



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.

Documentación Técnica



Presentado por Fernando Pisot Serrano
en Universidad de Burgos — 11 de enero
de 2025

Tutor: Carlos López Nozal

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	18
Apéndice B Especificación de Requisitos	27
B.1. Introducción	27
B.2. Objetivos generales	27
B.3. Catálogo de requisitos	28
B.4. Especificación de requisitos	31
Apéndice C Especificación de diseño	53
C.1. Introducción	53
C.2. Diseño de datos	53
C.3. Diseño arquitectónico	55
C.4. Diseño procedimental	63
Apéndice D Documentación técnica de programación	69
D.1. Introducción	69
D.2. Estructura de directorios	69
D.3. Manual del programador	70

D.4. Compilación, instalación y ejecución del proyecto	74
D.5. Pruebas del sistema	76
Apéndice E Documentación de usuario	81
E.1. Introducción	81
E.2. Requisitos de usuarios	81
E.3. Instalación	82
E.4. Manual del Usuario	83
Apéndice F Anexo de sostenibilización curricular	91
F.1. F.1. Introducción	91
F.2. Competencias de Sostenibilidad Adquiridas	91
F.3. Aplicación de Competencias en el Proyecto	92
F.4. Conclusión	93
Bibliografía	99

Índice de figuras

A.1. Gráfico Burnup del Sprint 6	2
A.2. Gráfico de velocidad de los sprints 1-10	3
A.3. Tarea 12 mostrada en GitHub con la descripción, hito y etiquetas de la tarea a realizar.	4
A.4. Tablero Kanban de Zube con la gestión de tareas del Sprint 1	4
A.5. Figura burndown del Sprint Kick-off.	6
A.6. Figura burndown del Sprint 1.	8
A.7. Figura burndown del Sprint 2.	9
A.8. Figura burndown del Sprint 3.	10
A.9. Figura burndown del Sprint 4.	11
A.10. Figura burndown del Sprint 5.	12
A.11. Figura burndown del Sprint 6.	13
A.12. Figura burndown del Sprint 7.	14
A.13. Figura burndown del Sprint 8.	15
A.14. Figura burndown del Sprint 9.	16
A.15. Figura burndown del Sprint 10.	17
A.16. Figura burndown del Sprint 11.	18
A.17. Tráfico de peticiones durante el desarrollo de <i>Eco City Tours</i>	20
A.18. Interfaz de Facturación de Cloud Console.	21
B.1. Diagrama de caso de uso CU00 - Configuración GPS	36
B.2. Diagrama de casos de uso general de Eco City Tours	38
B.3. Diagrama de caso de uso CU02 - Navegar en el mapa	42
C.1. Diagrama de entidades. Eco City Tour y <i>Punto de Interés</i>	54
C.2. Diagrama relacional.	54
C.3. Diagrama arquitectónico de paquetes y dependencias	56

C.4. Diagrama arquitectónico detallado de paquetes y clases de <i>Eco City Tours</i>	57
C.5. Diagrama de interacción asíncrona del patrón BLoC en la aplicación	61
C.6. Diagrama de secuencia - Generación de Eco City Tour	64
C.7. Diagrama de secuencia - Añadir un PDI	65
C.8. Diagrama de secuencia - Eliminar un PDI	66
C.9. Diagrama de secuencia - Guardar un Eco City Tour	67
C.10. Diagrama de secuencia - Cargar un Eco City Tour	67
D.1. Resumen de análisis de la rama principal en SonarCloud	78
D.2. Creación de nuevo proyecto en SonarCloud	79
D.3. Evolución de casos abiertos en SonarCloud	80
E.1. Habilitar el permiso de uso de GPS	83
E.2. Pantalla de configuración de Eco City Tour	84
E.3. Pantalla de navegación de mapa	85
E.4. Búsqueda de PDI	86
E.5. Detalle de PDI	86
E.6. Pantalla de resumen del Eco City Tour	87
E.7. Pantalla de guardado del Eco City Tour	87
E.8. Pantalla de carga del Eco City Tour	88
E.9. Unirse al Eco City Tour	88
E.10. Seguimiento en vivo de la posición del usuario	89

Índice de tablas

A.1. Costes de <i>personal</i>	19
A.2. Costes de <i>hardware</i>	19
A.3. Costes totales de <i>Eco City Tours</i>	22
B.1. A01 - Usuario	31
B.2. A02 - Cloud Firestore	32
B.3. A03 - Gemini Pro 1.5	33
B.4. A04 - Google Places Service	34
B.5. A05 - Google Direction Service	35
B.6. A06 - Crashlytics	36
B.7. CU00 Configuración GPS	37
B.8. CU01 Configuración parámetros Eco City Tour	39
B.9. CU01.1 Calcular Eco City Tour	40
B.10.CU02 Navegar en el mapa	41
B.11.CU02.1 Visualizar detalles de los <i>Punto de Interés</i> (PDI)	43
B.12.CU02.2 Eliminar <i>Punto de Interés</i>	44
B.13.CU02.3 Añadir <i>Punto de Interés</i> (PDI)	45
B.14.CU02.4 Unirse a la ruta calculada	46
B.15.CU02.5 Añadir seguimiento del usuario	47
B.16.CU03 Ver resumen Eco City Tour	48
B.17.CU04 Guardar Eco City Tour	49
B.18.CU05 Cargar Eco City Tour	50
B.19.CU06 Registro de log	51

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El Plan de Proyecto Software es el documento clave que dirige el proceso de desarrollo de la aplicación móvil creada. Este apéndice tiene como objetivo detallar los aspectos críticos de la planificación y gestión del proyecto, asegurando una implementación eficiente y efectiva. La planificación temporal del proyecto se ha llevado a cabo con el *uso de la metodología ágil* buscando dividir el desarrollo en tareas, sprints e hitos que producen un resultado iterativo y bien estructurado, lo que conlleva a una mayor flexibilidad y a la adaptación más efectiva frente los cambios.

A continuación, se determinará la viabilidad, que reflejará los *recursos humanos y materiales*, así como los costes asociados necesarios para su valoración. La viabilidad incluirá una estimación de los fondos que se basarán en el salario de un trabajador simulado, así como un análisis de los posibles riesgos y su mitigación. Los aspectos económicos y técnicos de la viabilidad son fundamentales, ya que de ellos depende de que el proyecto esté en los límites establecidos y cumpla con los objetivos propuestos.

Este plan es esencial para la gestión del proyecto ya que sirve como una guía detallada, ayudando así a identificar y mitigar riesgos así como a la utilización eficaz de los recursos. Con el enfoque estructurado y ágil, proporcionado por este plan, el equipo de desarrollo podría entregar un producto de alta calidad que, además, cumplirá con el nivel de satisfacción alcanzado entre los usuarios o clientes.

A.2. Planificación temporal

Como se ha mencionado anteriormente, la planificación temporal del proyecto se ha llevado a cabo con el uso de metodología ágil: se basa en la división del desarrollo en tareas, sprints e hitos que producen un resultado iterativo y bien estructurado. Esto conlleva una mayor flexibilidad y a la adaptación más efectiva frente a los cambios.

Algunas herramientas utilizadas para la planificación temporal han sido GitHub y Zube. Ésta última ha permitido la organización de las tareas en tableros Kanban o el uso de **métricas ágiles** con gráficos que sirven para evaluar el desarrollo del proyecto. Algunos de los artefactos más relevantes usados son los siguientes:

- **Gráficos burnup / burndown:** muestran a lo largo del tiempo de un sprint la evolución de tareas realizadas por el equipo de desarrollo. En la explicación de los sprints se pueden ver los gráficos asociados al mismo

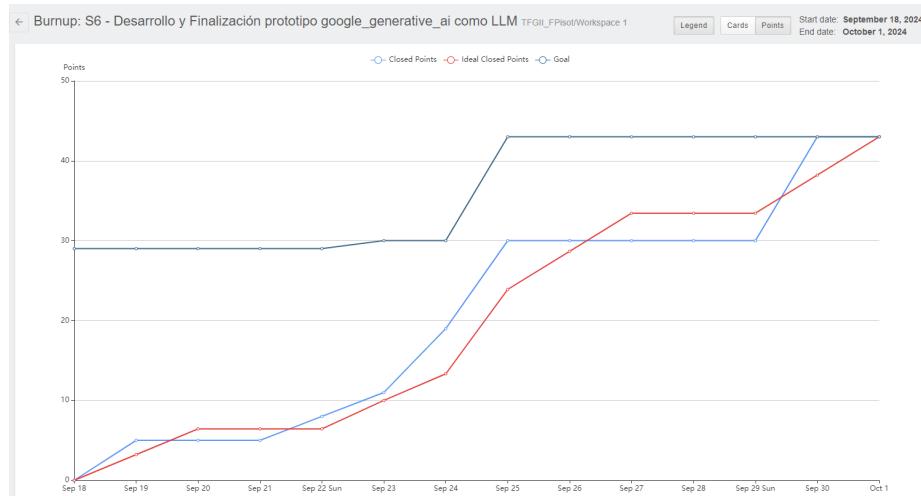


Figura A.1: Gráfico Burnup del Sprint 6

- **Gráfico de velocidad:** permite comprobar el trabajo realizado en los diferentes sprints de manera que resulte lo más constante posible.

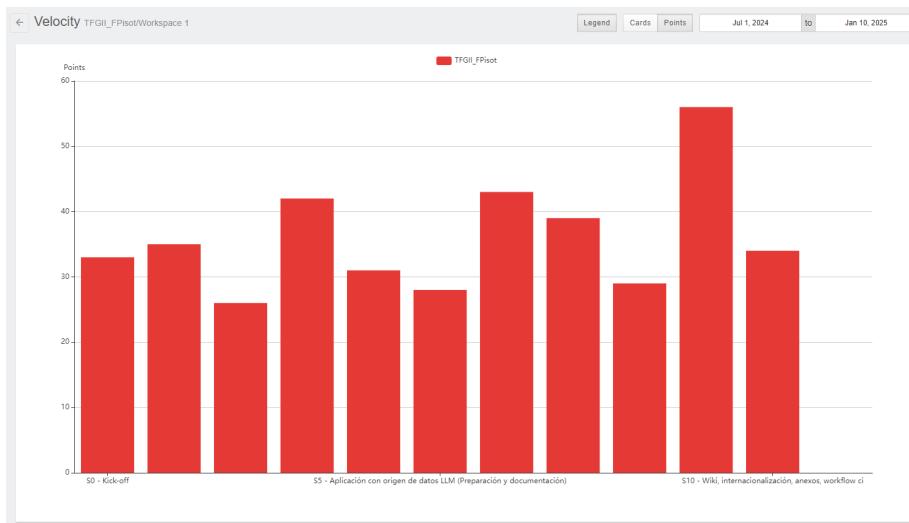


Figura A.2: Gráfico de velocidad de los sprints 1-10

A continuación veremos como la planificación temporal se ha llevado a cabo en diferentes sprints, cómo se ha ido iterando en las diferentes partes del proyecto y cómo se han ido cumpliendo los hitos propuestos. Para ello se mostrarán diferentes diagramas basadas en métricas ágiles.

Cada tarea se ha dividido en diferentes historias de usuario, que se han ido completando en cada sprint. Cada sprint ha tenido una duración media de dos semanas, y se han ido completando las tareas propuestas en cada uno de ellos. Un ejemplo se puede observar en la siguiente figura A.3

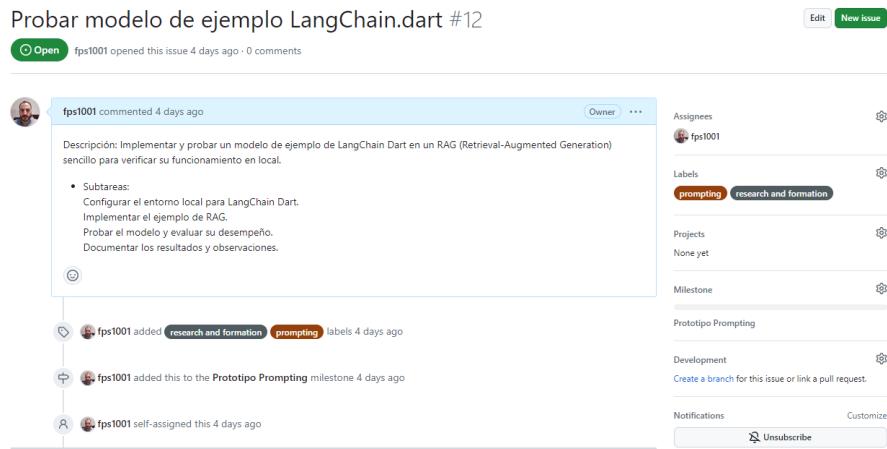


Figura A.3: Tarea 12 mostrada en GitHub con la descripción, hito y etiquetas de la tarea a realizar.

Gracias también al uso de *Zube*, se ha podido llevar un control de las tareas a realizar, las tareas completadas y las tareas pendientes. Además, se ha podido llevar un control de los hitos propuestos y de las historias de usuario completadas en cada sprint. Un ejemplo de ello se puede observar en la siguiente figura A.4

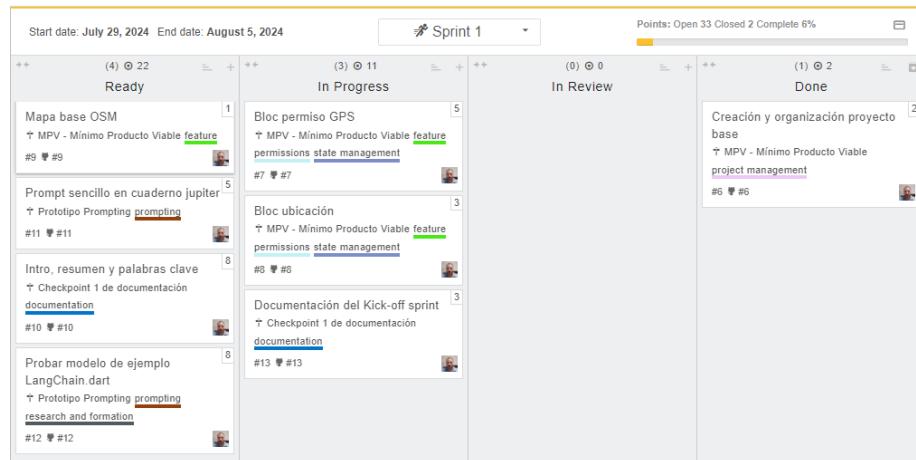


Figura A.4: Tablero Kanban de Zube con la gestión de tareas del Sprint 1.

Hitos

Los *hitos* o *milestones* son puntos de referencia que marcan el final de un conjunto de tareas. En este proyecto se han definido los siguientes hitos:

- **Kick-off** Completado el 30 de julio de 2024. Puesta en marcha del proyecto. A partir de las reuniones mantenidas con el tutor, se necesita tener todas las herramientas preparadas para empezar a desarrollar tanto la aplicación como su documentación.
- **MPV - Mínimo Producto Viable** Completado el 2 de septiembre de 2024. Se define el MVP como una aplicación móvil que sobre un mapa OSM muestre la ubicación de usuario, obtenga unos **PDI** básicos y una ruta que las una.
- **Checkpoint 1 de documentación** Completado el 2 de septiembre de 2024. Este milestone agrupa las tareas relacionadas con la creación y actualización de la documentación del proyecto hasta la reunión con el tutor el 1 de septiembre de 2024.
El objetivo es tener una documentación suficiente para que el tutor pueda dar feedback acerca de la misma y poder corregir errores.
- **Prototipo con tours generados por LLM** Completado el 1 de octubre de 2024. El objetivo es transitar desde una aplicación inicial capaz de mostrar lugares y rutas en un mapa, hacia una aplicación que sea capaz de conseguir que estos mismos marcadores y polilíneas sean generados a través de un LLM.
- **Prototipo Prompting** Completado el 15 de octubre de 2024. Este prototipo se puede realizar en un cuaderno Jupyter y su objetivo es mostrar la evolución en el prompt que dará como resultado unos **PDI** de mayor calidad.
- **Desarrollo de aplicación completo** Completado el 12 de noviembre de 2024. Se consideran todas las funcionalidades que debe tener la aplicación a presentar como completadas.
- **Finalización TFG**: Completado el 16 de enero de 2025. Se deja el proyecto en estado de entrega, completamente finalizado.

Organización en *Sprints*

Al comenzar este proyecto durante periodo no lectivo se realizaron los Sprint con variación de tiempo de una o dos semanas en función de la planificación personal. Una vez comenzado el curso y con la ayuda del tutor se realizaron reuniones que han servido para, siguiendo la metodología *Agile*, revisar el Sprint anterior, planificar el siguiente y hacer una pequeña retrospectiva para mejorar el trabajo conjunto.

Sprint 0 - Kick-off(22/07/2024 - 29/07/2024)

Después de las reuniones con el tutor, se establecen los objetivos del proyecto y se comienza a trabajar en la puesta en marcha del proyecto. Se establecen las herramientas a utilizar y se comienza a trabajar en la documentación del proyecto. 33 puntos de historia en 5 tareas:

1. Organización de repositorio.
2. Configuración de herramientas y equipos.
3. Trabajo en memoria I.
4. Investigación previa sobre LLM.
5. Determinación de ODS implicados.

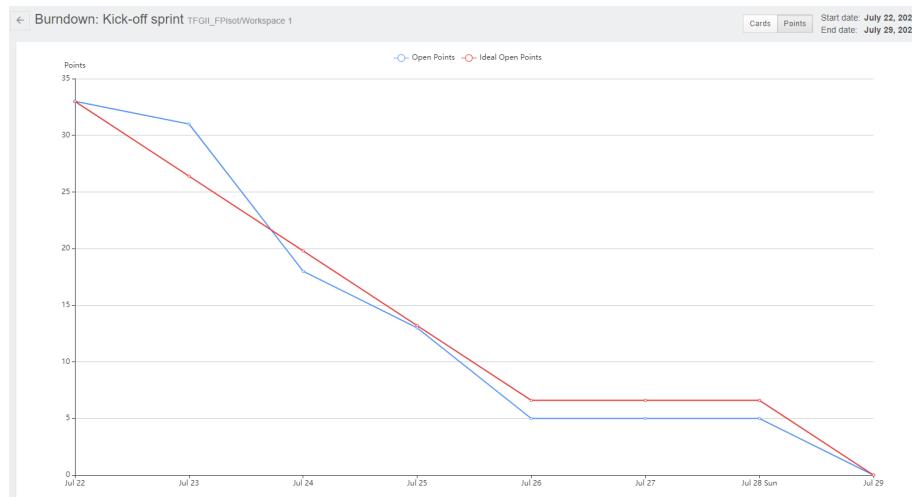


Figura A.5: Figura burndown del Sprint Kick-off.

Sprint 1 - Investigación LLM y desarrollo básico de aplicación con mapa (29/07/2024 - 05/08/2024)

Con las herramientas y una idea previa establecida, es el momento de empezar a desarrollar.

Objetivos: seguir formándome en LLM y las opciones que pueda implementar en el prototipo de prompt. Empezar a desarrollar la aplicación móvil con las características básicas. Aprender a documentar sprints, indicar qué elementos tendrá que documentar y aquellos que tenga claro ir documentando para hacer un avance significativo que pueda evaluar mi tutor. 35 puntos de historia en 8 tareas:

1. Creación y organización proyecto base.
2. Bloc permiso GPS.
3. Bloc ubicación.
4. Mapa base OSM.
5. Intro, resumen y palabras clave.
6. Prompt sencillo en cuaderno Jupyter.
7. Probar modelo de ejemplo LangChain.dart.
8. Documentación del Kick-off sprint.

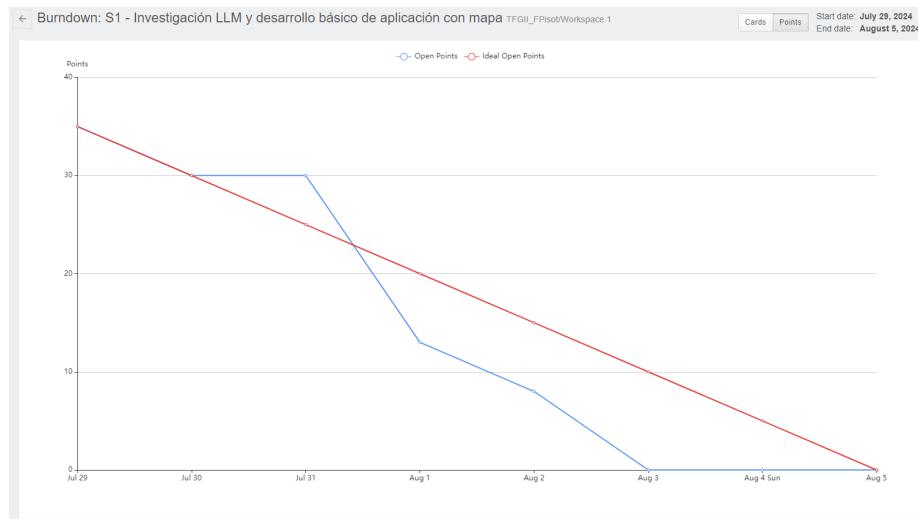


Figura A.6: Figura burndown del Sprint 1.

Sprint 2 - Implementación solución GIS (05/08/2024 - 12/08/2024):

A partir del concepto básico, se añaden pequeñas mejoras en los tres aspectos del proyecto.

Objetivos: mejorar el prototipo de prompting del cuaderno Jupyter hasta incorporar un sistema RAG, incluir marcadores al mapa en cuanto al desarrollo y continuar con la documentación. 26 puntos de historia en 6 tareas:

1. Redacción de los objetivos del proyecto.
2. Representación de marcadores en mapa.
3. Inicio de Trabajos relacionados.
4. Mostrar ubicación de usuario en el mapa.
5. Wikivoyage como agente para RAG.
6. Ingeniería del Prompting (cont.).

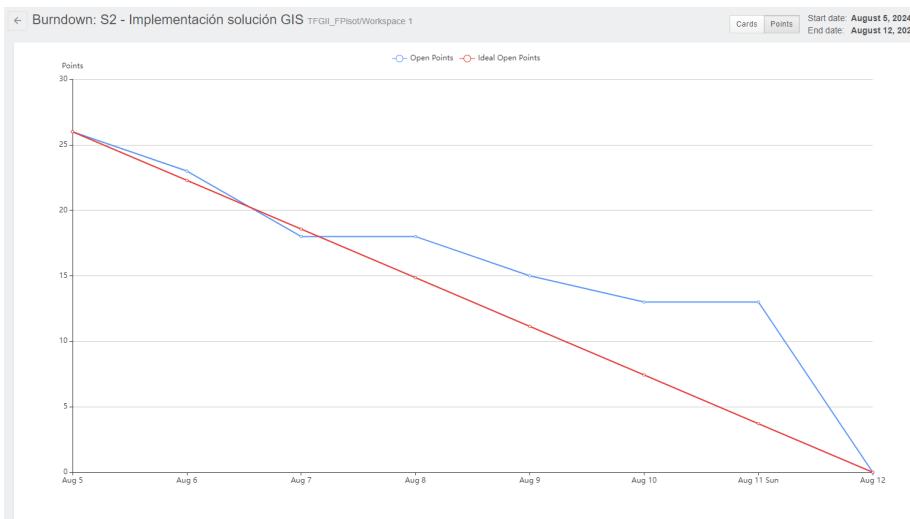


Figura A.7: Figura burndown del Sprint 2.

Dificultades encontradas: la documentación me hizo perder mucho tiempo debido a problemas con las librerías, después de mucho tiempo reinicié el proyecto desde la plantilla dada, insertando el texto, lo que solucionó el problema. En cuanto al diseño de la aplicación, el desarrollo fue lento al tener que evaluar diferentes opciones ya que la mayoría de fuentes utilizan mapas de Google, opción que se quería descartar.

Sprint 3 - MPV(12/08/2024 - 22/08/2024):

Este sprint fue más largo que los anteriores para mejorar el resultado final ya que la intención era dejar el proyecto en un estado de revisión lo más completo posible para afrontar la reunión prevista para inicio de septiembre con el tutor del mismo. Al intentar desarrollar la tecnología de enrutado del usuario se comprendió lo que ya se intuía en el sprint anterior y es que basar el trabajo en servicios de Google iba a reportar en un desarrollo más fácil y un resultado más robusto y fiable como se justifica en la sección 5 de la memoria de este [TFG](#). 42 puntos de historia en 5 tareas:

1. Obtención de marcadores POI.
2. Crear ruta entre dos puntos.
3. Trabajo inicial - Conceptos teóricos.
4. Agent RAG.

5. Adaptación a servicios Google.



Figura A.8: Figura burndown del Sprint 3.

Sprint 4 - Servicios MapBox y LLM, preparación reunión inicio curso.(22/08/2024 - 02/09/2024)

El objetivo es mostrar la versión más completa de la aplicación, la documentación y ahondar en el uso de nuevas herramientas como Figma, una herramienta de diseño de aplicaciones y LangFlow a la hora de utilizar otro modelo de prototipo. 31 puntos de historia en 5 tareas:

1. **Diseño de interfaz con Figma.**
2. **Uso de Mapbox como geocoding.**
3. **Marcadores POI y finalización MPV.**
4. **Documentación para Checkpoint 1.**
5. **LangFlow para Agent RAG.**



Figura A.9: Figura burndown del Sprint 4.

Sprint 5 - Aplicación con origen de datos LLM (Preparación y documentación) (04/09/2024 - 14/09/2024)

Después de la reunión con el tutor de inicio de septiembre y habiendo cumplido los objetivos de los primeros hitos se decide continuar intentando alcanzar el hito A.2. Para ello se prepara y documenta primero en este sprint el desarrollo necesario. 28 puntos de historia en 6 tareas:

1. Revisión documentación memoria - resumen.
2. Conceptos teóricos. Memoria.
3. Técnicas y Herramientas. Mem. C.4.
4. Modelo LLM en local.
5. Route Optimization dependientes del medio.
6. Generar icono y nombre de aplicación.

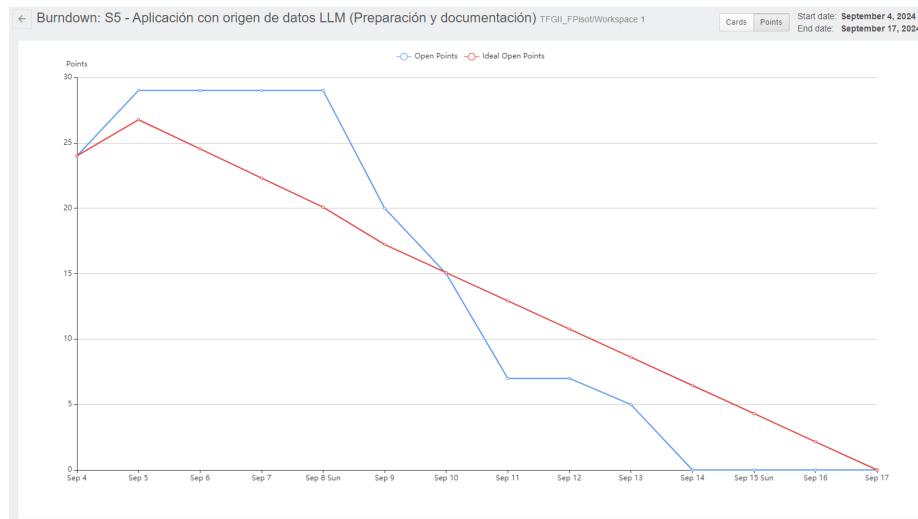


Figura A.10: Figura burndown del Sprint 5.

Sprint 6 - Desarrollo y Finalización prototipo google_generative_ai como LLM (18/09/2024-01/10/2024)

Habiendo encontrado una solución óptima al modelo LLM a utilizar se propone realizar un prototipo que implemente la interfaz de usuario y su conexión con el modelo LLM. En este sprint de gran avance en el desarrollo de la aplicación se consiguió reestructurar todo el código centralizando labores de gestión del tour generado y sus PDI manteniendo la modularidad del código. 43 puntos de historia en 6 tareas:

1. **Añadir tema a la aplicación.**
2. **Loading screen.**
3. **Modificar InfoWindow por Bottom Sheet.**
4. **Marcador básico de POI a marcador con imágenes.**
5. **Revisión documentación memoria v0.2.**
6. **Rehacer 3.3 Conceptos acerca de los modelos.**
7. **Estudiar guardado de rutas.**
8. **Reorganización de proyecto: Clase Eco City.**



Figura A.11: Figura burndown del Sprint 6.

S7 - Consolidación y Calidad (02/10/2024 - 22/10/2024)

Habiendo cumplido el hito A.2 y teniendo una aplicación con muchas funcionalidades implementadas, durante este sprint se busca consolidar el código y dotarlo de una calidad y mantenimiento con herramientas de soporte como Sonar Cloud o Logger. 39 puntos de historia en 9 tareas:

1. Establecer Requisitos.
2. Límite a la extensión del Eco City Tour.
3. Tour Screen: pantalla de resumen del tour.
4. Diagramas de paquetes/componentes.
5. Finalización prototipo Langflow.
6. Finalización prototipo Jupyter Notebook.
7. Instalación de Logger.
8. Integración de Sonar Cloud.
9. Convocatoria Prototipos Orientados al Mercado.



Figura A.12: Figura burndown del Sprint 7.

S8 - Cobertura de tests y guardado de rutas (23/10/2024 - 12/11/2024)

Se procede a implementar las últimas funcionalidades del desarrollo de la aplicación. Además, se busca que la integración con sonarcloud confirme que se trabaja con un standard de calidad para lo que se necesita la cobertura de tests y por último se retoma el trabajo de documentación. 29 puntos de historia en 7 tareas:

1. Configurar tarea de despliegue continuo Flutter.
2. Guardar Eco City Tour.
3. Corrección de diagramas.
4. Revisión de memoria/anexos.
5. Guardado de log compartido.
6. Investigación de opciones Google para base de datos.
7. Tests de aplicación / Cobertura.



Figura A.13: Figura burndown del Sprint 8.

S9 - Testing y selección de perfil de asistente IA (13/11/2024 - 03/12/2024)

En este sprint se exploró la posibilidad de generar diferentes perfiles de asistente, asignando al modelo roles específicos para personalizar las respuestas generadas. También se avanzó significativamente en la evaluación del código mediante la implementación de tests. Además, se inició la redacción de algunos apartados de los anexos relacionados con la documentación del proyecto. 56 puntos de historia distribuidos en 10 tareas:

1. **Test: servicios, modelos, repositorios. Pruebas.**
2. **Anexo E - Documentación de usuario.**
3. **Anexo D - Documentación T. de programación.**
4. **Anexo A3 - Estudio de viabilidad.**
5. **Refinamiento final de prompting.**
6. **Limpieza de repositorio.**
7. **Test: gestor de estados Bloc.**
8. **Añadir selección de asistente AI.**
9. **Apéndice C.**

10. Revisión final memoria.



Figura A.14: Figura burndown del Sprint 9.

S10 - Wiki, internacionalización, anexos, workflow ci (16/12/2024 - 20/12/2024)

Se estudia la posibilidad de generar diferentes asistentes dándole al modelo distintos roles. Se pretende realizar el test que llevará la aplicación en este sprint. Además, tendrá se realizará el inicio de apartados de anexos en cuanto a documentación. 56 puntos de historia en 10 tareas:

1. **Wiki.**
2. **Internacionalización.**
3. **Documentación de la programación (código).**
4. **Limpieza de repositorio.**
5. **Workflow Integración Continua.**
6. **Integración continua a manual de programador.**
7. **Mejoras en anexos.**
8. **Anexo de Diseño.**

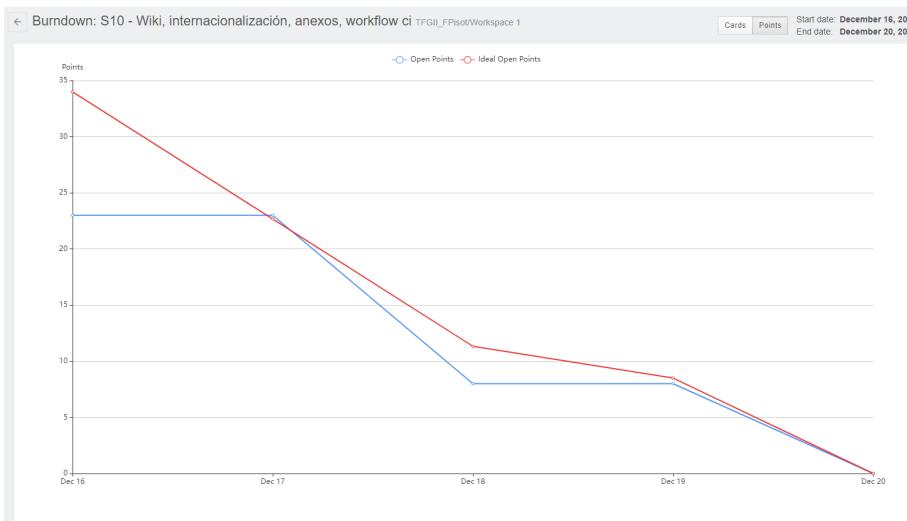


Figura A.15: Figura burndown del Sprint 10.

S11 - Finalización TFG (23/12/2024 - 09/01/2025)

Último sprint dedicado a finalizar las últimas funcionalidades, terminar la documentación y realizar los vídeos, en definitiva todos los elementos preparados para la entrega del *Trabajo de Fin de Grado (TFG)*. 39 puntos de historia en 7 tareas:

1. Revisión Anexo: planificación y análisis.
2. Internacionalización 2.
3. Mejora del README.md: Añadir insignias de herramientas utilizadas.
4. Video de presentación.
5. Video de demostración.
6. Revisión anexo de diseño y manual de usuario.
7. Adaptación de Memoria a la longitud permitida.



Figura A.16: Figura burndown del Sprint 11.

A.3. Estudio de viabilidad

En esta sección se analizan los aspectos clave para determinar si la implementación del proyecto y el desarrollo de la aplicación móvil *Eco City Tours* es viable desde un punto de vista legal y económico. Se destacan los valores positivos de la aplicación, como su independencia de intereses particulares y su enfoque sostenible, que la convierten en una propuesta valiosa en el mercado del fomento del turismo sostenible. Se evaluará si los beneficios esperados compensan los costes de desarrollo, el mantenimiento de los servicios y los posibles problemas legales, determinando así su viabilidad.

Viabilidad económica

Costes de personal

El desarrollo de la aplicación se realiza con un grupo de trabajo de un solo empleado de categoría profesional equivalente a la de un Ingeniero Informático. Según el XVIII Convenio Colectivo Estatal de Empresas de Consultoría, Tecnologías de la Información y Estudios de Mercado y de la Opinión Pública, publicado en el BOE el 26 de julio de 2023 [1] podría estar entorno a los 2.000 € antes de impuestos.

Concepto	Coste
Salario Desarrollador	1.500€
Retenciones y Seguridad Social	500
Total 4 meses	8.000€

Tabla A.1: Costes de *personal*

Coste de hardware

Para calcular el coste de los equipos se supone un período de amortización de 3 años para todo el hardware. Para realizar Eco City Tours se usará un ordenador de sobremesa con procesador i7 y 16 Gb de memoria RAM capaz de utilizar los programas necesarios para el desarrollo de manera fluida. Se tienen en cuenta también periféricos para dicho equipo y un teléfono móvil Android para pruebas de implementación en un dispositivo real no virtualizado. Por tanto y a modo resumen tenemos:

Concepto	Coste	Coste amortizado
Ordenador	1.200€	400€
Periféricos	240€	80€
Teléfono móvil Android	300	100
Total	1.740€	580€

Tabla A.2: Costes de *hardware*

Asociado a estos costes también podremos asociar el **uso de Internet** necesario para poder desarrollar sin problemas buscando información o descargando paquetes necesarios del repositorio de Flutter. Un servicio básico de este tipo **puede costar 30-60 euros** dependiendo de la velocidad del mismo o de otros productos asociados al cliente.

Coste de software

El software utilizado para el proyecto es libre. Visual Studio Code como IDE de desarrollo único, libre y gratuito y Android Studio en su versión gratuita para poder instalar emuladores de dispositivos móviles. El paquete SDK de Flutter no supone tampoco ningún gasto por parte del desarrollador.

Como sistema operativo se sugiere Ubuntu 22.04 LTS ya que tiene soporte a largo plazo, se trata de un entorno estable, confiable y se trata de una versión de código abierto. La herramienta y extensión de Visual Studio Code Copilot aunque útil no es imprescindible pudiendo usar . El resto de software para prototipado como Langflow o LMStudio son también gratuitos.

Coste de los Servicios Google

Durante el desarrollo de la aplicación la empresa incurrirá en gastos por uso de estos servicios. Una forma de pago tiene que estar asociada al proyecto y sobre el mismo se podrán asociar todas las API y servicios asociados al mismo: Firebase, Google Directions, Google Maps, Generative AI tienen gastos asociados. Sin embargo, el desarrollador tendrá un período de prueba de 3 meses donde no se cargará ningún gasto siempre que no supere los 300 euros.

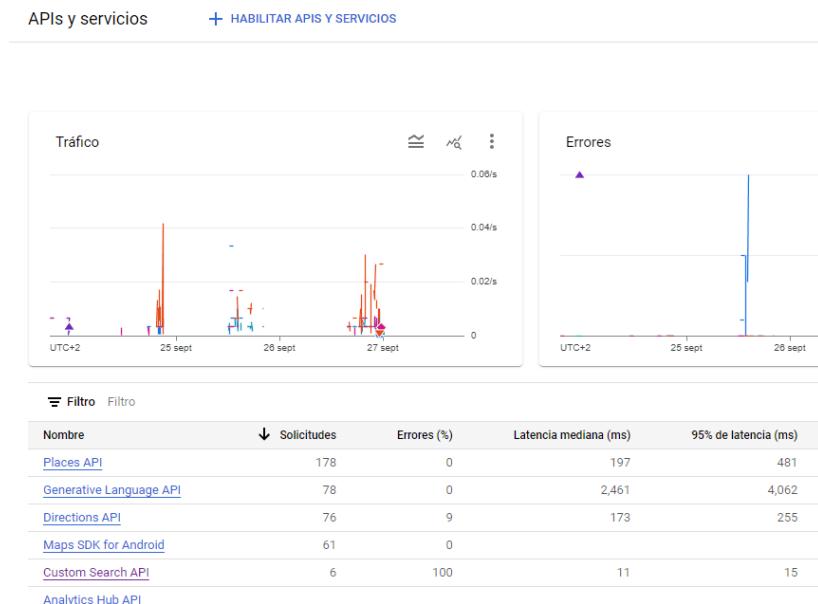


Figura A.17: Tráfico de peticiones durante el desarrollo de *Eco City Tours*

Desde la experiencia durante el desarrollo las peticiones están lejos de incurrir en gastos. En el período de pruebas donde se realizaron las mayores peticiones a los servicios como se puede ver en la imagen A.17 el coste llegó a los 40 euros, muy lejos de los 300 que supondrían un gasto para el desarrollador.



Figura A.18: Interfaz de Facturación de Cloud Console.

Pasado el período de tres meses, la cantidad que supone un umbral de coste se reduce a los 200 euros. En la imagen A.18 se puede ver como se inicia otro proceso de facturación diferente. Supondría por tanto algo a tener en cuenta si se despliega la aplicación en versiones tempranas en el repositorio de aplicaciones de Android puesto que las peticiones no serían solo de una persona, en este caso el desarrollador sino del público en general. La misma consideración habría que hacer si durante el desarrollo se cede el proceso de pruebas a varias personas que puedan hacer la función de beta-testers disparando el consumo de peticiones y por tanto el gasto. Para el caso de Eco City Tours se supone que el desarrollo y el mayor volumen de peticiones se concentran en los tres primeros meses donde el umbral de gasto es más alto, las pruebas las realiza el propio desarrollador que tendrá acceso al panel de Google Cloud para controlar los posibles gastos y la publicación de la aplicación será en un momento en el que el propio beneficio generado contrarreste los gastos en los que se pueda incurrir.

Conclusiones

El desarrollo de la aplicación se estima en 4 meses por el equipo auto gestionado por un solo desarrollador. No se tienen en cuenta alquileres de oficinas o co-working ya que se plantea la posibilidad de teletrabajar, pudiendo trabajar de manera deslocalizada ahorrando así costes. Teniendo en cuenta lo anteriormente citado obtenemos la siguiente tabla resumen:

Concepto	Coste
Mano de obra	8.000€
Hardware	580€
Internet	135€
Software	0€
Servicios Google	0€
Revisión y auditoría del tutor	750€
Total	9.465 €

Tabla A.3: Costes totales de *Eco City Tours*

Se ha incluido un coste estimado de 750 € por las tareas de **revisión y auditoría realizadas por el tutor**. Este coste considera una dedicación de 30 horas, remuneradas a un coste promedio de 25 €/hora, que incluye reuniones, revisiones parciales y finales del proyecto.

Una vez analizado los costes devengados de la creación de la aplicación el beneficio de su explotación debe ser mayor al gasto ocasionado por su creación. Además, una vez publicada la aplicación en la *Play Store* el beneficio obtenido por su explotación debe compensar el tiempo requerido por el personal en horas de mantenimiento de los posibles problemas que pueda experimentar como por ejemplo problemas por actualización de los componentes claves como modelos LLM, que puedan quedar obsoletos. El beneficio de la aplicación puede provenir de varias fuentes en función del modelo de explotación que se eligiese:

- **Ingresos por descarga:** si la aplicación tiene un precio por descarga, los ingresos corresponderían al dinero obtenido menos una comisión del 15 % cobrada por Google por cada venta [6]. Inicialmente, se propone un precio aproximado de lanzamiento de 5 euros, considerando que se obtendrían aproximadamente 4,40 euros como beneficio por descarga. Este precio se estima razonable para incentivar descargas durante la fase inicial. Tras un periodo de prueba, el precio podría ajustarse en función del número de descargas y del comportamiento de los usuarios. Los gastos derivados del uso de servicios externos, como peticiones a APIs, se compensarían con un equilibrio entre usuarios que realicen menos consultas y aquellos más demandantes, garantizando siempre un beneficio estimado en el modelo económico.

- Publicidad: empresas o entidades colaboradoras podrían financiar el desarrollo con publicidad siempre que ésta no suponga una perdida de independencia en los resultados de los modelos, fomente el turismo ecológico y no sea invasiva ni perjudique la experiencia de usuario.

Viabilidad legal

Al utilizar los servicios ofrecidos por Google, la compañía se convierte en un socio en el que se delegan muchas de las responsabilidades legales, técnicas y de cumplimiento normativo. Esto simplifica la implementación de Eco City Tours, a la vez que se adquieren ciertos compromisos con el gigante tecnológico.

Licencias de uso de las API de Google

Cada API tiene sus propios términos de uso [5] y se tiene que tener en cuenta los siguientes aspectos:

- **Uso permitido:** Las API deben ser utilizadas únicamente para finalidades relacionadas con el proyecto.
- **Prohibiciones:** Se prohíbe el *scraping*, almacenamiento de datos más allá de los límites permitidos o redistribución no autorizada.

Restricciones específicas:

- *Generative AI:* Regulada por políticas relacionadas con el uso de datos generados, especialmente en contextos comerciales.
- *Places API y Directions API:* Restringen el almacenamiento y redistribución de los datos obtenidos.
- *Maps SDK for Android:* Requiere incluir el logotipo de Google y atribuciones visibles en la interfaz.

Los servicios de Google tienen modelos de precios basados en uso para lo que es legalmente **obligatorio registrar un método de pago válido** en tu cuenta de Google Cloud para garantizar la continuidad del servicio.

Privacidad de datos

El cumplimiento de normativas sobre protección de datos es fundamental:

- **GDPR:** Si se cumplen algunos de los trabajos futuros de la aplicación, ésta recogería datos del usuario, si es un ciudadano europeo debe:
 - Firmar un *Acuerdo de Procesamiento de Datos* (DPA) con Google.
 - La aplicación debe mostrar cómo se manejan los datos obtenidos.

Se debe informar a los usuarios que los datos están sujetos a las Políticas de privacidad de Google.

Restricciones geográficas

Algunas API tienen restricciones en determinados países debido a leyes locales o sanciones internacionales.

Propiedad intelectual

Los datos obtenidos a través de las API de Google (como mapas y lugares) son propiedad intelectual de Google.

■ Prohibiciones:

- No se puede almacenar ni redistribuir datos más allá de los límites permitidos.
- No se puede usar los datos para minería o análisis masivo sin autorización.

Cumplimiento con normas de publicación

Para poder publicar la aplicación en el repositorio de aplicaciones Android hay que especificar claramente la política de privacidad y detallar el uso de los servicios.

Lista a cumplir para viabilidad legal

1. Revisar términos de uso de cada API.
2. Registrar método de pago.

3. Cumplir con las normativas de privacidad (GDPR, CCPA).
4. Incluir atribuciones visibles en la interfaz de usuario.
5. Respetar las restricciones de almacenamiento y uso de datos.
6. Desarrollar y publicar una política de privacidad clara para los usuarios.

Respetando el uso ético de la aplicación y cumpliendo con estos requisitos la viabilidad legal de Eco City Tours estaría cumplida y no supondría un impedimento para su ejecución.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se presentan los requisitos de la aplicación, abordando tanto los objetivos generales como los específicos del proyecto. Se incluye un catálogo detallado de los requisitos funcionales y no funcionales, que definen el comportamiento y las características técnicas de la aplicación. Además, se proporciona una especificación detallada de los requisitos a través de tablas de casos de uso, complementadas con su respectivo diagrama de casos de uso, lo que facilita una comprensión clara de las interacciones principales de los usuarios con el sistema.

B.2. Objetivos generales

La misión fundamental de este proyecto persigue conseguir los siguientes propósitos:

- **Fomentar el turismo sostenible:** Facilitar a los usuarios la explotación de ciudades y zonas rurales promoviendo al mostrar rutas no motorizadas y modos de transporte como caminar y el uso de bicicletas.
- **Optimización de experiencias turísticas personalizadas:** Ofrecer a los usuarios rutas personalizadas que se ajusten a sus intereses y preferencias, proporcionando información detallada y relevante sobre los puntos de interés seleccionados.
- **Promover el uso de tecnologías inteligentes en el turismo:** Utilizar tecnologías avanzadas como servicios GIS, Google Places y

LLM para mejorar la experiencia del usuario, facilitando la generación automática de rutas y la obtención de información actualizada sobre los destinos turísticos.

- **Mejorar la accesibilidad a la información turística:** Proporcionar una plataforma fácil de usar que permita a los usuarios acceder rápidamente a descripciones, fotos y otros datos sobre los puntos de interés, mejorando su experiencia de exploración.
- **Rutas generadas sin intereses comerciales:** Generar rutas turísticas sin influencias comerciales, ofreciendo una experiencia imparcial y auténtica, en contraste con otras aplicaciones de recomendaciones de viajes.

B.3. Catálogo de requisitos

Requisitos funcionales

- **RF-1 Solicitar permisos de uso de GPS:** La aplicación solicitará el permiso para acceder al GPS cuando se inicie por primera vez, ya que es necesario para calcular y mostrar la ubicación del usuario en tiempo real.
- **RF-2 Solicitud de activación de GPS:** Si el GPS está desactivado, la aplicación redirigirá a una pantalla que indicará al usuario la necesidad de activarlo para el correcto funcionamiento de la aplicación.
- **RF-3 Activación/Desactivación de seguimiento de usuario:** La aplicación mostrará en tiempo real el recorrido del usuario en el mapa, y este seguimiento podrá activarse o desactivarse en cualquier momento mediante un botón.
- **RF-4 Centrar la situación actual del usuario sobre el mapa:** El usuario podrá centrar manualmente su posición en el mapa mediante un botón dedicado. Además, existe la opción de fijar la ubicación del usuario en el centro del mapa durante su recorrido.
- **RF-5 Configuración de las preferencias del tour:** El usuario llenará un formulario indicando el lugar que desea visitar, la cantidad de puntos de interés que quiere ver, sus preferencias de transporte (a pie o bicicleta), sus intereses, y el tiempo máximo que quiere dedicar a la ruta.

- **RF-6 Cálculo de información a través de un LLM:** La aplicación usará un servicio Gemini para generar los puntos de interés de acuerdo con las propiedades de configuración de la ruta. Además, un servicio Google Places mejorará los datos proporcionando descripciones, fotos, URL, ratings y número de votos de los PDI.
- **RF-7 Eliminación de PDI:** El usuario podrá eliminar puntos de interés tanto desde la pantalla del mapa como desde el resumen de la ruta. Cada vez que un PDI es eliminado o añadido, la ruta se recalcula automáticamente para ofrecer el trayecto más óptimo.
- **RF-8 Cálculo de ruta optimizada:** La aplicación calculará la ruta más corta que conecte los puntos de interés seleccionados por el usuario, adaptándose al medio de transporte elegido (a pie o bicicleta).
- **RF-9 Capacidad de añadir un PDI:** El usuario podrá agregar manualmente un lugar introduciendo su nombre en la barra de búsqueda. Si el lugar existe en los servicios de Google, será añadido automáticamente a la ruta; de lo contrario, no se tomará ninguna acción.
- **RF-10 Unirse a Eco City Tour:** El usuario podrá unirse a la ruta existente en cualquier momento. La aplicación calculará la ruta más corta para conectarlo con el tour.
- **RF-11 Mejora de los puntos de interés con servicio de obtención de información:** Los datos de los puntos de interés se enriquecerán con información adicional obtenida de Google Places, incluyendo ratings, imágenes, URL y número de votos, mejorando la experiencia del usuario.
- **RF-12 Guardado y carga de las rutas turísticas:** el usuario podrá guardar los tours que quiera y tendrá acceso a los mismos desde la pantalla de configuración del Eco City Tour.

Requisitos no funcionales

- **RNF-1 Rendimiento:** la aplicación debe demostrar un tiempo de respuesta aceptable para que su manejo sea fluido y la carga de datos sea razonable al enlazar varios servicios asíncronos, de tal manera que no se perjudique la experiencia de usuario.

- **RNF-2 Usabilidad:** Eco City Tours debe ser intuitiva y fácil de entender y utilizar.
- **RNF-3 Disponibilidad:** la aplicación debe estar disponible independientemente de la localización del usuario.
- **RNF-4 Mantenibilidad:** la aplicación debe ser fácilmente modificable debido a su carácter modular, facilitando el mantenimiento para el desarrollador. Además, **el uso de SonarCloud** ayuda a asegurar la calidad del código mediante el análisis continuo, lo que permite identificar y corregir errores potenciales y optimizar el código, favoreciendo así la mantenibilidad a largo plazo.
- **RNF-5 Escalabilidad:** Eco City Tours debe ser capaz de gestionar eficientemente un crecimiento continuo en el número de usuarios, adaptándose sin problemas para ofrecer un rendimiento óptimo incluso en situaciones de alta demanda.
- **RNF-6 Soporte:** la aplicación debe funcionar en versiones actuales de Android sin problemas de rendimiento o fallos en alguna de sus funcionalidades.

B.4. Especificación de requisitos

Actores del sistema

Actor	A01
Nombre:	Usuario
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Persona que interactúa con la aplicación Eco City Tour para generar y gestionar rutas turísticas personalizadas.
Tipo	Usuario
Objetivo	Generar, visualizar y personalizar rutas turísticas basadas en puntos de interés y preferencias personales.
Responsabilidades	<ul style="list-style-type: none"> ■ Completar el formulario de preferencias para generar una ruta. ■ Visualizar detalles de puntos de interés. ■ Añadir o eliminar puntos de interés de la ruta. ■ Iniciar y detener el seguimiento de ubicación.
Relaciones con casos de uso	CU01(B.8), CU02.1(B.11), CU2.2(B.12), CU04(B.17), CU05(B.18).

Tabla B.1: A01 - Usuario

Actor	A02
Nombre:	Cloud Firestore
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Servicio que posibilita el guardado y carga de las rutas obtenidas por la aplicación.
Tipo	Sistema
Objetivo	Guardar y cargar Eco City Tours generados por el Usuario (B.1).
Responsabilidades	<ul style="list-style-type: none"> ■ Guardar ruta generada. ■ Cargar en el mapa el Eco City Tour guardado en base de datos. ■ Borrar los Eco City Tour almacenados.
Relaciones con casos de uso	CU04(B.17), CU05(B.18)

Tabla B.2: A02 - Cloud Firestore

Actor-ID	A03
Nombre:	Gemini-pro 1.5
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Servicio externo que proporciona una lista de puntos de interés (POI) basados en las propiedades de configuración de la ruta.
Tipo	Sistema
Objetivo	Suministrar información relevante sobre puntos de interés en función de los criterios del usuario.
Responsabilidades	<ul style="list-style-type: none"> ▪ Consultar la información de PDI según las preferencias de búsqueda del usuario. ▪ Enviar los datos de los <i>Punto de Interés (PDI)</i> a la aplicación para generar la ruta.
Relaciones con casos de uso	CU01.1(B.9)

Tabla B.3: A03 - Gemini Pro 1.5

Actor-ID	A04
Nombre:	Google Places Service
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Servicio externo encargado de mejorar la información sobre los puntos de interés proporcionando detalles adicionales como descripciones, fotos y valoraciones.
Tipo	Sistema
Objetivo	Enriquecer la información de los <i>Punto de Interés</i> para que el usuario obtenga datos detallados y actualizados sobre cada lugar.
Responsabilidades	<ul style="list-style-type: none"> ■ Proveer información detallada sobre los PDI. ■ Enviar las descripciones, fotos y valoraciones de los PDI a la aplicación.
Relaciones con casos de uso	CU01.1 (B.9)

Tabla B.4: A04 - Google Places Service

Actor-ID	A05
Nombre:	Google Direction Service
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Servicio externo encargado de generar una ruta actualizada entre los PDI que componen el Eco City Tour.
Tipo	Sistema
Objetivo	Proveer la mejor ruta a mostrar en el mapa que conecte los PDI de cada <i>Eco City Tour</i> en cada momento.
Responsabilidades	<ul style="list-style-type: none"> ■ Generar el camino que ha de seguir el usuario de manera más corta según el medio de transporte elegido. ■ Actualizar en cada momento el camino más corto entre los distintos lugares.
Relaciones con casos de uso	CU01.1 (B.9) CU02 (B.10)

Tabla B.5: A05 - Google Direction Service

Actor-ID	A06
Nombre:	Crashlytics
Versión	1.0
Autor	Fernando Pisot Serrano
Descripción	Servicio externo encargado de registrar en todo momento si se produce un error en la ejecución de la aplicación.
Tipo	Sistema
Objetivo	Dejar constancia en un log de los errores que se puedan llevar a cabo en la ejecución.
Responsabilidades	<ul style="list-style-type: none"> ■ Detectar errores fatales en la ejecución de la aplicación. ■ Registrar en la nube los detalles que formaron la excepción, para posterior análisis.
Relaciones con casos de uso	CU06 (B.19)

Tabla B.6: A06 - Crashlytics

Casos de uso

CU00 - Caso de uso de inicialización. Configuración GPS

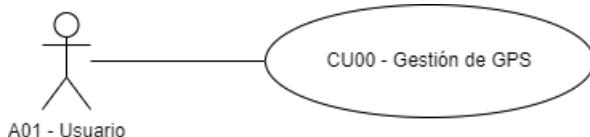


Figura B.1: Diagrama de caso de uso CU00 - Configuración GPS

Este caso de uso es fundamental, ya que sin el permiso de GPS o la activación del sensor, el resto de funcionalidades de la aplicación no pueden ejecutarse. Para mayor claridad y facilitar la lectura de los diagramas restantes, se presenta por separado, destacando su papel como requisito previo para las demás operaciones.

CU00	Configuración GPS
Versión	1.0
Actor	A01 Usuario (B.1)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-1, RF-2
Descripción	Configura la aplicación para tener acceso a la ubicación mediante GPS.
Precondición	La aplicación está instalada y abierta por primera vez o tras haber revocado permisos anteriormente.
Acciones	<ol style="list-style-type: none"> 1. La aplicación solicita permiso para el uso del GPS. 2. El usuario concede el permiso. 3. La aplicación detecta si el GPS está activado. 4. Si el GPS no está activado, solicita al usuario activarlo.
Postcondición	La aplicación tiene acceso a la ubicación en tiempo real.
Excepciones	<ul style="list-style-type: none"> ■ El usuario no concede el permiso de GPS. ■ El usuario no activa el GPS cuando se le solicita.
Importancia	Alta
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Permiso concedido: La aplicación solicita y obtiene acceso al GPS cuando el usuario concede el permiso. ■ Prueba 2 - Permiso denegado: El usuario deniega el permiso. La aplicación no avanza al mapa y no permite operar sobre la aplicación. ■ Prueba 3 - GPS desactivado: El GPS está apagado. La aplicación solicita activación y detecta si el usuario lo enciende.

Tabla B.7: CU00 Configuración GPS

Casos de uso general

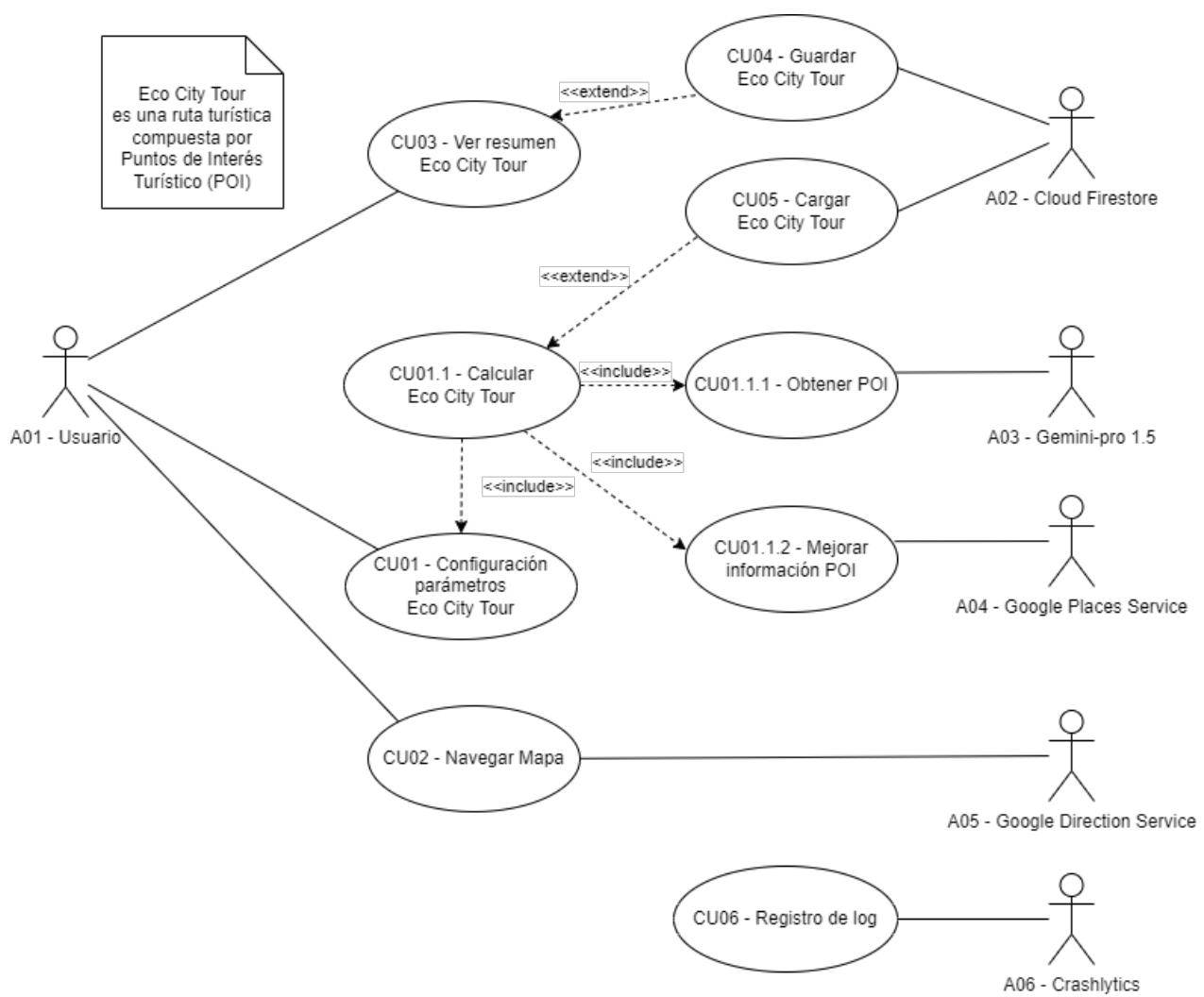


Figura B.2: Diagrama de casos de uso general de Eco City Tours

CU01 - Configuración de parámetros Eco City Tour

CU01	Configuración parámetros Eco City Tour
Versión	1.0
Actor	A01 Usuario (B.1)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-5
Descripción	Configurar los parámetros necesarios para generar un Eco City Tour personalizado según las propiedades de configuración de la ruta.
Precondición	El usuario accede a la pantalla de configuración.
Acciones	<ol style="list-style-type: none"> 1. El usuario completa el formulario de preferencias, incluyendo el lugar, número de PDI, selecciona un asistente IA (cultura, aventura o romántico), selecciona un medio de transporte y tiempo máximo.
Postcondición	Los parámetros quedan configurados y listos para generar la ruta.
Excepciones	<ul style="list-style-type: none"> ■ El usuario no completa el formulario de configuración. ■ Error en la carga de los datos de configuración.
Importancia	Alta
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Configuración correcta: El usuario completa el formulario correctamente con todos los campos requeridos. El sistema guarda los parámetros y los muestra confirmados en el log del modo debug. ■ Prueba 2 - Campos vacíos: El usuario deja el formulario sin modificar. El sistema muestra un <i>Eco City Tour</i> en la mejor ciudad del mundo: Salamanca. ■ Prueba 3 - Valores límites: El usuario introduce valores límite en los campos. El sistema valida los parámetros y los procesa correctamente.

Tabla B.8: CU01 Configuración parámetros Eco City Tour

CU01.1 - Calcular Eco City Tour

CU01.1	Calcular Eco City Tour
Versión	1.0
Actor	A03 (B.3), A04 (B.4), A05 (B.5)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-6, RF-8
Descripción	Generar una ruta optimizada conectando los puntos de interés seleccionados en función de las propiedades de configuración de la ruta.
Precondición	Los parámetros han sido configurados correctamente.
Acciones	<ol style="list-style-type: none"> 1. El usuario confirma la configuración de preferencias. 2. El sistema consulta un LLM para obtener PDI basados en las propiedades de configuración de la ruta. 3. El sistema envía los PDI a Google Places para obtener información mejorada. 4. El sistema consulta un servicio de optimización de rutas para generar la ruta optimizada.
Postcondición	La ruta optimizada es calculada y lista para ser visualizada.
Excepciones	<ul style="list-style-type: none"> ■ Fallo en la conexión con el LLM. ■ Error en el servicio de optimización de rutas.
Importancia	Alta
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Configuración correcta: El usuario confirma los parámetros correctamente. El sistema calcula la ruta sin errores y muestra los <i>Punto de Interés</i> optimizados. ■ Prueba 2 - Fallo en el LLM: El sistema no obtiene respuesta del LLM. Se muestra un mensaje de error y se sugiere reintentar la consulta. ■ Prueba 3 - Servicio de rutas inalcanzable: El sistema no puede conectarse al servicio de optimización de rutas (Google Directions). El sistema notifica el fallo al usuario y permite volver a intentarlo. ■ Prueba 4 - Valores límite en parámetros: El usuario configura un número mínimo y máximo de PDI. El sistema muestra correctamente estos valores calculando la ruta.

CU02 - Navegar en el mapa

CU02	Navegar en el mapa
Versión	1.0
Autor	Fernando Pisot Serrano
Actor	A01 Usuario (B.1)
Requisitos asociados	RF-4
Descripción	El usuario puede desplazarse y explorar el mapa interactivo de la aplicación.
Precondición	El mapa está visible.
Acciones	<ol style="list-style-type: none"> 1. El usuario explora el mapa desplazándose y haciendo zoom.
Postcondición	El usuario navega por el mapa para observar los PDI y la ruta.
Importancia	Media
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Desplazamiento funcional: El usuario desplaza el mapa arrastrando la pantalla con un gesto táctil. El sistema responde correctamente mostrando las áreas correspondientes sin retraso injustificado. ■ Prueba 2 - Zoom funcional: El usuario realiza gestos de zoom in y zoom out en el mapa. El sistema aumenta y reduce el nivel de zoom de manera fluida. ■ Prueba 3 - Carga de marcadores (PDI): Al navegar, los puntos de interés (PDI) se cargan dinámicamente y son visibles sin retrasos en la interfaz. ■ Prueba 4 - Valores límites de zoom: El sistema limita correctamente el nivel de zoom máximo y mínimo, evitando desplazamientos inválidos.

Tabla B.10: CU02 Navegar en el mapa

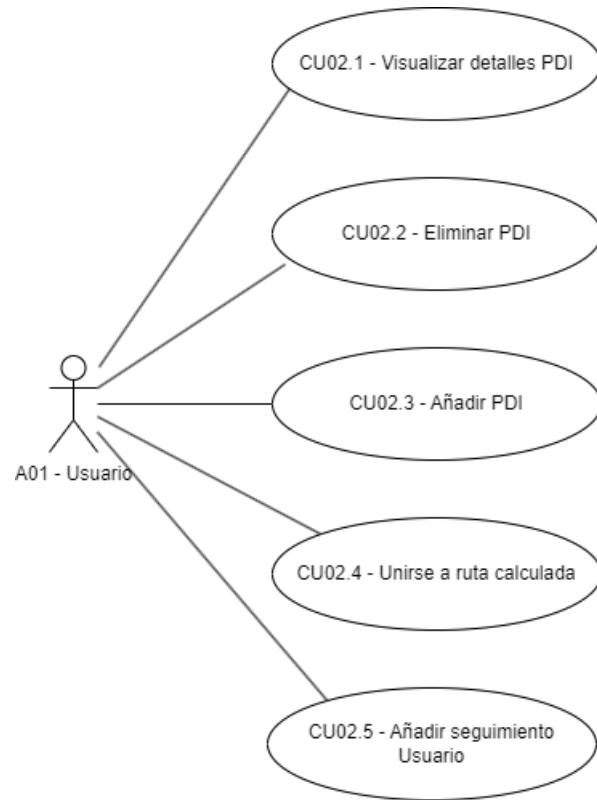


Figura B.3: Diagrama de caso de uso CU02 - Navegar en el mapa

CU02.1	Visualizar detalles de <i>Punto de Interés (PDI)</i>
Versión	1.0
Actor	A01 Usuario (B.1)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-6, RF-11
Descripción	El usuario puede obtener información detallada sobre un punto de interés seleccionado en el mapa.
Precondición	La ruta y los puntos de interés están visibles en el mapa.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un marcador de punto de interés en el mapa. 2. El sistema muestra la información detallada del punto de interés.
Postcondición	La información detallada del punto de interés es visible para el usuario.
Excepciones	<ul style="list-style-type: none"> ■ Fallo en la carga de información del <i>Punto de Interés</i> debido a problemas de conectividad. ■ Error en el servicio externo de obtención de información.
Importancia	Alta
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Información accesible: El usuario selecciona un marcador de <i>PDI</i> en el mapa. El sistema carga y muestra correctamente el nombre, descripción, imagen y detalles del <i>Punto de Interés</i>. ■ Prueba 2 - Punto sin detalles: El marcador seleccionado corresponde a un <i>PDI</i> que no tiene información asociada. El sistema muestra la imagen por defecto y el texto en blanco.

Tabla B.11: CU02.1 Visualizar detalles de los *Punto de Interés (PDI)*

CU02.2	Eliminar <i>Punto de Interés</i>
Versión	1.0
Actor	A01 (B.1), A05 (B.5)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-7, RF-8
Descripción	El usuario puede eliminar puntos de interés de la ruta desde el mapa.
Precondición	La ruta ha sido generada y los puntos de interés están visibles en el mapa.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un punto de interés en el mapa. 2. El sistema elimina el punto de interés de la ruta. 3. El sistema recalcula la ruta optimizada sin el punto eliminado.
Postcondición	La ruta es recalculada sin el punto de interés eliminado.
Excepciones	<ul style="list-style-type: none"> ■ Error en el recálculo de la ruta. ■ Problemas de conectividad al intentar actualizar la ruta.
Importancia	Media
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Eliminación exitosa: El usuario selecciona un PDI y el sistema lo elimina correctamente de la ruta. El recálculo de la ruta es exitoso. ■ Prueba 2 - Error en el recálculo: El sistema no logra recalcular la ruta después de la eliminación del PDI. Se muestra un mensaje de error y la ruta previa se mantiene visible.

Tabla B.12: CU02.2 Eliminar *Punto de Interés*

CU02.3	Añadir <i>Punto de Interés (PDI)</i>
Versión	1.0
Actor	A01 Usuario B.1
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-8, RF-9
Descripción	El usuario puede añadir manualmente un nuevo PDI a la ruta introduciendo su nombre en la barra de búsqueda.
Precondición	La ruta ha sido generada previamente.
Acciones	<ol style="list-style-type: none"> 1. El usuario introduce el nombre de un lugar en la barra de búsqueda. 2. El sistema muestra el resultado de la búsqueda y el usuario lo selecciona para añadirlo. 3. El sistema recalcula la ruta optimizada incluyendo el nuevo <i>Punto de Interés</i>.
Postcondición	El nuevo lugar es añadido a la ruta, y la ruta optimizada es recalculada.
Excepciones	<ul style="list-style-type: none"> ■ El lugar no se encuentra en el servicio de búsqueda. ■ Fallo en el recálculo de la ruta.
Importancia	Baja
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Añadir PDI exitoso: El usuario introduce el nombre del lugar, el sistema lo encuentra y lo añade correctamente. La ruta es recalculada y optimizada. ■ Prueba 2 - Lugar no encontrado: El usuario introduce un nombre de lugar inexistente. El sistema muestra el listado de lugares vacío pero permite al usuario volver a buscar otro lugar o volver al mapa. ■ Prueba 3 - Cálculo de ruta muy larga: El usuario selecciona un lugar válido, pero no es el que se esperaba al encontrarse muy lejos. El sistema es capaz de ejecutar el cálculo aunque le lleve más tiempo al ser una tarea síncrona. El PDI se puede eliminar manualmente por el usuario.

Tabla B.13: CU02.3 Añadir *Punto de Interés (PDI)*

CU02.4	Unirse a la ruta calculada
Versión	1.0
Actor	A01 Usuario B.1
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-10
Descripción	Permite al usuario unirse a una ruta previamente generada desde su ubicación actual.
Precondición	Una ruta ya ha sido generada y está activa. No hace falta que esté activado el seguimiento en vivo.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción para unirse a la ruta desde su ubicación actual. 2. El sistema calcula la ruta más corta para conectar la ubicación actual del usuario con la ruta generada.
Postcondición	El usuario es guiado desde su ubicación actual hasta la ruta generada.
Excepciones	<ul style="list-style-type: none"> ■ Error en el cálculo de la ruta de conexión.
Importancia	Media
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Unirse a la ruta exitosa: El usuario selecciona la opción para unirse y el sistema calcula correctamente la ruta de conexión. ■ Prueba 2 - Ubicación inválida: El sistema no puede determinar la ubicación actual del usuario. Se muestra la unión al punto más cercano viable del usuario. p.e. si está en mitad del mar lo situaría en la costa. ■ Prueba 3 - Ruta previa no generada: El usuario intenta unirse sin que exista una ruta activa pues es el único punto de la ruta. No se muestra ninguna ruta pero la ubicación formará parte de ella si se añade otra ruta. ■ Prueba 4 - Unirse a ruta si está ya unido: El usuario no puede tener activado el botón de unirse a ruta para evitar el conflicto de unirse múltiples veces.

Tabla B.14: CU02.4 Unirse a la ruta calculada

CU02.5	Añadir seguimiento del usuario
Versión	1.0
Actor	A01 Usuario (B.1)
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-3
Descripción	Permite al usuario activar o desactivar el seguimiento de su posición en tiempo real en el mapa.
Precondición	La aplicación tiene acceso a la ubicación del usuario.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de activar o desactivar el seguimiento de su ubicación en el mapa. 2. El sistema ajusta el mapa para mostrar o dejar de mostrar el movimiento del usuario en tiempo real.
Postcondición	El mapa sigue o deja de seguir la posición del usuario en tiempo real.
Excepciones	<ul style="list-style-type: none"> ■ Problemas de conexión con el GPS. ■ Pérdida de señal GPS.
Importancia	Baja
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Seguimiento activado: El usuario activa el seguimiento. El mapa ajusta su vista en tiempo real conforme a la ubicación del usuario. ■ Prueba 2 - Seguimiento desactivado: El usuario desactiva el seguimiento. El mapa deja de ajustarse a la posición del usuario. ■ Prueba 3 - Señal GPS débil: La señal del GPS se pierde temporalmente. El sistema mantendrá la ubicación del usuario en la última ubicación proporcionada (realizado por el Sistema).

Tabla B.15: CU02.5 Añadir seguimiento del usuario

CU03 - Ver resumen Eco City Tour

CU03	Ver resumen Eco City Tour
Versión	1.0
Actor	A01 Usuario B.1
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-4, RF-8
Descripción	Muestra un resumen de la ruta generada, incluyendo la distancia, duración y medio de transporte.
Precondición	La ruta ha sido generada.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la pantalla de resumen. 2. La aplicación muestra los detalles de la ruta, incluyendo distancia total, tiempo estimado y transporte elegido.
Postcondición	El resumen es visible para el usuario.
Excepciones	<ul style="list-style-type: none"> ■ Error en la carga de los datos de la ruta.
Importancia	Baja
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Resumen completo visible: El usuario accede a la pantalla de resumen y todos los detalles (distancia, duración y transporte) se muestran correctamente. ■ Prueba 2 - Valores límite de distancia y tiempo: Validar que el sistema maneja correctamente valores grandes (e.g., rutas muy largas) y pequeños (e.g., rutas cortas de menos de 1 km). ■ Prueba 3 - Eco City Tour vacío: Desde la pantalla se eliminan todos los PDI lo que devuelve al usuario a la pantalla del mapa pues no tiene información útil que mostrar.

Tabla B.16: CU03 Ver resumen Eco City Tour

CU04 - Guardar Eco City Tour

CU04	Guardar Eco City Tour
Versión	1.0
Actor	A01 B.1 , A02 B.2
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-12
Descripción	Permite al usuario guardar la ruta generada para acceder a ella en el futuro.
Precondición	La ruta ha sido generada.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de guardar la ruta.
Postcondición	La ruta queda guardada en el sistema.
Excepciones	<ul style="list-style-type: none"> ▪ Error en el guardado de la ruta. ▪ Problemas de conexión con la base de datos.
Importancia	Media
Casos de Prueba	<ul style="list-style-type: none"> ▪ Prueba 1 - Guardado exitoso: El usuario guarda una ruta y verifica que esta aparece correctamente en el sistema. ▪ Prueba 2 - Valores límite: Comprobar el guardado de rutas con diferentes tamaños (e.g., desde una ruta vacía hasta rutas con el número máximo permitido). ▪ Prueba 3 - Error en la conexión: Simular la pérdida de conexión (al limitar los permisos de escritura en Firestore) al intentar guardar una ruta y verificar que el sistema muestra un mensaje de error claro sin comprometer la estabilidad.

Tabla B.17: CU04 Guardar Eco City Tour

CU05 - Cargar Eco City Tour

CU05	Cargar Eco City Tour
Versión	1.0
Actor	A01 B.1, A02 B.2
Autor	Fernando Pisot Serrano
Requisitos asociados	RF-12
Descripción	Permite al usuario cargar una ruta guardada previamente.
Precondición	El usuario ha guardado al menos una ruta.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de cargar una ruta guardada. 2. El usuario selecciona un Eco City Tour guardado previamente que será cargado.
Postcondición	La ruta es cargada y visible en el mapa.
Excepciones	<ul style="list-style-type: none"> ■ Error en la carga de la ruta guardada. ■ Problemas de conexión con la base de datos.
Importancia	Media
Casos de Prueba	<ul style="list-style-type: none"> ■ Prueba 1 - Carga exitosa: El usuario selecciona y carga una ruta guardada, verificando que esta aparece correctamente en el mapa. ■ Prueba 2 - Valores límite: Validar la carga de rutas guardadas vacías y rutas con el número máximo de PDI. ■ Prueba 3 - Rutas no existente: La pantalla de carga solo muestra los Eco City Tour asociados al usuarios y no todos los creados por todos los usuarios y si no tiene ninguno es capaz de mostrar la pantalla vacía con un mensaje amigable.

Tabla B.18: CU05 Cargar Eco City Tour

CU06 - Registro de log

CU06	Registro de log
Versión	1.0
Actor	A06 (B.6)
Autor	Fernando Pisot Serrano
Descripción	Registra los eventos y actividades del usuario en la aplicación para fines de monitorización y depuración.
Precondición	La aplicación está en funcionamiento.
Acciones	<ol style="list-style-type: none"> 1. La aplicación registra automáticamente los eventos relevantes.
Postcondición	Los eventos quedan registrados en el sistema.
Excepciones	<ul style="list-style-type: none"> ▪ Error en la conexión con el sistema de registros. ▪ Fallo en la escritura de los eventos en el log.
Importancia	Baja
Casos de Prueba	<ul style="list-style-type: none"> ▪ Prueba 1 - Registro exitoso: Verificar que los eventos importantes (inicio de sesión, selección de ruta, activación del GPS) se registran correctamente en el sistema de logs. ▪ Prueba 2 - Evento límite: Validar que se registran eventos con mensajes largos o múltiples detalles (e.g., error detallado con trazas largas). ▪ Prueba 3 - Registro de una caída de la aplicación: Se puede simular un cierre inesperado que debe ser notificado y visible desde la consola de Crashlytics.

Tabla B.19: CU06 Registro de log

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo, se detallan las especificaciones de diseño del proyecto, enfocadas en los aspectos fundamentales para el desarrollo de la aplicación. Se describen cómo se organizan los datos que lo componen, el diseño arquitectónico, y el diseño procedimental con la interacción entre los objetos diseñados. Estas especificaciones son clave para asegurar la comprensión y la implementación de cada uno de los elementos que componen *Eco City Tours*.

C.2. Diseño de datos

En esta sección se presenta el diseño de datos del sistema a un alto nivel de abstracción. Para explicar dicho diseño se usarán varios diagramas donde se muestran las dos principales entidades del sistema y las relaciones entre ellas, centrándose en su estructura y atributos relevantes.

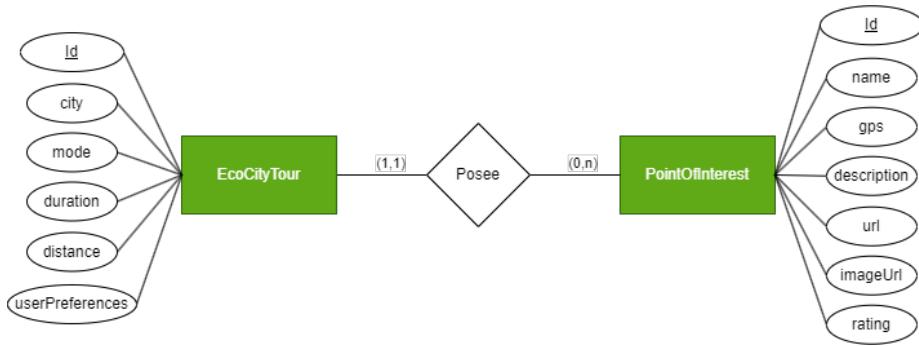
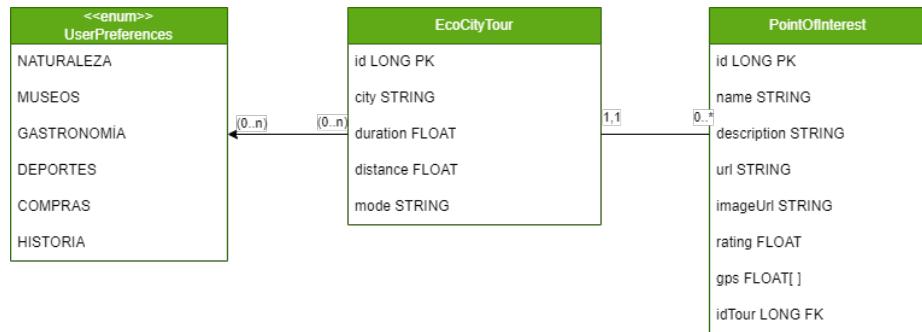
Figura C.1: Diagrama de entidades. Eco City Tour y *Punto de Interés*

Figura C.2: Diagrama relacional.

A continuación, se describen las entidades, sus atributos y las relaciones que existen entre ellas.

Entidades del Sistema

- **EcoCityTour:** Representa una ruta turística generada por la aplicación. Incluye la información general sobre el recorrido.
 - **city:** Ciudad, pueblo o lugar en el que se realiza la ruta.
 - **mode:** Medio de transporte elegido por el usuario (a pie o bicicleta).
 - **duration:** Duración estimada del recorrido en metros. Rango de valores permitidos (0-999 km.)
 - **distance:** Distancia total de la ruta en segundos. Rango de valores permitidos (0-24 horas).

- **userPreferences:** Preferencias del usuario: Naturaleza, Museos, Gastronomía, Deportes, Compras e Historia.
- **PointOfInterest:** Representa un *Punto de Interés* dentro de la ruta.
 - **name:** Nombre del lugar.
 - **gps:** Coordenadas GPS del sitio (latitud y longitud medida en grados). Rango válido: -90° a 90° para latitud y -180°-180° para longitud.
 - **description:** Breve descripción del lugar.
 - **url:** Enlace a información adicional sobre el lugar.
 - **imageUrl:** URL de la imagen representativa del lugar.
 - **rating:** Calificación promedio del punto de interés. Valoraciones numéricas de 0 a 5 estrellas.

Relaciones entre Entidades

- **EcoCityTour – PointOfInterest:** Una relación donde:
 - Un **EcoCityTour** posee **0..n** puntos de interés (**PointOfInterest**).
 - Cada **PointOfInterest** pertenece a exactamente un **EcoCity-Tour**.

C.3. Diseño arquitectónico

El diseño arquitectónico de la aplicación *Eco City Tours* se basa en una estructura modular, organizada para garantizar una clara separación de responsabilidades, facilitando el desarrollo, mantenimiento y escalabilidad del sistema.

La arquitectura a este nivel de abstracción se muestra en la Figura C.3 y está dividida en cinco capas funcionales principales: **Modelo**, **Interfaz de Usuario**, **Lógica de Negocio**, **Servicios Externos**, y **Persistencia**, cada una de las cuales agrupa componentes con responsabilidades específicas.

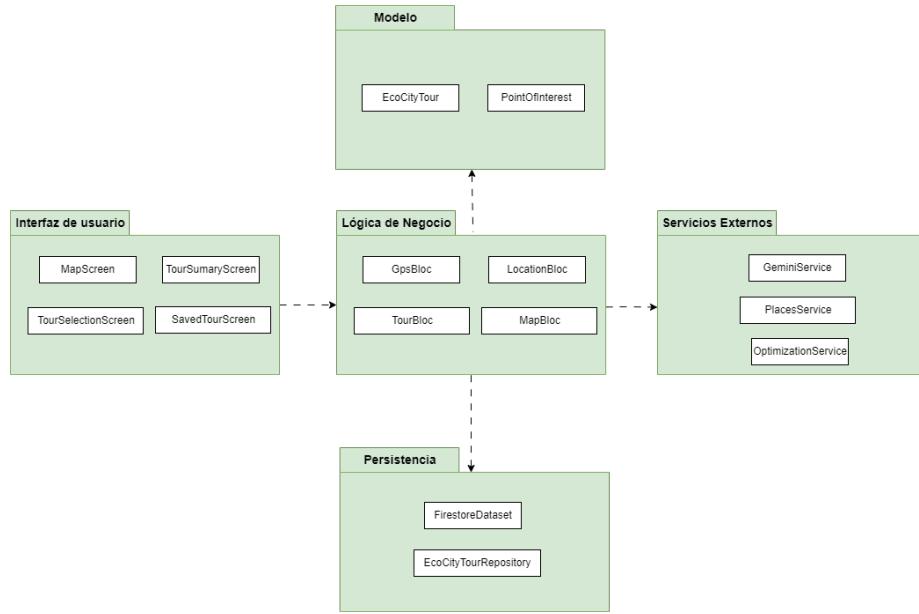


Figura C.3: Diagrama arquitectónico de paquetes y dependencias

Las responsabilidades de cada capa son las siguientes:

- **Modelo**: Define las estructuras de datos principales utilizadas en el sistema, como los puntos de interés (*PointOfInterest*) y los tours (*EcoCityTour*).
- **Interfaz de Usuario**: Contiene los componentes que permiten la interacción directa con el usuario, es decir, las pantallas que permiten configurar un tour y visualizarlo.
- **Lógica de Negocio**: Gestiona el flujo de información entre la interfaz de usuario y los servicios externos, utilizando los gestores de estado basados en BLoC.
- **Servicios Externos**: Incluye los servicios que interactúan con APIs externas, como Google Places, Gemini, y el servicio que calcula la ruta a seguir por el usuario.
- **Persistencia**: Maneja el almacenamiento y recuperación de datos, integrándose con Firestore para guardar información relevante.

Desde una perspectiva más específica, el diagrama de clases (Figura C.4) ilustra la relación entre los paquetes principales y sus clases internas. Este

nivel proporciona una visión más precisa de las interacciones entre los paquetes antes descritos.

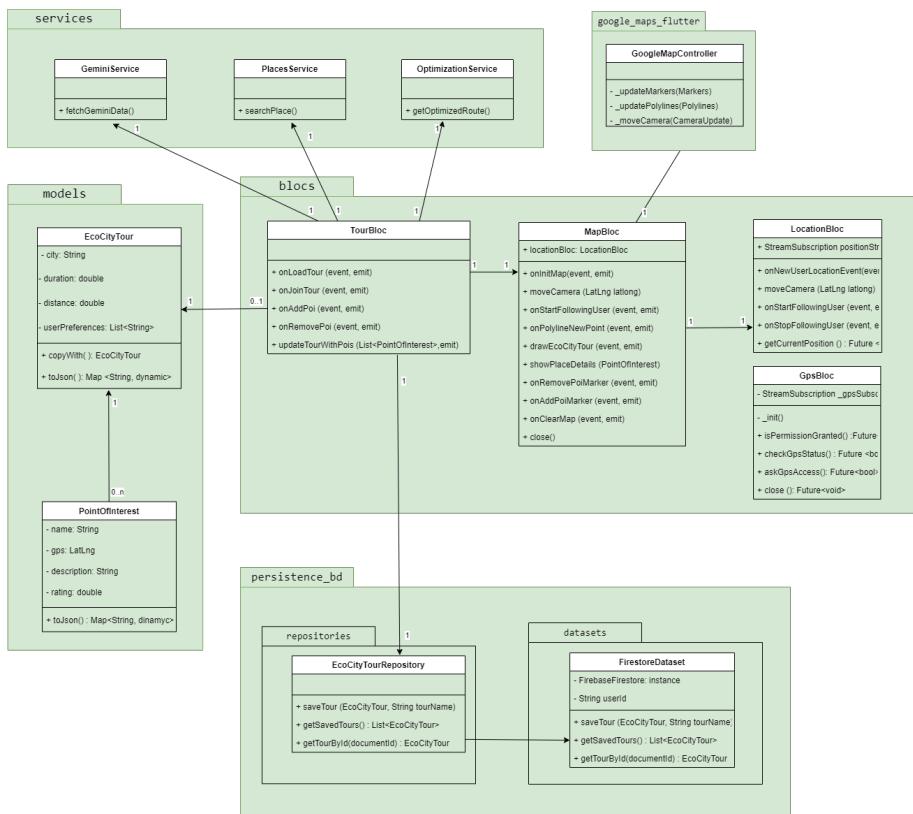


Figura C.4: Diagrama arquitectónico detallado de paquetes y clases de *Eco City Tours*

Este diagrama ilustra cómo se relaciona el modelo de datos con la lógica de negocio (usando BLoC) y el servicio de persistencia de los datos del modelo. Además, se describe cómo la lógica de negocios se apoya en servicios externos relacionados con la interacción con modelos de IA y con interacción en mapas geográficos.

Esta arquitectura propuesta basada en patrones de orientación a objetos bien conocidos [4] garantiza un diseño cohesivo, con bajo acoplamiento y alta cohesión, la extensibilidad del sistema coincidiendo también con cada carpeta dentro de la estructura del proyecto Flutter lo que garantiza también una fácil mantenibilidad.

A continuación, se describen los paquetes y componentes principales representados en el diagrama de clases:

- Services:

- **GeminiService**: Servicio encargado de interactuar con el modelo LLM *Gemini* para obtener información sobre *Punto de Interés* y generar recomendaciones personalizadas.
- **PlacesService**: Proporciona datos relacionados con lugares de interés mediante la integración con la API de *Google Places*.
- **OptimizationService**: Se encarga de calcular rutas optimizadas entre puntos de interés, teniendo en cuenta las propiedades de configuración de la ruta.

- **google_maps_flutter**:

- Clase del paquete oficial de Flutter para administrar la interacción con los mapas de Google en la interfaz de usuario. Su función principal es permitir la visualización y gestión de las rutas generadas.
- **models**: existen dos clases principales en las que se basa el diseño de la aplicación.
 - **Punto de Interés**: Clase que modela un PDI, almacenando información relevante como nombre, ubicación y descripción.
 - **EcoCityTour**: Clase que representa un tour turístico completo, que incluye una lista del modelo anterior y datos como duración, distancia y nombre del lugar donde se ha generado la ruta turística.

- **blocs**: este paquete gestionará la lógica del gestor de estados de la aplicación.

- **TourBloc**: Responsable de la gestión del estado relacionado con los tours, incluyendo la generación y modificación de rutas.
- **MapBloc**: Administra el estado relacionado con la visualización en el mapa, como el trazado de rutas.
- **LocationBloc**: Obtiene la información de la posición del usuario actual y la guarda si el stream de datos está activo.
- **GpsBloc**: Gestiona si el dispositivo tiene el GPS activado y la aplicación tiene permisos para el uso de dicho sensor.

- **persistence_bd**:

- **Repositories:** `EcoCityTourRepository`: Implementa la lógica necesaria para, guardar y cargar información de un *EcoCityTour*.
- **Datasets:** `FirestoreDataset`: Clase que gestiona la persistencia de datos en la base de datos en este caso concreto de Firestore.

Con esta organización, el sistema obtenido es flexible pero resulta dadas las pruebas realizadas muy robusto. La aplicación de patrones de diseño dan soporte a la extensibilidad en su futuro mantenimiento. Por ejemplo se pueden incluir mediante extensiones nuevos gestores de estado y nuevas bases de datos de almacenamiento.

Patrón de diseño *Business Logic Component* (BLoC)

El patrón de diseño **BLoC** [2] es fundamental en la arquitectura de la aplicación *Eco City Tours*. Este patrón se caracteriza por tener las siguientes propiedades:

- **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 mantible.
- **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.

En el contexto de Flutter, el patrón *Business Logic Component* utiliza un enfoque basado en flujos o *streams* y eventos (ver Fig. C.5), permitiendo que los estados y eventos fluyan entre los componentes sin acoplar directamente la lógica de negocio con la interfaz gráfica.

La aplicación *Eco City Tours* implementa este patrón de diseño a través de los componentes principales, cada uno con una funcionalidad clara y única.

- ***GpsBloc***: Gestiona tanto los permisos de uso de GPS como su activación actual en el dispositivo donde esté funcionando la aplicación.
- ***LocationBloc***: Gestiona la localización del usuario a través de un stream que guarda la posición GPS por la que va circulando.
- ***TourBloc***: Gestiona toda la lógica relacionada con los tours, desde la generación y modificación de rutas hasta la interacción con los servicios externos como *Servicio LLM* y *Google Places*.
- ***MapBloc***: Gestiona el estado y las interacciones del mapa, como la visualización de rutas generadas, la actualización de puntos de interés.

Un aspecto importante en la implementación del patrón `BLoC` en *Eco City Tours* es el uso de la clase `Equatable`, que permite simplificar la comparación de estados y eventos. En Flutter, cada vez que un estado cambia, se debe notificar a la interfaz de usuario para que se actualice. Esta interacción se alinea con el patrón de diseño Observador, definido en [4], establece una relación entre un sujeto (en este caso, el `BLoC`) y múltiples observadores (diferentes componentes de la interfaz de usuario), asegurando que estos últimos reaccionen a los cambios en el estado del primero. La clase `Equatable` facilita este proceso al definir de manera eficiente si dos instancias de estado o evento son equivalentes, evitando notificaciones redundantes y mejorando el rendimiento del sistema.

En el caso de *Eco City Tours*, todos los `BLoC` usados extienden de `Equatable`, asegurando que la lógica de comparación sea precisa y que la interfaz de usuario solo se actualice cuando sea necesario.

Principios clave del patrón BLoC

El patrón *Business Logic Component* (BLoC) sigue los siguientes principios:

- **Eventos y estados:** La interacción entre la interfaz de usuario y la lógica de negocio se realiza mediante el envío de eventos (**Event**) y la emisión de estados (**State**). Los eventos representan acciones realizadas por el usuario, mientras que los estados reflejan el resultado de procesar esos eventos.
- **Flujos (*Streams*):** El patrón utiliza *streams* de datos reactivos para manejar la comunicación entre la interfaz de usuario y la lógica de negocio.
- **Inmutabilidad:** Tanto los eventos como los estados son inmutables, lo que asegura un comportamiento asíncrono predecible y simplifica el manejo del flujo de datos.

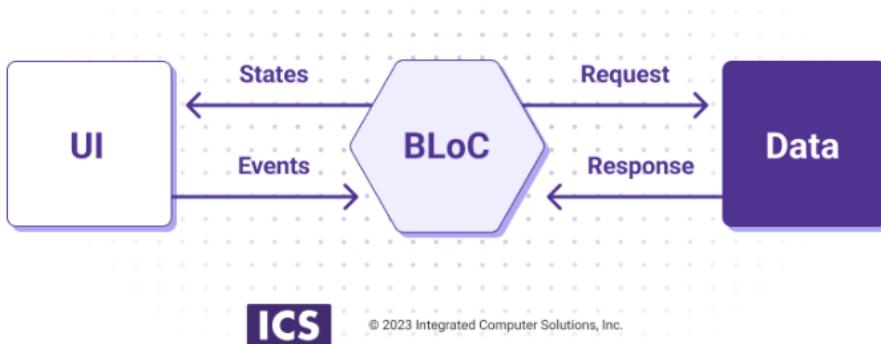


Figura C.5: Diagrama de interacción asíncrona del patrón BLoC en la aplicación

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. [?]. 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.

Listing C.1: Definición de variables de control en el BLoC

```
// Indicates whether GPS is enabled
final bool isGpsEnabled;

// Indicates whether the app has permission to access GPS
final bool isGpsPermissionGranted;

// Getter for the correct state of permissions and GPS
bool get isAllReady =>
    isGpsEnabled &&
    isGpsPermissionGranted;
```

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

Listing C.2: Carga en función del estado

```
class LoadingScreen extends StatelessWidget {
    const LoadingScreen({super.key});

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: BlocBuilder<GpsBloc, GpsState>(
                builder: (context, state) {
                    return state.isAllReady
                        // If GPS and permissions are ready
                        ? const MapScreen()
                        : const GpsAccessScreen();
                },
            ),
        );
    }
}
```

}

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

C.4. Diseño procedimental

En esta sección se explicará el flujo de trabajo relacionado con las tareas más importantes de la aplicación, asociando cada diseño procedimental a sus respectivos **Casos de Uso (CU)** descritos en la subsección **B.4**.

Calcular de *EcoCityTour* desde pantalla de selección de preferencias (CU01.1)

La generación de un nuevo tour turístico una vez que el usuario ha definido sus preferencias es el proceso fundamental que se muestra en la Figura C.6, correspondiente al **CU01.1 Calcular Eco City Tour**.

El proceso comienza con el evento `loadTourEvent()`, disparado por el usuario desde la pantalla de selección (*TourSelectionScreen*). Este evento desencadena la siguiente secuencia de llamadas asíncronas entre objetos:

1. El *TourBloc* inicia la generación del tour solicitando a *ServicioLLM* una lista inicial de *Punto de Interés* basados en los parámetros de configuración de la ruta.
2. *ServicioLLM* devuelve un conjunto de *Punto de Interés*, que luego se enriquecen con información adicional obtenida a través de *Google Places* mediante el método `searchPlacePOI()`.
3. La información enriquecida de los *Punto de Interés* es enviada a *OptimizationService*, que calcula una ruta optimizada utilizando criterios de sostenibilidad y los parámetros de configuración de la ruta. Esto se realiza a través del método `optimizedRoute()`.
4. La ruta optimizada es devuelta como *OptimizedRoute* y se envía al *MapBloc*, que se encarga de gestionar la actualización del mapa en la interfaz de usuario.
5. Finalmente, el *MapBloc* utiliza la ruta optimizada para invocar el método `updateMap()`, mostrando el tour generado en *MapScreen* y permitiendo al usuario interactuar con los puntos de interés.

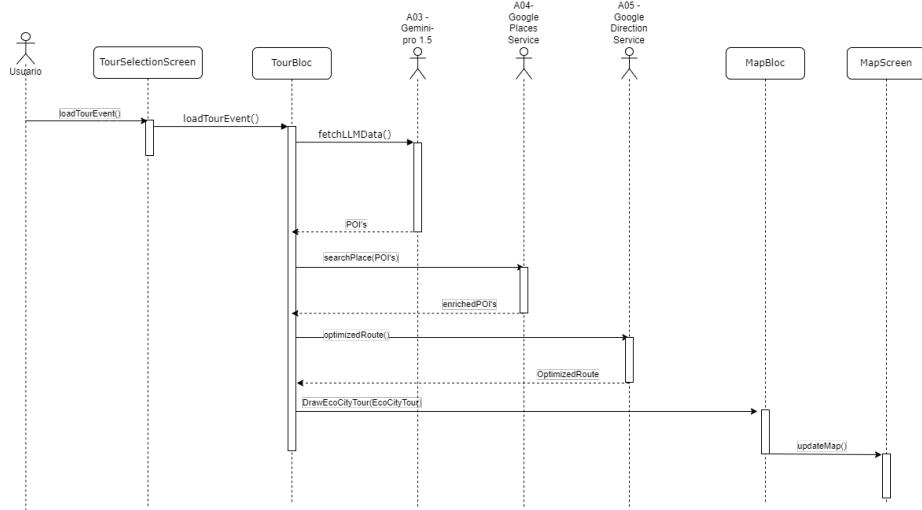


Figura C.6: Diagrama de secuencia - Generación de Eco City Tour

Añadir un *Punto de Interés (PDI)* (CU02.3)

El flujo para añadir un nuevo **PDI** corresponde al **CU02.3 Añadir PDI**. El proceso se inicia cuando el usuario selecciona la opción de añadir un **PDI** desde la interfaz de usuario *MapScreen*.

Este evento desencadena una serie de interacciones entre los componentes principales:

- El *MapBloc* gestiona el estado del mapa.
- El *TourBloc* actualiza la lista de **PDI** en el tour actual.

Finalmente, el sistema actualiza la vista del mapa con el **PDI** añadido y guarda los cambios en el repositorio correspondiente.

La Figura C.7 muestra el diagrama de secuencia de esta funcionalidad.

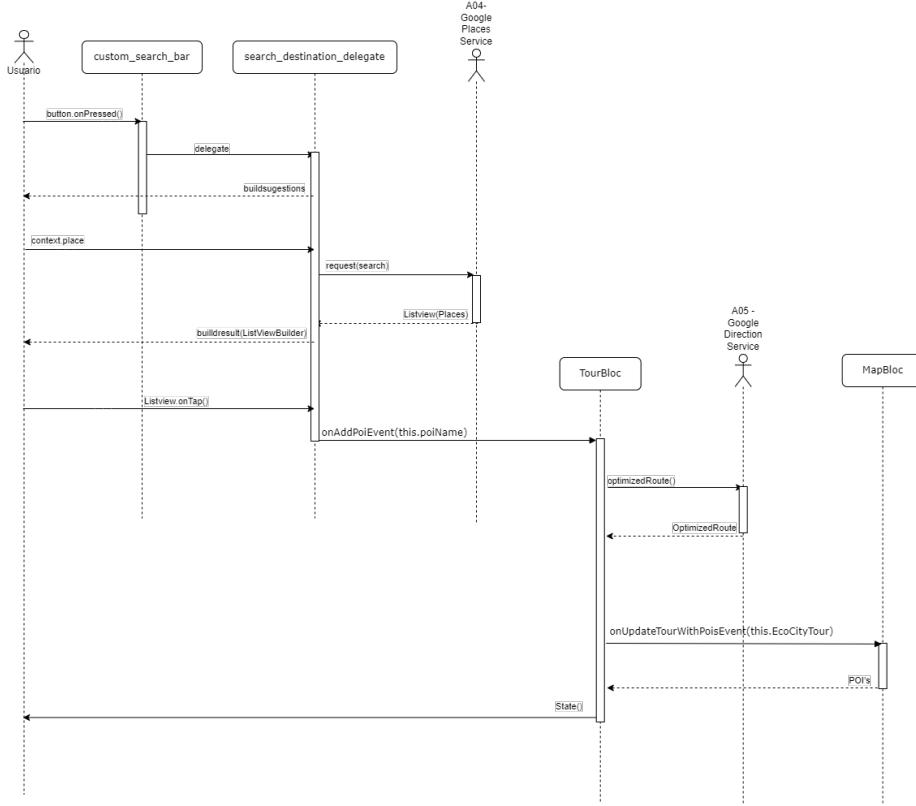


Figura C.7: Diagrama de secuencia - Añadir un PDI

Eliminar un PDI (CU02.2)

El flujo para eliminar un PDI, relacionado con el **CU02.2 Eliminar PDI**, permite al usuario ajustar su ruta eliminando aquellos PDI que no sean relevantes.

El usuario inicia este flujo seleccionando el PDI a eliminar desde:

- *MapScreen* a través del *custom_bottom_sheet*.
- La pantalla de resumen (*poi_list_item*).

Ambos eventos lanzan una actualización en el *TourBloc*, que solicita al *Google Directions Service* recalcular la ruta optimizada. Finalmente, el *MapBloc* actualiza la vista del mapa.

La Figura C.8 muestra el diagrama de secuencia correspondiente.

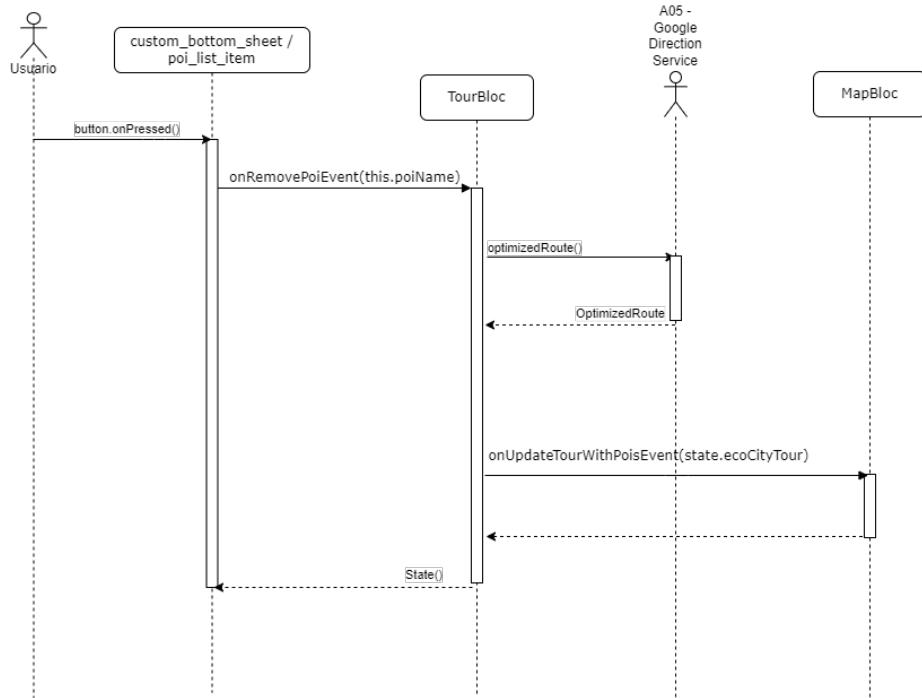


Figura C.8: Diagrama de secuencia - Eliminar un PDI

Guardar un Eco City Tour (CU04)

La funcionalidad de guardar un Eco City Tour, asociada al **CU04 Guardar Eco City Tour**, asegura que los usuarios puedan acceder a sus rutas en el futuro.

El flujo comienza cuando el usuario selecciona la opción de guardar desde la interfaz *SummaryScreen*. Este evento activa el *TourBloc*, que envía los datos al repositorio, donde se almacenan a través del *FirebaseDataset*. Una vez completado el guardado, se notifica al usuario mediante un *Snackbar*.

La Figura C.9 muestra el diagrama de secuencia correspondiente.

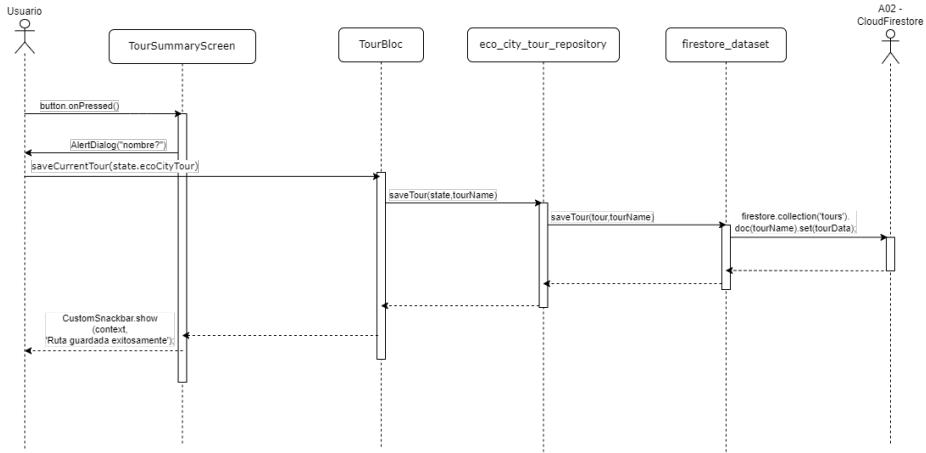


Figura C.9: Diagrama de secuencia - Guardar un Eco City Tour

Cargar un Eco City Tour (CU05)

La funcionalidad de cargar un tour previamente guardado está relacionada con el **CU05 Cargar Eco City Tour**.

El flujo inicia cuando el usuario selecciona un tour desde la lista de tours guardados en la interfaz *TourSelectionScreen*. Este evento activa el *TourBloc*, que recupera los datos del tour desde el repositorio (*FirestoreDataset*). Una vez recuperada la información, el *MapBloc* actualiza la vista del mapa para mostrar el tour cargado.

La Figura C.10 describe el diagrama de secuencia para este proceso.

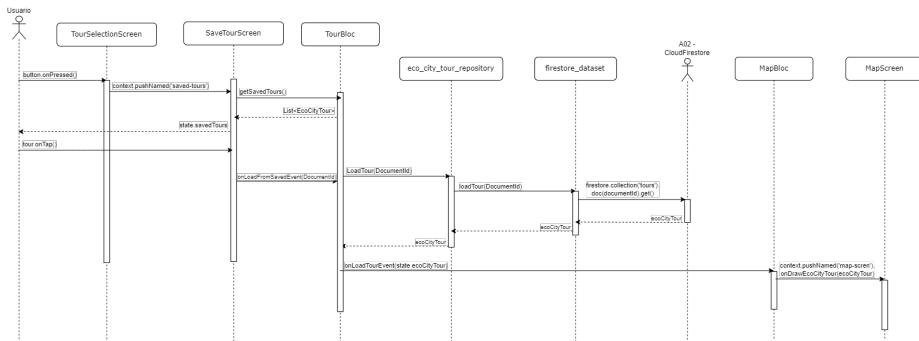


Figura C.10: Diagrama de secuencia - Cargar un Eco City Tour

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este anexo proporciona una guía técnica detallada para desarrolladores que deseen trabajar en el proyecto **Eco City Tours**. Aquí se describe cómo configurar desde el entorno de desarrollo, hasta compilar y ejecutar el proyecto, así como las pruebas realizadas y la configuración de servicios externos necesarios para el correcto funcionamiento de la aplicación.

D.2. Estructura de directorios

Se ha intentado seguir las buenas prácticas de programación adquiridas en mi formación a la hora de organizar los directorios de la siguiente manera:

- /: fichero de la licencia, .gitignore y el documento *Readme* con información del proyecto.
- /**project-docs**/: documentación del proyecto usando la plantilla L^AT_EX proporcionada.
- /**project-prototype**/: dos prototipos de uso de modelos LLM con ejemplos de prompting y técnicas RAG, tanto en un cuaderno Python como en LangFlow así como las conclusiones tras su experimentación.

- **/project-app/**: se trata del proyecto Flutter creado a raíz del asistente de Visual Studio y que fue construido de manera incremental desde una aplicación vacía. Se divide en las siguientes subcarpetas:
 - **/android** : directorio generado automáticamente por Flutter para configurar y compilar la aplicación en la plataforma *Android*.
 - **/test**: contiene las pruebas unitarias y de integración que verifican el correcto funcionamiento de los componentes, servicios y lógica del negocio.
 - **/assets**: incluye recursos estáticos como imágenes, fuentes o archivos de configuración de idioma que utiliza la aplicación.
 - **/lib**: contiene el código fuente principal de la aplicación, incluyendo los widgets de Flutter y la lógica del negocio. Dentro de esta carpeta destacan las siguientes subcarpetas:
 - **/screens**: incluye las distintas pantallas que componen la interfaz de usuario de la aplicación.
 - **/widgets**: contiene componentes reutilizables que forman partes más pequeñas y específicas de la interfaz de usuario, como botones personalizados.
 - **/models**: define las clases fundamentales de datos utilizadas en la aplicación, como '*EcoCityTour*' y '*PointOfInterest*'.
 - **/services**: alberga los servicios responsables de interactuar con las APIs externas, como *Google Maps*, *Google Places* o el modelo **LLM** utilizado *Gemini pro-1.5*.
 - **/blocs**: organiza la lógica del negocio bajo el patrón ***Business Logic Component (BLoC)***, permitiendo manejar el estado de forma eficiente.

D.3. Manual del programador

El objetivo de este manual es servir como referencia para cualquier programador que trabaje en este proyecto. Aquí se detallan los pasos necesarios para configurar el entorno de desarrollo, obtener el código fuente desde el repositorio, compilarlo, ejecutarlo y realizar pruebas. Los pasos son los siguientes:

Entorno de desarrollo

Para trabajar en el desarrollo de **Eco City Tours**, se recomienda configurar el entorno de desarrollo con las siguientes herramientas:

- **Visual Studio Code:** Editor de código recomendado por su integración con Flutter y extensiones útiles como Dart y Flutter.
- **Android Studio:** Utilizado principalmente para configurar emuladores de dispositivos móviles.
- **Flutter SDK:** Framework utilizado para el desarrollo de aplicaciones multiplataforma.
- **Copilot:** Extensión para *Visual Studio Code* que utiliza inteligencia artificial para sugerir código y acelerar el desarrollo.

Pasos para configurar el entorno

1. Descarga e instala **Visual Studio Code** desde <https://code.visualstudio.com/>.
2. Instala las extensiones necesarias para Flutter y Dart:
 - Abre la pestaña de extensiones (**Ctrl+Shift+X**) en Visual Studio Code.
 - Busca e instala las extensiones **Flutter** y **Dart**.
3. Descarga e instala **Android Studio** desde <https://developer.android.com/studio>.
4. Configura los emuladores Android en Android Studio:
 - Abre el **AVD Manager**.
 - Crea un nuevo emulador con las especificaciones necesarias (por ejemplo, Pixel 5 con Android 11 o superior).
5. Instala el **Flutter SDK**:
 - Descarga el SDK desde su web [3]
 - Agrega el binario de Flutter al PATH del sistema:

```
export PATH="$PATH:/ruta/del/flutter/bin"
```

- Verifica la instalación ejecutando:

```
flutter doctor
```

Este comando será muy útil para indicar si falta alguna librería del sistema operativo o indicará si hay problemas pendientes en la configuración.

6. Instala **Copilot**:

- Desde la pestaña de extensiones en Visual Studio Code, busca e instala [GitHub Copilot](#).
- Inicia sesión con tu cuenta de GitHub para activar la extensión.

El uso de Copilot es gratuito para estudiantes que puedan demostrar su vinculación con una cuenta educativa asociada a GitHub. Durante el proceso de verificación, se solicita capturar dos fotografías a través de una webcam: una que muestre un documento acreditativo, como un carné de estudiante, y otra que incluya una matrícula vigente emitida por la institución educativa correspondiente. Aunque el procedimiento puede parecer sencillo, también es necesario justificar la cercanía geográfica al centro educativo o explicar las razones por las cuales este requisito no se cumple. En el caso de programas de enseñanza online, es posible indicar que, debido a la naturaleza del programa, no es obligatorio residir cerca del centro. Para ello, además de las fotografías requeridas, se puede incluir un archivo PDF que acredite que el grupo de matriculación pertenece a un programa de educación online.

Además de las extensiones necesarias, como *Dart Language* y *Flutter Support*, y de la opcional *GitHub Copilot*, se pueden instalar otras extensiones en Visual Studio Code que dependen de las preferencias y estilo de trabajo de cada programador. A continuación, se presenta una lista de extensiones recomendadas que pueden facilitar y optimizar el desarrollo del proyecto:

- **Activitus Bar:** Simplifica la navegación entre vistas y herramientas, permitiendo cambiar rápidamente entre el explorador de archivos, la vista de control de versiones y otros paneles de trabajo.
- **Error Lens:** Muestra los errores y advertencias directamente en el editor, junto al código relevante, para que sean fáciles de identificar y resolver sin necesidad de abrir la consola de problemas.

- **Paste JSON as Code:** Permite pegar estructuras JSON directamente en el editor y convertirlas automáticamente en modelos o clases de código en lenguaje Dart u otros lenguajes.
- **Better Comments:** Mejora la legibilidad de los comentarios en el código al aplicar formatos visuales diferenciados, como colores y estilos, para tareas pendientes, advertencias, preguntas o anotaciones importantes.
- **Pubspec Assist:** Facilita la edición del archivo `pubspec.yaml` para agregar dependencias de Flutter o Dart sin necesidad de escribirlas manualmente, buscando automáticamente las versiones disponibles.
- **Bloc:** Ofrece herramientas específicas para desarrollar aplicaciones basadas en el patrón Bloc, ayudando en la organización del código y el manejo del estado.
- **Awesome Flutter Snippets:** Proporciona fragmentos de código (*snippets*) predefinidos para acelerar la escritura de widgets, estructuras comunes y patrones repetitivos en Flutter.
- **GitGraph:** Muestra una representación gráfica de las ramas y los *commits* en un repositorio Git, facilitando el seguimiento del historial y la gestión de versiones.

Estas extensiones no son obligatorias, pero pueden mejorar significativamente la productividad y la experiencia del programador al trabajar en el proyecto **Eco City Tours**. Su selección dependerá de las necesidades específicas y preferencias de cada desarrollador.

Obtención del código fuente

El código fuente del proyecto está disponible en un repositorio de GitHub. Sigue los pasos a continuación para clonar el repositorio:

1. Asegúrate de tener **Git** instalado en tu sistema. Si no lo tienes, descárgalo e instálalo desde <https://git-scm.com/>.
2. Clona el repositorio utilizando el siguiente comando en tu terminal:

```
git clone https://github.com/fps1001/TFGII_FPisot.git
```

3. Cambia al directorio donde se encuentra la aplicación ejecutando:

```
cd TFGII_FPisot/project-app/project_app
```

D.4. Compilación, instalación y ejecución del proyecto

Esta sección describe los pasos necesarios para compilar, instalar y ejecutar el proyecto **Eco City Tours**. Incluye la configuración de servicios externos como Google y Firebase, la obtención de claves API y la preparación del entorno de desarrollo.

Configuración de Google Cloud Platform

El proyecto utiliza los siguientes servicios de Google:

- **Google Maps SDK for Android:** Para mostrar mapas en la aplicación.
- **Google Places API:** Para obtener información sobre puntos de interés.
- **Google Directions API:** Para calcular rutas.
- **Generative AI (Gemini):** Para funcionalidades avanzadas basadas en modelos de lenguaje.

Pasos para configurar Google Cloud

1. Regístrate en [Google Cloud Platform](#) y crea un nuevo proyecto.
2. Activa las siguientes API desde la consola de Google Cloud:
 - Maps SDK for Android.
 - Places API.
 - Directions API.
 - Generative AI API (Gemini).

3. Genera claves API para cada servicio. Guarda estas claves en un archivo .env ubicado en la raíz del proyecto. El archivo debe tener el siguiente formato:

```
GOOGLE_API_KEY=<TU_CLAVE>
GEMINI_API_KEY=<TU_CLAVE>
GOOGLE_DIRECTIONS_API_KEY=<TU_CLAVE>
GOOGLE_PLACES_API_KEY=<TU_CLAVE>
```

Configuración de Firebase

Firebase se utiliza para almacenamiento en la nube y análisis de errores. Los servicios configurados incluyen:

- **Cloud Firestore:** Base de datos en tiempo real para gestionar datos de la aplicación.
- **Crashlytics:** Herramienta para el análisis y reporte de errores.

Pasos para configurar Firebase

1. Regístrate en [Firebase Console](#) y crea un proyecto con el nombre `eco-city-tour`.
2. Configura **Cloud Firestore**:
 - Accede a la consola de Firebase y habilita Firestore en modo "Producción".
3. Configura **Crashlytics**:
 - Ve a la sección de Crashlytics en Firebase Console y sigue las instrucciones para integrar Crashlytics con tu aplicación.
4. Descarga el archivo `google-services.json` desde la consola de Firebase y colócalo en el directorio `/android/app/` del proyecto.
5. Agrega las siguientes variables al archivo `.env`:

```
FIREBASE_API_KEY=<TU_CLAVE>
FIREBASE_APP_ID=<TU_CLAVE>
FIREBASE_MESSAGING_SENDER_ID=<TU_CLAVE>
```

```
FIREBASE_PROJECT_ID=eco-city-tour
FIREBASE_STORAGE_BUCKET=eco-city-tour.appspot.com
FIREBASE_PACKAGE_NAME=com.example.project_app
FIREBASE_PROJECT_NUMBER=<TU_NUMERO>
MOBILESDK_APP_ID=<TU_CLAVE>
```

Compilación y ejecución del proyecto

Una vez configurados los servicios externos, el proyecto puede compilarse y ejecutarse siguiendo estos pasos:

1. Asegúrate de que el archivo `.env` está en la raíz del proyecto.
2. Instala las dependencias necesarias ejecutando:

```
flutter pub get
```

3. Compila y ejecuta la aplicación en un dispositivo o emulador:

```
flutter run
```

D.5. Pruebas del sistema

El sistema fue sometido a diversas pruebas para garantizar su correcto funcionamiento, su calidad de código y la cobertura de los casos de uso. A continuación, se detallan los enfoques y herramientas utilizados durante el proceso de pruebas.

Control de calidad del código con SonarCloud

Para evaluar la calidad del código, se utilizó **SonarCloud**, una herramienta que permite analizar métricas clave como complejidad, mantenibilidad, duplicación de código y cobertura de tests. El análisis de calidad se automatizó a través del flujo de trabajo definido en el archivo `.github/workflows`, donde se configuraron los siguientes pasos:

- Ejecución de las pruebas automatizadas para garantizar que no existan errores en el código.

- Generación del archivo `lcov.info`, que contiene los datos de cobertura del proyecto.
- Envío automático de la cobertura de código y otros resultados a SonarCloud para su análisis.

Este flujo de trabajo se ejecuta automáticamente desde tres archivos, cada uno realizando una labor concreta: compilación, testing y calidad. Cada vez que se realiza un `push` al repositorio o se abre una `pull request` se valida la calidad del código incrementado, permitiendo un control continuo.

Pruebas funcionales y análisis inicial

Dada la naturaleza del proyecto y el uso de Modelos de Lenguaje de Gran Escala (LLM), inicialmente se realizaron pruebas manuales para evaluar la calidad de los resultados. Estas pruebas se centraron en generar rutas y puntos de interés en la ciudad donde resido, verificando la precisión y relevancia de las recomendaciones generadas.

Pruebas automatizadas

Se implementaron pruebas automatizadas en la carpeta `test/` del proyecto para garantizar que cada componente funcione correctamente. Las pruebas fueron desarrolladas utilizando los siguientes paquetes de **Flutter**:

- `test`: Paquete principal para escribir y ejecutar pruebas unitarias.
- `bloc_test`: Herramienta especializada para probar lógica de negocio implementada con el patrón Bloc.
- `mockito` y `mocktail`: Utilizados para generar *mocks* de servicios y (*blocs*) según las necesidades de los tests.

Cada archivo `.dart` cuenta con un conjunto de tests que validan su comportamiento esperado. Estos incluyen pruebas unitarias para funciones específicas y pruebas de integración para verificar el correcto flujo entre los componentes. En la Figura D.1 se puede ver el resumen que Sonar Cloud realizó a uno de los cambios en el repositorio del proyecto.

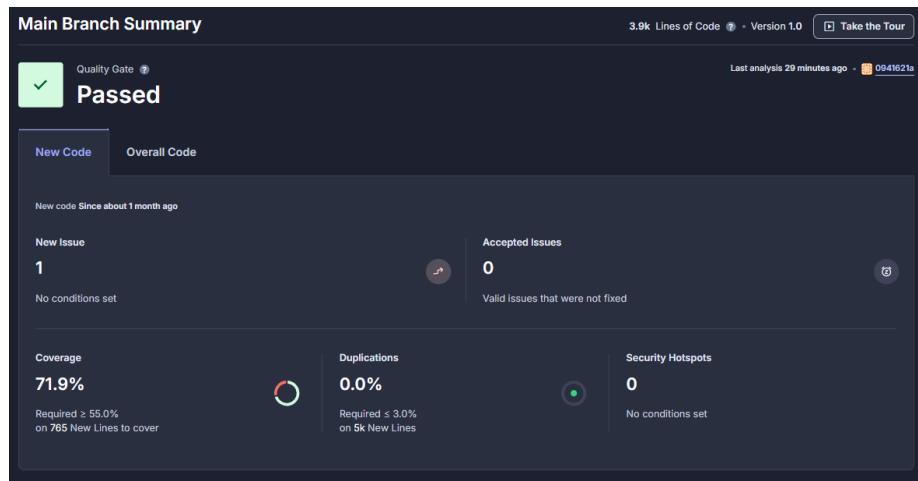


Figura D.1: Resumen de análisis de la rama principal en SonarCloud

Como se puede ver en el resumen se indica la cobertura y aparece una nueva *issue* correspondiente al nuevo código.

Para poder replicar la configuración llevada a cabo en este *Trabajo de Fin de Grado*, el programador debería crear o iniciar cuenta en SonarCloud [8] y crear una nueva organización. A continuación indicar que se quiere crear un nuevo proyecto. Como se puede ver en la Figura D.2 si la cuenta está asociada a Github aparecen automáticamente los repositorios del programador.

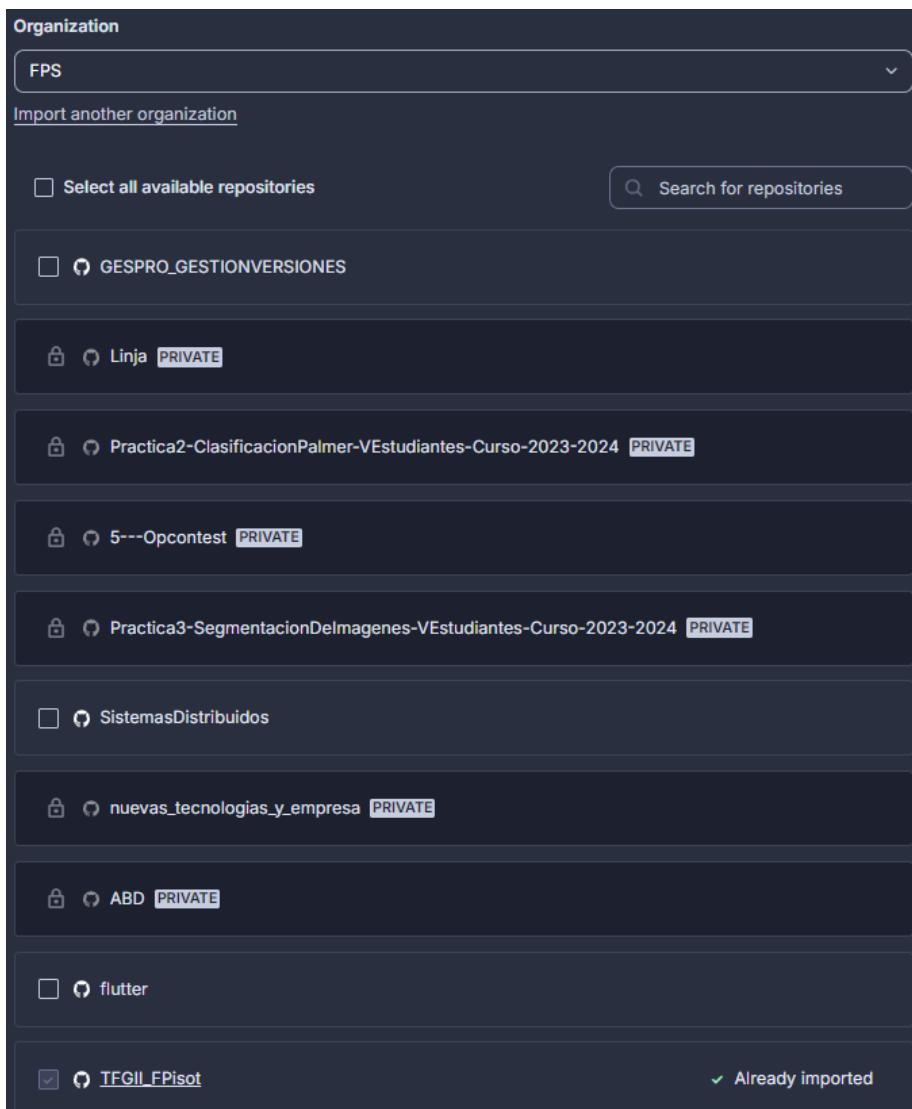


Figura D.2: Creación de nuevo proyecto en SonarCloud

Los proyectos creados en SonarCloud deben ser públicos para cumplir con los criterios de gratuidad. Si se desea analizar proyectos de forma privada, es necesario contratar un plan de pago mensual. Como alternativa, se puede instalar un servidor local de SonarQube, lo que permite realizar análisis directamente en la máquina del desarrollador.

Ejecución de los tests

Para ejecutar las pruebas automatizadas y generar un reporte de cobertura, es necesario ejecutar el siguiente comando en la raíz del proyecto:

```
flutter test --coverage
```

Este comando ejecuta todos los tests definidos y genera un archivo `lcov.info` con los datos de cobertura. Estos datos son posteriormente utilizados por SonarCloud para generar un reporte detallado sobre qué partes del código están cubiertas por los tests y cuáles no.



Figura D.3: Evolución de casos abiertos en SonarCloud

Conclusiones de las pruebas

El sistema fue sometido a un conjunto completo de pruebas funcionales y de calidad, logrando:

- Identificar y corregir errores en fases tempranas del desarrollo.
- Garantizar una alta cobertura del código, reflejada en los reportes de SonarCloud.
- Validar la precisión y relevancia de las recomendaciones generadas por los LLM en contextos reales.

Estas pruebas aseguran que el proyecto **Eco City Tours** cumple con los estándares de calidad necesarios para un entorno de producción.

Apéndice E

Documentación de usuario

E.1. Introducción

Esta sección tiene como objetivo proporcionar a los usuarios las instrucciones necesarias para instalar, configurar y utilizar la aplicación Eco City Tours. Este manual está dirigido a usuarios finales interesados en explorar ciudades de manera sostenible, sin necesidad de conocimientos técnicos avanzados. En primer lugar, se describirán los requisitos mínimos necesarios para ejecutar la aplicación. A continuación, se detallarán los pasos necesarios para instalarla. Finalmente, se presentará una guía práctica para que el usuario pueda aprovechar todas las funcionalidades que ofrece la aplicación.

E.2. Requisitos de usuarios

Para garantizar el correcto funcionamiento de **Eco City Tours**, el dispositivo debe cumplir con los siguientes requisitos:

Requisitos del Sistema Operativo

- **Versión mínima de Android:** 6.0 (API 23, Marshmallow).
- **Versión objetivo de Android:** 14 (API 34, Upside Down Cake).
- La aplicación está optimizada para las características más recientes de Android 14.

Permisos Requeridos

La aplicación solicita ciertos permisos para ofrecer funcionalidades avanzadas, como el acceso a internet y a la ubicación. Sin embargo, los permisos de localización no son necesarios para iniciar la aplicación y se solicitarán al usuario únicamente cuando sean relevantes para la funcionalidad requerida.

Requisitos de Hardware

- Dispositivo con soporte para **GPS** y ubicación.
- Procesador compatible con arquitecturas ARM o x86.
- Acceso a Internet mediante Wi-Fi o red móvil.

E.3. Instalación

Para instalar la aplicación **Eco City Tours** en un dispositivo compatible, siga los siguientes pasos:

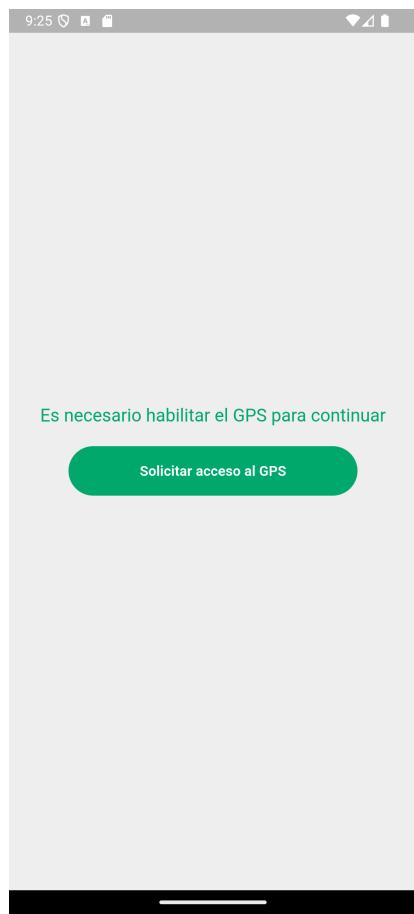
1. Acceda al repositorio oficial del proyecto en GitHub utilizando el siguiente enlace: https://github.com/fps1001/TFGII_FPisot/releases.
2. Busque la **versión más reciente** de las releases disponibles. Asegúrese de seleccionar la versión adecuada para su dispositivo.
3. Descargue el archivo .apk correspondiente a esa versión.
4. Una vez descargado, abra el archivo en su dispositivo móvil para iniciar el proceso de instalación.
5. Si es la primera vez que instala una aplicación fuera de Google Play Store, es posible que deba habilitar la opción de **Permitir aplicaciones de fuentes desconocidas** en los ajustes de seguridad de su dispositivo.
6. Siga las instrucciones que aparecen en pantalla para completar la instalación.

Una vez instalado, la aplicación estará lista para ser configurada y utilizada según las indicaciones del presente manual.

E.4. Manual del Usuario

A continuación, se describen todas las operaciones que puede realizar un usuario de **Eco City Tours**, explicando todas las funcionalidades de la aplicación, así como los pasos necesarios para utilizarlas de manera efectiva.

Habilitar el permiso de uso de GPS



La primera acción que debe realizar el usuario al iniciar la aplicación es habilitar el permiso de uso de GPS. Esto permite que la aplicación acceda a la ubicación del dispositivo para calcular rutas y mostrar información relevante. **Pasos a seguir:**

1. Al abrir la aplicación por primera vez, aparecerá la pantalla mostrada en la Figura E.1.
2. Pulse el botón **Solicitar acceso al GPS**.
3. Conceda el permiso solicitado en la ventana emergente.

La aplicación detecta igualmente el uso de GPS deshabilitado y en cualquier momento si se desactiva o se quitan los permisos volvería a esta pantalla.

Figura E.1: Habilitar el permiso de uso de GPS

Pantalla de configuración de Eco City Tour

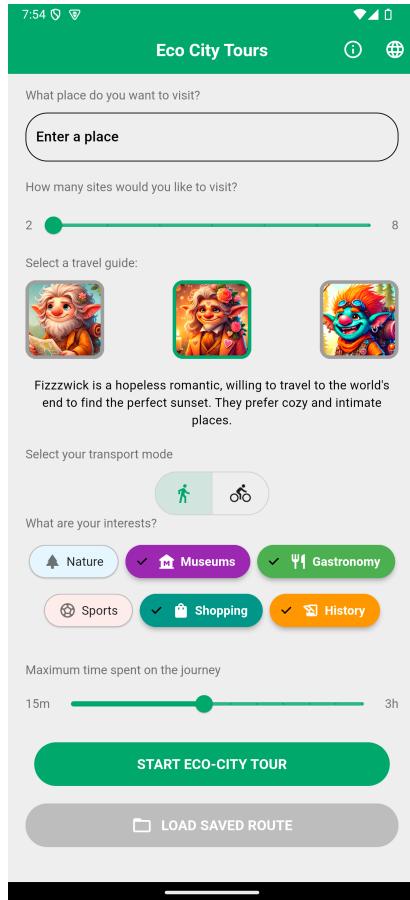


Figura E.2: Pantalla de configuración de Eco City Tour

Una vez establecidos los permisos veremos la pantalla de configuración del Eco City Tour, donde seleccionaremos nuestras preferencias a la hora de diseñar un tour turístico ecológico por un lugar.

Pasos a seguir:

1. En primer lugar introducir la ciudad o sitio que se quiera visitar.
2. Se seleccionará cuantos *Punto de Interés* queremos visitar.
3. Se seleccionará un asistente si lo deseamos entre las tres opciones disponibles.
4. Se indicará si queremos hacer la ruta a pie o en bici.
5. Se indicará nuestras preferencias o gustos a la hora de viajar.
6. Se indicará un tiempo máximo de tiempo en desplazamientos
7. Por último se pulsará el botón de **generar Eco City Tour**

Pantalla de navegación de mapa

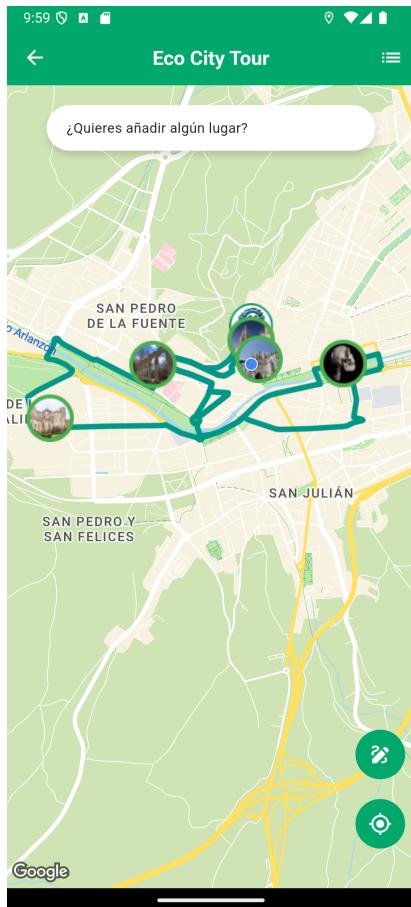


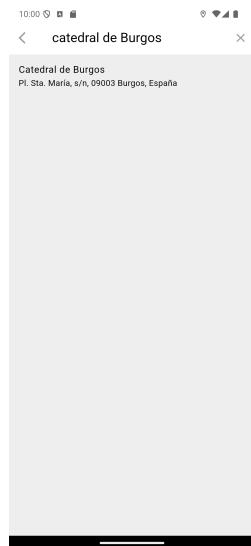
Figura E.3: Pantalla de navegación de mapa

Una vez cargado el tour podremos navegar por el mapa visualizando los distintos PDI que la aplicación ha seleccionado para visitar, así como una ruta optimizada que une los puntos.

Desde aquí hay varias opciones:

- Eliminar aquellos lugares que no queramos visitar.
- Añadir lugares a la ruta. Fig. E.4
- Guardar el Eco City Tour. Fig. E.7
- Unir la posición GPS del usuario a la ruta. Fig. E.9
- Visualizar el seguimiento GPS del usuario. Fig. E.10

Buscar y añadir un *Punto de Interés* al Eco City Tour



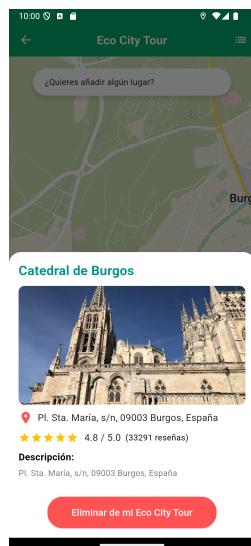
Si el usuario conoce un lugar o desea añadir algún tipo de comercio (hospital, parque, supermercado) a la ruta solo tiene que seguir los pasos desde la Fig. E.4

Pasos a seguir:

1. Introduce el nombre del lugar/lugares que quieras buscar en la barra de búsqueda. Por ejemplo en la Fig. E.4 se busca "Catedral de Burgos", mostrando un solo resultado
2. El usuario pulsará sobre el resultado que desea añadir.
3. El sistema añade el lugar a la ruta y recalcula el camino más corto.

Figura E.4: Búsqueda de PDI

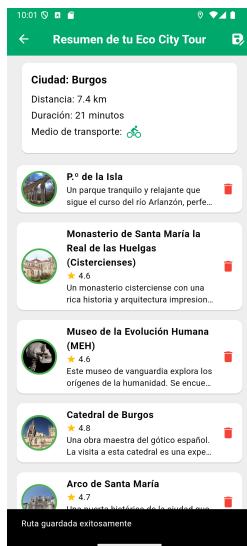
Conocer detalle de PDI



Uno de los elementos básicos de la navegación en el mapa que podemos ver en la Fig. E.4 es la de buscar información de un *Punto de Interés* al seleccionar un elemento se nos muestra una imagen, una descripción, una valoración entre las obtenidas en Google y la opción de eliminar la visita del tour si no es de nuestro agrado.

Figura E.5: Detalle de PDI

Resumen de Eco City Tour creado



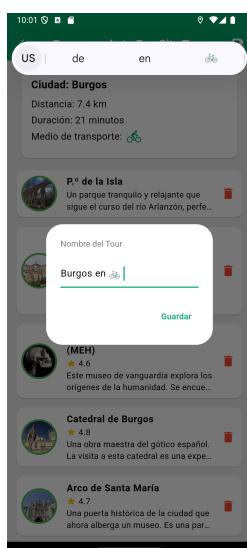
Desde el mapa podemos acceder a esta ventana que nos mostrará en su parte superior un resumen de las características del tour y en la parte inferior cada uno de los *Punto de Interés* que forman el Eco City Tour.

Opciones:

1. Podemos ampliar la información de la descripción.
2. Eliminar del Eco City Tour un PDI
3. Guardar el Eco City Tour actual (Fig. E.7) o volver al mapa (Fig. E.4)

Figura E.6: Pantalla de resumen del Eco City Tour

Guardar Eco City Tour

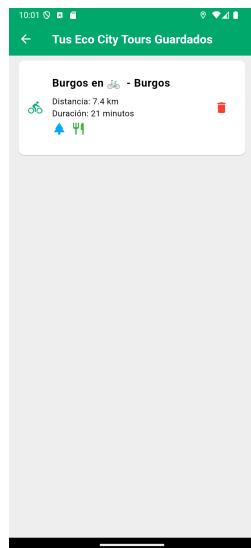


Si queremos guardar la ruta para verla en otra ocasión el usuario puede guardar la información. **Pasos a seguir:**

1. Desde la pantalla de resumen del Eco City Tour, pulsar el botón de guardado de ruta.
2. Elegimos un nombre.
3. Un mensaje en la parte inferior nos informará que el Eco City Tour ha sido guardado con éxito.

Figura E.7: Pantalla de guardado del Eco City Tour

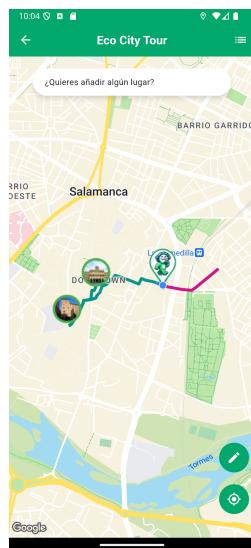
Cargar Eco City Tour



Desde la pantalla de configuración (Figura E.4) podemos ver todos nuestras rutas personales guardadas. Simplemente seleccionaremos sobre el Eco City Tour que queramos cargar sobre el mapa y la pantalla de navegación del mapa (Fig. E.4) se mostrará en el mapa con el Eco City Tour guardado, mostrando la información relevante y todos los *Punto de Interés*.

Figura E.8: Pantalla de carga del Eco City Tour

Unirse al Eco City Tour



Precisamente si el usuario está en un sitio clave que no quiere perder o simplemente quiere que la ubicación forme parte del Eco City Tour puede pulsar el botón inferior de la pantalla del mapa (Fig. E.4) y se añadirá un nuevo **PDI** con la ubicación del usuario y se recalculará de nuevo la ruta más corta entre los puntos que estaban creados y la posición del usuario.

Como si de un *Punto de Interés* corriente se tratase se puede eliminar en cualquier momento.

Figura E.9: Unirse al Eco City Tour

Seguimiento en vivo de la posición del usuario

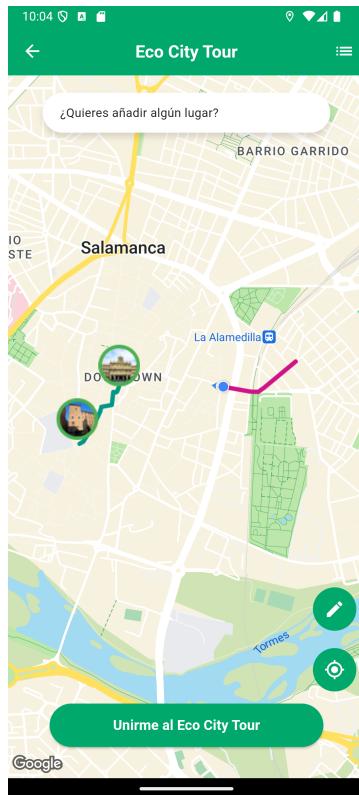


Figura E.10: Seguimiento en vivo de la posición del usuario

Otras opciones que se pueden llevar a cabo desde la pantalla de navegación del mapa (Fig. E.4) las llevamos a cabo con los botones flotantes inferiores, tenemos los dos siguientes:

1. Permite que se muestre sobre pantalla un seguimiento del usuario, el camino que ha seguido se mostrará en el mapa y podrá comprobar su recorrido en todo momento. Para desactivar la línea solo tenemos que pulsar de nuevo el mismo botón.
2. El segundo botón centrará la pantalla del mapa sobre la ubicación actual del dispositivo y permanecerá así hasta que se mueva de nuevo la pantalla.

Estas opciones amplían la funcionalidad de un turista, permitiendo controlar en todo momento su recorrido.

Apéndice F

Anexo de sostenibilización curricular

F.1. F.1. Introducción

Durante el desarrollo del Trabajo Final de Carrera, he utilizado e implementado las disciplinas de la sostenibilidad en el diseño de una aplicación móvil. Ha sido de hecho un gran reto el tratar de generar rutas turísticas de tal manera que promocionen la sostenibilidad. Al final la solución presentada cumple con el *Ciudades y Comunidades Sostenibles (ODS11)* fomentando principalmente la sostenibilidad de ciudades y los asentamientos humanos.

El valor principal de este *Objetivos de Desarrollo Sostenible (ODS)*, como ya se ha tratado, radica en su papel de facilitador en la medida que, al resolver de manera integral y transversal todas las cuestiones relacionadas con la sostenibilidad de las ciudades, tiene un efecto secundario positivo en los demás *ODS*.

F.2. Competencias de Sostenibilidad Adquiridas

A lo largo del trabajo de este proyecto, he seleccionado y elaborado las competencias de sostenibilidad en las siguientes áreas:

Contextualización Crítica del Conocimiento

Puedo contextualizar el conocimiento adquirido y relacionarlo críticamente con los desafíos globales y locales en las áreas sociales, económicas y ambientales tan influenciadas por el turismo sostenible.

Uso Sostenible de Recursos

El desarrollo de la aplicación se enfocó en la utilización sostenible de los recursos. Traté de aplicar la eficiencia en el desarrollo de software al usar tecnologías que consumen la menor energía posible y al priorizar el uso de herramientas open-source, siempre que me fue posible.

Participación en Procesos Comunitarios

El desarrollo de esta aplicación exige una comprensión en profundidad de los procesos comunitarios. Se ha llevado a cabo por ejemplo al decir que no solo los turistas se benefician de su uso, sino que las rutas turísticas también promueven la movilidad sostenible y mejoran el bienestar de la comunidad local.

Principios Éticos y Valores de Sostenibilidad

He incorporado prácticas éticas y valores de sostenibilidad en todo el trabajo descartando por ejemplo aquellas modificaciones que pudieran dar como resultado un conflicto entre la comunidad local y la turística.

F.3. Aplicación de Competencias en el Proyecto

Se utilizaron las siguientes competencias aprendidas en el desarrollo del proyecto:

Diseño y Funcionalidad de la Aplicación

El diseño de la aplicación ha seguido un enfoque centrado en el usuario cuyo propósito era diseñar una solución intuitiva, fácil de usar que promueva la movilidad sostenible: con ciclistas o peatones que disfruten de un viaje personalizado, uniendo los *Punto de Interés (PDI)* de una manera eficiente usando planificadores de rutas optimizados a tal efecto.

Impacto Social y Ambiental

La realización del proyecto busca no solo ser ventajosa para los turistas, sino que también reduce las emisiones de carbono al tiempo que permite a los miembros de la comunidad contribuir a la sostenibilidad de dicha ciudad en su conjunto.

Educación y Conciencia ambiental

El proyecto informará a los ciudadanos no solo con datos de rutas turísticas específicas, sino también buscará sensibilizar acerca de la movilidad sostenible y los beneficios ambientales que provienen de los dilemas referentes al transporte en los que participan.

F.4. Conclusión

Este Trabajo ha sido una experiencia informativa y esclarecedora para mí, tal y como indica en el artículo [7], trabajar en el desarrollo de una aplicación que pone en práctica soluciones que favorecen los ODS te hace más consciente de los múltiples factores que impiden su cumplimiento. Este tiempo me ha dotado del enfoque que se necesita para enfrentar, de manera consciente y sostenible, los desafíos actuales y futuros con una visión global que trabaje activamente hacia un desarrollo sostenible y me siento capaz de liderar nuevas soluciones tecnológicas que potencien los ODS.

Siglas

BLoC *Business Logic Component.* [IV](#), [57](#), [59](#), [60](#), [61](#), [62](#), [70](#)

IA *Inteligencia Artificial.* [15](#), [39](#), [57](#)

LLM *Large Language Models.* [70](#)

ODS *Objetivos de Desarrollo Sostenible.* [91](#)

ODS11 *Ciudades y Comunidades Sostenibles.* [91](#)

PDI *Punto de Interés.* [III](#), [IV](#), [V](#), [5](#), [12](#), [29](#), [33](#), [34](#), [35](#), [39](#), [40](#), [41](#), [43](#), [44](#), [45](#),
[48](#), [50](#), [54](#), [55](#), [58](#), [63](#), [64](#), [65](#), [66](#), [84](#), [85](#), [86](#), [87](#), [88](#), [92](#)

TFG *Trabajo de Fin de Grado.* [5](#), [9](#), [17](#), [78](#)

Glosario

Eco City Tour es una ruta turística compuesta por Puntos de Interés Turístico (PDI). **40**

Bibliografía

- [1] BOE. Xviii convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública, 2023. [Internet; consultado 15-noviembre-2024].
- [2] Flutter Community. flutter Bloc Package, 2024. Accessed: 2024-12-02.
- [3] Flutter Team. Install - flutter documentation, 2024. [Internet; consultado 16-noviembre-2024].
- [4] E. Gamma. *Patrones de diseño: elementos de software orientado a objetos reutilizable*. Addison-Wesley professional computing series. Pearson Educación, 2002.
- [5] Google Cloud. Google cloud terms of service, 2024. [Internet; consultado 15-noviembre-2024].
- [6] Google Play Console. Comisiones de la play store para transacciones y compras, 2024. [Internet; consultado 15-noviembre-2024].
- [7] Urtzi Markiegi, Iñigo Aldalur, and Alain Perez. Integrando los ODS en el grado de Ingeniería Informática. 8.
- [8] SonarSource. Sonarcloud - continuous code quality - code security, 2024. Último acceso: 18 diciembre 2024.