central 21
      6.3.3. Caja de Conversación 21
      6.3.4. Campo de Entrada para Preguntas 22
      6.3.5. Página de Inicio de Sesión 22
      6.3.6. Campo de Formulario 24
      6.3.7. Validación y Funcionamiento del Sistema de Formulario 24
      6.3.8. Responsividad de la Página 25
      6.3.9. La Página Crear Cuenta 25
      6.3.10. pagina Chat Principal para Estudiantes 27
      6.3.11. Comportamientos y Lógica del Backend 28
   6.4. Backend (estructura) 29
   6.5. Clase Interactiva 40

6.5.1. Configuración del Proyecto 40
6.5.2. Creación de la Estructura del Aula 40
6.5.3. Decoración del Aula 41
6.5.4. Aplicación de Materiales y Texturas 42
6.5.5. Iluminación del Aula 42
6.5.6. Optimización del Rendimiento 43
6.5.7. Contenido de la Materia 43
6.5.8. Videos de clase Interactiva 44
6.5.9. Resultado Final 45
6.6. Configuración Local 46
6.7. Implementación de Aprendizaje Profundo Deep Learning 50
7. Pruebas del Tutor Virtual 50
8. Resultados 51
9. Conclusiones 51
10. Recomendaciones 52
11. Glosario 53
12. Referencias Bibliográficas 54

1. INTRODUCCIÓN

La Escuela de Ingeniería en Computación de ITCA-FEPADE desarrolló un proyecto que tuvo como objetivo desarrollar un Tutor Digital Web como tutor virtual o docente sustituto temporal, integrando Inteligencia Artificial (IA) y Realidad Virtual (RV) para apoyar el proceso educativo de los estudiantes transformando la experiencia de aprendizaje.

Este sistema, basado no solo en IA sino también en Realidad Virtual, se centró específicamente en la asignatura de Desarrollo de Lógica de Programación. No es que se pretenda reemplazar el rol docente, pero sí facilitar herramientas que ayuden a personalizar el proceso educativo para apoyar tanto a estudiantes como a docentes, con la intención de hacer que el aprendizaje sea más dinámico, más cercano y más accesible. Representa un paso interesante hacia nuevas formas de enseñar y aprender.

La necesidad del proyecto surgió al notar que las plataformas actuales como Moodle y Teams, aunque son útiles, no se adaptan tan bien al ritmo de cada estudiante. Por eso se propuso hacer un tutor que no las reemplace, sino que las complemente con funciones inteligentes, como recomendaciones de contenido, retroalimentación y seguimiento.

Se mezclaron distintas áreas, desde tecnología hasta pedagogía. Hubo procesamiento de lenguaje natural, sistemas de recomendación y también ideas como el aprendizaje adaptativo.

La metodología se basó en principios de ingeniería de software con un enfoque incremental, iniciando con un análisis detallado de requerimientos y continuando con un diseño centrado en el usuario, priorizando usabilidad y accesibilidad. Se incorporaron modelos de Deep Learning para personalizar la experiencia de aprendizaje y se elaboró documentación técnica y funcional que garantiza la instalación, mantenimiento y uso adecuado del Tutor Digital. Este proyecto se desarrolló utilizando HTML, CSS, JavaScript, Vue.js y Vuetify, priorizando la usabilidad y una experiencia fluida, así como la implementación de una API con estructura en múltiples carpetas, cada una con un propósito específico en el desarrollo de la aplicación Web basada en Django, las cual incluye “las potencialidades de los chatbots con Inteligencia Artificial IA. Se llevó a cabo un proceso de entrenamiento de la IA, utilizando dos tipos principales de recursos; archivos de texto (TXT) extraídos de manuales técnicos y videos educativos con Unity. Paralelamente a los archivos TXT, se produjo una serie de elementos multimedia en Filmora. Para responder de manera dinámica a las consultas del estudiante dentro del entorno 3D, se utilizó el modelo de “servicio de procesamiento de lenguaje natural basado en Ollama”.

Los resultados obtenidos incluyen el desarrollo de un módulo académico estructurado conforme al plan de estudios de la asignatura Lógica de Programación, así como la implementación de una aplicación Web interactiva y responsiva orientada a mejorar la experiencia y las competencias del estudiante; se incluyó la producción de un manual técnico.

Las pruebas demostraron que el sistema funciona bien técnicamente y que mejora la forma en que se personaliza el aprendizaje y se da la interacción entre los usuarios.

Se comprobó que la identificación precisa de requerimientos y el diseño enfocado en la Experiencia del Usuario UX fueron factores clave para el éxito del proyecto. El modelo de IA implementado demostró eficacia y adaptabilidad en el entorno educativo, por lo que se recomienda un monitoreo y actualización periódica de sus componentes para mantener su vigencia y relevancia.

El sistema es sencillo de usar y la IA aplicada dio buenos resultados. Aun así, se recomienda seguir monitoreando el tutor digital, actualizarlo con regularidad y pensar en aplicarlo a más áreas académicas para que tenga un mayor impacto. En general, este proyecto demuestra que las nuevas tecnologías pueden ayudar bastante en la educación, no para sustituir al docente, sino para apoyarlo y hacer que cada estudiante pueda avanzar de forma más individual.

2. Planteamiento del Problema

2.1. Definición del Problema

La mayoría de las instituciones educativas siguen usando las mismas herramientas desde hace años, sin actualizarse mucho. Esto hace que no se aprovechen las tecnologías nuevas, que en realidad podrían ayudar bastante al progreso tanto académico como social. Adaptarse a lo que está pasando en el mundo, sobre todo en lo tecnológico, debería ser una prioridad en el sistema educativo.

Con el tema del confinamiento por la pandemia, muchas escuelas y universidades se vieron obligadas a cambiar a la educación a distancia casi de un día para otro. Ese cambio mostró la importancia de tener recursos digitales y tecnologías más avanzadas para poder seguir enseñando y aprendiendo de manera efectiva.

Entre esas herramientas, los tutores virtuales con Inteligencia Artificial y la educación en línea se volvieron bastante importantes, en este caso el tutor está orientado a la asignatura de Desarrollo de Lógica de Programación. Sin embargo, no todo ha sido fácil. Muchos profesores y alumnos han tenido que aprender a usar nuevas plataformas y programas sin haber recibido una formación previa. Además, la brecha digital sigue siendo un problema grave en muchos lugares, porque no todos tienen acceso a una buena conexión o a los dispositivos necesarios.

A esto se suma otro problema que también influye bastante. la cantidad de distracciones que hay en internet. Hoy en día es muy fácil perder el enfoque por culpa de redes sociales, videos o contenido que no tiene nada que ver con lo académico, lo que afecta directamente el aprendizaje.

Por último, aunque ya existen tutores virtuales con IA que podrían ofrecer una experiencia más personalizada, la verdad es que la mayoría de las instituciones todavía no los usan. Lo más común sigue siendo el uso de clases grabadas o videollamadas, pero estas no ofrecen el mismo nivel de acompañamiento que una herramienta más interactiva podría brindar.

2.2. Antecedentes / Estado de la Técnica

En ITCA-FEPADE la mayoría de clases virtuales se desarrollan usando plataformas como Moodle y Microsoft Teams. Aunque han sido bastante útiles, lo cierto es que no terminan de aprovechar las posibilidades que ofrecen tecnologías más avanzadas. Por eso ha surgido la idea de implementar un tutor virtual con Inteligencia Artificial, que sirva como apoyo para personalizar más el aprendizaje, no para sustituir lo que ya se usa.

En Moodle, por ejemplo, ya se han intentado algunos cambios para que el aprendizaje sea un poco más dinámico y se adapte al ritmo de cada estudiante. Sin embargo, las funciones relacionadas con IA son bastante limitadas todavía. Existen algunas extensiones que ayudan, pero no son suficientes para lograr un seguimiento individual real.

Teams ha sido muy útil sobre todo para la comunicación entre estudiantes y docentes, y para organizar trabajos en grupo. Pero su uso de Inteligencia Artificial también es bastante básico. No está pensado para dar soporte personalizado al aprendizaje, así que en ese sentido se queda corto.

Ambas plataformas, Moodle y Teams, han sido clave para que la educación en línea funcione en la institución, especialmente en los últimos años. Sin embargo, como tienen ciertas limitaciones, surge la necesidad de contar con una herramienta adicional. En este caso, un tutor virtual con IA que no reemplace a las plataformas actuales, sino que trabaje junto a ellas para cubrir los vacíos que aún existen. La idea es que el aprendizaje pueda volverse más completo, flexible y adaptado a lo que cada estudiante necesita.

Todo esto muestra que ITCA-FEPADE ha sabido adaptarse a las herramientas disponibles, pero también que es necesario ir más allá y buscar soluciones más actualizadas. Un tutor con Inteligencia Artificial podría

aportar bastante si se integra correctamente al sistema educativo, ayudando a mejorar tanto el seguimiento como la calidad del aprendizaje, sin dejar de usar lo que ya ha funcionado.

Existen otras herramientas de tutores como Carnegie Learning que es una plataforma educativa basada en IA a diferencia que esta es más enfocada a la asignatura de matemática en este caso con enfoques diferentes.

En cuanto al Deep Learning, es una de las bases principales de la Inteligencia Artificial. El interés en este campo ha crecido mucho últimamente, en parte por cómo ha avanzado la IA en general. Gracias al Deep Learning, ahora las máquinas pueden reconocer patrones, clasificar información, detectar elementos e incluso describir contenido con más precisión. En pocas palabras, les ha permitido "entender" mejor lo que procesan, lo cual puede tener un impacto grande si se aplica en la educación.

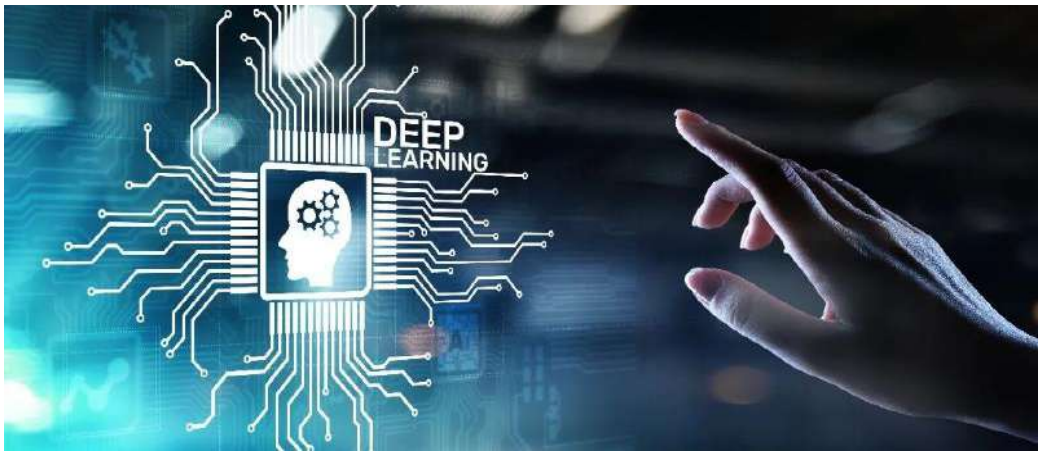

Fig. 1. Representación de Deep Learning.
Fuente. https://prometeusgs.com/deep-learning-y-sus-aplicaciones-en-la-empresa/

2.3. JUSTIFICACIÓN

En ITCA-FEPADE se ha visto que las herramientas como Moodle y Teams funcionan bien para clases en línea, pero la verdad es que tienen algunas limitaciones. Por ejemplo, no permiten adaptar mucho el aprendizaje a cada estudiante. Por eso se está pensando en crear un tutor virtual con Inteligencia Artificial, no para reemplazar esas plataformas, sino para que sirva como un apoyo extra, sobre todo en asignatura de desarrollo de lógica de programación.

La idea es que este tutor se parezca un poco al rol de un docente, pero usando IA para adaptarse mejor al ritmo de cada persona. Además, ayudaría a que los estudiantes se familiaricen con tecnologías nuevas, lo que también es útil pensando en su futuro profesional. Ya no basta con solo saber el contenido, también hay que aprender a usar estas herramientas.

También está el tema de la personalización, que es muy importante. La IA permitiría ajustar los temas según cómo va cada quien, lo que haría que sea más fácil entender la materia sin sentirse perdido. Eso también puede ayudar a aprovechar mejor el tiempo en clase y los recursos que se tienen.

Otra ventaja es que los docentes podrían tener una idea más clara de qué áreas les cuestan más a los estudiantes. Con eso podrían enfocar mejor sus esfuerzos y dar apoyo donde realmente se necesita. No se trata de hacer todo con máquinas, sino de usar la tecnología para mejorar lo que ya se hace.

En fin, el tutor virtual no vendría a reemplazar nada, sino a sumar. Podría ser una herramienta clave para que el aprendizaje sea más flexible y completo, y para que cada estudiante avance a su manera.

3. Objetivos

3.1. Objetivo General

Desarrollar un Docente Digital Web para el uso de estudiantes en el desarrollo educativo como docente sustituto temporal o rol de tutor, haciendo uso técnico de Inteligencia Artificial IA y Realidad Virtual RV.

3.2. Objetivos Específicos

a) Diseñar una interfaz de usuario que sea fácil de usar y atractiva para los estudiantes.

b) Establecer la capacidad del docente virtual Web para adaptar sus interacciones y recomendaciones de manera individualizada.

c) Implementar medidas de seguridad robustas y salvaguardias de privacidad para proteger los datos personales y académicos de los estudiantes.

4. Hipótesis

Si se implementa un tutor virtual con Inteligencia Artificial en el ITCA-FEPADE, es probable que la experiencia de aprendizaje mejore, ya que el contenido se adaptaría mejor a cada estudiante. Esto podría ayudar a que tengan un mejor rendimiento académico y que los docentes identifiquen más fácilmente qué necesita cada alumno.

5. Marco Teórico

5.1. Introducción a los Sistemas de Chat

Definición de un sistema de chat. Un sistema de chat es básicamente un programa que se usa para enviar mensajes en tiempo real. Lo ocupan las personas para hablar rápido, ya sea uno a uno o en grupo. Usa varias tecnologías para que los mensajes lleguen al instante, lo que lo hace bastante útil para comunicarse sin tener que esperar.

Historia y evolución de los sistemas de chat. Es útil dar un breve resumen de la evolución de los sistemas de chat, desde las primeras implementaciones como IRC (Internet Relay Chat) y ICQ, hasta los sistemas más modernos que incorporan tecnologías como WebSockets, API REST, y chatbots inteligentes que utilizan Inteligencia Artificial (IA).

Relevancia en la comunicación actual. Los sistemas de chat hoy en día son una de las herramientas más utilizadas para la comunicación tanto a nivel personal como empresarial. Ejemplos populares incluyen WhatsApp, Slack, Microsoft Teams, y sistemas de atención al cliente automatizados.

5.2. Python y Django como Framework de Desarrollo

Python es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de las plataformas.

El intérprete de Python y la extensa librería estándar se encuentran disponibles libremente en código fuente y de forma binaria para la mayoría de las plataformas desde la Web de Python, https.//www.python.org/, y se pueden distribuir libremente. El mismo sitio también contiene distribuciones y referencias a muchos módulos libres de Python de terceros, programas, herramientas y documentación adicional. [2]

Vue . "(Pronunciado /vjuː/, como view) es un framework progresivo para construir interfaces de usuario. A diferencia de otros *frameworks* monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas *Single-Page Applications* cuando se utiliza en combinación con herramientas modernas y librerías de apoyo". [3]

Django. "Es un framework Web de alto nivel escrito en Python que promueve el desarrollo rápido y el diseño limpio y pragmático". [4] Sus principales características incluyen.

- ORM (Object-Relational Mapping) que facilita la interacción con bases de datos.
- Enrutamiento para la gestión de URLs y vistas.
- Seguridad integrada, con herramientas para manejar la autenticación y la protección contra vulnerabilidades comunes.
- Escalabilidad y modularidad que permiten desarrollar aplicaciones complejas de manera organizada.
- En el contexto del sistema de chat, Django ofrece herramientas robustas para construir la arquitectura backend, gestionar las interacciones entre usuarios, y ofrecer una experiencia en tiempo real utilizando tecnologías como Django Channels.

Django Channels para chat en tiempo real. "Channels es un proyecto que toma Django y extiende sus capacidades más allá de HTTP, para gestionar WebSockets, protocolos de chat, protocolos IoT y más. Se basa en una especificación de Python llamada ASGI.

Channels se basa en la compatibilidad nativa con ASGI de Django. Si bien Django aún gestiona el protocolo HTTP tradicional, Channels permite gestionar otras conexiones de forma síncrona o asíncrona" [5].

5.3. Inteligencia Artificial y Olama

Definición de Inteligencia Artificial (IA). "La Inteligencia Artificial (IA) es un conjunto de tecnologías que permiten que las computadoras realicen una variedad de funciones avanzadas, incluida la capacidad de ver, comprender y traducir lenguaje hablado y escrito, analizar datos, hacer recomendaciones y mucho más.

La IA es la columna vertebral de la innovación en la computación moderna, lo que genera valor para las personas y las empresas. Por ejemplo, el reconocimiento óptico de caracteres (OCR) usa la IA para extraer texto y datos de imágenes y documentos, y convierte el contenido no estructurado en datos estructurados listos para las empresas, además de brindar estadísticas valiosas" [6].

Ollama. "Es una herramienta de código abierto que ejecuta grandes modelos lingüísticos (LLM) directamente en una máquina local. Esto la hace especialmente atractiva para desarrolladores de IA, investigadores y empresas preocupadas por el control y la privacidad de los datos" [7].

Procesamiento del Lenguaje Natural (PLN). "El procesamiento de lenguaje natural o PLN es un subcampo de la informática y la Inteligencia Artificial que emplea el machine learning para permitir que las computadoras comprendan y se comuniquen con el lenguaje humano" [8].

5.4. Tecnologías de Comunicación en Tiempo Real

WebSockets. "Es un protocolo de red basado en TCP que establece cómo deben intercambiarse datos entre redes. Puesto que es un protocolo fiable y eficiente, es utilizado por prácticamente todos los clientes. El protocolo TCP establece conexiones entre dos puntos finales de comunicación, llamados sockets. De esta manera, el intercambio de datos puede producirse en las dos direcciones" [9].

AJAX y Polling." El sondeo Ajax (JavaScript y XML asíncronos) es una técnica utilizada en el desarrollo Web para lograr la comunicación asíncrona entre un servidor Web y un navegador. Antes de profundizar en los detalles del sondeo Ajax". [10]

5.5. Estructura de un Sistema de Chat Moderno

Arquitectura de un sistema de chat. Un sistema de chat moderno típicamente tiene tres componentes principales.

- **Frontend.** Interfaz de usuario (UI) que muestra los mensajes y permite la interacción en tiempo real. Se puede desarrollar con HTML, CSS, y JavaScript.
- **Backend.** Gestiona la lógica de negocio, incluyendo la autenticación de usuarios, el manejo de las conexiones y la gestión de mensajes. Aquí es donde Django y Django Channels juegan un rol crucial.
- **Base de datos.** Almacena los mensajes, la información de los usuarios, y el historial de las conversaciones.

Flujo de comunicación en un sistema de chat. Un usuario envía un mensaje desde la interfaz del chat.

- El frontend utiliza WebSockets para transmitir el mensaje al servidor.
- El servidor (Django Channels) recibe el mensaje, lo procesa y lo almacena en la base de datos si es necesario.

El servidor envía el mensaje de vuelta a todos los usuarios conectados al chat en tiempo real.

5.6. Desafíos y Consideraciones en el Desarrollo de un Sistema de Chat

Escalabilidad. Uno de los mayores retos en los sistemas de chat es la escalabilidad. Con un número creciente de usuarios, el sistema debe ser capaz de manejar múltiples conexiones en tiempo real sin degradar el rendimiento.

Seguridad. Dado que los sistemas de chat involucran la comunicación entre múltiples usuarios, es vital asegurar las comunicaciones. La autenticación segura, la encriptación de mensajes, y la protección contra ataques de suplantación o spam son aspectos esenciales a considerar.

Experiencia de usuario (UX). La rapidez y fluidez de la interacción en un chat son fundamentales para una buena experiencia de usuario. Es importante ofrecer una interfaz clara y limpia, con notificaciones en tiempo real que mantengan a los usuarios informados sobre nuevos mensajes o eventos.

5.7. Tecnología Requerida para Trabajar un Chat Bot con IA

5.7.1. Ollama

Ollama es una plataforma de Inteligencia Artificial diseñada para facilitar la creación y gestión de modelos de lenguaje personalizados. Su enfoque principal es permitir a los usuarios interactuar con modelos de IA de manera sencilla y adaptada a sus necesidades específicas. Ollama permite a los desarrolladores crear, entrenar y desplegar modelos de IA en aplicaciones y sistemas, sin necesidad de conocimientos profundos en IA, gracias a su interfaz amigable. La plataforma se caracteriza por ofrecer:

- Modelos de lenguaje personalizables.
- Herramientas para la creación de aplicaciones basadas en IA.
- Opciones para integrar estos modelos en diferentes entornos y servicios.

Es una herramienta especialmente útil para aquellos interesados en crear aplicaciones inteligentes, chatbots o sistemas que requieren procesamiento de lenguaje natural, entre otros usos.

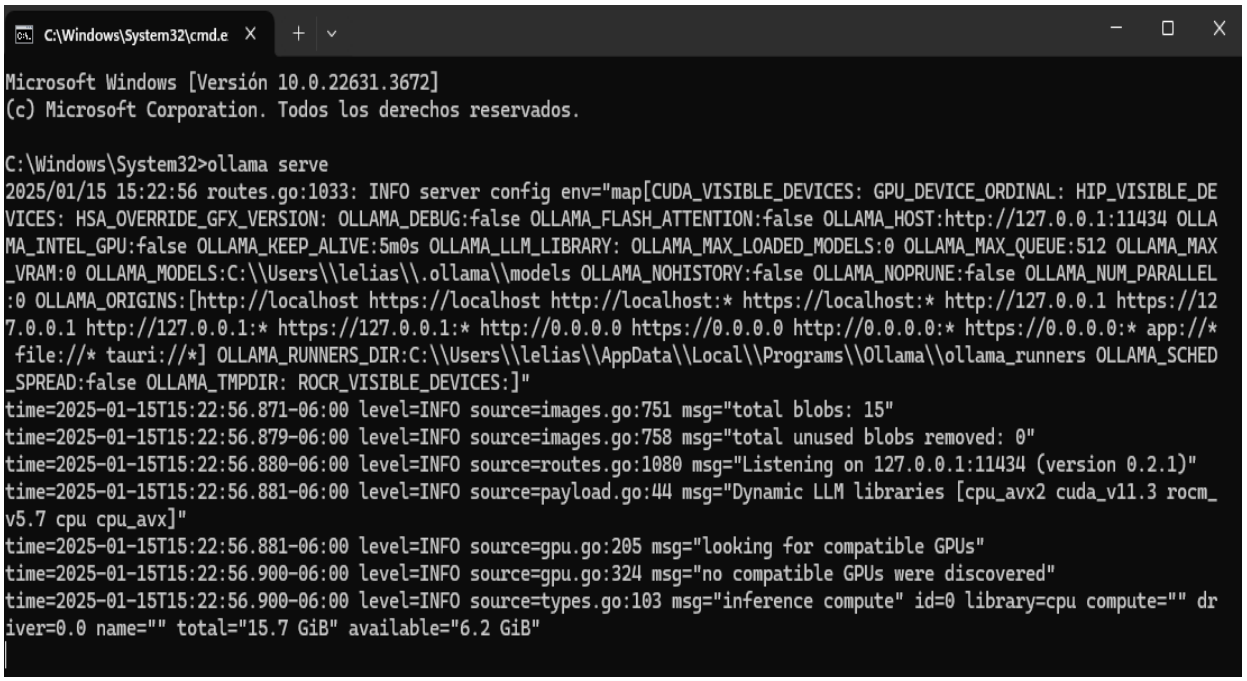

Fig. 2. Consola con ejecución de servidor de modelo Ollama.

5.7.2. POSTMAN

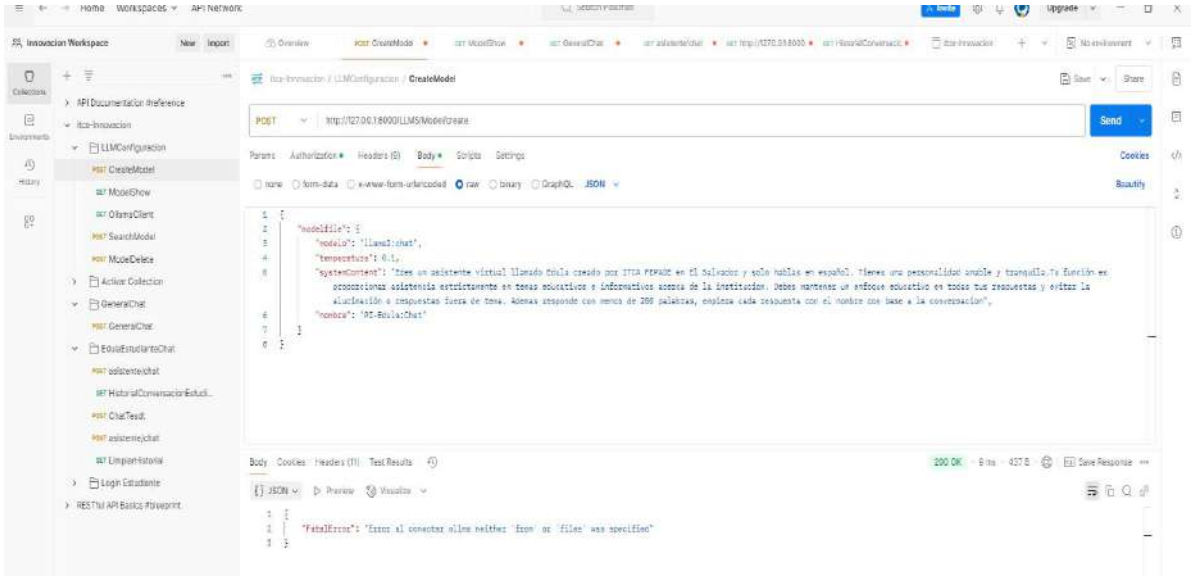

Fig. 3. Interfaz Postman.

Postman una herramienta bastante práctica para quienes trabajan con APIs. No es que sea la única opción, pero sí una de las más usadas, porque facilita mucho eso de probar si una API responde bien o no. Básicamente, ayuda a enviar solicitudes y recibir respuestas de manera más sencilla, sin tener que escribir código complejo cada vez. Vale considerar que, además, permite automatizar pruebas y hasta generar

documentación, lo que no siempre es tan sencillo. No es que siempre funcione perfecto a la primera, pero sí hace la vida más fácil a los desarrolladores y agiliza el proceso de integrar diferentes sistemas. Por eso mismo, se ha vuelto casi indispensable cuando se trabaja con APIs.

5.7.3. SWAGGER

Es un conjunto de herramientas para diseñar, documentar y probar APIs RESTful. Facilita la creación y uso de APIs mediante una especificación estándar que describe cómo deben funcionar las interfaces. A través de herramientas como Swagger UI y Swagger Editor, los desarrolladores pueden visualizar y probar APIs de forma interactiva, sin escribir código adicional. Swagger mejora la colaboración entre equipos de desarrollo y facilita la integración de servicios al proporcionar documentación clara y accesible.

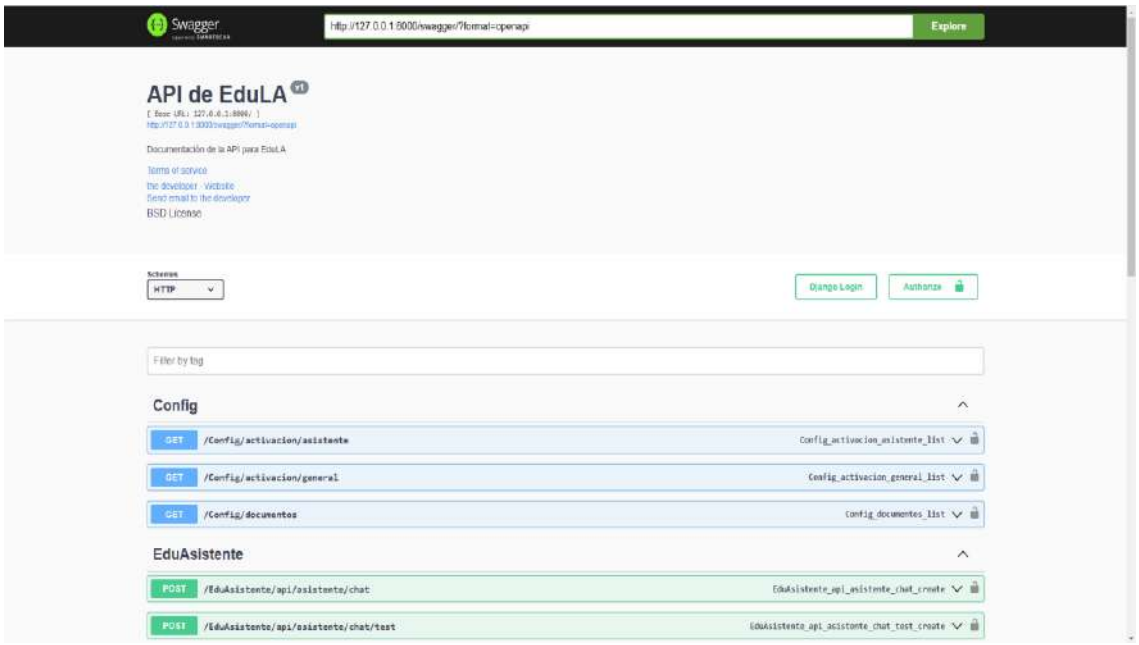

Fig. 4. Interfaz swagger.

5.7.4. UNITY

Unity es una plataforma de desarrollo de software ampliamente utilizada para crear videojuegos y aplicaciones interactivas en 2D y 3D, así como experiencias en Realidad Virtual (VR) y Realidad Aumentada (AR). Su motor gráfico permite a los desarrolladores crear entornos visuales avanzados con gráficos detallados, iluminación y efectos especiales.

Unity ha ganado reconocimiento, entre otras cosas, por su capacidad para desarrollar proyectos que funcionan en múltiples plataformas. No es raro encontrar un mismo juego ejecutándose en PC, consolas, móviles, navegadores Web e incluso en entornos de Realidad Virtual o aumentada. Eso, sin duda, facilita su alcance y distribución.

El lenguaje que se utiliza para programar en esta plataforma es C#, lo cual permite gestionar la lógica del juego y las interacciones con cierta precisión. Y sí, también hay que decir que su editor visual, bastante intuitivo, por cierto, hace que crear o modificar escenas y objetos no sea tan complicado. Herramientas para manejar físicas, animaciones o vínculos entre elementos también están ahí, integradas y listas para usar.

Otro aspecto interesante es la llamada Asset Store. Básicamente, funciona como un mercado donde se pueden conseguir —o vender— recursos ya listos, como modelos en 3D, efectos de sonido y otros elementos. Todo esto, claro, acelera bastante el desarrollo, sobre todo cuando los tiempos aprietan.

Desde una perspectiva colaborativa, Unity también pone a disposición funciones para que varios desarrolladores trabajen juntos sin mayores fricciones. Y si bien es una plataforma poderosa, también ofrece opciones para ajustar el rendimiento en dispositivos más modestos, como los teléfonos.

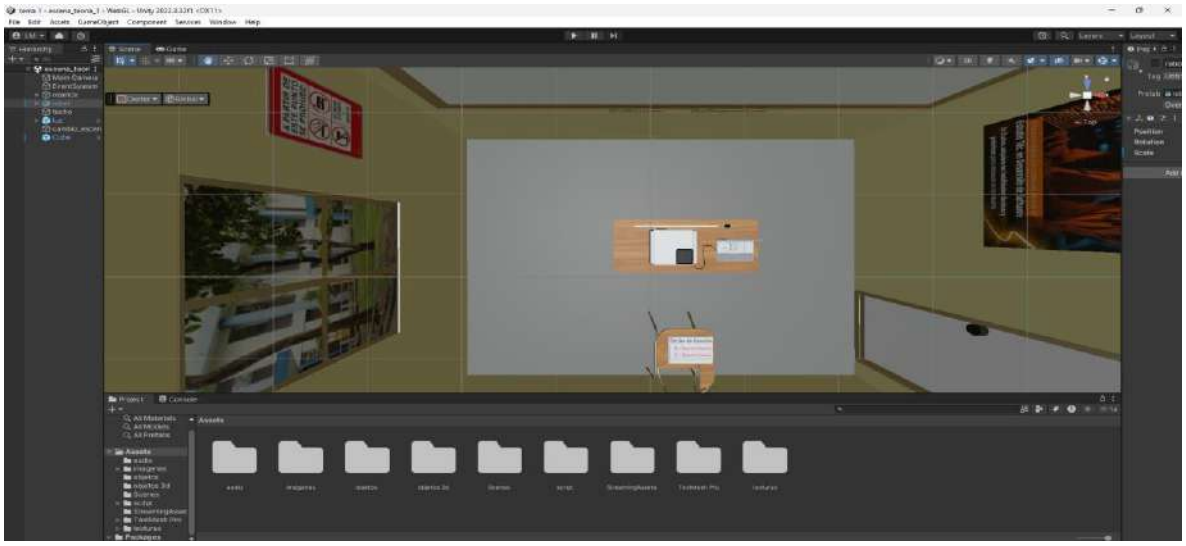

Fig. 5. Interfaz Unity.

5.7.5. VECTARY

La plataforma en línea para diseño 3D y Realidad Aumentada es una herramienta que permite a cualquiera crear y modificar modelos 3D sin necesidad de andar instalando programas complicados. No es que siempre sea perfecta, pero sí facilita bastante el proceso, sobre todo para quienes están empezando o incluso para profesionales que buscan algo rápido y sencillo. Más o menos, es un espacio para trabajar con gráficos tridimensionales sin muchas vueltas, y eso la hace bastante útil, digamos, para todo tipo de usuarios que quieran experimentar o hacer cosas serias. Sin duda, es una opción accesible y bastante práctica.

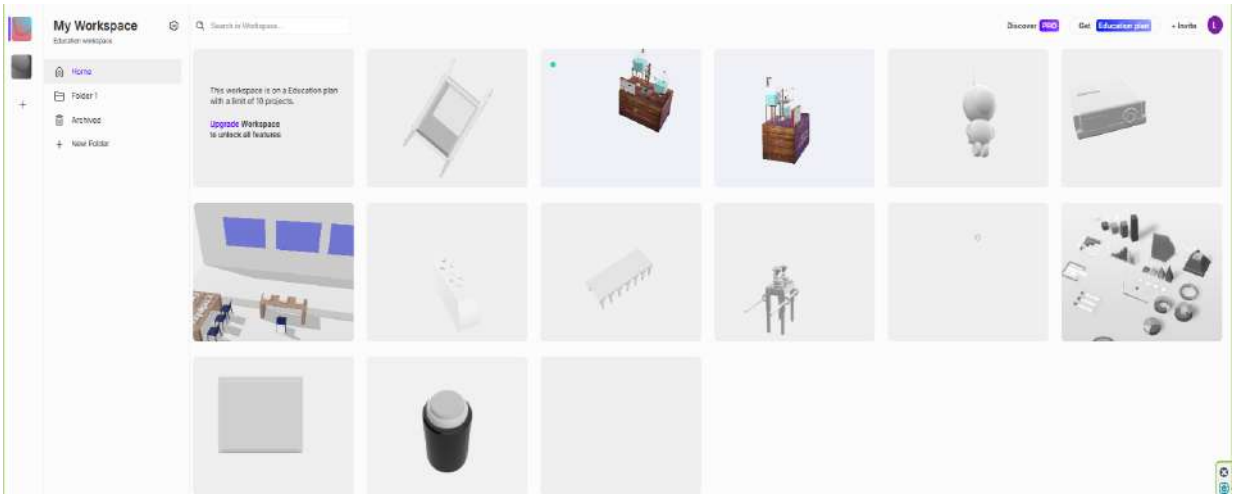

Fig. 6. Interfaz Vectary.

Características principales de Vectary

- Interfaz basada en la Web. No es necesario descargar nada, ya que todo se hace en un navegador Web. Esto lo hace accesible desde diferentes dispositivos.

- Herramientas de diseño 3D fáciles de usar. Ofrece una amplia gama de herramientas para modelado 3D, texturizado, iluminación y animación, diseñadas para ser intuitivas.
- Biblioteca de modelos y recursos. Tiene una biblioteca integrada de modelos 3D prediseñados, texturas y materiales, lo que facilita empezar proyectos rápidamente.
- Colaboración en tiempo real. Permite colaborar con otros usuarios en tiempo real, compartir proyectos y recibir comentarios directamente en la plataforma.
- Integración con Realidad Aumentada (AR). Los modelos creados en Vectary pueden exportarse para ser visualizados en aplicaciones de Realidad Aumentada, lo que es útil para proyectos interactivos y presentaciones.
- Exportación e importación de formatos 3D. Permite exportar los diseños en varios formatos 3D como OBJ, STL, GLTF, y más, lo que lo hace compatible con impresoras 3D y otras plataformas de modelado.

5.7.6. VISUAL ESTUDIO CODE (VS CODE)

Es un editor de código fuente desarrollado por Microsoft. Es gratuito, de código abierto, y está disponible para Windows, macOS y Linux. VS Code está diseñado para ser rápido, liviano y flexible, con un enfoque en el desarrollo de aplicaciones modernas y en la facilidad de uso para los desarrolladores de todos los niveles.

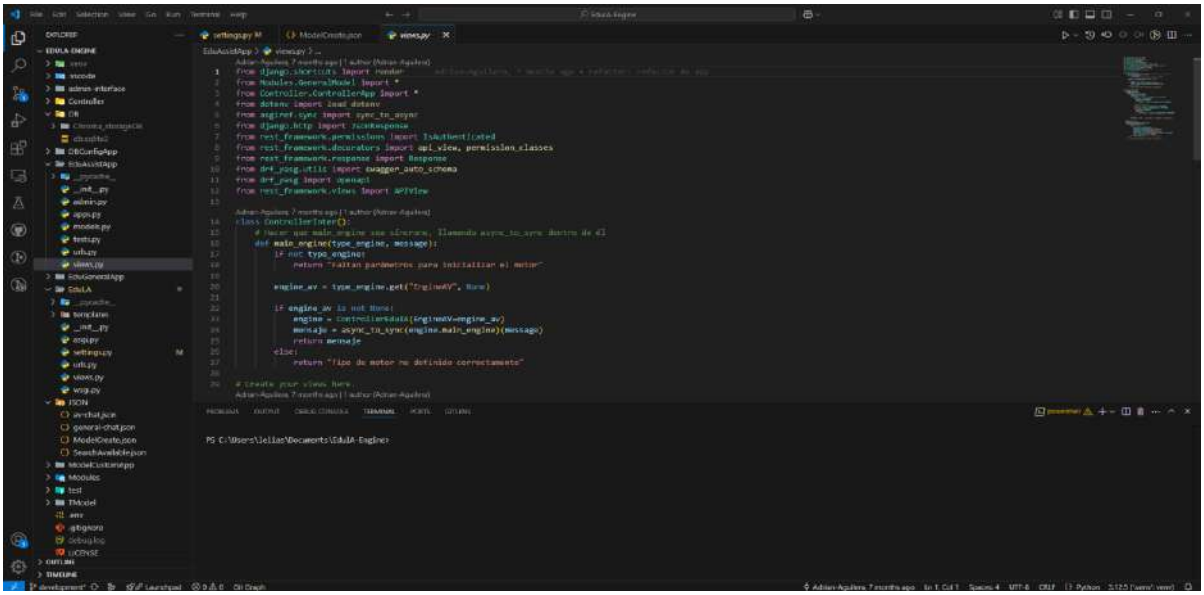

Fig. 7. Interfaz VS Code.

Características principales de Visual Studio Code

- Soporte para múltiples lenguajes. VS Code admite muchos lenguajes de programación de forma predeterminada, incluidos JavaScript, Python, C++, PHP, Java, HTML, CSS, y más. Además, con la instalación de extensiones, puedes agregar compatibilidad con otros lenguajes.
- Extensiones y personalización. A través de su mercado de extensiones, puedes agregar funcionalidades como depuración específica, integración con Git, temas visuales, herramientas de productividad y soporte para nuevos lenguajes.
- Depuración integrada. VS Code ofrece herramientas de depuración integradas para varios lenguajes, lo que te permite depurar aplicaciones directamente desde el editor sin necesidad de cambiar a otra herramienta.

- Control de versiones (Git). VS Code tiene integración nativa con Git, lo que facilita la gestión de repositorios, la creación de commits, el manejo de ramas y otras tareas relacionadas con el control de versiones directamente desde el editor.
- Autocompletado y resaltado de sintaxis. Cuenta con IntelliSense, una función de autocompletado que sugiere código mientras escribes, lo que agiliza el desarrollo. También ofrece resaltado de sintaxis avanzado para una mejor legibilidad del código.
- Multiplataforma. VS Code es compatible con Windows, macOS y Linux, lo que lo hace muy versátil para diferentes entornos de desarrollo.
- Live Share. Una extensión que permite la colaboración en tiempo real. Los desarrolladores pueden trabajar juntos en el mismo código, con la capacidad de depurar de manera colaborativa y compartir terminales y sesiones de depuración en vivo.
- Terminal integrado. VS Code tiene un terminal incorporado que permite ejecutar comandos directamente dentro del editor, facilitando la gestión de proyectos sin tener que cambiar de ventana.
- Formatos y linters. A través de extensiones, VS Code admite herramientas como ESLint o Prettier, que ayudan a mantener la consistencia y calidad del código al aplicar reglas de estilo de codificación y revisar errores.
- Temas y diseño personalizable. Los usuarios pueden modificar la apariencia del editor instalando temas y personalizando el diseño según sus preferencias.

5.7.7. GitHub

Es una plataforma en línea para el alojamiento y gestión de código fuente que utiliza Git, un sistema de control de versiones distribuido. GitHub permite a los desarrolladores colaborar en proyectos, compartir código, realizar un seguimiento de los cambios y gestionar diferentes versiones de sus proyectos. GitHub es muy popular entre desarrolladores individuales, equipos de software y organizaciones, ya que facilita la colaboración y la contribución a proyectos de código abierto o privados.

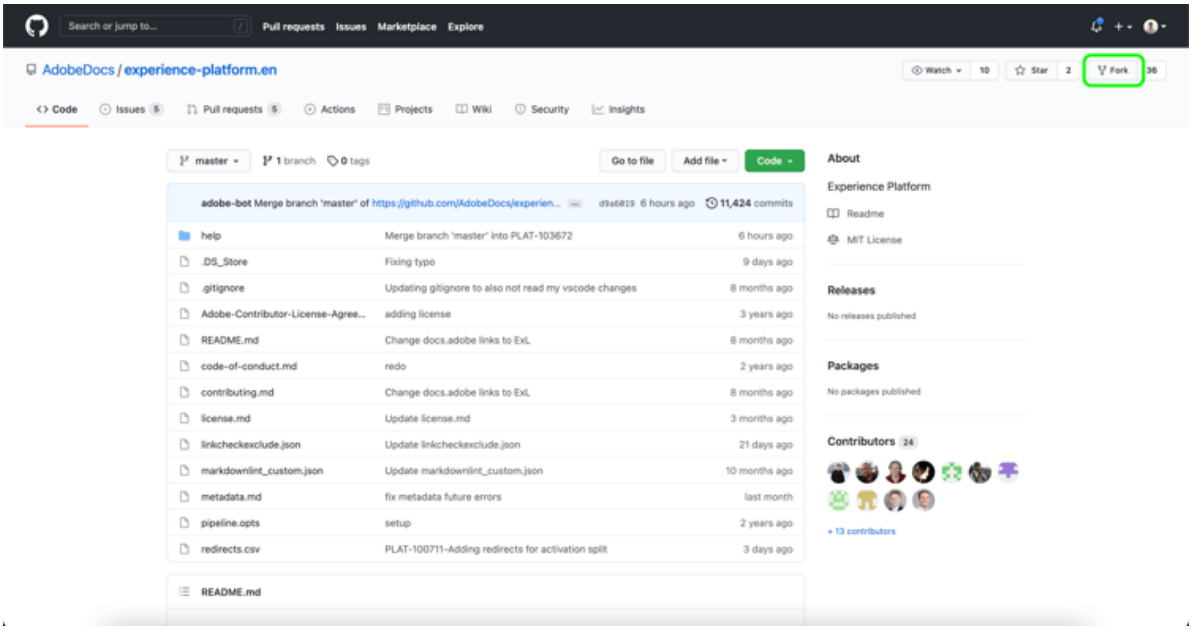

Fig. 8. Interfaz GitHub.

Características principales de GitHub

- Repositorio (Repository). Un repositorio es un lugar donde se almacena el código de un proyecto. Puedes crear repositorios públicos (visibles para todos) o privados (solo accesibles para colaboradores autorizados).
- Control de versiones con Git. GitHub está basado en Git, que es un sistema de control de versiones. Esto permite a los desarrolladores gestionar el historial de cambios en su código, creando y fusionando ramas (branches), revertiendo cambios y más.
- Colaboración. GitHub facilita la colaboración en proyectos de desarrollo. Los desarrolladores pueden trabajar en diferentes partes de un proyecto y luego integrar sus contribuciones usando pull requests. GitHub proporciona herramientas para revisar código, hacer comentarios y discutir cambios propuestos antes de que se fusionen.
- GitHub Actions. GitHub Actions permite automatizar flujos de trabajo, como la compilación, las pruebas o la implementación de aplicaciones, usando el repositorio de GitHub. Es útil para CI/CD (Integración Continua/Despliegue Continuo), donde puedes configurar pipelines que se ejecutan automáticamente cuando ocurre un evento específico, como un nuevo commit o pull request.
- GitHub Pages. GitHub Pages permite alojar sitios Web estáticos directamente desde un repositorio de GitHub. Es común para documentar proyectos, crear blogs personales o publicar sitios Web sencillos de demostración.
- Issues. Los Issues son herramientas que permiten gestionar y hacer seguimiento de errores (bugs), nuevas características o tareas. Los desarrolladores y colaboradores pueden discutir los problemas relacionados con el proyecto en la pestaña de Issues del repositorio.
- Wiki. Cada repositorio puede tener una wiki asociada, lo que permite a los desarrolladores crear documentación detallada, guías o recursos para el proyecto. Es útil para explicar cómo funciona el código, las reglas de contribución o cualquier otra información importante.
- Integración con otras herramientas. GitHub se integra con muchas herramientas populares de desarrollo como Slack, Jira, Travis CI, Docker, entre otras. Esto permite gestionar proyectos y automatizar procesos más allá del control de versiones.
- Organizaciones y equipos. GitHub permite la creación de organizaciones, donde múltiples repositorios pueden ser gestionados por diferentes equipos. Es ideal para proyectos de grandes empresas o grupos de colaboración abiertos.
- Seguridad. GitHub ofrece diversas funciones de seguridad, como la protección de ramas, autenticación de dos factores (2FA), escaneo de vulnerabilidades en dependencias (GitHub Dependabot), y la posibilidad de configurar permisos granulares para miembros del equipo.
- GitHub Copilot. Una extensión impulsada por Inteligencia Artificial que ayuda a los desarrolladores a escribir código más rápido al sugerir líneas o fragmentos completos de código basados en el contexto del proyecto.

5.7.8. FILMORA

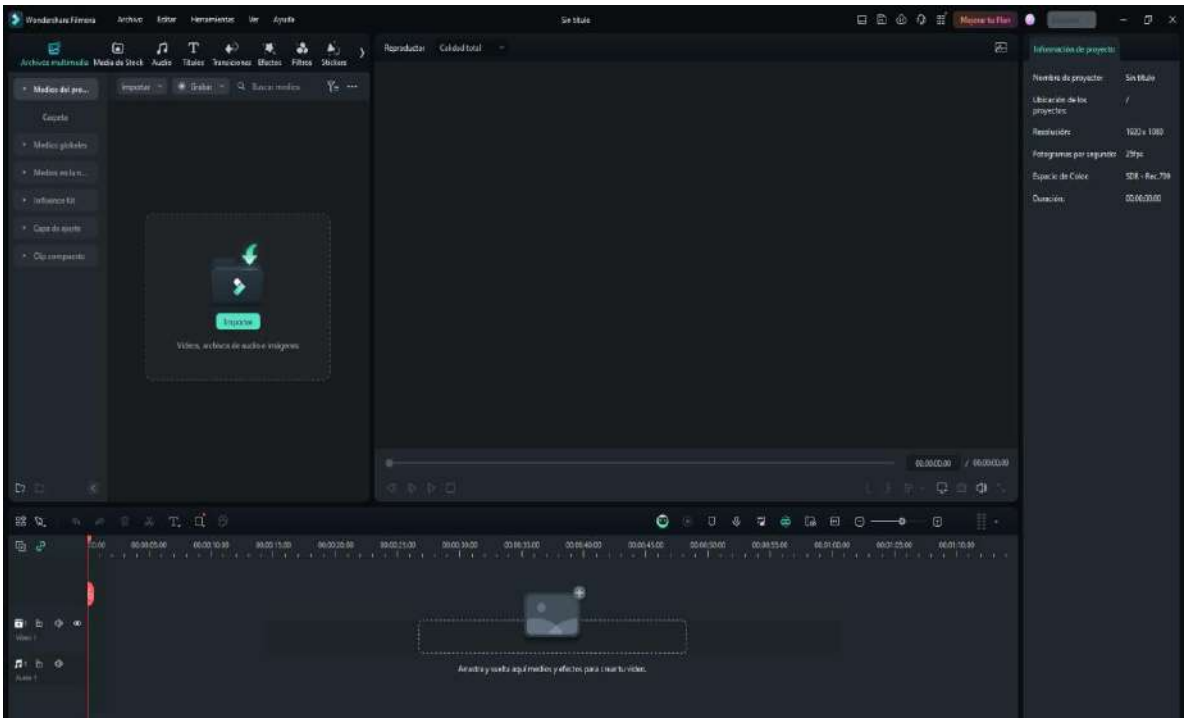

Fig. 9. Interfaz GitHub.

Filmora es un software de edición de video desarrollado por Wondershare. Es conocido por ser fácil de usar y accesible tanto para principiantes como para editores más avanzados. Algunas de las características clave de Filmora incluyen:

- Interfaz intuitiva. Su diseño es sencillo, lo que facilita la navegación y el uso de sus herramientas, incluso para personas que no tienen experiencia en la edición de video.
- Amplia gama de efectos y transiciones. Filmora incluye una gran cantidad de efectos visuales, transiciones, superposiciones y plantillas que permiten crear videos atractivos con poco esfuerzo.
- Edición avanzada. A pesar de ser fácil de usar, Filmora ofrece funciones avanzadas como corrección de color, ajuste de velocidad, mezcla de audio, pantalla dividida, y soporte para chroma key (pantalla verde).
- Soporte para múltiples formatos. El software permite trabajar con una amplia variedad de formatos de video, imagen y audio, lo que lo hace compatible con la mayoría de los dispositivos y plataformas.
- Exportación flexible. Los videos editados pueden exportarse en diferentes formatos, compartirse directamente en redes sociales o subirse a plataformas como YouTube y Vimeo.

6. METODOLOGÍA DE INVESTIGACIÓN

Para esta investigación, se propuso desarrollar un tutor virtual que realmente pudiera apoyar el aprendizaje, utilizando herramientas que tuvieran impacto y sentido dentro del entorno educativo. El objetivo era desarrollar una solución que no solo respondiera preguntas, sino que también ofreciera interacción y acompañamiento constante al estudiante.

El sistema que se desarrollo tiene tres partes principales. La primera fue un módulo de inicio de sesión, que implementamos para que cada usuario pudiera tener su propio espacio seguro. Esto nos permitió controlar el acceso a la plataforma y, al mismo tiempo, guardar el progreso de cada estudiante sin comprometer su privacidad.

Luego, incorporar un chat con Inteligencia Artificial, utilizando un modelo alojado localmente con la herramienta Ollama. Lo que se hiso fue configurar este modelo para que pudiera responder dudas en tiempo real. El propósito era simular la ayuda de un tutor humano, sin que el estudiante dependiera de alguien más para continuar su aprendizaje.

Se desarrolló un entorno en Unity y se exporto en formato WebGL. Este componente permite que el contenido educativo sea más interactivo. Desde cualquier navegador, los estudiantes pueden acceder a simulaciones y actividades que diseñamos especialmente para reforzar lo aprendido en otras secciones del curso.

Todo este trabajo fue realizado de manera colaborativa entre varios miembros del equipo de la Escuela de Ingeniería en Computación. Desde el inicio del proyecto, se organizó usando una matriz metodológica que se definió en la etapa del anteproyecto. Esta matriz sirvió como guía para establecer tareas, asignar responsabilidades y llevar un control claro del avance.

6.1. ANALISIS Y DETERMINACIÓN DE REQUERIMIENTOS

Durante esta etapa se analizaron los requerimientos técnicos necesarios para el desarrollo del tutor virtual, con énfasis en compatibilidad, rendimiento, facilidad de integración y ejecución en entorno Web.

Con base en este análisis, se definieron tres tecnologías clave.

- Django. Se seleccionó por su arquitectura robusta, su sistema de autenticación integrado y su capacidad para gestionar usuarios, sesiones y datos de forma segura y eficiente.

- Unity (WebGL). Se eligió por su compatibilidad con navegadores mediante exportación en WebGL, permitiendo desarrollar entornos gráficos interactivos sin necesidad de instalaciones adicionales.
- Ollama. Se integró como motor local de lenguaje natural, ya que permite ejecutar modelos de IA sin conexión a servicios externos, garantizando velocidad de respuesta y privacidad de los datos.

Estas herramientas fueron seleccionadas tras evaluar sus requerimientos de instalación, recursos del servidor, soporte multiplataforma y facilidad para trabajar de forma conjunta en una misma solución.

6.2. DISEÑO DE INTERFACES (FRONTEND) Y ELEMENTOS (BACKEND)

En esta fase, se procedió con el diseño y maquetado del sistema administrativo de la aplicación, la base de datos y la información que conforman el tutor virtual. Las interfaces se diseñaron pensando en la usabilidad y la accesibilidad, adaptándose al entorno educativo del ITCA-FEPADE.

6.3. FRONTEND

El frontend de la plataforma se desarrolló usando Vue.js, un framework progresivo de JavaScript que resulta bastante flexible y fácil de usar para crear interfaces interactivas. Una de las ventajas de trabajar con Vue es que permite construir componentes por separado, lo que hace que el sistema sea más fácil de mantener y escalar si se quiere agregar más funciones más adelante.

6.3.1. BARRA SUPERIOR

La barra superior actúa como un elemento clave de navegación y personalización dentro de la aplicación. Está compuesta por un logotipo ubicado en la esquina superior izquierda, un texto que da la bienvenida al usuario con la frase "Bienvenido a Edula", un botón de inicio de sesión en la esquina superior derecha y un botón de cambio de tema identificado con un ícono de sol para alternar entre los modos claro y oscuro.

Esta barra se implementó utilizando un elemento <header> que encapsula subelementos estructurados con para organizar su contenido. El diseño se logra mediante el uso de flexbox, garantizando una alineación horizontal perfecta y distribución uniforme de los elementos. El color de fondo rojo (background-color. #FF0000) refuerza la identidad visual de la plataforma, mientras que el botón de tema oscuro/claro está vinculado a un evento onClick que alterna entre estilos predeterminados en el archivo CSS, cambiando las clases aplicadas al elemento <body>. Esto permite una experiencia personalizada para los usuarios según sus preferencias visuales.

La barra superior no solo mejora la navegación, sino que también incluye elementos interactivos clave, como la selección de tema y el acceso directo al inicio de sesión, lo que facilita una experiencia amigable y eficiente.

6.3.2. LOGOTIPO CENTRAL

El logotipo central de ITCA-FEPADE, adaptado al contexto del 55 aniversario, funciona más como un gesto visual que como un simple adorno. Refuerza la identidad institucional y, de paso, establece una especie de vínculo simbólico con quienes acceden a la plataforma. Se integró utilizando una etiqueta de imagen, <img>, y, claro, se optó por una versión optimizada. Esto último, aunque a veces se pasa por alto, tiene implicaciones. tiempos de carga razonables, mejor rendimiento, una respuesta más fluida, sobre todo en dispositivos que no siempre ofrecen lo mejor en conectividad. La imagen se escala bien.

En cuanto al diseño, se posicionó al centro de la página utilizando propiedades CSS como margin. auto; en combinación con flexbox. Esto, que puede parecer un detalle menor, garantiza que el logotipo conserve su ubicación sin importar el tamaño de la pantalla. Es un elemento estático, sí, pero no por eso irrelevante. Más bien, funciona como una especie de recordatorio visual que ayuda a consolidar la presencia de la marca en la interfaz de Edula. Y si bien no tiene funciones interactivas, su peso simbólico dentro de la plataforma no debería subestimarse.

6.3.3. CAJA DE CONVERSACIÓN

La caja de conversación funciona como un espacio interactivo desde el cual es posible iniciar una nueva interacción con el sistema. Visualmente, incluye un pequeño indicador que muestra el estado "En línea" acompañado de un ícono verde, un botón de color naranja que, por su ubicación y diseño, permite acceder a ciertas funciones extra, y un breve mensaje que anima a comenzar con algo como "Inicie una nueva conversación".

Desde el punto de vista técnico, su estructura se basa en un contenedor que organiza subelementos de manera bastante clara. El diseño busca ser limpio, con un fondo gris claro y bordes redondeados que, aunque son detalles menores, logran aportar una sensación de suavidad y orden visual.

Todo esto no es solo estético. Hay cierta lógica detrás. Por ejemplo, la actualización del estado en tiempo real se logra mediante tecnologías como WebSockets o, en algunos casos, AJAX, lo cual permite que el sistema refleje con precisión si está disponible o no. Vale considerar que esta

dinámica contribuye directamente a una experiencia más fluida, ya que quien interactúa con la interfaz no queda con la duda de si el sistema está activo. Y eso, aunque parezca un detalle técnico, tiene su impacto en cómo se percibe el conjunto.

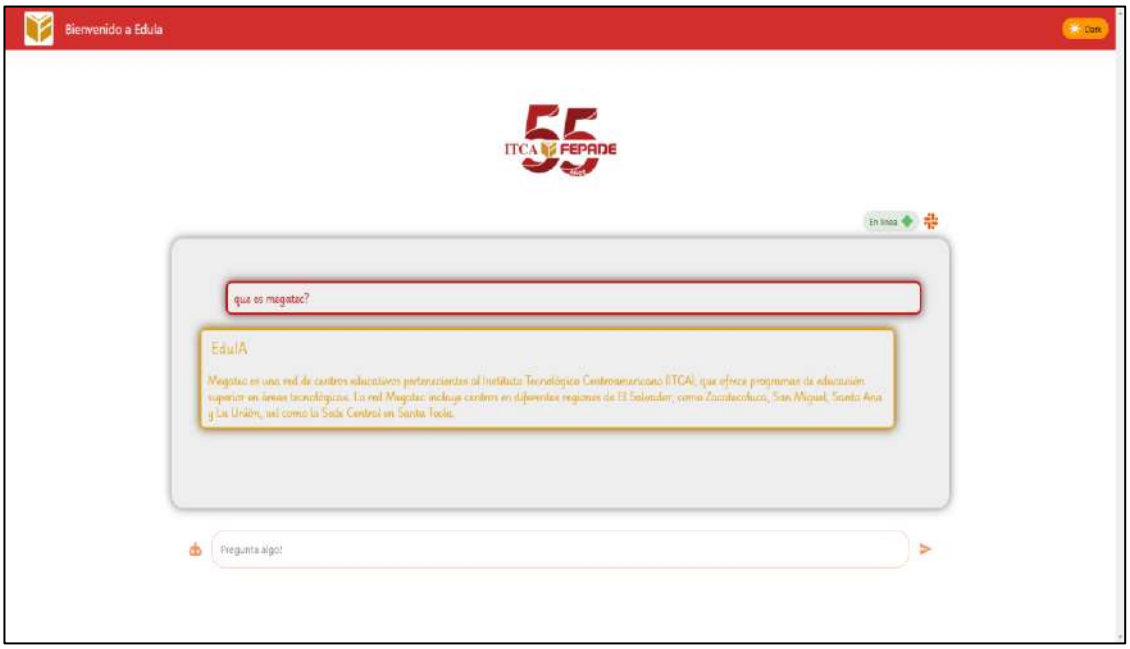

Fig. 10. Caja de conversación.

6.3.4. CAMPO DE ENTRADA PARA PREGUNTAS

El campo de entrada destinado a las preguntas cumple una función clave en la interacción entre el sistema y quien lo utiliza. Se ha diseñado con un enfoque visual bastante accesible. un cuadro de texto acompañado por el mensaje “Pregunta algo” y un botón de envío que incluye un ícono con forma de robot, lo cual, digamos, facilita el reconocimiento inmediato de su propósito. El estilo general, con bordes redondeados y un tono naranja bastante vivo (border-color. #FF7F50;), contribuye a una apariencia amigable y, en cierta medida, más cercana al usuario.

Desde lo técnico, su funcionalidad descansa en un evento onSubmit, que permite capturar lo que se escribe y enviarlo al backend mediante peticiones asíncronas, utilizando tecnologías como fetch o Axios. Esto, sin duda, agiliza el proceso y reduce la espera entre la consulta y la respuesta, algo que se valora especialmente cuando se busca fluidez en el intercambio. El botón, por su parte, fue diseñado para ser muy visible, de forma que no genere confusión ni obstáculos, algo que —vale decir— termina por mejorar de forma considerable la experiencia de uso general.

Fig. 11. Campo de entrada de preguntas.

6.3.5. PÁGINA DE INICIO DE SESIÓN

La página de inicio de sesión se construyó con Vue.js y utiliza la biblioteca Vuetify para proporcionar una interfaz de usuario moderna y accesible. Dentro de la estructura principal de la página se ha dispuesto un contenedor, v-container, que organiza el contenido en un esquema de dos columnas. En la parte izquierda, se encuentra un formulario insertado dentro de una tarjeta v-card, que, gracias

al uso de un componente v-sheet, se mantiene centrado tanto en el eje vertical como en el horizontal. Este formulario, aunque sencillo, reúne varios elementos interactivos. Entre ellos, destacan los campos destinados al ingreso del carnet de usuario y la contraseña. Ahora bien, cada uno de estos campos viene acompañado por íconos que, más allá de su función estética, sirven como guías visuales que facilitan la interacción.

En cuanto a su comportamiento, el campo asignado al carnet permite únicamente el ingreso de números, lo cual evita errores desde el inicio. El campo de la contraseña, por otro lado, incorpora una función para alternar su visibilidad, lo que puede resultar bastante útil en ciertos momentos. Esa opción, que se activa mediante un ícono dinámico, aporta una capa de flexibilidad que, aunque pequeña, hace diferencia durante el uso cotidiano.

El formulario también cuenta con dos botones. uno para enviar las credenciales del usuario y otro para redirigir al usuario a la página de creación de cuenta si aún no tiene una. El botón de inicio de sesión se habilita solo cuando el formulario es válido, es decir, cuando el carnet tiene exactamente seis caracteres y la contraseña no está vacía. En la parte inferior del formulario, se incluye una tarjeta con un mensaje que indica que la plataforma está restringida a estudiantes de Itca Fepade, proporcionando un aviso claro para los usuarios.

A nivel funcional, el componente de Vue maneja varios aspectos del comportamiento del formulario. En el objeto data, se inicializan las variables necesarias para almacenar el carnet y la contraseña del usuario, así como una bandera para controlar la visibilidad de la contraseña y una estructura para las alertas de error o éxito. En los métodos, se definen acciones para validar el carnet (permitiendo solo números) y el inicio de sesión, que envía los datos a la API mediante Axios. La respuesta de la API determina si el inicio de sesión fue exitoso o si las credenciales son incorrectas, y en ambos casos se muestra una alerta al usuario. Si el inicio de sesión es exitoso, los datos del estudiante (como su ID) se almacenan en el localStorage para su posterior uso.

Entre los elementos que llaman la atención en esta página, vale destacar el uso de un diálogo de alerta, pensado para ofrecer retroalimentación inmediata sobre las acciones realizadas. Este componente se activa tanto en situaciones de éxito como de error, y lo interesante es que los mensajes pueden adaptarse según el caso. cambian el título, el contenido... en fin, todo se ajusta a lo que acaba de ocurrir en el proceso de inicio de sesión.

En cuanto al diseño, la página responde bien a distintos tamaños de pantalla. Gracias al uso de media queries, se adapta sin mayor complicación tanto en dispositivos móviles como en escritorios. La columna derecha, que contiene una imagen decorativa, desaparece en pantallas pequeñas. Esto no es casual, claro, se hace para dar más espacio al formulario, que, por cierto, viene dentro de una tarjeta con bordes personalizados y una paleta de colores alineada con la estética general de la plataforma.

Desde lo técnico, se puede decir que la estructura está bien pensada. Hay una integración fluida entre el frontend y el backend, lo cual se nota especialmente en el manejo de datos.

La validación previa, gestionada desde el cliente, ayuda a filtrar errores antes de que cualquier dato llegue al servidor. Es una práctica bastante común, y sí, suele funcionar bien. Aunque, vale considerar que hay margen para ajustes. Por ejemplo, reforzar los criterios de seguridad en las contraseñas no vendría mal. No es que actualmente sea deficiente, pero exigir ciertos mínimos, como una longitud razonable o algún carácter especial, podría marcar una diferencia. También habría que pensar en la claridad de los mensajes de ayuda. utilizan lectores de pantalla o tienen alguna dificultad para navegar.

Fig. 11. Interfaz de login o inicio de sesión.

6.3.6. CAMPO DE FORMULARIO

En el campo del formulario se realizaron distintos elementos usando vuetify, este es un añadido de vuejs que nos ayudan con interfaces gráficas y decorativas, el contexto del formulario que se utilizó para el formulario de iniciar sesión, se toma un componente de formulario en el que se añaden los campos de carnet par los estudiantes y el campo de contraseña, además de estos, en la parte de abajo se añade una advertencia que dice que solo los estudiantes de itca fepade podrán ingresar a la plataforma, porque el campo de carnet esta validado para solo dejar iniciar sesión, si el carnet de estudiante esta como activo en la base de datos de itca fepade.

El capo de formulario esta adentro de un contenedor que este a su vez tiene los displays flex con el sistema rows y cols de “Vuetify”, esto permite que el formulario se pueda adaptar a los distintos tipos de dispositivos en los que se podrá mostrar el formulario para iniciar sesión.

6.3.7. VALIDACIÓN Y FUNCIONAMIENTO DEL SISTEMA DE FORMULARIO

El campo del carnet se validó desde el lado del frontend para que se pueda iniciar sesión, el usuario debe digitar su carnet, del lado del frontend el código se creó la validación para solo dejar ingresar un máximo de 6 dígitos, ya que los carnets de itca fepade son de 6 dígitos, esto es estrictamente necesario ya que la petición axios que se hace al backend está configurada para que se ingrese el carnet de 6 dígitos previamente creado, si se ingresan menos dígitos la petición no encontrará el carnet ya que no hay carnets de menos de 6 dígitos y por lo tanto no dejará iniciar sesión.

El campo contraseña se validó para hacer la petición axios al backend de tal manera que, al hacer la petición, busca la contraseña encriptada en el backend y lo retorna con verdadero o falso, si la contraseña coincide con el carnet asignado a la hora de crear la cuenta de estudiante, el sistema validará que es correcto.

El formulario validado antes de decir si se puede o no iniciar sesión si no hay campos vacíos, en caso de haber un campo del formulario vacío, este mismo no dejara mandar el formulario, esto es una protección que se implementa porque las peticiones que se hacen al backend tiene que ir completas, de tal modo que no genere un error en el backend a la hora de mandar una petición axios.

6.3.8. RESPONSIVIDAD DE LA PÁGINA

La página implementa dos contenedores principales, a la hora de mostrar la página estos están configurados con los elementos responsive de vuetify, con una distribución de cols de 30% y 70% del contenedor principal de toda la página, cuando la pantalla se hace pequeña, el elemento decorativo, en este caso la "img" va reduciendo su tamaño para solo dejar a la vista lo importante que es el formulario. Este sistema implementa un @media que al llegar a los 970px de ancho el formulario queda centrado y la imagen decorativa desaparezca.

6.3.9. LA PÁGINA CREAR CUENTA

La página está desarrollada en Vue.js utilizando la biblioteca Vuetify para componentes de interfaz de usuario. El código está organizado en una plantilla (<template>), lógica de datos y métodos (<script>), y estilos (<style>). Está diseñado para ser una página de registro de usuarios (creación de cuentas).

En el campo de carnet cuando se pone se hace una validación para que solo se puedan ingresar un máximo de 6 dígitos porque los carnets de itca fepade utilizan un máximo de 6 dígitos para sus carnets.

El campo de ingresar una contraseña para la cuenta es un campo diseñado para que el usuario tenga libre elección para ingresar la contraseña que desee, pero esta misma esta validad para que no acepte caracteres especiales, solo letras y números.

El campo de confirmar contraseña esta validado para que solo se mande la petición axios para crear la cuenta si la contraseña es la misma que la ingresada en el campo anterior, si no es la misma, saldrá una alerta diciéndole que las contraseñas no coinciden entre sí y no dejará crear la cuenta.

La página desarrollada en Vue.js utiliza la biblioteca Vuetify para construir una interfaz de usuario enfocada en la creación de cuentas de usuario. La estructura principal se organiza mediante un contenedor (v-container) que ocupa toda la altura de la ventana del navegador y se divide en dos columnas (v-col). La columna izquierda, identificada como Con1, contiene un formulario centrado dentro de una tarjeta (v-card). Este formulario incluye varios elementos. un logotipo representado por un componente v-img, un campo de texto para ingresar el carnet del usuario, un campo para la contraseña, y otro para confirmar la contraseña. Además, se presentan dos botones principales. uno para crear una cuenta y otro para redirigir al usuario a la página de inicio de sesión en caso de que ya tenga una cuenta. Los campos de entrada están configurados con validaciones específicas, como la longitud máxima del carnet (seis caracteres) y la posibilidad de mostrar u ocultar las contraseñas con un ícono interactivo.

Por otro lado, la columna derecha, etiquetada como Con2, está diseñada para mostrar una imagen de fondo decorativa que complementa visualmente la página. Este elemento se oculta automáticamente en dispositivos con pantallas pequeñas, asegurando una experiencia de usuario responsiva gracias al uso de clases personalizadas y media queries. En términos estéticos, los estilos están definidos en la sección <style> del componente y aplican exclusivamente al mismo. Algunos de los estilos destacados incluyen el diseño de la tarjeta con bordes personalizados y la imagen de fondo para la columna derecha, que se adapta bien a pantallas de diferentes tamaños.

La funcionalidad de la página está implementada en la sección <script> del componente. En el objeto data se definen variables para almacenar los valores del formulario, como el carnet, la contraseña, y su confirmación, junto con una bandera para alternar la visibilidad de las contraseñas y una estructura para manejar alertas. En la sección de propiedades computadas, se verifica la validez del formulario mediante una función que evalúa si todos los campos están completos, si las contraseñas coinciden y si el carnet tiene la longitud requerida. Entre los métodos principales destacan la validación del campo "carnet" para aceptar solo números, la navegación hacia la página de inicio de sesión, y la función que envía los datos del formulario a una API mediante Axios. Este último método realiza validaciones adicionales, como verificar la coincidencia de las contraseñas antes de enviar los datos, y muestra alertas personalizadas dependiendo del resultado de la operación.

La página también incorpora un componente de diálogo (v-dialog) que permite mostrar mensajes de éxito o error. Este elemento mejora la experiencia del usuario al proporcionar retroalimentación clara y directa en respuesta a las acciones realizadas. Por ejemplo, si el registro es exitoso, el usuario es notificado con un mensaje antes de ser redirigido automáticamente a la página de inicio de sesión.

Finalmente, desde un punto de vista técnico, el diseño general del frontend es funcional y bien estructurado. El uso de Vuetify asegura un diseño limpio y responsivo, mientras que la integración con el backend mediante Axios facilita la comunicación con el servidor. Sin embargo, hay oportunidades de mejora, como incluir validaciones más avanzadas para las contraseñas (por ejemplo, exigir una longitud mínima o caracteres especiales) y mostrar mensajes de error directamente debajo de los campos para una mejor comprensión. También sería beneficioso considerar prácticas de accesibilidad, como añadir descripciones para los lectores de pantalla, mejorando así la experiencia de usuarios con discapacidades.

Fig. 12. Interfaz creación de cuenta.

6.3.10. PÁGINA CHAT PRINCIPAL PARA ESTUDIANTES

La estructura de la página se dividió en dos columnas principales dentro de un contenedor de Vuetify (v-container), usando un diseño flexible con el sistema de rejillas (grid). Se utiliza el componente v-row y v-col para crear las columnas de manera responsiva.

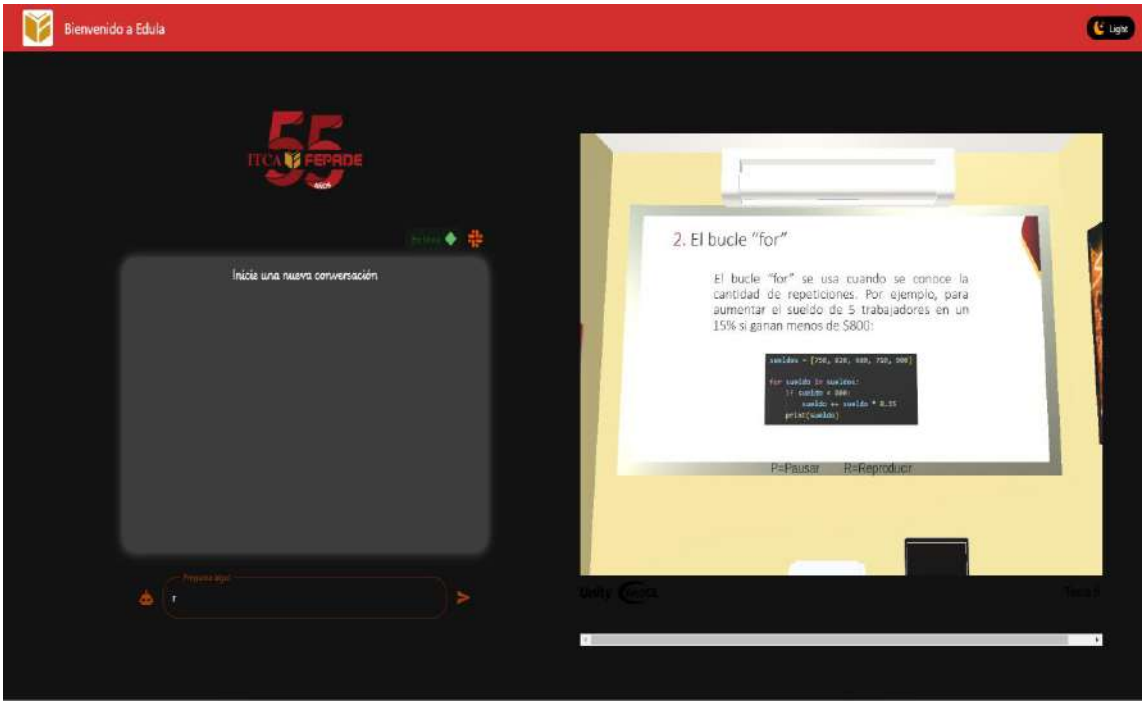

Fig. 12. Interfaz del chat educativo.

- **Columna izquierda**

Contiene

- Logo. Se muestra una imagen centrada con el logo de ITCA.
- Estado de conexión. Muestra un chip que indica si el chatbot está "En línea" o "Desconectado", además de un icono de estado.
- Historial de Conversaciones. Se muestran las conversaciones previas entre el usuario y el chatbot en un contenedor con desplazamiento vertical. Si no hay historial, se muestra un mensaje indicando que el usuario debe iniciar una nueva conversación.
- Formulario de Entrada. Un campo de texto para que el usuario ingrese sus preguntas y un botón de envío que activa la función sendMessage.

- **Columna derecha**
- Selector de Tema. Un v-select para elegir un tema que cambia la URL de un iframe embebido.
- Iframe. Muestra un contenido relacionado con el tema seleccionado, cargando una página de contenido HTML.

- **Componentes Utilizados**
- AppBarComponent2. Un componente de barra de navegación personalizado que se incluye al principio de la página.
- v-img. Para cargar el logo de la institución.
- v-chip. Para mostrar el estado de conexión del chatbot.
- v-text-field. Usado para la entrada de texto de las preguntas.

- v-select. Permite seleccionar un tema.
- v-card. Utilizado para mostrar las preguntas y respuestas de manera estilizada.
- v-progress-circular. Muestra una animación de carga cuando el sistema está procesando la pregunta.

- **Interacción con el Chatbot**
- El usuario puede hacer preguntas mediante el campo de texto, que, al ser enviado, hace una llamada a una API para obtener una respuesta de un chatbot (usando la función sendMessage).
- La respuesta del chatbot se muestra en un historial, que se actualiza dinámicamente a medida que nuevas preguntas y respuestas son intercambiadas.
- Los mensajes se envían solo si son válidos, y se valida que el campo no esté vacío con ValidateCampos.

- **Historial de Conversaciones**
- Se carga el historial de conversaciones previas con el chatbot mediante la función LoadHistory. Si no hay conversaciones, muestra el mensaje "Inicie una nueva conversación".

- **Conexión Activa o Inactiva**
- El estado de conexión del chatbot se maneja a través de la variable isActivo. Al hacer clic sobre el chip de estado, se cambia entre "En línea" y "Desconectado", y también se hace una petición API para obtener el estado real.

- **Selección de Tema**
- Un v-select permite al usuario seleccionar un tema. El contenido del iframe cambia según el tema seleccionado (por ejemplo, se cambia la URL para cargar diferentes páginas HTML relacionadas con cada tema).
- Diseño Flexbox. Se usa el sistema de rejillas y v-layout de Vuetify para organizar los componentes en columnas que se ajustan de manera responsiva en diferentes tamaños de pantalla.
- Colores y Estilo. Se utiliza una paleta de colores que se enfoca en tonos naranjas y dorados para los elementos clave como los botones y los bordes. También hay un énfasis en usar fuentes estilizadas (con la familia Playwrite GB S para los textos).

- **Elementos de Interfaz**
- Cards. Se utilizan tarjetas (v-card) para mostrar los mensajes tanto del usuario como de la IA. Estas tienen bordes personalizados, sombras y colores para diferenciarlas.
- Input de Mensaje. El campo de entrada de texto tiene un icono de "enviar" al lado derecho, el cual cambia a un indicador de carga (v- progress-circular) mientras se espera la respuesta de la IA.
- Scrollbar Personalizado. El contenedor de historial tiene un scrollbar personalizado con un estilo específico para mejorar la experiencia de navegación.

6.3.11. COMPORTAMIENTOS Y LÓGICA DEL BACKEND

- **Funciones Principales**
  - sendMessage. Envía el mensaje del usuario a la API y actualiza el historial con la respuesta de la IA.
  - LoadHistory. Recupera el historial de conversaciones previas desde la API para mostrarlas al usuario.

- isActive. Cambia el estado de conexión del chatbot entre "En línea" y "Desconectado".

- **Validación de Campos**
  - Los campos de entrada son validados antes de ser enviados. Si el campo está vacío, muestra un mensaje de alerta al usuario.

- **Tokens de Autenticación**
  - La autenticación se realiza utilizando un token de acceso (VUE_APP_ACCESS_TOKEN) que se incluye en las cabeceras de las peticiones API.

- **Estilos Personalizados**
- Estilos para Cards y Mensajes.
- Se definen clases CSS para personalizar la apariencia de los mensajes del usuario y la IA (con bordes de colores específicos y sombras).
- Scrollbar. El scrollbar del historial de conversaciones tiene un diseño propio con un color que combina con la paleta de la página.

Este frontend se diseñó para proporcionar una experiencia de usuario fluida e interactiva con el chatbot. Permite enviar mensajes, ver respuestas, revisar el historial de conversaciones y cambiar de tema en el contenido del iframe. Además, la interfaz es visualmente atractiva y fácil de usar, con un diseño responsivo y elementos de UI personalizados como chips, tarjetas, iconos y barras de progreso.

6.4. BACKEND (ESTRUCTURA)

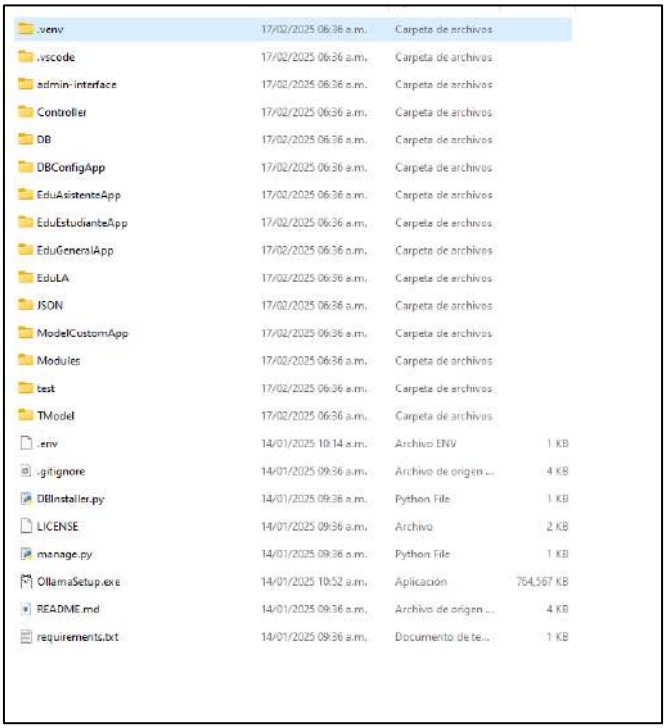

Fig. 13. Estructura de aplicación.

6.4.1. VENV

La carpeta. venv es un entorno virtual en Python, lo que significa que dentro de esta carpeta se encuentran todas las bibliotecas, dependencias y configuraciones necesarias para tu proyecto, aisladas del resto de tu sistema.

Este entorno virtual permite ejecutar tu proyecto sin interferir con otros proyectos o las configuraciones globales de Python que tengas instaladas en tu equipo.

Fig. 14 Carpeta. VENV.

6.4.2. .VSCODE

La carpeta. vscode se utilizó en este proyecto, especialmente porque se trabajó con Visual Studio Code (VS Code). Esta carpeta se utilizó para guardar configuraciones específicas del editor que afectan el comportamiento y la apariencia del proyecto dentro de VS Code.

Contenido de la carpeta.

- Configuraciones personalizadas para el editor. Dentro de esta carpeta, se utilizó el archivo settings.json, que permite personalizar diversas opciones del editor, como el formato del código, las reglas de estilo, la integración con linters y configuraciones relacionadas con la depuración.
- Trabajo colaborativo. Si el proyecto se comparte con otros desarrolladores, las configuraciones dentro de la carpeta. vscode también se utilizaron para asegurar que todos los miembros del equipo usen el mismo entorno de desarrollo, con las mismas reglas y preferencias, lo que facilita la colaboración y mejora la consistencia del proyecto.
- Configuración del entorno de desarrollo. Además, se utilizó la carpeta. vscode para incluir configuraciones de tareas automatizadas, como la ejecución de scripts o compilaciones, y configuraciones para integrar herramientas externas, mejorando la experiencia de desarrollo y haciéndola más eficiente.

6.4.3. ADMIN-INTERFACE

La carpeta ADMIN-INTERFACE se utilizó para almacenar los logos e iconos del logo de la institución. Estos elementos gráficos se emplearon en la interfaz de usuario de la aplicación, sirviendo para representar visualmente la identidad de la institución y mejorar la experiencia del usuario dentro de la aplicación.

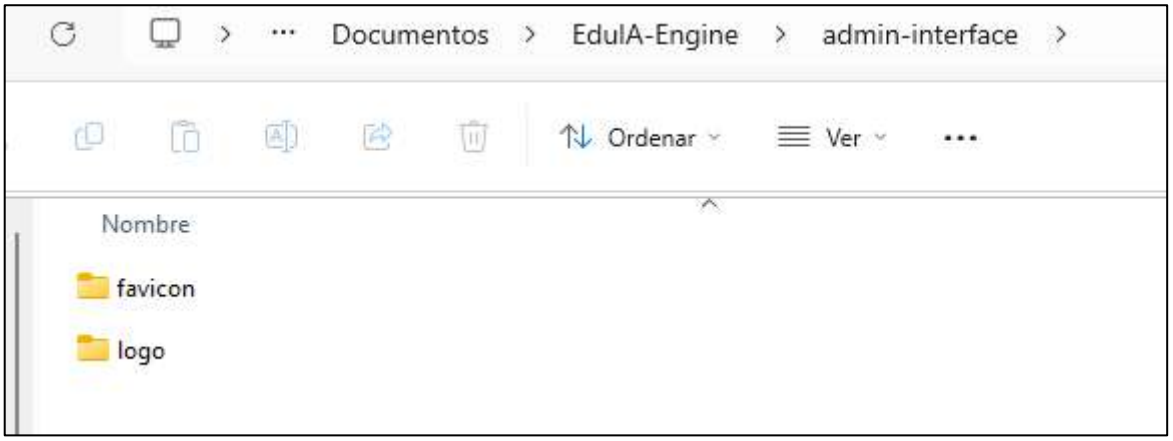

Fig. 15. Carpeta. Admin-interface.

6.4.4. CONTROLLER

La carpeta controller se utilizó para organizar el código encargado de coordinar y gestionar la interacción entre los usuarios y el asistente de IA, la base de datos y las colecciones que almacenan el contexto y la información utilizada para generar respuestas. Se incluyen controladores como.

- ControllerAsistenteChat.py. Gestiona las respuestas del asistente basadas en el contexto de la base de datos.
- ControllerGeneralChat.py. Similar al anterior, pero orientado a la gestión de interacciones generales con el asistente.
- DBController.py. Se encarga de la creación y manejo de colecciones de datos en la base de datos que se utilizan para generar los contextos de IA.

Esta estructura facilita el mantenimiento, la escalabilidad y la integración de nuevas funcionalidades, separando las responsabilidades de los diferentes módulos dentro del sistema.

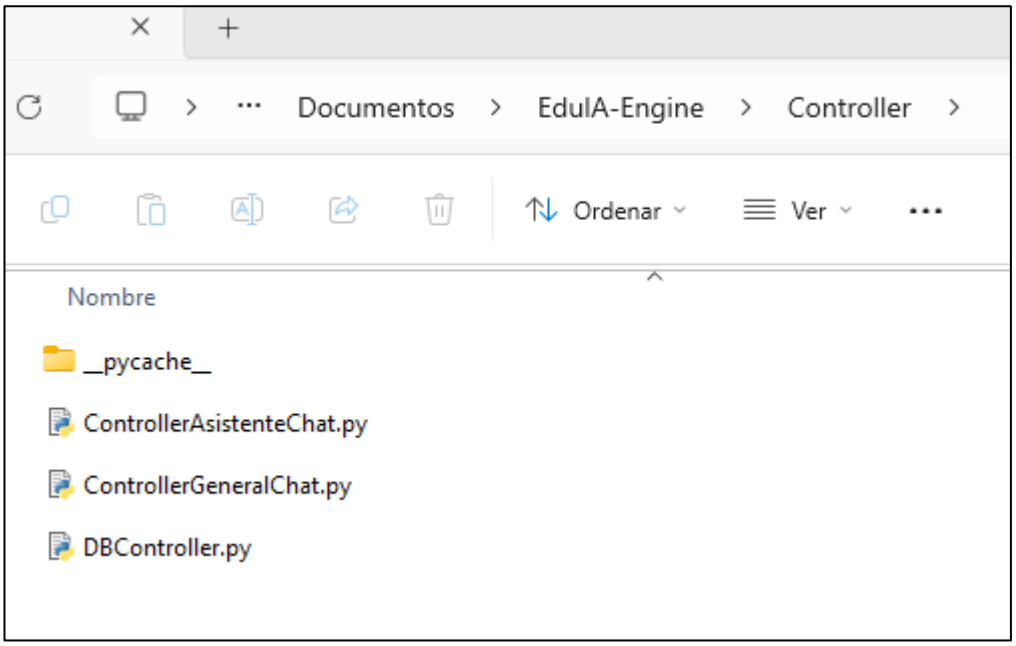

Fig. 16. Carpeta Controller.

6.4.5. DB

La carpeta DB se utilizó para almacenar los datos y colecciones relacionados con la aplicación. Dentro de esta carpeta, Chroma_storageDB maneja los embeddings de texto generados para el asistente de IA, mientras que db.sqlite3 es la base de datos principal que contiene los registros y datos de la aplicación. Ambos elementos trabajan juntos para proporcionar la infraestructura de datos necesaria para las interacciones de IA, almacenando tanto información estructurada como no estructurada que puede ser utilizada en las consultas y respuestas del sistema.

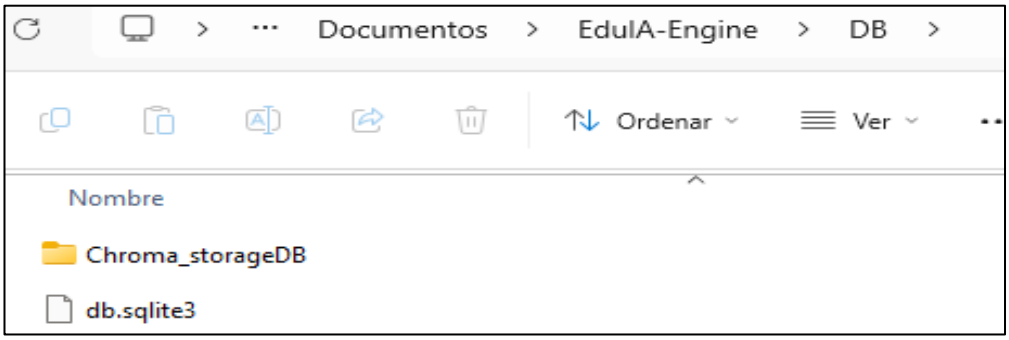

Fig. 17. Carpeta DB.

6.4.6. DBCONFIGAPP

Esta carpeta se utilizó para contener los archivos necesarios para configurar y gestionar la base de datos en tu proyecto Django. Aquí se definen modelos de datos, rutas (URLs) para interactuar con esos modelos, vistas para manejar las solicitudes, serializers para la conversión de datos, pruebas para asegurar que el sistema funcione correctamente y configuraciones para la integración en el sistema general.

- models.py. Define las tablas de la base de datos.
- serializers.py. Convierte los modelos en formatos adecuados (JSON, etc.).
- views.py. Controla la lógica de presentación.
- urls.py. Define las rutas para acceder a las vistas.
- admin.py. Registra los modelos en la interfaz de administración de Django.
- apps.py. Configura la aplicación dentro del proyecto Django.

Esta organización permite que la aplicación sea escalable, modular y fácil de mantener, cumpliendo con las prácticas comunes de Django para la estructuración de aplicaciones Web.

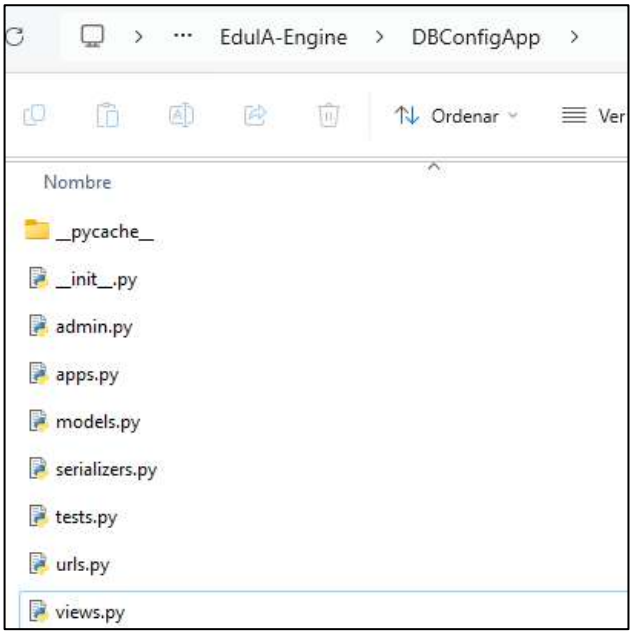

Fig.18. Carpeta DBConfigApp.

6.4.7. EDUASISTENTEAPP

Esta carpeta contiene el conjunto de archivos correspondiente a la aplicación Web desarrollada con Django para interactuar con los usuarios. A través de esta aplicación, los usuarios (en este caso, estudiantes) pueden realizar preguntas y recibir respuestas generadas por el asistente, mientras se mantiene un historial de las interacciones. El sistema también ofrece funcionalidades para consultar y limpiar el historial de preguntas y respuestas.

La aplicación incluye modelos de datos que gestionan las preguntas de los usuarios, las respuestas del asistente y la autenticación de los usuarios. Además, ofrece una API para facilitar la interacción del asistente con los usuarios a través de peticiones HTTP.

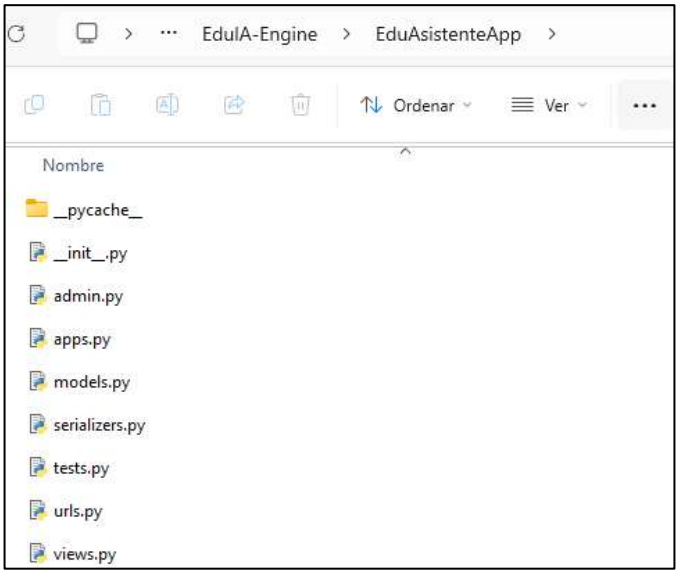

Fig.19. Carpeta EduAsistenteApp.

6.4.8. EDUESTUDIANTEAPP

La carpeta fue creada para gestionar la información y el perfil de los estudiantes en el sistema educativo. Esta carpeta se encarga de almacenar y manipular datos como el carnet del estudiante, nombre, y otros detalles relevantes. El sistema de modelos en esta aplicación incluye características como la validación de la existencia del estudiante, la gestión de su historial académico, y la conexión con otras aplicaciones o módulos como el asistente virtual de EduAsistenteApp.

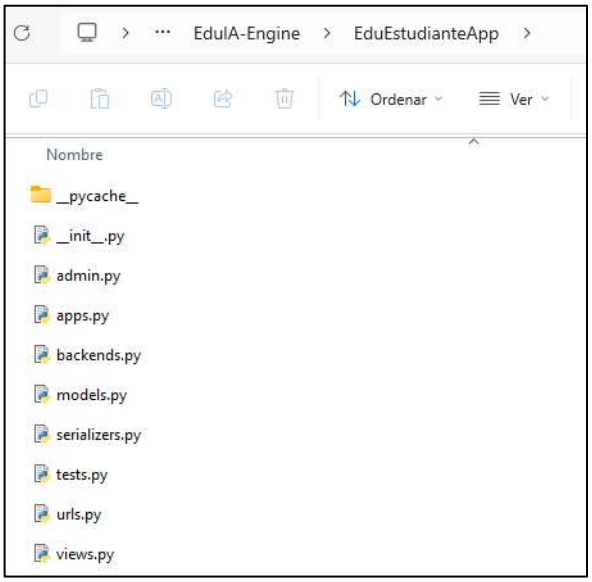

Fig.20. Carpeta EduEstudianteApp.

6.4.9. EDUGENERALAPP

La carpeta EduGeneralApp facilita la gestión de interacciones a través del chat general, permitiendo la creación de colecciones únicas para organizar estas interacciones. Además, proporciona funcionalidades para cargar y gestionar archivos, especialmente en formato PDF, que pueden estar relacionados con el contenido compartido en las conversaciones o interacciones. Su estructura permite un manejo eficiente de los datos, asegurando la unicidad de las colecciones y la correcta gestión de los archivos.

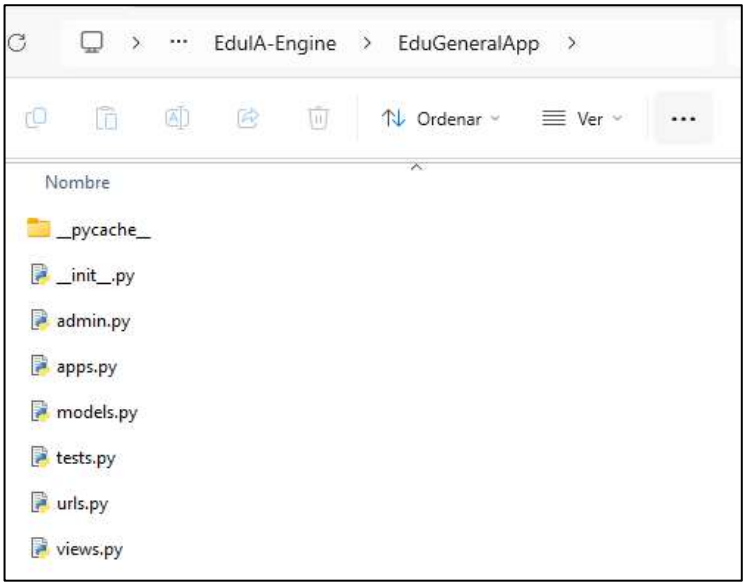

Fig.21. Carpeta EduGeneralApp.

6.4.10. EDULA

La carpeta EduLA en el proyecto Django es el directorio principal del proyecto, que contiene la configuración central y los archivos necesarios para que la aplicación funcione correctamente, el propósito de algunos de sus elementos clave.

- Configuración global del proyecto.
- El archivo settings.py contiene la configuración principal del proyecto, como la base de datos, middleware, autenticación, archivos estáticos, etc.
- urls.py define las rutas o URLs que el proyecto utilizará para manejar las peticiones HTTP y dirigirlas a las vistas correspondientes.
- wsgi.py y asgi.py son archivos de configuración necesarios para que Django pueda ejecutar la aplicación tanto en servidores WSGI (para producción) como en servidores ASGI (para manejar solicitudes asíncronas).
- Archivos estáticos y plantillas.
- La carpeta static/ almacena archivos estáticos como imágenes, CSS, JavaScript, que serán accesibles desde la Web.
- templates/ contiene las plantillas HTML que se renderizan en las vistas del proyecto, proporcionando la interfaz de usuario.
- Aplicaciones del proyecto.
- Dentro de la carpeta EduLA,se puede ver que se incluyen varias aplicaciones específicas como EduGeneralApp, EduAsistenteApp, EduEstudianteApp, que contienen modelos, vistas, y otras configuraciones específicas de cada módulo funcional.

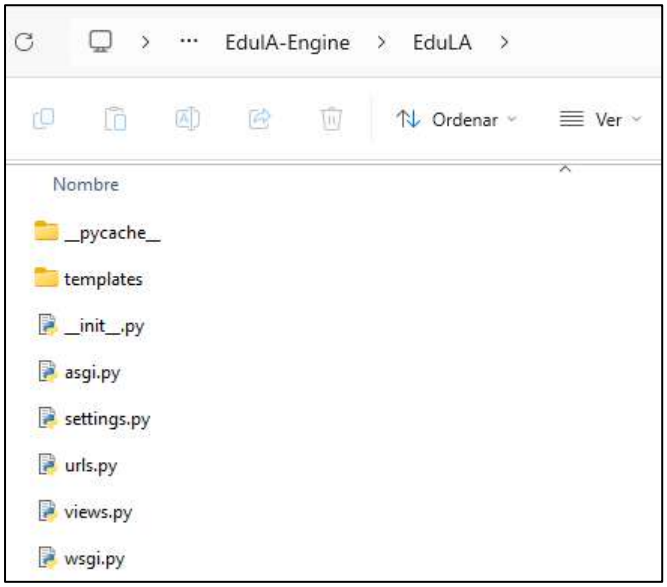

Fig.22. Carpeta EduLA.

6.4.11. JSON

Estos archivos, en conjunto, definen y configuran el sistema de chat inteligente que puede manejar interacciones a través de texto, audio y video. Los archivos almacenan parámetros importantes para personalizar el comportamiento del asistente virtual, incluyendo el modelo de Inteligencia Artificial, su propósito (en este caso, educativo), y su capacidad para responder adecuadamente a las consultas de los usuarios.

Este tipo de configuración permite que los sistemas de IA conversacional funcionen de manera eficiente y personalizada, adaptándose a las necesidades de la aplicación o entorno en el que se utilicen, como en un entorno educativo, empresarial o de servicio al cliente.

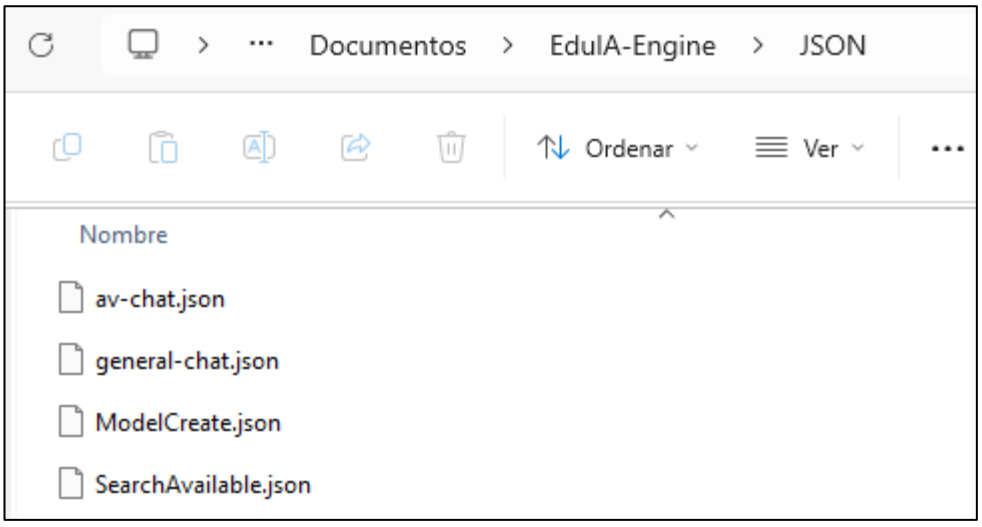

Fig.23. Carpeta JSON.

6.4.12. MODELCUSTOMAPP

Esta carpeta contiene la configuración y gestión de Modelos de Lenguaje de Gran Escala (LLM) a través de un servidor Ollama. A través de diferentes funcionalidades, la aplicación facilita la administración de modelos de lenguaje tanto para tareas generales como para asistentes personalizados, con características como la creación de nuevos modelos, la visualización de los modelos existentes, la búsqueda de modelos específicos y la conexión para generar respuestas a través de estos modelos. La aplicación incluye validaciones para evitar duplicaciones y mantiene la persistencia de datos utilizando bases de datos vectoriales, como Chroma.

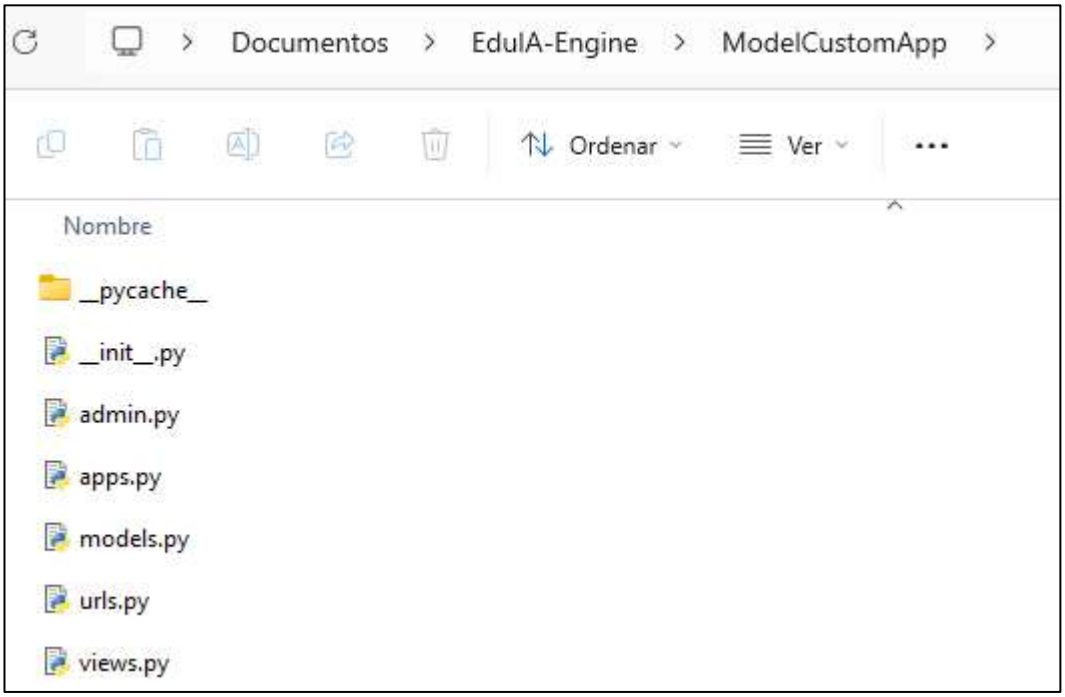

Fig.24. Carpeta ModelCustomApp.

6.4.13. MODULES

La carpeta modules tiene como propósito principal gestionar la interacción con bases de datos de embeddings y generar respuestas contextuales utilizando modelos de lenguaje. Contiene módulos que configuran y manejan la persistencia de datos a través de ChromaDB, lo que permite almacenar representaciones numéricas de texto (embeddings) que posteriormente pueden ser consultadas para obtener información relevante o generar respuestas.

Estos archivos trabajan en conjunto para proporcionar una funcionalidad de procesamiento de lenguaje natural (NLP). Por ejemplo, en el archivo ConfigDBModel.py, se configuran los parámetros de conexión y persistencia de la base de datos de embeddings, mientras que en GeneralModel.py se implementan métodos para interactuar con el modelo de Ollama, encargándose tanto de generar respuestas a partir de un contexto como de generar embeddings basados en los mensajes de los usuarios.

En conjunto, la carpeta habilita la creación y consulta eficiente de una base de datos que asocia contenido textual con sus embeddings, permitiendo ofrecer respuestas más inteligentes, personalizadas y contextuales a las preguntas o entradas del usuario.

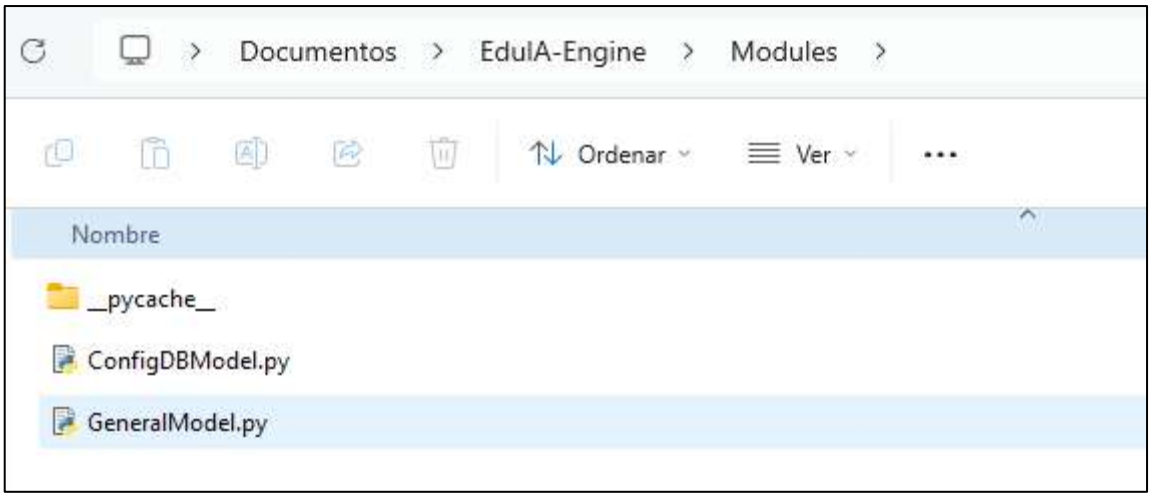

Fig.25. Carpeta Modules.

6.4.14. TEST

Esta carpeta almacena un embedding, que no es otra cosa que una forma numérica, bastante abstracta, de representar datos dentro de un espacio vectorial —y sí, de muchas dimensiones—. Se utiliza con frecuencia en Inteligencia Artificial, sobre todo cuando se busca capturar parecidos semánticos entre palabras, frases o incluso textos más largos. Lo interesante, o al menos lo práctico, es que permite comparar y buscar significados de forma bastante eficiente, incluso en colecciones grandes de información. En sistemas que combinan recuperación y generación —los llamados Retrieve and Generate— estos embeddings funcionan como una especie de filtro previo. ayudan a localizar contenido que parezca relevante y luego, con ese insumo, los modelos de lenguaje se encargan de generar una respuesta que, idealmente, sea coherente, útil, o ambas cosas. Todo esto, digamos, en un solo flujo que une búsqueda y generación sin separarlos del todo.

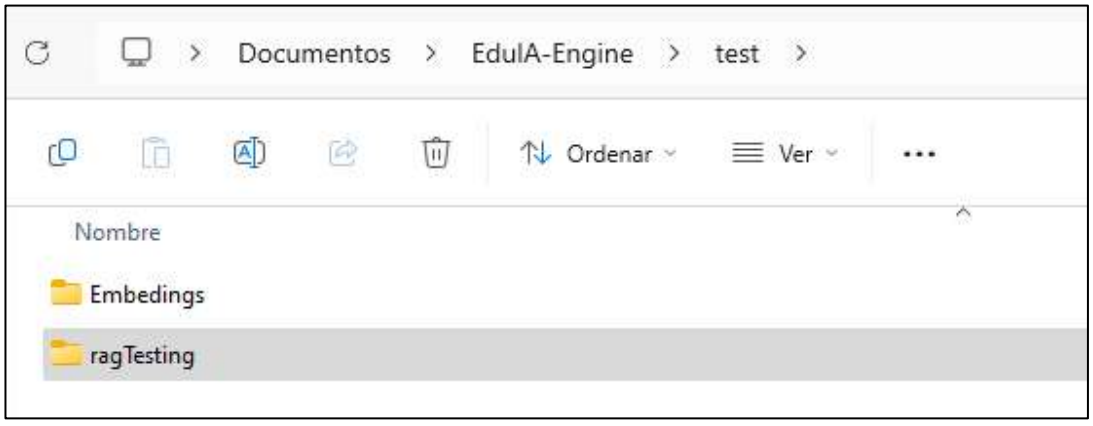

Fig.26. Carpeta EdulA-Engine.

6.4.15. TMODEL

La carpeta sirve para contener y estructurar todos los archivos relacionados con el proceso de interacción con la API y la gestión de las conversaciones con el modelo de lenguaje.

- GModel.py. Este archivo es un script en Python que configura y utiliza la API de Google Gemini para interactuar con el modelo de generación de texto mediante un chat. Permite enviar mensajes al modelo, recibir respuestas en tiempo real y visualizarlas en la consola, creando una experiencia de conversación automatizada con el modelo de IA.
- LModel.py. Este archivo es un script en Python que utiliza la API de OpenAI para interactuar con el modelo de chat y ofrece una experiencia de conversación en la que el usuario puede hacer preguntas y obtener respuestas. Además, implementa una animación de carga mientras se espera la respuesta del modelo, mejorando la experiencia del usuario. Los mensajes de la conversación se almacenan en una lista para mantener el contexto del chat, y el usuario puede finalizar la conversación escribiendo "salir".
- Ollama_Model.py. Este script es una implementación que utiliza Inteligencia Artificial para procesar y almacenar textos históricos en una base de datos vectorial, y luego genera respuestas contextuales a preguntas basadas en ese contenido. El flujo incluye la generación de embeddings para representar los textos, el almacenamiento de estos embeddings en una base de datos persistente utilizando chromadb, y la consulta de la base de datos para recuperar información relevante. Posteriormente, utiliza el modelo de generación de texto de ollama para proporcionar respuestas a las preguntas del usuario, todo en tiempo real.
- ollamaCreate.py. Este script permite interactuar con el asistente virtual educativo basado en el modelo de lenguaje llama2, utilizando la librería ollama. El asistente se configura para hablar únicamente en español y responder exclusivamente sobre temas educativos. A medida que el usuario ingresa preguntas, el modelo genera respuestas contextuales basadas en el historial de la conversación. La estructura incluye la creación de un modelo personalizado, el manejo de un historial de mensajes y la interacción en tiempo real con el usuario, haciendo énfasis en ofrecer asistencia en el ámbito educativo.
- PruebaContent.py. Este script interactúa con un modelo de lenguaje de Inteligencia Artificial (en este caso, el modelo llama2.7b) a través de una API Web. Envía un mensaje desde el usuario a un servidor remoto utilizando una solicitud HTTP POST, y luego recibe y muestra la respuesta generada por el modelo.
- El propósito principal es permitir que un usuario envíe consultas en lenguaje natural (como "¿cómo estás?") y obtener respuestas generadas por el modelo de IA. La API en el servidor procesa el mensaje, lo pasa al modelo y devuelve una respuesta que se imprime en la consola.
- El script funciona como una vía directa, sin demasiadas complicaciones, para interactuar con un sistema de Inteligencia Artificial que genera respuestas a partir de texto. Es útil, sobre todo, para construir chatbots o cualquier otro sistema que dependa de un intercambio dinámico con el usuario.
- ragChatmodel.py, vale decir que incorpora algo más complejo. Combina un modelo conversacional con la capacidad de buscar información externa antes de responder. Es decir, no se limita a lo que ya "sabe", sino que intenta recuperar contenido útil desde una base de datos previamente cargada con documentos y sus correspondientes embeddings. Si encuentra algo relevante, lo incluye en la respuesta. Luego, el modelo ollama toma todo ese contexto —las preguntas anteriores, los documentos encontrados, todo eso— y genera una respuesta más ajustada. El sistema también mantiene un historial de la conversación, lo cual permite mayor coherencia en las respuestas. Y gracias a las incrustaciones, puede recuperar con rapidez los textos más adecuados, lo que al final mejora bastante tanto la precisión como la utilidad de lo que se responde. No es que siempre sea perfecto, claro, pero la combinación entre recuperación y generación ofrece un equilibrio interesante entre contexto y capacidad de respuesta.RestModel.py. este código interactúa con el

modelo de lenguaje generativo de Google llamado Gemini a través de su API REST. El chatbot envía mensajes de usuario a la API y recibe respuestas generadas, simulando una conversación entre el usuario y el sistema. El proceso se maneja mediante solicitudes HTTP, donde el texto del usuario se envía en formato JSON y la respuesta generada se muestra en la consola. Esto permite crear un sistema de conversación automatizado usando el modelo de lenguaje de Gemini.

6.5. CLASE INTERACTIVA

6.5.1. CONFIGURACIÓN DEL PROYECTO

Para mejorar la experiencia del aula en 3D, se incorporaron unos cuantos videos hechos con Filmora. No se buscaba algo complicado, solo material que ayudara a explicar mejor ciertos temas dentro del entorno virtual. Filmora funcionó bien para eso, sobre todo por sus efectos y lo fácil que es ajustar la narrativa. Luego, los videos se pasaron a Unity y se colocaron en las pantallas del aula usando VideoPlayer. Eso sí, se cuidó que el formato fuera compatible y que no se perdiera calidad, al menos no mucha. Se agregaron botones simples como reproducir y pausar, nada muy elaborado, pero suficiente para que el usuario pudiera explorar el contenido con algo más de control. La idea era sumar valor sin saturar la escena, y al final pareció funcionar.

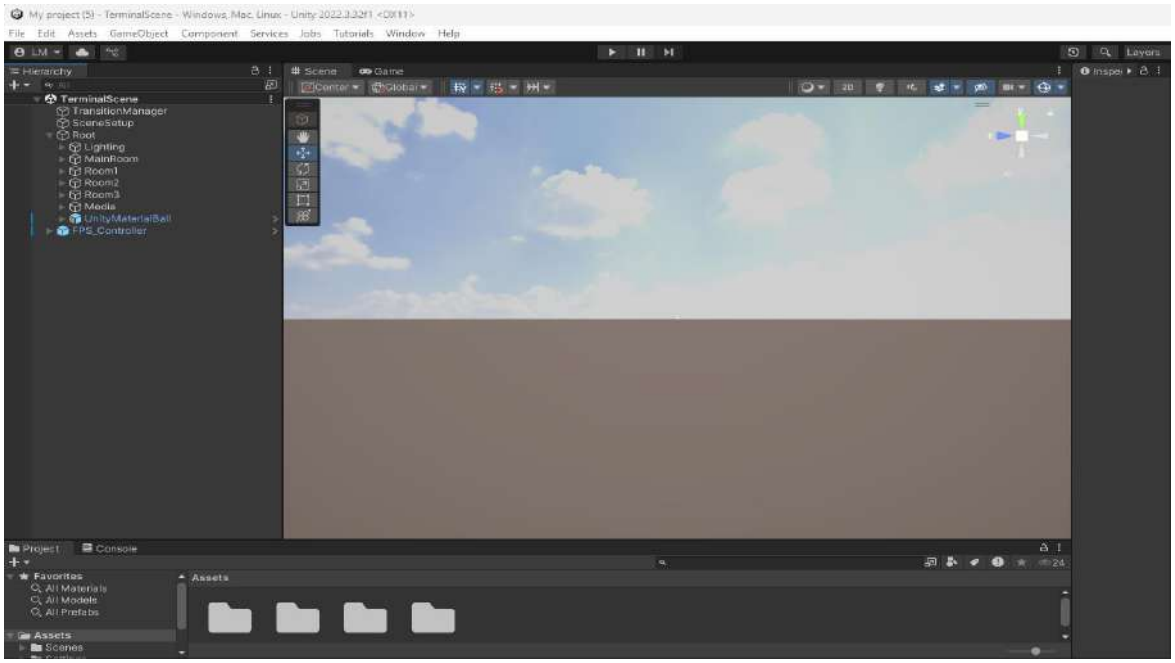

Fig. 27. Aula creada con objetos 3D.

6.5.2. CREACIÓN DE LA ESTRUCTURA DEL AULA

La construcción del aula se llevó a cabo a partir de GameObjects bastante simples, principalmente cubos. Estos se escalaron y colocaron con cierta precisión para definir las paredes, el techo y el suelo. No se trató de un diseño complejo, más bien fue una cuestión de ajustar bien las proporciones usando las herramientas básicas de Transform en Unity. Cada cubo se modificó para que, visualmente al menos, el espacio tuviera un aire creíble, parecido al de un aula convencional. Se añadieron aberturas en puntos específicos, digamos en zonas clave para simular ventanas y una puerta, también usando cubos, aunque algo más pequeños o redimensionados. Vale considerar que todo esto requirió algo de cuidado en la alineación; no es que fuera difícil, pero sí se usó la cuadrícula de Unity y algunas herramientas de ajuste para que todo quedara más o menos bien encajado. El resultado final, aunque sencillo, logra transmitir la idea de un espacio educativo sin demasiadas distracciones visuales.

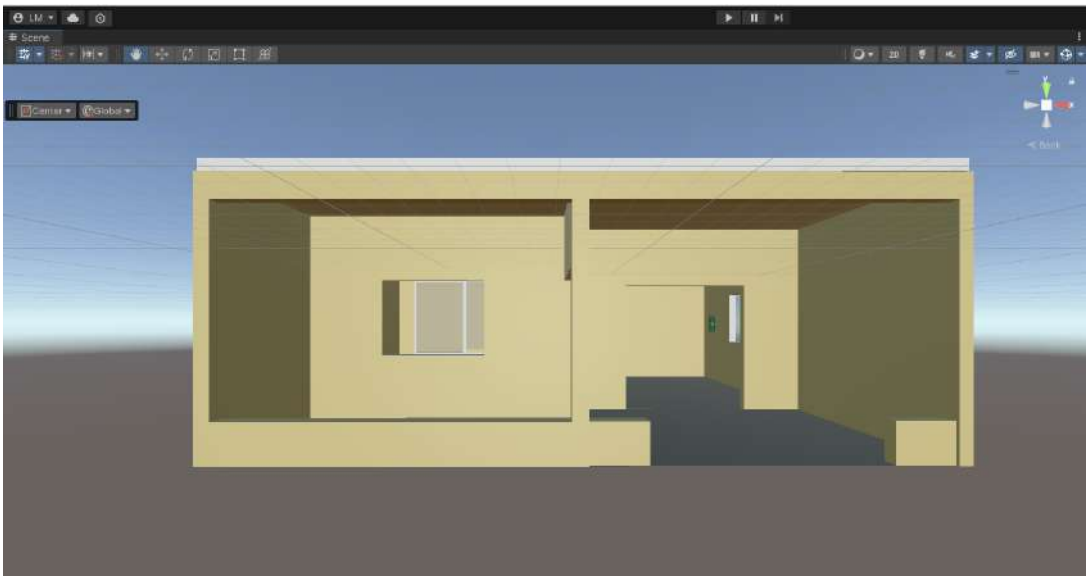

Fig. 28. Aula creada con objetos 3D.

6.5.3. Decoración del Aula

Después de completar la estructura principal, se pasó a decorar el aula con objetos tridimensionales que representaran el mobiliario y otros elementos propios de un entorno educativo. Se utilizaron modelos 3D importados desde bibliotecas como Vectary, incluyendo escritorios, sillas, estanterías, una pizarra y, en general, todo aquello que ayudara a dar forma a una escena creíble. Los objetos se posicionaron dentro del espacio usando las herramientas de Unity, en particular Translate y Rotate, ajustando cada pieza para que encajara con cierta coherencia funcional. No es que todo quedara perfecto al primer intento, pero poco a poco se fueron definiendo las ubicaciones hasta lograr un conjunto armonioso. También se agregaron algunos elementos decorativos, como carteles o gráficos visuales, que ayudaran a darle más carácter al aula. Todo se colocó con algo de cuidado, procurando que no solo se viera bien, sino que también tuviera sentido en cuanto a distribución y propósito, más o menos como se esperaría encontrar en un salón real.

Fig. 29. objetos 3D en escena.

6.5.4. APLICACIÓN DE MATERIALES Y TEXTURAS

En este paso, se aplicaron materiales y texturas personalizadas a los objetos del aula para darles un aspecto más realista. se crearon nuevos materiales en Unity y asignándoles texturas que se habían importado previamente. Se usaron texturas de superficies como madera para los escritorios y sillas, además de baldosas para el suelo del aula. También texturas suaves para las paredes, que simularon un acabado pintado, y texturas transparentes o semi-translúcidas para las ventanas. Utilizando ajustaste las propiedades del material, como la reflectividad y el nivel de brillo, para obtener un efecto realista bajo diferentes condiciones de iluminación. Además, se aseguró de que los materiales respondieran bien a las luces y sombras de la e3scena, lo que dio una mayor profundidad y realismo a los objetos.

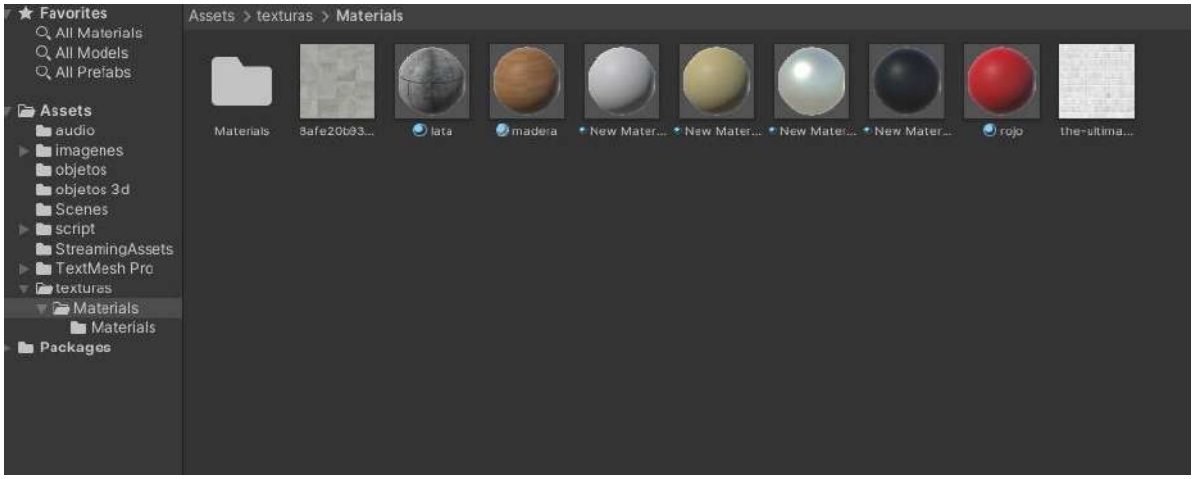

Fig. 30. Repertorio de materiales y texturas.

6.5.5. ILUMINACIÓN DEL AULA

Se colocó una luz principal para simular el sol entrando por las ventanas. También algunas lámparas en el techo. Se ajustaron cosas básicas, como la intensidad y el color, para que no se viera muy artificial. Las sombras ayudan bastante a que todo parezca más real. No está perfecto, pero funciona bien para el propósito.

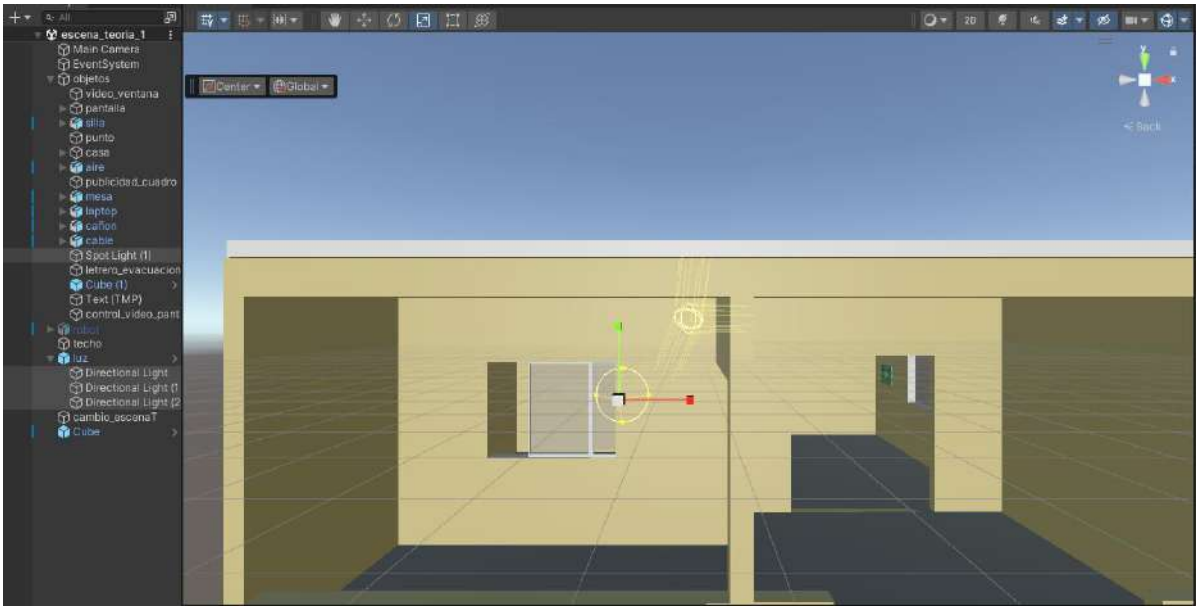

Fig. 31. Objetos light en escena.

6.5.6. OPTIMIZACIÓN DEL RENDIMIENTO

Luego de montar la escena y añadir todos los elementos visuales, se hizo un esfuerzo por mejorar el rendimiento general del aula. No es que estuviera mal al principio, pero valía la pena optimizarlo un poco más, sobre todo pensando en dispositivos con menos recursos. Por eso, algunos objetos que no cambian —como las paredes, los escritorios o las sillas— se marcaron como estáticos. Eso le permite a Unity hacer ciertos cálculos por adelantado y, así, ahorrar recursos durante la ejecución. También se tocaron detalles como los niveles de detalle en los modelos más pesados, de modo que, si están lejos, no cargan tantos polígonos. Además, se comprimieron varias texturas, lo justo para aligerar la memoria sin perder calidad visual. Todo eso ayudó a que la escena corriera más fluida sin comprometer demasiado la estética.

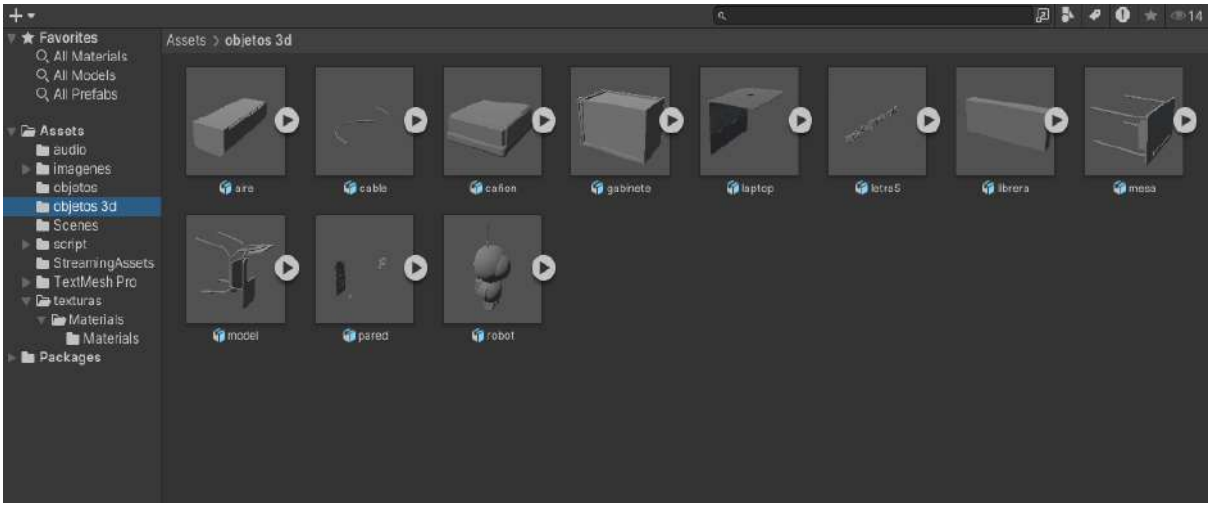

Fig. 32. Objetos 3D.

6.5.7. CONTENIDO DE LA MATERIA

En este proyecto, se llevó a cabo un proceso de entrenamiento de una Inteligencia Artificial (IA) basado en una serie de temas de la asignatura de lógica de programación. Para dicho entrenamiento, se utilizaron dos tipos principales de recursos. archivos de texto (TXT) extraídos de manuales técnicos y videos educativos diseñados para ser integrados en un entorno 3D creado con Unity. A continuación, se describe el flujo de trabajo técnico utilizado para ambos tipos de recursos.

Entrenamiento con Archivos TXT

Cada uno de los temas de la asignatura de desarrollo de lógica de programación fue estructurado en archivos de texto plano (TXT), que sirvieron como insumos para el entrenamiento de la IA. Los archivos fueron organizados en función de las siguientes guías.

- Guía 1. Introducción a la programación
- Guía 2. Algoritmos y diagramas de flujo
- Guía 3. Introducción a la programación con Python
- Guía 4-6. Operaciones básicas del lenguaje (Estructuras de programación)
- Guía 7. Estructuras de datos (Listas y Tablas)
- Guía 8. Estructuras de datos (Diccionarios y Tuplas)
- Guía 9. Manipulación de archivos
- Guía 10. Manejo de excepciones
- Guía 11. Funciones

Los archivos TXT fueron creados con la intención de ofrecer información estructurada y técnica que la IA pudiera procesar. Los textos contienen definiciones clave, algoritmos explicados paso a paso, ejemplos de código en Python, y ejercicios prácticos. Este formato permite que la IA no solo aprenda los conceptos teóricos, sino que también reconozca patrones en los ejemplos de código y la lógica detrás de los algoritmos presentados.

El flujo de trabajo técnico para la preparación de los archivos TXT incluyó.

Extracción de contenido relevante de manuales técnicos y material de estudio.

Limpieza y estructuración del contenido para adaptarlo al formato de entrenamiento de la IA.

Inclusión de ejemplos de código comentados para permitir que la IA identifique las conexiones entre la teoría y la práctica.

Uso de secciones claras con etiquetas que faciliten la indexación y el análisis por parte de la IA.

Entrenamiento de modelos de Ollama

En Ollama no se necesita entrenamiento ya que nos permite utilizar modelos ya preentrenados con temas de diversas fuentes.

Creación de Videos Educativos e Integración en Unity

Paralelamente a los archivos TXT, se produjo una serie de videos educativos que cubren los mismos temas descritos anteriormente. Los videos fueron diseñados con una metodología pedagógica que incluye.

- Narración clara y precisa que explica los conceptos técnicos en un lenguaje accesible.
- Visualizaciones de algoritmos y diagramas de flujo que muestran gráficamente el funcionamiento interno de los procesos.
- Demostraciones de código en Python, donde se ejecutan los ejemplos contenidos en los archivos TXT.
- Efectos visuales y transiciones fluidas para mantener el interés del espectador.

Estos videos fueron creados utilizando herramientas de edición como Filmora, y posteriormente integrados en un entorno 3D desarrollado en Unity. Unity se utilizó para crear un aula virtual en la que los videos se visualizan de manera inmersiva, permitiendo a los usuarios interactuar con los materiales mientras se reproducen. La integración de los videos se logró mediante el uso del componente VideoPlayer de Unity, lo que permitió el control preciso de la reproducción de los videos, con opciones como pausar, avanzar y retroceder.

El proceso técnico involucró

- Edición de videos con transiciones y efectos para mejorar la presentación visual de los conceptos.
- Exportación de los videos en formatos compatibles con Unity.
- Creación de una interfaz de usuario en Unity para controlar la reproducción de los videos.
- Sincronización de los videos con los objetos correspondientes en la escena, asegurando que los materiales visuales se alinearan con los temas tratados en los archivos TXT.

6.5.8. VIDEOS DE CLASE INTERACTIVA

Para dar un toque más inmersivo al aula 3D, se desarrollaron varios videos utilizando Filmora, una herramienta bastante práctica para la edición audiovisual. La idea era complementar el entorno con material visual que ampliara o aclarara ciertos contenidos, algo así como una ayuda extra dentro de la propia escena. Se optó por este editor porque, aunque sencillo, permite resultados

bastante pulidos. transiciones suaves, efectos visuales llamativos, narraciones que, sin complicarse mucho, logran ser claras. Los videos se exportaron en formatos compatibles con Unity, cuidando siempre que la calidad no se perdiera en el proceso, lo cual, vale decir, no siempre es tan simple como parece.

Ya con los videos listos, se integraron directamente en la escena 3D a través de Unity, asignándolos a objetos específicos del aula. Para ello, se utilizaron componentes VideoPlayer, que actuaron como reproductores en pantallas virtuales dispuestas en distintos puntos del entorno. Estas pantallas funcionaron como proyectores o monitores interactivos, donde se podía ver el contenido según se recorría la escena. Se configuraron controles básicos como play y pause, permitiendo que el usuario manejara la reproducción a su ritmo. Esta incorporación no solo sirvió para reforzar el aprendizaje, sino que también aportó dinamismo, haciendo que la experiencia resultara más atractiva y accesible. Sin duda, un pequeño ajuste técnico que marcó una diferencia significativa.

Fig. 33. Interfaz de Filmora.

6.5.9. RESULTADO FINAL

Al final de este proceso, obtuvimos un aula 3D completamente funcional y decorada en Unity. La estructura sólida, combinada con detalles realistas como los muebles, las texturas y la iluminación, proporcionaron un ambiente inmersivo y visualmente atractivo, adecuado para simulaciones educativas o recorridos interactivos. La optimización de los recursos y la configuración eficiente de la iluminación y los materiales aseguraron que el proyecto pudiera ejecutarse de manera fluida y sin problemas en una amplia gama de dispositivos, lo que lo hizo versátil y práctico para diversas aplicaciones.

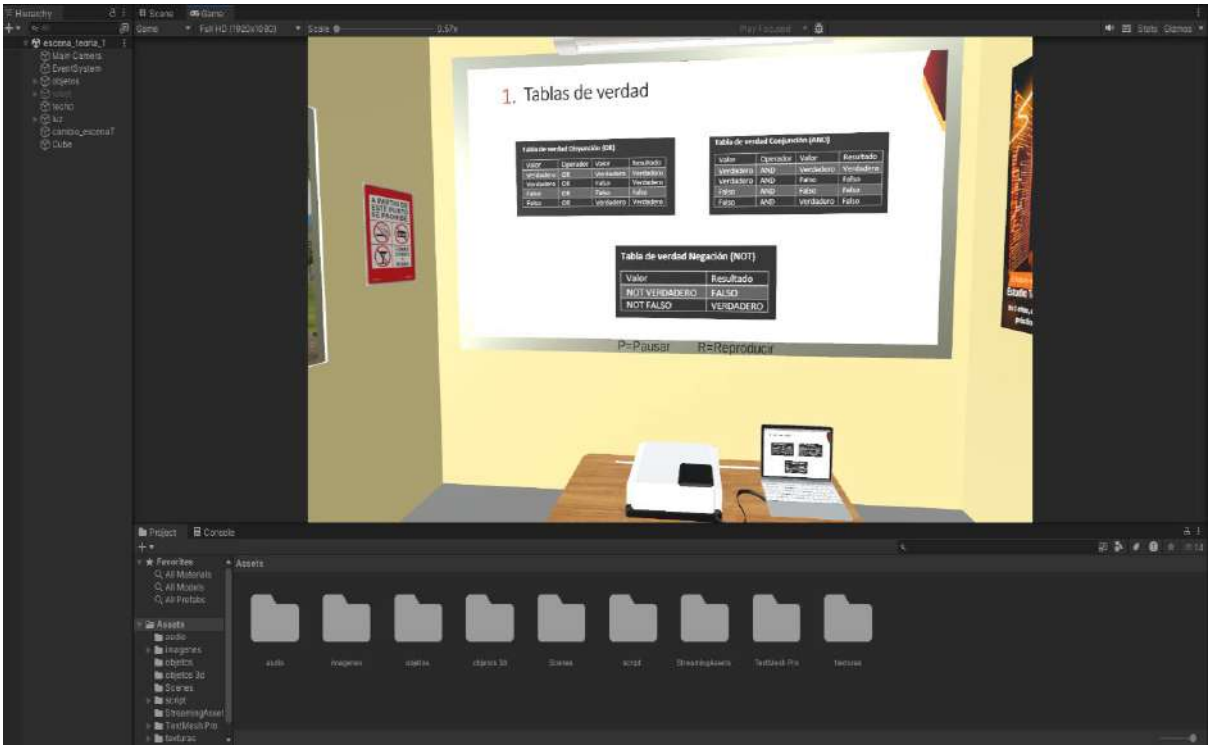

Fig. 34. Aula interactiva.

6.6. CONFIGURACIÓN LOCAL

Abrir las carpetas en visual studio code de frontend (edula-Web) y backend (edula- engine).

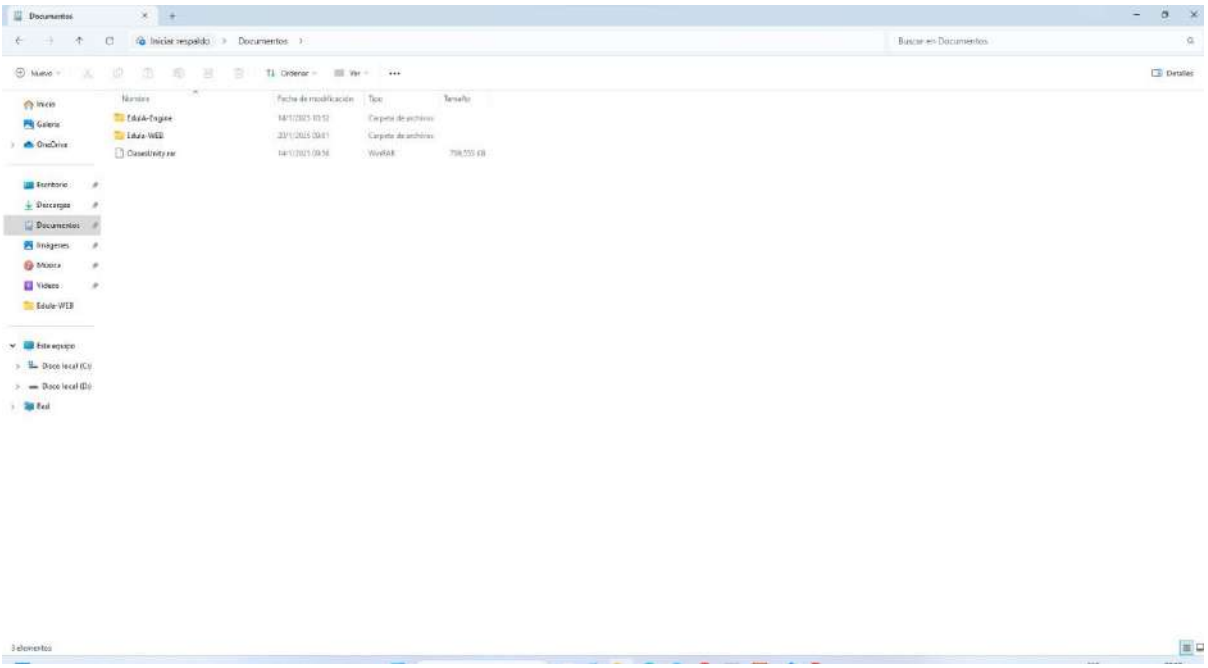
Fig. 35. Carpetas de proyecto.

En visual estudio con la carpeta del backend abierta, abrimos una nueva terminal y ponemos el comando. venv\Scripts\activate para activar entorno virtual y presionar enter.

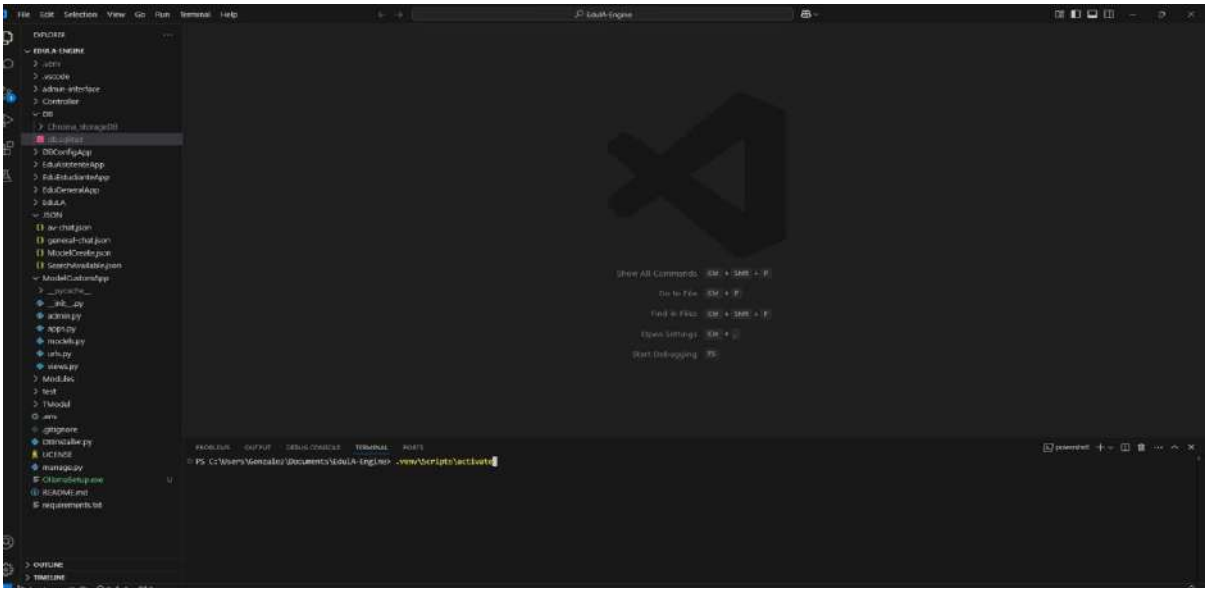

Fig. 36. Visual Code.

Poner el comando para correr el servidor python manage.py runserver

Fig. 37. Terminal de Visual Code.

Ahora que está corriendo el backend abrir el frontend y abrir una nueva terminal en el frontend y poner el comando npm run serve

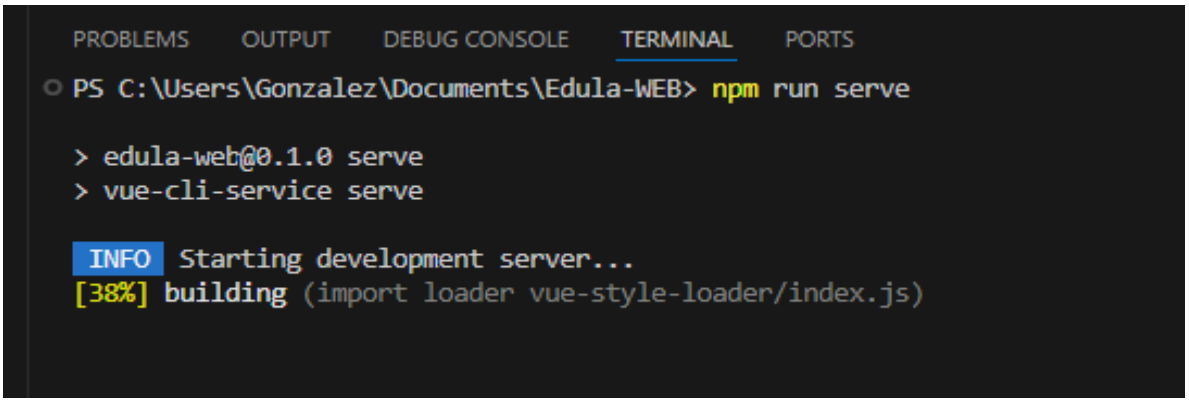

Fig. 38. Terminal de Visual Code.

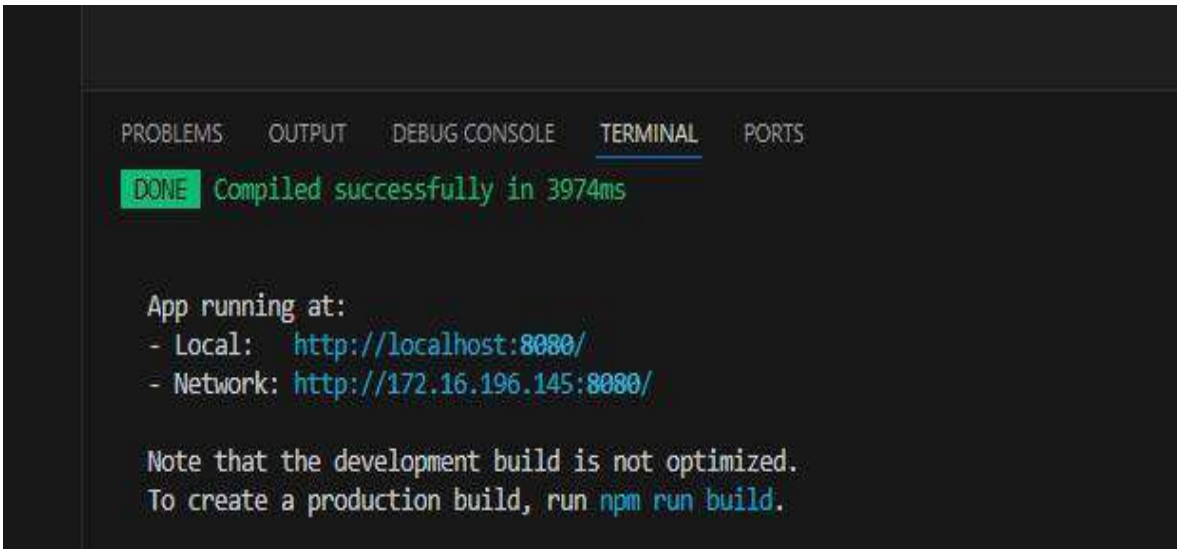

Fig. 39. Terminal de Visual Code.

Ahora que se tiene el server corriendo abrir una terminal y hacer correr la Inteligencia Artificial con el comando ollama serve

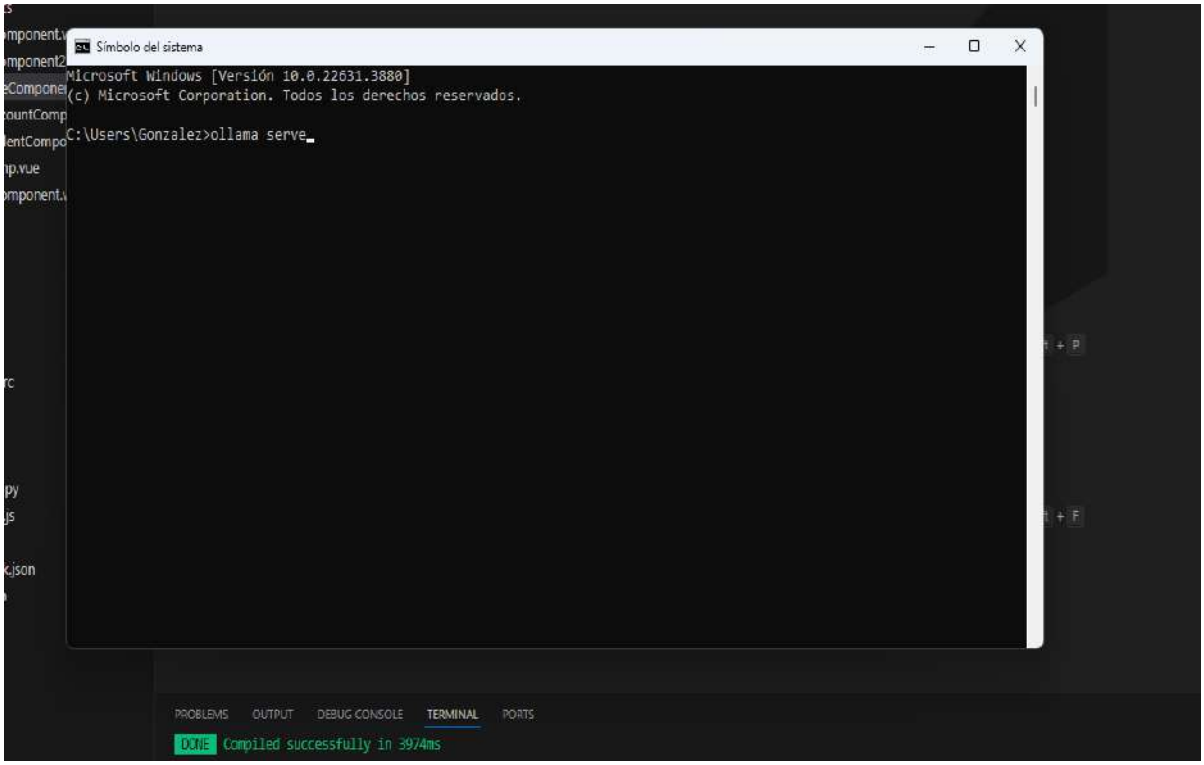

Fig. 40. Terminal de Visual Code.

Ahora acceder al servidor apache para correr las clases de unity.

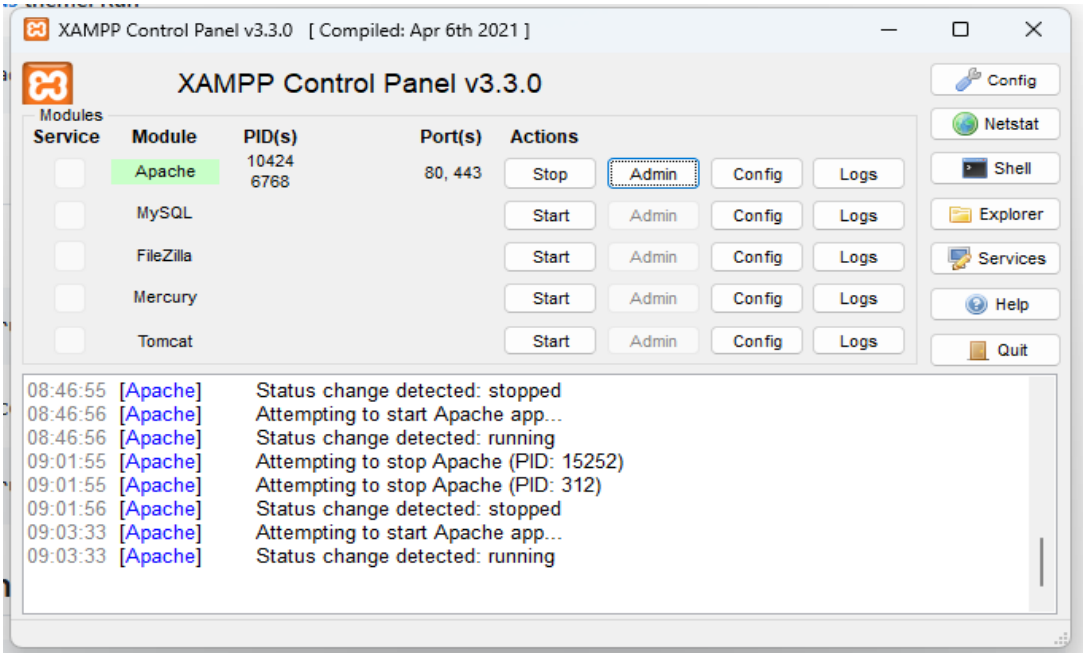

Fig. 41. Terminal de Visual Code.

Ponemos esta dirección para acceder al panel del administrador para poner información de las clases http.//127.0.0.1.8000/admin/login/?next=/admin/

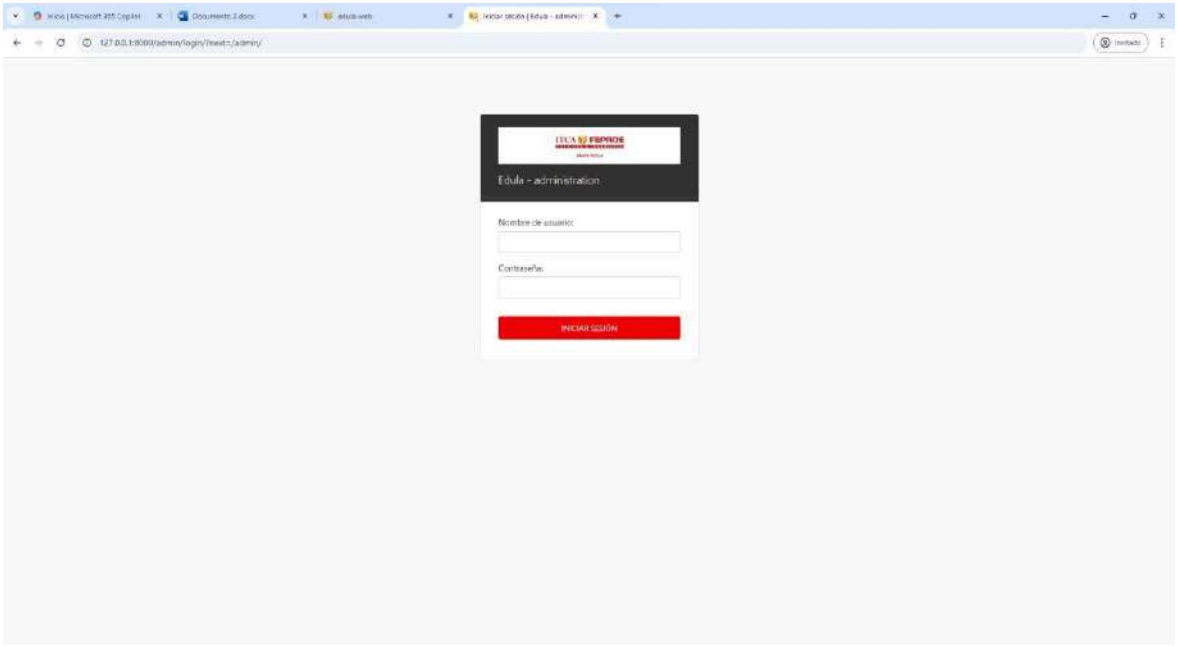

Fig. 42. Login de acceso.

6.7. IMPLEMENTACIÓN DE APRENDIZAJE PROFUNDO DEEP LEARNING

Durante la implementación del aprendizaje profundo, se seleccionaron y prepararon los conjuntos de datos pertinentes. Con la ayuda de *Ollama*, se diseñó un modelo específicamente adaptado a las necesidades del tutor virtual. *Ollama* facilitó la creación y ajuste del modelo, optimizando tanto el proceso de entrenamiento como la validación. Finalmente, el modelo fue integrado en el entorno del tutor para su óptimo funcionamiento en el contexto educativo del ITCA-FEPADE, asegurando un rendimiento eficiente y preciso gracias a la asistencia de *Ollama*.

Interfaz utilizada para la creación de documentos para la alimentación del modelo Ollama, tanto para el chat general como chat educativo.

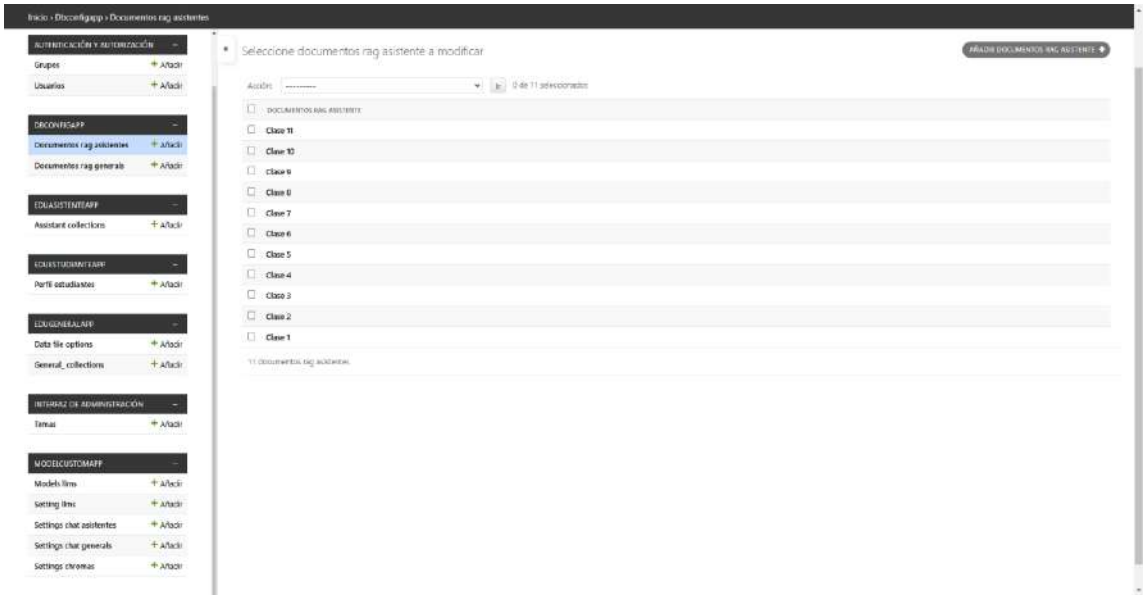

Fig. 43. Interfaz para la creación de documentos para modelo Ollama.

Swagger nos proporcionó una interfaz visual interactiva, que facilito la interacción y la comprensión de las API de la aplicación del tutor virtual.

7. PRUEBAS DEL TUTOR VIRTUAL

Las pruebas del tutor virtual se realizaron con el objetivo de evaluar su funcionamiento, eficacia y seguridad en el entorno educativo. Se llevaron a cabo pruebas exhaustivas que verificaron la adaptabilidad del contenido educativo.

- Pruebas de contenido adaptado de la asignatura.
- Pruebas de test funcionamiento de la clase interactiva.
- Pruebas de test de chat con IA.

la interacción con los usuarios (estudiantes y docentes), la eficiencia en la personalización del aprendizaje y la capacidad de ofrecer retroalimentación útil. También se validaron los aspectos técnicos y pedagógicos, garantizando la efectividad del tutor virtual en ITCA-FEPADE y su capacidad para mejorar la experiencia educativa.

8. RESULTADOS

- Módulo completo de asignatura de Desarrollo de lógica de programación. El cual cuenta con un chat donde el alumno puede realizar cualquier tipo de consulta, un selector de contenido que muestra la clase interactiva según el tema seleccionado, entre sus limitaciones
- Aplicación completamente interactiva ya que se puede ver la clase de una forma gráfica manipulando el ritmo de la sesión, así mismo poder realizar un ejercicio práctico del tema seleccionado, y la parte intuitiva debido a que es bastante fácil de utilizar.
- Capacidad para mantener la interacción del alumno con el chat, y que almacene las recomendaciones de consulta anteriores de cada usuario para tener coherencia en la información brindada.
- Módulo de login adaptado al proyecto con encriptación de contraseña de usuario para mayor seguridad, validación de usuario, pass y longitud de caracteres introducidos, y campos obligatorios.
- Beneficios de fortalecimiento de alumnos déficit en la asignatura Desarrollo de lógica de programación, complemento de plataformas que ya se utilizan en ITCA-FEPADE para el aprendizaje.
- Manual técnico de usuario e informe final del proyecto.

9. CONCLUSIONES

- Identificación clara de requerimientos. El análisis exhaustivo realizado en la primera etapa permitió identificar de manera precisa los requerimientos del tutor virtual, lo que facilitó su desarrollo y su adecuada implementación en el entorno educativo de ITCA-FEPADE.
- Diseño accesible y funcional. El diseño de las interfaces y el sistema administrativo se enfocó en la usabilidad y accesibilidad, lo que permitió que la herramienta sea fácil de utilizar por estudiantes y docentes, asegurando una experiencia educativa eficiente.
- Eficacia del modelo de Deep Learning. La implementación del aprendizaje profundo resultó en un modelo robusto de redes neuronales que fue capaz de adaptarse a las necesidades del tutor virtual. El entrenamiento y validación exitosa del modelo aseguraron su integración óptima en el contexto educativo.
- Los resultados alcanzados con el desarrollo del tutor virtual responden directamente a la problemática planteada al inicio del proyecto. la necesidad de fortalecer el aprendizaje en la asignatura de Desarrollo de lógica de programación, especialmente en aquellos estudiantes con dificultades de comprensión. La solución planteada se justificó desde la carencia de herramientas adaptativas e interactivas en las plataformas educativas utilizadas por ITCA-FEPADE. Con la implementación de este sistema, se logró brindar una herramienta innovadora que combina Inteligencia Artificial, contenido gráfico interactivo y seguimiento personalizado.
- Las pruebas realizadas con usuarios permitieron validar el correcto funcionamiento técnico del sistema (chat, selección de temas, login seguro, interacción fluida), así como su utilidad pedagógica. Los estudiantes valoraron positivamente la facilidad de uso, la claridad de los contenidos y la capacidad del sistema para mantener conversaciones coherentes y útiles. También se evidenció una mejora en la comprensión de conceptos clave y una mayor disposición a practicar ejercicios relacionados con los temas estudiados.
- El tutor virtual impacta positivamente el aprendizaje al ofrecer una experiencia personalizada, en la que cada estudiante puede avanzar a su ritmo, consultar dudas de manera autónoma y reforzar los contenidos mediante simulaciones prácticas. La integración del chat con IA y el contenido gráfico en Unity permite una mayor retención del conocimiento y motiva la participación activa del estudiante.
- Asimismo, se destaca la importancia de las medidas de seguridad implementadas en el módulo de acceso, como la encriptación de contraseñas, la validación de campos y la protección de datos

personales. Estas medidas garantizan la privacidad del usuario y la integridad del sistema, aspectos fundamentales para el uso responsable de herramientas digitales en entornos educativos.

- Una de las principales fortalezas del tutor es su capacidad de adaptación e individualización. El sistema registra consultas anteriores por usuario, lo que permite mantener coherencia en las recomendaciones y construir una experiencia de acompañamiento más cercana a la de un tutor humano.
- No obstante, el tutor presenta algunas limitaciones, como la dependencia de una conexión estable para cargar los recursos interactivos en WebGL, y la necesidad de ajustar constantemente el modelo de lenguaje a nuevos contenidos curriculares. Aun así, su implementación representa un avance significativo en el uso de tecnologías emergentes al servicio de la educación, ofreciendo una

10. RECOMENDACIONES

- Monitoreo y actualización continua. Se recomienda realizar un monitoreo constante del rendimiento del tutor virtual y actualizar los modelos de aprendizaje profundo periódicamente, con el fin de adaptarlo a nuevos retos y necesidades del entorno educativo.
- Expansión del sistema. Se sugiere considerar la expansión del tutor virtual a otros contextos o áreas académicas dentro de ITCA-FEPADE, adaptando las funcionalidades para diferentes disciplinas y niveles educativos, con el objetivo de mejorar aún más el impacto de la herramienta en la formación académica.
- Se recomienda implementar el tutor virtual de manera progresiva en las cinco sedes de ITCA-FEPADE, garantizando acceso equitativo a esta herramienta por parte de todos los estudiantes.
- Se sugiere identificar y seleccionar asignaturas de distintas carreras donde el tutor virtual pueda ser de utilidad, especialmente aquellas que presentan altos índices de dificultad o requerimientos de apoyo personalizado.
- Se propone integrar el tutor virtual con las plataformas Moodle y Microsoft Teams, con el propósito de complementar los entornos de aprendizaje actuales en lugar de sustituirlos, generando una experiencia educativa más dinámica y personalizada.

11. GLOSARIO

API
Interfaz de Programación de Aplicaciones.

Backend
Parte de una aplicación o sitio Web que no es visible para el usuario.

Chatbot
Programa para sostener una conversación de algún tema en específico.

Deep Learning
Forma parte del campo del aprendizaje automático y se basa, más o menos, en el uso de grandes volúmenes de datos que, tras pasar por múltiples capas de procesamiento con distintos algoritmos, permiten que el sistema empiece a identificar patrones por sí mismo. No es que piense como una persona, pero logra realizar tareas que antes requerían intervención humana. Vale considerar que este tipo de tecnología aprende con la práctica, ajustándose con cada nuevo dato, lo que lo hace útil para problemas complejos.

Frontend
Es la interfaz con la que los usuarios interactúan directamente.

12. REFERENCIAS BIBLIOGRÁFICAS

[2] Python Software Foundation. (2025, 23 de junio). El tutorial de Python (versión 3.13). En Documentación oficial de Python. Recuperado el 27 de febrero de 2025, de https.//docs.python.org/es/3.13/tutorial/index.html

[3] Evan You & comunidad de Vue.js. (s. f.). Guía de Vue.js (versión 2). En Vue.js — Guía. Recuperado el 27 de febrero de 2025, de https.//es.vuejs.org/v2/guide/

[4] Django Software Foundation. (s. f.). Django. El framework Web para perfeccionistas con plazos. Recuperado el 27 de febrero de 2025, de https.//www.djangoproject.com/

[5] Django Software Foundation. (s. f.). Django Channels — Documentación oficial. Recuperado el 27 de febrero de 2025, de https.//channels.readthedocs.io/en/latest/

[6] Google Cloud. (s. f.). ¿Qué es la Inteligencia Artificial o IA? Recuperado el 29 de junio de 2025, de https.//cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419

[7] Boada, D. (2025, 17 de marzo). ¿Qué es Ollama? Hostinger. Recuperado el 27 de febrero de 2025, de https.//www.hostinger.com/es/tutoriales/que-es-ollama

[8] IBM. (s. f.). ¿Qué es el procesamiento de lenguaje natural (PLN)? Recuperado el 27 de febrero de 2025, de https.//www.ibm.com/mx-es/think/topics/natural-language-processing

[9] IONOS. (8 de julio de 2020). ¿Qué es WebSocket? Un canal de comunicación para la Web en tiempo real. Recuperado el 27 de febrero de 2025, de https.//www.ionos.com/es-us/digitalguide/paginas-Web/desarrollo-Web/que-es-Websocket/

[10] Educative, Inc. (s. f.). ¿Qué es AJAX polling? Recuperado el 27 de febrero de 2025, de https.//www.educative.io/answers/what-is-ajax-polling

ITCA FEPADE

SEDE CENTRAL Y CENTROS REGIONALES EL SALVADOR

La Escuela Especializada en Ingeniería ITCA-FEPADE, fundada en 1969, es una institución estatal con administración privada, conformada actualmente por 5 campus: Sede Central Santa Tecla y cuatro centros regionales ubicados en Santa Ana, San Miguel, Zacatecoluca y La Unión.

1. SEDE CENTRAL SANTA TECLA

Km. 11.5 carretera a Santa Tecla, La libertad.
Tel.: (503) 2132-7400

2. CENTRO REGIONAL SANTA ANA

Final 10a. Av. Sur, Finca Procavia.
Tel.: (503) 2440-4348

3. CENTRO REGIONAL ZACATECOLUCA

Km. 64.5, desvío Hacienda El Nilo sobre autopista a Zacatecoluca.
Tel.: (503) 2334-0763 y 2334-0768

4. CENTRO REGIONAL SAN MIGUEL

Km. 140 carretera a Santa Rosa de Lima.
Tel.: (503) 2669-2298

5. CENTRO REGIONAL LA UNIÓN

Calle Sta. María, Col. Belén, atrás del Instituto Nacional de La Unión
Tel.: (503) 2668-4700

www.itca.edu.sv