Tutorías inteligentes: un enfoque personalizado para la educación universitaria.

Índice:

[Índice 0](#_Toc199448776)

[Iteración inicial: 1](#_Toc199448777)

[Diagramas y utilidades 1](#_Toc199448778)

[Introducción 1](#_Toc199448779)

[Propósito del sistema 1](#_Toc199448780)

[Alcance del sistema 1](#_Toc199448781)

[✅Requerimientos 2](#_Toc199448782)

[Funcionales 2](#_Toc199448783)

[No funcionales: 2](#_Toc199448784)

[Requisitos técnicos: 2](#_Toc199448785)

[Consideraciones Futuras: 3](#_Toc199448786)

[🔍 Análisis 3](#_Toc199448787)

[Análisis del problema 3](#_Toc199448788)

[Usuarios y actores involucrados 3](#_Toc199448789)

[Requisitos informativos 3](#_Toc199448790)

[Requisitos de procesamiento 4](#_Toc199448791)

[Análisis de viabilidad 4](#_Toc199448792)

[Casos de Uso: 4](#_Toc199448793)

[Descripcion de casos de uso: 5](#_Toc199448794)

[Diagramas Business Process Model and Notation (BPMN): 6](#_Toc199448795)

[Diseño: 7](#_Toc199448796)

[Primera iteración: 7](#_Toc199448797)

[Implementación: 8](#_Toc199448798)

[Testing: 8](#_Toc199448799)

[Segunda iteración: 9](#_Toc199448800)

[🔍 Análisis: 9](#_Toc199448801)

[Diagrama de Secuencia: 9](#_Toc199448802)

[Descripcion de casos de uso: 10](#_Toc199448803)

[Diagrama de despliegue: 10](#_Toc199448804)

[Arquitectura: 13](#_Toc199448805)

[¿Por qué utiliza arquitectura Cliente-Servidor? 13](#_Toc199448806)

[¿Qué API utiliza? 13](#_Toc199448807)

[¿Cómo funciona el sistema? 13](#_Toc199448808)

[Referencias de la Api 14](#_Toc199448809)

[Aclaraciones: 14](#_Toc199448810)

[Notas 15](#_Toc199448811)

# **Iteración inicial:**

## Diagramas y utilidades

* Casos de Uso: [Diagrama de Casos de Uso](Diagrama%20de%20Casos%20de%20Uso)
* Secuencia: [Diagrama de Secuencia](Diagrama%20de%20Secuencia)
* Despliegue: [Diagrama de Despliegue](Diagrama%20de%20Despliegue)
* BPMN: [Diagramas BPMN](Diagramas%20BPMN)
* Descripción de casos de uso: [Descripción de Casos de Uso](Descripcion%20de%20Casos%20de%20Uso)
* Hojas de testing: <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:**

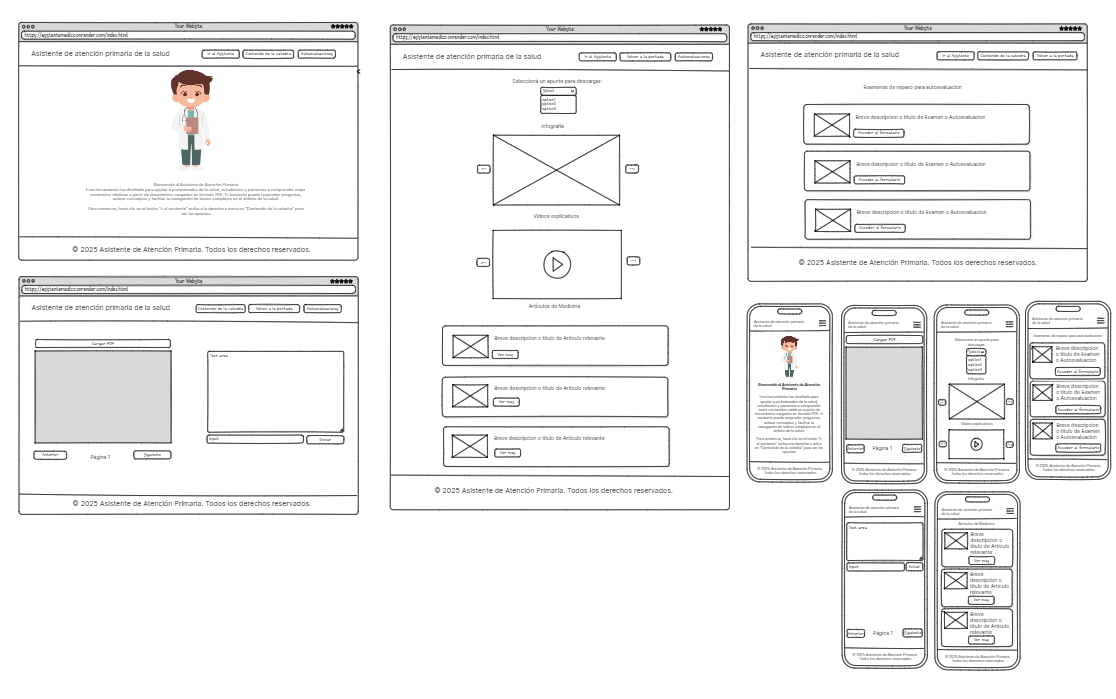
ARCHIVO ADJUNTO “Descripcion\_Casos\_Uso.docx”: [Descripción de Casos de Uso](file:///C:\Users\felip\OneDrive\Escritorio\Facultad\APUNTES%20TERCER%20ANO\DISENO%20DE%20SISTEMAS%20DE%20INFORMACION\TP\Descripcion%20de%20Casos%20de%20Uso)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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).