

Documento de Propuesta de Diseño de Software I, II y III.

NeuroSteam - Asistente Inteligente – Fase III

Autores

Dana Paola Alegría Durango (dalegriadurango22@correo.unicordoba.edu.co)

María Inés Bedoya Ortega (mbedoyaortega16@correo.unicordoba.edu.co)

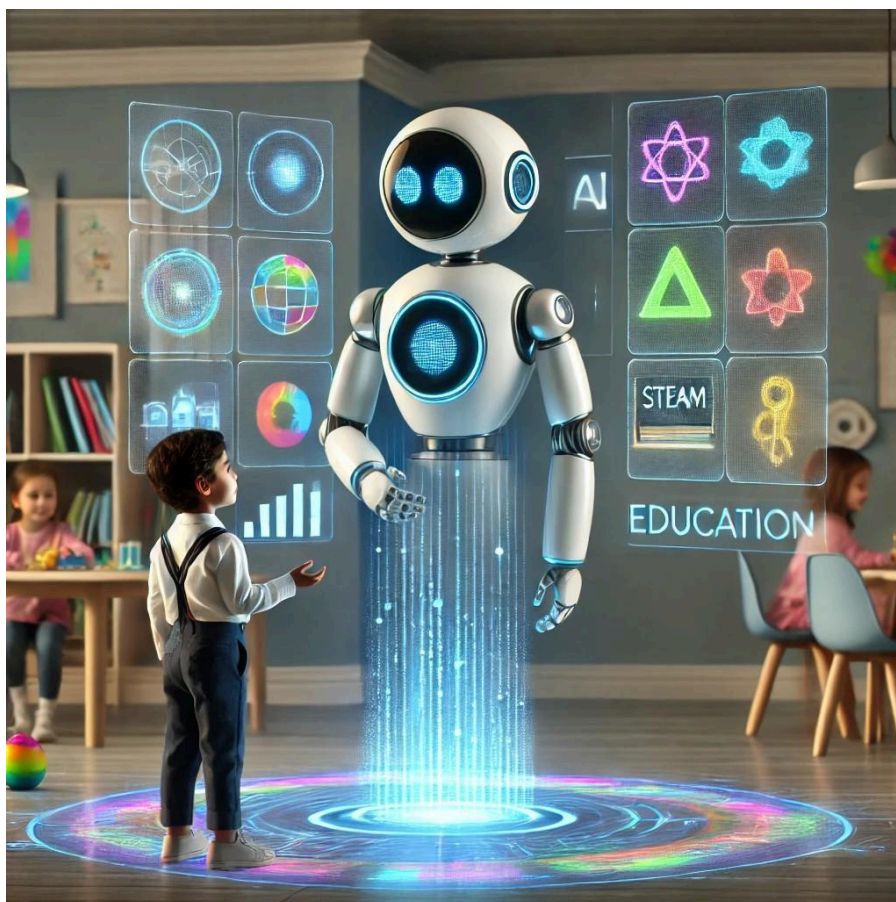
Johan Mercado Fernández (jmercadofernandez82@correo.unicordoba.edu.co)

María José Padilla Urueta (mpadillaurueta68@correo.unicordoba.edu.co)

Gloria Elena Cordero Almario (gcorderoalmario69@correo.unicordoba.edu.co)

Tutor

RAÚL EMIRO TOSCANO MIRANDA (rtoscano@correo.unicordoba.edu.co)



Breve reseña

En el marco del desarrollo de soluciones educativas innovadoras, este proyecto presenta el diseño e implementación de un asistente inteligente basado en inteligencia artificial generativa, orientado a la generación de actividades personalizadas bajo el enfoque STEAM. Su propósito es adaptar dinámicamente las experiencias de aprendizaje según las necesidades de desarrollo neurocognitivo de los estudiantes de grado primero, promoviendo una enseñanza más efectiva y alineada con sus habilidades y ritmos de aprendizaje.

ETAPA 1 DISEÑO DE LA APLICACIÓN Y ANÁLISIS DE REQUISITOS	5
INTRODUCCIÓN	5
PROPÓSITO DEL DOCUMENTO	5
ALCANCE DEL PROYECTO	6
DEFINICIONES Y ACRÓNIMOS	6
DESCRIPCIÓN GENERAL	6
OBJETIVOS DEL SISTEMA	7
FUNCIONALIDAD GENERAL	8
USUARIOS DEL SISTEMA	8
RESTRICCIONES	8
REQUISITOS FUNCIONALES	8
CASOS DE USO	8
DESCRIPCIÓN DETALLADA DE CADA CASO DE USO	8
DIAGRAMAS DE FLUJO DE CASOS DE USO	8
PRIORIDAD DE REQUISITOS	8
REQUISITOS NO FUNCIONALES	8
REQUISITOS DE DESEMPEÑO	8
REQUISITOS DE SEGURIDAD	9
REQUISITOS DE USABILIDAD	9
REQUISITOS DE ESCALABILIDAD	9
MODELADO E/R	9
DIAGRAMA DE ENTIDAD-RELACIÓN	9
DESCRIPCIÓN DE ENTIDADES Y RELACIONES	9
REGLAS DE INTEGRIDAD	9
ANEXOS (SI ES NECESARIO)	9
DIAGRAMAS ADICIONALES	9
REFERENCIAS	9
ETAPA 2: PERSISTENCIA DE DATOS CON BACKEND	10
INTRODUCCIÓN	10
PROPÓSITO DE LA ETAPA	10
ALCANCE DE LA ETAPA	10

DEFINICIONES Y ACRÓNIMOS	10
DISEÑO DE LA ARQUITECTURA DE BACKEND	10
DESCRIPCIÓN DE LA ARQUITECTURA PROPUESTA	10
COMPONENTES DEL BACKEND	10
DIAGRAMAS DE ARQUITECTURA	10
ELECCIÓN DE LA BASE DE DATOS	10
EVALUACIÓN DE OPCIONES (SQL o NoSQL)	10
JUSTIFICACIÓN DE LA ELECCIÓN	10
DISEÑO DE ESQUEMA DE BASE DE DATOS	11
IMPLEMENTACIÓN DEL BACKEND	11
ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN	11
CREACIÓN DE LA LÓGICA DE NEGOCIO	11
DESARROLLO DE ENDPOINTS Y APIs	11
AUTENTICACIÓN Y AUTORIZACIÓN	11
CONEXIÓN A LA BASE DE DATOS	11
CONFIGURACIÓN DE LA CONEXIÓN	11
DESARROLLO DE OPERACIONES CRUD	11
MANEJO DE TRANSACCIONES	11
PRUEBAS DEL BACKEND	11
DISEÑO DE CASOS DE PRUEBA	11
EJECUCIÓN DE PRUEBAS UNITARIAS Y DE INTEGRACIÓN	12
MANEJO DE ERRORES Y EXCEPCIONES	12
ETAPA 3: CONSUMO DE DATOS Y DESARROLLO FRONTEND	13
INTRODUCCIÓN	13
PROPÓSITO DE LA ETAPA	13
ALCANCE DE LA ETAPA	13
DEFINICIONES Y ACRÓNIMOS	13
CREACIÓN DE LA INTERFAZ DE USUARIO (UI)	13
DISEÑO DE LA INTERFAZ DE USUARIO (UI) CON HTML Y CSS	13
CONSIDERACIONES DE USABILIDAD	13
MAQUETACIÓN RESPONSIVA	13

PROGRAMACIÓN FRONTEND CON JAVASCRIPT (JS)	13
DESARROLLO DE LA LÓGICA DEL FRONTEND	13
MANEJO DE EVENTOS Y COMPORTAMIENTOS DINÁMICOS	13
USO DE BIBLIOTECAS Y FRAMEWORKS (SI APLICABLE)	14
CONSUMO DE DATOS DESDE EL BACKEND	14
CONFIGURACIÓN DE CONEXIONES AL BACKEND	14
OBTENCIÓN Y PRESENTACIÓN DE DATOS	14
ACTUALIZACIÓN EN TIEMPO REAL (SI APLICABLE)	14
INTERACCIÓN USUARIO-INTERFAZ	14
MANEJO DE FORMULARIOS Y VALIDACIÓN DE DATOS	14
IMPLEMENTACIÓN DE FUNCIONALIDADES INTERACTIVAS	14
MEJORAS EN LA EXPERIENCIA DEL USUARIO	14
PRUEBAS Y DEPURACIÓN DEL FRONTEND	14
DISEÑO DE CASOS DE PRUEBA DE FRONTEND	14
PRUEBAS DE USABILIDAD	14
DEPURACIÓN DE ERRORES Y OPTIMIZACIÓN DEL CÓDIGO	15
IMPLEMENTACIÓN DE LA LÓGICA DE NEGOCIO EN EL FRONTEND	15
MIGRACIÓN DE LA LÓGICA DE NEGOCIO DESDE EL BACKEND (SI NECESARIO)	15
VALIDACIÓN DE DATOS Y REGLAS DE NEGOCIO EN EL FRONTEND	15
INTEGRACIÓN CON EL BACKEND	15
VERIFICACIÓN DE LA COMUNICACIÓN EFECTIVA CON EL BACKEND	15
PRUEBAS DE INTEGRACIÓN FRONTEND-BACKEND	15

Etapas 1 Diseño de la Aplicación y Análisis de Requisitos

Introducción

En el constante avance de la tecnología educativa, la creación de herramientas innovadoras para la gestión de actividades se convierte en una prioridad para mejorar la experiencia de aprendizaje. En este contexto, surge la propuesta de desarrollar un Asistente Inteligente para la generación de actividades STEAM personalizadas, empleando tecnologías de vanguardia basadas en inteligencia artificial. Este proyecto representa un hito significativo en la evolución de la enseñanza digital, al integrar dos pilares fundamentales: la adaptabilidad del agente inteligente, que permite la creación de actividades dinámicas y ajustadas al desarrollo neurocognitivo de cada estudiante; y su capacidad de aprendizaje automático, que optimiza la personalización del contenido según el progreso individual del alumno.

Este documento tiene como objetivo presentar la concepción y desarrollo de este Asistente Inteligente, destacando su capacidad para revolucionar la forma en que se diseñan, gestionan y evalúan las actividades educativas en entornos digitales. Al ofrecer una solución integral basada en tecnologías punteras, se aspira a potenciar la interactividad, personalización y seguimiento del aprendizaje, brindando a educadores y estudiantes una herramienta poderosa para alcanzar sus objetivos educativos con mayor eficiencia y efectividad.

Propósito del Documento

El objetivo de este componente es desarrollar un Asistente Inteligente dentro de la plataforma educativa, capaz de generar, personalizar y asignar actividades STEAM adaptadas a las necesidades de los estudiantes de primera infancia. A través de inteligencia artificial, el asistente analizará el desarrollo neurocognitivo de cada estudiante para ofrecer actividades adecuadas, optimizando su aprendizaje. Esto permitirá a los profesores gestionar de forma eficiente el proceso educativo, facilitando la personalización del contenido y alineando las actividades con los objetivos pedagógicos establecidos en la plataforma.

- Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Esta etapa cumple la tarea de recoger todas las competencias desarrolladas en todas las áreas de formación del currículo de la licenciatura en Informática y Medios Audiovisuales y ponerlas a prueba en el diseño y análisis de un producto educativo que se base en las teorías de aprendizaje estudiadas, articule las estrategias de enseñanza con uso de TIC y genere innovaciones en educación con productos interactivos que revelen una verdadera naturaleza educativa. Estos productos deben aprovechar las fortalezas adquiridas en las áreas de tecnología e informática, técnicas y herramientas, medios audiovisuales y programación y sistemas, para generar productos software interactivos que permitan a los usuarios disfrutar de lo que aprenden, a su propio ritmo. Todo esto en el marco de un proceso metodológico (metodologías de desarrollo de software como MODESEC, SEMLI, etc.) que aproveche lo aprendido en la línea de gestión y lo enriquezca con elementos de la Ingeniería de Software.

- Etapa 2: Persistencia de Datos con Backend – Servidor

En la etapa 2 se continúa con los lineamientos de la etapa 1, para seguir adicionando elementos de diseño e implementación de software, enfocados en el desarrollo de APIs, servidores o microservicios que permitan soportar aplicaciones cliente del software educativo; en este sentido, el curso presenta los conceptos de los sistemas de bases de datos, su diseño lógico, la organización de los sistemas manejadores de bases de datos, los lenguaje de definición de datos y el lenguaje de manipulación de datos SQL y NoSQL; de tal manera que los estudiantes adquieran las competencias para analizar, diseñar y desarrollar aplicaciones para gestionar y almacenar grandes cantidades de datos, mediante el

uso de técnicas adecuadas como el diseño y modelo lógico y físico de base datos, manejo de los sistemas de gestión de bases de datos, álgebra relacional, dominio del lenguaje SQL como herramienta de consulta, tecnología cliente / servidor; igualmente, se definirán los elementos necesarios para el acceso a dichas bases de datos, como la creación del servidor API, utilizando tecnologías de vanguardia como node.js, express, Nest.js, Spring entre otros; para, finalmente converger en el despliegue de la API utilizando servicios de hospedaje en la nube, preferiblemente gratuitos. También podrá implementar servidores o API's con inteligencia artificial o en su defecto crear una nueva capa que consuma y transforme los datos obtenidos de la IA. El desarrollo del curso se trabajará por proyectos de trabajo colaborativo que serán evaluados de múltiples maneras, teniendo en cuenta más el proceso que el resultado.

- Etapa 3: Consumo de Datos y Desarrollo Frontend – Cliente

La etapa 3 el estudiante está en capacidad de establecer la mejor elección de herramientas de consumo de datos y técnicas en aras de lograr el mejor producto a nivel de software o hardware acorde a los requerimientos funcionales y no funcionales del problema a solucionar. En este punto el estudiante puede consumir los datos a través de un cliente que puede ser una aplicación de celular, una aplicación de escritorio, una página web, IoT (internet de las cosas) o incluso, artefactos tecnológicos. El diseño gráfico es de los requisitos esenciales en la capa de presentación, por lo tanto, se requieren los cursos de diseño gráfico vistos previamente. Los elementos anteriores nos permiten elegir el paradigma y tecnología para desarrollar nuestras aplicaciones, teniendo en cuenta que podríamos desarrollar aplicaciones de tipo cliente.

Alcance del Proyecto

El proyecto busca desarrollar un Asistente Inteligente para la generación de actividades STEAM personalizadas, diseñado para apoyar a docentes en la creación, edición y gestión de recursos educativos adaptados a los hitos neurocognitivos de los estudiantes. Este asistente permitirá optimizar el diseño de experiencias de aprendizaje dinámicas, promoviendo un enfoque flexible y accesible. Abarca la generación automática de actividades, la recomendación de materiales, la edición y almacenamiento de recursos previos, la exportación en múltiples formatos, y la creación de un entorno colaborativo para docentes. Además, su interfaz intuitiva facilita su uso a educadores con o sin experiencia en herramientas tecnológicas.

Funcionalidades del Agente Inteligente:

- Iniciar sesión (Login)
- Instrucciones de creación de actividad
- Generar actividad
- Guardar actividad
- Modificar actividad
- Descargar actividad
- Editar prompt
- Historial de creación de actividades

Funcionalidades Futuras:

- Registro de actividades descargadas por usuarios.
- Visualización del estado de finalización de las actividades.
- Formatos y configuraciones variadas.

Definiciones y Acrónimos

API: Interfaz de Programación de Aplicaciones (Application Programming Interface).

DBMS: Sistema de Gestión de Bases de Datos (Database Management System).

SQL: Lenguaje de Consulta Estructurada (Structured Query Language).

HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).

REST: Transferencia de Estado Representacional (Representational State Transfer).

JSON: Notación de Objetos de JavaScript (JavaScript Object Notation).

JWT: Token de Web JSON (JSON Web Token).

CRUD: Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete).

ORM: Mapeo Objeto-Relacional (Object-Relational Mapping)

MVC: Modelo-Vista-Controlador (Model-View-Controller). API RESTful:

API que sigue los principios de REST.

CI/CD: Integración Continua / Entrega Continua (Continuous Integration / Continuous Delivery).

SaaS: Software como Servicio (Software as a Service).

SSL/TLS: Capa de sockets seguros/Seguridad de la Capa de Transporte (Secure Sockets Layer/Transport Layer Security).

HTML: Lenguaje de Marcado de Hipertexto (Hypertext Markup Language).

CSS: Hojas de Estilo en Cascada (Cascading Style Sheets).

JS: JavaScript.

DOM: Modelo de Objeto del Documento (Document Object Model).

UI: Interfaz de Usuario (User Interface).

UX: Experiencia del Usuario (User Experience).

SPA: Aplicación de Página Única (Single Page Application).

AJAX: Asíncrono JavaScript y XML (Asynchronous JavaScript and XML).

CMS: Sistema de Gestión de Contenido (Content Management System).

CDN: Red de Distribución de Contenido (Content Delivery Network). SEO:

Optimización de Motores de Búsqueda (Search Engine Optimization).

IDE: Entorno de Desarrollo Integrado (Integrated Development Environment).

CLI: Interfaz de Línea de Comandos (Command Line Interface).

PWA: Aplicación Web Progresiva (Progressive Web App).

Asistente inteligente: IA que genera respuestas y contenido de forma autónoma.

Prompt: Instrucción que guía la respuesta de un modelo de IA.

Página web: Plataforma digital accesible desde un navegador.

DEFINICIONES

- Asistente inteligente: Aplicación basada en inteligencia artificial (IA) que responde y genera contenido de manera autónoma, en este caso, utilizando el modelo de lenguaje Chat GPT-4 para diseñar actividades STEAM según criterios neurocognitivos.
- Entrenamiento mediante indicaciones (prompts): Proceso de ajuste del modelo de IA mediante la creación de instrucciones (prompts) optimizadas que guían su comportamiento y respuestas dentro de un contexto específico.
- Usuarios (Administradores y Docentes):
 - Administradores (Desarrolladores): Personas responsables de la configuración técnica, mantenimiento y mejora del asistente inteligente, incluyendo ajustes en los avisos y optimización del modelo.
 - Docentes: Usuarios finales que interactúan con la plataforma para generar actividades STEAM personalizadas a partir del asistente.
- API (Interfaz de Programación de Aplicaciones): Conjunto de reglas y herramientas que permiten integrar el asistente inteligente con otras aplicaciones, facilitando su acceso desde una página web en lugar de depender de la interfaz original de Chat GPT.
- Página web: Interfaz de usuario personalizada que actúa como el entorno principal de interacción, brindando acceso a las funcionalidades del asistente y permitiendo una experiencia optimizada para los docentes.
- Optimización de la experiencia del usuario (UX/UI): Diseño de la interfaz y las interacciones de la plataforma para hacerla accesible, intuitiva y eficiente para los docentes que utilizarán el asistente.
- IA generativa: Es una inteligencia artificial que crea contenido original, como texto, imágenes o música, basándose en patrones aprendidos de datos previos.
- Chatbot: Es un programa de inteligencia artificial diseñado para simular conversaciones con usuarios, respondiendo preguntas o realizando tareas de manera automática.
- Login: Es el proceso de acceder a un sistema o plataforma ingresando un nombre de usuario y una contraseña para autenticarse.
- Base de datos: Sistema de almacenamiento estructurado donde se guardan y gestionan los datos del asistente inteligente, como registros de usuarios, actividades generadas y configuraciones.
- Machine Learning (Aprendizaje automático): Rama de la inteligencia artificial que permite a los sistemas mejorar su rendimiento a partir de la experiencia y los datos sin ser explícitamente programados para cada tarea.
- Modelo de lenguaje: Algoritmo de inteligencia artificial entrenado en grandes volúmenes de datos para generar y comprender texto de manera coherente y contextual.
- Módulo: Sección o componente específico del software que cumple una función particular dentro de la plataforma, como el módulo de autenticación o el módulo de generación de actividades.
- Feedback: Opinión o retroalimentación proporcionada por los usuarios sobre su experiencia con el sistema, utilizada para mejorar su funcionamiento y usabilidad.

Descripción General

Objetivos del Sistema

El objetivo general de este componente es el desarrollo de un asistente inteligente que facilite la creación y adaptación de actividades STEAM personalizadas, optimizando la gestión del aprendizaje por parte de los docentes. Este asistente, basado en inteligencia artificial, generará actividades alineadas con los hitos del desarrollo neurocognitivo de los estudiantes de primer grado, permitiendo a los profesores acceder a recursos educativos ajustados a las necesidades de su grupo.

El sistema busca mejorar la eficiencia en la planificación docente, aumentar la flexibilidad del aprendizaje, personalizar la enseñanza según el desarrollo de los estudiantes y optimizar la interacción con los contenidos educativos. Además, proporcionará herramientas para evaluar el impacto de las actividades, permitiendo a los docentes realizar ajustes en tiempo real y mejorar continuamente la experiencia de aprendizaje.

Funcionalidad	Administrador	Docente
Iniciar sesión (Login)	✓	✓
Especificar parámetros de la actividad		✓
Generar actividad		✓
Guardar actividad		✓
Modificar actividad		✓
Descargar actividad		✓
Editar prompt	✓	
Generar registro de actividades	✓	✓

Continúa de las entidades

Funcionalidad General

El asistente inteligente NeuroSteam permite a los docentes generar actividades educativas STEAM personalizadas, alineadas con el desarrollo neurocognitivo de los estudiantes de primer grado. Para ello, el asistente inteligente ofrece una serie de funcionalidades clave:

Iniciar sesión: Permite a los docentes acceder al sistema de manera segura mediante credenciales personales. Esta funcionalidad garantiza la protección de la información y el acceso a un entorno personalizado.

Instrucciones de creación de actividad: Proporciona una guía paso a paso para que los docentes generen actividades STEAM adaptadas a las necesidades de sus estudiantes. Incluye recomendaciones basadas en el desarrollo neurocognitivo.

Generar actividad: Utiliza inteligencia artificial para crear actividades personalizadas a partir de los parámetros definidos por el docente. Las actividades generadas están alineadas con los objetivos de aprendizaje y las características del grupo de estudiantes.

Guardar actividad: Permite almacenar las actividades creadas en el sistema, asegurando su disponibilidad para futuras modificaciones o reutilización.

Modificar actividad: Facilita la edición de actividades previamente creadas, permitiendo ajustes en los contenidos, la dificultad y la estructura según las necesidades cambiantes del grupo de estudiantes.

Descargar actividad: Ofrece la opción de exportar las actividades en distintos formatos para su impresión o uso en otras plataformas educativas.

Editar prompt: Permite a los docentes modificar las instrucciones utilizadas por la inteligencia artificial para generar actividades más precisas y alineadas con sus objetivos pedagógicos.

Historial de creación de actividades: Proporciona un registro de todas las actividades generadas y modificadas, permitiendo a los docentes realizar un seguimiento de su evolución y reutilizar materiales previamente creados.

Usuarios del Sistema

El sistema cuenta con dos tipos de usuarios principales:

- **Docentes:** Son los usuarios finales encargados de interactuar con el asistente inteligente para generar, modificar, guardar y descargar actividades STEAM personalizadas, orientadas al desarrollo neurocognitivo de sus estudiantes.
- **Administradores:** Son los desarrolladores o personal técnico que gestionan aspectos internos del sistema, como la edición de los prompts, supervisión de funcionamiento y mejoras técnicas del asistente.

Restricciones

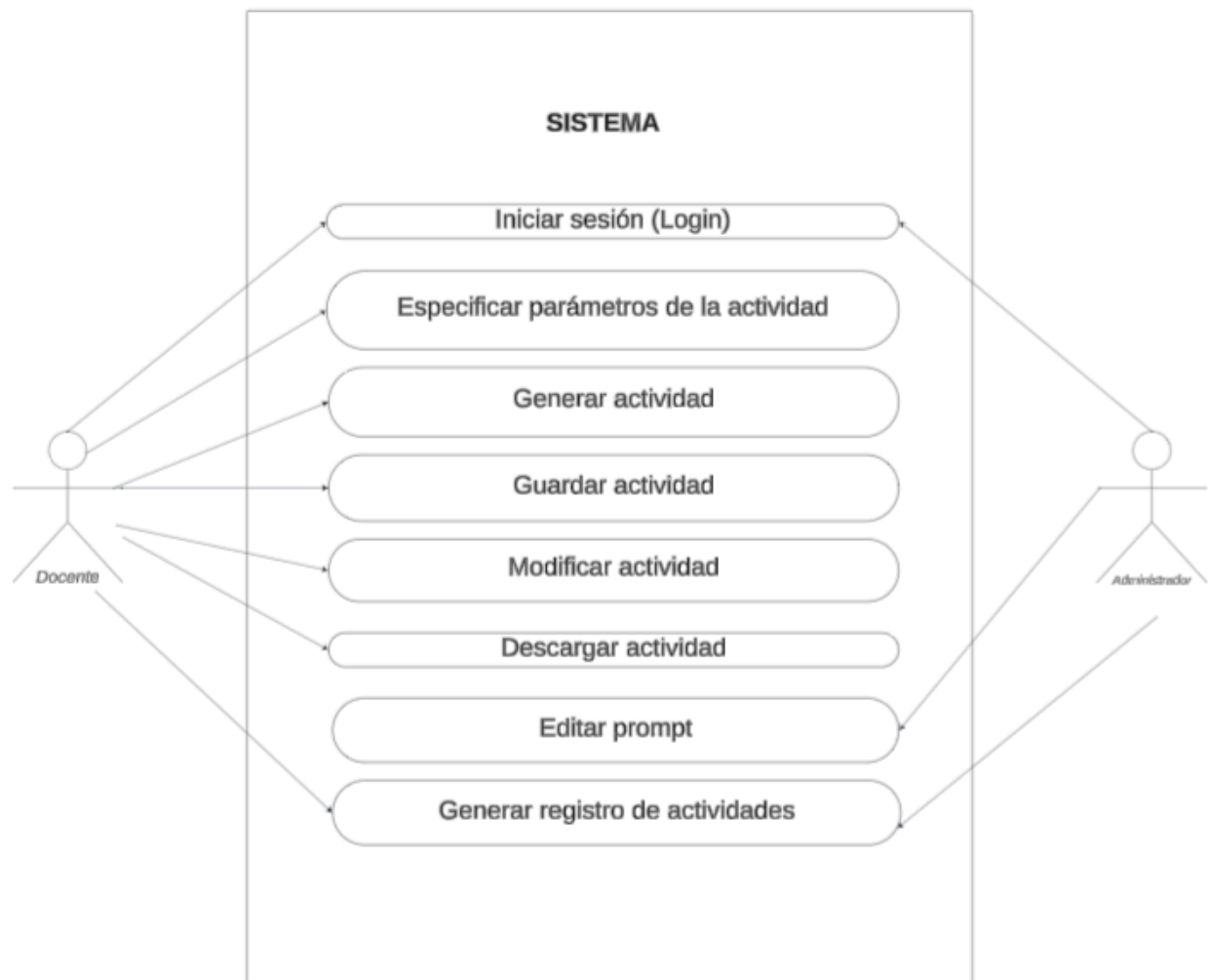
1. El usuario debe tener conexión a internet para acceder a las funcionalidades del sistema, ya que depende del consumo de datos y servicios en línea.

- 2.Sólo los usuarios previamente registrados pueden iniciar sesión y acceder a las funcionalidades disponibles.
- 3.Las funcionalidades del asistente están limitadas a actividades enfocadas en el nivel de primer grado, alineadas con los hitos del desarrollo cognitivo definidos.
- 4.El sistema está diseñado para funcionar en navegadores compatibles y no se garantiza su rendimiento en dispositivos o software obsoletos.

Requisitos Funcionales

- El sistema debe permitir a los usuarios autenticarse mediante un formulario de inicio de sesión con correo electrónico y contraseña.
- El sistema debe ofrecer una sección con instrucciones para guiar al docente en el proceso de creación de actividades.
- El sistema debe generar actividades educativas STEAM personalizadas a partir de los parámetros ingresados por el docente.
- El sistema debe permitir guardar las actividades generadas en la base de datos para su posterior consulta o edición.
- El sistema debe permitir la modificación de actividades previamente creadas por el usuario.
- El sistema debe permitir la descarga de actividades en un formato compatible (PDF, DOCX u otro).
- El sistema debe permitir a los administradores editar los prompts utilizados por la IA para personalizar la generación de actividades.
- El sistema debe almacenar un historial de actividades creadas o modificadas por cada usuario, accesible desde su perfil.

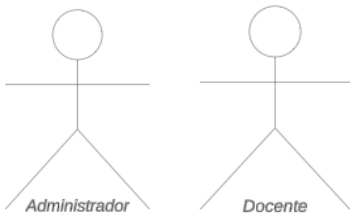
Casos de Uso



Descripción detallada de cada caso de uso

Diagramas de Flujo de Casos de Uso

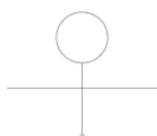
CASO No. 1 Iniciar sesión (Login)

 <p> Versión: 1 Urgencia: 5 Esfuerzo: 3 </p> <p> ICC: Introducir correo y contraseña VC: Verificar credenciales AC: Acceso concedido </p>	<p>1. Iniciar sesión</p> <p>Flujo: Iniciar sesión</p> <p>Prueba: Usuario introduce sus credenciales, el sistema las válida y permite el acceso si son correctas.</p> <p>a - Iniciar Sesión</p> <p>Flujo: ICC,VC,AC</p>
--	--

ID:	CDU-1	
Nombre	Iniciar sesión	
Actores	Docente y administrador	
Objetivo	Permitir el acceso al sistema mediante credenciales	
Urgencia	5	
Esfuerzo	3	
Precondiciones	El usuario debe estar registrado en el sistema	
Flujo normal	Docente	Sistema
	El docente procede a colocar su correo y contraseña	

		El sistema le retorna un mensaje de validación
	El docente le da a “Iniciar sesión”	
		El sistema le permite el acceso al docente
Flujo alternativo 1	El docente procede a colocar su correo y contraseña	
		El sistema le retorna un mensaje de validación
	El docente le da a “Iniciar sesión”	
		El sistema no permite el acceso porque no encuentra al usuario registrado
Flujo alternativo 2	El docente procede a colocar su correo y contraseña	
		El sistema no arroja nada porque no hay conexión

CASO No. 2 Especificar parámetros de la actividad

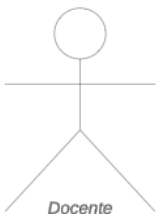


<p>Versión: 1 Urgencia: 4 Esfuerzo: 2</p> <p>AC: Acceder al chat VI: Visualizar instrucciones CC: Confirmar comprensión</p>	<p>1. Instrucciones de creación de actividad Flujo: Instrucciones de creación de actividad Prueba: Usuario accede a la sección de instrucciones, revisa los pasos y confirma su comprensión.</p> <p>a - Instrucciones de creación de actividad Flujo: AC,VI,CC</p>
---	--

ID:	CDU-2	
Nombre	Especificar parámetros de la actividad	
Actores	Docente	
Objetivo	Brindar una guía sobre cómo crear actividades en el sistema	
Urgencia	4	
Esfuerzo	2	
Precondiciones	El usuario debe haber iniciado sesión	
Flujo normal	Docente	Sistema
	El docente le da a “Ver instrucciones”	
		El sistema le retorna un recuadro con todas las instrucciones
	El docente revisa las instrucciones	

		El sistema le retorna un mensaje de “Listo”
	El docente confirma el mensaje	
		El sistema le retorna un recuadro para generar actividad
Flujo alternativo 1	El docente le da en “Ver instrucciones”	
		El sistema le retorna un recuadro con todas las instrucciones
	El docente revisa las instrucciones	
		El sistema le retorna un mensaje de “Listo”
	El docente no confirma el mensaje	

CASO No. 3 Generar actividad

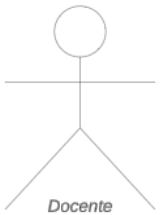
 <p>Docente</p> <p>Versión: 1 Urgencia: 5 Esfuerzo: 4</p> <p>GA: Generar actividad DP: Describir parámetros</p>	<p>1. Generar actividad Flujo: Generar actividad Prueba: Usuario ingresa los parámetros de la actividad, el sistema genera la actividad y la almacena correctamente.</p> <p>a - Generar actividad Flujo: GA,DP,CD,GAG</p>
--	---

CD: Cargar datos	
GAG: Guardar actividad generada	

ID:	CDU-3	
Nombre	Generar actividad	
Actores	Docente	
Objetivo	Crear una actividad personalizada basada en IA	
Urgencia	5	
Esfuerzo	4	
Precondiciones	Se deben proporcionar los parámetros de la actividad	
Flujo normal	Docente	Sistema
		El sistema retorna un recuadro para colocar los parámetros de la actividad
	El docente coloca los parámetros de la actividad	
		El sistema le retorna un mensaje de confirmación
	El docente confirma la actividad	
		El sistema le retorna la actividad generada
		El sistema guarda la actividad
Flujo alternativo 1		El sistema retorna un recuadro para colocar los parámetros de la actividad

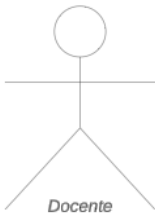
	El docente coloca los parámetros	
		El sistema le retorna un mensaje de confirmación
	El docente no confirma la actividad	
		El sistema retorna un mensaje de “Desea cambiar los parámetros”
	El docente confirma y cambia los parámetros	
		El sistema retorna la actividad generada
		El sistema guarda la actividad correctamente

CASO No. 4 Guardar actividad

 <p>Docente</p> <p>Versión: 1 Urgencia: 5 Esfuerzo: 5</p> <p>SA: Seleccionar actividad</p> <p>GD: Guardar datos</p> <p>VI: Validar información</p> <p>CG: Confirmar guardado</p>	<p>1. Guardar actividad Flujo: Guardar actividad Prueba: Usuario selecciona una actividad y la guarda en el sistema, verificando que los datos se almacenen correctamente.</p> <p>a - guardar actividad Flujo: SA,GD,VI,CG</p>
---	--

ID:	CDU-4	
Nombre	Guardar actividad	
Actores	Docente	
Objetivo	Permitir almacenar una actividad generada en el sistema	
Urgencia	5	
Esfuerzo	5	
Precondiciones	La actividad debe haber sido generada previamente	
Flujo normal	Docente	Sistema
	El docente selecciona una actividad	
		El sistema le retorna la actividad
		El sistema le retorna un botón de guardar actividad
	El docente presiona el botón	
		El sistema guarda la actividad correctamente
		El sistema retorna un mensaje “Actividad guardada correctamente”

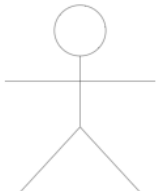
CASO No. 5 Modificar actividad

 <p>Docente</p> <p>Versión: 1 Urgencia: 5 Esfuerzo: 2</p> <p>SA: Seleccionar actividad MD: Modificar datos VC: Validar cambios CA: Cargar actividad</p>	<p>1. Modificar actividad Flujo: Modificar actividad Prueba: Usuario selecciona una actividad, edita sus datos y la carga nuevamente en el sistema.</p> <p>a -Modificar actividad Flujo: SA,MD,VC,CA</p>
--	--

ID:	CDU-5	
Nombre	Modificar actividad	
Actores	Docente	
Objetivo	Editar los datos de una actividad previamente guardada	
Urgencia	5	
Esfuerzo	2	
Precondiciones	La actividad debe existir en el sistema	
Flujo normal	Docente	Sistema
	El docente selecciona "Modificar actividad"	
		El sistema le retorna una lista de las actividades guardadas

	El docente selecciona una actividad	
		El sistema le retorna la actividad editable
	El docente edita los datos de la actividad	
		El sistema le retorna un mensaje “Confirmar cambios”
	El docente confirma los cambios	
		El sistema le retorna la actividad modificada
		El sistema guarda la actividad correctamente

CASO No. 6 Descargar actividad

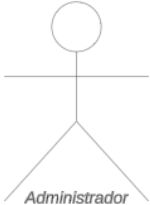
	1. Descargar actividad
---	------------------------

<p>Versión: 1 Urgencia: 4 Esfuerzo: 2</p> <p>SA: Seleccionar actividad DA: Descargar archivo CD: Confirmar descarga</p>	<p>Flujo:Descargar actividad Prueba: Usuario selecciona una actividad y la descarga correctamente en formato compatible.</p> <p>a - Descargar actividad Flujo: SA,DA,CD</p>
---	---

ID:	CDU-6	
Nombre	Descargar actividad	
Actores	Docente	
Objetivo	Permitir la descarga de una actividad en un formato compatible	
Urgencia	4	
Esfuerzo	2	
Precondiciones	Debe haber al menos una actividad disponible	
Flujo normal	Docente	Sistema
	El docente selecciona descargar actividad	
		El sistema le retorna una lista de actividades
	El docente selecciona una actividad en concreto	
		El sistema tiene un botón de

		“Descargar actividad”
	El docente selecciona “Descargar actividad”	
		El sistema le presenta la actividad en un formato descargable
	El docente confirma la descarga	

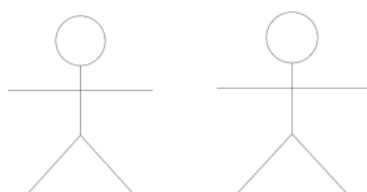
CASO No.7 Editar prompt

 <p>Versión: 1 Urgencia: 2 Esfuerzo: 2</p> <p>SP: Seleccionar prompt EP: Editar contenido VC: Validar cambios GP: Guardar prompt</p>	<p>1. Editar prompt Flujo: Editar prompt Prueba: Usuario selecciona un prompt, lo edita y guarda los cambios en el sistema.</p> <p>a - Editar prompt Flujo: SP,EP,VC,GP</p>
--	---

ID:	CDU-7
Nombre	Editar prompt
Actores	Administrador

Objetivo	Modificar el contenido de un prompt utilizado en la generación de actividades	
Urgencia	2	
Esfuerzo	2	
Precondiciones	Debe existir al menos un prompt guardado	
Flujo normal	Administrador	Sistema
	El administrador selecciona "Editar prompt"	
		El sistema le retorna el recuadro donde se encuentra el prompt
	El administrador edita los datos del prompt	
		El sistema retorna un mensaje de confirmación "Confirmar cambios"
	El administrador confirma los cambios	
		El sistema guarda los cambios del prompt

CASO No. 8 Generar registro de actividades



<p>Versión: 1 Urgencia: 1 Esfuerzo: 5</p> <p>AH: Acceder al historial</p> <p>VH: Visualizar actividades previas</p> <p>FF: Filtrar por fecha o tipo</p>	<p>1. Historial de creación de actividades Flujo: Historial de creación de actividades Prueba: Usuario accede al historial, visualiza actividades previas y puede filtrarlas según criterio.</p> <p>a - Historial de creación de actividades</p> <p>Flujo: AH,VH,FF</p>
---	---

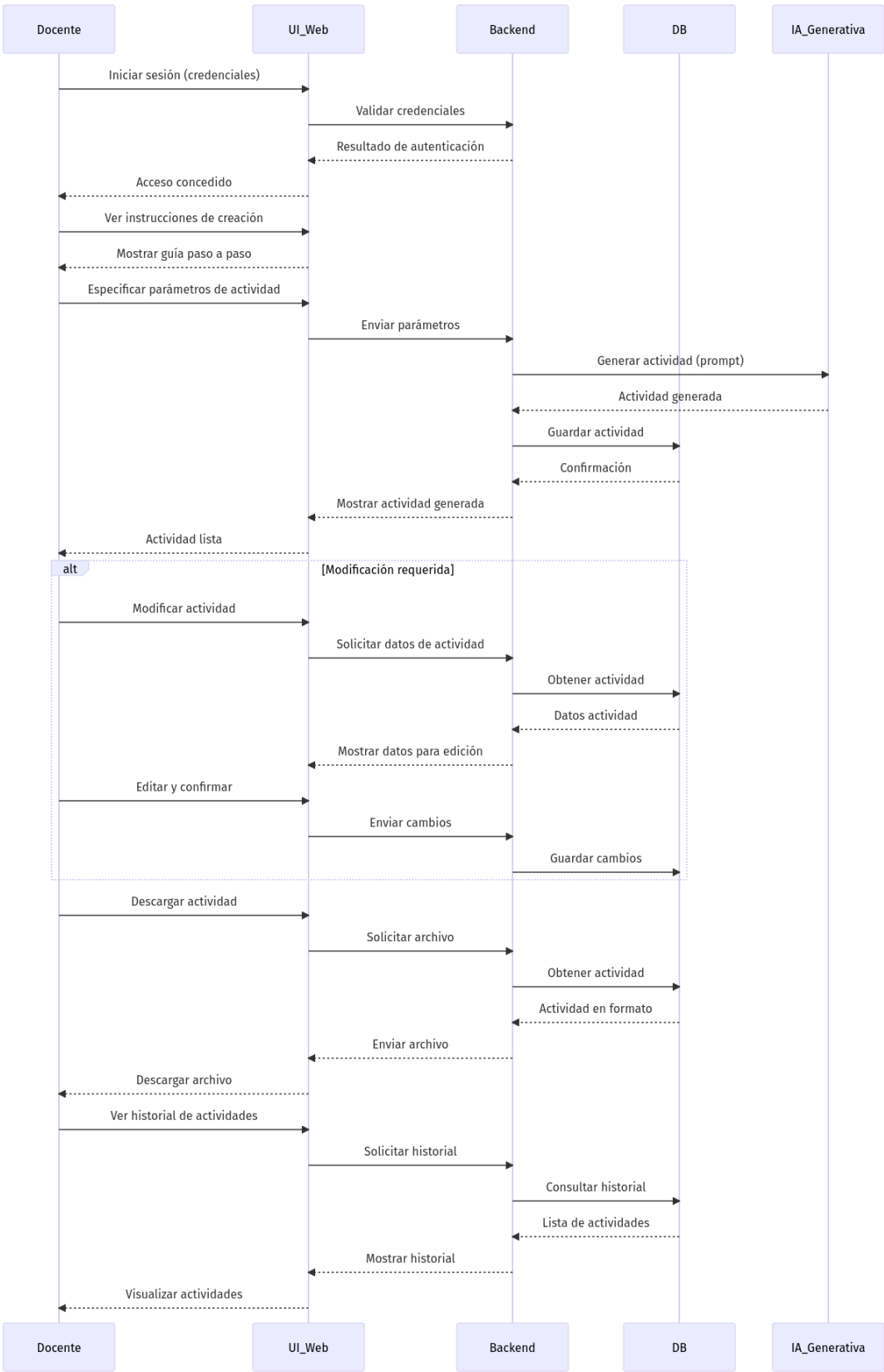
ID:	CDU-8	
Nombre	Historial de creación de actividades	
Actores	Docente y administrador	
Objetivo	Consultar actividades creadas previamente y filtrarlas según criterio	
Urgencia	1	
Esfuerzo	5	
Precondiciones	Deben existir actividades almacenadas en el historial	
Flujo normal	Docente	Sistema
	Selecciona ver historial de actividades creadas	
		Retorna una lista de todas las actividades creadas
	Indica criterios de filtrado para las actividades	

		Retorna una lista de las actividades creadas de acuerdo a los criterios
Flujo alternativo 1	Selecciona ver historial de actividades creadas	
		Retorna “No hay actividades existentes”
Flujo alternativo 2	Selecciona ver historial de actividades creadas	
		No retorna nada por falta de conexión a internet

Matriz de prioridades

Urgencia						
E s f u e r z o		1 Baja	2 menor	3 Moderada	4 Alta	5 obligatoria
	5 Muy Alto	5	10	15	20	25
		CDU-8				CDU-4 CDU-6
	4 Alto	4	8	12	16	20
					CDU-3	
	3 Medio	3	6	9	12	15
						CDU-1
	2 bajo	2	4	6	8	10
			CDU-7		CDU-2 CDU-6	CDU-5
	1 Muy bajo	1	2	3	4	5

Diagramas de Secuencia



Prioridad de Requisitos

Requisito	Urgencia	Esfuerzo	Prioridad
Inicio de sesión	5	3	Alta
Instrucciones de creación	4	2	Media
Generar actividad	5	4	Alta
Guardar actividad	5	5	Alta
Modificar actividad	5	2	Alta
Descargar actividad	4	2	Media
Editar prompt (admin)	2	2	Baja
Historial de actividades	1	5	Baja

Requisitos No Funcionales

Requisitos de Desempeño

El sistema debe ofrecer un tiempo de respuesta ágil en sus principales funcionalidades. La generación de actividades debe completarse en un máximo de cinco segundos tras el ingreso de los parámetros por parte del docente. De igual manera, las operaciones de guardar, modificar y descargar actividades deben ejecutarse en menos de tres segundos. El sistema debe mantener un rendimiento estable cuando se atienden hasta diez usuarios simultáneos, sin afectar la experiencia de uso.

Requisitos de Seguridad

Para garantizar la protección de los datos y la privacidad de los usuarios, el sistema debe implementar autenticación mediante correo electrónico y contraseña cifrada. Toda la comunicación entre el cliente y el servidor debe utilizar el protocolo HTTPS para evitar accesos no autorizados. Además, se deben establecer controles de acceso que distingan claramente entre docentes y administradores, asegurando que solo usuarios autorizados puedan acceder o modificar determinadas funciones y datos del sistema.

Requisitos de Usabilidad

La interfaz del sistema debe ser intuitiva, accesible y diseñada para docentes con distintos niveles de experiencia tecnológica. Las instrucciones de uso deben estar claramente disponibles y explicadas paso a paso, permitiendo una navegación sencilla. Los botones deben estar bien identificados, acompañados de íconos representativos, y deben incluir mensajes visuales de confirmación o error que orienten al usuario durante el uso del asistente.

Requisitos de Escalabilidad

El sistema debe estar preparado para crecer tanto en funcionalidad como en capacidad. Esto implica que su arquitectura permita añadir nuevas herramientas, como el seguimiento del rendimiento de los estudiantes o la generación de reportes. Además, debe admitir una ampliación de la base de datos sin comprometer la velocidad ni la estabilidad. También debe facilitar la integración con otras plataformas educativas mediante APIs, y tener la flexibilidad para adaptarse a nuevos niveles escolares en futuras versiones.

Modelado E/R

Caracterización de los datos

Diagrama de Entidad-Relación

Diagrama relacional

Descripción de Entidades y Relaciones

Reglas de Integridad

Diagramas Adicionales

Referencias

Etapa 2: Persistencia de Datos con Backend

Introducción

Esta segunda etapa del proyecto se centra en la construcción de la capa de persistencia de datos del asistente inteligente NeuroSTEAM. En esta fase, se aborda el diseño y desarrollo de la infraestructura que permitirá almacenar, consultar, modificar y gestionar la información generada por el sistema, tales como las actividades STEAM, los registros de usuarios y los prompts utilizados. Para ello, se integran conocimientos técnicos sobre bases de datos, servidores y tecnologías backend, asegurando una base sólida y escalable que respalde el funcionamiento del sistema educativo.

Propósito de la Etapa

El propósito de esta etapa es diseñar e implementar la arquitectura de backend que permita al asistente inteligente interactuar con una base de datos para garantizar la persistencia de la información. Esto incluye la construcción de APIs, el desarrollo de lógica de negocio, la configuración de conexiones seguras y la creación de endpoints que conecten eficientemente el frontend con la base de datos. A través de esta estructura, se busca asegurar que los datos generados por los docentes se almacenen correctamente y estén disponibles para futuras consultas, ediciones y reportes.

Alcance de la Etapa

Esta fase abarca desde la elección del tipo de base de datos (SQL o NoSQL) hasta la implementación completa de la lógica backend del sistema. Incluye el diseño del modelo de datos, el desarrollo de los servicios necesarios para la creación, lectura, actualización y eliminación de registros (operaciones CRUD), y la integración con el asistente inteligente mediante APIs. Asimismo, contempla la autenticación y autorización de usuarios, y la preparación del sistema para ser desplegado en un entorno en la nube. Todo esto se realiza considerando prácticas seguras, eficientes y escalables para soportar futuras expansiones del proyecto.

Definiciones y Acrónimos

API (Application Programming Interface): Conjunto de funciones y protocolos que permiten la comunicación entre el frontend y el backend del sistema.

CRUD (Create, Read, Update, Delete): Conjunto de operaciones básicas que se pueden realizar sobre los datos almacenados.

DBMS (Database Management System): Sistema que permite la administración de bases de datos.

SQL (Structured Query Language): Lenguaje utilizado para manejar bases de datos relacionales.

NoSQL: Tipo de base de datos no relacional, ideal para datos flexibles y no estructurados.

Backend: Parte del sistema encargada del procesamiento interno y la lógica de negocio, no visible para el usuario final.

Endpoint: Punto de acceso a una funcionalidad específica del backend, normalmente expuesto a través de una API.

Autenticación: Proceso mediante el cual se verifica la identidad del usuario.

Autorización: Control sobre los permisos que tiene un usuario autenticado dentro del sistema.

JSON (JavaScript Object Notation): Formato ligero de intercambio de datos utilizado en la comunicación entre cliente y servidor.

Node.js / Express.js: Tecnologías de desarrollo backend utilizadas para construir servidores ligeros y eficientes.

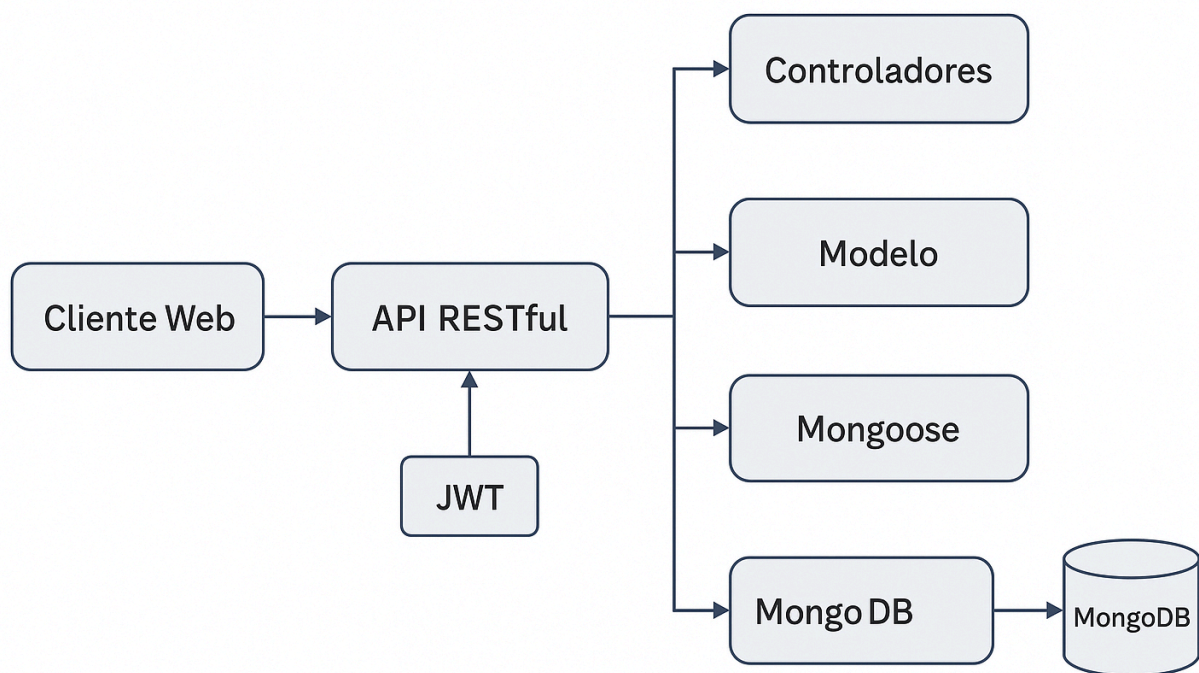
Diseño de la Arquitectura de Backend

La arquitectura propuesta para el backend se basa en el patrón Modelo-Vista-Controlador (MVC), utilizando Node.js y el framework Express.js. Esta estructura permite una separación de responsabilidades que facilita el mantenimiento del código, la reutilización de componentes y la escalabilidad del sistema. El backend expone una serie de endpoints a través de una API RESTful que se conectará a una base de datos NoSQL (MongoDB), y se encargará de gestionar las operaciones de persistencia, autenticación, autorización y lógica de negocio.

Descripción de la Arquitectura Propuesta

La arquitectura del sistema backend para NeuroSTEAM adopta un enfoque modular basado en el patrón Modelo-Vista-Controlador (MVC). Este patrón separa la lógica de negocio, la manipulación de datos y las rutas, facilitando el mantenimiento, la escalabilidad y la reutilización del código. La solución se estructura como una API RESTful desarrollada con Node.js y Express.js, tecnologías modernas y eficientes para la construcción de servidores livianos y performantes.

El servidor backend actúa como intermediario entre el frontend (interfaz de usuario) y la base de datos, gestionando las solicitudes del cliente, procesando la lógica del negocio y realizando operaciones de lectura y escritura en la base de datos. La comunicación entre el cliente y el servidor se realiza mediante solicitudes HTTP utilizando el formato JSON como estándar para el intercambio de datos. El almacenamiento de información se gestiona a través de MongoDB, una base de datos NoSQL alojada en la nube (MongoDB Atlas), elegida por su flexibilidad para trabajar con estructuras de datos dinámicas como las actividades STEAM, prompts y configuraciones personalizadas. Para interactuar con la base de datos, se utiliza Mongoose, una biblioteca que proporciona una capa de abstracción que permite definir esquemas, realizar validaciones y simplificar las consultas.

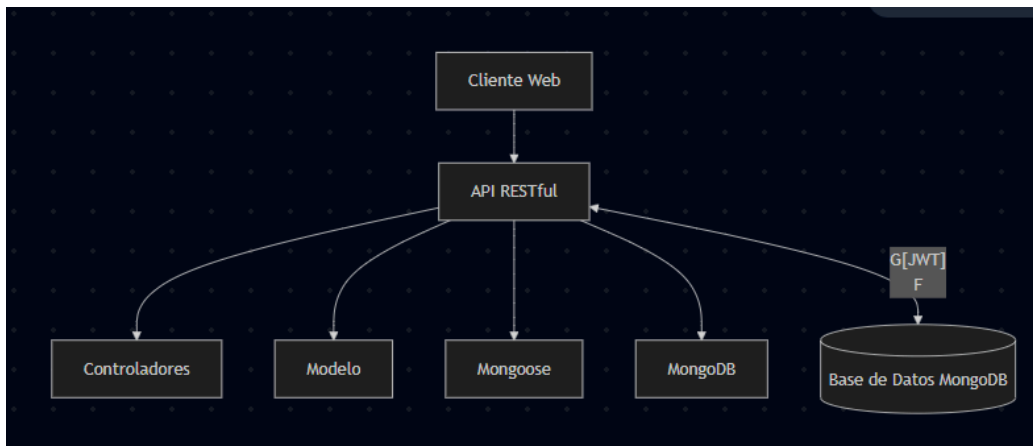


Componentes del Backend

- Servidor HTTP (Express.js): Gestiona las rutas y recibe las solicitudes del frontend.
- Controladores: Implementan la lógica de negocio para cada operación (crear, consultar, modificar y eliminar datos).
- Modelos (Mongoose): Definen la estructura de los datos almacenados en la base de datos.
- Middleware: Gestiona tareas transversales como autenticación, manejo de errores y validación de solicitudes.
- Base de Datos (MongoDB): Almacena persistentemente usuarios, actividades, prompts e historial de creación.
- Módulo de autenticación (JWT): Crea, válida y protege los tokens de sesión.
- Archivo de configuración (.env): Contiene credenciales y variables sensibles como claves de API y URI de conexión.

Diagramas de Arquitectura

El flujo general del aplicativo es: Cada solicitud enviada por el cliente se enruta hacia un controlador específico, el cual consulta o actualiza la base de datos según la lógica del sistema. La respuesta se envía nuevamente al cliente en formato JSON.



Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

Se compararon las opciones entre bases de datos relacionales (SQL) y no relacionales (NoSQL). Mientras que SQL es ideal para datos estructurados, NoSQL ofrece mayor flexibilidad y escalabilidad para estructuras de datos dinámicas como las que maneja este asistente.

- Bases de datos SQL (relacionales): como MySQL o PostgreSQL, ofrecen integridad referencial, normalización y estructuras rígidas.
- Bases de datos NoSQL: como MongoDB, permiten una mayor flexibilidad, almacenamiento basado en documentos, y escalabilidad horizontal.

Justificación de la Elección

Se eligió MongoDB como base de datos por su modelo de documentos, que se ajusta al tipo de datos generados por el sistema (actividades, prompts y usuarios). Además, su integración con Mongoose permite una gestión eficiente de esquemas, validaciones y relaciones entre datos. Para varias, la elección de mongoDB como gestor de la base de datos del asistente, se dio también por el manejo que tiene el repositorio interno del grupo de investigación dentro de GitHub, para así facilitar el manejo de la información y estar más a la vanguardia de las nuevas tecnologías, en este caso, el manejo de información en bases de datos no relacionales.

Diseño de Esquema de Base de Datos

Se definieron tres actores o colecciones principales para el aplicativo:

- Usuarios:

```
{
  "_id": "ObjectId",
  "nombre": "string",
  "email": "string",
  "password": "hashed",
  "rol": "docente | administrador"
}
```

- Actividades:

```
{
  "_id": "ObjectId",
  "usuarioId": "ObjectId",

```

```
"titulo": "string",  
"contenido": "string",  
"fechaCreacion": "Date"  
}
```

- **Prompts:**

```
{  
  "_id": "ObjectId",  
  "descripcion": "string",  
  "contenido": "string",  
  "fechaModificacion": "Date"  
}
```

Implementación del Backend

Elección del Lenguaje de Programación

Se empleó JavaScript, ejecutado en el entorno de Node.js, por ser ligero, asíncrono y ampliamente utilizado para desarrollo web. Junto con Express.js y Mongoose, permite una estructura coherente y eficiente.

Creación de la Lógica de Negocio

La lógica implementa funciones para la autenticación de usuarios, gestión de actividades, edición de prompts y registro de historial. Cada operación incluye validaciones y mensajes de respuesta. Cada funcionalidad está encapsulada en controladores, que se encargan de:

- Validar la solicitud del usuario.
- Ejecutar reglas de negocio.
- Manipular los datos mediante operaciones sobre los modelos.
- Enviar la respuesta al cliente en formato JSON.

Desarrollo de Endpoints y APIs

Se desarrollaron rutas y endpoints, de los más relevantes, como:

- POST /login – Autenticación del usuario.
- POST /actividad – Crear nueva actividad.
- GET /actividades/:usuarioId – Consultar historial.
- PUT /actividad/:id – Modificar actividad.
- GET /prompts – Obtener lista de prompts.
- PUT /prompt/:id – Editar contenido del prompt.

Autenticación y Autorización

El sistema emplea tokens JWT para proteger rutas privadas. Estos tokens se generan tras una autenticación exitosa y permiten controlar el acceso de usuarios a funcionalidades específicas, según su rol.

Conexión a la Base de Datos

Configuración de la Conexión

Se utilizó un archivo `.env` para configurar la URI de MongoDB y otras variables sensibles. La conexión se establece con `mongoose.connect`, incluyendo opciones de reconexión y tolerancia a fallos.

Desarrollo de Operaciones CRUD

Todas las operaciones de Crear, Leer, Actualizar y Eliminar se realizan desde controladores especializados, que usan los modelos definidos en Mongoose para interactuar con la base de datos.

Manejo de Transacciones

Para garantizar integridad en operaciones múltiples, se pueden usar transacciones con “session” de Mongoose, por ejemplo, al eliminar un usuario y sus actividades asociadas.

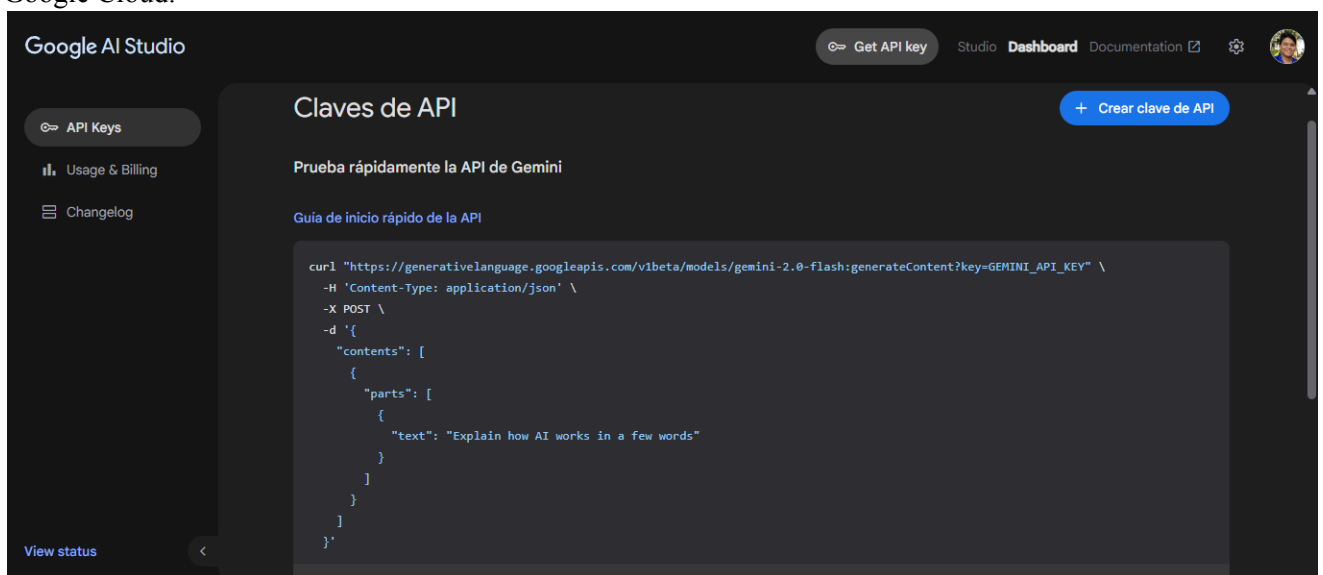
Pruebas del Backend

Diseño de Casos de Prueba

Las pruebas incluyen:

- Inicio de sesión con credenciales válidas/erróneas.
- Generación de actividades con parámetros válidos/incompletos.
- Autorización de rutas protegidas.
- Comportamiento ante peticiones incorrectas.

Algunos de los casos de prueba, más importantes, fueron al momento de conectar la Api de una inteligencia artificial, para que el asistente respondiera como tal y no con propiedades que le fueran dadas. La conexión de una Api es algo no tan sencillo, pero al funcionar, se logró que respondiera. Se usó la Api de Gemini, dad por Google Cloud.



Ejecución de Pruebas Unitarias y de Integración

Durante el proceso de desarrollo del backend, se implementó una estrategia de aseguramiento de calidad basada en pruebas unitarias y pruebas de integración, con el fin de garantizar el correcto funcionamiento de cada componente del sistema y su interacción con el resto de la arquitectura.

Para las pruebas unitarias, se utilizó Postman, debido a su facilidad de configuración, velocidad de ejecución y su capacidad para realizar pruebas asincrónicas, ideales para aplicaciones Node.js. Con Postman, se testean funciones individuales, de forma local, de los controladores, como validaciones de entrada, respuestas esperadas y comportamientos bajo condiciones límites. Por ejemplo, se validó que la función `crearActividad()` devuelva un error si se omiten campos obligatorios, o que `loginUsuario()` retorne un token JWT válido tras una autenticación exitosa.

Adicionalmente, se realizaron pruebas de integración utilizando Supertest, una herramienta que permite simular peticiones HTTP contra la API como si fueran realizadas desde el cliente. Esto permitió validar el comportamiento completo del sistema, desde el punto de entrada hasta la base de datos. Se verificó, por ejemplo, que al realizar un POST `/actividad` con parámetros válidos, la actividad se almacene correctamente y se devuelva una respuesta con estado 201 Created y el objeto guardado. Las pruebas de integración incluyeron escenarios como:

- Inicio de sesión con credenciales válidas e inválidas.
- Creación y recuperación de actividades.
- Comprobación de tokens JWT en rutas protegidas.
- Verificación del flujo de edición y eliminación de registros.

Estas pruebas se ejecutaron en un entorno de desarrollo aislado, utilizando una base de datos de prueba para evitar interferencias con los datos reales del sistema.

Manejo de Errores y Excepciones

El sistema backend fue diseñado con una política robusta de manejo de errores y excepciones para mejorar la estabilidad, facilitar la depuración y proporcionar retroalimentación clara tanto al usuario como al equipo de desarrollo. Para ello, se implementó un middleware centralizado de manejo de errores en Express.js, que intercepta cualquier error no capturado durante la ejecución de los controladores o rutas. Este middleware analiza el tipo de error y genera una respuesta estructurada en formato JSON que incluye:

- Código de estado HTTP (`statusCode`).
- Mensaje descriptivo del error (`message`).
- Detalles técnicos opcionales en modo desarrollo (`stack`).

Se definen diferentes tipos de errores según el contexto:

- 400 Bad Request: cuando la solicitud no contiene los datos requeridos o hay errores de validación.
- 401 Unauthorized: cuando el usuario no está autenticado o el token JWT es inválido o ha expirado.
- 403 Forbidden: cuando el usuario autenticado intenta acceder a una función sin los permisos necesarios.

- 404 Not Found: cuando se solicita un recurso inexistente.
- 429 Error en la solicitud: cuando simplemente no respondía el asistente.
- 500 Internal Server Error: para errores inesperados del servidor, como fallos de conexión o excepciones no manejadas.

Además, en los controladores se emplean estructuras try/catch para capturar errores de lógica o fallos en operaciones asincrónicas, como la conexión a la base de datos o el acceso a colecciones inexistentes. En caso de error, estos se lanzan con clases de error personalizadas que son interpretadas por el middleware central.

Etapas 3: Consumo de Datos y Desarrollo Frontend

Introducción

La tercera etapa del desarrollo del proyecto NeuroSTEAM se enfoca en la creación de la interfaz de usuario y en el consumo de datos provenientes del backend. En esta fase se materializa la experiencia del usuario mediante el diseño y desarrollo del frontend, el cual permite a los docentes interactuar de forma intuitiva y efectiva con el asistente inteligente. Esta etapa articula conocimientos en HTML, CSS y JavaScript, así como el uso de bibliotecas y frameworks que permiten construir una plataforma web dinámica, responsiva y funcional. Además, se garantiza una comunicación efectiva con el servidor mediante llamadas a las APIs implementadas en la etapa anterior.

Propósito de la Etapa

El propósito de esta etapa es implementar la capa de presentación del sistema, desarrollando una interfaz web clara y accesible para los docentes. Esta interfaz debe facilitar el ingreso de parámetros, la visualización de actividades generadas por la IA, así como la gestión y descarga de dichas actividades. Igualmente, se busca establecer una conexión estable y segura con el backend a través de solicitudes HTTP, permitiendo la consulta, envío y actualización de datos en tiempo real. La etapa también incluye la validación de datos en el cliente, manejo de eventos, pruebas de usabilidad y la optimización de la experiencia del usuario (UX).

Alcance de la Etapa

El alcance de esta etapa comprende el diseño gráfico de la interfaz, la programación del comportamiento dinámico del sistema en el navegador y la implementación de funcionalidades que permitan consumir los datos desde la API del backend. Se desarrollarán formularios interactivos, botones de acción, menús y paneles informativos que permitan al usuario realizar operaciones como iniciar sesión, generar actividades, consultar el historial, modificar o descargar archivos. Asimismo, se contempla la validación de datos en el cliente, la implementación de estilos responsivos para diferentes dispositivos y la integración de mensajes de retroalimentación que guíen al usuario en su interacción con el sistema.

Definiciones y Acrónimos

Frontend: Parte visible del sistema con la que interactúa el usuario final. Se desarrolla con tecnologías como HTML, CSS y JavaScript.

UI (User Interface): Interfaz de usuario. Conjunto de elementos visuales que permiten la interacción con el sistema.

UX (User Experience): Experiencia del usuario. Evalúa qué tan fácil, agradable y eficiente es usar una aplicación.

HTML (HyperText Markup Language): Lenguaje de marcado utilizado para estructurar contenido web.

CSS (Cascading Style Sheets): Lenguaje de diseño gráfico que permite aplicar estilos visuales a los elementos HTML.

JS (JavaScript): Lenguaje de programación que permite agregar interactividad a las páginas web.

AJAX (Asynchronous JavaScript and XML): Técnica que permite la comunicación asincrónica entre el frontend y el backend sin recargar la página.

API (Application Programming Interface): Conjunto de funciones que permiten la conexión e intercambio de

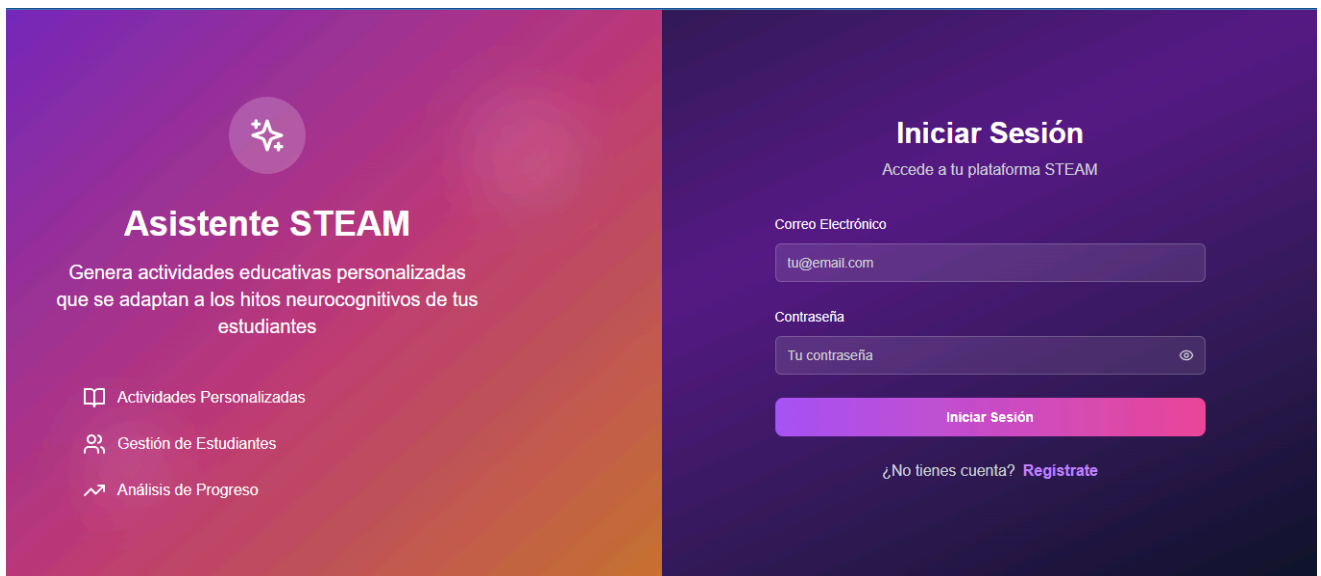
datos entre sistemas.

SPA (Single Page Application): Aplicación web que carga una sola página HTML y actualiza dinámicamente su contenido sin recargar.

Creación de la Interfaz de Usuario (UI)

Diseño de la Interfaz de Usuario (UI) con HTML y CSS

La interfaz fue desarrollada utilizando HTML5 y CSS3, integrando también preprocesadores y utilidades modernas como variables CSS, tipografías personalizadas y diseño en capas. En la versión local, se prioriza un estilo educativo con colores suaves y fondo con íconos escolares, resaltando la marca “NEUROSTEAM”. En la versión desplegada en Hostinger Horizon, se aplicó un diseño más sofisticado con gradientes dinámicos, íconos animados y layouts divididos (pantalla dual), lo que brinda una experiencia inmersiva y moderna. Ambas versiones utilizan cajas centradas para el formulario de inicio de sesión, etiquetas claramente legibles, y contraste de color apropiado para accesibilidad.



STEAM Assistant

Panel Principal

Generador

Mis Actividades

Colaboración

Análisis

Configuración

Conectado como:

Usuario STEAM

Institución Educativa

Mis Actividades

Bienvenido a tu plataforma educativa STEAM

Q

Buscar por título, área o hito...

Filtrar

Tecnología

Actividad Tecnología para Memoria y Aprendizaje

Nivel: Primaria

Dificultad: Básico

Hito: Memoria y Aprendizaje

Construcción de una estructura simple siguiendo instrucciones visuales (diagrama). Luego, intentar replicarla de memoria.

+ Nueva Actividad

NEURO
STEAM

Usuario

Ingresa tu usuario

Contraseña

Ingresa tu contraseña

Iniciar Sesión

https://www.neurosteam.com

Hitos 1ra Infancia

Soporte

Mis Actividades

Favoritos

Actividades Editadas

¡El momento es hoy!

Crea tu actividad. Tus Estudiantes aprenden usando habilidades STEAM

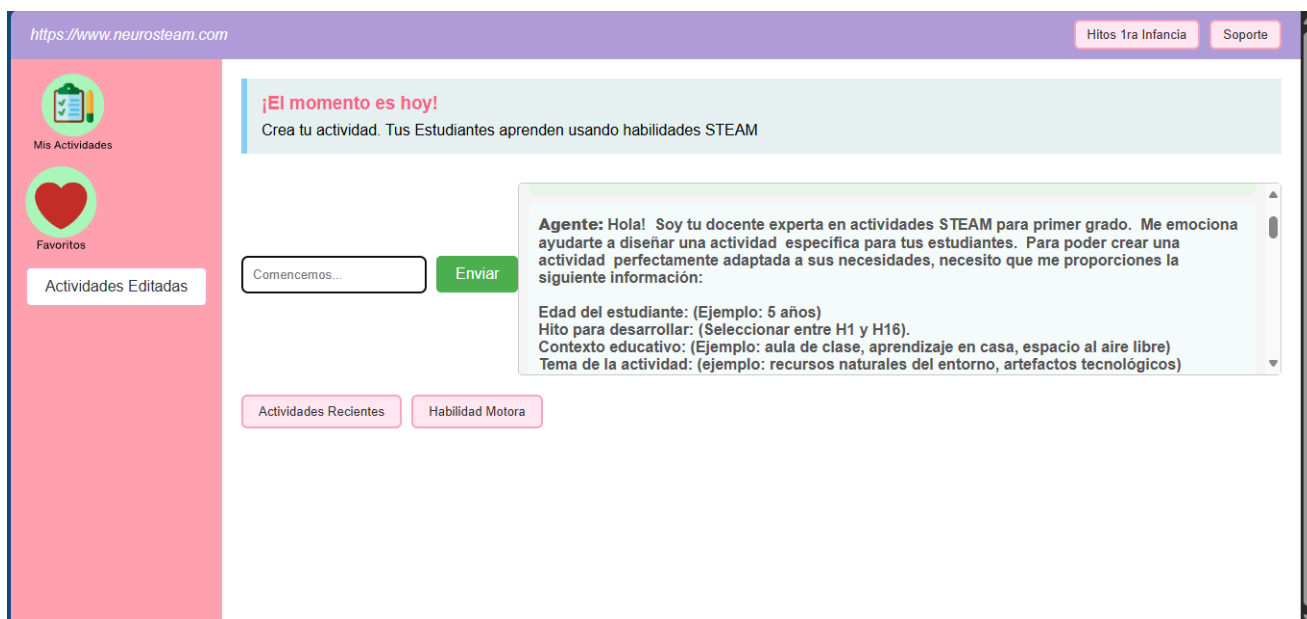
Comencemos...

Enviar

Agente: ¡Hola! Soy tu asistente STEAM. ¿En qué puedo ayudarte hoy?

Actividades Recientes

Habilidad Motora



Consideraciones de Usabilidad

El diseño fue pensado para docentes con diferentes niveles de alfabetización digital. Se usaron:

- Instrucciones visibles sobre cada campo (ej. “Ingresa tu contraseña”).
- Íconos representativos en los menús laterales (actividad, análisis, configuración).
- Mensajes de éxito como “¡Bienvenido! Has iniciado sesión correctamente”.
- Sugerencias dentro de los campos (placeholder).
- Contrastes altos y botones grandes para mejorar la navegación en pantallas táctiles.

Maquetación Responsiva

Se emplearon media queries y contenedores flexbox y grid para adaptar la UI a dispositivos móviles, tablets y PC. La estructura del menú lateral colapsa adecuadamente en pantallas pequeñas y los formularios se adaptan a una columna única para facilitar el llenado.

Programación Frontend con JavaScript (JS)

Desarrollo de la Lógica del Frontend

Se implementaron scripts JS modulares para controlar:

- La autenticación mediante envío de datos por fetch.
- El almacenamiento temporal del token JWT.
- La interacción con formularios dinámicos (como el generador de actividades STEAM).
- La visualización condicional de mensajes y contenidos.

```

Click to add a breakpoint
77 const loginContainer = document.getElementById("loginContainer");
78 const mainContainer = document.getElementById("mainContainer");
79 const loginBtn = document.getElementById("loginBtn");
80 const chatInput = document.getElementById("chatInput");
81 const chatSubmitBtn = document.getElementById("chatSubmitBtn");
82 const chatResponse = document.getElementById("chatResponse");
83
84 let conversationHistory = [];
85 let isFirstInteraction = true;
86
87 chatSubmitBtn.addEventListener("click", async function () {
88   const userText = chatInput.value.trim();
89   if (userText !== "") {
90     chatResponse.innerHTML += `<div class="user-message"><strong>Tú:</strong> ${userText}</div>`;
91     conversationHistory.push({ role: "user", text: userText });
92
93     let messages = [];
94     if (isFirstInteraction) {
95       messages.push({ text: SYSTEM_PROMPT });
96       isFirstInteraction = false;
97     }
98     messages = messages.concat(conversationHistory.map(m => ({ text: m.text })));
99
100     try {
101       const response = await fetch(

```

Manejo de Eventos y Comportamientos Dinámicos

El sistema responde a eventos como:

- onClick en botones de “Iniciar sesión”, “Generar Actividad”, “Guardar”, etc.
- onChange en selectores de hitos, edades, contexto educativo.
- Visualización de mensajes flotantes con setTimeout() para retroalimentar acciones del usuario.

```

frontend > JS script.js > ...
87 chatSubmitBtn.addEventListener("click", async function () {
139   chatResponse.innerHTML += `<div class="bot-message"><strong>Agente:</strong> ${error.message}</div>`;
140 }
141
142 chatInput.value = "";
143 chatResponse.scrollTop = chatResponse.scrollHeight;
144 } else {
145   alert("Por favor, escribe algo en el chat.");
146 }
147 });
148
149 chatInput.addEventListener("keyup", function (event) {
150   if (event.key === "Enter") {
151     chatSubmitBtn.click();
152   }
153 });
154
155 document.addEventListener("DOMContentLoaded", function () {
156   chatResponse.innerHTML += `<div class="bot-message"><strong>Agente:</strong> ¡Hola! Soy tu asistente STEAM. ¿En qué
157 `);
158
159 document.getElementById('btnHijos').addEventListener('click', () => {
160   const infoHitos = document.getElementById('infoHitos');
161   infoHitos.innerHTML = `
162     <h2>Hitos de la Primera Infancia</h2>
163     <strong>Tú:</strong>
${userText}</div>`;
    conversationHistory.push({ role: "user", text: userText });

    let messages = [];
    if (isFirstInteraction) {
      messages.push({ text: SYSTEM_PROMPT });
      isFirstInteraction = false;
    }
    messages = messages.concat(conversationHistory.map(m => ({ text: m.text
})));

    try {
      const response = await fetch(
`https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash:gen
erateContent?key=${GEMINI_API_KEY}`,
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          contents: [
```

```

        {
            parts: messages
        }
    ]
    })
}
);

if (!response.ok) {
    throw new Error(`Error en la solicitud: ${response.status}
${response.statusText}`);
}

const data = await response.json();

```

Obtención y Presentación de Datos

Los datos recibidos (ej. actividades, historial) son parseados desde JSON y mostrados mediante elementos dinámicos como:

- Tarjetas de resumen (div con íconos, cifras e indicadores).
- Listados interactivos para ver actividades generadas.
- Formularios pre-llenados en caso de edición.

Actualización en Tiempo Real (si aplicable)

Aunque no se utiliza WebSocket, se implementaron funciones como:

- Refrescar automáticamente los datos del historial tras guardar una actividad.
- Notificaciones emergentes que informan acciones exitosas sin recargar la página.

Interacción Usuario-Interfaz

Manejo de Formularios y Validación de Datos

Cada formulario válida:

- Campos requeridos (required).
- Longitud mínima.
- Formato correcto (correo electrónico, selección obligatoria).
En caso de error, se muestra un mensaje visible con estilos CSS como bordes rojos y texto de advertencia.

Implementación de Funcionalidades Interactivas

El sistema permite:

- Crear actividades seleccionando múltiples parámetros.
- Mostrar resultados generados dinámicamente por la IA.
- Guardar, modificar o eliminar actividades mediante botones accionables.
- Descarga de archivos (PDF/DOCX).
- Ver la actividad en un formato de texto plano-
- Tener la respuesta del asistente de manera inmediata.

Mejoras en la Experiencia del Usuario

Se utilizaron recursos para optimizar la experiencia:

- Interfaz clara, iconografía educativa y diseño visual atractivo.
- Carga rápida y navegación fluida sin recargas completas (SPA).
- Mensajes en tiempo real como “Has iniciado sesión correctamente” o “Actividad guardada”.

Pruebas y Depuración del Frontend

Diseño de Casos de Prueba de Frontend

Se diseñaron escenarios como:

- Usuario intenta iniciar sesión con credenciales inválidas.
- Validación de formularios vacíos.
- Pruebas de generación de actividades con parámetros erróneos.
- Navegación desde el menú lateral.

Se pueden ver mejor estos casos, en la etapa 1 del desarrollo del asistente.

Pruebas de Usabilidad

Se aplicaron pruebas con usuarios docentes reales, quienes identificaron mejoras como:

- Aumentar contraste de botones.
- Reubicar el botón “Generar Actividad” más cerca del campo objetivo.

Estas sugerencias fueron integradas en la versión final.

Depuración de Errores y Optimización del Código

Con herramientas como **Chrome DevTools**, se depuraron errores de consola, validación DOM y trazas de llamadas fallidas a la API. El código fue modularizado para reducir duplicación, mejorando el rendimiento y la legibilidad.

Implementación de la Lógica de Negocio en el Frontend

Migración de la Lógica de Negocio desde el Backend (si necesario)

Algunas validaciones, como que el campo "edad del estudiante" esté completo, fueron replicadas en el frontend para evitar peticiones innecesarias. Esto agiliza la interacción y disminuye la carga del servidor. También, a la hora de agregar el prompt general del asistente, para que creara las actividades, se colocó el prompt general en el backend, para que no fuese repetido cada que el cliente realizará una petición.

Validación de Datos y Reglas de Negocio en el Frontend

Las reglas implementadas incluyen:

- Restricción de campos según el rol docente vs. administrador.
- Evitar generar actividad sin al menos un objetivo definido.
- Lógica para impedir duplicación de actividades iguales.

Integración con el Backend

Verificación de la Comunicación Efectiva con el Backend

Se realizaron pruebas de flujo completo:

1. Login.
2. Consulta de historial.
3. Generación y guardado de actividad.
4. Modificación y descarga.

Se verificó que los tokens JWT se gestionan correctamente y que los datos persistidos en MongoDB se reflejan en la interfaz.

Pruebas de Integración Frontend-Backend

Estas pruebas simulan el uso completo del sistema:

- Login → Generación → Guardado → Visualización → Descarga.
Se usaron datos de prueba para comprobar la coherencia entre lo que el usuario genera y lo que se almacena.