UNIVERSIDAD ANDINA DEL CUSCO
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

SPRINT 4 PRISSILA CHATBOT HISTORY

EQUIPO: GINZA SURVIVORS

ASIGNATURA: INTELIGENCIA ARTIFICIAL

DOCENTE: MGT. ING. ESPETIA HUAMANGA, HUGO

PRESENTADO POR:

- CORDOVA CAMACHO, KENSHIN JASON (SM) - 100%
- MOSTACERO FLORES, LEONARDO ENRIQUE - 100%
- HUAMANÑAHUI CONDORI, JADEE NASHIRA - 100%
- ZAVALETA HURTADO, KEVYN DIEGO - 100%
- ZUÑIGA CARDENAS, LUCIANA VALERIA - 100%

https://github.com/LeonardoUandina/Proyecto-IA

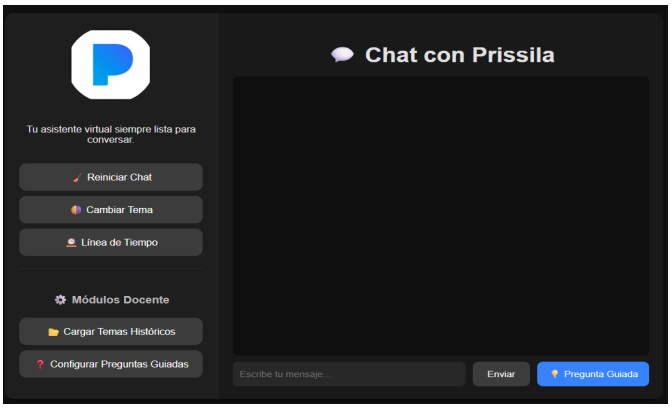
CUSCO - PERÚ
2025

1) Resumen ejecutivo

En este sprint final se culminó la implementación del Módulo de Línea de Tiempo Histórica y se realizó el despliegue funcional del sistema. Además, se integró el botón 💡**"Pregunta Guiada"** en el chat para activar dinámicas docentes sin comandos manuales.

- ➢ Chat con Prissila (LLM GPT-oss-20B).
- ➢ Carga de temas históricos por CSV.
- ➢ Línea de tiempo visual basada en los temas cargados.
- ➢ Módulo de preguntas guiadas y botón rápido en el chat.

Estructura de proyecto organizada para despliegue local.



2) Objetivos del sprint

- ➢ Implementar un componente visual interactivo que muestre eventos históricos en una línea de tiempo.
- ➢ Conectar la línea de tiempo con el CSV cargado por el docente.
- ➢ Añadir botón "Pregunta Guiada" en el chat para evitar comandos ("activar preguntas").
- ➢ Documentar y ejecutar el despliegue del sistema (entorno local Windows).

Criterios de aceptación

- La vista /timeline muestra los eventos del archivo uploads/temas.csv.
- Si no hay archivo o está vacío, se muestra un mensaje claro al usuario.
- El botón-Pregunta Guiada devuelve una pregunta desde uploads/preguntas.csv.
- El sistema levanta correctamente en http://127.0.0.1:PORT/ (con fallback de puerto si 5000 está ocupado).

3) Alcance entregado

- Backend (Flask):
  - Rutas: /, /chat, /upload, /upload_csv, /preguntas, /guardar_pregunta, /timeline, /delete_timeline (opcional si lo agregaste).
  - Lectura de CSV (temas y preguntas) con pandas.
  - Integración con GPT-OSS-20B vía langchain_community.llms.openai
- Frontend (HTML/CSS/JS):
  - Plantillas index.html, upload.html, preguntas.html, timeline.html.
  - Botón "Pregunta Guiada" en el chat.
  - Estilos modo oscuro/claro y layout con sidebar.
- Archivos de datos:
  - uploads/temas.csv → fecha, evento, descripcion
  - uploads/preguntas.csv → titulo_sesion, pregunta, categoria, dificultad

Estructura de carpetas del proyecto

```
mi_chatbot/
├── app.py
├── uploads/
│   ├── temas.csv
│   └── preguntas.csv
├── templates/
│   ├── index.html
│   ├── upload.html
│   ├── preguntas.html
│   └── timeline.html
└── static/
    ├── styles.css
    └── logo.png
```

4) Diseño técnico (resumen)

4.1 Arquitectura

- Flask como servidor web.
- LangChain + GPT-OSS-20B
- pandas para E/S de CSV.
- Frontend clásico (HTML + CSS + JS), sin framework adicional.

Diagrama simple de arquitectura

4.2 Rutas clave

- GET / → Chat.
- POST /chat → Interfaz con LLM y modo "pregunta guiada".
- GET /upload → Form de carga de CSV.
- POST /upload_csv → Guarda uploads/temas.csv.
- GET /preguntas → Form de creación + listado.
- POST /guardar_pregunta → Agrega fila a uploads/preguntas.csv.

- ➢ GET /timeline → Renderiza eventos desde uploads/temas.csv.
- ➢ POST /delete_timeline → Borra uploads/temas.csv (si lo habilitaste).

[Código 1 – extracto rutas Flask]

5) Implementación del Módulo de Línea de Tiempo

5.1 Estructura del CSV de Temas

- ➢ Columnas requeridas:

- ○ fecha → formato ISO recomendado YYYY-MM-DD ó solo YYYY.
- ○ evento → título corto del hito.
- ○ descripcion → texto breve para contexto.

[Código 2 – ejemplo de temas.csv]

```
fecha,evento,descripcion
1492-10-12,Descubrimiento de América,Cristóbal Colón llega al continente americano.
1789-07-14,Revolución Francesa,La toma de la Bastilla marca el inicio del proceso revolucionario.
1969-07-20,Llegada del hombre a la Luna,Neil Armstrong pisa la superficie lunar (Apolo 11).
```

5.2 Backend: preparación de datos

En /timeline se lee uploads/temas.csv, se transforma en lista de objetos y se envía a la plantilla.

[Código 3 – extracto app.py /timeline]

```
@app.route("/timeline")
def timeline_page():
    temas_path = os.path.join(UPLOAD_FOLDER, "temas.csv")
    eventos = []
    if os.path.exists(temas_path):
        df = pd.read_csv(temas_path)
        for _, row in df.iterrows():
            eventos.append({
                "fecha": str(row.get("fecha", "")),
                "evento": row.get("evento", ""),
                "descripcion": row.get("descripcion", "")
            })
    return render_template("timeline.html", eventos=eventos)
```

5.3 Frontend: visualización

La plantilla timeline.html recibe eventos y los dibuja en una línea vertical estilizada (o en tu versión con vis.js si la estás usando).

[Código 4 – extracto de timeline.html iterando eventos]

```
<script>
  const eventos = {{ eventos | tojson | safe }};
  const contenedor = document.getElementById("timeline");

  if (!eventos.length) {
    contenedor.innerHTML = "<p>No hay eventos. Sube un CSV con columnas 'fecha', 'evento' y 'descripcion'.</p>";
  } else {
    eventos.forEach(e => {
      contenedor.innerHTML += `
        <div class="event">
          <strong>${e.fecha} — ${e.evento}</strong>
          <div class="descripcion">${e.descripcion}</div>
        </div>
      `;
    });
  }
</script>
```

[Captura 7 – Línea de tiempo con eventos cargados]

5.4 Eliminar línea de tiempo (opcional)

Se agregó /delete_timeline para borrar uploads/temas.csv y dejar la línea de tiempo vacía.

[Código 5 – extracto delete_timeline]

```
@app.route("/delete_timeline", methods=["POST"])
def delete_timeline():
                    temas_path     = os.path.join(UPLOAD_FOLDER, "temas.csv")
    if os.path.exists(temas_path):
        os.remove(temas_path)
            flash("Línea de tiempo eliminada correctamente.")
    else:
         flash("No existe una línea de tiempo para eliminar.")
                                  return redirect(url_for("timeline_page"))
```

[Captura 8 – Confirmación de eliminación]

6) Integración del botón "Pregunta Guiada"

Se añadió un botón junto a Enviar en index.html, que invoca /chat con el input "activar preguntas" para que el backend elija una pregunta aleatoria desde uploads/preguntas.csv.

[Código 6 – extracto index.html]

```
<div class="input-container">
            <input      type="text" id="user-input" placeholder="Escribe tu mensaje..." />
          <button     id="send-btn" onclick="sendMessage()">Enviar</button>
         <button     id="guided-btn" onclick="loadGuidedQuestion()">💡 Pregunta Guiada</button>
</div>
```

```
<script>
                async       function loadGuidedQuestion() {
                const    chatBox    = document.getElementById("chat-box");
              const    guidedBtn    = document.getElementById("guided-btn");
    guidedBtn.disabled = true;

        chatBox.innerHTML  +=  `<div class="user-msg"><b>Tú:</b> (Solicitar pregunta guiada)</div>`;

          const  response  =  await fetch("/chat", {
      method: "POST",
         headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ input: "activar preguntas" })
    });

            const   data   =   await response.json();
        chatBox.innerHTML  +=  `<div class="ai-msg"><b>Prissila:</b> ${data.response}</div>`;
              chatBox.scrollTop    = chatBox.scrollHeight;
    guidedBtn.disabled = false;
  }
</script>
```

[Captura 9 – Chat mostrando pregunta guiada]

7) Pruebas realizadas

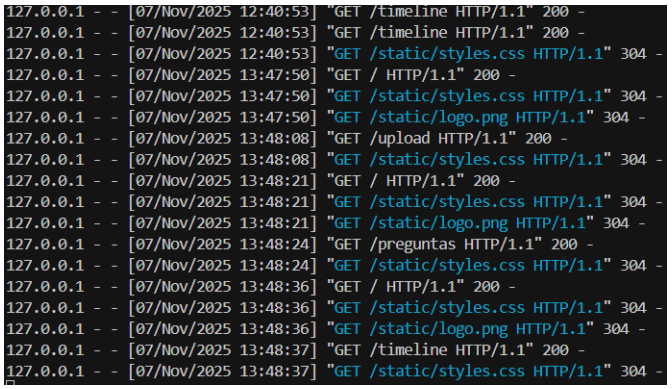
7.1 Unitarias (manuales)

- Carga de CSV válido/ inválido en /upload.
- Render de eventos en /timeline con:
  - Fechas YYYY-MM-DD y YYYY.
  - CSV vacío.
  - Falta de columnas → mensaje guiado.
- Botón Pregunta Guiada con:

- preguntas.csv inexistente.
- preguntas.csv con 1..n preguntas.

7.2 Integración

- ➢ Flujo completo Docente: crear pregunta → Alumno: Pregunta Guiada.
- ➢ Subir temas.csv → ver línea de tiempo → borrar línea de tiempo.



8) Guía de despliegue (local Windows)

8.1 Requisitos

- Python 3.10–3.12 (funciona también en 3.13, pero mejor 3.10/3.11).

- pip install -r requirements.txt
- GPT-OSS20B instalado y corriendo.

[Código 7 – requirements.txt sugerido]

```
flask==3.0.3
pandas==2.2.2
langchain==0.3.7
langchain-community==0.3.7
```

8.2 Levantar servidor

```
python app.py
# Si 5000 está ocupado:
# app.run(debug=True, port=5050)
```

Ingresar a:

- http://127.0.0.1:5000/ (o :5050 si cambiaste puerto)

[Captura 12 – Terminal mostrando Flask levantado]

9) Seguridad y datos

- ➢ Los CSV viven en uploads/ localmente (no se exponen por ruta estática).
  No se procesan archivos distintos a .csv.
- ➢ Los datos de sesión de chat viven en memoria (si se reinicia el server, se limpian).

10) Rendimiento

- ➢ Carga de CSV con pandas (rápido para decenas/miles de filas).
- ➢ Interfaz ligera sin frameworks pesados.
- ➢ LLM local (latencia depende del hardware del equipo).

11) Riesgos y mitigaciones

- ➢ Puerto ocupado → usar port=5050.

- ➢ Fechas mal formateadas → guía de formato y validación básica en el front/back.

12) Próximos pasos (roadmap)

- ➢ Filtros en línea de tiempo (rango de años, categorías).
- ➢ Corrección automática de respuestas en Preguntas Guiadas.
- ➢ Exportación de resultados/ sesiones a CSV.
- ➢ Autenticación básica para el módulo docente.
- ➢ Persistencia de historial de chat por usuario.

13) Anexos

13.1 CSV ejemplo – Temas (listo para subir)

```
temas.csv

fecha,evento,descripcion
476-09-04,Caída del Imperio Romano de Occidente,Depósito de Rómulo Augústulo marca el fin de Roma.
```

```
1492-10-12,Descubrimiento de América,Cristóbal Colón llega al continente americano.
1789-07-14,Revolución Francesa,La toma de la Bastilla inicia el proceso revolucionario.
1914-07-28,Primera Guerra Mundial,Inicio del conflicto mundial que duró hasta 1918.
1969-07-20,Llegada del hombre a la Luna,Neil Armstrong pisa la Luna (Apolo 11).
2020-03-11,Declaración de pandemia de COVID-19,La OMS declara pandemia por SARS-CoV-2.
```

13.2 CSV ejemplo – Preguntas Guiadas

preguntas.csv

titulo_sesion,pregunta,categoria,dificultad
Descubrimiento de la penicilina,"¿Quién descubrió la penicilina y en qué circunstancias?",Medicina,Media
Revolución Francesa,"¿Qué suceso simboliza el inicio de la Revolución Francesa?",Historia Moderna,Fácil
Guerra Fría,"¿Qué evento simboliza el fin de la Guerra Fría?",Contemporánea,Media

FASE DE DESPLIEGUE

Objetivo	Descripción
Desplegar el sistema completo	Ejecutar el chatbot con todos los módulos integrados (chat, preguntas guiadas, línea de tiempo y carga de temas).
Implementar módulo de línea de tiempo	Incorporar una interfaz visual moderna e interactiva que muestra eventos históricos.
Refinar módulo de preguntas guiadas	Permitir que el docente genere preguntas y los usuarios las activen desde el chat.
Optimizar la interfaz	Garantizar una experiencia fluida y moderna con un diseño adaptable en modo claro y oscuro.
Validar estabilidad	Asegurar que la aplicación funcione sin errores en todas sus rutas.

3. Descripción del Entorno de Despliegue

Elemento	Descripción
Backend	Flask (Python 3.13)
Frontend	HTML5, CSS3 y JavaScript
Motor de IA	GPT-OSS-20B
Estructura de datos	Archivos CSV (temas y preguntas guiadas)
Entorno de ejecución	Localhost (http://127.0.0.1:5050)
Sistema operativo	Windows 10 / 11
Dependencias clave	langchain, flask, pandas, tensorflow

4. Despliegue del Módulo de Línea de Tiempo

El módulo de **línea de tiempo histórica** permite visualizar de forma cronológica los eventos cargados en formato CSV.
Los docentes pueden subir un archivo con estructura:

fecha,evento,descripcion
1492,Descubrimiento de América,Cristóbal Colón llega al Nuevo Mundo.
1789,Revolución Francesa,Inicio del cambio político y social en Francia.
1914,Inicio de la Primera Guerra Mundial,Conflicto global entre potencias europeas.

Funcionamiento:

1. El archivo se sube desde /upload.

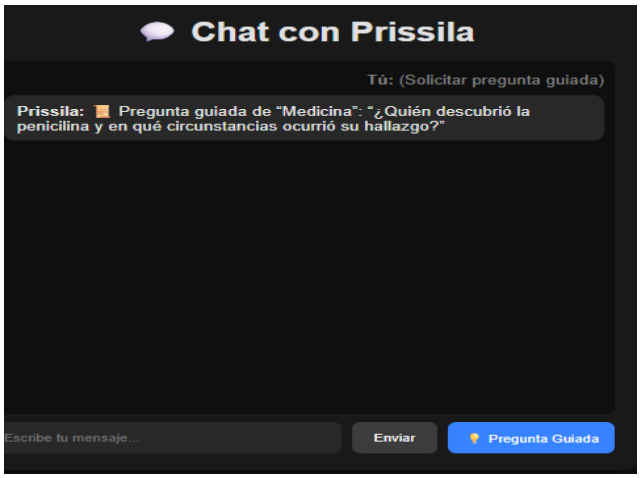
2. Flask lo guarda en /uploads/temas.csv.
3. En /timeline, el sistema renderiza los eventos dinámicamente con HTML + JS.
4. Los eventos se muestran con animaciones y estilos oscuros y elegantes.

5. Actualización del Módulo de Preguntas Guiadas

En este sprint se incorporó el **botón de activación directa** dentro del chat principal.

Anteriormente, el usuario debía escribir "activar preguntas"; ahora basta con presionar el botón **Pregunta Guiada**, el cual:

- Envía la orden automáticamente al backend (/chat).
- Recupera una pregunta aleatoria del archivo preguntas.csv.
- Prissila muestra la pregunta guiada contextualizada en el chat.



6. Pruebas de Despliegue

Durante la etapa de validación se verificaron los siguientes puntos:

Prueba	Resultado	Observaciones
Carga de modeloGPT OSS 20B	Exitosa	Flujo estable en entorno local
Envío de mensajes en chat	Correcto	Respuestas coherentes y contextualizadas
Activación de pregunta guiada	Correcto	Flujo instantáneo con botón
Carga y lectura de CSVs	Correcto	Archivos reconocidos sin errores
Visualización de línea de tiempo	Correcto	Renderización ordenada por fecha
Cambio de tema (dark/light)	Correcto	Interfaz adaptable

7. Despliegue Final

El sistema se ejecuta desde la consola mediante:

```
python app.py
```

Y se accede desde el navegador:

http://127.0.0.1:5050

El despliegue local está preparado para un futuro alojamiento en la nube (por ejemplo, **Render, Railway o Hugging Face Spaces**) con mínima adaptación.

8. Conclusiones

El despliegue del proyecto **Prissila** consolidó con éxito una solución educativa y tecnológica basada en IA.

El sistema combina interacción conversacional, recursos docentes configurables y visualización de datos históricos, cumpliendo con todos los requerimientos funcionales del backlog.

Este sprint final permitió:

- ➢ Integrar componentes visuales dinámicos.
- ➢ Mejorar la usabilidad del entorno educativo.
- ➢ Finalizar el despliegue de una aplicación lista para demostración académica o piloto institucional.