Escuela
Politécnica
Superior

TouristApp

Planificador automático de viajes turísticos

Trabajo Fin de Grado

Universitat d'Alacant
Universidad de Alicante

Autor:
Pablo García Carbonell

Tutor/es:
Elena Lloret

Resumen

Viajar es una de las mayores aficiones a lo largo del mundo. A cualquier persona que preguntes si le gusta viajar o no, posiblemente siempre te responda de la misma manera: sí. El humano es un ser social al que visitar nuevas culturas y conocer gente nueva con ideologías distintas le gusta. Sin embargo, a la hora de visitar una ciudad, ¿qué lugares de esa ciudad puede interesar realmente al usuario? ¿Cómo puedo saber si un lugar está adaptado a mis gustos y características?

Actualmente existen infinidades de aplicaciones tanto *web* como para móvil dónde puedes buscar sitios de interés de una ciudad. Sin embargo, estos resultados suelen ser muy imprecisos y extensos.

De esta idea nace TouristApp, una aplicación *web* cuyo fin es encontrar cuáles pueden ser los lugares de interés de un usuario dependiendo de su rango de edad y del objetivo por el que viaja. Dando tus datos de entrada, te dará como respuesta cuatro lugares de interés dependiendo de estos datos. Por tanto, nuestro objetivo es dar un resultado concreto y sin mucha cantidad de lugares.

Mediante herramientas de procesamiento de lenguaje natural hemos tanto identificado qué ciudad va a visitar el usuario como creado un pequeño resumen de cada uno de los lugares. Además, haciendo uso de las tan conocidas inteligencias artificiales, hemos realizado este filtro de los lugares que puedan interesar al usuario dependiendo de los valores que nos haya proporcionado.

TouristApp consta con interfaces sencillas de usar e intuitivas para que cualquier persona pueda utilizar nuestro proyecto sin ningún problema.

Como resultado, mostramos esos cuatro lugares de interés acompañados con una pequeña imagen y el resumen creado.

El proyecto se ha desarrollado tanto con Angular la parte de frontend como con *Python* la parte del backend. De este último no se tenía mucho conocimiento, pero se ha terminado el proyecto con bastante facilidad a la hora de usarlo en cualquier implementación.

Motivación, justificación y objetivo general

Viajar es una de las mayores aficiones a lo largo del mundo, a todo el mundo le gusta visitar y conocer nuevos destinos y culturas. Sin embargo, antes de realizar cualquier viaje es necesario investigar sobre el lugar que vamos a visitar, viendo cuáles son las zonas de interés de la ciudad para cada usuario y eso no siempre es divertido. Perdemos mucho tiempo viendo qué puede ser lo más interesante para cada uno, leyendo comentarios y opiniones en páginas como *Tripadvisor*[1] o *Trivago*[2].

Por ello, comencé a investigar un poco sobre el tema de *apps* y *webs* turísticas, llegando a la conclusión de que existen dos opciones a la hora de realizar el estudio previo al viaje: por una parte, contratar una agencia de viajes y que ellos se encarguen de organizarlo por ti; y por otra parte buscar tú por tu propia cuenta esas zonas que te puedan llamar la atención del lugar que vas a visitar. Cada opción tiene sus ventajas e inconvenientes: por ejemplo, al contratar una agencia no pierdes tiempo buscando e investigando, pero tienes que pagar a esta para que haga el trabajo por ti; y la otra forma es gratuita, pero pierdes tiempo indagando sobre el viaje que vas a hacer.

Esta decisión que tiene que hacer el usuario antes de viajar para planificar fue la principal razón que me motivó a elegir este trabajo de final de grado (TFG), queriendo realizar una combinación que recoja las ventajas de cada opción y elimine los inconvenientes, es decir, realizar un servicio gratuito que realice el trabajo de búsqueda por ti. Además, a nivel personal, viajar es uno de mis principales *hobbies*.

Por otra parte, el afán por aprender nuevas tecnologías es otro añadido por el que escogí realizar este trabajo de final de grado. En el caso concreto del elegido, deberé utilizar técnicas de procesamiento de lenguaje natural (PLN), que se suele implementar utilizando el lenguaje de programación Python, el cual no he estudiado en profundidad a lo largo del grado, y por tanto me va a permitir adquirir nuevos conocimientos y destrezas con este lenguaje.

Por tanto, utilizando técnicas de procesamiento de lenguaje natural, el objetivo de este TFG es proponer y desarrollar una aplicación web gratuita que ejerza una búsqueda de lugares de interés adaptado a las características del usuario (edad, objetivo por el que viaja), reduciendo

[1] https://www.tripadvisor.es/
[2] https://www.trivago.es/

así de manera considerable el tiempo que pasaría buscando e investigando sobre la ciudad destino.

Agradecimientos

Tras un largo trabajo a lo largo de la carrera y de este proyecto no sabía con quién empezar. Pero como por algo hay que empezar, comenzaré con lo más importante. Gracias a mi padre y a mi madre por ayudarme, apoyarme y animarme en los momentos malos, tanto académicos como personales. Sin ellos y sin mi familia en general no sería la persona que soy ahora mismo.

También me gustaría agradecer a todo mi grupo de amigos, que siempre confiaron en mí, a mis amigos Iván, Tomás, Tomy, Michi, Javi y Guille por todos vuestros apoyos y ánimos en los momentos duros y por vuestras enhorabuenas en los buenos momentos. No podría haber conseguido tanto sin ellos.

En el ámbito académico, doy gracias por haber encontrado un grupo tan sano como el nuestro. Gracias Antonio, Dani, Miguel, Rubén, Vanesa y Mari Cruz. Gracias por haber hecho la época universitaria tan única como dicen y por haber hecho un tiempo tan malo como el de la cuarentena una época amena y divertida. Gracias a todos ellos hacer trabajos de universidad no era un problema, sino todo lo contrario, un reto a batir entre todos nosotros.

Gracias a mis dos primos, Carlos y Raúl, quiénes puedo decir que son más mis hermanos que mis primos por haberse interesado en todas las prácticas y proyectos que he ido realizando a lo largo de la carrera.

No me puedo olvidar de mi tutora Elena Lloret, quién ha hecho que este trabajo de fin de grado haya sido un poco menos complicado de lo que debería, con sus reuniones semanales y resolviendo todas las dudas con velocidad.

Y, sobre todo, gracias a mí mismo por haber tenido las ganas de hacer todos los trabajos con para alcanzar el mejor resultado posible, por el afán de querer ser un buen ingeniero y por la actitud y responsabilidad que he mostrado a lo largo de la carrera; que siempre que aparecía un percance he intentado resolverlo de la mejor forma posible.

De verdad, gracias a todos, no solo por haberme ayudado en mi época universitaria; sino por haberme ayudado a ser mejor persona.

Citas

"Es duro fracasar, pero es todavía peor no haber intentado nunca triunfar"

Theodore Roosevelt.

"El talento es un recurso finito, mientras que la motivación y el trabajo son ilimitados"

Charles Schwab.

"Si algo se vuelve demasiado complicado, se atasca o no te convence: reinicia"

José Vicente Berná

"El trabajo duro supera al talento cuando el talento no funciona bien"

Kevin Durant.

Aclaraciones

En este apartado se dejará claro aquello que pueda generar dudas:

- Para escribir algo del código se ha utilizado la fuente Courier New, sea en el idioma que sea.
- Los términos *frontend* y *backend* se han mantenido del inglés por su frecuente uso a lo largo de la memoria.
- Todas las figuras que no aparece ninguna fuente en el título significa que son de fuente propia, ya sea por alguna captura de código, por alguna herramienta para hacer algún diagrama o cualquier cosa de ese estilo.

Índice de contenidos

Resumen 1

Motivación, justificación y objetivo general 2

Agradecimientos 4

Citas 5

Aclaraciones 6

Índice de contenidos 7

Índice de figuras 10

Índice de tablas 12

1. Introducción 13

2. Estudio de viabilidad 15

2.1. Análisis DAFO 15

2.2. Análisis de riesgos 18

3. Planificación 20

4. Estado del arte. 21

4.1 Procesamiento del Lenguaje Natural (PLN) 21

4.1.1 Modelos para el PLN 21

4.1.2 Niveles/Componentes del PLN 22

4.1.3 Aplicaciones del PLN 23

4.2 Estudio de la competencia 23

4.2.1. Tripadvisor 23

4.2.2 Triposo 25

4.2.3 FourSquare 25

4.2.4 Agencias de viaje 27

4.2.5 ChatGPT 27

4.2.6 Comparación de la competencia 30
5. Objetivos 31
6. Metodología 33
6.1 Metodología *Scrum* 33
6.2 *Kanban* 34
7. Análisis y especificación 36
7.1 Requisitos funcionales 36
7.2 Requisitos no funcionales 38
8. Diseño 40
8.1. Diseño arquitectura conceptual 40
8.2. Diseño API Rest 41
8.3. Diseño arquitectura tecnológica Front/Backend 43
8.3.1 Frontend 44
8.3.2 Backend 45
8.3.3 Traza frontend-backend 46
8.4. Diseño Interfaces – UI 46
8.4.1 Interfaz inicio 47
8.4.2 Interfaz resultado 47
8.5. Guías de estilos 48
8.5.1 Paleta de colores 48
8.5.2 Tipografía 49
8.6. Diseño de pruebas y validación 49
9. Implementación 50
9.1 Página Inicio 51
9.1.1 Rellenar el formulario 51
9.1.2 Filtro de los lugares de interés 52
9.1.3 Diseño final 55
9.2 Página resultado 55

9.2.1 Creación de los resúmenes 56
9.2.2 Creación de las imágenes 61
9.2.3 Cambio un lugar por otros 64
9.2.4 Diseño final 65
10. Pruebas, validación y resultados 66
10.1 Pruebas backend/frontend 66
10.2 Pruebas con usuarios 67
10.3 Resultados 71
10.4 Costes Temporales 73
10.5 Estado actual de la aplicación 74
11. Conclusiones y trabajo futuro 75
11.1 Conclusiones personales 75
11.2 Trabajo futuro 75
Referencias 77

Índice de figuras

Figura 1. Ingresos por año en España por el turismo. 13
Figura 2. Esquema de un análisis DAFO 15
Figura 3. Página de inicio de Tripadvisor 24
Figura 4. Resultado de búsqueda de Madrid en Tripadvisor 24
Figura 5. Página principal de FourSquare 26
Figura 6. Resultado búsqueda FourSquare 26
Figura 7 Respuesta “lugares de interés para pareja de 50 años en Madrid” por el ChatGPT. 28
Figura 8. Respuesta “lugares de interés para jóvenes de 20 años en Madrid” por el ChatGPT. 28
Figura 9. Resultado a comida para jóvenes de 20 años por ChatGPT. 29
Figura 10. Metodología ágil Scrum. 33
Figura 11. Lienzo Kanban. 34
Figura 12. Kanban del proyecto desde Trello 35
Figura 13. Diagrama de flujo de TouristApp. 40
Figura 14. Entrada de la ruta /api/envio 42
Figura 15. Respuesta de la ruta /api/envio 42
Figura 16. Entrada de la ruta /api/resumen 43
Figura 17. Respuesta de la ruta /api/resumen 43
Figura 18. Arquitectura frontend backend 44
Figura 19. Diseño interfaz inicio 47
Figura 20. Diseño interfaz resultado 48
Figura 21. Paleta de colores 48
Figura 22. Tipografía de TouristApp 49
Figura 23. Flujo entre páginas 50
Figura 24. Métod `get_ciudad()` 52
Figura 25. Conexión con api de IA 52
Figura 26. Respuesta del filtro de lugares 53
Figura 27. Lugares interés de Madrid obtenidas por el api 53
Figura 28. Split de los lugares de interés 54
Figura 29. Resultado final de lugares de interés de Madrid 54
Figura 30. Conexión con página resultado 55
Figura 31. Interfaz final de inicio 55

Figura 32. Método `OnInit()` 56
Figura 33. Crear resumen en frontend 57
Figura 34. Datos de búsqueda Wikipedia 58
Figura 35. Resultados de búsqueda 'El museo del prado' 58
Figura 36. Búsqueda en Wikipedia 59
Figura 37. Diagrama de flujo de la búsqueda 60
Figura 38. Creación del resumen 61
Figura 39. Conexión con el api de Google 62
Figura 40. Valor de ítems en la respuesta del api de Google 63
Figura 41. Adición de imágenes al vector 63
Figura 42. Cambio de lugar 64
Figura 43. Diseño final página resultado 65
Figura 44. Llamada correcta /api/envio 66
Figura 45. Llamada correcta /api/resumen 66
Figura 46. Conexión con la ruta 67
Figura 47. Gráfico pregunta 1 68
Figura 48. Gráfico pregunta 2 69
Figura 49. Gráfico pregunta 3 69
Figura 50. Gráfico pregunta 4 70
Figura 51. Gráfico pregunta 5 70
Figura 52. Gráfico pregunta 6 71
Figura 53. Costes temporales Clockify 74

Índice de tablas

Tabla 1. Análisis DAFO 16
Tabla 2. Análisis de riesgos 18
Tabla 3. Análisis de la competencia 30
Tabla 4. Requisitos funcionales 36
Tabla 5. Requisitos no funcionales 38
Tabla 6. Endpoints del API 41
Tabla 7. Tabla Resultado de objetivos principales 72
Tabla 8. Tabla Resultados de objetivos personales 73

1. Introducción

Viajar es un estilo de vida, conocer otras culturas, otros comportamientos. El ser humano es un ser social y le gusta viajar [1]. Existen otros tipos de viaje, como puede ser viajar por trabajo o viajar para visitar a algún familiar o conocido. Sin embargo, también existen otros tipos de viaje, los que se realizan por diversión, como podrían ser viajes como forma de bienestar, viajes como capricho, viajes en familia o con amigos. Pero todos estos últimos tienen algo en común: viajar por turismo y ocio.

Los últimos años los viajes han sufrido un descenso considerable por una razón obvia, la pandemia causada por el COVID-19 [2], dando como ejemplo España, que se ve cómo ha perdido hasta 50 mil millones de euros en turismo a causa del coronavirus, como se puede ver en la Figura 1.

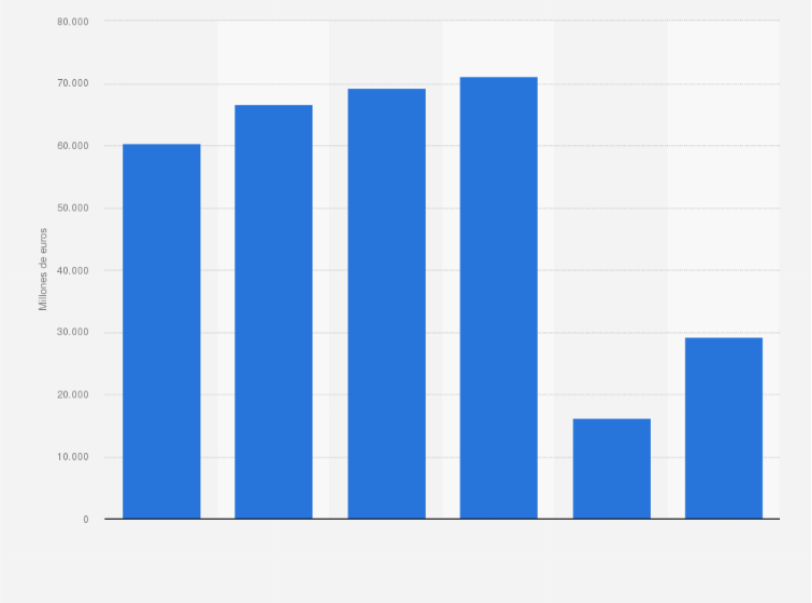


Figura 1. Ingresos por año en España por el turismo.
(en asterisco (*) se marcan los años con COVID)

https://es.statista.com/estadisticas/673456/ingresos-anuales-por-turismo-extranjero-espana/

Sin embargo, actualmente se podría decir que se ha superado la pandemia y convivimos con dicho virus sin problema. Por tanto, hoy en día la mayoría de gente aprovecha cualquier hueco para viajar, normalmente siendo escapadas de un par de días, fin de semana la mayoría de las veces [3].

Los amantes de los viajes, al querer visitar un país y otro sin parar, planifican su viaje rápidamente principalmente de dos formas: la contratación de agencias de viaje como Viajes el

Corte Inglés[3] o ser ellos mismos los que se planifican sus viajes mediante *webs* como *Tripadvisor*[4], cada una de lcon sus ventajas e inconvenientes. Al contratar una agencia no se pierde tiempo investigando, pero es necesario pagar a esta. De la otra forma, se realiza gratuitamente, pero es necesario realizar la búsqueda de forma personal y no por algún agente externo, inviertiendo así bastante tiempo.

Entonces, ¿por qué no crear un servicio que recoja estas ventajas, pero suprima los inconvenientes? Es decir, una página web que realice la búsqueda de los lugares de interés por el usuario y de forma gratuita. Este es el principal objetivo de TouristApp, el trabajo de fin de grado que se recogerá en este documento.

Es posible que exista algún servicio parecido, pero el resultado suele ser muy general. Sin embargo, con esta propuesta se busca un resultado más concreto y orientado al usuario. Por ejemplo, en el caso de una escapada a Barcelona con el objetivo de ocio, no es lo mismo que realice el viaje un matrimonio de 50 años que un grupo de adolescentes, ya que pueden tener diferentes intereses a la hora de visitar la ciudad.

Cogiendo como idea las 5 W del periodismo [4], la búsqueda de los lugares de interés se realizará mediante tres preguntas clave: dónde vas a viajar, quién va a realizar el viaje (que cantidad de gente y rango de edad) y para qué viaja (con que objetivo). Respondiendo estas tres preguntas se podrá dar un resultado muy concreto y preciso para el usuario, resolviendo así el problema comentado anteriormente de tener un resultado muy general.

Por tanto, con TouristApp se busca realizar un algoritmo de búsqueda de lugares turísticas adaptado a las condiciones, gustos y características de cada usuario basado en herramientas de procesamiento de lenguaje natural (PLN).

[3] https://www.viajeselcorteingles.es/
[4] https://www.tripadvisor.es/

2. Estudio de viabilidad

Antes de ponernos manos a la obra con el proyecto se realizará un estudio previo de viabilidad con el fin de tener una mejor perspectiva del proyecto a desarrollar, estudiando y analizando todas sus características y factores externos. Este estudio de factores internos y externos del producto se realizará mediante un análisis DAFO.

Una vez realizado este punto, se realizará un análisis de riesgos para poder ver los percances que pueden surgir a lo largo del desarrollo del TFG, y con ello preparar planes para poder evitar o solucionar todos los problemas que puedan aparecer.

2.1. Análisis DAFO

El análisis DAFO consiste en un esquema de trabajo utilizado para el estudio de un proyecto. Este esquema se divide en dos partes, factores externos e internos, las cuales se dividen en otras dos partes nuevas. La parte de factores externos se divide en **amenazas** y **oportunidades**, y la parte de factores internos en **debilidades** y **fortalezas**. De estas cuatro iniciales es de donde proviene el nombre este esquema: debilidades, amenazas, fortalezas y oportunidades, DAFO.

Figura 2. Esquema de un análisis DAFO
Fuente: https://dircomfidencial.com/diccionario/analisis-dafo-20161113-1643/

Como se observa en el esquema y se ha comentado anteriormente, el análisis DAFO se divide en dos secciones, y éstas, a su vez, en otras dos subsecciones. Por un lado, los factores internos son los que se pueden cambiar por el propio desarrollador del proyecto, mientras que por otro lado, los factores externos son los que afectan al proyecto y no han sido causados por el desarrollado del mismo, y por tanto, no dependen de él.

En la Tabla 1 se muestra el análisis DAFO del proyecto para poder saber cuáles son los factores que mejorar y sobresalir entre las diferentes alternativas existentes en el mercado.

Tabla 1. Análisis DAFO

Factores externos	AMENAZAS - Competencia con otras *apps* más conocidas.	OPORTUNIDADES - Aprender a utilizar herramientas de PLN. - Aprender *Python.* - Conseguir experiencia. - Pocas aplicaciones *web* con el mismo objetivo que el proyecto.
Factores internos	DEBILIDADES - Falta de experiencia en proyectos de tales dimensiones. - Creación y desarrollo de un proyecto en solitario. - Poco conocimiento de Python y PLN.	FORTALEZAS - Ambición tanto a la hora de afrontar el proyecto como de aprender nuevas tecnologías. - Conocimiento en creación de aplicaciones *web*. - A nivel personal, viajar es uno de mis mayores hobbies.

En la Tabla 1 se ha realizado el análisis DAFO del proyecto, y a continuación se explicará de una forma más extendida:

Factores externos: agentes positivos y negativos externos al desarrollador del proyecto

- Amenazas: a la hora de hablar de las amenazas del proyecto no encontramos demasiadas más allá de la más obvia, que es la competencia con otras *apps* más conocidas, ya que existen alternativas a crear rutas turísticas o ver lugares de interés de una ciudad que visites.
- Oportunidades: a nivel personal y profesional, existe la oportunidad de aprender nuevas técnicas como son el uso de herramientas de procesamiento de lenguaje natural (PLN) y el lenguaje de programación de *Python*, con el que se implementarán la mayoría de los algoritmos de PLN. Por la parte más relacionada con el proyecto, la oportunidad de crear una *app web* como ninguna otra, aunque en amenazas se haya comentado que hay alternativas, existe la oportunidad de realizar un proyecto único como ningún otro. Además, todo esto aportará experiencia al desarrollador a la hora de realizar un proyecto de dicha magnitud.

Factores internos: agentes positivos y negativos del desarrollador del proyecto.

- Debilidades: La principal debilidad sería la falta de experiencia a la hora de realizar aplicaciones *web.* A lo largo de los cuatro años del grado se han realizado bastantes proyectos, pero siempre han sido de forma guiada, con pautas y ayudas. Sin embargo, ahora el proyecto se realizará desde cero y prácticamente sin guías ni ayudas. Además, realizar este proyecto de tal envergadura en solitario es otro problema a tener en cuenta, teniendo hacer todas las subtareas del proyecto por mí mismo. Por último, y hablando más de la parte del desarrollo en sí, el poco uso tanto de *Python* como de PLN a lo largo de la carrera es una gran debilidad, ya que tocará aprender un nuevo lenguaje de programación.
- Fortalezas: Ante las debilidades que han aparecido, la principal fortaleza es la ambición y las ganas, tanto de aprender nuevas tecnologías como *Python* o herramientas de PLN, como de sacar adelante un proyecto de tales dimensiones como es el TFG. Viajar es uno de los mayores hobbies a nivel personal, cosa que es un inventivo a la hora de querer afrontar este proyecto. Además, tengo bastante conocimiento previo en la creación de aplicaciones *web*.

2.2. Análisis de riesgos

Los riesgos de un proyecto son los distintos problemas e imprevistos que pueden surgir a lo largo del desarrollo del producto los cuales pueden llegar a parar o frenar el transcurso normal de este.

Por todo esto, se realizará un análisis de riesgos del proyecto con el objetivo de intentar evitarlos, y en el caso de sufrir un problema, poder abordarlos de la manera más eficiente posible y menos perjudicial para el proyecto.

De cada riesgo se estudiarán los siguientes aspectos:

- **Riesgo:** descripción del riesgo que se puede sufrir.
- **Probabilidad:** posibilidad de que ocurra el riesgo. Determinaremos distintos niveles de probabilidad:
    - Improbable
    - Poco probable
    - Probable
    - Muy probable
- **Gravedad:** consecuencias que hay si el riesgo ocurre. Dentro de este aspecto se ha realizado una división para diferenciar la gravedad de cada riesgo:
    - **1:** Muy leve, retraso de un par de días, una semana como mucho**.**
    - **2:** Leve, retraso de semanas.
    - **3:** Grave, retraso de un mes.
    - **4:** Muy grave, tener que empezar el proyecto de nuevo.
- **Plan de contingencia:** método para solucionar un riesgo que ya ha sucedido de la manera más rápida y eficiente posible.
- **Plan de prevención:** método para prevenir que suceda un riesgo.

En la Tabla 2 se muestra el análisis de riesgos del proyecto.

Tabla 2. Análisis de riesgos

Riesgo	Probabilidad	Gravedad	Plan de contingencia	Plan de prevención
Enfermedad del desarrollador	Probable	2	Descansar y seguir las pautas del médico	Intentar cuidar la salud tanto física como mental

Pérdida del código del proyecto	Poco probable	4	Realizar el código de nuevo con todo lo aprendido anteriormente	Copias de seguridad en la nube y subir código en *GitHub*[5]
Avería del ordenador	Probable	3	Arreglarlo de la mejor forma posible mientras se utiliza otro para no perder el hilo de trabajo	Tener todo ordenado y no descargar nada extraño de Internet
Mala planificación del proyecto	Poco probable	3	Retrasar las fechas de las tareas o realizar una nueva planificación con nuevas estrategias	Realizar una buena planificación mediante las estrategias y metodologías correctas. Trabajar de forma eficiente
No terminar a tiempo el proyecto	Poco probable	3	Intentar tener una buena planificación	Organizarse bien y trabajar eficientemente
Pérdida de la memoria del TFG	Poco probable	4	Al igual que con el código, volver a realizarla e intentar que quede lo más similar posible	Realizar guardados automáticos tanto en local como en la nube
Problemas con el lenguaje de Python	Probable	1	Seguir investigando y buscando información sobre el lenguaje y ver similitudes con otros conocidos	Realizar un estudio e informarse sobre el lenguaje para tener una buena base
Problemas con las herramientas de PLN	Probable	1	Investigar e indagar más sobre el tema además de buscar otras alternativas	Realizar un estudio previo sobre PLN y las librerías necesarias para la realización del proyecto
Elección incorrecta de las tecnologías	Poco probable	2	Ver las razones por las qué no ha funcionado una tecnología y ver qué otra se puede usar	Estudiar y documentarse bien sobre las tecnologías que vamos a usar y ver cuáles son las óptimas
Saturación entre el TFG y ABP[6]	Probable	2	Reorganizar de forma correcta para poder compaginar los dos proyectos	Tener una buena organización e intentar solventar el ABP lo antes posible para poder centrar el 100% del trabajo al TFG

[5] https://github.com/
[6] https://eps.ua.es/es/ingenieria-multimedia/gestioncontenidos/que-es-abp.html

3. Planificación

Antes de ponernos manos a la obra con el desarrollo del proyecto, es necesario realizar una buena planificación del proyecto para no sufrir ningún percance. Se han utilizado las siguientes herramientas a lo largo del proyecto hablando de planificación:

- OneDrive[7]: con esta plataforma evitamos cualquier pérdida de la memoria, ya que se encarga de ir subiendo a la nube cada versión de esta.
- GitHub[8]: al igual que con OneDrive buscamos no perder la memoria, con GitHub buscamos no perder el código del proyecto. Hemos creado un repositorio para ir subiendo cada actualización y no tener ningún susto.
- Trello[9]: hemos utilizado Trello para poder organizar de manera correcta las tareas que se van a realizar. Más adelante, en la sección 6 sobre las metodologías se explica más detalladamente
- Clockify[10]: es una aplicación que permite al desarrollador contabilizar el tiempo de cada una de las tareas que realiza. Esto es muy útil para saber si se cumplen los tiempos de forma correcta. Así, en función del tiempo que se trabaje y del que se dispone en el presupuesto inicial, se puede determinar la eficiencia y el estado del proyecto frente a la fecha límite.

Con todas estas tecnologías conseguiremos tanto que no haya ningún percance a la hora de perder código o algún fragmento de memoria como a la hora de organizar las tareas.

A lo largo del proyecto se realizarán reuniones con la tutora cada semana para comprobar que todo está funcionando como se esperaba. El orden para ir avanzando en el TFG será el definido en la memoria. Se irá punto a punto, alternando entre la memoria y la implementación del código.

[7] https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage
[8] https://github.com/
[9] https://trello.com/
[10] https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage

4. Estado del arte.

En este apartado se realizará una investigación sobre el tema que se tratará en el TFG y el entorno al que pertenece el desarrollo de este, es decir, a aplicaciones *webs* destinadas a crear rutas turísticas, las características que podemos encontrar y las tecnologías con la que son desarrolladas. Esto permitirá conocer más sobre el tema y así poder aplicar soluciones correctas en nuestro producto final.

Por una parte, se estudiará e investigará sobre el tema principal a tratar en este proyecto, el procesamiento del lenguaje natural, y por otra parte la generación de rutas turísticas, estudiando otras opciones existentes dentro del mercado.

4.1 Procesamiento del Lenguaje Natural (PLN)

El procesamiento del lenguaje natural [5] es el campo de conocimiento de la Inteligencia Artificial, computación y de la lingüística que estudia el comportamiento entre el humano y la máquina, es decir, estudia y hace posible las interacciones entre ambos. Desde que la máquina fue creada, el humano ha tenido la necesidad de comunicarse con ella. Hoy en día, la relación humano-máquina está aumentando de forma exponencial, no solo en comunicarse con la máquina sino también a través de ella, como ocurre con las redes sociales. Por tanto, el PLN busca permitir, facilitar y suavizar esta comunicación, por lo que este sistema debe conocer el lenguaje, como puede ser las propias palabras, cómo realizar enunciados con sentido, saber el contexto de estos, etc.

4.1.1 Modelos para el PLN

Tratar computacionalmente una lengua del mundo significa un proceso de modelización matemática. Normalmente, los mensajes son codificados en los lenguajes de programación C, Java o Python. Por tanto, hay que realizar una preparación para que el programador lo implemente en un código eficiente y funcional. Existen dos modelos principales:

- **Modelos lógicos** (gramáticas): Los lingüistas escriben reglas de reconocimiento de patrones estructurales, empleando un formalismo gramatical concreto. Estas reglas, en combinación con la información almacenada en diccionarios computacionales, definen los patrones que hay que reconocer para resolver la tarea (buscar información, traducir, etc.).

- **Modelos probabilísticos del lenguaje** (basados en ejemplos): La aproximación es a la inversa: los lingüistas recogen colecciones de ejemplos y datos y a partir de ellos se calculan las frecuencias de diferentes unidades lingüísticas y su probabilidad de aparecer en un contexto determinado. Calculando esta probabilidad, se puede predecir cuál será la siguiente unidad en un contexto dado, sin necesidad de recurrir a reglas gramaticales explícitas. Es el paradigma de "aprendizaje automático" que se ha impuesto en las últimas décadas en Inteligencia Artificial: los algoritmos infieren las posibles respuestas a partir de los datos observados anteriormente.

4.1.2 Niveles/Componentes del PLN

La arquitectura de un sistema de PLN se basa en una definición de lenguaje natural por niveles, los cuales son los siguientes cinco:

- **Nivel fonológico**: trata de cómo las palabras se relacionan con los sonidos que representan.

- **Nivel morfológico**: trata de cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas.

- **Nivel sintáctico**: trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y qué sintagmas son parte de otros sintagmas.

- **Nivel semántico**: trata del significado de las palabras, y de cómo los significados se unen para dar significado a una oración, también se refiere al significado independiente del contexto, es decir, de la oración aislada.

- **Nivel pragmático**: trata de cómo las oraciones se usan en distintas situaciones y de cómo el uso afecta al significado de las oraciones.

Una vez vistos los niveles podemos ver cómo sería el algoritmo que realiza la máquina una vez se le proporcione el mensaje:

1. El usuario expresa qué es lo que desea hacer.

2. La computadora analiza las oraciones proporcionadas, a nivel morfológico y sintáctico.

3. El siguiente paso es analizar las oraciones semánticamente, es decir, saber cuál es el significado de cada oración.

4. Una vez realizado el paso anterior, podemos hacer el análisis pragmático de la instrucción, es decir, una vez analizadas las oraciones, ahora se analizan todas juntas, tomando en cuenta la situación de cada oración. Una vez realizado este paso ya tiene la expresión final.

5. Una vez obtenida la expresión final, el siguiente paso es la ejecución de ésta, para obtener así el resultado y poder proporcionárselo al usuario.

4.1.3 Aplicaciones del PLN

Una vez profundizado en el tema del PLN podemos ver qué tipo de aplicaciones lo utilizan. Las aplicaciones más comunes o conocidas son el análisis y síntesis de voz, la traducción automática, la recuperación de la información, respuesta a preguntas, extracción de la información, reconocimiento del habla o creación de resúmenes, entre otras.

En el caso de *TouristApp* utilizaremos herramientas de extracción de información para saber qué ciudad quiere visitar el usuario, recuperación de información de cada lugar de interés y generación de resúmenes de toda la información recogida anteriormente.

4.2 Estudio de la competencia

Tras un estudio sobre aplicaciones *web* relacionadas con el turismo, se han encontrado algunos resultados cuyo objetivo es parecido al del proyecto, es decir, crear rutas turísticas o aconsejar sobre los sitios de importancia a nivel de turismo de una ciudad. Además de forma adicional, se pedirá al ChatGPT que realice una ruta turística, ya que viendo últimamente la cantidad de usuarios que hacen uso de este, por qué no probarlo con nuestro mismo objetivo.

4.2.1. Tripadvisor

Posiblemente a nivel de consejos a la hora de viajar, Tripadvisor[11] sea la página *web* más conocida por la mayoría de los usuarios. Esta página recoge las reseñas de usuarios de distintos aspectos del turismo, como pueden ser los hoteles, restaurantes, cosas que hacer en el destino que visites. Su página principal es muy sencilla: consta de un simple buscador en el cual puedes poner tu ciudad destino, como se puede ver en la Figura 3.

[11] https://www.tripadvisor.es/

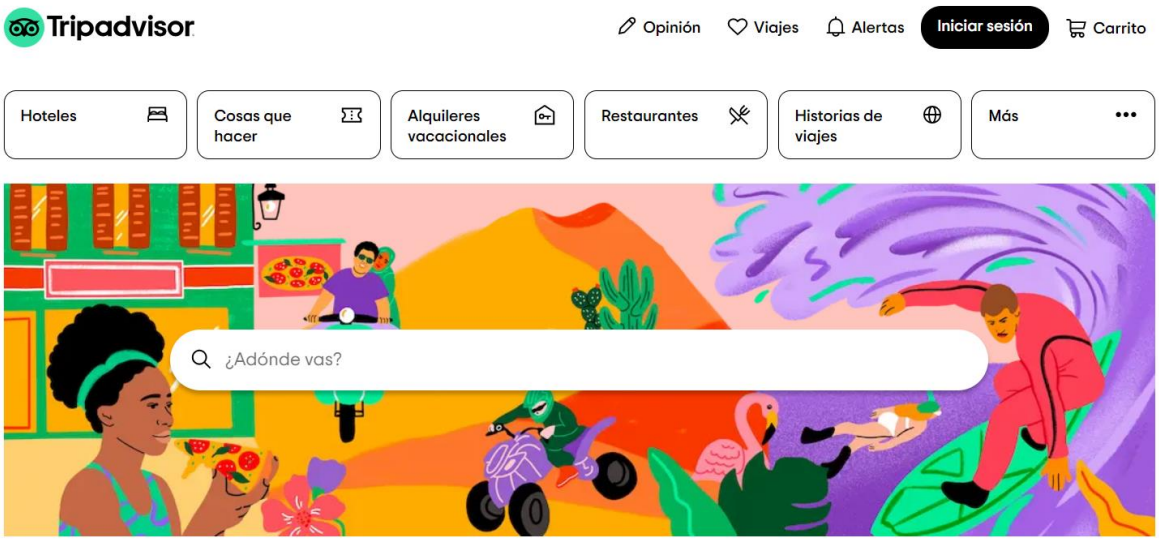


Figura 3. Página de inicio de Tripadvisor.
Fuente: https://www.tripadvsor.es/

Una vez realices una búsqueda, te mostrará una lista sobre las entradas con mejores opiniones de los usuarios, ordenando estas entradas en tres grupos: qué cosas hacer, alójate y come. Además, la propia página proporciona una *review* sobre cada entrada. A continuación, en la Figura 4 se mostrará el resultado al buscar Madrid.

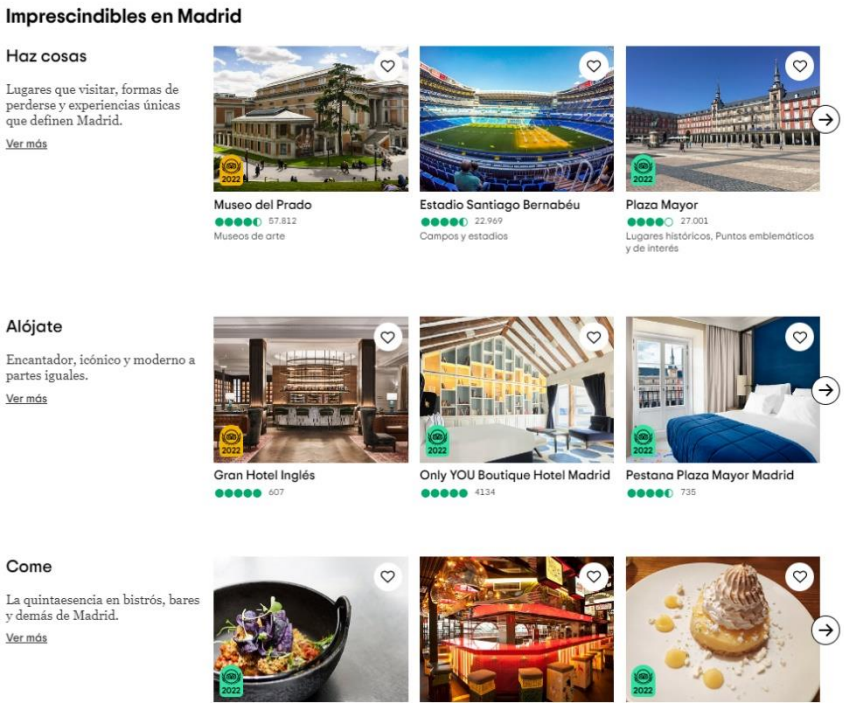


Figura 4. Resultado de búsqueda de Madrid en Tripadvisor
https://www.tripadvisor.es/Tourism-g187514-Madrid-Vacations.html

Esta opción de mercado a parte de recoger valoraciones, comentarios y opiniones de otros usuarios es gratuita y en la mayoría de casos te da la opción de comprar o reservar, ya sea una entrada para algún museo, un hotel, etc. Sin embargo, te da una lista de resultados muy general. Por ejemplo, al buscar Madrid, como se ha hecho anteriormente ha salido una lista de resultados de más de mil opciones en cada una de las secciones, es decir, ha dado como resultado mil entradas sobre restaurantes, mil entradas sobre hoteles y mil entradas sobre lugares de interés. Al ser un resultado tan amplio no se está adaptando realmente a las características del usuario, es decir, Tripadvisor daría el mismo resultado tanto a un hombre de 50 años que a una chica de 19.

4.2.2 Triposo

Triposo es una aplicación para móviles completamente gratuita que aconseja al usuario sobre dónde puede ir de viaje según sus gustos. La *app* conoce los gustos del usuario ya que este se los habrá proporcionado antes o los habrá extraído de Facebook[12], al haber iniciado sesión con él anteriormente. Además, mientras el usuario esté de viaje, esta *app* recomendará lugares de interés al instante. Sin embargo, la aplicación solo está disponible en inglés, lo que dificulta el uso a una cantidad de usuarios, además de presentar el problema de la mayoría de *las webs* de este ámbito: presenta un resultado muy general.

4.2.3 FourSquare

FourSquare [13] es una aplicación tanto web como para móvil que, dada una ciudad y un objetivo, es decir, qué es lo que buscas, te da una lista de resultados. Como se puede ver en la Figura 5, la interfaz de su página de búsqueda es muy sencilla.

[12] https://es-es.facebook.com/
[13] https://es.foursquare.com/city-guide

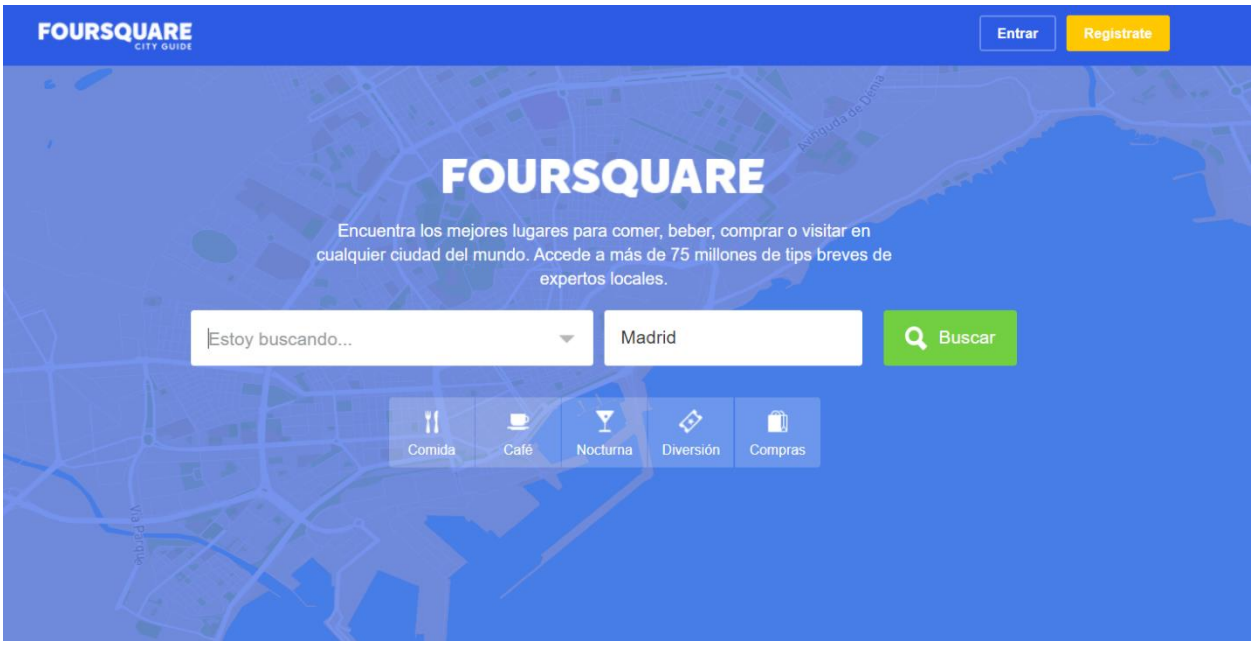


Figura 5. Página principal de FourSquare
https://es.foursquare.com/city-guide

A continuación, realizaremos una búsqueda para comprobar los resultados que proporciona, en este caso buscaremos Madrid, con el objetivo 'Diversión'. Como se puede ver en la Figura 6, el resultado de la búsqueda se muestra de dos formas: una en forma de entrada, a la cual puedes acceder y ver las opiniones y valoraciones del usuario, y otra en forma de ubicación dentro del mapa de la ciudad que has indicado en el buscador

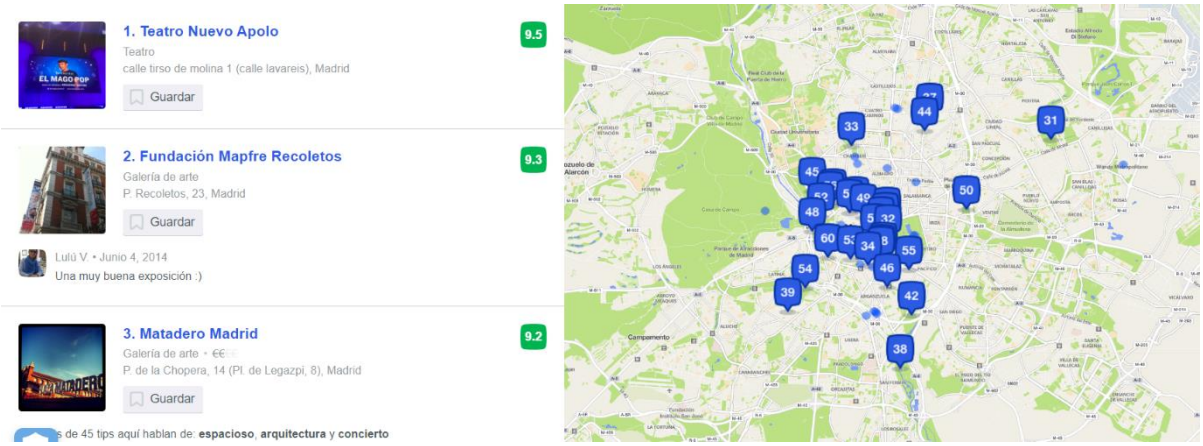


Figura 6. Resultado búsqueda FourSquare
https://es.foursquare.com/explore?mode=url&ne=40.517192%2C-3.526096&q=Diversi%C3%B3n&sw=40.347591%2C-3.813801

Foursquare da un resultado más preciso de lo que se quiere, es decir, se puede indicar si se busca un sitio para comer o un lugar de diversión. Sin embargo, se vuelve al mismo inconveniente que Tripadvisor, y es que el resultado es muy general, es decir, el mismo resultado saldrá tanto para un hombre de 50 años como para una chica de 19.

4.2.4 Agencias de viaje

Otra opción puede ser contratar una agencia de viaje, como puede ser el caso de Viajes el Corte Inglés[14], uno de los más conocidos a nivel nacional. Sin embargo, estas agencias se centran más en las partes de hospedaje y el medio de transporte por el que se viaja, es decir, se encargan de encontrar varias opciones de hotel y de medios de transporte. Una vez encontradas las mejores y dadas las mejores opciones al usuario, este tendrá que elegir cuál es la que más le gusta o la que más le conviene.

Este no es principalmente el objetivo del proyecto. Las agencias se centran en la parte de llegar al lugar y hospedarse. Sin embargo, nosotros buscamos más la parte de turismo, qué zonas le interesan al usuario dependiendo de sus gustos. Además, las agencias no son gratuitos, se debe pagar por sus servicios.

4.2.5 ChatGPT

El ChatGPT es [15] una inteligencia artificial (IA) que en los en los últimos tiempos está de moda, ya que ha sido entrenada para poder tener conversación con cualquier persona. Lo más sorprendente de esta IA concreto es que es capaz de dar unas respuestas muy acertadas y completas, incluso de varios párrafos. Además, en estas respuestas es capaz de expresarte de manera natural y con información muy exacta, lo que hace muy complicado distinguir que el texto ha sido generado por IA o por un humano [6]. ChatGPT nos interesa porque realiza todo este tipo de búsquedas mediante herramientas de procesamiento natural.

Por probar cuál puede ser el resultado, se le pedirá al ChatGPT que nos realice una ruta turística por Madrid, primero para una pareja de 50 años, y después para un grupo de 20 años, y así se estudiarán los dos resultados, viendo si es capaz de adaptarse a las características del usuario.

En la Figura 7 Respuesta “lugares de interés para pareja de 50 años en Madrid” por el ChatGPT. podemos ver el resultado que daría a lugares de interés para el matrimonio de 50 años, y en la Figura 8 los lugares de interés para un grupo de 20 años, ambos viajando a Madrid.

[14] https://www.viajeselcorteingles.es/
[15] https://chat.openai.com/chat

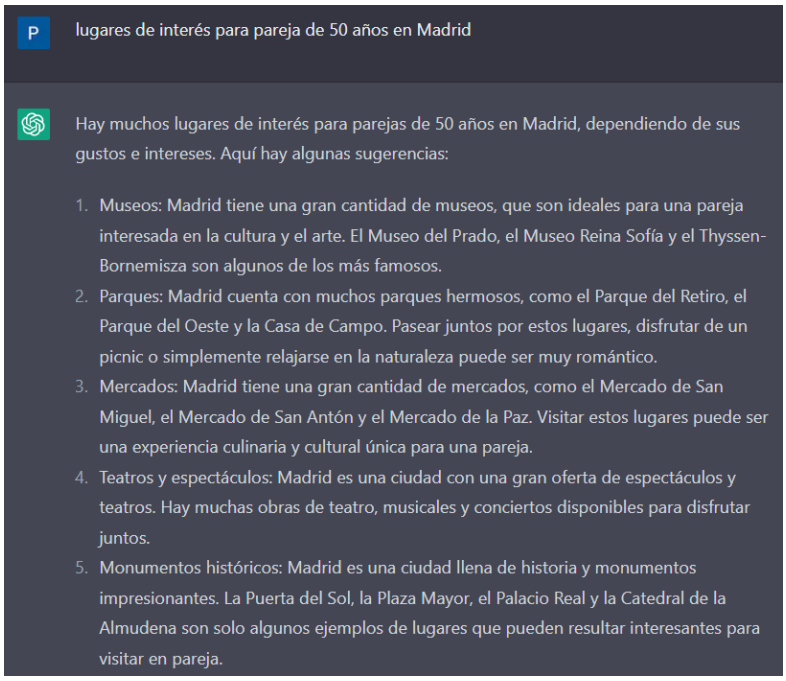


Figura 7 Respuesta "lugares de interés para pareja de 50 años en Madrid" por el ChatGPT.

https://chat.openai.com/chat

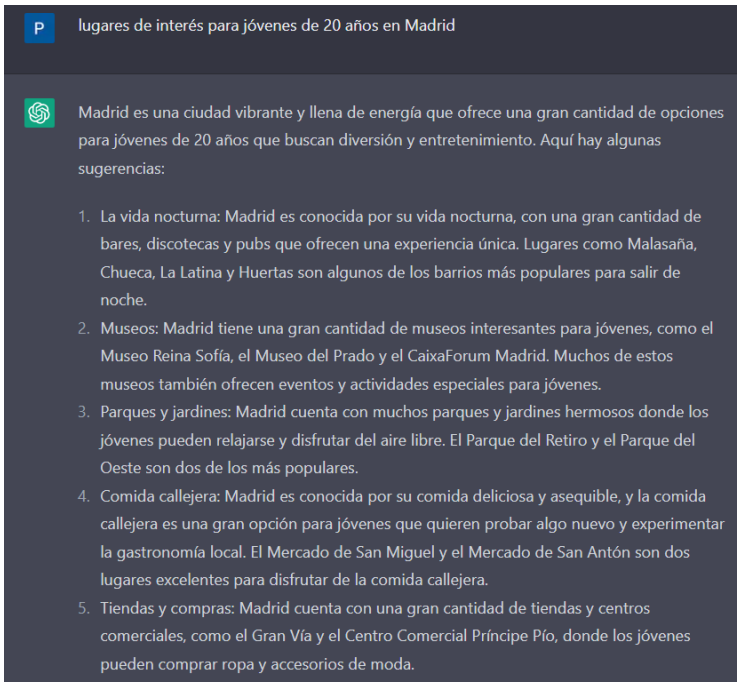


Figura 8. Respuesta "lugares de interés para jóvenes de 20 años en Madrid" por el ChatGPT.https://chat.openai.com/chat

Como se puede observar, el ChatGPT sí realiza una selección de lugares de interés en función de las características del usuario, diferenciando los lugares de ocio entre los dos casos dados. Además, puede llegar a ser más específico, como por ejemplo dando solo restaurantes y bares si se le solicita, como se puede observar en la Figura 9.

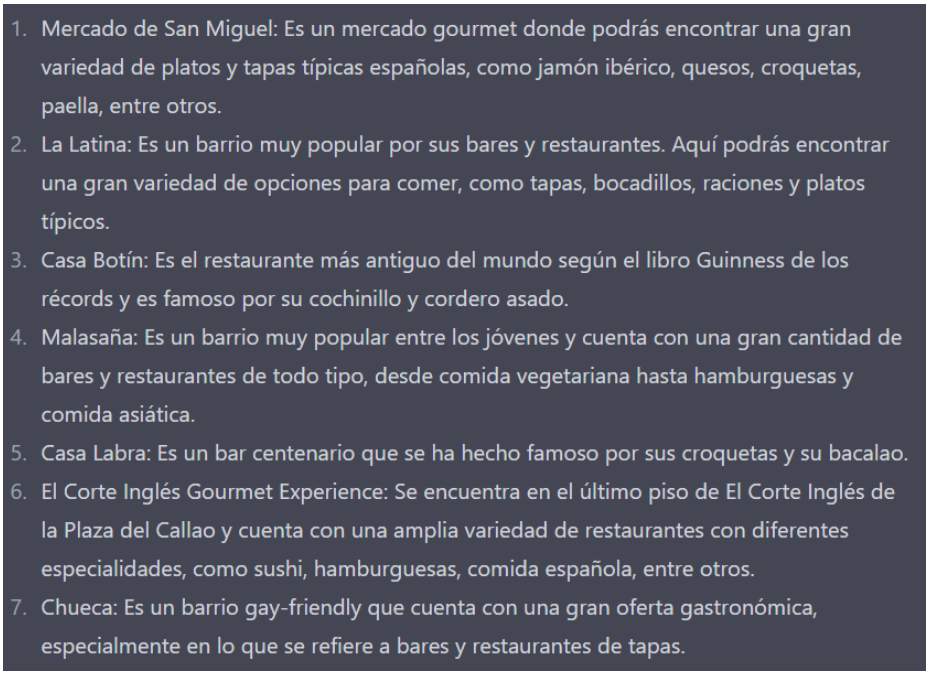


Figura 9. Resultado a comida para jóvenes de 20 años por ChatGPT.

https://chat.openai.com/chat

Por tanto, el ChatGPT sí que da un resultado concreto y adaptado a las características del usuario. Sin embargo, Chatgpt puede presentar algunos problemas relacionados con inventarse la información "alucinación" en el texto generado para cada lugar [7].

En este proyecto se busca un resultado parecido al dado por ChatGPT, pero siendo un servicio solo de turismo y viajes, y no tan general como ChatGPT, que aborda todos los temas posibles. Además, si es posible, se apoyará la respuesta con algún tipo de elemento gráfico, como puede ser imágenes o algún mapa.

4.2.6 Comparación de la competencia

Tabla 3. Análisis de la competencia

	Ventajas	**Inconvenientes**
Tripadvisor	-Recoge opiniones variadas de distintos usuarios sobre cada entrada. - Divide los resultados de búsqueda en 'Come', 'Alójate' y 'Haz cosas'. - Es gratuito. - Interfaz sencilla e intuitiva.	- Resultado de búsqueda muy general y que no está adaptado a las características del usuario. - Lista muy amplia de resultados (mil entradas aproximadamente por búsqueda).
Triposo	- Consejos sobre el viaje adaptados a los gustos y características del usuario. - Es gratuito.	- Solo disponible en inglés. - Resultado de búsqueda muy amplio (hasta 500 entradas por búsqueda).
FourSquare	- Disponible en *app web* y *app* móvil - Interfaz sencilla e intuitiva. - Doble resultado de búsqueda: tanto tipo entrada como ubicación dentro del mapa (apoyo gráfico). - Se puede indicar el objetivo de la búsqueda (comida, ocio, deporte…) - Es gratuito	- Lista muy amplia de resultados (hasta 700 entradas por búsqueda). - Resultado de búsqueda muy general. - Resultado no adaptado al usuario (mismo resultado a un usuario de 50 años que a uno de 20).
Agencias de viaje	- Se centra en los gustos del usuario - No pierdes tiempo previo para estudiar las zonas de interés.	- Es de pago - Se centra mayoritariamente en el hospedaje y en el medio de transporte
ChatGPT	- Es gratuito. - Se centra en los gustos y características del usuario. - No da una lista muy amplia de resultados (entre 5 y 10)	- No es una página centrada en el turismo. - No se acompaña con elementos gráficos como imágenes.

Tras estudiar la competencia llegamos a la conclusión de que la mayoría dan un resultado muy amplio o un resultado no adaptado al usuario. Por tanto, buscaremos realizar un servicio que dé un resultado con cuatro entradas, adaptado a las características dadas por el usuario y, si es posible, acompañado con algún elemento gráfico como imágenes.

5. Objetivos

En este apartado se definirá el objetivo principal a alcanzar con el desarrollo del proyecto y los subobjetivos a desarrollar para llevar a cabo el objetivo principal.

Para ello se hará uso de la estrategia SMART (*Specific, Measurable, Attainable, Realistic and Timely*) [8], con la que podremos basar nuestros objetivos en unas características imprescindibles para dar valor a nuestro proyecto.

El objetivo principal del proyecto es proponer y desarrollar una aplicación *web* accesible y gratuita que proporcione al usuario una lista de los lugares que más le puedan interesar de una ciudad sabiendo las características y gustos de este, como la edad y el objetivo por el que viaja. Cada respuesta se acompañará tanto con elementos gráficos como por un breve resumen. La búsqueda de estos lugares de interés se desarrollará utilizando herramienta de procesamiento de lenguaje natural, así como los resúmenes.

Además, surgen los siguientes subobjetivos:

- Diseñar de una interfaz sencilla e intuitiva.
- Elaborar un estudio de mercado de forma completa analizando las alternativas existentes a nuestro proyecto.
- Analizar y decidir las tecnologías más apropiadas para desarrollar el trabajo.
- Permitir a cualquier usuario acceder a la aplicación.
- Planificar el desarrollo de la aplicación mediante las herramientas de gestión de proyectos correspondiente.
- Evaluar la utilidad y la efectividad de la aplicación desarrollada.
- Aprender nociones básicas sobre el procesamiento del lenguaje natural y de las tecnologías del lenguaje humano, orientadas a la comprensión y generación de información.
- Aprender y utilizar herramientas y librerías específicas de Procesamiento de Lenguaje Natural, así como *frameworks* para diseñar arquitecturas.

A parte de estos subobjetivos a nivel profesional, surgen otros subobjetivos más enfocados a lo personal:

- Aprender un lenguaje de programación nuevo como Python.
- Realizar un documento de las dimensiones de la memoria de un TFG.

- Aprender a realizar completamente una aplicación pasando por todas sus fases: desde investigar a fondo lo que existe ahora en el mercado, pensar qué se necesita –tecnologías, etc. - y cómo añadirlo en el proyecto, diseñar sus interfaces y, desarrollarlo hasta que esté terminada.

6. Metodología

6.1 Metodología *Scrum*

La metodología que se usará a lo largo del proyecto será la metodología ágil de *Scrum*. La metodología Scrum permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Está basada en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes [9].

Además, en la metodología *Scrum* existen tres roles diferentes [10]:

- *Scrum Máster:* gestionar el correcto funcionamiento del proyecto y eliminar impedimentos
- *Product Owner:* encargado de optimizar y maximizar el valor del producto
- Equipo de desarrollo: desarrollar el producto, autoorganizándose y autogestionándose para conseguir entregar un incremento de software al final del ciclo de desarrollo.

En nuestro caso, al ser el desarrollo de un trabajo en individual no habrá reuniones de equipo, sino que serán con la tutora del TFG. En el caso de este proyecto, serán reuniones semanales. Y aunque no nos centremos en la parte de mejorar el trabajo en equipo, sí que usaremos esta metodología por la parte de las iteraciones, con límite de tiempo cada una, de modo de a cada una se le establecen unas tareas y funcionalidades.

En la Figura 10 se muestra una representación gráfica de la metodología *Scrum*.

Figura 10. Metodología ágil Scrum.

https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum

6.2 *Kanban*

El método de *Kanban* es una forma de gestionar los proyectos basados en metodología ágil. *Kanban* es un método *Lean*, muy popular, de gestión del flujo de trabajo para definir, gestionar y mejorar los servicios que proporciona el trabajo de conocimiento. Te ayuda a visualizar el trabajo, maximizar la eficiencia y mejorar continuamente. El trabajo se representa en tableros Kanban, lo que te permite optimizar la entrega de trabajo a través de múltiples equipos y manejar, incluso los proyectos más complejos en un solo entorno [11].

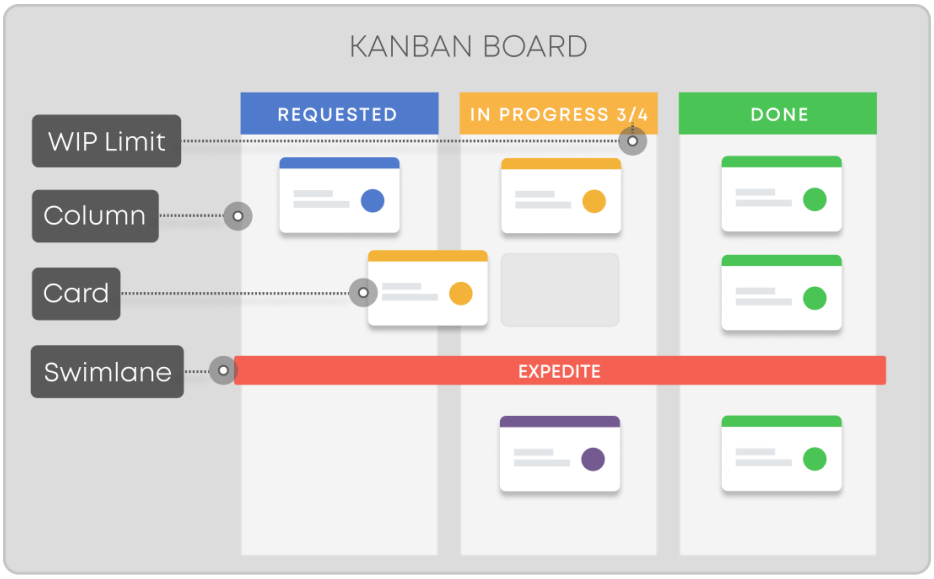


Figura 11. Lienzo Kanban.

https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

El lienzo *Kanban* consta de un tablero dividido en cuatro secciones principalmente:

- Lista de tareas: conjunto de todas las tareas a realizar
- *To do:* conjunto de tareas con más prioridad para hacer
- En proceso: conjunto de tareas que están realizándose en ese momento
- Hecho: conjunto de tareas que ya están acabadas

Para aplicar el método Kanban se ha utilizado la herramienta online de Trello[16], la cual te permite crear un lienzo y poder mover las secciones entre las diferentes columnas de forma sencilla. En la Figura 12 se muestra nuestro *Kanban* desde *Trello*.

[16] https://trello.com/

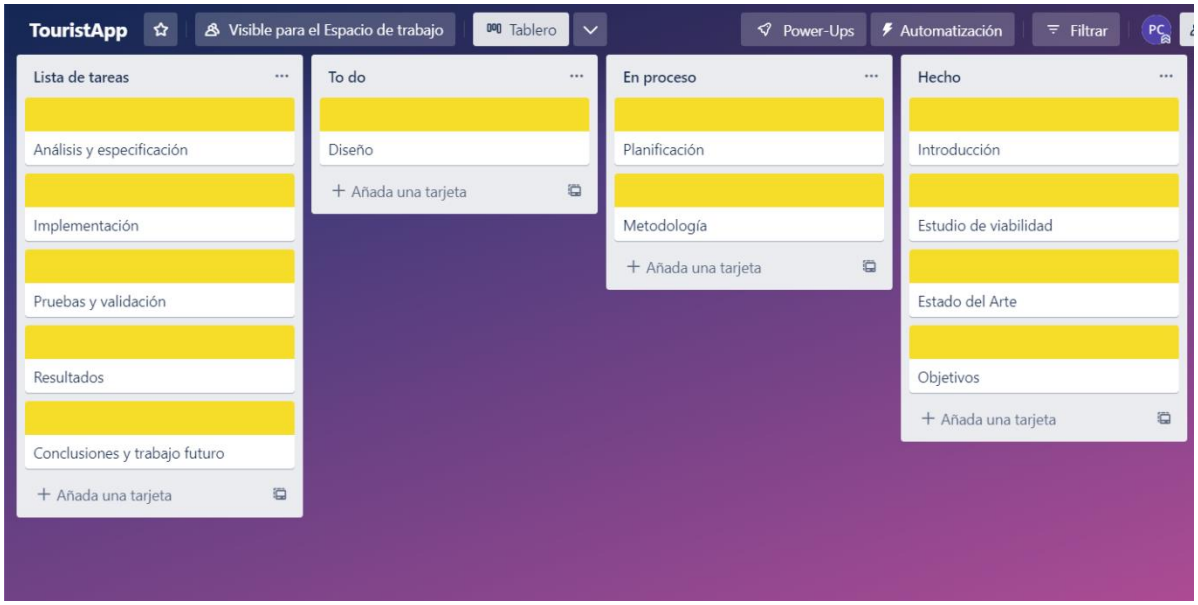


Figura 12. Kanban del proyecto desde Trello.

Como se puede apreciar, he realizado una etiqueta por cada uno de los apartados en los que está dividida la memoria del TFG, asignando una tarea en *Trello* para cada uno de ellos. Además, de forma simultánea se ha realizado el desarrollo del proyecto, haciendo a la vez tanto este documento, es decir, la memoria del TFG, y la implementación del código del trabajo.

7. Análisis y especificación

En la siguiente sección se diseñarán y definirán los requisitos a implementar para conseguir los objetivos del proyecto funcionales. Para realizar esta definición de forma correcta, se utilizará el estándar IEEE 830 [12], estándar que se ha utilizado además en otras asignaturas a lo largo de la carrera.

Este estándar IEEE 830 divide los requisitos en dos tipos, funcionales y no funcionales. Se usará la siguiente nomenclatura: RF – N para los requisitos funcionales y RNF – N para los no funcionales, siendo N el número de requisito. Además del identificador y el nombre del requisito, se definirá la prioridad y una pequeña definición de cada uno.

7.1 Requisitos funcionales

Los requisitos funcionales son definiciones de las funcionalidades que se van a desarrollar a lo largo del proyecto. En otras palabras, muestran qué es lo que se debe hacer para que el proyecto tenga un resultado óptimo y satisfaga las necesidades y expectativas de los usuarios. Por dar un ejemplo, una *web* de una tienda de ropa tiene como un requisito funcional tener sus productos actualizados a la última moda. A continuación, en la Tabla 4. se muestran los requisitos funcionales del proyecto.

Tabla 4. Requisitos funcionales

Nombre	Realizar formulario con la información.
Identificador	RF - 01
Descripción	Realizar formulario donde los usuarios puedan completar toda la información necesaria: dónde van, qué edad tienen, cuándo viajan y con qué objetivo
Prioridad	Prioritario

Nombre	Reconocer la ciudad dada por el usuario
Identificador	RF - 02
Descripción	Diferenciar dentro de la información que ha dado el usuario la ciudad donde quiere viajar

Prioridad	Prioritario

Nombre	Reconocer la edad del usuario
Identificador	RF - 03
Descripción	Diferenciar dentro de la información que ha dado el usuario la edad que tiene
Prioridad	Prioritario

Nombre	Reconocer el objetivo del viaje del usuario
Identificador	RF - 04
Descripción	Diferenciar dentro de la información que ha dado el usuario el objetivo por el que viaja
Prioridad	Prioritario

Nombre	Proporcionar un resultado adaptado
Identificador	RF - 05
Descripción	Con toda la información dada por el usuario, proporcionar un resultado adecuado a las características de cada uno
Prioridad	Prioritario

Nombre	Proporcionar información de cada resultado
Identificador	RF - 06
Descripción	Realizar un pequeño resumen de cada uno de los lugares para acompañar el resultado final
Prioridad	Prioritario

Nombre	Proporcionar imagen de cada resultado
Identificador	RF - 07
Descripción	Acompañar cada resultado con una imagen para dar más información sobre el lugar

Prioridad	Medio

Nombre	Realizar interfaz intuitiva
Identificador	RF - 08
Descripción	Realizar interfaz en la que el usuario pueda desenvolverse sin ningún tipo de duda ni problema
Prioridad	Medio

Nombre	Cambiar un lugar de interés por otro
Identificador	RF - 9
Descripción	El usuario podrá cambiar un lugar de interés por otro.
Prioridad	Medio

7.2 Requisitos no funcionales

Los requisitos no funcionales son aspectos del proyecto ajenos a las funcionalidades, como la accesibilidad, escalabilidad o usabilidad entre otros, es decir, aquellos requisitos que definen características del funcionamiento. Por tanto, el principal objetivo de los requisitos no funcionales es satisfacer las necesidades del cliente en aspectos de experiencia de usuario, sin incumplir ninguna restricción[17]. A continuación, en la Tabla 5 se muestran los requisitos no funcionales del proyecto.

Tabla 5. Requisitos no funcionales

Nombre	Usabilidad
Identificador	RNF - 01
Descripción	La aplicación debe ser fácil de usar y de sencilla comprensión para el usuario. Realizando interfaces sencillas se conseguirá

[17] https://visuresolutions.com/es/blog/non-functional-requirements/#:~:text=Los%20requisitos%20no%20funcionales%20(NFR,la%20confiabilidad%20y%20muchos%20m%C3%A1s.

Nombre	Accesibilidad
Identificador	RNF - 02
Descripción	La aplicación debe permitir a cualquier usuario hacer uso de esta misma

Nombre	Disponibilidad
Identificador	RNF - 03
Descripción	La aplicación debe estar disponible para cualquier tipo de usuario en cualquier momento

Nombre	Mantenibilidad
Identificador	RNF - 04
Descripción	El código se programará de manera limpia para conseguir que sea reutilizable

Nombre	Fiabilidad
Identificador	RNF - 05
Descripción	La aplicación estará preparada para manejar todas las acciones del usuario

8. Diseño

El siguiente apartado es el más importante dentro del TFG, ya que es el diseño del proyecto. Un buen diseño es una de las claves para después saber qué debemos implementar

8.1. Diseño arquitectura conceptual

La aplicación se ha dividido en tres partes fundamentales, por tanto, diremos que la arquitectura conceptual del proyecto es el conjunto de estos tres procedimientos.

Por una parte, tenemos la recogida de la información que nos da el usuario en el formulario, la cual usaremos para diferenciar sobre qué ciudad vamos a buscar información y las características del viaje, en este caso la edad del usuario y el objetivo por el que viaja. Por otra parte, tenemos la búsqueda y filtro de los lugares que pueden interesar a cada usuario. Y, por último, la creación de un resumen con la información de los lugares que hemos determinado en el paso anterior, para finalmente mostrárselo al usuario junto con una imagen. Para que quede claro de manera más gráfica, en la Figura 13 se muestra un diagrama de flujo de la aplicación.

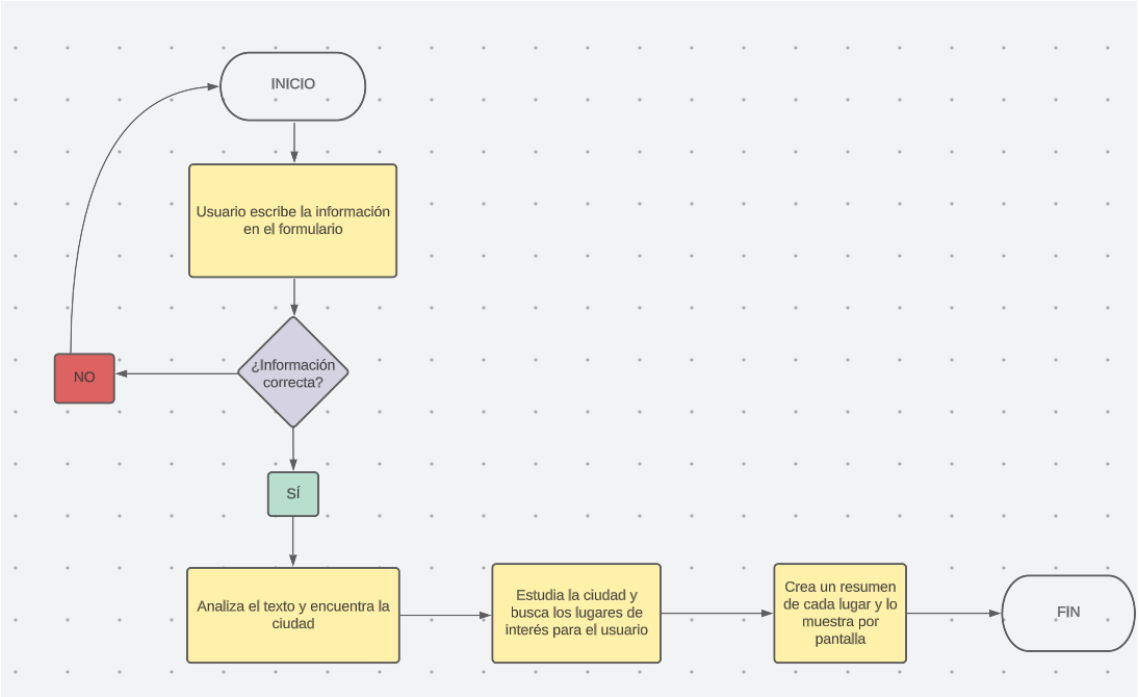


Figura 13. Diagrama de flujo de TouristApp.

Toda la implementación y explicación de cada una de las fases del esquema de lujo se comentará en el punto 9 de la memoria relacionado con la implementación del proyecto.

8.2. Diseño API Rest

API es el acrónimo de interfaz de programación de aplicaciones (*application programming interface* en inglés). Es un conjunto de reglas bien definidas que se utilizan para especificar formalmente la comunicación entre dos componentes de software [13].

Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la solicitud) y el que requiere el productor (la respuesta).

Por ejemplo, el diseño de una API de servicio meteorológico podría requerir que el usuario escribiera un código postal y que el productor diera una respuesta en dos partes: la primera sería la temperatura máxima y la segunda, la mínima.

En otras palabras, las API le permiten interactuar con una computadora o un sistema para obtener datos o ejecutar una función, de manera que el sistema comprenda la solicitud y la cumpla [14].

Para poder realizar el proyecto de manera correcta ha sido necesario realizar una pequeña API. Gracias a la herramienta de Postman[18] ha sido fácil poder estudiar los endpoints, tanto la entrada del *body* como la respuesta de cada ruta.

Tabla 6. Endpoints del API

TIPO	ENDPOINT	BODY	RESPUESTA
POST	/api/envio	JSON que recoge del *input* del frontend	*Url* del lugar que corresponda
POST	/api/resumen	JSON que envía el lugar de interés que se va a buscar en Wikipedia junto con el nombre de la ciudad	Resumen de la página con el algoritmo contado anteriormente junto con el lugar de interés

[18] https://www.postman.com/

Como se puede ver hemos creado dos rutas para el desarrollo del proyecto. La primera se ha utilizado básicamente para enviar el texto del usuario desde el frontend al backend, analizarlo para diferenciar la ciudad y enviar en formato *JSON* la respuesta con todos los lugares de interés ya filtrados para cada usuario. La segunda se ha utilizado para enviar los lugares de interés y enviar el resumen de cada uno de ellos.

En las Figura 15 y Figura 14 se muestra de manera más clara tanto la entrada como la respuesta de la llamada *POST* a /api/envio. Vamos a dar como ejemplo que el usuario escribiese “Me voy a ir a Madrid”. En este *endpoint* hemos llamado a una IA para que realice el filtro de lugares de interés, proceso que se explicará más adelante.

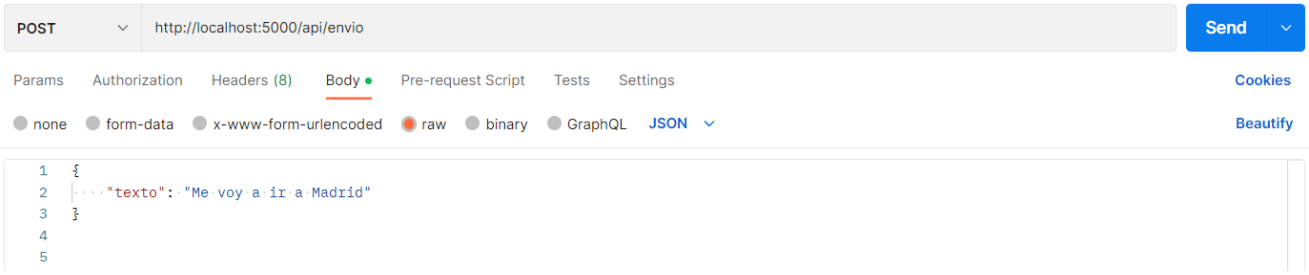


Figura 14. Entrada de la ruta /api/envio

Figura 15. Respuesta de la ruta /api/envio

Como se ve nos devuelve bastantes valores, pero para el proyecto solo serán útiles el valor de *message['content']* el cual tiene los lugares de interés, y los valores de ciudad, edad y objetivo, que se envían para tener esa información tanto en el backend como en el frontend.

En las Figura 16 y Figura 17 se muestra más claramente tanto la solicitud como la salida de la llamada *POST* a /api/resumen. Vamos a poner como ejemplo de lugar de interés 'Museo del Prado'.

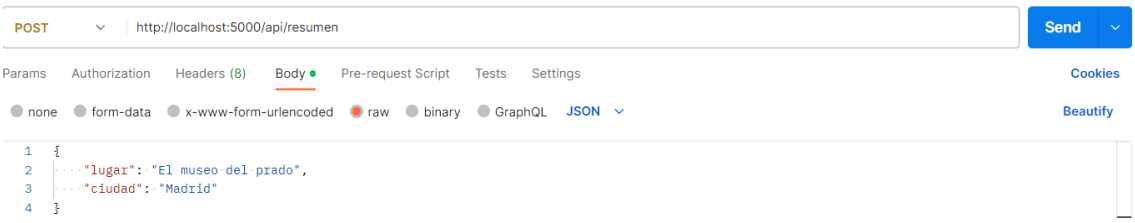


Figura 16. Entrada de la ruta /api/resumen

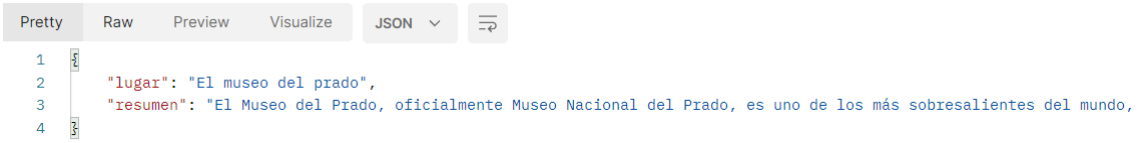


Figura 17. Respuesta de la ruta /api/resumen

Como se puede ver, hemos añadido el valor de la ciudad en la entrada y el valor del lugar a la salida. Más adelante se explicará el por qué.

8.3. Diseño arquitectura tecnológica Front/Backend

En el siguiente apartado se explicará detalladamente la arquitectura tecnológica que se utilizará para realizar nuestra aplicación *web*, explicando de forma concisa cada uno de los componentes que la conforman.

En este caso, la arquitectura escogida ha sido Frontend-Backend, ya que necesitaremos una parte de interfaz gráfica donde se recojan los datos que sean necesarios, y por otra parte una lógica de negocio que realice todos los métodos e implementaciones utilizando la información

recogida en la interfaz para dar el resultado óptimo. Por esta principal razón se utilizará esta arquitectura, siendo el frontend la parte gráfica y el backend la lógica, teniendo que realizar una conexión entre ambas partes.

En la Figura 18 se muestra de manera gráfica como es una estructura básica de frontend-backend.

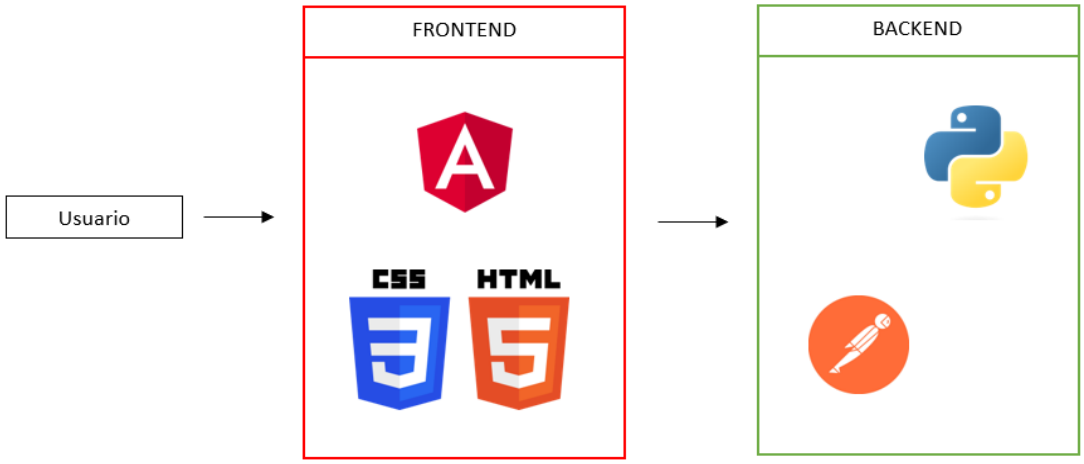


Figura 18. Arquitectura frontend backend

8.3.1 Frontend

El frontend sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, secciones, entre otros. Su uso es determinante para que el usuario tenga una buena experiencia dentro del sitio o aplicación [15]. Por tanto, el frontend tiene una conexión fuerte con la experiencia de usuario y la interfaz de usuario (UX y UI).

En el caso de nuestro proyecto, para realizar el frontend se utilizará Angular[19] , un *framework* para *apps web* desarrollado en TypeScript de código abierto que se utiliza. Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el tratamiento de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar).

Cada componente está formado por tres ficheros fundamentales:

- **Componente.html**: este fichero es donde se estructura y se le da forma a la página.

[19] https://angular.io/

- **Componente.css**: este archivo es donde se le da estilo al fichero componente.html.
- **Componente.ts**: este fichero se encarga de las distintas funcionalidades de las interfaces, como puede ser recoger información de un formulario y enviarla al servicio que corresponda.

Además, en los proyectos basados en Angular es necesario crear un nuevo fichero, normalmente conocido como servicio, el cual realiza la conexión entre el frontend y el backend.

Por dar un ejemplo más aplicado a nuestro proyecto, tenemos una interfaz llamada inicio.html donde hay un formulario, el cual le hemos dado estilo en inicio.css. Una vez se envíe la información, inicio.ts la recibe y la envía al servicio, enviar.service.ts, que hará una llamada a nuestra API, y por tanto, conectará con nuestro backend.

8.3.2 Backend

El backend son todos los códigos ocultos que sirven para que una página web o aplicación funcione correctamente. Además, de su estructura y organización depende la experiencia de usuario. De igual forma, el backend se encarga de optimizar otros elementos y recursos como la seguridad y privacidad en un sitio web o aplicación [15]. En nuestro caso, al no haber login ni nada del estilo no nos centraremos en esta segunda parte, sino que usaremos el backend para la lógica de negocio.

El backend se ha implementado en el lenguaje de *Python*, que como se comentó anteriormente, es el más apropiado para utilizar herramientas de PLN. Por una parte, se analiza el texto enviado desde el frontend gracias a la librería *Spacy*[20], la cual tiene un módulo de detección de entidades nombradas que nos permiten extraer distintos tipos de entidades que hay en un texto, como lugares, personas, fechas, empresas u organizaciones. De todas estas entidades solo nos incumbe la entidad de lugar, la cual es marcada mediante la etiqueta de *LOC*.

Además de *Spacy*, utilizada para la detección de entidades, se han utilizado las librerías de *Sumy*[21] *y Wikipedia-api*, también de *Python.* La primera librería nos ayudará a hacer los resúmenes. Existen distintos tipos de algoritmos, y el que se ha elegido ha sido *LexRankSummarizer.* La segunda, para extraer información de la página de Wikipedia[22], una fuente fiable de información sobre lugares del mundo.

[20] https://spacy.io/
[21] https://pypi.org/project/sumy/
[22] https://es.wikipedia.org/wiki/Wikipedia:Portada

8.3.3 Traza frontend-backend

Para hacer un resumen de cómo funciona esta conexión se va a hacer una pequeña traza con todos los pasos que se van realizando

1. El fichero del frontend *inicio.component.ts* recibe la información dada en el formulario creado en inicio.component.html
2. *Inicio.component.ts* envía la información al servicio creado de nombre *envio.service.ts*
3. *Envio.service.ts* hace una llamada a la *url http://localhost:5000/api/envio,* conectando así con el backend y pasando por parámetro la información recibida del formulario.
4. Ese *endpoint* analiza el texto que ha recibido, diferencia qué palabras representan una ciudad y devuelve una respuesta en formato *JSON* con los lugares de interés de la ciudad.
5. Una vez tenemos la ciudad, pasamos la información al segundo componente del frontend, *resultado.component.ts*, el cual envía el nombre de cada lugar de interés al backend, realizando una conexión con nuestro *endpoint http://localhost:500/api/resumen.*
6. En el backend, se realiza una búsqueda en Wikipedia de cada lugar de interés, realizando un resumen de cada una de ellas.
7. Se devuelve al frontend para mostrar toda la información.

8.4. Diseño Interfaces – UI

Las interfaces cumplen un papel muy importante en las aplicaciones *web*. Para que una *web* sea efectiva, muchas veces necesita que sus interfaces sean fáciles de usar e intuitivas, ya que normalmente se busca que los servicios sean usados por el mayor número de personas posible. Por esta razón no es conveniente realizar una UI muy compleja y que pueda causar alguna contradicción o confusión.

En nuestro proyecto buscamos realizar una interfaz sencilla y muy intuitiva, pero a la vez efectiva, que no dé lugar a ningún tipo de problema de entendimiento.

Para hacer los *mockups* de las interfaces de nuestro proyecto hemos utilizado la herramienta de *Balsamiq*[23].

[23] https://balsamiq.com/

Nuestra aplicación *web* constará de dos interfaces, a las cuales llamaremos inicio y resultado para diferenciarlas.

8.4.1 Interfaz inicio

Primero hablaremos de la interfaz inicio, la cual será la principal, donde aparecerá el formulario que el usuario deberá rellenar con la información que vea necesaria. En la Figura 19 se puede ver cómo es la idea de interfaz de la página de inicio.

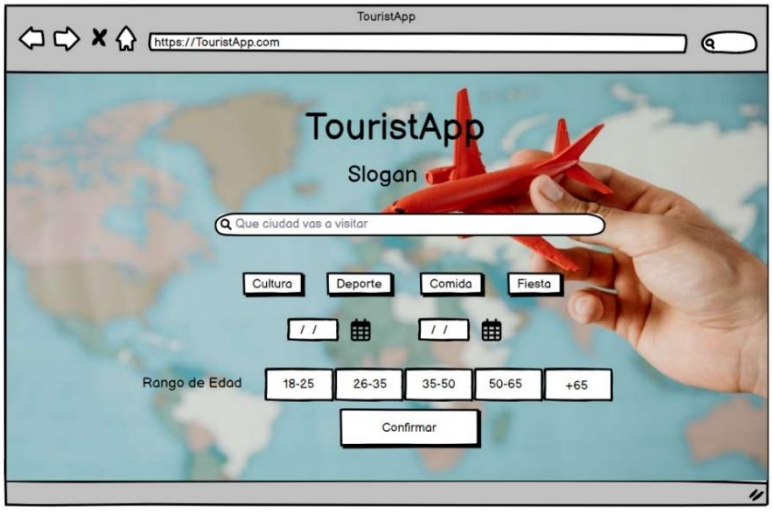


Figura 19. Diseño interfaz inicio

Como se puede apreciar, el formulario se divide en cuatro partes fundamentales: un cuadro de texto donde el usuario escribirá la ciudad que quiere visitar, una segunda parte donde el usuario indicará el objetivo por el que viaja, el siguiente para marcar las fechas de inicio y fin, y, para terminar, el rango de edad del usuario.

Se ha realizado la interfaz de esta forma para que no haya ningún tipo de confusión ni problema a la hora de rellenar la información necesaria.

8.4.2 Interfaz resultado

En la interfaz del resultado se mostrarán aquellos lugares de interés del usuario, tanto una imagen de cada lugar como un pequeño resumen o reseña de él. De manera adicional se podrán añadir algún tipo de enlace, como puede ser con un museo o un teatro que haya alguna exposición o concierto en los días que el usuario ha marcado que está de viaje.

En la Figura 20 se muestra como es la idea de la interfaz del resultado.

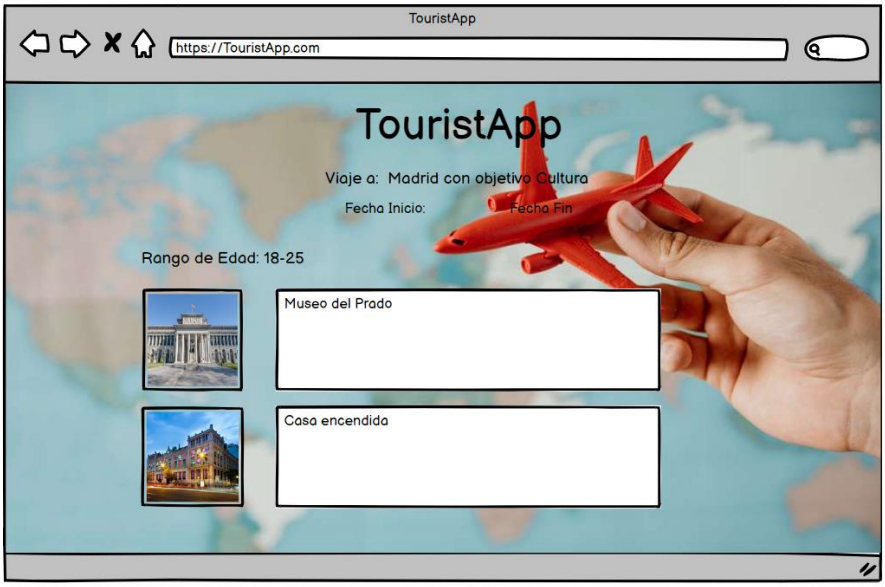


Figura 20. Diseño interfaz resultado

8.5. Guías de estilos

8.5.1 Paleta de colores

La paleta de colores utilizada para *TouristApp*, además de negro y blanco para pintar la fuente, ha sido una gama de azules, como se puede apreciar en la Figura 21.

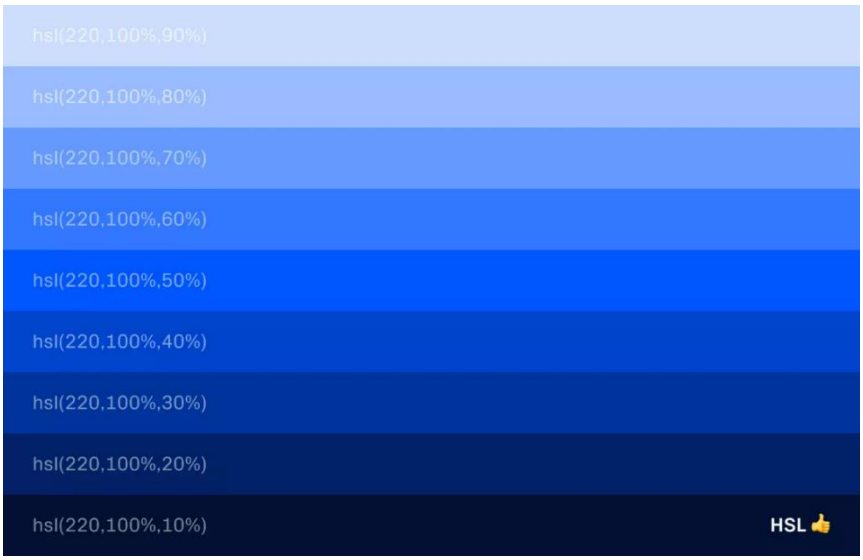


Figura 21. Paleta de colores

Se han escogido estos colores ya que el tema principal de nuestro proyecto es viajar, y el azul es un color que simboliza libertad, viaje, ya sea porque se le asocia con el color del cielo o con el tono del mar, cosas que simbolizan viajar. A la hora de hacer las hojas de estilos se ha utilizado el modelo de color, para poder jugar con la luminosidad del color azul. El modelo HSL está formado por tres componentes: matiz, saturación y luminosidad. El nombre son las iniciales de las tres componentes en inglés (*Hue, Saturarion, Luminosity*).

8.5.2 Tipografía

La tipografía usada en el proyecto ha sido *TT-Travel*. Esta fuente tiene más de diez estilos, pero el elegido para ha sido *TT-Travel DemiBold.* En la Figura 22 se muestra como es esta tipografía.

Figura 22. Tipografía de TouristApp

8.6. Diseño de pruebas y validación

Al tener una API para poder realizar el proyecto, es recomendable validar que todas las rutas funcionan de manera óptima. En el caso de TouristApp ha sido necesario crear una ruta para poder hacer la conexión de backend con frontend haciendo una llamada de tipo *POST*. Para confirmar que las llamadas a esta ruta han funcionado correctamente se ha utilizado la herramienta *Postman*[24].

En esta herramienta se ha creado un proyecto llamado TFG donde hemos almacenado la petición.

[24] https://www.postman.com/

9. Implementación

El siguiente apartado es el más importante a la hora de hablar de la implementación del código, ya que se va a explicar todo lo relacionado con este tema. Para que no haya ningún tipo de duda se explicará todo el desarrollo del proyecto paso a paso, desde que el usuario escribe sus datos en el formulario, hasta que se muestra toda la información de los lugares de interés de cada uno.

Para que quede todo más esquematizado, *TouristApp* consta de dos páginas llamadas inicio y resultado, siendo la primera la página en la que se rellena el formulario y la segunda en la que se muestra toda la información procesada. En la Figura 23 se muestra como es el flujo entre una página y otra y todos los procesamientos que ocurren a lo largo de estas.

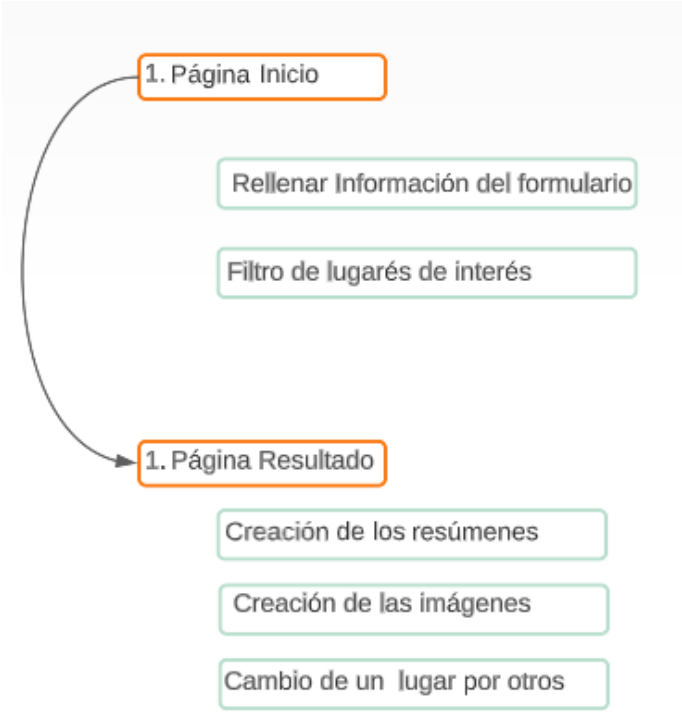


Figura 23. Flujo entre páginas

A continuación, se explicará paso a paso cada proceso de cada una de las dos páginas, con todos sus procesos, algoritmos y conexiones entre ellas. No solo es importante la explicación del código, sino también qué librerías se han ido usando a lo largo del proyecto y por qué, y en nuestro caso qué apis externas hemos utilizado y con qué fin.

Además de todo esto, al utilizar una estructura de front/back iremos saltando entre métodos implementados y desarrollados en el backend con otras funciones realizadas en el frontend. Cabe recordar que hemos desarrollado el frontend con Angular y el backend en Python.

9.1 Página Inicio

La primera página a explicar con todos sus procesos será la página de inicio, la cual es la primera que aparece al abrir la página *web*, lo que se conoce como la página de *Home*.

La página de inicio tiene como principal objetivo recoger toda la información que el usuario rellene en el formulario, con esa información filtrar sus lugares de interés, y una vez realizado todo eso, enviar la información a la página de resultado.

9.1.1 Rellenar el formulario

Se ha creado un formulario con tres valores a completar o seleccionar: la ciudad a la que se viaja, el objetivo por el que se viaja y el rango de edad del usuario. El primero de ellos es un *input* de tipo de texto, y los otros dos valores se han hecho mediante botones. Como objetivos hemos puesto las siguientes opciones: Comida, cultura, en familia y deporte. Como rango de edad hemos puesto las siguientes opciones: 18-25, 26-35, 36-50, 51-65 y 65+.

Una vez el usuario rellena el formulario con todos los datos necesarios, se pasa la información al método `enviar()`, el cual envía la información al servicio creado en el frontend llamado *`envio.service.ts`*. Este servicio es el que realiza todas las conexiones entre el frontend y el backend gracias a la librería de *httpClient*, la cual permite hacer llamadas a distintas *urls*, en nuestro caso, utilizamos la librería para hacer llamadas a los *endpoints* del api que hemos creado.

Esta función `enviar()` manda la información al método `sendData()` del servicio, el cual envía toda la información recogida en el formulario al backend. En este caso conectamos con el endpoint de nuestro api, realizando una llamada de tipo *POST* en /api/envio (explicado en el punto 8.2).

Una vez se conecta con este endpoint se realiza un paso crucial para el flujo del proyecto: detectar la ciudad dentro del texto del formulario.

En nuestro backend, gracias al uso de herramientas de procesamiento de lenguaje natural conseguimos saber cuál es la ciudad. El método que hemos llamado desde el frontend realiza lo siguiente: encuentra la ciudad y realiza una llamada a un api externo, la cual filtra los lugares

más adecuados de esa ciudad, dependiendo de la edad y el objetivo, valores que recordemos que se han enviado también.

Este proceso se ha realizado gracias a la librería de *Spacy*. Dicha librería permite realizar cualquier estudio de palabras en cualquier contexto (análisis sintáctico, morfológico, pragmático…). En nuestro caso se ha utilizado el método `label_` para estudiar si alguna palabra es una ciudad.

```python
def get_ciudad(span):
    if span.label_ in ("LOC"):
        entity_text = span.text.replace(" ", "_")
        return entity_text
```

Figura 24. Métod `get_ciudad()`

Como se aprecia en la Figura 24, el método recibe por parámetro el valor del *input*. Esa cadena se recorre y en caso de que una de las palabras sea una localización, es decir, una ciudad, se guarda en una variable y se devuelve.

9.1.2 Filtro de los lugares de interés

Una vez se ha rellenado el formulario y se ha diferenciado la ciudad es hora de filtrar los lugares de interés del usuario dependiendo de la información dada.

Para realizar el filtro de los lugares de interés de la ciudad se ha hecho uso de una inteligencia artificial. En la *web* de RapidApi se ha buscado un api acorde a nuestras necesidades, encontrando el api de Glavier/chatGPT.

```python
url = "https://chatgpt53.p.rapidapi.com/"

payload = {
        "messages": [
            {
                "role": "user",
                "content": "Enumera 8 Lugares de interés de" + str(lista[0]) +" para personas de "+ str(data['edad'])
            }
        ]
    }
headers = {
    "content-type": "application/json",
    "X-RapidAPI-Key": "df160a13f9mshb2095a7423acfb5p15a633jsn7e7ba094823e",
    "X-RapidAPI-Host": "chatgpt53.p.rapidapi.com"
}

# Guardo la respuesta y añado otros valores como la ciudad para tenerlo también en el frontned

response = requests.post(url, json=payload, headers=headers)
```

Figura 25. Conexión con api de IA

Como se ve en la Figura 25, se realiza la llamada gracias a la librería *request* de *Python*. Para realizar la llamada correctamente se necesita enviar un *JSON* que consta de dos elementos:

`payload` y `headers`. Nos importa el elemento `content` dentro de `payload`, donde hay que poner la solicitud al api. El mensaje que se envía es el siguiente: 'Enumera 8 lugares de interés de **ciudad** para personas de **edad** años con el objetivo de **objetivo**', siendo ciudad el valor que se ha recibido desde el método `get_ciudad()` explicado anteriormente, y edad y objetivo los valores enviados del frontend.

Una vez realizamos la llamada y tenemos le respuesta del api, lo pasamos a formato *JSON* y añadimos los valores de la ciudad, la edad y el objetivo a la respuesta para poder enviar toda esa información al frontend, como se puede apreciar en la Figura 26.

```
res = response.json()
res['ciudad'] = lista[0]
res['edad'] = data['edad']
res['objetivo'] = data['objetivo']
return jsonify(res), 201
```

Figura 26. Respuesta del filtro de lugares

Ya se ha hecho el filtro de los lugares desde el backend y se ha enviado al frontend. Una vez tenemos los datos en el frontend hay que realizar un pequeño procesamiento para poder tener los nombres de los lugares sin ningún detalle más como puede ser una descripción o un número que nos ha podido dar el api externo.

Tras realizar un estudio del api que se ha utilizado se han llegado a dos conclusiones: Todos los lugares de interés vienen separados por un salto de línea o más y cada lugar viene introducido por un número, ya que esta IA enumera los lugares, como se puede ver en Figura 27.

```
1. Museo del Prado
2. Palacio Real de Madrid
3. Templo de Debod
4. El Retiro
5. Plaza Mayor de Madrid
6. Gran Vía
7. Museo Nacional Centro de Arte Reina Sofía
8. Casa de Campo
```

Figura 27. Lugares interés de Madrid obtenidas por el api

Para solucionar esto hemos utilizado el método `split()`, el cual separa una cadena a partir de un carácter. Primero hemos separado la cadena por saltos de línea, es decir, por el carácter '\n', y una vez tenemos todos separados dentro de un vector, se ha vuelto a utilizar la función

`split()` para separar siempre que aparezcan los caracteres '.', '-' o ':', los cuales son los que este api pone junto al número a la hora de enumerar. En la Figura 28 se muestra como se ha realizado.

```
const resultado = res['choices']['0']['message']['content']
console.log(resultado);
const splitted = resultado.split(/\n+/, 8);
console.log(splitted)

for(let i = 0 ; i<8 ; i++){
  let aux = splitted[i].split(/[.:-]/)[1];
  this.listaLugares[i] = aux.replace(" ", "");
}
```

Figura 28. Split de los lugares de interés

En la variable resultado es en la que se guarda la información enviada desde el backend. Para hacer las separaciones de las cadenas hemos utilizado expresiones regulares, que no son más que patrones que se utilizan para hacer coincidir combinaciones de caracteres en cadenas [16].

Se han decidido guardar ocho lugares de interés, ya que en la página de resultado se van a mostrar cuatro, y para poder hacer la funcionalidad de cambiar un lugar de interés por otro hemos dejado otros cuatro guardados, se quedan de reserva para cuando el usuario quiera cambiar algún lugar por otro más adecuado.

Tras hacer todos estos filtros, conseguimos tener todos los lugares de interés sin nada más que pueda incordiar a la hora de utilizar los valores, como se puede ver en la Figura 29.

```
0: "Museo del Prado"
1: "Palacio Real de Madrid"
2: "Templo de Debod"
3: "El Retiro"
4: "Plaza Mayor de Madrid"
5: "Gran Vía"
6: "Museo Nacional Centro de Arte Reina Sofía"
7: "Casa de Campo"
```

Figura 29. Resultado final de lugares de interés de Madrid

Una vez tenemos todos los lugares de interés como se quiere, se almacena la información en el *LocalStorage* para poder usar esa información en otras páginas. El *LocalStorage* es una propiedad que accede al objeto *Storage* y tiene la función de almacenar datos de manera local

almacenando la información de forma indefinida o hasta que se decida limpiar los datos del navegador [17]. Por último, redirigimos nuestro proyecto a la página de resultado, como se muestra en la Figura 30.

```
localStorage.setItem('listaLugares', JSON.stringify( this.listaLugares ));
localStorage.setItem('ciudad', res['ciudad']);
this.router.navigate(['resultado']);
```

Figura 30. Conexión con página resultado

9.1.3 Diseño final

Antes de finalizar con la parte de inicio, se va a mostrar en la Figura 31 como ha quedado la interfaz final de dicha página

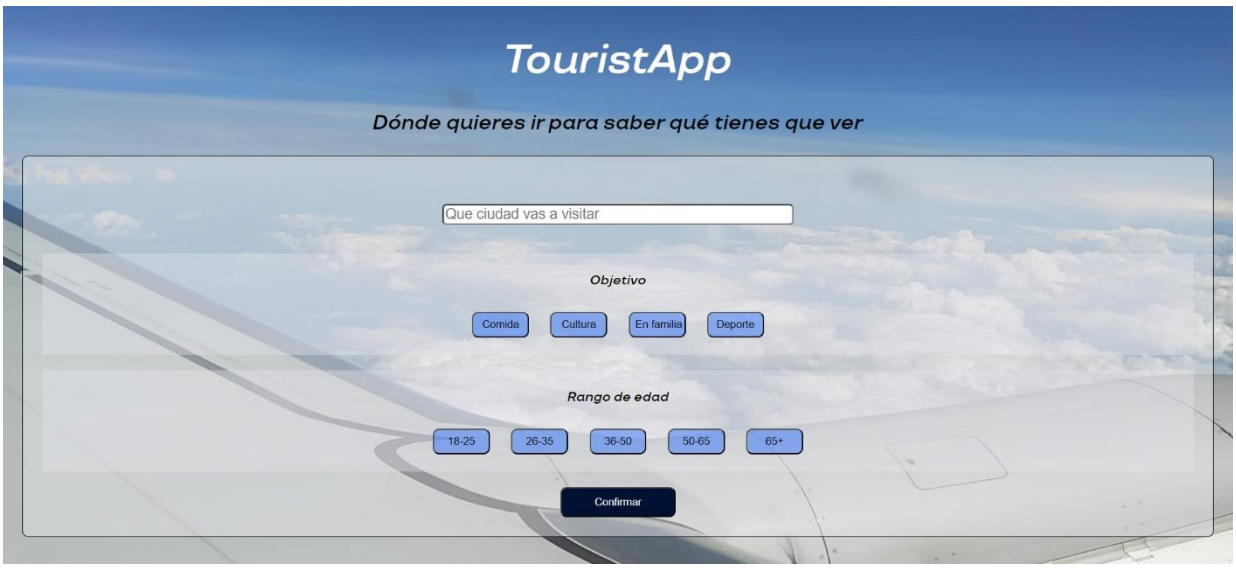


Figura 31. Interfaz final de inicio

Viendo el diseño de las interfaces y los *Mockups* mostrados en el punto 8.4, se puede decir que se han cumplido los objetivos en términos del diseño de la interfaz, quedando bastante parecida al diseño original pensado al principio del proyecto.

9.2 Página resultado

Una vez hecho todo lo comentado anteriormente en la página de inicio, se continuará con la página de resultado. Por poner en contexto, acabamos de redireccionar a la página de resultado

desde la página de inicio, teniendo guardado en el *LocalStorage* los ocho lugares de interés filtrados desde el backend y refinados desde el componente de inicio.

Por tanto, la página de resultado es en la que se muestra los resultados de la búsqueda, con los datos pasados en la página anterior, en la de inicio.

Antes de realizar las funcionalidades implementadas a lo largo de la página, en el método ngOnInit() de Angular recogemos los valores de los lugares de interés. Una vez se almacenan, se proceden a hacer tanto los resúmenes con la creación de las imágenes. Angular tiene un ciclo de vida de los eventos, y el método ngOnInit() es el segundo que se realiza. Esta función te permite inicializar el componente una vez ha recibido las propiedades de entrada [18].

Estos tres valores se han almacenados en vectores distintos de tamaño ocho cada uno, teniendo cada posición relación con la misma posición de otro vector. Para que se entienda mejor, se han creado los tres vectores, de nombre listaLugares[], resumenes[] y imageUrl[]. Entonces la primera posición de cada uno guardara un atributo de cada lugar de interés. Por ejemplo, si el lugar de interés es 'Museo de Prado', en listaLugares[] se almacena el nombre, en resúmenes[] el resumen del lugar y en imageUrl[] el url de la imagen del lugar.

En la Figura 32 se muestra cómo se han rellenado estos tres vectores:

```
ngOnInit(): void {
  this.listaLugares = JSON.parse(localStorage.getItem('listaLugares') || '{}');
  this.crearResumen()
  for(let i=0; i<8; i++){
    this.searchImages(this.listaLugares[i])
  }
    console.log(this.imageUrl)
}
```

Figura 32. Método OnInit()

A continuación, se explicará cómo se ha conseguido la información adicional para acompañar cada lugar de interés. En este caso, cada lugar de interés ha sido acompañado de un pequeño resumen y de una imagen. Además, se ha creado una funcionalidad en la que, si algún lugar no se adapta del todo al usuario, puede cambiarlo por otro pulsando un botón.

9.2.1 Creación de los resúmenes

Para la creación de resúmenes se han utilizado dos métodos, uno en el frontend y otro en el backend, además de la función que conecta uno con otro. Estos métodos son crearResumen en el frontend y createSummary() en el backend.

En el primer método recogemos los valores de la ciudad y de los lugares de interés para ser enviados al servicio y que este mande la información al backend para poder tratar con esta. Una vez es procesada la información, se almacena en la posición del vector de resúmenes correspondiente, como se puede apreciar en la Figura 33.

```
crearResumen(){
  for(let i = 0; i<8; i++){
    this.datosP.get('lugar').setValue(this.listaLugares[i])
    this.datosP.get('ciudad').setValue(localStorage.getItem('ciudad'))
    this.envio.sendSummary(this.datosP.value)
    .subscribe((res: any)=>{
      this.resumenes[i] = res['resumen']
    })
  }
  console.log(this.resumenes)
}
```

Figura 33. Crear resumen en frontend

El método que hemos creado en nuestro servicio para crear los resúmenes es `sendSummary()` y la llamada al backend se ha realizado de la misma manera que en la conexión en el filtro de lugares, con la librería *httpRequest*, realizando una llamada de tipo *POST* al *endpoint* de /api/resumen, pasando como parámetro un *JSON* con el lugar de interés y el nombre de la ciudad. Este proceso se hace ocho veces, una vez por cada uno de los lugares de interés que tenemos.

Una vez llega la información ya se puede crear el resumen del lugar de interés. Para realizar este resumen hemos utilizado principalmente: *Wikipedia-api* y *sumy.* La primera, para extraer información fiable; la segunda, para realizar el resumen. El resumen se ha realizado con el algoritmo conocido como *LexRankSummarizer,* el cual analiza un texto pasado, detecta cuáles son las palabras más importantes y relevantes, y realiza a partir de estas palabras con el número de oraciones que plazca al usuario. Existen otros tipos de algoritmos para resumir, pero se ha elegido este porque realiza un resumen extractivo. Con resumen extractivo nos referimos a que selecciona y extrae las oraciones más importantes del texto que se le ha pasado como entrada.

Por tanto, en el método de crear el resumen realizaremos una búsqueda en Wikipedia de los lugares de interés, extraeremos el texto de la página de inicio, y realizaremos el resumen con esta información.

Primero de todo, almacenamos en una variable los datos de la búsqueda, como se puede apreciar en la Figura 34.

```
search_params = {
    'action': 'query',
    'format': 'json',
    'list': 'search',
    'srsearch': lugar
}
```

Figura 34. Datos de búsqueda Wikipedia

La variable lugar que se muestra en el valor de *srsearch* es el nombre del lugar de interés, el cual hemos enviado desde el frontend y se ha recogido desde el *body* de la solicitud.

Una vez están todos los datos de búsqueda correctos, se procede a buscar en Wikipedia. Primero de todo, realizamos una búsqueda del lugar de interés, y de todos los resultados de búsqueda nos quedamos con el primero de ellos, guardándonos el título. Esto se realiza para que no haya ninguna confusión a la hora de buscar. En un primer intento se probó concatenando directamente el lugar de interés con la *url*, pero aparecía un problema. La ruta de *Wikipedia* tiene los nombres de manera literal, por tanto, si nosotros tenemos guardado como lugar de interés 'El museo del prado' en vez de 'Museo del prado' nos dará como resultado una página en blanco.

Ante este error se concluyó en usar el buscador, como se acaba de comentar, ya que no existe ninguna diferencia entre realizar una búsqueda de 'El museo del prado' o 'Museo del prado'; siempre nos dará como primer resultado la página del Museo del prado. En la Figura 35 es muestran los resultados de búsqueda de 'El museo del prado'.

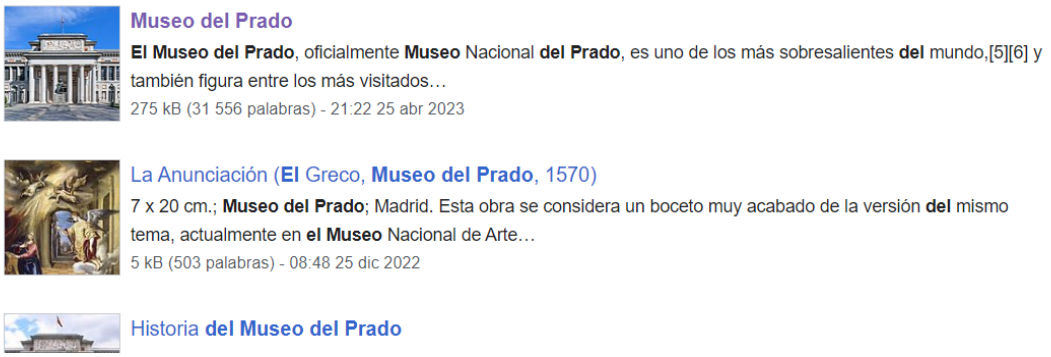


Figura 35. Resultados de búsqueda 'El museo del prado'

Una vez tenemos guardado el título de la página en nuestro código, nos aseguramos de en la cadena del título esté presenta el nombre de la ciudad. Este hecho se ha realizado tras analizar algunos fallos que aparecían. Por ejemplo, si como lugar de interés tenemos 'Mercado central', ¿cómo sabemos de dónde es el mercado central? ¿Qué solución tiene? Comprobar si la ciudad está presente, y si no lo está, la añadimos en la cadena del título. Por tanto, si se tiene 'Mercado central', pasará a llamarse 'Mercado central de Madrid'; y si se tiene como lugar de interés 'Teatro de Barcelona' se quedará como está porque el nombre de la ciudad sí está presente.

Una vez hemos refinado el nombre del lugar de interés, procedemos a realizar una segunda y última búsqueda en *Wikipedia*. Al igual que se ha hecho antes, se busca con el título ya refinado y asignamos a una variable el valor del texto de la página correspondiente, como se puede apreciar en la Figura 36.

```
# Comprobamos que el título tenga el nombre de la ciudad para que no haya fallos como Teatro Real (¿de dónde?)
if page_title.find(cadena1) == -1:
    # Si no está la ciudad en el título, lo añadimos al tituloy y realizamos una nueva búsqueda (ej. Teatro Real + 'de Madrid')
    page_title+= " de " + str(ciudad)
    page_title.replace("(desambiguación)", "")
    search_params = {
        'action': 'query',
        'format': 'json',
        'list': 'search',
        'srsearch': page_title
    }

    response = requests.get(search_url, params=search_params)
    data2 = response.json()
    print(page_title)

    # Hacemos la segunda búsqueda y recogemos el título del primer resultado de búsqueda
    if 'query' in data2 and 'search' in data2['query']:
        first_result = data2['query']['search'][0]
        page_title = first_result['title']
        wiki_wiki = wikipediaapi.Wikipedia('es')
        page_py = wiki_wiki.page(page_title)
        # Asignamos a la variable el valor del texto de la web
        resumen = page_py.summary

    else:
        print(f"No se encontró la página de Wikipedia para {page_title}")
```

Figura 36. Búsqueda en Wikipedia

Antes de seguir con la explicación, en la Figura 37 se muestra un diagrama de flujo de cómo se realiza la búsqueda para guardar el texto de la página de manera correcta para que no haya ninguna duda, empezando desde que se recibe la información desde el frontend hasta que se guarda el texto de la página web después de haberse refinado el texto que se va a utilizar y haber realizado dos búsquedas para asegurarnos que no hay ninguna confusión a la hora de coger la página correcta.

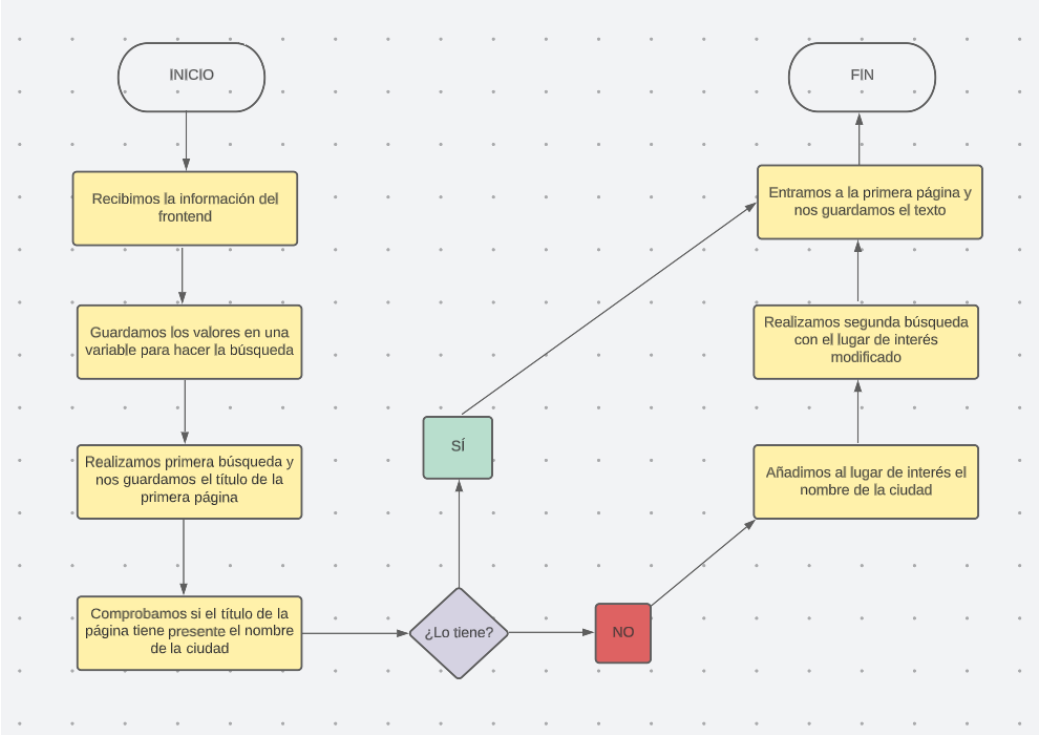


Figura 37. Diagrama de flujo de la búsqueda

Una vez ha quedado claro cómo se ha hecho la búsqueda y la recogida de la información, en este apartado queda por explicar cómo hemos resumido ese texto. Como se ha comentado antes, hemos usado la librería *sumy* para hacer uso de su herramienta de *LexRankSummarizer*. Se ha explicado antes que este algoritmo encuentra las palabras relevantes de un texto, y hace un resumen a partir de las frases del texto que contiene esas palabras clave. Pero antes de pasar el texto tal cual se ha recogido de *Wikipedia,* hay que eliminar aquellos caracteres que no son necesarios y no importan a la hora de hacer el resumen. En este caso, hemos eliminado de este texto las referencias. Estas aparecen con el siguiente formato: **[número]**, siendo el número la referencia. Para eliminarlas hemos hecho uso del método replace() de *Python,* y con ayuda de una expresión regular hemos eliminado todas las referencias. En la Figura 38 se muestra cómo se ha hecho este proceso en el código.

```
# Retiramos las referencias del texto para que no se muestren en el resumen
resumenSinCorchetes = re.sub(r'\[\d+\]', ' ', resumen)

# Hacemos el resumen con el algoritmo LenRankSummarizer y con tamaño de 2 oraciones
parser = PlaintextParser.from_string(resumenSinCorchetes, Tokenizer("spanish"))
summarizer = LexRankSummarizer()
resumenFinal = summarizer(parser.document, sentences_count=2)

# Asignamos al valor que devolvemos en el método el valor del resumen
for oracion in resumenFinal:
    res+=str(oracion)
```

Figura 38. Creación del resumen

Como se puede ver la expresión regular utilizado es '\[\d+\]', siendo \d+\ cualquier valor numérico, por tanto, cualquier valor numérico entre corchetes se sustituirá por un espacio en blanco.

Una vez se ha realizado todo este proceso, se guarda el resumen en una variable y se envía al frontend en formato *JSON* junto con el lugar de interés, acabando así con la explicación de la creación del resumen.

9.2.2 Creación de las imágenes

Ya tenemos completos tanto el nombre de los lugares de interés como sus resúmenes correspondientes. Con esta información tendríamos de sobra para poder mostrar los lugares de manera óptima. Sin embargo, en un principio se decidió que cada uno de los lugares, además de con un pequeño resumen, se acompañaría con una imagen.

Aunque en este apartado es en el que se ha tenido más percances y complicaciones para poder implementarlo de manera óptima, se ha conseguido implementar de la manera que se quería en un principio.

En esta función no hemos tenido que hacer uso del backend, se ha desarrollado todo desde el frontend, ya que en ningún momento se han guardado las imágenes en una base de datos ni nada por el estilo.

Al tener que encontrar una imagen por cada lugar de interés, lo óptimo ha sido utilizar un api externo dónde se ha podido buscar imágenes a partir de un título, en este caso, cada lugar de interés.

En un principio se intentó utilizar el api de la *web* de Pixabay[25], y se consiguió implementar de manera correcta. Sin embargo, encontramos un problema: al realizar la búsqueda había a veces que no existían imágenes. Por ejemplo, no había imágenes del museo MACA de Alicante.

Ante este problema se decidió utilizar el api de Google. Antes de poder utilizar esta herramienta se tuvo que crear tanto una clave de api de Google (*Api Key*) y una clave para el buscador o un ID de buscador (cx). La clave del api se creó en los servicios de Google, concretamente en la página de APIs y Servicios de Google Cloud[26]. Se tuvo que crear un proyecto, en este caso llamado 'Trabajo de fin de grado', y una vez creado pudimos obtener nuestra clave para poder utilizar el api de Google. Para poder crear nuestra clave del buscador tuvimos que crear un buscador programable dentro de la página de Buscador Programable de Google[27], al que llamamos 'TFG'. Este buscador se modificó de tal forma que se pudiese buscar además de texto, imágenes, que realmente es lo que queríamos desde un principio.

Una vez tenemos creadas las dos claves, la del api de Google y la de nuestro buscador, podemos empezar a implementar la llamada al api desde nuestro código.

Al igual que se ha realizado con los resúmenes, se ha llamado ocho veces al método que realiza la obtención de la *url* de la imagen, para tener una imagen por lugar de interés. Desde nuestro backend se ha llamada al servicio para seguir el mismo proceso que con las otras llamadas a nuestro api: realizar una llamada al api, en este caso a la de Google, mediante la librería *httpRequest* de Angular. Para utilizar el api de Google hay que realizar la llamada a la siguiente ruta: *https://www.googleapis.com/customsearch/v1*. En la Figura 39 se muestra el método `searchImage()`, el cual conecta con esta api.

```
searchImage(query: string) {
  const url = `${this.urlGoogle}?key=${this.apiKey}&cx=${this.cx}&q=${encodeURIComponent(query)}&searchType=image&num=1`;
  console.log(url)
  return this.http.get(url);
}
```

Figura 39. Conexión con el api de Google

En este caso las variables que se ven son: *urlGoogle*, que es la ruta comentada anteriormente, *apiKey* y *cx*, las claves del api de Google y del buscador, necesarias para utilizar este api; y por último, *query*, que es el valor que hemos pasado desde el frontend, en este caso, el título del lugar de interés. El api de Google devuelve un *JSON* con una gran cantidad de valores. Para el proyecto solo se ha utilizado al valor de *items[0].link,* es decir la dirección de la primera imagen

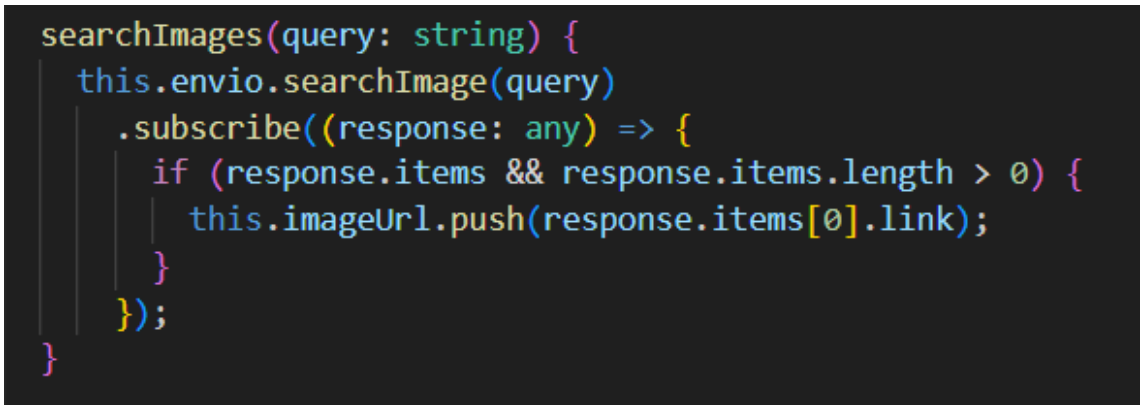
[25] https://pixabay.com/api/docs/
[26] https://console.cloud.google.com/apis/dashboard?project=trabajo-fin-de-grado-387417
[27] https://programmablesearchengine.google.com/controlpanel/all

del resultado de búsqueda. En la Figura 40 se muestra como es la respuesta del api de Google al pedir una imagen del museo del prado.


```
"items": [
  {
    "kind": "customsearch#result",
    "title": "Mythological Passions launches the Museo del Prado's temporary ...",
    "htmlTitle": "Mythological Passions launches \u003cb\u003ethe Museo del Prado&#39;s\u003c/b\u003e temporary ...",
    "link": "https://content3.cdnprado.net/imagenes/Documentos/imgsem/19/195b/195b4b65-60e0-67f3-b670-e4396234daa3/717aaa24-eff9-7ee4-fda9-d3cb2d9e68db.jpg",
    "displayLink": "www.museodelprado.es",
    "snippet": "Mythological Passions launches the Museo del Prado's temporary ...",
    "htmlSnippet": "Mythological Passions launches \u003cb\u003ethe Museo del Prado&#39;s\u003c/b\u003e temporary ...",
    "mime": "image/jpeg",
    "fileFormat": "image/jpeg",
    "image": {
      "contextLink": "https://www.museodelprado.es/en/whats-on/new/mythological-passions-launches-the-museo-del/195b4b65-60e0-67f3-b670-e4396234daa3",
      "height": 487,
      "width": 730,
      "byteSize": 308551,
      "thumbnailLink": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS-vN_SEunzj9ynf7Zt403B_1SGM3DQ9P3bU90KXAaKc8Q-eeZGfhRUcIo&s",
      "thumbnailHeight": 94,
      "thumbnailWidth": 141
    }
  }
]
```


Figura 40. Valor de ítems en la respuesta del api de Google

Esta llamada se recoge en el método `searchImages()` del frontend, en el que seleccionamos el valor del *link* comentado anteriormente y se va añadiendo cada *url* al vector de imágenes, como se puede apreciar en la Figura 41.



```
searchImages(query: string) {
  this.envio.searchImage(query)
    .subscribe((response: any) => {
      if (response.items && response.items.length > 0) {
        this.imageUrl.push(response.items[0].link);
      }
    });
}
```

Figura 41. Adición de imágenes al vector

Como se aprecia, antes de añadir la imagen comprobamos que el api nos ha devuelto algo, comprobando que el tamaño de la respuesta es mayor que 0.

Para concluir el tema de las imágenes, existe un problema. Este api de Google tiene un límite de llamadas diarias. Por tanto, si se realizan demasiadas solicitudes al servicio, nos dará un error del api, el error 429: *Too Many Requets.* Es un problema que a futuro se solucionaría, cambiando el api o creando algún tipo de base de datos.

9.2.3 Cambio un lugar por otros

La última funcionalidad a explicar el cambio de lugar. Una vez terminada la implementación del filtro de lugares de interés junto con su resumen e imagen apropiado, se llega a una conclusión. Puede ser que el usuario no esté interesado en uno de los lugares que le hemos proporcionado.

En la *web* se muestran cuatro lugares de interés. Sin embargo, como se ha comentado antes, dentro del *LocalStorage* tenemos guardados ocho. Esta funcionalidad es la principal razón.

Si el usuario no está de acuerdo con alguna de los lugares dados, puede utilizar el botón de cambio para pasar al siguiente lugar de interés guardado. Por ejemplo, se ha realizado una llamada con 'Madrid', 'Cultura' y '18-25' para cada uno de los valores del formulario. Por pantalla, se muestran como resultado los lugares de 'Museo del Prado', 'Palacio Real de Madrid', 'Museo *Thyssen'* y 'Museo Reina Sofía'. Sin embargo, en el *localStorage* hemos guardado otros cuatro valores además de estos: 'CaixaForum Madrid', 'Mercado de San Miguel', 'Gran Vía' y 'Parque del Retiro'. Digamos que el usuario no está interesado en el 'Museo del Prado'. Al pulsar en el botón de cambio, 'Museo del Prado', se cambiaría por el siguiente valor del vector, en este caso 'CaixaForum Madrid', junto con su resumen y su imagen. En la Figura 42 se muestra como se ha realizado este cambio desde el frontend.

```
cambiarLugar(indice: any){
  this.listaLugares[indice] = this.listaLugares[this.cont]
  this.imageUrl[indice] = this.imageUrl[this.cont]
  this.resumenes[indice] = this.resumenes[this.cont]
  this.cont++
  if(this.cont >= 7){
    this.cont = 7
  }
}
```

Figura 42. Cambio de lugar

Hemos creado una variable llamada *cont*, la cual comienza con valor 4, ya que es en la posición en la que empieza los lugares que tenemos reservados para el cambio. Por parámetro pasamos la posición vamos a cambia. Por tanto, siguiendo con el ejemplo anterior del 'Museo del Prado' y 'CaixaForum Madrid', cambiaríamos las posiciones cero por la 4, tanto del vector de resúmenes como del de lugares y al de imágenes. Una vez se ha realizado este proceso, aumentamos en uno el valor del contador, por si el usuario quiere cambiar algún lugar más.

Actualmente, tenemos cuatro lugares guardados para el cambio. En un futuro este número se podría ampliar algo más.

9.2.4 Diseño final

Teniendo en cuenta cómo se realizaron los *mockups*, el resultado está bastante parecido. Se quería mostrar cada lugar de interés junto con un pequeño resumen y una imagen del lugar; y es lo que se ha conseguido finalmente. Pulsando el botón de 'CAMBIO', se pasaría de un lugar de interés a otro, como se ha comentado anteriormente. En la Figura 43 se muestra el diseño final de la página de resultado, mostrando solo dos resultados, ya que la página necesita hacer *scroll*, y estaríamos mostrando una figura demasiado grande.

En el siguiente se ha realizado la búsqueda de lugares de Segovia para el rango de edad entre 18 y 25 con el objetivo de cultura.

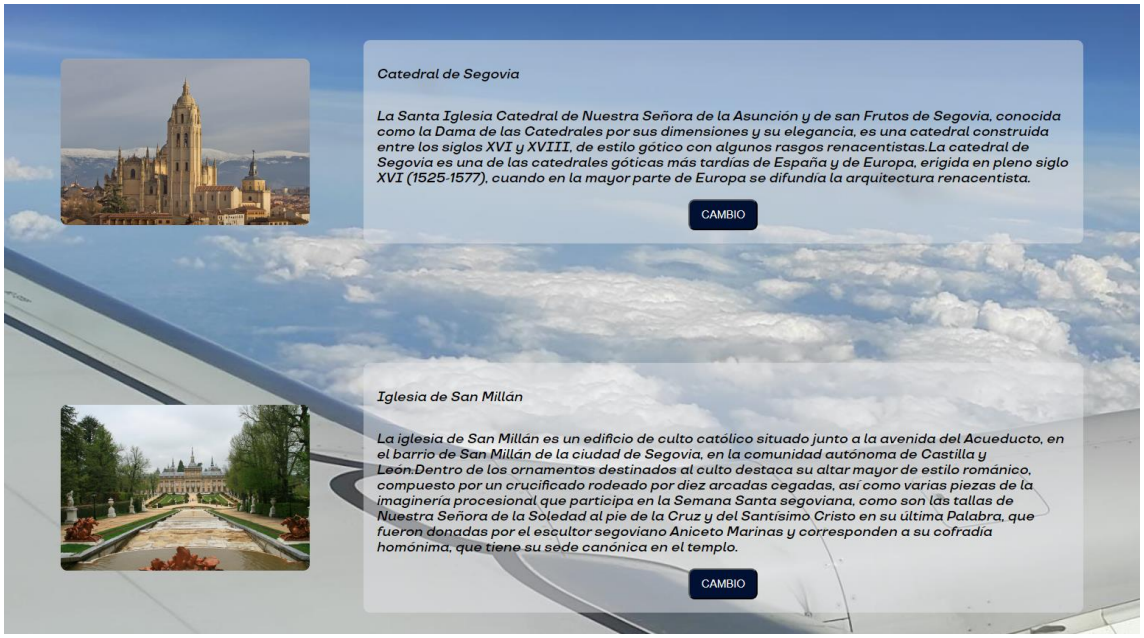


Figura 43. Diseño final página resultado

10. Pruebas, validación y resultados

A lo largo del proyecto se han ido realizado distintas pruebas para validar que todo funciona correctamente. Al tener un api, se ha tenido que comprobar muy minuciosamente que funciona todo de forma correcta.

10.1 Pruebas backend/frontend

Para comprobar que todas las llamadas funcionaban correctamente se han seguido los siguientes pasos: primero se crea lo necesario en el frontend y se verifica que funciona, después se crea lo necesario en el backend y se comprueba que todo funciona correctamente desde Postman[28], y por último realizamos la conexión.

El primer paso de validar, el frontend, al no tener ninguna herramienta se ha ido mostrando por consola todas las variables que queríamos usar, validando que todo tenía el valor esperado.

Por otra parte, en Postman se ha validado todas las llamadas en el api, como se ha comentado en el punto 8.2. En las Figura 45 y Figura 44 se muestra como todas las llamadas funcionan sin ningún tipo de percance.

Figura 44. Llamada correcta /api/envio

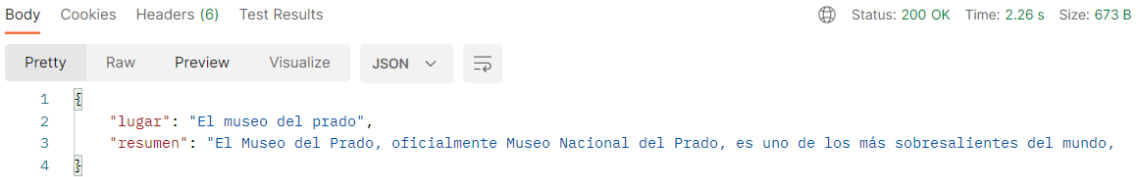


Figura 45. Llamada correcta /api/resumen

[28] https://www.postman.com/

Como se puede ver en ambas figuras, dan como resultado: *STATUS 200* y *STATUS 201*. El resultado 200 indica que la solicitud ha salido con éxito. El resultado 201 indica que se ha creado la respuesta correctamente. En este caso se puso 201 porque al atacar a un api externo, había que comprobar que todo se había creado óptimamente.

Una vez hemos validado que todo funciona correctamente, se ha único un elemento con otro mediante la librería de Angular *httpRequets*, como se ha comentado varias veces anteriormente.

El backend reconoce la ruta mediante la librería *Cors* de *Python*, la cual se encarga de conectar un método la ruta que se le determine. En la Figura 46 se muestra como se ha conectado el método `create_resource()` con la ruta /api/envio.

```
@app.route('/api/envio', methods=['POST'])
```

Figura 46. Conexión con la ruta

Para el método de crear el resumen se ha seguido el mismo procedimiento, pero cambiando la ruta a /api/resumen.

10.2 Pruebas con usuarios

Para validar que la interfaz y el funcionamiento es correcto y adaptado al usuario, se ha realizado un cuestionario de Google[29] y diez usuarios han probado la aplicación y han completado este formulario. Las respuestas del formulario son totalmente anónimas y los datos del que responda realmente no son importantes, ya que está enfocado a cualquier edad y lo que se quiere realmente es el resultado final, sin depender de la edad o el género de cada persona que realice el formulario.

Cada respuesta del formulario será utilizando la escala de Likert. Esta escala es una de las principales metodologías adoptadas en encuestas para investigación. Es un método de investigación que utiliza una escala de calificación para conocer el nivel de acuerdo y desacuerdo de las personas sobre un tema.

En el caso de nuestra encuesta, se ha hecho una escala de uno a cinco, siendo uno lo mínimo y cinco lo máximo. Si la pregunta es algún aspecto negativo, si la respuesta fuese cinco sería algo malo y si fuese uno sería algo bueno. Sin embargo, si la pregunta fuese sobre algún aspecto

[29] https://forms.gle/bu2L8xsfyDThcQLQ6

positivo y la respuesta fuese cinco significaría que se ha hecho bien el trabajo y viceversa. El formulario consta de las siguientes preguntas:

1. La interfaz ha sido sencilla e intuitiva
2. Los resultados han sido acordes a lo solicitado
3. Se han cambiado los lugares de manera correcta
4. Te has perdido algún momento usando la aplicación
5. Los resúmenes e imágenes tienen sentido
6. La aplicación puede ser útil

A continuación, se muestran los gráficos de los resultados de cada pregunta.

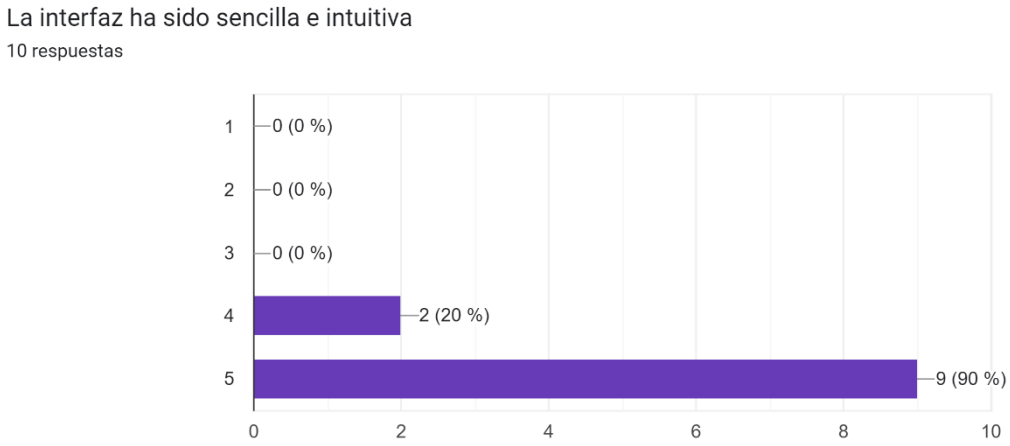


Figura 47. Gráfico pregunta 1

Como se puede apreciar en la Figura 47, los resultados sobre la interfaz han sido un éxito, dando como conclusión que se ha conseguido lo que se buscaba desde un principio: una interfaz en la que el usuario no tuviese ninguna duda a la hora de usarla, que fuese fácil de usar y, sobre todo, intuitiva y sencilla.

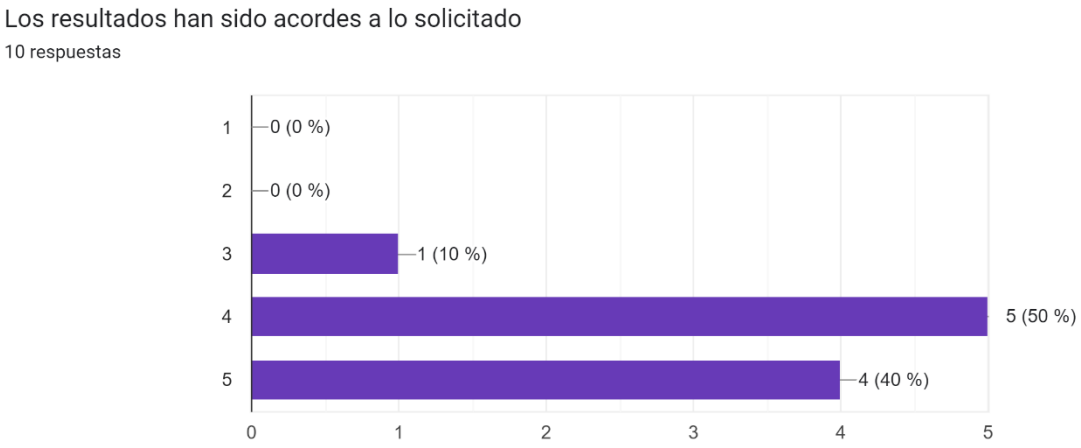


Figura 48. Gráfico pregunta 2

En la Figura 48 se muestran los resultados de la segunda pregunta, relacionada con los resultados dados. Excepto en uno de los casos, ha sido también un éxito todos los resultados. Cabe recalcar que en la encuesta que se dio un tres como resultado, el usuario añadió un comentario que, al cambiar tres veces el lugar de interés, hubo uno que tardó en cambiarse, de ahí esa evaluación.

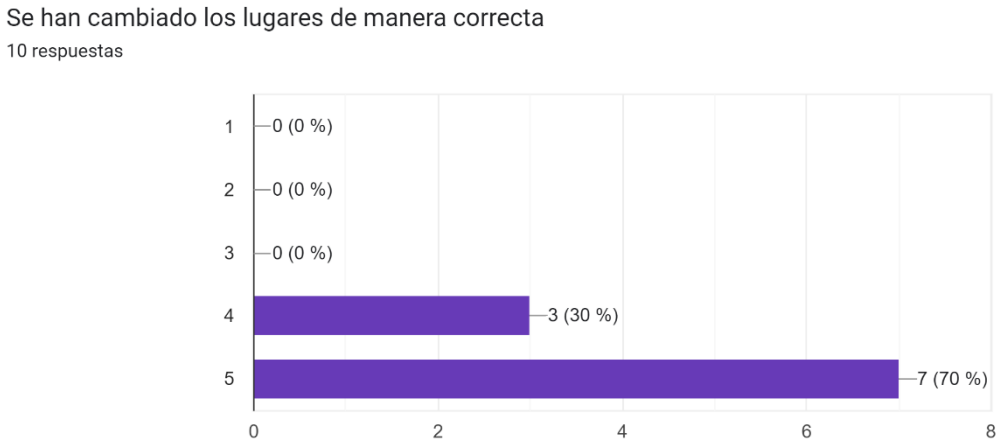


Figura 49. Gráfico pregunta 3

En la Figura 49 se muestran los resultados sobre la pregunta relacionada con el cambio de lugar. Como se puede apreciar, añadir esta funcionalidad ha sido un éxito, además de que funciona de manera correcta siempre.

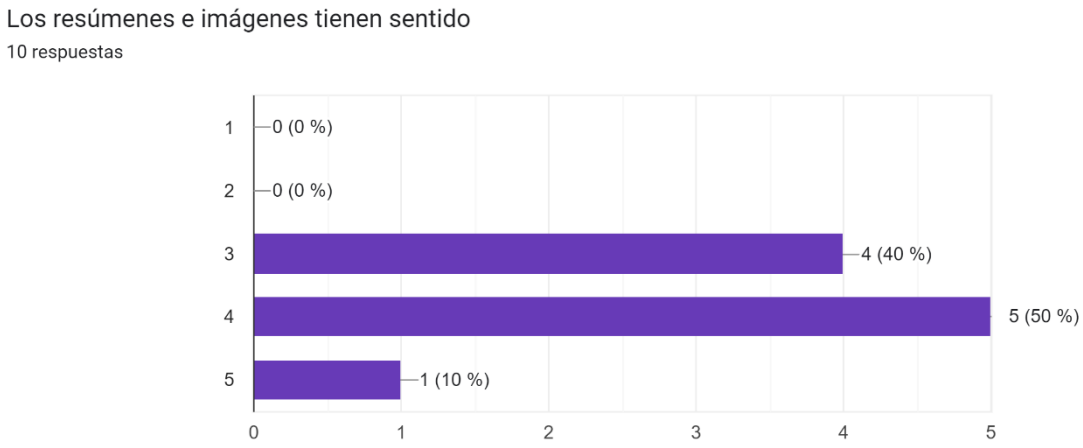


Figura 50. Gráfico pregunta 4

En la Figura 50 se ven los resultados relacionados con los resúmenes e imágenes, relacionada con la primera pregunta del formulario, pero en este caso centrándose más en las imágenes y resúmenes. Como se puede ver, se han tenido algunos problemas, sobre todo con las imágenes. Ha habido algún momento en el que las imágenes del segundo y tercer lugar se han intercambiado. Eso ocurrió por el tiempo en el que tarda en cargarse una y otra. A pesar de todo esto, ya se ha conseguido arreglar.

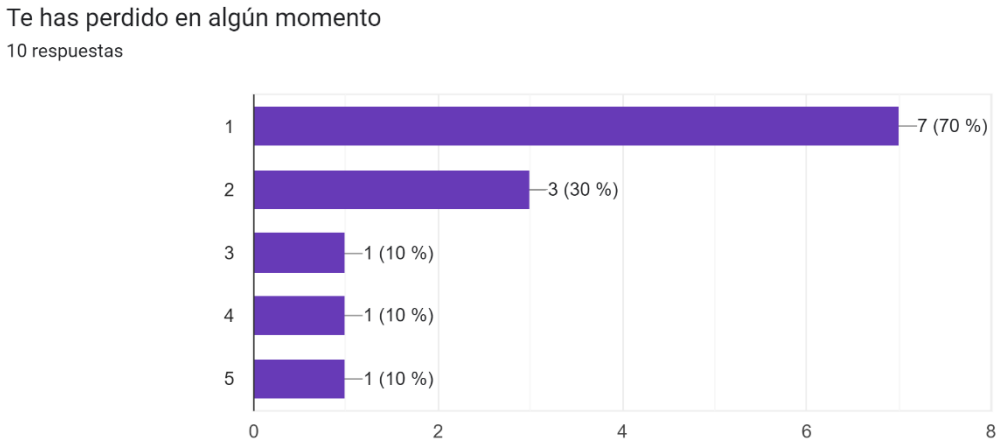


Figura 51. Gráfico pregunta 5

En la Figura 51 se muestran los resultados de la pregunta relacionada con el flujo de la aplicación. Cabe destacar que los usuarios que respondieron 4 y 5 tuvieron una confusión, creyendo que era lo buena, cuando era todo lo contrario. Sin contar esto, se pueden ver que los resultados muestran que no ha habido ninguna pérdida del usuario a la hora de utilizar *TouristApp.*

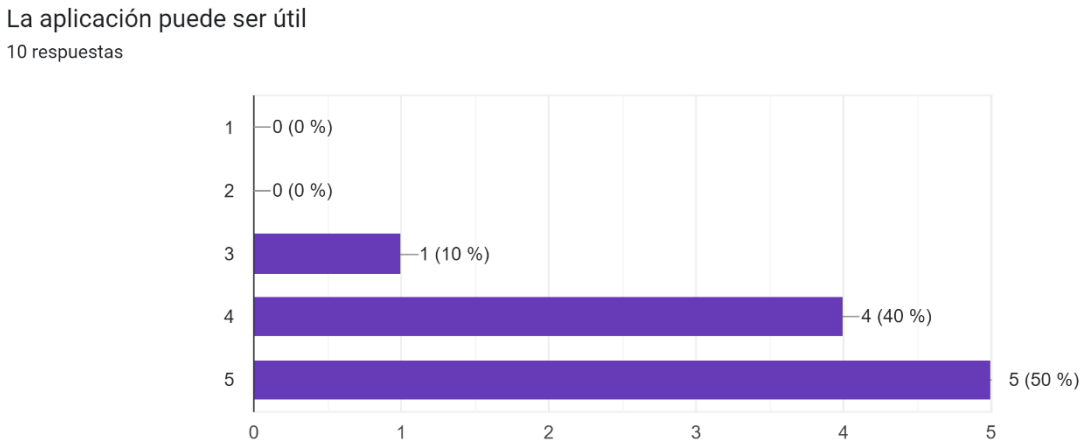


Figura 52. Gráfico pregunta 6

En la Figura 52 y la última sobre la encuesta, se muestran los resultados sobre la pregunta más pensada a futuro, si ven futuro a la aplicación, si puede llegar a ser útil, y como se pueda comprobar todos han pensado igual que los desarrolladores de este proyecto desde un principio, que sí que es útil.

Por tanto, tras ver todos los resultados y los gráficos podemos llegar a la conclusión e que la interfaz es bastante intuitiva y el usuario no ha tenido ningún problema a la hora de usarla, la mayoría de las veces los resultados son adaptados, los lugares se cambian sin problema, que hay veces que las imágenes no salen tan acorde como debería y que la aplicación sí que es realmente útil.

10.3 Resultados

Al comenzar este proyecto se estaba proponiendo una aplicación que filtrase los lugares de interés de una ciudad dependiendo de la edad y el objetivo por el que se viajaba. Una vez terminado el proyecto podemos decir que se ha conseguido lo que se quería, incluso con alguna funcionalidad adicional como mostrar alguna imagen de cada lugar o la posibilidad de que el usuario cambie algún lugar de interés que crea que no es de su gusto por otro distinto.

Una vez se ha terminado con la implementación del código, se va a comprobar si todos los objetivos marcados en un principio han sido cumplidos.

Tabla 7. Tabla Resultado de objetivos principales

OBJETIVO	RESULTADO
Diseñar una interfaz sencilla e intuitiva	Se ha conseguido hacer una interfaz usable y fácil de usar para el usuario. Además, eso se ha validado en el formulario de Google que se ha creado
Elaborar un estudio de mercado de forma completa analizando las alternativas existentes a nuestro proyecto	Se realizó antes de comenzar con el proyecto, estudiando toda la competencia del mercado para ver qué podíamos añadir para ser una opción distinta a las demás
Analizar y decidir las tecnologías más apropiadas para desarrollar el trabajo	Se ha realizado y de manera correcta. Se han elegido Angular y *Python* para el backend y frontend, elección que no ha causado ningún problema a la hora de implementar el código
Permitir a cualquier usuario acceder a la aplicación	Este objetivo no se ha cumplido realmente ya que la *web* solo está disponible en local. Para que esto se hubiese cumplido se tendría que haber subido el proyecto a un servidor
Planificar el desarrollo de la aplicación mediante las herramientas de gestión de proyectos correspondiente	Definitivamente sí. Se planificó y gestionó mediante una metodología *Scrum* y *Kanban*, las cuáles han surgido efecto a la hora de realizar el proyecto
Evaluar la utilidad y la efectividad de la aplicación desarrollada.	Sí se ha cumplido gracias al formulario comentado en el punto anterior
Aprender nociones básicas sobre el procesamiento del lenguaje natural y de las tecnologías del lenguaje humano, orientadas a la comprensión y generación de información	Es el objetivo que más se ha cumplido. Además de utilizar las herramientas adecuadas para el proyecto, se han estudiado otras por interés propio
Aprender y utilizar herramientas y librerías específicas de Procesamiento de Lenguaje Natural, así como frameworks para diseñar arquitecturas.	Junto al anterior, el objetivo más importante, y claramente sí se ha complicado. Tanto para detectar la ciudad como para crear el resumen se han utilizado herramientas de PLN.

Tabla 8. Tabla Resultados de objetivos personales

OBJETIVO	RESULTADO
Utilizar un lenguaje de programación nuevo como Python.	Se ha utilizado un lenguaje nuevo a lo largo del proyecto el cual no se había visto en todo el grado de ingeniería multimedia
Realizar un documento de las dimensiones de la memoria de un TFG.	Gracias a las revisiones de la tutora, se ha conseguido hacer una memoria bastante correcta
Aprender a realizar completamente una aplicación pasando por todas sus fases: desde investigar a fondo lo que existe ahora en el mercado, pensar qué se necesita – tecnologías, etc. - y cómo añadirlo en el proyecto, diseñar sus interfaces y, desarrollarlo hasta que esté terminada.	Se ha ido paso a paso en el proyecto, y finalmente se puede concluir que ha sido la mejor opción, cumpliendo así este objetivo

10.4 Costes Temporales

Como se ha comentado en la memoria, se ha ido utilizando la herramienta de Clockify para ir midiendo los costes temporales. Como mínimo hay que realizar 300 horas de trabajo en este proyecto. Como se puede ver en la Figura 53 se ha superado con creces ese mínimo de 300 horas marcadas para el TFG.

Se puede ver que la mayoría del trabajo se ha hecho entre abril y mayo, ya que el trabajo se empezó a finales de marzo del 2023.

Figura 53. Costes temporales Clockify

10.5 Estado actual de la aplicación

Actualmente la aplicación se encuentra terminada, a falta de limar algunos detalles y añadir alguna funcionalidad, que se comentarán en el punto de trabajo futuro. Solo se puede utilizar en local. Sin embargo, esta aplicación terminada al 100% puede llegar a tener un gran futuro comercial.

El resultado final de la implementación del código está accesible en GitHub, en la siguiente ruta: https://github.com/pabloskyjr/tfg.git

11. Conclusiones y trabajo futuro

11.1 Conclusiones personales

Este trabajo de fin de grado ha sido un trayecto largo y trabajado en el que se ha ido aprendiendo nuevas herramientas y nos hemos enfrentado a un proyecto de manera individual. Sin embargo, no significa que haya sido algo aburrido, sino todo lo contrario, ha sido un trabajo complicado pero divertido, ya que eran temas interesantes con nuevas tecnologías para usar. Una vez terminado el proyecto se han llegado a las siguientes conclusiones:

- Las herramientas de PLN son realmente útiles y quizá deberían ser vistas en algún momento a lo largo de la carrera.
- Al igual que el PLN, el lenguaje de *Python* es muy polivalente y se podría ver en algún momento en el grado de ingeniería multimedia.
- Es tan importante el proceso del proyecto como la planificación previa de este.
- A nivel personal prefiero realizar trabajos de manera colectiva que individual.
- Este trabajo implementado de manera aún mejor puede tener mucho futuro a nivel comercial.
- Muchas veces se tarde más en pensar cómo hacer algo que en realmente hacerlo.

11.2 Trabajo futuro

Se han cumplido con la mayoría de los objetivos. Sin embargo, durante el proceso de implementación del código han ido surgiendo funcionalidades que se podrían haber implementado. Mirando hacia el futuro esas funciones se podrían realizar de manera funcional e incluso podríamos añadir más cosas.

- Cambiar la forma de crear las imágenes ya que el api de Google tiene un límite de llamadas al día.
- Poder descargar de alguna forma los lugares de interés que se quieran, ya sea en formato .pdf o algo por el estilo.
- Poder jugar con las fechas de inicio y fin, viendo por ejemplo qué obras teatrales hay en algún teatro o qué exposiciones hay en un museo.
- Cambiar la fuente de información por las oficinas de turismo o algo por el estilo.

- Subir la página *web* a un servidor para que cualquier persona pueda utilizar *TouristApp.*
- Agilizar el tiempo de respuesta de la IA que realiza el filtro de lugares.
- Añadir más lugares a la reserva para poder cambiar más de cuatro veces.

Referencias

1. Esta es la razón científica por la que alguien ama viajar (2016). https://www.semana.com/hogar-y-familia/articulo/por-que-me-gusta-viajar/60709/
2. Hinojosa, V. (2022). Cómo ha cambiado el COVID nuestra forma de viajar: ochos nuevos hábitos.https://www.hosteltur.com/150165_como-la-covid-ha-cambiado-nuestra-forma-de-viajar-ocho-nuevos-habitos.html
3. Turismo en España: las cifras que reflejan cómo los viajes y escapadas hicieron crecer la economía en 2022. (2023) https://www.cronista.com/espana/economia-finanzas/turismo-en-espana-las-cifras-que-reflejan-como-los-viajes-y-escapadas-hicieron-crecer-la-economia-en-2022/
4. Waisbord, S. (2019). The 5Ws and 1H of digital journalism. *Digital Journalism*, *7*(3), 351-358
5. Moreno, A (2022). Procesamiento del lenguaje natural, ¿Qué es?.https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/
6. Fernández, Y. (2023). ChatGPT: qué es, cómo usarlo y qué puedes hacer con este chat de inteligencia artificial GPT-3. https://www.xataka.com/basics/chatgpt-que-como-usarlo-que-puedes-hacer-este-chat-inteligencia-artificial
7. Ribiero, J.Antonio. (2023). ChatGTP y las alucinaciones de IA generativa. https://medium.com/chatgpt-learning/chatgtp-y-las-alucinaciones-de-ia-generativa-1cc0f0219499
8. Doran, G. T. (2013)). There's a S.M.A.R.T. Way to Write Management's Goals and Objectives. Management Review, Vol. 70 (11), pp. 35-36. 1981
9. Hurtado Sánchez, J (2021). Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla. https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/
10. Roche, J. (2021). Scrum: roles y responsabilidades. Los 3 roles de la metodología Scrum. https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html.
11. ¿Qué es Kanban? Explicación para principiantes (2020) .https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban.
12. Especificación de requisitos según el estándar de IEEE 830. (2008). https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf
13. Qué es una API REST, para qué sirve y ejemplos. (2022). https://blog.hubspot.es/website/que-es-api-rest

14. . ¿Qué es una API de REST? (2022). https://www.redhat.com/es/topics/api/what-is-a-rest-api
15. Frontend y backend: qué son, en qué se diferencian y ejemplos. (2022). https://blog.hubspot.es/website/frontend-y-backend
16. Expresiones regulares. (2022). https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_expressions
17. Mora, A. (2018). ¿Qué es y cómo utilizar localStorage y sessionStorage? https://ed.team/blog/que-es-y-como-utilizar-localstorage-y-sessionstorage
18. Oriol. E. (2018). Ciclos de vida en Angular: La guía definitiva. http://blog.enriqueoriol.com/2018/10/ciclos-de-vida-en-angular-la-guia-definitiva.html