



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 — 20 de septiembre
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, 20 de septiembre 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. Personalización de LLM	31

5.2. Elección de servicios Google sobre tecnología <i>Open Street Maps</i>	32
5.3. Elección de servicios Geocoding MapBox sobre servicios Google	32
5.4. Elección de google generative ai como primer LLM	33
5.5. Descarte de modelo en local	33
6. Trabajos relacionados	35
6.1. Aplicaciones móviles planificadoras de rutas turísticas	35
6.2. Otros <i>Trabajo de Fin de Grado</i> (TFG)	38
7. Conclusiones y Líneas de trabajo futuras	41
Bibliografía	43

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. RAG mostrado en la herramienta Langflow usando https://astra.datastax.com	15
4.1. Uso del snippet: mateapp	24
4.2. Ejemplo de Ghost Text de Copilot	25
4.3. Código real a utilizar en vez de propuesta Copilot	26
4.4. Uso de GitHub con la extensión de VSCode Git Graph	28
6.1. Chatbot de Wanderlog	36

Índice de tablas

6.1. Aplicaciones Similares	39
6.2. Comparación de aplicaciones similares	40

Índice de Códigos

4.1. Definición de variables de control en el Bloc	23
4.2. Carga en función del estado	23

1. Introducción

El crecimiento de población en las ciudades [17] 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 ODS número 11 titulado *Ciudades y Comunidades Sostenibles*. Según el informe de la UNESCO, [9], 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*. La aplicación se enfoca en las **preferen-**

cias 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 si realizar la ruta a pie o en bicicleta, qué ciudad visitar y cuántos *Puntos de Interés* incluir en la ruta. La “optimización” no se refiere al rendimiento técnico, sino a la búsqueda del camino más corto, calculado entre los distintos **PDI** a visitar, a través de 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 [16], 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, 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 elegir.
- **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 usará un servicio de navegación de calidad que debe ser capaz de calcular rutas seguras para peatones y priorizar carriles bicis sobre carreteras compartidas con vehículos motorizados.

- **Gestión de rutas:** La aplicación permitirá a los usuarios crear, guardar y compartir 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:

el usuario no
tendrá tal poder

- **Integración de inteligencia artificial usando *Procesamiento del Lenguaje Natural (NLP)*:** la aplicación podrá configurarse con distintos modelos de lenguaje sin afectar su mantenimiento. El usuario tendrá la opción de decidir si los modelos LLM se cargan localmente o son accedidos a través de servicios de terceros, permitiendo flexibilidad en la configuración. Con la tecnología de LangChain, los modelos se consultarán para implementar los objetivos funcionales de la ***propuesta de rutas turísticas personalizadas*** y ***obtención de los Puntos de Interés (PDI)***. Esta arquitectura permite una integración robusta y adaptable, evaluando los resultados generados por los modelos para asegurar la precisión y relevancia en la creación de rutas.
- **Uso de Herramientas Open-Source:** se priorizará para el desarrollo de la aplicación programas, paquetes, servicios o librerías que sean de código abierto, siempre que sea posible y no repercuta en la calidad del producto final. Se priorizará en todo caso una solución que no incurra en gasto alguno para el desarrollador o al usuario final por su uso. De esta manera se intenta fomentar el **ODS 4: Educación de calidad** que busca garantizar una educación inclusiva, equitativa y de calidad y promover 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. Debe ser también, altamente configurable para parametrizar todas las preferencias de usuario adaptándose a sus gustos.

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. [20] 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 uno pueda presentar la información efectivamente 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 [8] 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.[14]
- **Servicio de navegación y de optimización de rutas:** el servicio de navegación [13] 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 [15]. De esta manera se soluciona este problema popularmente conocido como *el problema del viajante*[24] 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* [23], “*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 [3], pueden ayudar a identificar zonas de masificación turística. Al combinar estos datos con otras fuentes de datos 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 las éstas en los mismos términos de lenguaje generando una comunicación entre el usuario y la máquina.

Uso de LLMs en Eco City Tours

En este proyecto, los *Modelos de Lenguaje de Gran Escala* se han utilizado para obtener los *Puntos de Interés (PDI)*, a través de un intercambio de diálogos entre la *Inteligencia Artificial* y el usuario, donde este último solicita recomendaciones de lugares para realizar turismo sostenible. En la sección de *prototipos* se implementa una serie de diálogos con un cuaderno de Python para evaluar las mejoras de las respuestas aplicando distintas técnicas de prompting[19]. El prototipo implementa una conversación básica con resultados mediocres o incluso alucinados, hasta respuestas generadas según las peticiones de estructura de datos por parte del usuario. La aplicación se beneficia de todo ello y genera una respuesta acorde al código que se quiere obtener en la aplicación móvil.

Técnicas de Ingeniería del prompt

Zero-shot y Few-shot learning

Zero-shot 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 un trabajo de aproximación mayor una técnica que siempre mejora la conversación con el modelo es la técnica **few-shot learning**: se facilita en el prompt al modelo unos ejemplos de lo que se quiere obtener. Para comprenderlo mejor veamos el siguiente ejemplo de prompt:

Clasifica los siguientes comentarios como Positivos,
Negativos o Neutros:

1. "El producto llegó a tiempo y en perfectas condiciones."

Clasificación: Positivo

2. "El artículo no cumplió con mis expectativas, estoy decepcionado."

Clasificación: Negativo

3. "La atención al cliente fue aceptable, pero podría mejorar."

Clasificación: Neutro

4. "El servicio fue excelente, muy recomendable."

Clasificación:

Al facilitar tres ejemplos de lo que se quiere obtener, la salida obtenida mejora y es lo que se espera por parte del usuario. Expresar en lenguaje natural lo que se quiere obtener es a veces más difícil y se puede malinterpretar por parte del modelo que darle unos ejemplos para que sepa con precisión el contexto. Más información al respecto se pueden observar en el prototipado del proyecto. Para terminar de ajustar la salida obtenida se usa la siguiente técnica:

Tool calling o function calling

Cuando la información del modelo tiene que ser muy precisa se recurre a esta técnica. En el caso del trabajo la información tenía que ser basada en una estructura que desde la programación se pudiera procesar fácilmente. Un archivo cuya estructura fuese en forma de json era vital. Para ello se le pide al modelo qué tipo de salida se requiere y para que no hubiese dudas se le facilitan un par de ejemplos. Una vez establecida la forma de la salida, se procede con el prompt de entrada usando la técnica que se quiera o cumpliendo con las especificaciones del modelo en concreto que se esté usando. De esta manera también se realiza una separación de abstracción que facilita la modularidad del código: se puede cambiar de origen en los datos, es decir, elegir otro modelo **LLM**, pero la salida del mismo siempre debe cumplir con estos requisitos desde el punto de vista de la programación. Es el mismo caso de abstracción usada en otros lenguajes de programación donde existe un repositorio y una fuente de datos. El programa se nutre de uno dejando el otro para acceder a datos de manera más concreta, donde el cambio de uno deja inalterado el funcionamiento del programa.

Retrieval-Augmented Generation (RAG)

Generación Aumentada por Recuperación es una técnica usada en modelos de inteligencia artificial en la cual se obtiene información para nutrir a un modelo 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/tokenización:** la información proporcionada se mide en tokens y cada modelo tiene una cantidad que puede usar como contexto, además del coste que algunos modelos pueden cobrar al usuario por token, es por ello que transformar una cadena de texto inicial que ocupa más espacio del estrictamente necesario en una cadena separada en pequeños trozos de información que además usa ciertos tokens especiales para mayor comprensión es una tarea previa a la recuperación de información.
2. **Embeddings:** consiste en transformar la información facilitada y presentarla en vectores de n dimensiones. Para ellos se usa comúnmente otro modelo entrenado para transformar la información en vectores.

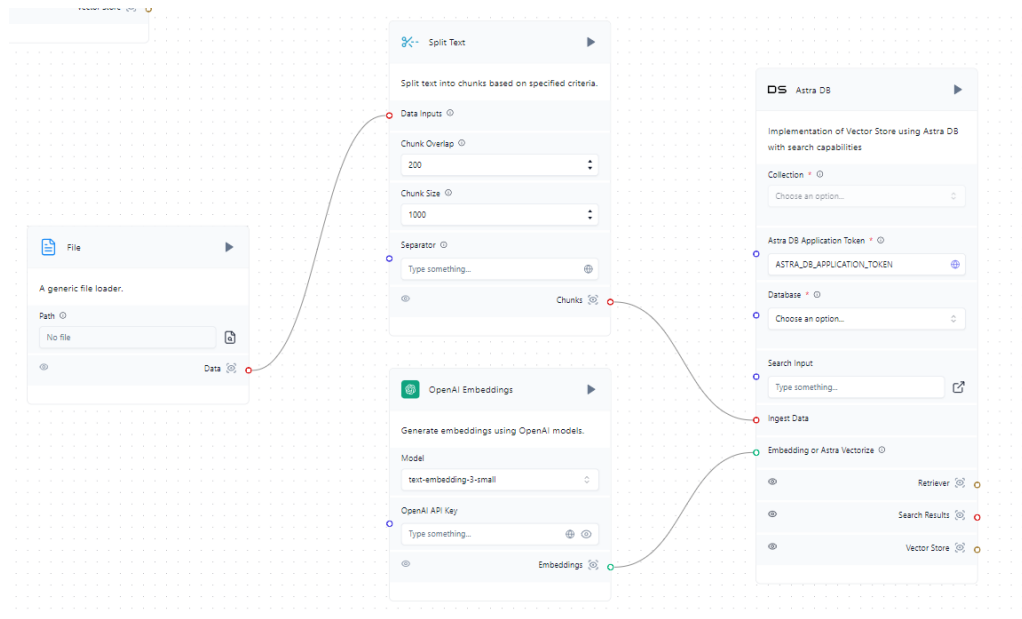


Figura 3.2: Preparación de la información de un RAG mostrada en la herramienta Langflow

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

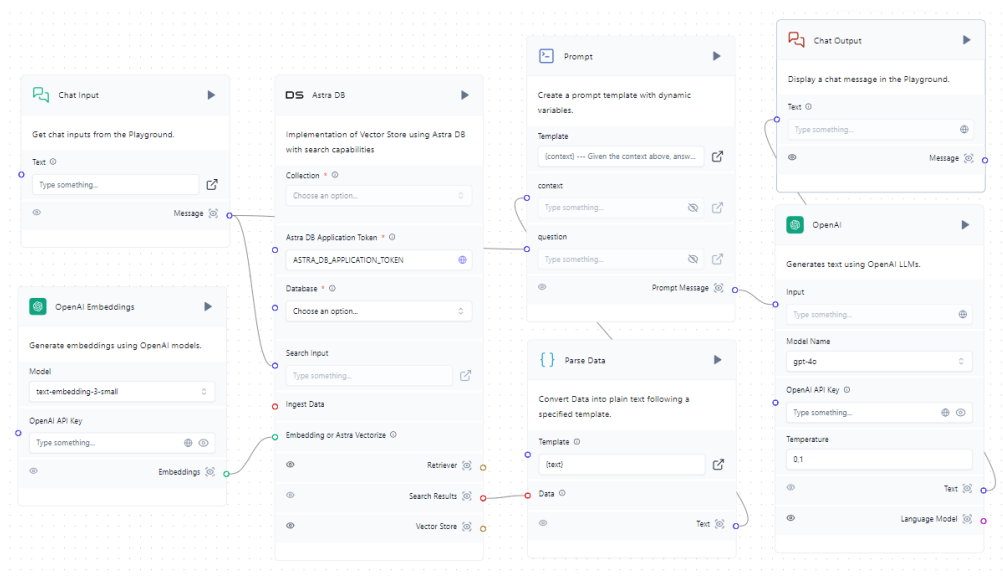


Figura 3.3: RAG mostrado en la herramienta Langflow usando <https://astra.datastax.com>

Uso de RAGs en la aplicación de este TFG

La utilidad de los *Retrieval Augmented Generation (RAG)* en aplicaciones es muy amplia. La más habitual se usa para conseguir un chatbot de empresas que sirvan como atención al cliente. En nuestro caso se alimenta a la base de datos con embeddings la información actualizada de la web usando agentes que serán explicados a continuación, esta información funciona como una entrada de datos de un sistema RAG para la mayor comprensión del mundo que le rodea al modelo. De esta manera y con un juego de prompts **se obtienen los mejores resultados posibles** que serán después tratados por la aplicación móvil para mostrar dicha información al usuario.

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. Estas múltiples herramientas pueden ser vistas como 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 Agentes en este TFG

Se utilizan varios con el fin de obtener a través de la web información actualizada de los puntos de interés de los lugares que se van a visitar.

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[2]. 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. 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 evalúa el trabajo realizado durante el sprint, se

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. Se ha seguido una división en función de su

4.1. Desarrollo relacionado con LLM

LM Studio

LM Studio [11] 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 especialmente 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 [18].

pendiente ver si
se usa

LangChain

En el último año LangChain se ha establecido como uno de los marcos de trabajo más populares del mercado. Esta herramienta multiusos aúna aplicaciones tan necesarias para el mundo de los *Modelos de Lenguaje de Gran Escala (LLM)* como pueden ser base de datos de vectores, memoria, prompts, herramientas, agentes como ya hemos visto en la sección 3.3 y cadenas de pensamiento (de ahí su nombre chain). En el prototipo de prompting de este TFG se puede ver el anidamiento de componentes como son estas cadenas para obtener la mejor entrada posible al modelo y obtener la mejor salida posible, estas cadenas pueden unir componentes como prompts, retrievers, processors, tools o incluso otras cadenas para procesos más complejos. Con todo ello LangChain supone una manera de combinar el poder de los LLM con la lógica de cualquier aplicación.

LangFlow

LangFlow es una herramienta de código abierto[10] 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[5] 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.

Bloc como gestor de actualización de estados

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**, 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 y mantenibilidad:** con las extensiones de Visual Studio Code, es posible generar nuevos blocs fácilmente, lo que facilita la creación de nuevas funcionalidades sin añadir complejidad innecesaria al código. Cada nueva funcionalidad de la aplicación puede estar controlada por un bloc independiente, lo que resulta en un código escalable y altamente 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. [6]. 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

```
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 4.1: Definición de variables de control en el Bloc

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 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 4.2: 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.

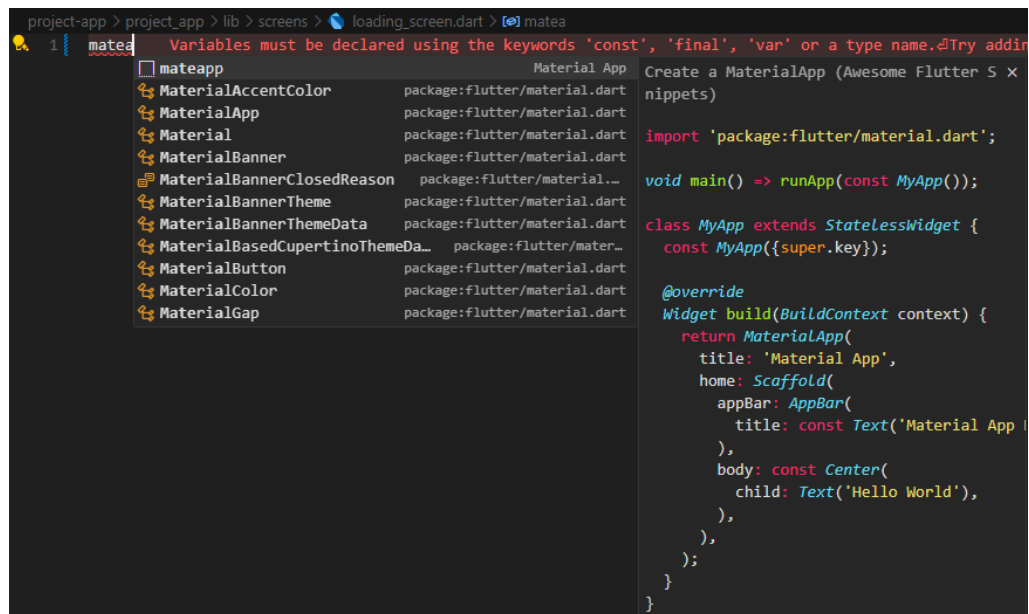


Figura 4.1: Uso del snippet: `mateapp`

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^[25] son dos extensiones de Visual Studio Code que se obtienen gratuitamente entre otras ventajas al identi-

ficarse 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), previendo 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.

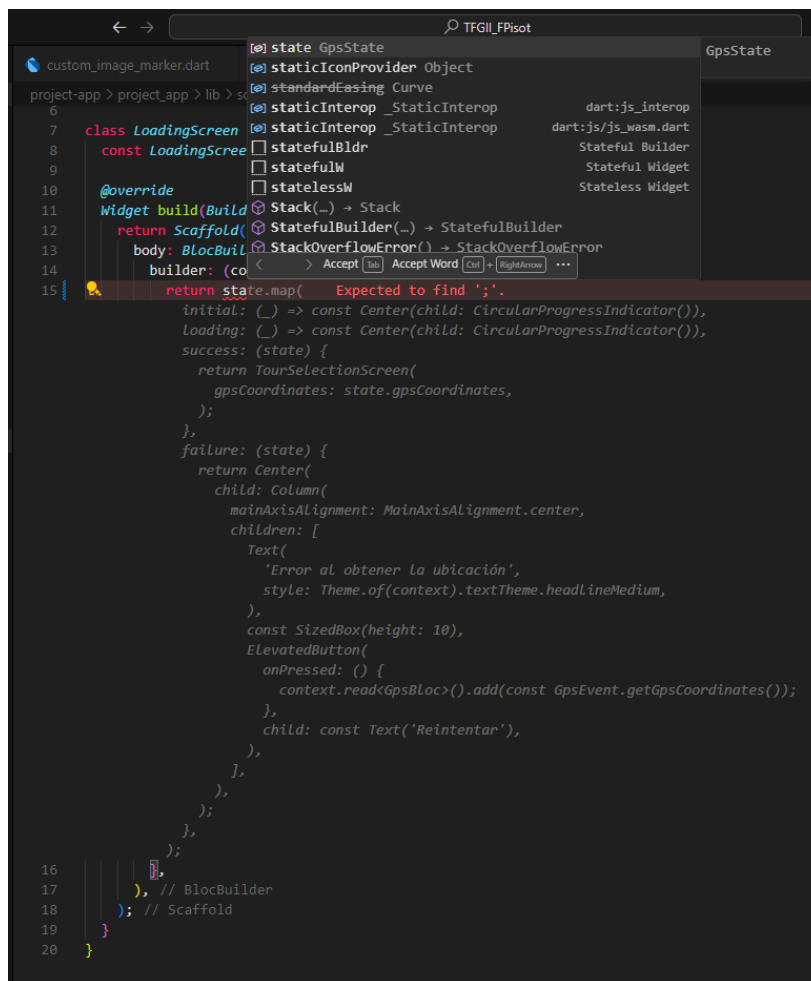


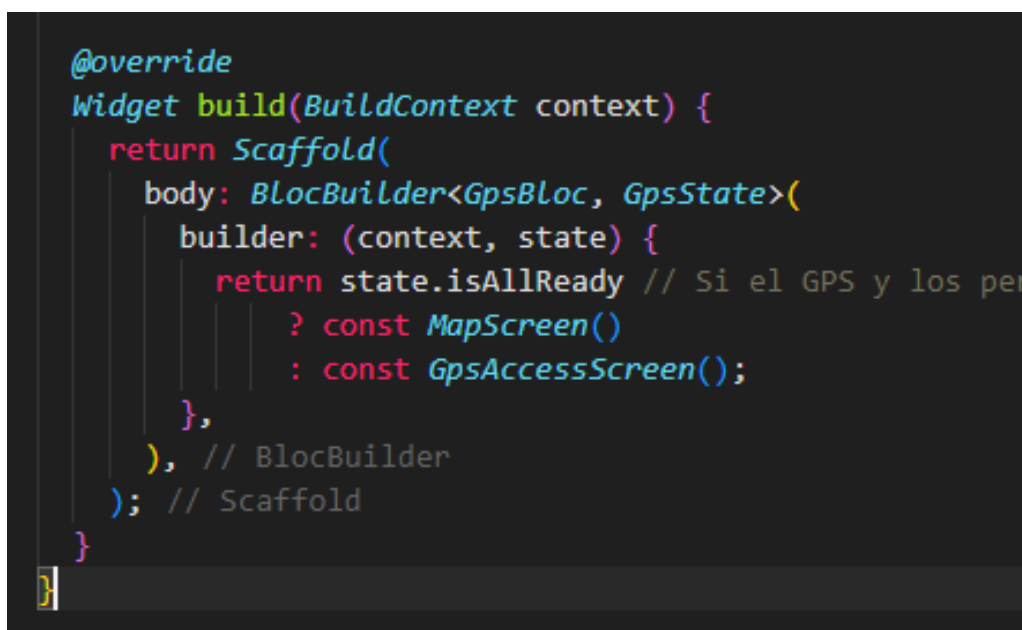
Figura 4.2: Ejemplo de Ghost Text de Copilot

La ventaja principal es el incremento de productividad: como se muestra en la imagen 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 con herramientas de terceros:** GitHub se integra fácilmente con una amplia variedad de herramientas de desarrollo, como servicios de CI/CD, plataformas de despliegue y gestores de proyectos como Zube, lo que permite automatizar tareas y mejorar el flujo de trabajo.
- **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. Además, GitHub ofrece 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 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.

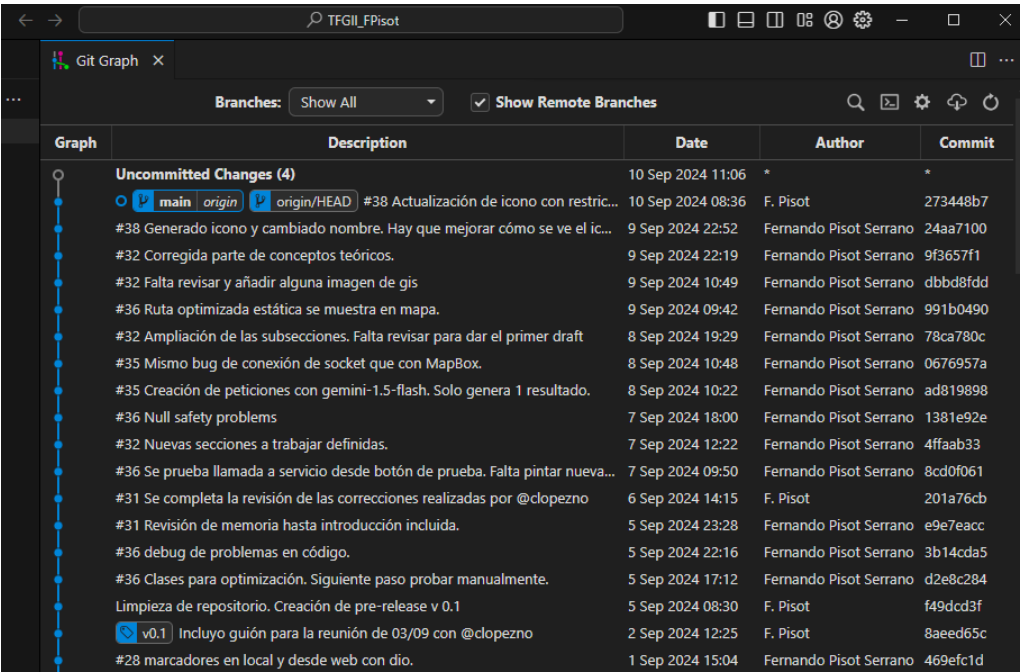


Figura 4.4: Uso de GitHub con la extensión de VSCode Git Graph

Zube

Zube [27] es una plataforma de gestión de proyectos con un enfoque colaborativo durante el seguimiento del desarrollo del mismo. Las características principales que lo hacen la herramienta usada 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. 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. En primer lugar 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. En este caso se buscaron alternativas que funcionaran de manera análoga a Tripadvisor para nutrir el **RAG**.

5.2. 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 alumno, 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 [4], un paquete marcado como favorito por Flutter, garantiza un soporte nativo y más estable en el ecosistema de Google.

5.3. Elección de servicios Geocoding MapBox sobre servicios Google

Los servicios de geocoding son herramientas que permiten convertir direcciones físicas (como “Calle Mayor, Ciudad, País”) en coordenadas geográficas (latitud y longitud), y viceversa. Esto es esencial para aplicaciones que requieren localización geográfica precisa, como mapas interactivos, planificación de rutas, análisis espacial, o cualquier función que dependa de ubicaciones

5.4. ELECCIÓN DE GOOGLE GENERATIVE AI COMO PRIMER LLM

específicas y los puntos de interés alrededor de una ubicación. Estos servicios en Google tienen el nombre de *Google Places* y su uso supone un coste económico para el desarrollador, incluso con el mínimo tráfico posible.

Para una versión inicial de la aplicación de este TFG se decidió usar MapBox[12], una empresa de mapas con un soporte similar a Google pero que no incurre gastos en volúmenes de tráfico como en los que se incurre durante la etapa de desarrollo. Con Eco City Tours ya publicada y si esta alcanza un uso elevado de peticiones siempre se puede cambiar a *Google Places* ya que la implementación como pasaba con el resto de servicios es nativa y por tanto muy sencilla de llevar a cabo, si se quiere conseguir unificación de servicios y costes. Desde el punto de vista de la programación solo cambia la manera de acceder a los datos pues las coordenadas se presentan primero con longitud y después con latitud, es decir, de manera inversa a los servicios Google. La modularidad del código realizado y el tratamiento de interceptores hace que un futuro cambio de servicios y peticiones GET sean fácilmente implementados.

5.4. Elección de google generative ai como primer LLM

Durante el desarrollo se estudiaron múltiples opciones a la hora de obtener la información desde un LLM. Siguiendo con el ecosistema de Google

5.5. Descarte de modelo en local

Aunque en un principio el ser capaz de probar la aplicación de una manera

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[22] 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[26] 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 6.1 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](#).

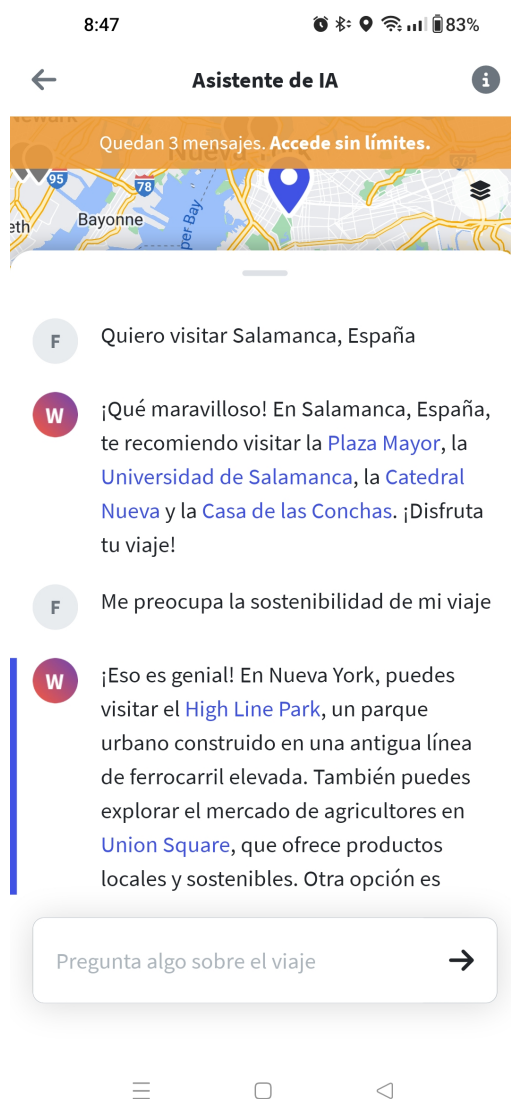


Figura 6.1: Chatbot de Wanderlog

Visit A City

Visit A City[1] ofrece itinerarios prediseñados 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[21] 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 [7] 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.

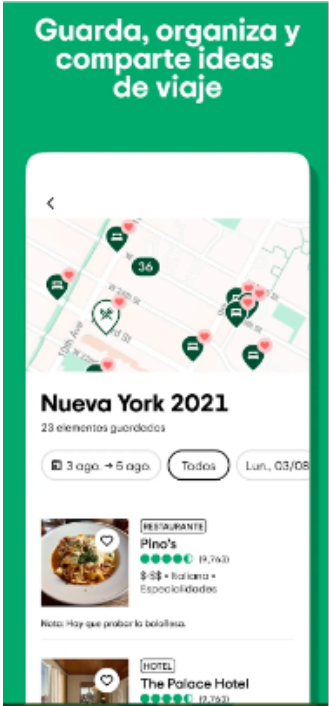

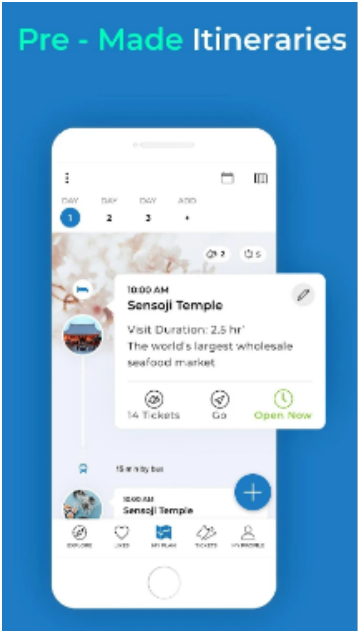
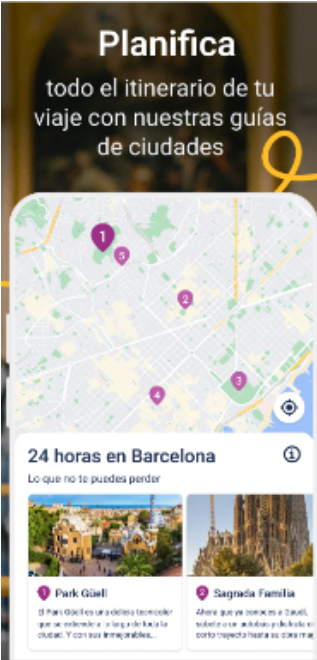
<p>Tripadvisor</p> 	<p>Wanderlog</p> 
<p>Visit A City</p> 	<p>Tiqets - Museos y Atracciones</p> 

Tabla 6.1: Aplicaciones Similares

Característica	Tripadvisor Wanderlog		Visit A City	Tiquets	Eco C Tour
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

Tabla 6.2: Comparación de aplicaciones similares

7. Conclusiones y Líneas de trabajo futuras

Conclusiones

Líneas de trabajo futuras

Existen múltiples maneras de expandir y llenar de nuevas funcionalidades a la aplicación llevada a cabo. Por citar algunas que puedan resultar más útiles al usuario:

- **Gamificación:** Recompensas por rutas completadas o distancia recorrida con un medio ecológico.
- **Ratings:** Valorar las rutas permitiendo la búsqueda de los mismos.
- **Mejora en planificador de rutas:** Determinación de la zona 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.

Bibliografía

- [1] Visit a City. Visit a city - planificador de viajes. <https://www.visitacity.com>. Último acceso: 19 de septiembre de 2024.
- [2] Agile Alliance. Agile 101 - What is Agile? <https://www.agilealliance.org/agile101/>, 2024. [En línea; consultado el 19-septiembre-2024].
- [3] Centro de Estudios y Experimentación de Obras Públicas (CEDEX). Los Mapas de Ruido - SICAwab. <https://sicaweb.cedex.es/los-mapas-de-ruido/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [4] Dio Package. Dio - Powerful HTTP client for Dart. <https://pub.dev/packages/dio>, 2024. [En línea; consultado el 20-septiembre-2024].
- [5] Flutter. Flutter - Beautiful native apps in record time. <https://flutter.dev>, 2024. [En línea; consultado el 18-septiembre-2024].
- [6] 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.
- [7] GitHub - jrg1013. Chatbot - A Simple Chatbot Implementation. <https://github.com/jrg1013/chatbot>, 2024. [En línea; consultado el 19-septiembre-2024].
- [8] Google Maps. Google Maps Platform Documentation. <https://developers.google.com/maps>, 2024. [En línea; consultado el 18-septiembre-2024].

- [9] 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.
- [10] Langflow. Langflow - Build LLM applications with ease. <https://www.langflow.org>, 2024. [En línea; consultado el 18-septiembre-2024].
- [11] LM Studio. LM Studio - Unlocking the Power of Large Models. <https://lmstudio.ai>, 2024. [En línea; consultado el 18-septiembre-2024].
- [12] Mapbox. Mapbox - Maps, Navigation, and Location Services. <https://www.mapbox.com>, 2024. [En línea; consultado el 20-septiembre-2024].
- [13] Mapbox. Mapbox Directions API Documentation. <https://docs.mapbox.com/api/navigation/directions/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [14] Mapbox. Mapbox Geocoding API Documentation. <https://docs.mapbox.com/api/search/geocoding/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [15] Mapbox. Mapbox Optimization API Documentation. <https://docs.mapbox.com/api/navigation/optimization-v1/>, 2024. [En línea; consultado el 18-septiembre-2024].
- [16] 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.
- [17] 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.
- [18] Ollama. Ollama - Explore and deploy large language models locally. <https://ollama.com>, 2024. [En línea; consultado el 18-septiembre-2024].
- [19] Prompting Guide. Prompting Guide - Todo sobre el Prompt Engineering. <https://www.promptingguide.ai/es>, 2024. [En línea; consultado el 20-septiembre-2024].

- [20] 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].
- [21] Tiqets. Tiqets - entradas a museos y atracciones turísticas. <https://www.tiqets.com/es>. Último acceso: 19 de septiembre de 2024.
- [22] Tripadvisor. Tripadvisor - web de viajes. <https://www.tripadvisor.es>. Último acceso: 19 de septiembre de 2024.
- [23] United Nations. Objetivos de Desarrollo Sostenible, 2024. [En línea; consultado el 8-septiembre-2024].
- [24] 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].
- [25] Visual Studio Code. GitHub Copilot - AI Pair Programmer Overview. <https://code.visualstudio.com/docs/copilot/overview>, 2024. [En línea; consultado el 19-septiembre-2024].
- [26] Wanderlog. Wanderlog - planificador de viajes. <https://wanderlog.com/home>. Último acceso: 19 de septiembre de 2024.
- [27] Zube. Zube - project management for agile development. <https://zube.io>. Accessed: 2024-09-19.