

Informe Final

— ASISTENTE TURÍSTICO DE TENERIFE —

1. Objetivo y alcance

El objetivo de este proyecto es el desarrollo de un **asistente conversacional turístico** capaz de responder preguntas sobre Tenerife a partir de un documento base (TENERIFE.pdf) y de herramientas adicionales.

El sistema combina técnicas de **Retrieval-Augmented Generation (RAG)** y **function calling** con un modelo LLM de OpenAI, permitiendo:

- Responder preguntas sobre lugares de interés turístico directamente basándose en la guía PDF.
- Mantener un **diálogo multiturno** con memoria limitada.
- Consultar información complementaria mediante herramientas:
 - `get_weather(fecha, lugar)` → pronóstico simulado.
 - `get_sunlight(fecha, lugar)` → horarios de amanecer y atardecer reales.

Alcance del proyecto:

- Construcción de un **índice vectorial** del documento base.
- Generación de respuestas con **citaciones obligatorias** al PDF para asegurar trazabilidad.
- Implementación de un flujo de conversación multiturno que refuerza la contextualización.
- Integración de herramientas externas simuladas/realistas mediante function calling.
- Evaluación cuantitativa y cualitativa de los resultados.

2. Arquitectura y flujo

La arquitectura del asistente combina tres bloques principales: **ingesta de datos**, **generación aumentada con recuperación (RAG)** y **orquestración con herramientas externas**.

2.1. Ingesta e indexación

1. **Fuente de datos:** se parte del documento TENERIFE.pdf, que contiene información turística.
2. **Extracción de texto:** el PDF se procesa con PyPDF para obtener el contenido de cada página.
3. **Segmentación (*chunking*):**
 - Se aplica tokenización con tiktoken para dividir el texto en fragmentos de unas **450 tokens** con un solape de **80 tokens**.
 - Este enfoque mejora la cobertura y evita perder contexto entre secciones.
4. **Embeddings:** cada fragmento se transforma en un vector semántico utilizando sentence-transformers con el modelo **all-MiniLM-L6-v2**.
5. **Base vectorial:** los embeddings se almacenan en **ChromaDB**, en modo persistente (chroma_db/), lo que permite reutilizar el índice en ejecuciones posteriores.

2.2. Recuperación y generación (RAG)

Cuando el usuario formula una pregunta:

1. Se realiza una búsqueda semántica en Chroma para obtener los fragmentos más relevantes.
2. Los fragmentos recuperados se pasan al LLM como **contexto adicional** junto con la instrucción de que la respuesta debe basarse **únicamente** en la información proporcionada.

El modelo debe finalizar cada respuesta con una **citación obligatoria** en el formato:

‘ [TENERIFE.pdf · p. X, Y]’

3. Esto garantiza la trazabilidad de las respuestas al documento fuente.

2.3. Diálogo multiturno

- Se implementa un historial de conversación limitado con deque, que conserva pares usuario-asistente hasta un máximo de **8 turnos**.
- En cada nueva pregunta, además del historial, se ejecuta nuevamente la fase de recuperación para actualizar el contexto.
- Este mecanismo permite mantener la coherencia en interacciones sucesivas sin sobrepasar los límites de tokens.

2.4. Orquestación con herramientas externas

El sistema integra dos herramientas mediante **function calling**:

- **get_weather(fecha, lugar):**
 - Valida la fecha en formato ISO (YYYY-MM-DD) y normaliza el lugar.
 - Devuelve un pronóstico **simulado** con temperaturas mínimas y máximas y un resumen textual.
 - La validación de argumentos se implementa con **Pydantic**, lo que añade robustez y control de errores.
- **get_sunlight(fecha, lugar):**
 - Calcula horas reales de amanecer y atardecer utilizando la librería astral y coordenadas predefinidas de varios municipios de Tenerife.

La orquestación sigue un esquema de **doble llamada** al modelo:

1. **Primera llamada:** el LLM decide si es necesario invocar alguna herramienta.
2. **Ejecución en Python:** si corresponde, se ejecuta la tool y se incorpora el resultado a la conversación.
3. **Segunda llamada:** el LLM redacta la respuesta final combinando contexto RAG, historial y salida de la tool.

Este diseño asegura la integración fluida entre recuperación documental y acceso a datos externos.

3. Diseño de prompts y citaciones

El diseño de los *prompts* constituye un aspecto clave para garantizar la calidad y la trazabilidad de las respuestas. En este proyecto se han definido distintos *system prompts* y directrices de uso.

3.1. Prompt para RAG

En la fase de recuperación aumentada, el modelo recibe:

- **Contexto:** fragmentos recuperados del índice vectorial.
- **Instrucciones:** un mensaje del sistema que indica de forma explícita:
 - Responder **exclusivamente** con la información del contexto.
 - Evitar la invención de datos externos.
 - Finalizar siempre con una citación a las páginas relevantes.

Ejemplo simplificado de prompt del sistema:

'Eres un asistente turístico de Tenerife.

Responde SOLO con el contexto proporcionado

y termina SIEMPRE con la citación exacta al PDF en formato [TENERIFE.pdf · p. ...].'

3.2. Prompt para diálogo multiturno

En el caso del diálogo con memoria corta:

- El *system prompt* refuerza la instrucción de utilizar el contexto y mantener las citaciones.
- Se incluyen los mensajes históricos (usuario y asistente) más recientes para mantener coherencia.
- Cada nueva pregunta vuelve a inyectar fragmentos recuperados, garantizando que el modelo disponga de información actualizada.

3.3. Prompt para orquestación con tools

Cuando el sistema combina RAG y function calling:

- El *system prompt* incluye indicaciones específicas sobre **cuándo debe usarse una herramienta**.
 - Ejemplo: *“Si la pregunta es sobre el tiempo en una fecha/lugar, usa `get_weather`. Si preguntan por amanecer/atardecer, usa `get_sunlight`. Para información general, responde con el contexto RAG.”*
- El modelo recibe la lista de herramientas disponibles (`tools`) con su descripción y parámetros.
- En la primera llamada, puede devolver un `tool_call`, que posteriormente es gestionado en Python.

3.4. Estrategia de citaciones

Cada respuesta final debe incluir una citación en formato:

`'[TENERIFE.pdf · p. X, Y]'`

- El sufijo de citación (`citation_tail`) se genera automáticamente a partir de los fragmentos recuperados.
- De esta forma, se garantiza que el asistente muestre de manera clara qué partes del documento sustentan su respuesta.
- La citación funciona tanto en respuestas basadas únicamente en RAG como en aquellas que integran resultados de tools.

4. Integración de herramientas (Function Calling)

La integración de herramientas externas permite ampliar las capacidades del asistente más allá de la información contenida en el documento PDF. En este proyecto se han implementado dos herramientas principales utilizando la funcionalidad de *function calling* de los modelos de OpenAI.

4.1. Herramienta `get_weather(fecha, lugar)`

- **Propósito:** ofrecer un pronóstico meteorológico simulado para una fecha y localidad concreta de Tenerife.
- **Validación de entrada:**
 - Se exige formato de fecha ISO (YYYY-MM-DD).
 - El lugar debe corresponderse con alguno de los topónimos definidos en un diccionario interno (Santa Cruz, La Laguna, Costa Adeje, La Orotava).
- **Implementación:**
 - Se utiliza **Pydantic** para validar los argumentos y generar mensajes de error claros en caso de parámetros inválidos.
 - El resultado incluye fecha, lugar normalizado, temperaturas mínimas y máximas, y un resumen fijo ("*pronóstico simulado*").

Ejemplo de salida:

```
{  
  
  "fecha": "2025-09-12",  
  
  "lugar": "Costa Adeje",  
  
  "resumen": "Soleado con intervalos nubosos · (pronóstico simulado)",  
  
  "temperatura_min": 18,  
  
  "temperatura_max": 26,  
  
  "unidad": "°C"  
  
}
```

4.2. Herramienta `get_sunlight`(fecha, lugar)

- **Propósito:** calcular horas reales de amanecer y atardecer en distintas localidades de Tenerife.
- **Implementación:**
 - Se apoya en la librería **Astral**, que permite obtener efemérides solares a partir de coordenadas geográficas.
 - Se han configurado coordenadas para varias ciudades (Santa Cruz, La Laguna, Costa Adeje, La Orotava).
 - El resultado devuelve hora de amanecer y atardecer en zona horaria **Atlantic/Canary**.

Ejemplo de salida:

```
{  
  
  "lugar": "La Laguna",  
  
  "fecha": "2025-09-12",  
  
  "amanecer": "07:49",  
  
  "atardecer": "20:13",  
  
  "tz": "Atlantic/Canary"  
}
```

4.3. Orquestación del flujo

La orquestación de tools se gestiona en tres pasos dentro de la función `chat_with_tools(...)`:

1. **Primera llamada al LLM:**
 - El modelo recibe la pregunta del usuario y el contexto recuperado del PDF.
 - Según las instrucciones, puede decidir invocar una herramienta y devolver un bloque `tool_calls`.

2. Ejecución en Python:

- Cada `tool_call` es procesado por el *router* (`tool_router`), que ejecuta la función correspondiente y devuelve el resultado en formato JSON.
- Los resultados se añaden como mensajes de tipo `tool` a la conversación.

3. Segunda llamada al LLM:

- El modelo redacta la respuesta final, integrando el resultado de la `tool`, el contexto RAG y el historial de conversación.
- La respuesta incluye obligatoriamente la citación al PDF.

4.4. Manejo de errores y robustez

- **Validación con Pydantic:** previene fallos comunes en `get_weather` (fechas mal formateadas, lugares no admitidos).
- **Logging con Loguru:** registra entradas, salidas y errores de cada `tool` en la carpeta `logs/`.
- **Mensajes de error amigables:** en caso de fallo, el asistente informa al usuario con un texto claro en lugar de interrumpir la ejecución.

5. Evaluación y resultados

Para verificar el correcto funcionamiento del asistente, se diseñó un **bloque de pruebas automáticas** que combina preguntas de distinta naturaleza:

- **Consultas RAG puras**, basadas únicamente en el PDF.
- **Consultas meteorológicas** (≥ 3), que deben activar la herramienta `get_weather`.
- **Consultas astronómicas**, que utilizan la herramienta `get_sunlight`.

5.1. Conjunto de pruebas

Ejemplo de *dataset* de evaluación:


```
tests = [  
    "Itinerario sencillo por Santa Cruz para medio día.",  
    "¿Qué recomiendan ver en La Orotava?",  
    "¿Cómo estará el tiempo el 2025-09-12 en Costa Adeje?",  
    "Dime el pronóstico para 2025-09-13 en Santa Cruz de Tenerife",  
    "Pronóstico para 2025-09-14 en La Laguna",  
    "¿A qué hora amanece y atardece el 2025-09-12 en La Laguna?",  
]
```

5.2. Métricas recogidas

El bloque de pruebas calcula automáticamente:

- **% de respuestas con citación:** mide la adherencia a la instrucción de trazabilidad.
- **Nº de llamadas a cada herramienta** (heurístico): permite verificar que las tools se activan cuando corresponde.
- **Latencia media por consulta:** tiempo total desde la pregunta hasta la respuesta.

Ejemplo de salida:

```
MÉTRICAS  
{  
  'n_tests': 6,  
  '%_respuestas_con_citación': 100.0,  
  'latencia_media_s': 7.3,  
  'calls_get_weather(heurístico)': 3,  
  'calls_get_sunlight(heurístico)': 1  
}
```

5.3. Resultados cualitativos

- **Consultas RAG:** el asistente devuelve itinerarios y recomendaciones claras, siempre con referencia a las páginas del PDF.

- **Consultas meteorológicas:** el modelo invoca correctamente la herramienta `get_weather`, proporcionando un pronóstico simulado con datos estructurados.
- **Consultas de amanecer/atardecer:** se activa `get_sunlight` y se devuelven horarios realistas basados en la librería Astral.
- **Diálogo multiturno:** las pruebas confirman que el sistema mantiene coherencia en las respuestas y actualiza contexto en cada interacción.

5.4. Conclusiones de la evaluación

- El sistema cumple con los **requisitos obligatorios** de la entrega (RAG con citas, multiturno, function calling con ≥ 3 llamadas a `get_weather`).
- Se incorporan además **bonificaciones**:
 - Validación de parámetros con **Pydantic**.
 - Registro en **logs** de entradas/salidas de tools.
 - **Métricas cuantitativas** de evaluación.
- Las métricas confirman un **100 % de respuestas con citación**, lo que demuestra que las instrucciones de prompt y la estrategia de citación automática son efectivas.

6. Limitaciones

Aunque el asistente cumple con los requisitos planteados y ha demostrado un funcionamiento correcto en las pruebas realizadas, se identifican varias limitaciones que condicionan su alcance:

6.1. Pronóstico meteorológico simulado

- La herramienta `get_weather` devuelve un resultado sintético y no consulta fuentes meteorológicas reales.
- Esto implica que los valores de temperatura o el resumen del tiempo no reflejan la realidad y solo sirven como ejemplo de integración de tools.

6.2. Cobertura limitada de topónimos

- El sistema reconoce únicamente un conjunto reducido de localidades de Tenerife (Santa Cruz, La Laguna, Costa Adeje, La Orotava).
- Preguntas sobre otras poblaciones o variaciones de nombre no incluidas en el diccionario pueden no resolverse correctamente.

6.3. Memoria conversacional acotada

- La memoria multiturno se gestiona con un historial fijo de 8 turnos.
- Esto permite mantener la coherencia en diálogos cortos, pero en conversaciones más extensas se pierde parte del contexto antiguo.
- No se implementaron técnicas más avanzadas de memoria (resúmenes, distillation).

6.4. Dependencia del modelo LLM

- El sistema depende del cumplimiento estricto del *prompt* por parte del modelo.
- En algunos casos, el LLM podría generar respuestas sin citación, aunque en las pruebas el mecanismo de `citation_tail` ha mitigado este riesgo.

6.5. Ausencia de interfaz gráfica

- El proyecto se ejecuta en un notebook, sin una interfaz gráfica para el usuario final.
- Aunque no era requisito obligatorio, la ausencia de una UI limita su usabilidad en entornos no técnicos.

7. Riesgos y consideraciones éticas

El desarrollo y uso de un asistente basado en LLMs conlleva ciertos riesgos técnicos y éticos que es importante reconocer y mitigar.

7.1. Riesgos de alucinación

- Aunque el sistema está diseñado para responder exclusivamente con el contexto recuperado del PDF, los modelos de lenguaje pueden generar información inventada.
- La estrategia de **citaciones obligatorias** ayuda a reducir este riesgo, ya que obliga al modelo a justificar sus respuestas con referencias al documento base.
- Aun así, no se garantiza al 100 % la ausencia de alucinaciones si el modelo ignora las instrucciones.

7.2. Limitaciones de la información

- La guía turística utilizada (TENERIFE.pdf) puede estar desactualizada o incompleta.
- El sistema solo es tan fiable como lo es su fuente de datos.
- Para aplicaciones reales, sería necesario incorporar fuentes actualizadas y mecanismos de verificación externa.

7.3. Uso de datos y privacidad

- El proyecto no procesa datos personales, únicamente el contenido del PDF y entradas de usuario durante la conversación.
- Sin embargo, al utilizar la API de OpenAI, las consultas se envían a un servicio externo.
- Es fundamental que los usuarios no introduzcan información sensible en el sistema.

7.4. Transparencia con el usuario

- El asistente informa explícitamente cuando ofrece un **pronóstico simulado**, evitando confusión con datos reales.
- La transparencia es clave para evitar que el usuario tome decisiones prácticas basadas en información ficticia.

7.5. Dependencia tecnológica

- El sistema depende de servicios externos (OpenAI, embeddings de sentence-transformers) y de la conectividad a Internet.

- Para entornos de misión crítica o desconectados, sería necesario evaluar alternativas locales o autoalojadas.

8. Trabajo futuro

El proyecto ha alcanzado los objetivos planteados, pero existen múltiples oportunidades de mejora y extensión que permitirían aumentar su utilidad y robustez:

8.1. Integración con APIs externas

- Sustituir el pronóstico meteorológico simulado por datos reales obtenidos de APIs abiertas como **AEMET** u **Open-Meteo**.
- Añadir caché local para reducir latencia y evitar exceso de llamadas externas.

8.2. Ampliación de cobertura geográfica

- Extender el diccionario de topónimos para cubrir más municipios y variantes de nombres (acentos, sinónimos, abreviaturas).
- Permitir reconocimiento flexible con modelos de *fuzzy matching*.

8.3. Mejora de la memoria conversacional

- Implementar técnicas de **resumen dinámico** del historial o **message distillation** para diálogos largos.
- Investigar integraciones con librerías de *memory management* para agentes LLM.

8.4. Evaluación ampliada

- Diseñar un *benchmark* más amplio con preguntas de distinto nivel de dificultad.

- Medir métricas adicionales: precisión en recuperación de fragmentos, calidad percibida de respuestas, robustez a preguntas ambiguas.

8.5. Interfaz de usuario

- Desarrollar una UI web con **Streamlit** o **Gradio** para hacer el sistema accesible a usuarios no técnicos.
- Implementar chat interactivo con historial visible y despliegue en un servicio cloud (ej. Azure App Service, Hugging Face Spaces).

8.6. Despliegue y escalabilidad

- Contenerización con **Docker** para facilitar la portabilidad.
- Uso de pipelines de CI/CD en GitHub Actions para validación y despliegue automático.
- Evaluación de despliegues multiusuario y almacenamiento compartido de índices vectoriales.

9. Conclusiones

El desarrollo del **Asistente Turístico de Tenerife** ha permitido integrar en un único sistema varias de las competencias clave en el ámbito de los modelos de lenguaje y la ingeniería de *prompts*, logrando un prototipo funcional que cumple de manera satisfactoria con los requisitos establecidos en la entrega final.

En primer lugar, el uso de **Retrieval-Augmented Generation (RAG)** ha demostrado ser una estrategia eficaz para garantizar que las respuestas del modelo estén ancladas en el contenido de la guía turística TENERIFE.pdf. La división del documento en *chunks* mediante tokenización y solape ha permitido optimizar la recuperación semántica, y la utilización de **ChromaDB** como vector store persistente ha facilitado la reutilización del índice en distintas ejecuciones, reduciendo tiempos de inicialización y asegurando reproducibilidad. La exigencia de **citaciones obligatorias** en cada respuesta añade un nivel de trazabilidad y transparencia que mitiga el riesgo de alucinaciones, reforzando la fiabilidad del sistema.

En segundo lugar, la implementación del **diálogo multiturno** ha puesto de manifiesto la importancia de gestionar un historial acotado que permita mantener coherencia en interacciones sucesivas sin comprometer los límites de tokens del modelo. Aunque la memoria está limitada a un número fijo de turnos, esta funcionalidad proporciona una experiencia más realista y cercana a la interacción natural entre guía y visitante.

El componente de **function calling** constituye otro de los logros principales del proyecto. La integración de herramientas externas como `get_weather` y `get_sunlight` ha permitido enriquecer las respuestas más allá de la información contenida en el documento base. Aunque el pronóstico meteorológico es simulado, la incorporación de validación robusta mediante **Pydantic** y el uso de librerías especializadas como **Astral** para cálculos astronómicos evidencian la capacidad del sistema para interoperar con datos externos de manera controlada y extensible. Este enfoque abre la puerta a futuras integraciones con APIs reales, consolidando el sistema como una plataforma versátil.

Desde el punto de vista de la **evaluación**, el bloque de pruebas diseñado ha permitido verificar tanto el cumplimiento de requisitos mínimos como la incorporación de características de valor añadido. Los resultados obtenidos muestran un 100 % de respuestas con citación y una correcta activación de las herramientas cuando procede, lo que confirma la solidez del diseño. La inclusión de métricas cuantitativas (latencia media, número de llamadas a cada tool, porcentaje de citaciones) aporta un marco objetivo para valorar el desempeño del asistente y constituye un avance respecto a evaluaciones puramente cualitativas.

A nivel metodológico, este proyecto ha supuesto un ejercicio completo de **diseño, implementación y validación** de un sistema LLM aplicado a un caso de uso concreto. Se han puesto en práctica técnicas avanzadas como la recuperación semántica, la orquestación de llamadas a herramientas y la gestión de memoria conversacional, todo ello en un entorno controlado y reproducible. Además, se ha prestado especial atención a la documentación (README y notebook comentado), lo que garantiza que otros usuarios puedan entender, ejecutar y extender el trabajo desarrollado.

Finalmente, es importante subrayar que este proyecto no solo cumple con los requisitos académicos, sino que también aporta un **valor formativo significativo**:

- Permite comprender de manera práctica los beneficios y limitaciones de los modelos de lenguaje actuales.

- Muestra cómo integrar distintas piezas tecnológicas (OpenAI API, ChromaDB, Sentence-Transformers, Astral, Pydantic) en un flujo coherente.
- Pone de relieve la necesidad de considerar aspectos éticos, de seguridad y de fiabilidad en el uso de IA generativa.

10. Referencias

A continuación se detallan las principales fuentes, librerías y recursos utilizados durante el desarrollo del proyecto:

Bibliografía y documentación técnica

- **OpenAI API Documentation** – *Chat Completions & Function Calling*:
<https://platform.openai.com/docs>
- **ChromaDB Documentation** – *Vector database for embeddings*:
<https://docs.trychroma.com>
- **Sentence-Transformers Documentation** – *Embeddings and pretrained models*:
<https://www.sbert.net>
- **Pydantic Documentation** – *Data validation and settings management*:
<https://docs.pydantic.dev>
- **Astral Documentation** – *Astronomical calculations (sunrise, sunset)*:
<https://astral.readthedocs.io>
- **Loguru Documentation** – *Python logging library*:
<https://loguru.readthedocs.io>
- **tiktoken Documentation** – *Tokenization for LLMs*:
<https://github.com/openai/tiktoken>
- **PyPDF Documentation** – *PDF parsing and text extraction*:
<https://pypdf.readthedocs.io>

Librerías de Python utilizadas

- openai – conexión con LLMs.
- python-dotenv – gestión de variables de entorno.
- chromadb – base vectorial persistente.

- sentence-transformers – generación de embeddings.
- pydantic – validación de parámetros.
- astral – cálculo de amanecer y atardecer.
- pypdf – lectura de documentos PDF.
- tiktoken – tokenización de texto.
- loguru – gestión avanzada de logs.
- jupyter, ipykernel – ejecución en notebook.

Recursos adicionales

- Guía turística de Tenerife (TENERIFE.pdf), proporcionada como dataset base para la práctica.
- Material docente de la asignatura sobre **LLMs**.

11. Anexos

En este apartado se incluyen ejemplos prácticos y material complementario que ilustran el funcionamiento del asistente.

11.1. Ejemplo de interacción RAG

Entrada:

Itinerario sencillo por Santa Cruz para medio día.

Respuesta:

```
- Q: Itinerario sencillo por Santa Cruz para medio día.
  • citación=True • t=12.49s
  • Respuesta:
Aquí tienes un itinerario sencillo para medio día en Santa Cruz de Tenerife:

1. **Aparcamiento**: Comienza aparcando en el aparcamiento del Parque Marítimo César Manrique, donde podrás disfrutar de un conjunto de piscinas naturales.

2. **Avenida Marítima**: Desde el parque, camina por la Avenida Marítima en dirección a la Plaza de España. Durante el recorrido, no te pierdas el impresionante Auditorio de Tenerife.

3. **Plaza de España**: Al llegar a la Plaza de España, tómate un tiempo para explorar. Puedes subir por la Calle Castillo hacia la Plaza Weyler y luego dirigirte al Parque García Sanabria, un hermoso parque urbano.

4. **Callejear**: Desde el Parque García Sanabria, baja de nuevo hacia la Plaza de España, pasando por la Plaza del Príncipe. Disfruta de la arquitectura y el ambiente local.

5. **Playa de Las Teresitas**: Finalmente, regresa al coche y dirígete a la playa de Las Teresitas, donde podrás relajarte y disfrutar del sol y el mar.

Este itinerario te permitirá conocer algunos de los puntos más emblemáticos de Santa Cruz en un tiempo limitado. ¡Disfruta de tu visita!
[TENERIFE.pdf • p. 1, 22, 25]
```

11.2. Ejemplo de interacción con get_weather

Entrada:

¿Cómo estará el tiempo el 2025-09-12 en Costa Adeje?

Respuesta:

```
-----
- Q: ¿Cómo estará el tiempo el 2025-09-12 en Costa Adeje?
  • citación=True • t=4.8s
  • Respuesta:
El tiempo en Costa Adeje el 12 de septiembre de 2025 se prevé soleado con intervalos nubosos, con una temperatura mínima de 18°C y máxima de 26°C. [TENERIFE.pdf · p. 8, 12, 22]
```

11.3. Ejemplo de interacción con get_sunlight

Entrada:

¿A qué hora amanece y atardece el 2025-09-12 en La Laguna?

Respuesta:

```
-----
- Q: ¿A qué hora amanece y atardece el 2025-09-12 en La Laguna?
  • citación=True • t=3.07s
  • Respuesta:
Amanecer: 07:49, Atardecer: 20:13 [TENERIFE.pdf · p. 3, 12, 22].
```

11.4. Ejemplo de diálogo multiturno

Turno 1:

¿Qué puedo visitar en La Orotava?

Turno 2:

¿Y qué restaurantes recomiendan cerca?

Turno 1: ¿Qué puedo visitar en La Orotava?

Respuesta:

En La Orotava, hay varios lugares interesantes que puedes visitar. Te recomiendo comenzar en la conocida Plaza de Anita, donde podrás disfrutar del Liceo de Taoro y los Jardines Victoria. Luego, puedes recorrer la Calle de la Carrera, donde se encuentran el ayuntamiento de La Orotava, la Iglesia de La Concepción, la Casa Lercaro y la Casa de los Balcones. Al final de esta calle, también está el molino de Chano, donde puedes comprar gofio.

Además, es importante mencionar que el casco histórico de La Orotava es un Conjunto Histórico Artístico Nacional y Patrimonio Cultural Europeo. También es el municipio más alto de España, con un gran desnivel que va desde el mar hasta el pico del Teide. Si tienes la oportunidad, no te pierdas las fiestas de La Orotava, donde se crea una impresionante alfombra gigante de arenas y tierras del Teide, que tiene el récord Guinness del mayor tapiz de tierra del mundo. Por último, te sugiero visitar al menos una de sus playas, como la Playa del Bollullo.

Para más detalles, puedes consultar el documento [TENERIFE.pdf · p. 9, 22, 25].

Turno 2: ¿Y qué restaurantes recomiendan cerca?

Respuesta:

Cerca de La Orotava, hay varias opciones de restaurantes que te recomiendo:

1. ****Arepera La Carajita****: Ubicada entre La Orotava y el Puerto de la Cruz, es famosa por sus deliciosas arepas, especialmente la Fogalera y la Reina Pepiada. No olvides pedir una degustación de cachapitas como entrante. ¡Importante ir con hambre!

2. ****Atuvera****: También entre La Orotava y el Puerto de la Cruz, este restaurante es muy recomendable, pero es mejor hacer una reserva con antelación.

3. ****Pizzería Gioconda****: Situada en la zona de La Orotava, es una buena opción si te apetece pizza.

4. ****El Palestra****: Un clásico en La Orotava, ideal para pedir a domicilio. Su plato feliz de pizza es muy popular.

5. ****Cafetería Deyfe, Bar-Cafetería Australia y Bar El Paraíso****: Estos lugares son conocidos por sus bocadillos, especialmente el menú feliz de bocadillo de pollo, que es muy recomendable.

6. ****Restaurante Fujiyama****: Si te apetece sushi, este restaurante en la zona de Santa Cruz es una excelente opción.

No dudes en probar la comida local y disfrutar de la experiencia gastronómica que ofrece la isla. Para más información, puedes consultar el documento [TENERIFE.pdf · p. 19, 24, 25].

11.5. Ejemplo de métricas de evaluación

Salida obtenida tras ejecutar el bloque de pruebas:

```
----- MÉTRICAS -----
{'n_tests': 6, '%_respuestas_con_citación': 100.0, 'latencia_media_s': 6.49, 'calls_get_weather(heurístico)': 3, 'calls_get_sunlight(heurístico)': 1}
```

11.6. Estructura de carpetas y logs

La ejecución del notebook genera automáticamente varias carpetas auxiliares que forman parte de la estructura del proyecto:

- **chroma_db/** → contiene el índice vectorial persistente utilizado por ChromaDB.
 - En su interior se crean múltiples subcarpetas identificadas con UUIDs y un fichero chroma.sqlite3.

- Este contenido confirma que los embeddings del PDF se han calculado y almacenado correctamente.
- **logs/** → carpeta destinada al almacenamiento de registros de ejecución generados mediante la librería Loguru.
 - **ingest.log**: traza del proceso de ingesta e indexación (lectura del PDF, *chunking*, inserción en Chroma, recuento de vectores).
 - **run.log**: traza de ejecución del asistente (peticiones, decisiones de function calling, entradas y salidas de herramientas, avisos/errores controlados).

Estos archivos facilitan la auditoría y la depuración, y demuestran la trazabilidad del sistema durante la ejecución.

