



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

Eco City Tours

Aplicación móvil para la generación de
rutas turísticas sostenibles propuestas
por modelos de lenguaje de gran escala



Presentado por Fernando Pisot Serrano
en Universidad de Burgos — 25 de diciembre
de 2024

Tutor: Carlos López Nozal



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Carlos López Nozal, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Fernando Pisot Serrano, con DNI 70873328R, ha realizado el Trabajo final de Grado en Ingeniería Informática.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 25 de diciembre de 2024

Vº. Bº. del Tutor:

D. Carlos López Nozal tutor

Resumen

Eco City Tours es una aplicación móvil desarrollada con Flutter que propone al usuario rutas turísticas sostenibles. La aplicación se enfoca en la promoción de rutas no motorizadas, optimizadas para ciclistas y peatones, que conectan lugares de interés con el objetivo de fomentar la movilidad sostenible. Las rutas se generan con tecnologías usando *Geographic Information Systems (SIG)*. Los *Puntos de Interés (PDI)* se enriquecen con información detallada sobre lugares turísticos mediante inteligencia artificial consultando en lenguaje natural a *Modelos de Lenguaje de Gran Escala (LLM)*.

Esta aplicación se alinea con el concepto de *Smart City*, promoviendo activamente los *Objetivos de Desarrollo Sostenible (ODS)*, con un enfoque particular en el *ODS11: Ciudades y Comunidades Sostenibles*.

Descriptores

Movilidad Sostenible, ODS, LLM, Smart City, Flutter.

Abstract

Eco City Tours is a mobile application **developed with Flutter** that suggests tourist routes to users. The application focuses on promoting non-motorized routes, optimized for cyclists and pedestrians, connecting points of interest with the goal of fostering sustainable mobility. The routes are generated using technologies based on *Sistemas de Información Geográfica (GIS)*. The *Points of Interest (POI)* are enriched with detailed information about tourist spots through artificial intelligence, consulting *Large Language Models (LLM)* via natural language queries.

This application aligns with the concept of **Smart City**, actively promoting the *Sustainable Development Goals (SDG)*, with a particular focus on the **SDG11: Sustainable Cities and Communities**.

Keywords

Sustainable Mobility, SDGs, LLM, Smart City, Flutter.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
Índice de Códigos	vii
1. Introducción	1
2. Objetivos del proyecto	3
2.1. Objetivos funcionales	3
2.2. Objetivos no funcionales	4
2.3. Objetivos personales	5
3. Conceptos teóricos	7
3.1. Sistemas de Información Geográfica (SIG)	7
3.2. Objetivos de Desarrollo Sostenible (ODS)	8
3.3. Modelos de lenguaje a gran escala (LLM)	11
3.4. Agilidad y método SCRUM	16
4. Técnicas y herramientas	19
4.1. Desarrollo relacionado con LLM	19
4.2. Desarrollo aplicación móvil	21
4.3. Gestión de proyectos	27
5. Aspectos relevantes del desarrollo del proyecto	31
5.1. Elección de servicios Google sobre tecnología <i>Open Street Maps</i>	31

5.2. Elección de google generative ai como primer LLM	32
5.3. Aclaración sobre el coste de Servicios Google	32
5.4. Puntos de interés con LLM: alucinaciones y optimización de rutas turísticas	33
6. Trabajos relacionados	37
6.1. Aplicaciones móviles planificadoras de rutas turísticas	37
6.2. Otros Trabajo de Fin de Grado (TFG)	40
7. Conclusiones y Líneas de trabajo futuras	43
Bibliografía	53

Índice de figuras

3.1. Objetivos de Desarrollo Sostenible	9
3.2. Preparación de la información de un RAG mostrada en la herramienta Langflow	14
3.3. Chat Output a través de la salida de un RAG	15
3.4. Resumen de la compilación del proyecto en SonarCloud	18
4.1. Uso del snippet: mateapp	24
4.2. Ejemplo de <i>Ghost Text de Copilot</i>	25
4.3. Código real a utilizar en vez de propuesta Copilot	26
4.4. Control de calidad de SonarCloud respecto a issues abiertas. . .	28
4.5. Uso de GitHub con la extensión de VSCode Git Graph	29
5.1. Estadísticas de solicitudes de servicios	33
6.1. Chatbot de Wanderlog	38

Índice de tablas

6.1. Aplicaciones móviles para la planificación de itinerarios turísticos	41
6.2. Comparación de aplicaciones	42

Índice de Códigos

3.1. Ejemplo de uso de <i>few-shot</i> y <i>tool-calling</i> para obtener puntos de interés turísticos	12
4.2. Definición de variables de control en el BLoC	23
4.3. Carga en función del estado	23

1. Introducción

El crecimiento de población en las ciudades [23] y el turismo como catalizador de la gentrificación supone el gran campo de batalla para gobiernos locales de los países occidentales que han visto como la falta de una legislación controlada del turismo supone un grave problema afectando múltiples niveles de la convivencia, economía y el medio ambiente. A pesar de los avances en la promoción de un nuevo modelo urbano, muchas urbes aún enfrentan desafíos significativos en la integración de prácticas sostenibles en la vida cotidiana de sus habitantes. La falta de información accesible y personalizada sobre rutas y actividades que promuevan la movilidad sostenible y el turismo responsable es un marco común que se debe desarrollar si se quiere evitar que el conflicto crezca sin fin. Esta brecha de información impide que tanto residentes como turistas adopten hábitos más sostenibles que beneficien a la comunidad local y al medio ambiente en un marco global.

Fomentar el Turismo Sostenible supone una gran oportunidad para intentar contrarrestar la deriva actual. Y es que el turismo es un motor fundamental de la economía a nivel global y por tanto tiene la capacidad de transformarse para ayudar a la sostenibilidad del planeta. Destaca en este marco de trabajo el objetivo *Ciudades y Comunidades Sostenibles* (ODS11) titulado *Ciudades y Comunidades Sostenibles*. Según el informe de la UNESCO, [14], no solo es crucial por sí mismo, sino que actúa como un factor multiplicador, influyendo indirectamente en la consecución de otros ODS debido a su enfoque integral y transversal.

Eco City Tours **aúna estos esfuerzos al proporcionar una herramienta práctica y accesible** para la promoción del ODS11 y la movilidad sostenible. La aplicación ha sido desarrollada en Flutter y utiliza *Modelos de Lenguaje de Gran Escala* (LLM) para generar rutas turísticas personalizadas que conecten *Puntos de Interés* (PDI). La aplicación se enfoca en las

preferencias del usuario, ofreciendo **rutas optimizadas** para ciclistas y peatones promoviendo así la movilidad sostenible.

Las preferencias del usuario se comunicarán al modelo a través de un menú, donde éste podrá elegir qué lugar visitar, si realizar el tour a pie o en bicicleta, cuántos *Puntos de Interés* incluir en la ruta y sus gustos a la hora de viajar. El sistema generará el camino más corto, calculado entre los distintos **PDI** a visitar, usando para ello un servicio de geolocalización.

Mientras que muchas de las aplicaciones similares revisadas solo utilizan información comercial para definir rutas turísticas, *Eco City Tours* solicita que se tengan en cuenta prácticas sostenibles como la deslocalización del turismo a la hora de elegir destinos. Todas estas consideraciones enriquecen la experiencia turística de los visitantes [22], e incluso promueven el crecimiento económico de las comunidades locales. De este modo, *Eco City Tours* logra un impacto positivo tanto a nivel local como global.

2. Objetivos del proyecto

2.1. Objetivos funcionales

Estos objetivos se centran en las funcionalidades y características que debe tener la aplicación *Eco City Tours* para satisfacer las necesidades y expectativas de los usuarios. A continuación se detallan los objetivos funcionales del proyecto:

- **Propuesta de rutas turísticas personalizadas:** La aplicación debe ser capaz de generar rutas turísticas personalizadas basadas en las preferencias del visitante utilizando *Modelos de Lenguaje de Gran Escala (LLM)*. Para llevarlo a cabo, el usuario facilitará al modelo sus preferencias, eligiendo entre otras opciones el medio de transporte elegido o el número de *Puntos de Interés (PDI)* a visitar.
- **Obtener los *Puntos de Interés (PDI)*:** A través de la interacción con el modelo *LLM*, la aplicación le indicará que debe priorizar un *PDI* sobre otro en función de criterios sostenibles como la deslocalización del turismo y preferencias de usuario como puedan ser duración de la visita o medio de transporte ecológico a utilizar.
- **Visualización de rutas en mapa:** La aplicación debe mostrar las rutas sugeridas en un mapa utilizando herramientas *GIS*.
- **Optimización para ciclistas y peatones:** la aplicación implementará un sistema de navegación que considere la seguridad de los peatones y la priorización de carriles exclusivos para bicicletas sobre carreteras compartidas con vehículos motorizados. Según la documentación oficial de *Google Maps Directions API* [12], el modo de transporte

en bicicleta incluye funcionalidades que pueden alinearse con estos objetivos.

- **Gestión de rutas:** La aplicación permitirá a los usuarios crear, guardar y cargar rutas turísticas, facilitando una mayor personalización y aprovechamiento de la experiencia turística.

2.2. Objetivos no funcionales

Los objetivos no funcionales se refieren a los desafíos y metas que se deben abordar para desarrollar el software. Estos objetivos abarcan aspectos como la arquitectura del sistema, las tecnologías a utilizar y las metodologías de desarrollo. A continuación se detallan los objetivos no funcionales del proyecto:

- **Integración de inteligencia artificial usando *Procesamiento del Lenguaje Natural (NLP)*:** La aplicación contará con un modelo de lenguaje preseleccionado, que facilita la propuesta de rutas turísticas personalizadas y la obtención de información relevante de *Puntos de Interés*. Esta elección garantiza la estabilidad y el rendimiento de la aplicación, proporcionando información precisa y relevante sin la necesidad de cambiar modelos, lo cual reduce la complejidad de mantenimiento.
- **Preferencia por Herramientas *Open-Source*:** en el desarrollo de la aplicación se procurará, siempre que sea viable, el uso de programas, paquetes, servicios o librerías de código abierto, priorizando aquellas que no generen costos para el desarrollador o el usuario final. Sin embargo, se optará por soluciones alternativas cuando estas sean necesarias para garantizar la calidad del producto final. Este enfoque contribuye al cumplimiento del *Objetivos de Desarrollo Sostenible: Educación de Calidad (ODS4)*, promoviendo una educación inclusiva, equitativa y de calidad, y fomentando oportunidades de aprendizaje para todos.
- **Usabilidad:** la interacción del usuario con la aplicación debe ser intuitiva y sencilla, permitiendo un rápido aprendizaje de todas sus funcionalidades. El diseño de la interfaz debe estar orientado a ofrecer una experiencia de uso fluida.

2.3. Objetivos personales

- **Formación en LLM y su integración en aplicaciones software.**
Dada la rápida evolución de los *Modelos de Lenguaje de Gran Escala (LLM)* y la amplitud de campos del conocimiento en los que se pueden utilizar, obtener una base de conocimientos destacable en este área sería un objetivo que me permitiría expandir mi futuro académico y por tanto distinguir mi perfil profesional especializándome en un sector con fuerte expansión.
- **Desarrollo de aplicación móvil profesional:** poner en práctica lo aprendido en varios cursos de auto-formación online en **Dart y Flutter**. La aplicación de este proyecto puede ser parte de mi porfolio con aplicaciones que muestren mis habilidades a futuros empleadores.
- **Finalización del TFG y Grado:** tras no haber completado la Ingeniería Técnica Informática en su momento por no haber realizado el Proyecto Fin de Carrera, la realización de este **TFG** marca la culminación de mi formación académica como ingeniero.

3. Conceptos teóricos

En este capítulo se describen los conceptos necesarios para comprender el funcionamiento de la aplicación desarrollada.

3.1. Sistemas de Información Geográfica (SIG)

Un Sistema de Información Geográfica, comúnmente abreviado como **GIS** o **SIG**, es cualquier herramienta hardware o software que permita la realización de tareas sobre información que esté georreferenciada, es decir que los datos incluyen una ubicación en coordenadas geográficas [26]. Estas herramientas están específicamente diseñadas para trabajar con datos que no solo realizan operaciones usuales de inserción, eliminación, actualización y extracción en una base de datos común (**CRUD**), sino que realizan análisis con respecto a la ubicación o características geoespaciales o topográficas. Junto con los análisis espaciales, proporciona una mejor visualización de los datos con mapas, de modo que se pueda presentar la información en una manera legible que de otro modo sería difícil de interpretar.

Estas herramientas están cada vez más presentes en diversos campos de la ciencia y la informática. Facilitan la toma de decisiones en áreas como el desarrollo urbano o la optimización de rutas. Además, son capaces de mejorar la gestión de empresas que ofrecen servicios descentralizados, como el suministro de agua o energía. Es por ello que su aplicación es clave también en el análisis del cambio climático y otros desafíos globales como los *Objetivos de Desarrollo Sostenible*.

Servicios GIS empleados en *Eco City Tours*

Los sistemas o herramientas con referencias geográficas son comúnmente utilizadas como servicios en el desarrollo de aplicaciones con componentes de información geográfica. Los proveedores de estos servicios facilitan datos georreferenciados según las peticiones del usuario. *Eco City Tours* utiliza varios servicios GIS para su funcionamiento. Entre ellos se incluyen:

- **Servicio de mapas:** con Google como proveedor [13] se usa este servicio para proveer a la aplicación de un mapa para mostrar de fondo así como marcadores y polilíneas de rutas.
- **Geocodificación:** sirve para obtener las coordenadas GPS de un *Puntos de Interés* o conseguir el nombre de un lugar próximo a unas coordenadas dadas (*inverse geocoding*). Se ha utilizado este servicio con el proveedor MapBox [19].
- **Servicio de navegación y de optimización de rutas:** el servicio de navegación [18] nos proporciona una ruta entre dos *Puntos de Interés* y el servicio de optimización permite generar una ruta que sea la más corta entre múltiples **PDI** en el mapa. El proveedor de este servicio es MapBox [20]. De esta manera se soluciona este problema popularmente conocido como *el problema del viajante* [31] dando el recorrido más corto posible para el medio de transporte elegido en la petición al servicio.

3.2. Objetivos de Desarrollo Sostenible (ODS)

Los *Objetivos de Desarrollo Sostenible* son una lista de 17 objetivos globales adoptados por la Asamblea General de la *Organización de las Naciones Unidas (ONU)* el 25 de septiembre de 2015, como parte de la Agenda 2030 para el Desarrollo Sostenible. Según la **ONU** [30], “*los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años*”.



Figura 3.1: Objetivos de Desarrollo Sostenible

Los **ODS** suceden a los *Objetivos de Desarrollo del Milenio (ODM)*, que fueron formulados en el año 2000 y finalizados en 2015. A diferencia de los **ODM**, que se enfocaban principalmente en los países en desarrollo, los **ODS** son universales y están diseñados para ser aplicados por todos los países, sin importar su nivel de desarrollo. Los *Objetivos de Desarrollo Sostenible* son la incorporación de los tres pilares del desarrollo sostenible: económico, social y ambiental, están definidos por una perspectiva universal e indivisible que tiene en cuenta las relaciones entre los tres ámbitos.

Los 17 ODS son un conjunto integral que abarca desafíos globales que afectan a la humanidad y el planeta (ver Fig. 3.1). Cada objetivo se desglosa en metas específicas y medibles. Algunos de los objetivos más relevantes para el marco de sostenibilidad y desarrollo urbano son los siguientes:

- **ODS 11: Ciudades y comunidades sostenibles:** Busca hacer que las ciudades y los asentamientos humanos sean inclusivos, seguros, resilientes y sostenibles. Dado que más del 55 % de la población mundial vive en áreas urbanas, este objetivo es crucial para mejorar la calidad de vida y reducir el impacto ambiental en las ciudades. Incluye metas como el acceso a una vivienda adecuada y asequible, el transporte sostenible, la planificación urbana inclusiva y la reducción del impacto ambiental urbano.

- **ODS 13: Acción por el clima:** Llama a tomar medidas urgentes para combatir el cambio climático y sus impactos, incluyendo la mitigación de emisiones de gases de efecto invernadero y la adaptación a los efectos adversos del cambio climático.
- **ODS 7: Energía asequible y no contaminante:** Promueve el acceso universal a energía moderna, asequible, confiable y sostenible, lo que implica la expansión de las energías renovables y la mejora en la eficiencia energética. Este objetivo es clave para reducir las emisiones de carbono y combatir el cambio climático.

ODS y tecnología GIS en la planificación urbana

El uso de tecnologías como los Sistemas de Información Geográfica (GIS) desempeña un papel crucial en la implementación y seguimiento de los ODS, especialmente en el ámbito urbano. Los GIS permiten en el campo de la planificación urbana:

- **Monitorización del desarrollo urbano:** Ayudan a analizar cómo crecen las ciudades y qué impacto tienen en el medio ambiente, lo cual es vital para cumplir con las metas de ciudades sostenibles.
- **Optimización de rutas sostenibles:** Facilitan la creación de rutas para peatones y ciclistas que minimicen el uso de vehículos motorizados, contribuyendo a la reducción de emisiones de gases de efecto invernadero.
- **Evaluación de riesgos ambientales:** Permiten identificar áreas vulnerables a riesgos naturales, como inundaciones o deslizamientos de tierra, y planificar medidas de adaptación al cambio climático.
- **Gestión de la masificación turística:** Herramientas diagnósticas como el *Sistema de Información sobre Contaminación Acústica (SICA)*, una plataforma dependiente del Ministerio de Transportes, Movilidad y Agenda Urbana [5], pueden ayudar a identificar zonas de masificación turística. Al combinar estos datos con otras fuentes como la ubicación de personas a través de sus dispositivos móviles, es posible diseñar alternativas o implementar restricciones que redistribuyan el flujo de turistas, minimizando su impacto negativo en el entorno local.

3.3. Modelos de lenguaje a gran escala (LLM)

Los modelos de lenguaje a gran escala son un tipo de inteligencia artificial que ha sido entrenada para realizar tareas de *Procesamiento del Lenguaje Natural (NLP)*. Estas inteligencias artificiales son entrenadas con ingentes cantidades de datos que los hacen capaces de comprender peticiones y responder a éstas en los mismos términos de lenguaje, generando una comunicación entre el usuario y la máquina.

Técnicas de Ingeniería del prompt

En el desarrollo de este TFG, se han implementado y evaluado diversas técnicas de Ingeniería del prompt con el fin de mejorar la interacción y los resultados generados por los LLM. A continuación, se detallan las principales técnicas empleadas durante este proceso. Para obtener más información sobre estas técnicas y otras estrategias de Ingeniería del prompt, se puede consultar la guía disponible en *Prompting Guide* [25].

Zero-shot learning se trata de una técnica en la que el usuario no facilita al modelo ningún ejemplo de cómo realizar una tarea. El LLM por tanto interpreta basado en el contexto y su propio entrenamiento lo que se ha requerido y responde acorde a estos datos. Esta técnica se usa cuando lo que se prioriza es la rapidez del modelo frente a la precisión de la salida aportada. Cuando se requiere una mayor exactitud en la interacción, es recomendable emplear la técnica de *Few-shot learning*, que consiste en proporcionar al modelo algunos ejemplos en el prompt para guiarlo hacia una respuesta mejorada según el criterio del desarrollador.

Una vez que se logra establecer el formato de las respuestas, es importante asegurar que el modelo sea capaz de interactuar con herramientas externas o APIs. Aquí es donde entra en juego la técnica de *Tool calling (Function calling)*. Se trata de una habilidad clave para construir chatbots o agentes potenciados por LLM que necesitan recuperar contexto o interactuar con herramientas externas convirtiendo el *Procesamiento del Lenguaje Natural* en llamadas a APIs. Este enfoque de *Tool calling (Function calling)* **permite una mejora en la modularidad del proyecto**. Al separar el nivel de abstracción, permite por ejemplo cambiar el modelo LLM de origen sin afectar al funcionamiento general del sistema, ya que la estructura de la salida permanece constante.

A continuación, se ilustra un ejemplo de cómo se nutre al modelo con una plantilla que sirve de estructura para la respuesta, obteniendo de manera consistente datos en formato JSON. Este ejemplo se puede encontrar en el [prototipo Python](#) del proyecto:

```

1      # Un ejemplo de respuesta que queremos obtener seria la
2      siguiente:
3      plantilla_salida = """[{
4          "nombre": "Sagrada Familia",
5          "gps": "41.4036, 2.1744",
6          "descripcion": "Basilica catolica disenada por Gaudi."
7      },
8      {
9          "nombre": "Parc Guell",
10         "gps": "41.4145, 2.1527",
11         "descripcion": "Parque disenado por Gaudi."
12     }]"""
13
14     ciudad = "Salamanca"
15     complete_and_print(f"""Tu rol es un guia turistico
16         comprometido con el medio ambiente preocupado por la
17         gentrificacion de las ciudades y el turismo masivo.
18         Dime 3 sitios para visitar en {ciudad}. Quiero que la
19         respuesta consista SOLO en los lugares de interes de
20         este lugar, siguiendo el ejemplo de la {
21         plantilla_salida}. Insisto en que no quiero que me
22         des informacion adicional.""")
23
24     # Resultado esperado:
25     [{
26         "nombre": "Catedral Vieja de Salamanca",
27         "gps": "40.9708, -5.6619",
28         "descripcion": "Edificio gotico del siglo XIII."
29     },
30     {
31         "nombre": "Puente de San Esteban",
32         "gps": "40.9713, -5.6624",
33         "descripcion": "Puente medieval que cruza el rio
34             Tormes."
35     },
36     {
37         "nombre": "Jardines de la Alamedilla",
38         "gps": "40.9694, -5.6632",
39         "descripcion": "Jardin barroco en el corazon de la
40             ciudad."
41     }]

```

Código 3.1: Ejemplo de uso de *few-shot* y *tool-calling* para obtener puntos de interés turísticos

Retrieval-Augmented Generation (RAG)

Generación Aumentada por Recuperación o **RAG** es una técnica usada en modelos de inteligencia artificial en la cual se obtiene información para nutrir a un *Modelos de Lenguaje de Gran Escala* que ya ha sido entrenado, de esta manera amplía su conocimiento y es capaz de generar una respuesta más precisa, actualizada y completa.

El problema que subyace en los modelos tradicionales es que una vez alimentados con un conjunto de datos, sufren de un aislamiento del mundo que los rodea. Para prevenir este problema se nutre de información que el usuario facilita siguiendo los siguientes pasos:

1. **Splitter y Tokenización**: La información proporcionada al modelo se divide en tokens, donde cada modelo tiene un límite máximo de tokens que puede procesar como contexto. Además, algunos modelos pueden cobrar al usuario por cada token procesado. Por lo tanto, es fundamental transformar una cadena de texto larga en fragmentos más pequeños y optimizados, utilizando tokens específicos para facilitar la comprensión por parte del modelo. Esta división previa es un paso necesario antes de recuperar la información.
2. **Embeddings**: consiste en transformar la información facilitada y representarla en vectores de n dimensiones. Para ello se usa comúnmente otro modelo entrenado para transformar la información en vectores. En la Figura 3.2 se puede ver cómo la información de Wikipedia API [2] se prepara para guardarse en una base de datos vectorial Astra DB [1] para ello usa el modelo Ollama Embeddings

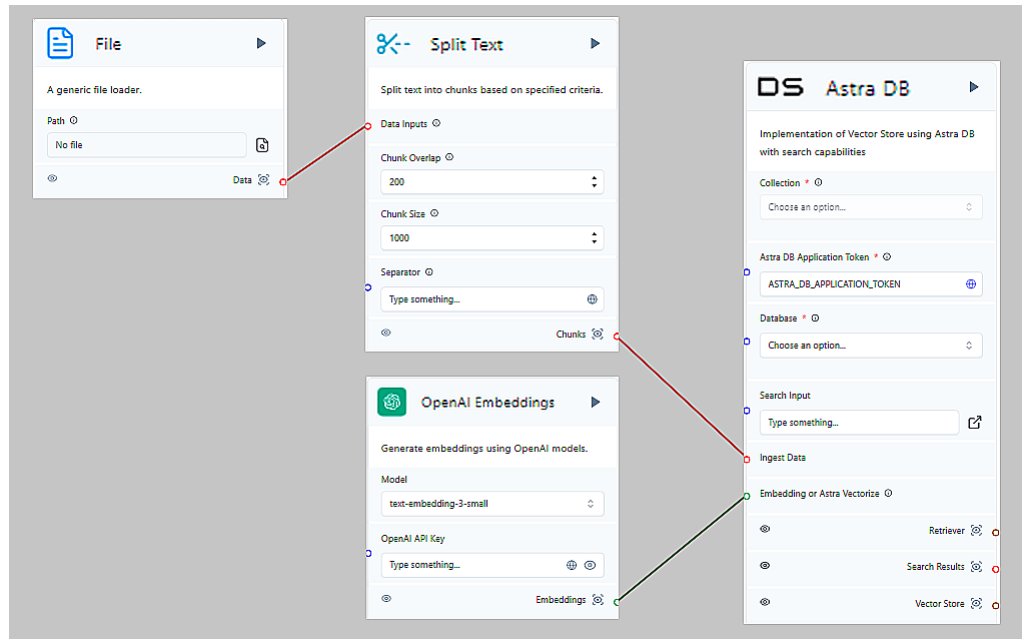


Figura 3.2: Preparación de la información de un RAG mostrada en la herramienta Langflow

3. **RAG**: con la información ampliada ya vectorizada en una base de datos, el usuario genera una entrada o prompt al modelo. Como se ve en la Figura 3.3 el LLM OpenAI en este caso selecciona la información más afín de los datos aportados para generar así un prompt ampliado o mejorado que será facilitado al modelo para un procesamiento de información habitual, consiguiendo así un mejor resultado.

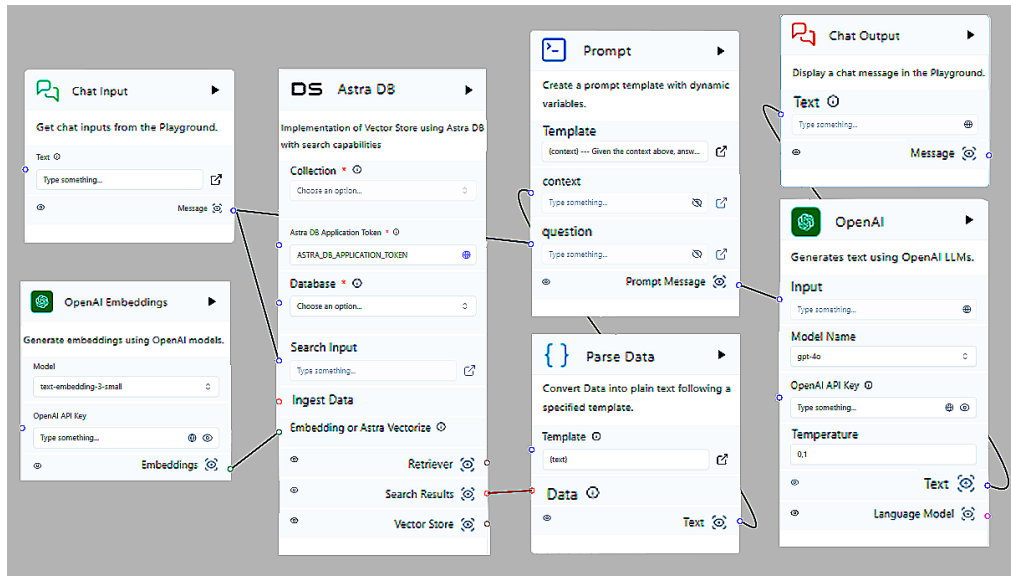


Figura 3.3: Chat Output a través de la salida de un RAG

Agentes

La información que alimenta a los RAG puede ser un archivo de texto con información general de un tema sin embargo hay veces en los que la información no está físicamente en un archivo y se tiene que obtener a través de **Agentes**. Son aplicaciones que alimentarán al modelo con un conjunto de herramientas tales como motores de búsqueda, bases de datos, páginas web, etc. Una vez provisto con esta información, el modelo es capaz de razonar acerca de las acciones que debe cumplir para obtener el mejor resultado.

Uso de *Modelos de Lenguaje de Gran Escala* en *Eco City Tours*

En este proyecto, los *Modelos de Lenguaje de Gran Escala* se han empleado para obtener los *Puntos de Interés* (PDI) a través de un diálogo interactivo entre la *Inteligencia Artificial* y el usuario, donde este último solicita recomendaciones de lugares enfocados en el turismo sostenible. En la sección de **prototipos**, además de un prototipo con Langflow, se implementa un cuaderno de Python que evalúa las mejoras en las respuestas aplicando diversas técnicas de prompting [25]. Inicialmente, el prototipo presenta una conversación básica con respuestas de baja calidad o incluso alucinadas, que progresivamente mejoran al incluir estructuras de datos predefinidas en las

respuestas. Este enfoque permite que la aplicación aproveche el aprendizaje, generando recomendaciones más precisas y acordes al código requerido en la aplicación móvil.

En cuanto a los *Retrieval Augmented Generation (RAG)*, su mayor utilidad suele encontrarse en la mejora de chatbots empresariales, donde se busca ofrecer una atención al cliente más eficiente y precisa. En el contexto de los prototipos desarrollados para este TFG, se realizaron pruebas con sistemas RAG que recopilaban información a partir de búsquedas en Internet. Las páginas web resultado de las búsquedas eran después tratadas para, una vez procesada su información, incluirla en el contexto del prompt, lo que permitía al modelo generar respuestas más completas y actualizadas. Sin embargo, los resultados de estas pruebas revelaron que, aunque el sistema proporciona un contexto enriquecido y relevante, los tiempos de respuesta se incrementan significativamente, lo que podría afectar negativamente la experiencia del usuario. Por esta razón, **se decidió no implementar estas herramientas en la versión final de la aplicación**, aunque los prototipos quedan disponibles como experimentos, con el fin de inspirar a otros desarrolladores interesados en explorar esta tecnología.

3.4. Agilidad y método SCRUM

La gestión ágil de proyectos de desarrollo de software tiene como pilares fundamentales la flexibilidad, la colaboración y la entrega continua de valor al cliente [4]. La agilidad se basa en la idea de ciclos de trabajo cortos e iterativos denominados **sprints**, donde se produce software funcional al final de cada iteración.

El Manifiesto Ágil (2001) establece cuatro valores fundamentales:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcional sobre documentación exhaustiva.
- Colaboración con el cliente sobre negociación de contratos.
- Respuesta ante el cambio sobre seguir un plan.

SCRUM es un marco de trabajo ágil que facilita la auto-organización del equipo, promueve la transparencia y mejora la capacidad de adaptación del proyecto.

Este enfoque es particularmente valioso en el desarrollo de *Eco City Tours*, iterando rápidamente sobre incrementos funcionales de la aplicación, validando con pruebas de usuario con prototipos funcionales y ajustando el desarrollo a las necesidades que surgían durante el proceso. El tutor ha jugado los tres roles definidos en SCRUM: usuario para validar los incrementos funcionales, equipo cuando revisa contenido de las tareas y SCRUM master para ayudar a aplicar la agilidad. El rol principal del estudiante y autor de trabajo es el de equipo. Ambos participantes se han reunido para:

- **Planificación de Sprint:** consiste en la creación y asignación de tareas que se realizarán durante el período de tiempo en que consiste el sprint. En este TFG la duración de los sprints ha sido entre dos y tres semanas. A cada tarea se le asigna unos puntos de historia que reflejan el tiempo planificado para realizar dicha tarea. También se planifican etiquetas e hitos para los que las diferentes tareas ayudan a lograr.
- **Sprint review y sprint retrospective:** después de cada sprint y con la ayuda del tutor del TFG, se presenta el incremento funcional y se revisan las tareas realizadas durante el sprint. Además, estas reuniones incluyen una evaluación continua de la calidad del desarrollo y los procesos asociados.
- **Integración continua con pruebas y compilación:** como parte del enfoque de calidad continua, se integraron prácticas de Integración Continua (CI) durante el ciclo de desarrollo. Estas prácticas incluyen la **automatización de pruebas y la compilación del software** utilizando GitHub Actions. En particular, se ejecutaron flujos de trabajo que realizan análisis automatizados del código con SonarCloud [27], garantizando la detección temprana de errores y problemas de estilo o rendimiento en el código como se muestra en la figura 3.4. Esto asegura que cada incremento funcional presentado en la *sprint review* sea estable y cumpla con los estándares de calidad establecidos.

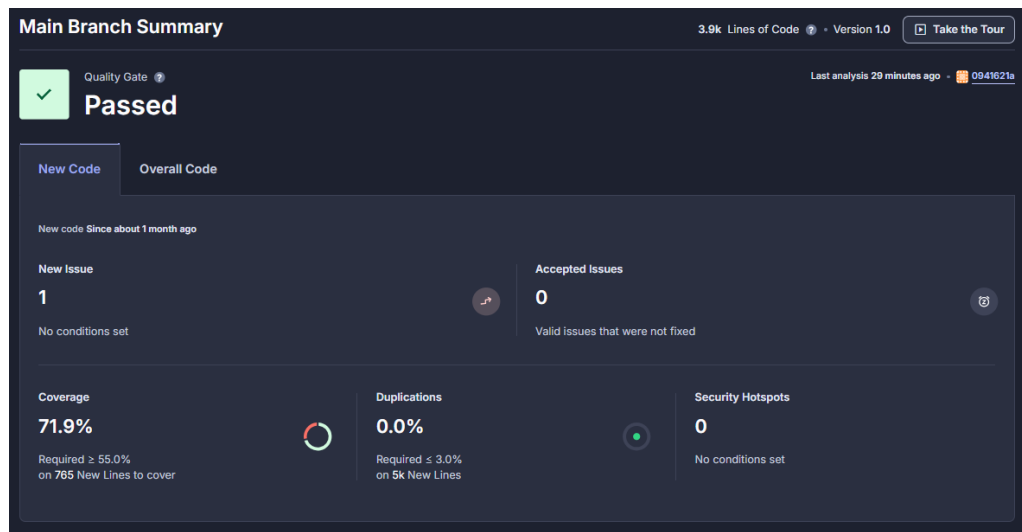


Figura 3.4: Resumen de la compilación del proyecto en SonarCloud

La combinación de la sprint review con prácticas de integración y calidad continua refuerza el principio ágil de entrega de valor constante y garantiza que el software presentado al finalizar cada sprint sea de calidad, estable y funcional.

El uso de la metodología ágil y el marco SCRUM fue esencial para el éxito del desarrollo de *Eco City Tours*. La agilidad permitió una colaboración constante y una entrega continua de valor, mientras que SCRUM proporcionó la estructura necesaria para gestionar eficazmente el trabajo, resolver problemas y optimizar el proceso de desarrollo en cada iteración.

4. Técnicas y herramientas

El siguiente capítulo presenta las técnicas y herramientas usadas a lo largo del desarrollo de la aplicación *Eco City Tours*. Se detallarán los aspectos más destacados de cada una de ellas, justificando el porqué de su utilización sobre otras alternativas si las hubiera. Para mejorar la comprensión de la sección las herramientas se han agrupado en tres subsecciones. La primera relacionadas con el desarrollo con modelos **LLM**, la segunda con las herramientas utilizadas en el desarrollo de la aplicación móvil, y la tercera con las empleadas para la gestión del proyecto.

4.1. Desarrollo relacionado con LLM

LM Studio

LM Studio [16] es una aplicación enfocada en el despliegue de modelos de lenguaje. Su principal objetivo es facilitar la experimentación con *Modelos de Lenguaje de Gran Escala* ofreciendo un entorno completo y que integra funcionalidades en el procesamiento de lenguaje natural. Estas son algunas de las más destacadas:

- **Búsqueda y despliegue de modelos en local:** a través de un buscador intuitivo, los usuarios pueden seleccionar, descargar y desplegar cualquier modelo disponible en plataformas como Hugging Face, facilitando su uso local.
- **Servidor local de un el modelo:** LM Studio permite crear un servidor local que gestione peticiones de información. Esto es especial-

mente útil para automatizar procesos que requieran el procesamiento de lenguaje o la gestión de grandes volúmenes de datos.

- **Entrenamiento y ajuste de modelo:** LM Studio simplifica el proceso de fine-tuning o ajuste fino, que consiste en optimizar un modelo pre-entrenado mediante conjuntos de datos específicos y tareas concretas para mejorar su rendimiento. Gracias a su interfaz gráfica, esta tarea se vuelve más accesible para los desarrolladores, eliminando gran parte de la complejidad técnica.
- **Optimización y monitorización:** permite observar el rendimiento de los modelos analizando su consumo de GPU, memoria o CPU así como su tiempo de respuesta, lo que permite realizar al usuario ajustes con el fin de mejorar el rendimiento del sistema.

La experimentación con diferentes modelos para decidir cuál usar en *Eco City Tours* resultó mucho más rápida y eficiente gracias a esta herramienta, en comparación con el uso inicial de un solo modelo a través de Ollama [24].

LangFlow

LangFlow es una herramienta de código abierto [15] que proporciona una *Interfaz Gráfica de Usuario (GUI)*, permitiendo la interacción con *Modelos de Lenguaje de Gran Escala* (Modelos de Lenguaje a Gran Escala) sin necesidad de escribir código. Los usuarios pueden configurar componentes y módulos de forma visual, conectándolos entre sí como si se tratara de un diagrama de flujo, facilitando la creación de aplicaciones complejas de procesamiento de lenguaje.

La aplicación se puede instalar localmente, donde se ejecuta como un servicio en un puerto específico del localhost, permitiendo interactuar con la interfaz a través de un navegador web, como si fuera una aplicación web estándar.

- **Integración con múltiples LLM y herramientas de terceros,** existe una gran variedad de modelos con los que se puede interactuar con solo configurar sus claves API, pero también herramientas como agentes o tokenizadores, lo que amplía las posibilidades de personalización y experimentación.
- **Herramienta de exportación e importación,** la plataforma permite exportar el flujo de trabajo generado a un archivo JSON, lo que facilita la portabilidad del proyecto.

- **Prototipos iniciales**, se puede encontrar en el arranque de la aplicación proyectos por defecto que facilitan el desarrollo. Por ejemplo, si se quiere conseguir un sistema RAG, se puede partir de una plantilla ya creada que permite ahorrar tiempo a la hora de personalizar un prototipo personal.

En resumen, la principal característica que hace a esta herramienta tan potente es que favorece la experimentación y configuración personal de los elementos que componen un prototipo que usa interacciones con modelos **LLM**. Esto la convierte en una herramienta ideal para desarrolladores que están empezando a explorar el mundo de los **LLM**, ya que su enfoque visual ofrece una ventaja significativa frente a alternativas más técnicas, como los cuadernos Jupyter. Además, en el futuro próximo se presenta como una herramienta docente donde presentar a los alumnos los *Modelos de Lenguaje de Gran Escala* llevando su utilización a campos como el tratamiento de la información, machine-learning o automatización de procesos.

4.2. Desarrollo aplicación móvil

Se utilizó el framework Flutter [9] para desarrollar *Eco City Tours*. El entorno de desarrollo elegido fue Visual Studio Code, que ofrece una amplia gama de extensiones y herramientas que facilitan el proceso de desarrollo.

Patrón de diseño **Business Logic Component (BLoC)**

Un aspecto fundamental en una aplicación móvil reactiva es la gestión del estado, cuyo propósito es automatizar la actualización de las vistas cuando los valores de la lógica de control cambian. Flutter ofrece diversos gestores de estado, como *Provider*, *Cubit*, *GetX*, entre otros. Sin embargo, BLoC destaca como un paquete **Flutter Favorite** [7], lo que lo sitúa como una opción preferida debido a su soporte activo, calidad de código, seguridad y frecuencia de actualizaciones.

Entre todos los gestores aprendidos durante mi formación, BLoC resultó ser una opción muy robusta. Una vez comprendida su sintaxis, se aprecia su facilidad de uso y la modularidad de sus elementos, que se estructuran en eventos, estados y definiciones de BLoCs.

- **Gestión clara del estado**: BLoC proporciona una manera estructurada de gestionar los diferentes estados de la aplicación. Cada cambio

de estado es manejado a través de eventos, lo que permite una clara separación entre la lógica de negocio y la presentación visual.

- **Escalabilidad:** el uso del diseño reactivo orientado a eventos, característico del patrón BLoC, facilita la gestión del estado de la aplicación y permite que esta crezca en funcionalidad sin comprometer su rendimiento. Cada nueva funcionalidad puede ser controlada por un BLoC independiente, lo que asegura que el sistema se incremente de manera estructurada.
- **Mantenibilidad:** gracias a las extensiones de Visual Studio Code, es posible generar nuevos BLoCs fácilmente. Esto simplifica el desarrollo de nuevas funcionalidades. La clara separación de responsabilidades entre los BLoCs individuales contribuye a un código modular y fácilmente mantenible.
- **Reutilización de lógica:** Uno de los beneficios clave de BLoC es que la lógica de negocio se puede reutilizar fácilmente en diferentes partes de la aplicación. Esto facilita la implementación de componentes que comparten comportamientos similares sin duplicar código. Para lograrlo, se utiliza el *BLoCProvider*, un widget que comparte el BLoC y permite que los widgets hijos accedan a su lógica de estado. De esta manera, cualquier widget dentro del contexto del BLoCProvider puede interactuar con el BLoC correspondiente.

Ejemplo de uso de **BLoC** en *Eco City Tours*

El **BLoC** ubicado en `gps_bloc.dart` gestiona el permiso de uso del GPS de la aplicación y la activación del sensor GPS del móvil. Se aplica el patrón de diseño Observador, estudiado en la asignatura *Diseño y Mantenimiento del Software*, que es uno de los patrones fundamentales descritos por Gamma et al. [10]. El **BLoC** actúa como el Sujeto del patrón Observador, gestionando el estado de si el GPS está activado y si los permisos de ubicación han sido concedidos.

Los widgets que dependen de esta información, como los mapas o botones de la interfaz de usuario, actúan como Observadores. Cada vez que el estado del **BLoC** cambia, estos widgets son notificados y se actualizan automáticamente, permitiendo que la interfaz reaccione en tiempo real a los cambios en el estado del GPS o los permisos. Cada vez que el estado del GPS o de los permisos cambia, se dispara un evento *OnGpsAndPermissionEvent*, que actualiza el estado y notifica a los widgets correspondientes. Esto permite

```
1      // Indicates whether GPS is enabled
2      final bool isGpsEnabled;
3
4      // Indicates whether the app has permission to access
5      // GPS
6      final bool isGpsPermissionGranted;
7
8      // Getter for the correct state of permissions and
9      // active GPS.
10     bool get isAllReady =>
11         isGpsEnabled &&
12         isGpsPermissionGranted;
```

Código 4.2: Definición de variables de control en el BLoC

que la interfaz de usuario reaccione dinámicamente y muestre la información correcta según el estado actual.

```
1     class LoadingScreen extends StatelessWidget {
2       const LoadingScreen({super.key});
3
4       @override
5       Widget build(BuildContext context) {
6         return Scaffold(
7           body: BlocBuilder<GpsBloc, GpsState>(
8             builder: (context, state) {
9               return state.isAllReady
10                 // If GPS and permissions are ready, show the map;
11                 // otherwise, show the GPS access screen
12                 ? const MapScreen()
13                 : const GpsAccessScreen();
14             },
15           ),
16         );
17     }
```

Código 4.3: Carga en función del estado

Este ejemplo muestra la comunicación entre lógica de control e interfaz gráfica usando el patrón Observador con BLoC.

Extensiones de Visual Studio Code más destacadas

Se citan a continuación brevemente algunas de las extensiones que facilitaron el desarrollo de la aplicación aumentando la producción:

- **Pubspec Assist:** las dependencias de librerías incluidas en el archivo `pub_spec.yaml` son automáticamente instaladas, ordenadas y gestionadas en definitiva por este asistente que ahorra múltiples comandos en consola mejorando la rapidez a la hora de programar.
- **Snippets:** de manera análoga los snippets de flutter permiten programar rápidamente estructuras del código repetitiva.

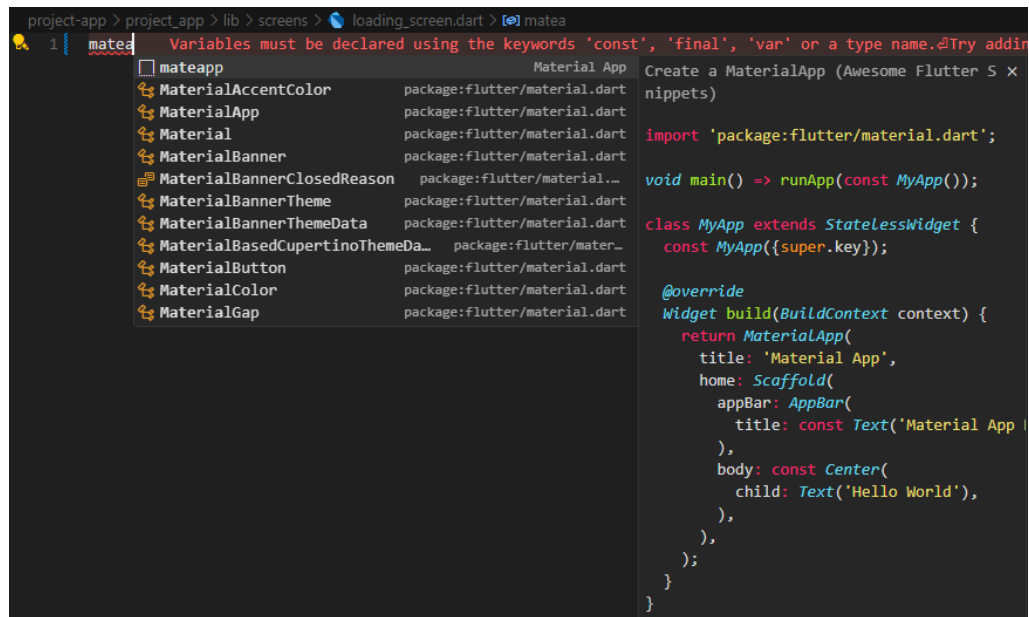


Figura 4.1: Uso del snippet: mateapp

Sirva de ejemplo la figura 4.1 donde se ve que sin llegar a escribir `mateapp` si pulsamos la tecla tabulador se generará todo el código asociado a la derecha y nos dejará el cursor en los campos `MyApp` para así cambiar el campo a continuación mejorando la velocidad de construcción del código.

Github Copilot

Github Copilot y Github Copilot Chat [33] son dos extensiones de Visual Studio Code que se obtienen gratuitamente entre otras ventajas al identificarse como estudiante en GitHub. Dado que este TFG promueve el uso de *Inteligencia Artificial (IA)* y de los LLM, se decidió incorporar GitHub Copilot para mejorar la eficiencia en el desarrollo. Copilot es una herramienta que ofrece sugerencias automáticas mientras se escribe código (ver Fig. 4.2),

previando las próximas líneas y ofreciendo opciones que ahorran tiempo al programador. La extensión Copilot Chat permite la interacción directa con la IA, de manera que se pueden hacer preguntas sobre el código sin necesidad de copiarlo en el chat, ya que el código del archivo abierto se incluye automáticamente en el contexto de la comunicación con Copilot.

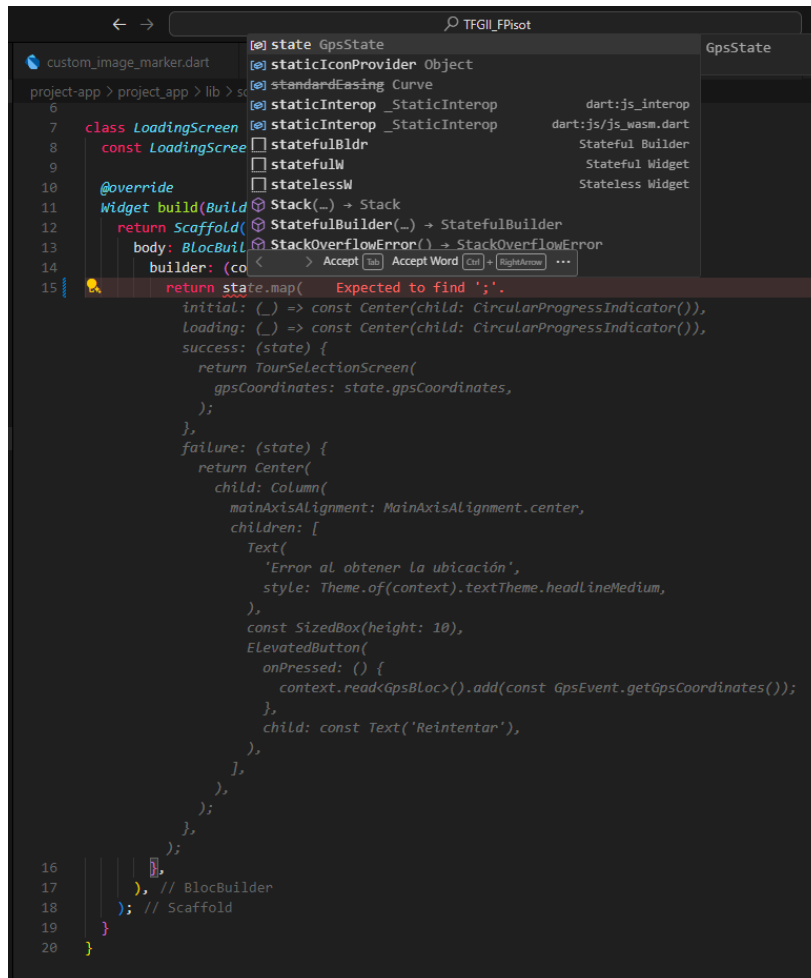


Figura 4.2: Ejemplo de *Ghost Text* de Copilot

La ventaja principal es el incremento de productividad: como se muestra en la Figura 4.2, Copilot es capaz de generar prácticamente una clase entera o un widget al interpretar correctamente las primeras líneas de código, evitando escribir manualmente un fragmento largo de texto.

Expectativas vs Realidad.

Es responsabilidad del programador mantener el control total sobre su

código. Aunque en ocasiones Copilot acierta completamente, solucionando problemas de forma eficiente, también puede provocar inconsistencias si se aplican sugerencias sin la supervisión adecuada. En algunos casos, las sugerencias automáticas pueden alterar partes del código que no estaban relacionadas con el problema original, introduciendo nuevos errores.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: BlocBuilder<GpsBloc, GpsState>(
      builder: (context, state) {
        return state.isAllReady // Si el GPS y los per
          ? const MapScreen()
          : const GpsAccessScreen();
      },
    ), // BlocBuilder
  ); // Scaffold
}
```

Figura 4.3: Código real a utilizar en vez de propuesta Copilot

Evidentemente es una tecnología en evolución que irá mejorando pero sirva de ejemplo la Figura 4.3 para ilustrar el código real usado de apenas unas líneas frente a la propuesta de Copilot de la Figura 4.2.

En resumen, por el momento Copilot resulta muy útil en escenarios donde se trabaja con patrones de código comunes o tareas repetitivas, pero es menos efectivo cuando se trata de resolver problemas más específicos. Aunque no representa un cambio en el paradigma de la programación, sí puede convertirse en una herramienta habitual que, con el tiempo, aumente la productividad del desarrollador.

4.3. Gestión de proyectos

GitHub

GitHub es una plataforma en la nube para el alojamiento y gestión de código fuente, que se basa en el sistema de control de versiones Git. A lo largo de la carrera de Ingeniería Informática, ha sido utilizada en diversas asignaturas, lo que la convierte en una herramienta fundamental para el desarrollo de proyectos. En el contexto de este TFG, GitHub ha sido esencial para centralizar y gestionar el código generado en varios IDE, permitiendo un control de versiones eficiente, la conservación del trabajo y una colaboración estructurada en la nube. Algunas de sus características principales usadas en el desarrollo de *Eco City Tours* fueron las siguientes:

- **Control de versiones con Git:** gracias a los commits realizados a lo largo del desarrollo se permite comprobar la evolución de un proyecto así como volver a estados del trabajo gestionados en sus ramas. Otras herramientas como pull-requests permiten solicitar cambios al código facilitando un uso colaborativo durante la gestión del proyecto.
- **Integración continua y análisis de calidad:** GitHub se integra fácilmente con una amplia variedad de herramientas de desarrollo, como servicios de CI/CD y plataformas de despliegue, lo que permite automatizar tareas y mejorar el flujo de trabajo. En este proyecto, se definieron acciones en *GitHub Actions* que ejecutan pruebas automatizadas y análisis de calidad del código mediante *SonarCloud*. Estas acciones garantizan un estándar elevado de calidad del software, detectando errores, problemas de estilo y optimización como se observa en la Figura 4.4.



Figura 4.4: Control de calidad de SonarCloud respecto a issues abiertas.

- **Documentación y wikis:** GitHub permite mantener documentación clara y estructurada directamente en el repositorio, facilitando la creación de archivos README para proporcionar información detallada sobre el proyecto.
- **Gestión de tareas y entrega de releases:** GitHub facilita la planificación y el seguimiento del progreso del proyecto mediante el uso de issues y milestones. Estas herramientas permiten gestionar tareas asignándoles prioridades y etiquetas, lo que proporciona una vista clara del flujo de trabajo. GitHub ofrece también una funcionalidad de releases que permite empaquetar y distribuir versiones finales o intermedias del software de forma eficiente, asegurando una entrega organizada y documentada.

Al trabajar desde diferentes equipos GitHub fue especialmente útil en el desarrollo del proyecto: se utilizó la integración de Visual Studio Code con Github para los cambios en la programación de la aplicación y los prototipos en cuadernos Jupyter y se usó la extensión Git Graph (ver Fig. 4.5) para comprobar el funcionamiento y navegación a través de los commits cuando fue necesario. Para realizar commits en los cambios de la documentación se usó GitHub Desktop una vez realizados los cambios con la IDE TeXstudio.

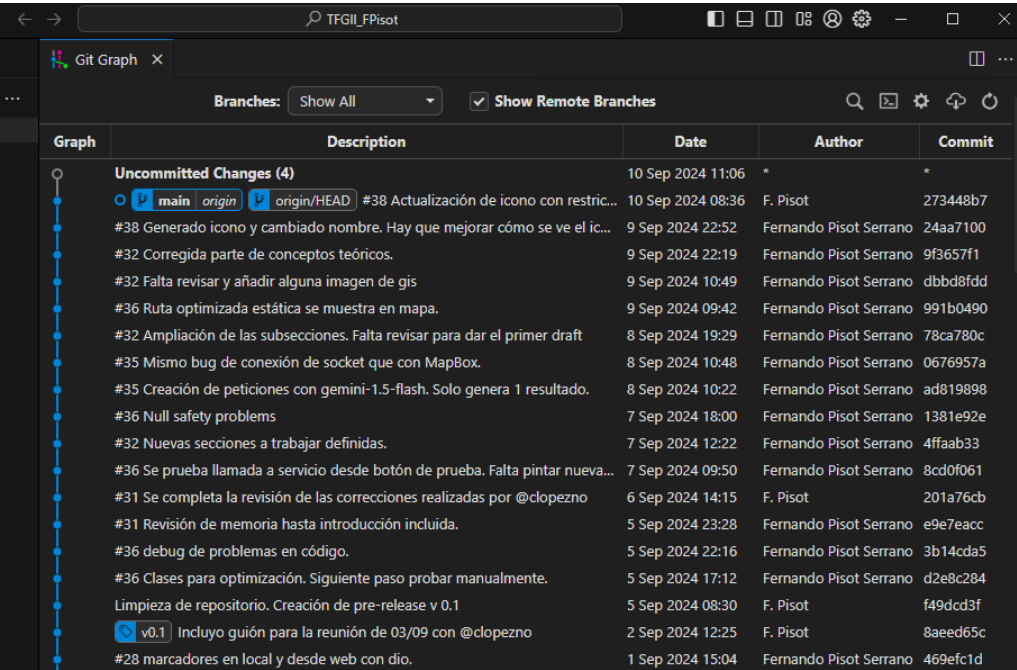


Figura 4.5: Uso de GitHub con la extensión de VSCode Git Graph

Zube

Zube [35] es una plataforma de gestión de proyectos con un enfoque colaborativo durante el seguimiento del desarrollo del mismo. Las características principales son:

- Versión gratuita: Zube ofrece un plan gratuito que cubre las necesidades básicas de gestión de proyectos autogestionados.
- Conexión con GitHub: su integración con el repositorio del proyecto para su seguimiento es una condición indispensable. Además, La sincronización es inmediata lo que produce por ejemplo que al crear una tarea, inmediatamente esté disponible para su gestión desde Zube.
- Tableros Kanban: se facilita la visualización del flujo de trabajo. Las tarjetas de tareas pueden moverse fácilmente entre columnas como *Pendiente*, *En progreso* y *Completado*, lo que permite un seguimiento claro y eficaz del estado de cada tarea.
- Herramientas gráficas para control de los sprints: otra manera de consultar el flujo de trabajo durante los sprints con gráficos burndown, burnup, throughput o de velocidad.

El uso de esta aplicación fue vital a lo largo del proyecto, ya que se utilizó principalmente para el control de sprints y sus tareas en el panel Kanban y la asignación con puntos de historia. La integración con GitHub permitió un flujo de trabajo eficiente, manteniendo el código fuente y la gestión de tareas en perfecta sincronización, lo que favoreció un desarrollo ágil y organizado.

TeXstudio

Para la elaboración de la documentación de este **TFG**, se ha optado por utilizar TeXstudio, un *Entorno de Desarrollo Integrado (IDE)* especializado en la edición de textos en LaTeX. Esta herramienta facilita la redacción de trabajos académicos y científicos mediante características como el resaltado de sintaxis, corrección ortográfica y semántica en tiempo real, y la posibilidad de compilar el documento durante su edición, lo que permite visualizar el resultado final en formato PDF de manera inmediata.

TeXstudio es una herramienta multiplataforma, y se seleccionó por encima de otras alternativas debido a su facilidad de uso, interfaz intuitiva, y la consola de errores integrada, que simplifica la detección de fallos en la secuencia de comandos LaTeX.

Una de las características más destacadas es la funcionalidad de **compilación automática al editar**, la cual permite previsualizar continuamente el documento mientras se trabaja, lo que mejora la productividad al ofrecer un resultado inmediato sobre los cambios realizados.

5. Aspectos relevantes del desarrollo del proyecto

Durante el desarrollo de un *Trabajo de Fin de Grado*, es inevitable tomar decisiones que impactan de manera significativa en el resultado final. En este capítulo se describen las decisiones clave que han influido en la evolución y estado final de *Eco City Tours*. La justificación de estas decisiones se espera que pueda servir como una referencia útil para futuros compañeros o desarrolladores que enfrenten objetivos similares.

5.1. Elección de servicios Google sobre tecnología *Open Street Maps*

Desde un comienzo en el proyecto se quería utilizar **código abierto**, pues su filosofía se alinea mejor con los valores aprendidos en la Universidad, donde se promociona el uso de herramientas que no supongan un coste para el estudiante, se fomenta su uso evitando la posible discriminación económica y una forma de trabajar colaborativa.

De manera renuente se decide cambiar los servicios necesarios para la visualización y gestión de marcadores y rutas a los establecidos por Google. Los motivos que propiciaron este gran cambio fueron los siguientes:

- **Soporte de un gigante tecnológico:** las herramientas de código abierto aunque algunas tienen un gran seguimiento por la comunidad no pueden competir con la documentación, ejemplos de desarrolladores y tecnología de uso de una potencia como Google.

- **Integración:** Flutter forma parte de Google, lo que supone una integración nativa que hace de su funcionamiento y robustez una de las herramientas usadas.
- **Riesgos asociados a complementos de terceros:** Durante el desarrollo, se exploraron soluciones como *Open Source Routing Machine (OSRM)* para gestionar rutas, pero estas requerían procesar manualmente las conexiones con el servicio o confiar en paquetes de terceros. Estos paquetes, aunque facilitan el desarrollo, presentan un riesgo mayor debido a su posible discontinuidad o incompatibilidad con futuras actualizaciones. En cambio, el uso de herramientas como Dio [6], un paquete marcado como favorito por Flutter, garantiza un soporte nativo y más estable en el ecosistema de Google.

5.2. Elección de google generative ai como primer LLM

Durante el desarrollo se estudiaron múltiples opciones a la hora de obtener la información desde un modelo LLM. Siguiendo con el ecosistema de Google se probó con éxito el modelo *gemini-1.5-flash* gracias al paquete *google_generative_ai* que permite configurar el modelo desde su página web para más tarde exportar la configuración a un archivo Dart. Sus resultados se ajustaban dinámicamente a las preferencias del usuario. Tras mejorar la precisión de las coordenadas GPS, los datos generados comenzaron a reflejar ubicaciones correctas de manera consistente.

5.3. Aclaración sobre el coste de Servicios Google

Para la versión inicial de la aplicación de este TFG, se optó por utilizar MapBox [17], una plataforma de mapas con características similares a las de Google. Sin embargo, a medida que avanzaba el desarrollo, se fueron integrando cada vez más servicios de Google, hasta el punto de reemplazar el servicio de optimización de rutas por el de Google. En cada cambio se evaluaba cuidadosamente si el número de peticiones mensuales superaría el límite que generaría costos adicionales para el desarrollador. No obstante, tras estimar que el número de peticiones estaría muy por debajo del umbral que implicaría un coste, se decidió aprovechar las ventajas de trabajar dentro de un mismo ecosistema. Como se muestra en la Figura 5.1 las peticiones

5.4. PUNTOS DE INTERÉS CON LLM: ALUCINACIONES Y OPTIMIZACIÓN DE RUTAS TURÍSTICAS

33

de los diferentes servicios distan mucho de las 40.000 peticiones que permite gratuitamente Google durante el desarrollo de sus aplicaciones.

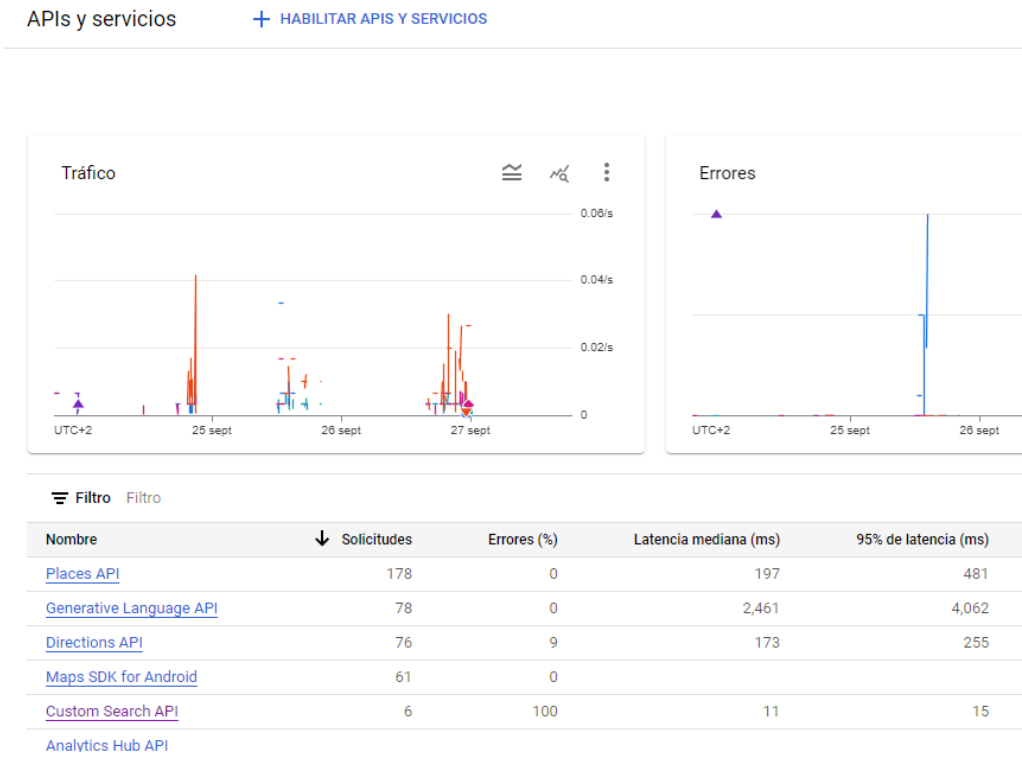


Figura 5.1: Estadísticas de solicitudes de servicios

5.4. Puntos de interés con LLM: alucinaciones y optimización de rutas turísticas

Durante las exhaustivas pruebas realizadas con el modelo flash de Gemini se observó que las direcciones web obtenidas desde el modelo tanto para las imágenes de los **PDI** como para la información eran casi siempre no válidas. Quizá porque en el tiempo desde que se publicó el modelo hasta su uso, esta información quedó desactualizada o bien simplemente porque el modelo en su afán por aportar una respuesta la alucinaba. Para obtener mejor información de los *Puntos de Interés* se recurrió al ecosistema Google optando por Google Places.

Elección de Google Places

Se consideró utilizar otros servicios de obtención de imágenes de los **PDI** en Internet, como Unsplash [32], o incluso la API de búsqueda de Google. Sin embargo, en términos de coste, resultaba preocupante que estas alternativas incurrieran en gastos tras superar el límite de 100 peticiones diarias. Finalmente, se descartaron estas opciones y se adoptó como solución el servicio de Google Places, que utiliza la misma clave API que se emplea para la gestión de mapas. Este servicio, además de proporcionar más información sobre los *Puntos de Interés*, ofrece una imagen fiable que se puede almacenar en caché gracias al paquete de flutter `cached_network_image`.

Personalización de LLM

A la hora de obtener información que procesar por el método **RAG** se valoraron muchas fuentes de datos. Uno de los orígenes de datos de información turística favoritos de los usuarios es Tripadvisor. Contar con la información actualizada de este gigante turístico suponía un gran aliciente. Sin embargo se desestimó su uso por varias razones: la información se podía obtener a través del método scraping o webscraping que toma la información en bruto de la página web y se podía postprocesar. Dicha práctica incumpliría los Términos de Uso del Servicio, ya que Tripadvisor usa un acceso a través de API para obtener la información de su base de datos. Se necesita una forma de pago para poder empezar a usar el servicio y su utilización, si sobrepasa las 5.000 peticiones al mes, incurriría en gastos al desarrollador. Se buscaron alternativas que funcionaran de manera análoga a Tripadvisor para nutrir el **RAG**.

Descarte del modelo local con **RAG** en favor de la versión Pro de Gemini

En este proyecto, se ha realizado un prototipo con Langflow, que se puede consultar en el archivo **Langflow Explanation** para más detalles. En este prototipo se demostró que un **RAG** alimentado con búsquedas en Internet no era apropiado por la rapidez que una petición que un dispositivo móvil requiere. El proceso de vectorización se demora muchos minutos y ese proceso se ha de hacer para cada generación de tour, con lo que no resulta sostenible. A pesar de mejorar la información con Google Places, la descripción a veces podía estar alucinada.

Se evaluó la posibilidad de utilizar resúmenes de servicios como Wikivoyage o Wikipedia. Sin embargo, el éxito de las consultas dependía de que

el nombre del **PDI** coincidiera exactamente con el registrado en la base de datos, lo cual rara vez ocurría. Por ejemplo, el *Arco de San Esteban* en Burgos también es conocido como la *Puerta de San Esteban*, lo que generaba inconsistencias, si incluías la ciudad en la búsqueda se perdía la coincidencia y por lo tanto los *Puntos de Interés* nunca se actualizaban. Además, otras fuentes alternativas a Google Places no proporcionaban descripciones satisfactorias a través de sus API.

Ante estos desafíos, se decidió probar la versión Gemini 1.5 Pro, obteniendo resultados significativamente mejores. **Esta versión mantenía la originalidad y la precisión de la información, sin mostrar signos de alucinaciones** en las descripciones. Aunque la creación de la ruta demoraba unos segundos más, el resultado final era mucho más sólido. Las descripciones generadas incluso reflejaban un mayor entendimiento del contexto turístico, lo que implica que el rol del guía turístico *concienciado sobre la gentrificación y la masificación de los lugares de interés* se aplicaba ahora dejándose notar no solo en la elección de los lugares a visitar.

Además, gracias a la optimización en la ingeniería del prompt, se logró que las rutas propuestas se ajustaran mejor a límites razonables en los trayectos.

Mentoría y autoformación

El desarrollo de este proyecto no habría sido posible sin un proceso continuo de autoformación y la valiosa guía de David Miguel Lozano, un profesional experto en Flutter y LangChain, antiguo alumno de la Universidad de Burgos a quien contacté con el apoyo de mi tutor. Desde el inicio del proyecto, David desempeñó un papel fundamental como mentor, brindándome orientación técnica en aspectos clave del desarrollo, como la selección e integración de *Modelos de Lenguaje de Gran Escala* o herramientas para el control de calidad del código. Su experiencia y disposición para resolver dudas específicas, marcaron una diferencia significativa en mi comprensión y ejecución del proyecto.

Además, tuve la oportunidad de analizar su *Trabajo de Fin de Grado* [21], que me sirvió como referencia técnica. Este enfoque me permitió adaptar rápidamente soluciones efectivas al proyecto, asegurando una base sólida para el desarrollo.

En paralelo, recurrí a cursos en línea como parte de mi autoformación, lo que me permitió familiarizarme con conceptos avanzados de *Flutter y Dart*, consultando cuando me fue necesario conceptos complejos como la integración

de *Google Generative AI* y *Firebase* en aplicaciones Flutter. Aunque estos recursos fueron utilizados como material de consulta y aprendizaje, el trabajo realizado en este proyecto es completamente original y adaptado a las necesidades específicas planteadas.

6. Trabajos relacionados

A continuación, se comparan aplicaciones de referencia o similares usadas por usuarios para obtener lugares de interés turístico. Al final de la sección se presenta una tabla comparativa de estos trabajos relacionados con *Eco City Tours*.

6.1. Aplicaciones móviles planificadoras de rutas turísticas

Tripadvisor

Tripadvisor [29] permite a los usuarios planificar y organizar sus viajes, con recomendaciones basadas en reseñas y experiencias de otros viajeros. Se puede planificar un viaje personalizado indicando fechas pero prioriza los tours de pago, lo que hace difícil seleccionar rutas saludables gratuitas. Además, los establecimientos pueden promocionarse en la plataforma para mejorar su visibilidad lo que interfiere con la objetividad cuando se busca un lugar a visitar.

Wanderlog

Wanderlog [34] es una aplicación para la planificación de viajes que simplifica la creación de itinerarios, permitiendo agregar lugares de interés fácilmente. La aplicación tiene infinidad de funcionalidades. Tiene una versión Pro con asistente IA pagando 5 euros al mes. Gratuitamente se pueden enviar 5 mensajes, pero el LLM no almacena ningún contexto de los mensajes previos como se puede ver la imagen ?? donde muestra lugares

de Salamanca, pero en la respuesta siguiente cambia a la ciudad de Nueva York, sin haber sido ésta citada. Además, la respuesta ofrecida podría ser el resultado de un prompting sencillo como se puede ver en el prototipo del cuaderno Jupyter [prompting.ipynb](#).

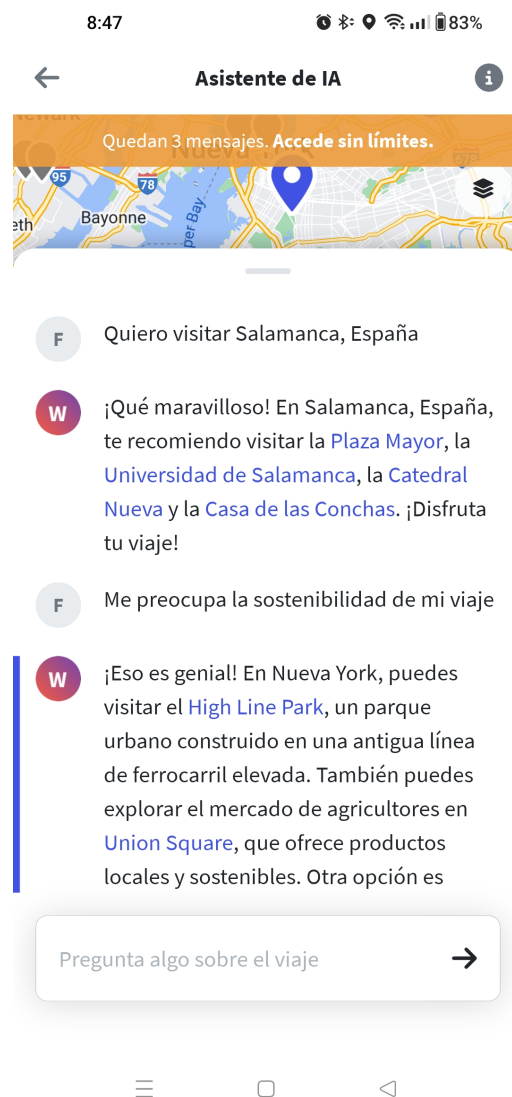


Figura 6.1: Chatbot de Wanderlog

Visit A City

Visit A City [3] ofrece itinerarios pre diseñados para destinos turísticos, permitiendo a los usuarios explorar lugares recomendados según su tiempo disponible. Aunque el uso de la aplicación es gratis, las rutas son todas comerciales y por tanto de pago, aunque existen lo que la aplicación llama *planes* que son itinerarios generados por los propios usuarios. Las rutas son generadas por aplicaciones de terceros.

Tiqets - Museos y Atracciones

Tiqets [28] es una aplicación que permite comprar entradas para museos y atracciones, ofreciendo guías digitales para planificar visitas. Sólo muestra información de museos, la gestión de mapas es fija, no se puede modificar el zoom y para rutas usa una aplicación por defecto del dispositivo en el que se ejecuta.

6.2. Otros *Trabajo de Fin de Grado (TFG)*

Chatbot

Trabajo Fin de Carrera de José María Redondo Guerra [11] que tuvo como tutores a José Ignacio Santos Martín y Carlos López Nozal.

Este trabajo fue desarrollado usando *Modelos de Lenguaje de Gran Escala (LLM)* para la generación de un chat con un sistema *RAG* para obtener la información de las normas de los *TFG* y así mejorar sus respuestas.

Visualización de las actividades socioculturales en Castilla y León CULTURALCyL

Trabajo Fin de Carrera de Yanela Lozano Pérez con tutores José Ignacio Santos, Virginia Ahedo y Silvia Díaz. Esta aplicación mostraba información de eventos culturales para lo cual usaba una aplicación móvil con servicios API. Es especialmente interesante como ejemplo de una aplicación móvil con una fuerte característica de usabilidad, ya que se centra en la visualización de mucha información que debe recibir el usuario de manera clara y sencilla.

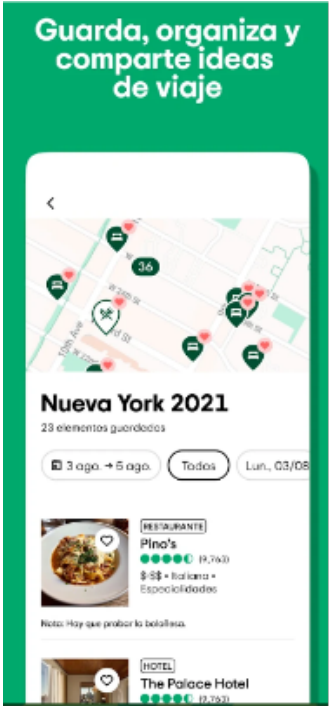

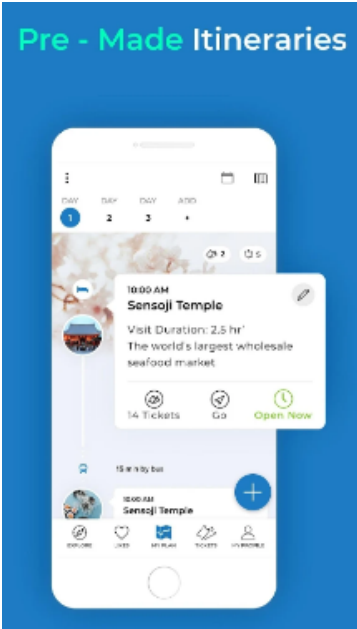
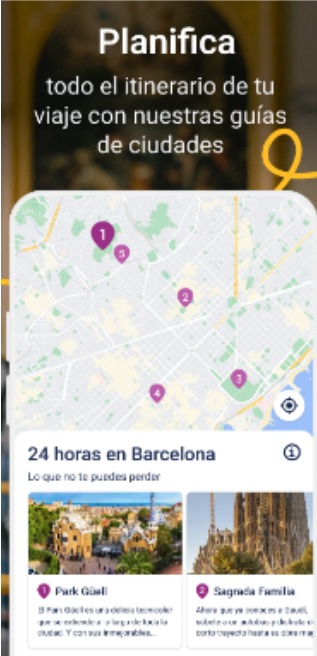
Tripadvisor	Wanderlog
	
Visit A City	Tiqets - Museos y Atracciones
	

Tabla 6.1: Aplicaciones móviles para la planificación de itinerarios turísticos

Característica	Tripadvisor	Wanderlog	Visit A City	Tiquets	<i>Eco City Tour</i>
Versión de pago	Sí	Sí	No	No	No
Recomendaciones tienen en cuenta factores de sostenibilidad	No	No	No	No	Sí
Modificación dinámica de rutas	Sí	No	No	No	Sí
Intereses de terceros pueden afectar a los resultados	Sí	Sí	No	No	No
Fuente de datos	Propia	Usuarios - LLM	Usuarios	Propia	LLM
Guardado de rutas	Sí	No	Sí	No	Sí
Seguimiento en vivo del usuario	No	No	No	No	Sí

Tabla 6.2: Comparación de aplicaciones

7. Conclusiones y Líneas de trabajo futuras

Conclusiones

A continuación, se detalla cómo se han cumplido todos los **Objetivos del Proyecto** planteados. Esto ha sido posible gracias a las capacidades adquiridas a lo largo de mi formación, que incluyen no solo habilidades técnicas, sino también capacidades resolutorias, las cuales se han visto fortalecidas durante la realización de este *Trabajo de Fin de Grado (TFG)*.

Cumplimiento de los objetivos

El proyecto ha logrado cumplir con todos los objetivos propuestos, tanto funcionales como no funcionales y personales:

Objetivos funcionales

- Generación de rutas turísticas personalizadas mediante modelos de lenguaje (**LLM**), encontrando como mejor solución el modelo *gemini pro 1.5* de acceso a través de un servicio API.
- Obtención de puntos de interés (**PDI**) priorizados según criterios sostenibles, como deslocalización del turismo y medios de transporte ecológicos gracias al desarrollo de prototipos que han permitido utilizar técnicas y mejorar el prompt de comunicación con el modelo.
- Visualización de rutas en mapas utilizando herramientas **GIS** de Google

- Optimización para ciclistas y peatones, priorizando carriles exclusivos y seguridad gracias a las directivas de *Google Directions*.
- Sistema de gestión de rutas que permite a los usuarios guardar y cargar tours gracias a *Firestore Cloud de Firebase*.

Objetivos no funcionales

Se cumplieron los siguientes aspectos:

- Integración de tecnologías avanzadas como **NLP** mediante **LLM**, garantizando un rendimiento estable y una experiencia de usuario satisfactoria.
- Diseño de una interfaz centrada en la usabilidad, logrando una interacción sencilla e intuitiva para el usuario final aplicando técnicas aprendidas en asignaturas de la carrera aplicadas al entorno *Flutter y Dart*.

Objetivos personales

Este proyecto ha representado un hito personal, permitiéndome:

- Formarme en la integración de **LLM** en aplicaciones software.
- Poner en práctica los conocimientos adquiridos en cursos de autoformación en **Dart** y **Flutter**.
- Desarrollar una aplicación profesional que será parte de mi portafolio para futuros empleadores.
- Culminar mi formación académica como Ingeniero Informático.

Reflexiones sobre el proceso de investigación

Uno de los mayores retos durante el desarrollo de este **TFG** fue la dificultad para encontrar referencias y recursos debido a la novedad de algunas tecnologías utilizadas. Por ejemplo:

- Herramientas para el uso de modelos **LLM**, como Langflow.
- Implementación de soporte para Dart en SonarCloud.
- Conexión de servicios de modelos **LLM** en dispositivos móviles.

Esta situación requirió mantenerse actualizado y adaptar el proyecto a un entorno de rápida evolución tecnológica. Sin embargo, estas dificultades se convirtieron en una oportunidad para desarrollar habilidades clave que seguro serán valiosas en mi futuro profesional y académico.

Valoración personal

Considero que este proyecto ha sido todo un éxito. Ha sido una experiencia enriquecedora desarrollar una aplicación desde sus cimientos. He aplicado y asimilado el enfoque de mi tutor tanto para el desarrollo técnico como para la investigación, documentación y resolución de problemas. Este enfoque me ha permitido completar el TFG y obtener una valiosa experiencia que trasciende el ámbito académico.

La aplicación desarrollada es funcional, robusta y sobre todo útil. Con pequeños ajustes para garantizar su sostenibilidad económica, podría publicarse sin problemas. Su diseño cubre un nicho en el mercado que podría ser aprovechado por instituciones locales o regionales para fomentar el turismo sostenible.

Los *Modelos de Lenguaje de Gran Escala (LLM)* han transformado el paradigma de la Informática desde su aparición, consolidándose como una valiosa fuente de conocimiento en múltiples áreas, incluida la Ciencia de Datos. El desarrollo de esta aplicación demuestra cómo estas herramientas, lejos de ser vistas como una amenaza, pueden integrarse como fuente de datos en aplicaciones de uso general. Con el entrenamiento adecuado, su potencial puede ampliarse aún más para desarrollar aplicaciones móviles o web diseñadas para casos de uso específicos, trascendiendo su aplicación tradicional en chatbots.

Por último, el desarrollo de *Eco City Tours* demuestra la viabilidad de integrar tecnologías modernas como LLM, Flutter y servicios de Google para crear soluciones que beneficien tanto a los usuarios como a la sociedad en general.

Líneas de trabajo futuras

Este TFG termina con la entrega de una versión funcional completa y estable para poder ser utilizada por usuarios que quieran planificar rutas turísticas sostenibles. Durante el desarrollo de este TFG han surgido muchas líneas de trabajo futuro que pueden servir como punto de partida en el caso

de ponerse en explotación. A continuación se enuncian algunas líneas que pueden ayudar a definir una futura evolución funcional.

- **Gamificación:** Incluir recompensas por rutas completadas o distancia recorrida con un medio ecológico. Logros que desbloqueen aspectos visuales del icono de la aplicación.
- **Ratings:** Publicar y valorar las tours generados permitiendo la búsqueda de los mismos a otros usuarios.
- **Planificador de rutas:** Añadir otras fuentes de datos que optimicen la sostenibilidad de los tours, por ejemplo, utilizando información por satélite para definir rutas basadas en áreas con mayor cobertura de sombra.
- **Multiplataforma:** La aplicación podría beneficiarse de su adaptación a otras plataformas, donde se tendría que tener en cuenta principalmente los permisos de localización. Al utilizar Flutter esta adaptación se podría realizar sobre el mismo código base, facilitando en gran medida su consecución. De hecho se trató de conseguir desde un principio en el desarrollo de este *Trabajo de Fin de Grado* su adaptación a iOS configurando los permisos de localización, sin embargo para probar estas funcionalidades es necesario contar con un equipo Mac. Legalmente, no es posible emular macOS en máquinas virtuales en otras plataformas, y tampoco se disponía de un equipo Mac propio ni de acceso a uno en préstamo.
- **Reconocimiento de lugares (Landmark Recognition):** Firebase ofrece una función avanzada de reconocimiento de lugares [8] que permitiría añadir una capa de personalización en la aplicación. Esta funcionalidad permitiría al usuario tomar una foto de uno de los *Puntos de Interés* visitados, y la aplicación podría identificar automáticamente el lugar y actualizar el estado de la ruta en tiempo real, indicando que se ha completado la visita. Esta mejora no solo facilitaría el seguimiento de la ruta, sino que también proporcionaría una experiencia de usuario más interactiva y dinámica.
- **Registro y autenticación de usuarios:** Para implementar algunas de las mejoras propuestas, es necesaria una gestión integral de usuarios. Actualmente, no existe un proceso de registro o autenticación, ya que es Cloud Firestore quien asigna un identificador único a cada usuario. Aunque esta gestión permite identificar los *Eco City Tours*

generados por cada usuario, presenta limitaciones, como la pérdida de acceso a los tours al reinstalar la aplicación, debido a la asignación de un nuevo identificador. Además, los tours previos quedarían en la base de datos sin posibilidad de ser recuperados. Implementar un sistema de registro y autenticación una vez configurado Firebase sería una tarea relativamente sencilla, que incluiría generar una pantalla de registro y acceso, así como modificar la configuración de Cloud Firestore para incluir módulos de autenticación mediante servicios como correo electrónico o cuentas de Google.

Siglas

BLoC *Business Logic Component.* 21, 22

CI *Integración Continua.* 17

CRUD *Create, Read, Update, Delete.* 7

GIS *Sistemas de Información Geográfica.* II, 3, 7, 43

GUI *Interfaz Gráfica de Usuario.* 20

IA *Inteligencia Artificial.* 15, 24

IDE *Entorno de Desarrollo Integrado.* 27, 28, 30

LLM *Modelos de Lenguaje de Gran Escala.* I, 1, 3, 5, 11, 13, 14, 15, 19, 20, 21, 24, 32, 35, 37, 40, 43, 44, 45

LLM *Large Language Models.* II

NLP *Procesamiento del Lenguaje Natural.* 4, 11, 44

ODM *Objetivos de Desarrollo del Milenio.* 9

ODS *Objetivos de Desarrollo Sostenible.* I, 1, 7, 8, 9

ODS11 *Ciudades y Comunidades Sostenibles.* I, 1

ODS4 *Objetivos de Desarrollo Sostenible: Educación de Calidad.* 4

ONU *Organización de las Naciones Unidas.* 8

OSM *Open Street Maps*. **III**, **31**

OSRM *Open Source Routing Machine*. **32**

PDI *Puntos de Interés*. **I**, **1**, **2**, **3**, **4**, **8**, **15**, **33**, **34**, **35**, **43**, **46**

POI *Points of Interest*. **II**

RAG *Retrieval Augmented Generation*. **16**, **34**, **40**

SDG *Sustainable Development Goals*. **II**

SDG11 *Sustainable Cities and Communities*. **II**

SICA *Sistema de Información sobre Contaminación Acústica*. **10**

SIG *Geographic Information Systems*. **I**, **7**

Smart City *Ciudades Inteligentes*. **I**, **II**

TFG *Trabajo de Fin de Grado*. **IV**, **5**, **11**, **16**, **17**, **24**, **27**, **30**, **31**, **32**, **35**, **40**, **41**, **43**, **44**, **45**, **46**

Glosario

Agentes Herramientas que alimentan a un LLM con información adicional obtenida de diversas fuentes como bases de datos, motores de búsqueda o páginas web. 15

Embeddings Técnica que convierte información en vectores numéricos de n dimensiones, utilizados por los modelos LLM para representar datos en un espacio vectorial. 13

Few-shot learning Técnica en la que el modelo recibe algunos ejemplos en el prompt para comprender con mayor precisión lo que se espera.. 11

Ingeniería del prompt Proceso de optimización y diseño de instrucciones (prompts) para mejorar las respuestas de los modelos de lenguaje a gran escala (LLM). 11

RAG *Generación Aumentada por Recuperación*, técnica en la que un modelo LLM se nutre de información adicional para mejorar sus respuestas. 13

Tokenización Proceso de dividir una cadena de texto en elementos más pequeños (tokens) para que sean procesados por un modelo de lenguaje. 13

Tool calling (Function calling) Técnica que garantiza que el modelo genere una salida en un formato estructurado y procesable, como JSON.. 11

Zero-shot learning Técnica en la que el modelo no recibe ejemplos previos sobre cómo realizar una tarea. El modelo interpreta la solicitud basado únicamente en el contexto.. [11](#)

Bibliografía

- [1] Datastax astra - cloud-native nosql database. <https://www.datastax.com/resources/datasheet/datastax-astra>. Último acceso: 22 de septiembre de 2024.
- [2] Getting started with wikimedia apis. https://api.wikimedia.org/wiki/Getting_started_with_Wikimedia_APIs. Último acceso: 22 de septiembre de 2024.
- [3] Visit a City. Visit a city - planificador de viajes. <https://www.visitacity.com>. Último acceso: 19 de septiembre de 2024.
- [4] Agile Alliance. Agile 101 - What is Agile? <https://www.agilealliance.org/agile101/>, 2024. [En línea; consultado el 19-septiembre-2024].
- [5] Centro de Estudios y Experimentación de Obras Públicas (CEDEX). Los Mapas de Ruido - SICAwEB. <https://sicaweb.cedex.es/los-mapas-de-ruido/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [6] Dio Package. Dio - Powerful HTTP client for Dart. <https://pub.dev/packages/dio>, 2024. [En línea; consultado el 20-septiembre-2024].
- [7] Felix Angelov et al. bloc - Flutter State Management Package. <https://pub.dev/packages/bloc>, 2024. [En línea; consultado el 26-noviembre-2024].
- [8] Firebase. Recognize Landmarks with ML Kit on Android and iOS. <https://firebase.google.com/docs/ml-kit/recognize-landmarks?hl=es>, 2024. [En línea; consultado el 2-noviembre-2024].

- [9] Flutter. Flutter - Beautiful native apps in record time. <https://flutter.dev>, 2024. [En línea; consultado el 18-septiembre-2024].
- [10] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Patrones de diseño: elementos de software orientado a objetos reutilizable*. Addison-Wesley, Madrid, 1st ed. edition, 2002.
- [11] GitHub - jrg1013. Chatbot - A Simple Chatbot Implementation. <https://github.com/jrg1013/chatbot>, 2024. [En línea; consultado el 19-septiembre-2024].
- [12] Google Developers. Directions API - Travel Modes. <https://developers.google.com/maps/documentation/directions/get-directions?hl=es-419#TravelModes>, 2024. [En línea; consultado el 26-noviembre-2024].
- [13] Google Maps. Google Maps Platform Documentation. <https://developers.google.com/maps>, 2024. [En línea; consultado el 18-septiembre-2024].
- [14] G.H. Ionescu, D. Firoiu, A.-M. Manda, R. Pîrvu, E. Jianu, and M.-E. Antoniu. Progress towards the 2030 Sustainable Development Goals for EU Urban Communities (SDG11). *Sustainability (Switzerland)*, 16(11), 2024.
- [15] Langflow. Langflow - Build LLM applications with ease. <https://www.langflow.org>, 2024. [En línea; consultado el 18-septiembre-2024].
- [16] LM Studio. LM Studio - Unlocking the Power of Large Models. <https://lmstudio.ai>, 2024. [En línea; consultado el 18-septiembre-2024].
- [17] Mapbox. Mapbox - Maps, Navigation, and Location Services. <https://www.mapbox.com>, 2024. [En línea; consultado el 20-septiembre-2024].
- [18] Mapbox. Mapbox Directions API Documentation. <https://docs.mapbox.com/api/navigation/directions/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [19] Mapbox. Mapbox Geocoding API Documentation. <https://docs.mapbox.com/api/search/geocoding/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [20] Mapbox. Mapbox Optimization API Documentation. <https://docs.mapbox.com/api/navigation/optimization-v1/>, 2024. [En línea; consultado el 18-septiembre-2024].

- [21] David Miguel. Go bees. <https://github.com/davidmigloz/go-bees>, 2024. Accessed: 2024-11-26.
- [22] O. Mitas, R. Badal, M. Verhoeven, K. Verstraten, L. de Graaf, H. Mitsova, W. Weijdemá, and J. Klijs. Tell Me Where to Go: An Experiment in Spreading Visitor Flows in The Netherlands. *International Journal of Environmental Research and Public Health*, 20(8), 2023.
- [23] M.J. Nieuwenhuijsen. Urban and transport planning pathways to carbon neutral, liveable and healthy cities; A review of the current evidence. *Environment International*, 140, 2020.
- [24] Ollama. Ollama - Explore and deploy large language models locally. <https://ollama.com>, 2024. [En línea; consultado el 18-septiembre-2024].
- [25] Prompting Guide. Prompting Guide - Todo sobre el Prompt Engineering. <https://www.promptingguide.ai/es>, 2024. [En línea; consultado el 20-septiembre-2024].
- [26] QGIS Development Team. Introducing GIS - A Gentle Introduction to GIS. https://docs.qgis.org/3.34/en/docs/gentle_gis_introduction/introducing_gis.html, 2023. [En línea; consultado el 18-septiembre-2024].
- [27] SonarSource. Sonarcloud - clean code as a service, 2024. Accessed: 2024-11-26.
- [28] Tiqets. Tiqets - entradas a museos y atracciones turísticas. <https://www.tiqets.com/es>. Último acceso: 19 de septiembre de 2024.
- [29] Tripadvisor. Tripadvisor - web de viajes. <https://www.tripadvisor.es>. Último acceso: 19 de septiembre de 2024.
- [30] United Nations. Objetivos de Desarrollo Sostenible, 2024. [En línea; consultado el 8-septiembre-2024].
- [31] Universidad de Burgos. Algoritmia (Curso 3) - Bibliografía Recomendada. <https://leganto.ubu.es/leganto/readinglist/lists?courseCode=8824>, 2024. [En línea; consultado el 18-septiembre-2024].
- [32] Unsplash. Unsplash: Fotos gratis. <https://unsplash.com/es>, 2024. [Internet; descargado 27-septiembre-2024].

- [33] Visual Studio Code. GitHub Copilot - AI Pair Programmer Overview. <https://code.visualstudio.com/docs/copilot/overview>, 2024. [En línea; consultado el 19-septiembre-2024].
- [34] Wanderlog. Wanderlog - planificador de viajes. <https://wanderlog.com/home>. Último acceso: 19 de septiembre de 2024.
- [35] Zube. Zube - project management for agile development. <https://zube.io>. Accessed: 2024-09-19.