Tutorías inteligentes: un enfoque personalizado para la educación universitaria.

Índice:

[**Índice** 0](#_Toc210168999)

[Iteración inicial: 1](#_Toc210169000)

[Diagramas y utilidades 1](#_Toc210169001)

[Introducción 2](#_Toc210169002)

[Propósito del sistema 2](#_Toc210169003)

[Alcance del sistema 2](#_Toc210169004)

[✅Requerimientos 2](#_Toc210169005)

[Funcionales 2](#_Toc210169006)

[No funcionales: 3](#_Toc210169007)

[Requisitos técnicos: 3](#_Toc210169008)

[Consideraciones Futuras: 3](#_Toc210169009)

[🔍 Análisis 3](#_Toc210169010)

[Análisis del problema 3](#_Toc210169011)

[Usuarios y actores involucrados 4](#_Toc210169012)

[Requisitos informativos 4](#_Toc210169013)

[Requisitos de procesamiento 4](#_Toc210169014)

[Análisis de viabilidad 4](#_Toc210169015)

[Casos de Uso: 5](#_Toc210169016)

[Descripcion de casos de uso: 5](#_Toc210169017)

[Diagramas Business Process Model and Notation (BPMN): 6](#_Toc210169018)

[Diseño: 7](#_Toc210169019)

[Primera iteración: 7](#_Toc210169020)

[Implementación: 8](#_Toc210169021)

[Testing: 8](#_Toc210169022)

[Segunda iteración: 9](#_Toc210169023)

[🔍 Análisis: 9](#_Toc210169024)

[Diagrama de Secuencia: 9](#_Toc210169025)

[Descripcion de casos de uso: 10](#_Toc210169026)

[Diagrama de despliegue: 10](#_Toc210169027)

[Arquitectura: 13](#_Toc210169028)

[¿Por qué utiliza arquitectura Cliente-Servidor? 13](#_Toc210169029)

[¿Qué API utiliza? 13](#_Toc210169030)

[¿Cómo funciona el sistema? 13](#_Toc210169031)

[Referencias de la Api 14](#_Toc210169032)

[Aclaraciones: 14](#_Toc210169033)

[Notas 15](#_Toc210169034)

[Tercera iteración: 15](#_Toc210169035)

[🔍 Análisis: 15](#_Toc210169036)

[Objetivo de la Iteración: 15](#_Toc210169037)

[Cambios Principales: 16](#_Toc210169038)

[Justificación del Cambio: 16](#_Toc210169039)

[Consideraciones Técnicas: 16](#_Toc210169040)

[✅Requerimientos: 17](#_Toc210169041)

[Funcionales 17](#_Toc210169042)

[No Funcionales 17](#_Toc210169043)

[Referencias de la API (Actualizadas): 17](#_Toc210169044)

[LISTA DE ENDPOINTS: 17](#_Toc210169045)

[Implementación: 19](#_Toc210169046)

[Cuarta iteración: 19](#_Toc210169047)

[✅Requerimientos 19](#_Toc210169048)

[No funcionales 19](#_Toc210169049)

[Quinta iteración: 20](#_Toc210169050)

[Posibles problemas de seguridad 20](#_Toc210169051)

[1) Secretos expuestos o mal gestionados 20](#_Toc210169052)

[2) Cookies/sesiones débiles y CSRF 20](#_Toc210169053)

[3) Sin rate limiting (login/chat) y abuso de recursos 20](#_Toc210169054)

[4) Cabeceras de seguridad y CSP ausentes 21](#_Toc210169055)

[5) Validación de entradas y manejo de PDFs 21](#_Toc210169056)

[6) Control de acceso/admin por email hardcode (RBAC) 21](#_Toc210169057)

[Sexta iteración: 22](#_Toc210169058)

[Plan de Implementación y Entrega 22](#_Toc210169059)

[Septima iteración: 24](#_Toc210169060)

[Diagrama de clases: 24](#_Toc210169061)

[Diagrama de comunicación 1](#_Toc210169062)

[Documentación del usuario Administrador 1](#_Toc210169063)

# Iteración inicial:

## Diagramas y utilidades

* Casos de Uso: [Diagrama de Casos de Uso](file:///C:\Users\felip\Downloads\Diagrama%20de%20Casos%20de%20Uso)
* Secuencia: [Diagrama de Secuencia](file:///C:\Users\felip\Downloads\Diagrama%20de%20Secuencia)
* Despliegue: [Diagrama de Despliegue](file:///C:\Users\felip\Downloads\Diagrama%20de%20Despliegue)
* BPMN: [Diagramas BPMN](file:///C:\Users\felip\Downloads\Diagramas%20BPMN)
* Descripción de casos de uso: [Descripción de Casos de Uso](file:///C:\Users\felip\Downloads\Descripcion%20de%20Casos%20de%20Uso)
* Hojas de testing: [TestCaseAV](file:///C:\Users\felip\Downloads\TestCaseAV)

## Introducción

El **Asistente Virtual PDF** es una herramienta web interactiva diseñada para facilitar la comprensión y el estudio de documentos en formato PDF, especialmente en el ámbito de la atención primaria de la salud. A través de una interfaz intuitiva, permite a los usuarios cargar apuntes, libros o artículos en PDF y realizar preguntas en lenguaje natural sobre el contenido de dichos documentos. El sistema utiliza inteligencia artificial para analizar el texto y proporcionar respuestas claras y contextualizadas, ayudando así a estudiantes, docentes y profesionales a resolver dudas, repasar conceptos y navegar textos complejos de manera eficiente.

## Propósito del sistema

El propósito principal del Asistente Virtual PDF es brindar un apoyo inteligente y personalizado a quienes necesitan comprender información contenida en documentos PDF. El sistema está orientado a responder preguntas específicas sobre el contenido cargado, permitiendo que el usuario obtenga explicaciones, resúmenes o aclaraciones sin necesidad de buscar manualmente en el texto. De esta manera, se busca optimizar el tiempo de estudio, mejorar la retención de conocimientos y facilitar el acceso a la información relevante, especialmente en contextos educativos y profesionales.

## Alcance del sistema

Esta primera versión del Asistente Virtual PDF permite a los usuarios cargar archivos PDF desde su dispositivo, visualizar el contenido de los documentos y realizar preguntas sobre el texto cargado. El sistema extrae automáticamente el contenido textual del PDF y lo utiliza como contexto para que la inteligencia artificial pueda generar respuestas precisas y adaptadas a cada consulta. El alcance actual incluye la interacción básica con el asistente, la navegación por el documento y la obtención de respuestas en tiempo real.

## ✅Requerimientos

#### Funcionales**:**

1. El sistema debe poder **Mostrar contenido de la catedra** (infografías o videos y artículos relevantes.)
2. El sistema debe permitir **Descargar contenido de la catedra** contenido médico o de la catedra.
3. El sistema debe poder **Mostrar exámenes de autoevaluación.**
4. El sistema debe permitir al usuario **Cargar apuntes** con contenido médico o de la catedra.
5. El sistema debe poder **Mostrar apunte cargado** mientras interactúa con el chatbot.
6. El sistema debe poder **Mostrar otro apunte** **cargado**.
7. El sistema debe poder **Registrar una pregunta** mediante un asistente virtual (chatbot).
8. El sistema debe poder **Mostrar la respuesta del asistente.**

#### No funcionales:

1. El sistema debe funcionar en navegadores modernos (Chrome, Firefox, Edge).
2. El sistema debe ser **accesible** desde computadoras y dispositivos móviles.
3. El procesamiento de datos debe ser **rápido y seguro**.
4. El chatbot debe responder en **lenguaje natural y comprensible**.
5. El chatbot debe respetar **límites de contexto** para evitar respuestas fuera de tema.
6. Se debe priorizar el uso de **tecnología gratuita u open source** (por ejemplo, modelos gratuitos en OpenRouter).
7. El sistema debe tener una **interfaz web amigable**.

#### Requisitos técnicos:

1. Soporte para archivos .pdf.
2. Framework web: HTML, CSS, JavaScript.
3. Backend opcional: Node.js / Python (Flask o FastAPI si hace falta procesar los PDFs).
4. Uso de APIs como OpenAI, OpenRouter u otra que permita embeddings o LLMs.

## Consideraciones Futuras:

En futuras versiones del Asistente Virtual PDF se planea incorporar nuevas funcionalidades para enriquecer la experiencia del usuario. Entre las mejoras previstas se encuentran la posibilidad de guardar el historial de preguntas y respuestas, implementar autenticación y perfiles de usuario, permitir la descarga de respuestas o resúmenes generados, y ofrecer soporte para otros formatos de documentos además de PDF. También se contempla optimizar la precisión de las respuestas mediante el uso de modelos de inteligencia artificial más avanzados y agregar herramientas de accesibilidad para usuarios con necesidades especiales. Estas mejoras buscan hacer del sistema una herramienta aún más completa y versátil para el estudio y la consulta de información en documentos digitales.

## 🔍 Análisis

### **Análisis del problema**

Actualmente, los estudiantes de medicina deben estudiar apuntes extensos, muchas veces en formato PDF, que contienen terminología compleja y fragmentos densos de información. El problema principal es la **dificultad para encontrar información puntual** **o entenderla** dentro de los textos y la **falta de una guía interactiva** que les ayude a comprender los temas en tiempo real.

Por eso, se busca desarrollar una solución basada en inteligencia artificial que permita:

* Visualizar apuntes en PDF.
* Descargar apuntes en PDF.
* Cargar apuntes en PDF.
* Consultar esos apuntes mediante preguntas.
* Recibir respuestas claras, precisas y contextuales.

#### Usuarios y actores involucrados

* **Estudiantes de medicina:** usuarios finales, consultan los apuntes mediante el chatbot.
* **Docentes:** pueden validar las respuestas y usar el sistema como apoyo didáctico.
* **Equipo de desarrollo:** diseñadores, programadores y testers.
* **Sistema de IA externo (LLM):** modelo que genera las respuestas basadas en los textos cargados.
* **Plataforma de ejecución (navegador):** donde el usuario interactúa con el sistema.

#### Requisitos informativos

* El sistema debe poder acceder al contenido completo del PDF.
* Es necesario implementar algún método para **buscar o recuperar información de modo de tener un historial en la conversación**.
* Se debe garantizar que el chatbot **no invente información**.

#### Requisitos de procesamiento

* Extracción del texto desde PDFs (incluso con imágenes, tablas o estructuras complejas).
* Envío del contenido relevante al modelo de lenguaje con el prompt del usuario.

#### Análisis de viabilidad

* **Técnica:** existen herramientas para la extracción de texto (pdf.js, pdfplumber) y APIs accesibles para IA (OpenRouter, Hugging Face).
* **Económica:** si se usa infraestructura gratuita, no hay costos elevados. Ideal para universidades o proyectos académicos.
* **Temporal:** puede desarrollarse en etapas. La versión mínima viable (MVP) podría estar lista en pocas semanas.
* **Legal/Ética:** se debe verificar que los contenidos de los PDFs no infrinjan derechos de autor si se usa en entornos públicos o compartidos.

#### Casos de Uso:

ARCHIVO ADJUNTO “Diagrama Casos de uso.mdj”: [Diagrama de Casos de Uso](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Diagrama%20de%20Casos%20de%20Uso)

**Diagrama

El contenido generado por IA puede ser incorrecto.**

#### Descripcion de casos de uso:

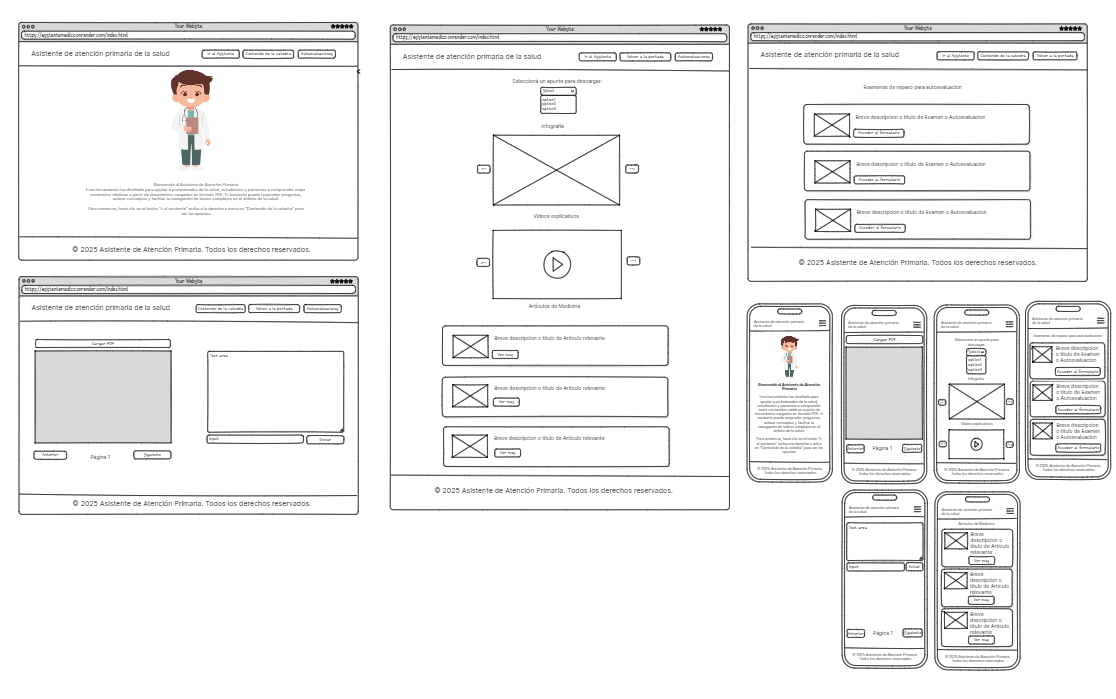
ARCHIVO ADJUNTO “Descripcion\_Casos\_Uso.docx”: [Descripción de Casos de Uso](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Descripcion%20de%20Casos%20de%20Uso)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Registrar pregunta | | | | |
| **ID: 1** | | | **Fecha:** | |
| **Descripción**: Registrar una pregunta y enviarla | | | | |
| **Actores Principales**: Persona | | **Actores Secundarios**: Asistente virtual | | |
| **Observaciones:** | | | | |
| **Precondiciones**: - | | | | |
| **Post- Condiciones** | **Éxito:** Se registra una nueva pregunta con estado “Answered” o “Error” | | | |
| **Fracaso:** No seregistra una nueva pregunta con estado “Answered” o “Error” | | | |
| Flujo Principal | | | | Flujo Alternativo |
| 1. **El caso de uso inicia cuando** la **Persona** accede al sitio web para realizar una consulta. | | | |  |
| 2. La **Persona** carga un apunte para realizar una pregunta al mismo. | | | |  |
| 3. La **Persona** escribe el prompt para realizar una **pregunta**. | | | |  |
| 4. Se verifica si el archivo está cargado correctamente y lo está. | | | | 4.1 Se verifica si el archivo está cargado correctamente y no lo está.  4.2 La persona escribe un nuevo prompt.  4.3 Se sigue con el paso 5. |
| 5. Se envía la pregunta a el **Asistente Virtual.** | | | |  |
| 6. La consulta es exitosa. | | | | 6.1 La consulta no es exitosa.  6.2 Se instancia una nueva pregunta en estado “Error" y se muestra en el historial de consultas.  6.3 Se sigue con el paso 9. |
| 7. Para gestionar la pregunta, incluir al caso de uso “Mostrar respuesta del Asistente” | | | |  |
| 8. Se instancia un nuevo pedido en estado “**Answered**” y se muestra en el historial de consultas. | | | |  |
| 9. Finaliza el caso de uso. | | | |  |

#### Diagramas Business Process Model and Notation (BPMN):

ARCHIVOS ADJUNTOS: “DBPMN 1-8.BPMN”: [Diagramas BPMN](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Diagramas%20BPMN)

## Diseño:

ARCHIVO ADJUNTO: “PROTOTIPO.F0”

# Primera iteración:

Durante la primera iteración del desarrollo del Asistente Virtual PDF se avanzó en la consolidación de las funcionalidades principales y en la incorporación de recursos complementarios para enriquecer la experiencia del usuario. A continuación, se detallan los principales logros y tareas realizadas en esta etapa:

**1. Mejora de la interfaz de usuario**

Se optimizó la interfaz web para hacerla más intuitiva y atractiva. Se reorganizaron los elementos visuales, se mejoró la navegación entre páginas del PDF y se agregaron indicadores visuales para facilitar la orientación del usuario dentro del documento.

**2. Integración de recursos multimedia y artículos**

Se incorporó una sección de recursos adicionales, accesible desde la interfaz, que incluye videos informativos, artículos relevantes y material de apoyo sobre atención primaria de la salud. Estos recursos permiten al usuario complementar la información obtenida a través del asistente virtual.

**3. Implementación de autoevaluaciones**

Se desarrolló una página de autoevaluación donde los usuarios pueden poner a prueba sus conocimientos mediante preguntas interactivas relacionadas con los temas abordados en los documentos PDF y los recursos multimedia.

**4. Optimización del procesamiento de PDFs**

Se mejoró el proceso de carga y extracción de texto de los archivos PDF, asegurando una mayor compatibilidad con diferentes tipos de documentos y una extracción de texto más precisa para su análisis por parte de la inteligencia artificial.

**5. Refuerzo de la comunicación con la API de IA**

Se revisó y optimizó la integración con la API externa de inteligencia artificial, mejorando la gestión de errores y la presentación de las respuestas generadas por el asistente virtual.

**6. Pruebas de usabilidad y ajustes**

Se realizaron pruebas de usabilidad con usuarios reales para identificar posibles mejoras en la experiencia de uso. A partir de los resultados, se ajustaron detalles en la navegación, la presentación de respuestas y la accesibilidad de los recursos.

## Implementación:

Código fuente:

<https://github.com/FelipeDiazAimar/TutorInteligente>

Enlace a la página web:

<https://asistentemedico.onrender.com/>

## Testing:

ARCHIVO ADJUNTO “TestCaseAV.XSLX”: [TestCaseAV](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\TestCaseAV)

# Segunda iteración:

## 🔍 Análisis:

#### Diagrama de Secuencia: [Diagrama de Secuencia](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Diagrama%20de%20Secuencia)

Tabla

El contenido generado por IA puede ser incorrecto.

La **representación gráfica** del flujo del diagrama de secuencia es visible usando notación textual UML (puedes copiarlo en [PlantUML](https://plantuml.com/es/sequence-diagram) para generar el diagrama visual):

*@startuml*

*actor Usuario*

*boundary "Interfaz Web (UI)" as UI*

*control "Servidor Node.js" as Servidor*

*entity "PDF Seleccionado" as PDF*

*entity "API IA (OpenRouter)" as IA*

*Usuario -> UI: Carga PDF y visualiza contenido*

*Usuario -> UI: Escribe pregunta sobre el PDF*

*UI -> Servidor: Envía pregunta y texto del PDF (contexto)*

*Servidor -> IA: Solicita respuesta IA (con pregunta y contexto)*

*IA --> Servidor: Devuelve respuesta generada*

*Servidor --> UI: Envía respuesta al usuario*

*UI --> Usuario: Muestra respuesta en el chat*

*@enduml*

#### Descripcion de casos de uso:

ARCHIVO ADJUNTO “Descripcion\_Casos\_Uso.docx”: [Descripción de Casos de Uso](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Descripcion%20de%20Casos%20de%20Uso)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caso de Uso**: Cargar Apunte | | | | |
| **ID: 1** | | | **Fecha:** | |
| **Descripción**: Cargar un apunte | | | | |
| **Actores Principales**: Persona | | **Actores Secundarios**: | | |
| **Observaciones:** | | | | |
| **Precondiciones**: - | | | | |
| **Post- Condiciones** | **Éxito:** Se carga un nuevo apunte | | | |
| **Fracaso:** No secarga un nuevo apunte | | | |
| Flujo Principal | | | | Flujo Alternativo |
| 1. **El caso de uso inicia cuando** la **Persona** accede al sitio web para cargar un nuevo apunte. | | | |  |
| 2. La **Persona** hace “click” en cargar un Apunte. | | | |  |
| 3. La **Persona** selecciona un archivo a cargar. | | | |  |
| 5. Se verifica si el apunte está cargado en formato PDF y lo está**.** | | | | 5.1 Se verifica si el apunte está cargado en formato PDF y no lo está**.**  5.2 Finaliza el caso de uso. |
| 6. Se verifica si hay un archivo cargado previamente y lo está. | | | | 6.1 Se verifica si hay un archivo cargado previamente y no lo está.  6.2 Se extiende al caso de uso “Mostrar otro apunte cargado”  6.3 Se sigue con el paso 7. |
| 7. Para gestionar el apunte, incluir al caso de uso “Mostrar apunte cargado” | | | |  |
| 8. Finaliza el caso de uso. | | | |  |

#### Diagrama de despliegue:

[Diagrama de Despliegue](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Diagrama%20de%20Despliegue)

Diagrama

El contenido generado por IA puede ser incorrecto.

La **representación gráfica** diagrama de despliegue es visible usando notación textual UML (puedes copiarlo en [PlantUML](https://plantuml.com/es/sequence-diagram) para generar el diagrama visual):

*@startuml*

*' Nodos principales*

*node "Usuario" as usuario*

*node "Navegador Web" as navegador {*

*artifact "index.html"*

*artifact "script.js"*

*artifact "style.css"*

*artifact "PDF.js"*

*}*

*node "Servidor Node.js" as servidor {*

*artifact "server.js"*

*artifact "public/"*

*}*

*node "API de IA\n(OpenRouter)" as ia*

*' Relaciones*

*usuario --> navegador : Interacción\n(clic, carga PDF,\npregunta)*

*navegador --> servidor : Solicitud HTTP\n(GET/POST)*

*servidor --> ia : Solicitud a API IA\n(pregunta + contexto)*

*ia --> servidor : Respuesta IA*

*servidor --> navegador : Respuesta JSON\n+ archivos estáticos*

*' Recursos estáticos*

*folder "public/" {*

*artifact "pdfs/"*

*artifact "videos/"*

*artifact "img/"*

*}*

*servidor --> "public/" : Sirve recursos*

*@enduml*

## Arquitectura:

**Arquitectura utilizada en mi proyecto: Cliente-Servidor**

Mi proyecto de asistente virtual inteligente implementa la **arquitectura Cliente-Servidor**, un estilo arquitectónico distribuido ampliamente utilizado en el desarrollo de aplicaciones web. En esta arquitectura, existen dos componentes principales: el **cliente** y el **servidor**.

#### ¿Por qué utiliza arquitectura Cliente-Servidor?

En mi proyecto, el **cliente** está representado por la interfaz web (archivos HTML, CSS y JavaScript ubicados en la carpeta public/), que es la encargada de interactuar con el usuario final. El **servidor** está implementado en Node.js (archivo [server.js](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)), y es responsable de procesar las solicitudes del cliente, gestionar la lógica de negocio y comunicarse con servicios externos.

Esta separación permite que el cliente se enfoque únicamente en la presentación y la interacción con el usuario, mientras que el servidor centraliza la lógica, el procesamiento y el acceso a los datos. Además, ambos componentes pueden desarrollarse y actualizarse de forma independiente, facilitando el mantenimiento y la escalabilidad del sistema.

#### ¿Qué API utiliza?

El servidor de mi proyecto utiliza una **API externa de inteligencia artificial** (OpenRouter) para procesar las preguntas que envía el usuario desde la interfaz web. Cuando el usuario realiza una consulta, el cliente envía una petición HTTP al servidor, el cual a su vez se comunica con la API de OpenRouter para obtener una respuesta inteligente.

#### ¿Cómo funciona el sistema?

1. **El usuario** interactúa con la página web y envía una pregunta.
2. **El cliente** (navegador) envía la pregunta al servidor mediante una solicitud HTTP.
3. **El servidor** recibe la solicitud, procesa la pregunta y la reenvía a la API de OpenRouter.
4. **La API de OpenRouter** responde con una respuesta generada por inteligencia artificial.
5. **El servidor** recibe la respuesta de la API y la envía de vuelta al cliente.
6. **El cliente** muestra la respuesta al usuario en la interfaz web.

Este flujo de trabajo refleja claramente la arquitectura Cliente-Servidor, donde el cliente depende del servidor para acceder a la funcionalidad principal y el servidor centraliza la lógica y el acceso a servicios externos.

## Referencias de la Api

#### Aclaraciones:

* **Obligatorio de enviar, se ingresa manualmente:** pregunta (string)
* **Obligatorio de enviar, se ingresa automáticamente, pero se puede modificar:** contexto (texto extraído del PDF)
* **Obligatorio de enviar, se ingresa automáticamente y no se puede modificar:** N/A

**POST /ask**

Envía una pregunta sobre el contenido de un PDF y recibe una respuesta generada por IA.

**Parámetros de entrada (body JSON):**

| **Atributo** | **Tipo** | **Descripción** | **Obligatorio** |
| --- | --- | --- | --- |
| question | string | Pregunta que realiza el usuario sobre el PDF | Sí |
| context | string | Texto extraído del PDF seleccionado (contexto para la IA) | Sí |

**Ejemplo de solicitud**

*{*

*"question": "¿Cuál es el objetivo principal de la atención primaria de la salud?",*

*"context": "La atención primaria de la salud es un enfoque integral..."*

*}*

**Respuestas:**

* **200 OK**
  + Respuesta exitosa con la contestación generada por la IA.
  + Atributos:
    - [answer](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): string – Respuesta generada por el asistente virtual.

**Ejemplo:**

*{*

*"answer": "El objetivo principal de la atención primaria de la salud es brindar atención accesible, integral y continua a toda la población."*

*}*

*{*

*"question": "¿Cuál es el objetivo principal de la atención primaria de la salud?",*

*"context": "La atención primaria de la salud es un enfoque integral..."*

*}*

*{*

*"question": "¿Cuál es el objetivo principal de la atención primaria de la salud?",*

*"context": "La atención primaria de la salud es un enfoque integral..."*

*}*

*{*

*"question": "¿Cuál es el objetivo principal de la atención primaria de la salud?",*

*"context": "La atención primaria de la salud es un enfoque integral..."*

*}*

* **400 Bad Request**
  + **Faltan parámetros obligatorios o el formato es incorrecto.**

**Ejemplo:**

{

*"error": "Faltan parámetros obligatorios."*

*}*

* **500 Internal Server Error**
  + **Error interno del servidor o de la API externa.**

**Ejemplo:**

*{*

*"answer": "Error del servidor"*

*}*

#### Notas

* El endpoint /ask es el único expuesto para interacción con el asistente virtual.
* El contexto debe ser el texto extraído del PDF cargado por el usuario.
* La respuesta es generada por un modelo de IA externo (OpenRouter).

# Tercera iteración:

## 🔍 Análisis:

### Objetivo de la Iteración:

Transformar el proyecto de una aplicación estática (HTML, CSS, JS) a una aplicación dinámica con backend real, utilizando Supabase como base de datos y almacenamiento, y modernizando la interfaz y la estructura del código.

***A partir de esta iteración, el proyecto evoluciona a una aplicación dinámica y escalable, con backend real y gestión avanzada de usuarios y recursos.***

### Cambios Principales:

* Base de datos:  
  Ahora el proyecto utiliza Supabase (PostgreSQL) para almacenar usuarios, recursos educativos, sesiones, etc.
* Backend y autenticación:  
  Implementación de autenticación con JWT y manejo de sesiones, inspirada en ejemplos de Firebase pero adaptada a Supabase.
* **Frontend moderno:**  
  Migración a Next.js + TypeScript, uso de TailwindCSS y componentes Radix UI.
* **Gestión dinámica de recursos:**  
  Los recursos (PDFs, imágenes, videos, artículos) ya no están hardcodeados, sino que se cargan dinámicamente desde la base de datos.
* **Infraestructura y scripts:**  
  Nuevos scripts para setup de la base de datos, hashing de contraseñas, y limpieza de sesiones.
* **Middleware de autenticación:**  
  Protección de rutas administrativas y manejo de permisos.

### Justificación del Cambio:

* **Escalabilidad:**  
  Permite agregar, editar y eliminar recursos sin modificar el código fuente.
* **Seguridad**:  
  Mejor manejo de usuarios y sesiones.
* **Mantenibilidad**:  
  Separación clara entre frontend, backend y base de datos.
* **Experiencia de usuario:**  
  Interfaz más moderna y responsiva.

### Consideraciones Técnicas:

* Variables de entorno en [.env.local](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para credenciales y configuración.
* Uso de Supabase para autenticación y almacenamiento.
* Estructura de carpetas organizada por componentes, hooks, utilidades, etc.
* Middleware para proteger rutas sensibles.

## ✅Requerimientos:

#### Funcionales**:**

* **RF09.** Registrar y autenticar usuarios (con Supabase Auth/JWT).
* **RF10.** Gestionar sesiones de usuario (login/logout, expiración, etc.).
* **RF11.** Guardar historial de preguntas y respuestas por usuario.
* **RF12.** Permitir gestión dinámica de recursos (agregar, editar, eliminar PDFs, videos, artículos desde la base de datos, no hardcodeados).
* **RF13.** Permitir roles de usuario (admin, usuario normal) y proteger rutas administrativas.
* **RF14.** Subir y almacenar archivos en Supabase Storage.
* **RF15.** CRUD de recursos educativos (PDFs, imágenes, videos, artículos).
* **RF16.** Middleware para proteger rutas sensibles.
* **RF17.** Scripts para setup de base de datos, hashing de contraseñas, limpieza de sesiones.
* **RF18.** Integración con IA (OpenRouter, Gemini) para chat inteligente.
* **RF19.** Visualización y gestión de recursos desde la interfaz web (no solo desde el código).

#### No Funcionales**:**

* **RNF08.** Seguridad en el manejo de sesiones y datos personales (por uso de autenticación y base de datos).
* **RNF09.** Escalabilidad para soportar múltiples usuarios y recursos.
* **RNF10.** Mantenibilidad y modularidad del código (estructura clara de carpetas, componentes, hooks, etc.).
* **RNF11.** Configuración mediante variables de entorno.
* **RNF12.** Accesibilidad mejorada (por ejemplo, soporte para lectores de pantalla, navegación por teclado).
* **RNF13.** Respaldo y recuperación de datos (por uso de base de datos real).
* **RNF14.** Cumplimiento de buenas prácticas de desarrollo seguro (protección contra XSS, CSRF, etc.).

## Referencias de la API (Actualizadas):

### LISTA DE ENDPOINTS:

#### 🔐 Autenticación:

| **Endpoint** | **Método** | **Descripción** | **Autenticación** |
| --- | --- | --- | --- |
| /api/auth/login | [POST](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Iniciar sesión de profesor/admin | ❌ No |
| /api/auth/logout | [POST](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Cerrar sesión | ✅ Sí |
| /api/auth/me | [GET](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Obtener información del usuario autenticado | ✅ Sí |

#### 👥 Gestión de Profesores (Admin):

| **Endpoint** | **Método** | **Descripción** | **Autenticación** | **Campos** |
| --- | --- | --- | --- | --- |
| /api/admin/professors | [GET](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Listar profesores | ✅ Admin | - |
| /api/admin/professors | [POST](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Crear profesor | ✅ Admin | [name](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [email](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [password](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [department](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) |
| /api/admin/professors/{id} | [PUT](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Actualizar profesor | ✅ Admin | [name](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [email](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [department](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [is\_active](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) |
| /api/admin/professors/{id} | [DELETE](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Eliminar profesor | ✅ Admin | - |

#### 📚 Gestión de Recursos (Admin):

| **Endpoint** | **Método** | **Descripción** | **Autenticación** | **Campos** |
| --- | --- | --- | --- | --- |
| /api/admin/resources | [GET](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Listar recursos educativos | ✅ Admin | - |
| /api/admin/resources | [DELETE](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Eliminar recurso (por query param [id](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) | ✅ Admin | - |

#### 🔄 Gestión de Sesiones (Admin):

| **Endpoint** | **Método** | **Descripción** | **Autenticación** | **Campos** |
| --- | --- | --- | --- | --- |
| /api/admin/sessions | [GET](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Listar sesiones activas | ✅ Admin | - |
| /api/admin/sessions | [DELETE](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Cerrar sesión específica (por query param [id](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)) | ✅ Admin | - |
| /api/admin/sessions/cleanup | [POST](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Limpiar sesiones expiradas | ✅ Admin | - |
| /api/admin/sessions/cleanup | [GET](vscode-file://vscode-app/c:/Users/franc/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) | Obtener count de sesiones expiradas | ✅ Admin | - |

## Implementación:

Código fuente:

<https://github.com/FelipeDiazAimar/AsistenteInteligente>

Enlace a la página web:

<https://asistente-inteligente.vercel.app/>

# Cuarta iteración:

## ✅Requerimientos

#### No funcionales**:**

1. Usabilidad - Facilidad de Aprendizaje y Operabilidad

**Descripción:** El sistema debe proporcionar una interfaz intuitiva y fácil de aprender que permita a los estudiantes y profesionales de la salud utilizar todas las funcionalidades de manera efectiva sin requerir entrenamiento o conocimiento extensivo.

**Como lo podríamos implementar ?**

* Usar un diseño claro y moderno, con menús y botones bien identificados, agrupando las funciones por categorías (por ejemplo: PDFs, videos, imágenes, artículos, autoevaluación).
* Incluir textos y descripciones breves en cada sección y botón, para que el usuario sepa para qué sirve cada cosa sin tener que buscar ayuda externa.
* Mantener la navegación simple: máximo 2 o 3 clics para llegar a cualquier recurso o función importante.
* Hacer la interfaz responsive, para que se vea bien y funcione igual en computadoras, tablets y celulares.
* Agregar mensajes de ayuda y feedback inmediato (por ejemplo, toasts o alertas) cuando el usuario realiza una acción, para que sepa si fue exitosa o si hubo un error.
* Usar iconos y colores consistentes en toda la app, para que el usuario se ubique rápido y no se confunda.
* Probar la app con usuarios reales y ajustar los textos, botones y flujos según sus comentarios.

1. Security - Confidencialidad y Autenticidad

**Descripción:** El sistema debe garantizar que la información sensible y los recursos educativos estén protegidos mediante autenticación segura y control de acceso adecuado, asegurando que solo usuarios autorizados puedan acceder a funcionalidades administrativas.

# Quinta iteración:

## Posibles problemas de seguridad

### 1) Secretos expuestos o mal gestionados

* Qué puede pasar: robo de claves (API/DB/JWT), cuentas comprometidas.
* Cómo lo arreglo:
  + Mover todos los secretos a Variables de Entorno en Vercel.
  + Eliminar defaults débiles (no más || 'dev-secret-key...').
  + Evitar usar process.env.\* en componentes cliente.
  + Activar escaneo de secretos (GitHub Secret Scanning, Snyk).
* Implementación (tecnología/archivos):
  + Vercel Project Settings > Environment Variables.
  + Purgar históricos si se subieron: BFG o git-filter-repo.
  + Cambios en: [auth.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [admin-auth.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (obligatorio JWT\_SECRET).

### 2) Cookies/sesiones débiles y CSRF

* Qué puede pasar: secuestro de sesión, acciones no autorizadas desde otros sitios.
* Cómo lo arreglo:
  + Cookies: HttpOnly, Secure, SameSite=Strict, expiración corta; access token corto + refresh rotativo (hash en DB).
  + CSRF: token anti-CSRF (doble envío o header) para POST/PUT/DELETE con cookies.
  + Validar “huella” (UA/IP con tolerancia) para reducir uso de tokens robados.
* Implementación:
  + Endurecer set-cookie en [route.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) y logout/route.ts.
  + Verificación CSRF en [middleware.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) y en APIs mutantes (src/app/api/\*\*).
  + Guardar/validar refresh en [auth.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (tabla sesiones).

### 3) Sin rate limiting (login/chat) y abuso de recursos

* Qué puede pasar: fuerza bruta a login, DoS al chat/IA, costos inesperados.
* Cómo lo arreglo:
  + Rate limiting por IP y por usuario (login, chat, admin).
  + CAPTCHA progresivo cuando hay intentos fallidos.
  + Guardrails de coste: límites de longitud y concurrencia en chat, timeout y backoff.
* Implementación:
  + Upstash Redis + @upstash/ratelimit en [middleware.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) por ruta.
  + Límite de input y concurrencia en [route.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (Zod, timeouts).

### 4) Cabeceras de seguridad y CSP ausentes

* Qué puede pasar: XSS, clickjacking, sniffing, permisos de navegador excesivos.
* Cómo lo arreglo:
  + Añadir: CSP estricta, HSTS, X-Frame-Options=DENY, X-Content-Type-Options=nosniff, Referrer-Policy, Permissions-Policy mínima.
* Implementación:
  + [next.config.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) → async headers() con políticas (listar fuentes/imágenes externas que usas).
  + Validar con Mozilla Observatory / securityheaders.com.

### 5) Validación de entradas y manejo de PDFs

* Qué puede pasar: XSS, DoS por PDFs grandes/maliciosos, prompt injection al chat.
* Cómo lo arreglo:
  + Validar y truncar inputs (Zod) en APIs.
  + PDFs: validar tipo por firma, tamaño (p.ej., ≤10MB), páginas (≤N), timeouts y sandbox/aislamiento; opcional AV (ClamAV).
  + Chat: límites de caracteres; prompt “guardrails”; filtrar contenido de respuesta si es necesario.
* Implementación:
  + [pdf-service.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): límites y timeouts; si es pesado, procesarlo en función aislada/serverless.
  + [route.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Zod + límites + timeout.

### 6) Control de acceso/admin por email hardcode (RBAC)

* Qué puede pasar: acceso admin indebido si se compromete un email o no se valida bien.
* Cómo lo arreglo:
  + Roles en DB (p.ej., role = 'admin'), incluidos en el JWT y validados en cada request.
  + Middleware que bloquee /admin/\*\* si no es admin (403).
  + Auditoría básica de acciones admin (fecha, userId, acción).
* Implementación:
  + [admin-auth.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) y [auth.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): JWT con rol; validar en [middleware.ts](vscode-file://vscode-app/c:/Users/felip/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
  + Endpoints bajo /api/admin/\*\* verifican rol server-side.

# Sexta iteración:

## Plan de Implementación y Entrega

**Duración:** 30/09/2025 – 30/11/2025 (8 semanas)  
**Desarrollador:** 1 desarrollador  
**Punto de partida:** Proyecto avanzado con funcionalidades core implementadas (chat IA, CRUD recursos, autenticación, dashboard) testing, documentación y optimización.

**Semana 1 (30/09 – 05/10): Revisión y Organización**

* Revisar el código existente y documentar el estado actual del sistema.
* Listar funcionalidades ya implementadas (chat IA, gestión recursos, auth JWT) y pendientes.
* Organizar el repositorio y actualizar documentación.
* Definir los requisitos funcionales y no funcionales finales.
* Crear un backlog de tareas priorizadas y setup de herramientas de testing.  
  **Requisitos:** RT02, RNF07

**Semana 2 (06/10 – 12/10): Testing y Seguridad**

* Implementar suite de testing para componentes y APIs críticas.
* Robustecer el sistema de autenticación JWT y protección de rutas administrativas.
* Mejorar validaciones con y manejo de errores en frontend/backend.
* Implementar rate limiting y headers de seguridad (CORS, CSP).
* Documentar endpoints de autenticación y flujos de seguridad.  
  **Requisitos:** RNF03, RT03

**Semana 3 (13/10 – 19/10): CRUD y Gestión de Recursos**

* Completar y testear el CRUD de recursos (PDFs, videos, imágenes, artículos, evaluaciones).
* Optimizar queries de base de datos PostgreSQL y mejorar performance.
* Implementar validaciones robustas para upload de archivos y gestión de storage.
* Mejorar feedback de formularios y confirmaciones para acciones destructivas.
* Documentar APIs de recursos y flujos de datos principales.  
  **Requisitos:** RF01, RF02, RF04, RT01

**Semana 4 (20/10 – 26/10): Dashboard, Chat IA y Optimización**

* Optimizar el chat IA con OpenRouter y mejorar contexto de PDFs.
* Finalizar dashboard administrativo con métricas y gestión de usuarios/sesiones.
* Mejorar responsive design y accesibilidad.
* Optimizar Core Web Vitals y rendimiento general de la aplicación.  
  **Requisitos:** RF05, RF07, RF08, RT04, RNF04, RNF05

**Semana 5 (27/10 – 02/11): Features Avanzadas y UX**

* Implementar analytics básicos y reportes de uso del sistema.
* Agregar funcionalidades de export/import de recursos y datos.
* Mejorar UX/UI con loading states, error boundaries y feedback visual.
* Implementar búsquedas avanzadas y filtros en gestión de recursos.
* Completar adaptación móvil y testing cross-browser.  
  **Requisitos:** RF03, RF06, RNF02, RNF01

**Semana 6 (03/11 – 09/11): Testing E2E y QA**

* Implementar testing E2E para flujos críticos.
* Realizar testing manual exhaustivo de autenticación, chat y administración.
* Testing de performance y carga con usuarios concurrentes.
* Corrección de bugs detectados durante las pruebas.
* Validación de accesibilidad y compatibilidad móvil.  
  **Requisitos:** RNF03, RNF01, RNF02

**Semana 7 (10/11 – 16/11): Monitoreo y Documentación**

* Configurar alertas automatizadas.
* Completar documentación técnica (APIs, arquitectura, deployment).
* Crear manual de usuario y guías de administración.
* Documentar procedimientos de backup, recovery y mantenimiento.  
  **Requisitos:** RNF03, RNF06

**Semana 8 (17/11 – 30/11): Deployment y Entrega Final**

* Realizar pruebas completas de funcionalidad, usabilidad y seguridad.
* Implementar estrategia de backup automatizado y testing de recuperación.
* Preparar el entorno de producción y verificar configuraciones.
* Handover completo con documentación, credenciales y procedures de mantenimiento.  
  **Requisitos:** Validación final de todos los RF y RNF, RT.

**Distribución de Requisitos**

**Requerimientos Funcionales:**  
• RF01: Mostrar contenido de la cátedra (infografías, videos, artículos, PDFs).  
• RF02: Permitir descarga de contenido médico y recursos educativos.  
• RF03: Mostrar exámenes de autoevaluación interactivos.  
• RF04: Cargar apuntes con contenido médico o de la cátedra.  
• RF05: Mostrar apunte cargado mientras interactúa con el chatbot.  
• RF06: Permitir cambio y visualización de diferentes apuntes.  
• RF07: Registrar preguntas mediante asistente virtual (chatbot).  
• RF08: Mostrar respuestas del asistente en lenguaje natural.

**Requerimientos No Funcionales:**  
• RNF01: Compatibilidad con navegadores modernos (Chrome, Firefox, Edge).  
• RNF02: Accesibilidad desde computadoras y dispositivos móviles.  
• RNF03: Procesamiento de datos rápido y seguro.  
• RNF04: Respuestas del chatbot en lenguaje natural y comprensible.  
• RNF05: Respeto de límites de contexto para evitar respuestas fuera de tema.  
• RNF06: Uso de tecnología gratuita u open source (OpenRouter).  
• RNF07: Interfaz web amigable e intuitiva.

**Requerimientos Técnicos:**  
• RT01: Soporte para archivos PDF con extracción de texto.  
• RT02: Framework web moderno (Next.js, TypeScript, Tailwind CSS).  
• RT03: Backend con Node.js para procesamiento de PDFs y autenticación.  
• RT04: Integración con APIs de LLM (OpenRouter, OpenAI) para el chatbot.

# Septima iteración:

## Diagrama de clases:

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

La **representación gráfica** diagrama de despliegue es visible usando notación textual UML (puedes copiarlo en [PlantUML](https://plantuml.com/es/sequence-diagram) para generar el diagrama visual):

@startuml

entity "professors" as professors {

\* id : uuid <<PK>>

--

name : varchar

email : varchar

password\_hash : varchar

department : varchar

is\_active : bool

created\_at : timestamptz

updated\_at : timestamptz

}

entity "professor\_sessions" as professor\_sessions {

\* id : uuid <<PK>>

--

professor\_id : uuid <<FK>>

session\_token : varchar

expires\_at : timestamptz

created\_at : timestamptz

}

entity "resources" as resources {

\* id : int4 <<PK>>

--

title : text

description : text

type : text

link : text

file\_url : text

file\_name : text

cover\_image\_url : text

cover\_image\_name : text

ai\_hint : text

google\_form\_url : text

created\_at : timestamp

}

' Relaciones

professors ||--o{ professor\_sessions : "1 a muchos"

@enduml

## Diagrama de comunicación

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

La **representación gráfica** diagrama de despliegue es visible usando notación textual UML (puedes copiarlo en [PlantUML](https://plantuml.com/es/sequence-diagram) para generar el diagrama visual):

@startuml DiagramaComunicacion

!theme plain

skinparam sequenceMessageAlign center

participant "Usuario" as User

participant "Aplicación Web" as App

participant "API" as API

participant "Base de Datos" as DB

participant "IA (OpenRouter)" as AI

== Autenticación ==

User -> App : 1: Ir a login

App -> API : 2: Verificar sesión

API -> DB : 3: Consultar profesor

DB -> API : 4: Datos del profesor

API -> App : 5: Estado de autenticación

App -> User : 6: Mostrar formulario

User -> App : 7: Enviar credenciales

App -> API : 8: POST /login

API -> DB : 9: Validar usuario

DB -> API : 10: Confirmar usuario

API -> App : 11: Token de sesión

App -> User : 12: Redirigir a dashboard

== Chat con IA ==

User -> App : 13: Subir PDF + pregunta

App -> API : 14: Procesar chat

API -> API : 15: Extraer texto del PDF

API -> AI : 16: Enviar pregunta + contexto

AI -> API : 17: Respuesta de IA

API -> App : 18: Respuesta procesada

App -> User : 19: Mostrar respuesta

== Panel Administrativo ==

User -> App : 20: Acceder a admin

App -> API : 21: Verificar permisos

API -> DB : 22: Consultar admin

DB -> API : 23: Confirmar admin

API -> App : 24: Acceso permitido

App -> User : 25: Mostrar dashboard

== Gestión de Recursos ==

User -> App : 26: Administrar recursos

App -> API : 27: CRUD recursos

API -> DB : 28: Operaciones en BD

DB -> API : 29: Resultado operación

API -> App : 30: Confirmar cambios

App -> User : 31: Actualizar interfaz

== Cerrar Sesión ==

User -> App : 32: Hacer logout

App -> API : 33: Eliminar sesión

API -> DB : 34: Borrar token

DB -> API : 35: Confirmar borrado

API -> App : 36: Sesión cerrada

App -> User : 37: Redirigir a login

note over User, AI

Comunicación Principal:

• Autenticación con JWT

• Chat con procesamiento de PDF

• Panel admin con permisos

• Gestión de recursos educativos

• Control de sesiones

end note

@enduml

## Documentación del usuario Administrador

**Acceso al Panel Administrativo**

**1. Inicio de Sesión**

* URL: https://asistente-inteligente.vercel.app/admin/login
* Credenciales de administrador:
  + Email: admin1@admin1.com o admin@university.edu
  + Contraseña: La configurada en el sistema

**2. Dashboard Principal**

* URL: https:// asistente-inteligente.vercel.app/admin/dashboard
* Vista general del sistema con 3 secciones principales

**Funcionalidades Administrativas**

**Gestión de Profesores**

* **Ver profesores**: Lista completa de usuarios registrados
* **Crear profesor**: Agregar nuevos usuarios al sistema
* **Editar profesor**: Modificar datos (nombre, email, departamento)
* **Activar/Desactivar**: Control de acceso de usuarios
* **Eliminar profesor**: Remover usuarios del sistema

**Gestión de Recursos Educativos**

* **Ver recursos**: Lista de PDFs, videos, artículos e imágenes
* **Agregar recurso**: Subir nuevo contenido educativo
* **Editar recurso**: Modificar título, descripción, tipo
* **Eliminar recurso**: Remover contenido del sistema
* **Tipos soportados**: PDF, Video, Artículo, Imagen, Evaluación

**Gestión de Sesiones**

* **Ver sesiones activas**: Lista de usuarios conectados
* **Información de sesión**: IP, tiempo de conexión, expiración
* **Limpiar sesiones expiradas**: Mantenimiento automático
* **Cerrar sesión específica**: Forzar logout de usuarios

**Navegación del Sistema**

**Menú Principal:**

* 🏠 Inicio: Chat principal con IA
* 📚 Recursos: Biblioteca de materiales
* 📝 Autoevaluación: Exámenes interactivos
* ⚙️ Admin: Panel administrativo (solo admins)

**EXPLICACIÓN DEL DIAGRAMA DE COMUNICACIÓN**

**Flujos de Comunicación**

**1. Autenticación (Pasos 1-12)**

* El usuario accede al login
* La aplicación verifica si hay sesión activa
* Se validan credenciales contra la base de datos
* Se genera token JWT y se redirige al dashboard

**2. Chat con IA (Pasos 13-19)**

* Usuario sube PDF y hace pregunta
* Sistema extrae texto del PDF usando pdf-parse
* Se envía contexto y pregunta a OpenRouter/Mistral
* IA procesa y devuelve respuesta en lenguaje natural

**3. Panel Administrativo (Pasos 20-25)**

* Verificación de permisos de administrador
* Consulta a base de datos para confirmar rol admin
* Acceso concedido al dashboard administrativo

**4. Gestión de Recursos (Pasos 26-31)**

* Operaciones CRUD en tabla de recursos
* Validación de permisos administrativos
* Actualización de interfaz en tiempo real

**5. Cerrar Sesión (Pasos 32-37)**

* Eliminación de token de sesión
* Limpieza de cookies de autenticación
* Redirección a página de login

**Componentes del Sistema**

**Frontend (Next.js)**

* Interfaz de usuario responsive
* Componentes React con TypeScript
* Autenticación con Context API

**Backend (API Routes)**

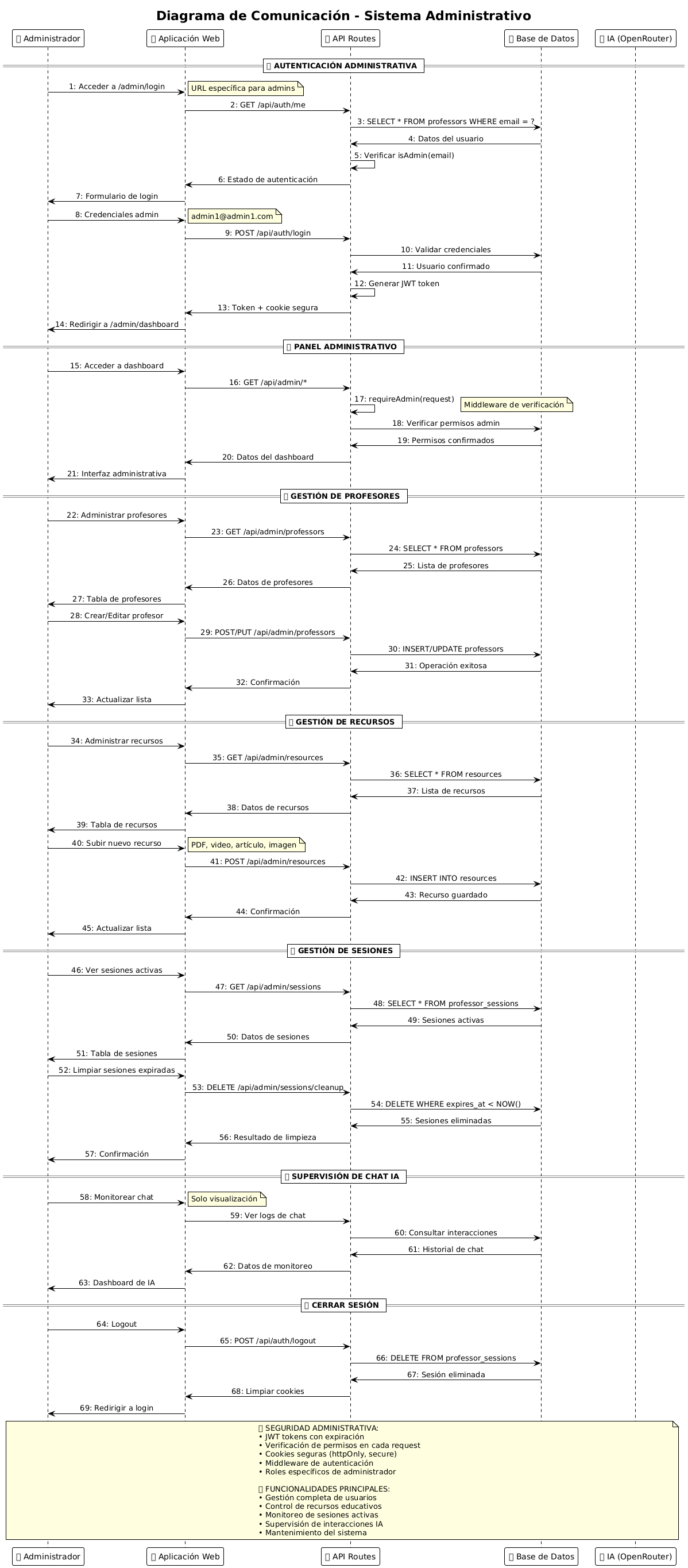
* Endpoints RESTful para todas las operaciones
* Middleware de autenticación JWT
* Validación de permisos administrativos

**Base de Datos (PostgreSQL)**

* Tabla professors: Usuarios del sistema
* Tabla professor\_sessions: Control de sesiones
* Tabla resources: Contenido educativo

**Servicios Externos**

* OpenRouter: API de inteligencia artificial
* Supabase: Hosting de base de datos
* Vercel: Deployment y hosting



Texto

El contenido generado por IA puede ser incorrecto.

La **representación gráfica** diagrama de despliegue es visible usando notación textual UML (puedes copiarlo en [PlantUML](https://plantuml.com/es/sequence-diagram) para generar el diagrama visual):

@startuml DiagramaComunicacionAdmin

!theme plain

skinparam sequenceMessageAlign center

skinparam note {

BackgroundColor lightyellow

BorderColor black

}

title Diagrama de Comunicación - Sistema Administrativo

participant "👤 Administrador" as Admin

participant "💻 Aplicación Web" as App

participant "🔗 API Routes" as API

participant "🗄️ Base de Datos" as DB

participant "🤖 IA (OpenRouter)" as AI

== 🔐 AUTENTICACIÓN ADMINISTRATIVA ==

Admin -> App : 1: Acceder a /admin/login

note right: URL específica para admins

App -> API : 2: GET /api/auth/me

API -> DB : 3: SELECT \* FROM professors WHERE email = ?

DB -> API : 4: Datos del usuario

API -> API : 5: Verificar isAdmin(email)

API -> App : 6: Estado de autenticación

App -> Admin : 7: Formulario de login

Admin -> App : 8: Credenciales admin

note right: admin1@admin1.com

App -> API : 9: POST /api/auth/login

API -> DB : 10: Validar credenciales

DB -> API : 11: Usuario confirmado

API -> API : 12: Generar JWT token

API -> App : 13: Token + cookie segura

App -> Admin : 14: Redirigir a /admin/dashboard

== 📊 PANEL ADMINISTRATIVO ==

Admin -> App : 15: Acceder a dashboard

App -> API : 16: GET /api/admin/\*

API -> API : 17: requireAdmin(request)

note right: Middleware de verificación

API -> DB : 18: Verificar permisos admin

DB -> API : 19: Permisos confirmados

API -> App : 20: Datos del dashboard

App -> Admin : 21: Interfaz administrativa

== 👥 GESTIÓN DE PROFESORES ==

Admin -> App : 22: Administrar profesores

App -> API : 23: GET /api/admin/professors

API -> DB : 24: SELECT \* FROM professors

DB -> API : 25: Lista de profesores

API -> App : 26: Datos de profesores

App -> Admin : 27: Tabla de profesores

Admin -> App : 28: Crear/Editar profesor

App -> API : 29: POST/PUT /api/admin/professors

API -> DB : 30: INSERT/UPDATE professors

DB -> API : 31: Operación exitosa

API -> App : 32: Confirmación

App -> Admin : 33: Actualizar lista

== 📚 GESTIÓN DE RECURSOS ==

Admin -> App : 34: Administrar recursos

App -> API : 35: GET /api/admin/resources

API -> DB : 36: SELECT \* FROM resources

DB -> API : 37: Lista de recursos

API -> App : 38: Datos de recursos

App -> Admin : 39: Tabla de recursos

Admin -> App : 40: Subir nuevo recurso

note right: PDF, video, artículo, imagen

App -> API : 41: POST /api/admin/resources

API -> DB : 42: INSERT INTO resources

DB -> API : 43: Recurso guardado

API -> App : 44: Confirmación

App -> Admin : 45: Actualizar lista

== 🔗 GESTIÓN DE SESIONES ==

Admin -> App : 46: Ver sesiones activas

App -> API : 47: GET /api/admin/sessions

API -> DB : 48: SELECT \* FROM professor\_sessions

DB -> API : 49: Sesiones activas

API -> App : 50: Datos de sesiones

App -> Admin : 51: Tabla de sesiones

Admin -> App : 52: Limpiar sesiones expiradas

App -> API : 53: DELETE /api/admin/sessions/cleanup

API -> DB : 54: DELETE WHERE expires\_at < NOW()

DB -> API : 55: Sesiones eliminadas

API -> App : 56: Resultado de limpieza

App -> Admin : 57: Confirmación

== 🤖 SUPERVISIÓN DE CHAT IA ==

Admin -> App : 58: Monitorear chat

note right: Solo visualización

App -> API : 59: Ver logs de chat

API -> DB : 60: Consultar interacciones

DB -> API : 61: Historial de chat

API -> App : 62: Datos de monitoreo

App -> Admin : 63: Dashboard de IA

== 🚪 CERRAR SESIÓN ==

Admin -> App : 64: Logout

App -> API : 65: POST /api/auth/logout

API -> DB : 66: DELETE FROM professor\_sessions

DB -> API : 67: Sesión eliminada

API -> App : 68: Limpiar cookies

App -> Admin : 69: Redirigir a login

note over Admin, AI

🔒 SEGURIDAD ADMINISTRATIVA:

• JWT tokens con expiración

• Verificación de permisos en cada request

• Cookies seguras (httpOnly, secure)

• Middleware de autenticación

• Roles específicos de administrador

📋 FUNCIONALIDADES PRINCIPALES:

• Gestión completa de usuarios

• Control de recursos educativos

• Monitoreo de sesiones activas

• Supervisión de interacciones IA

• Mantenimiento del sistema

end note

@enduml