ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

UNIDAD DE TITULACIÓN

IDENTIFICACIÓN DE INTERESES TURÍSTICOS DE UN USUARIO, ANALIZANDO SUS GUSTOS Y PREFERENCIAS ENCONTRADAS EN PUBLICACIONES REALIZADAS EN FACEBOOK

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

ROGER DAVID LAZA GAVILANEZ

roger.laza@epn.edu.ec

JONATHAN SANTIAGO VARGAS NILVE

jonathan.vargas01@epn.edu.ec

Director: Ing. REGINA MARITZOL TENEMAZA VERA, MSc.

maritzol.tenemaza@epn.edu.ec

marzo, 2022

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jonathan Santiago Vargas Nilve y Roger David Laza Gavilanez, bajo mi supervisión.

REGINA MARITZOL TENEMAZA VERA

Firmado digitalmente por REGINA MARITZOL TENEMAZA VERA
Nombre de reconocimiento (DN): cn=REGINA MARITZOL TENEMAZA VERA, serialNumber=201221114438, ou=ENTIDAD DE CERTIFICACION DE INFORMACION, o=SECURITY DATA S.A. 2, c=EC
Fecha: 2022.05.12 10:37:11 -05'00'

Ing. Regina Maritzol Tenemaza Vera, MSc.

DIRECTOR

DECLARACIÓN DE AUTORÍA

Nosotros, **JONATHAN SANTIAGO VARGAS NILVE** y **ROGER DAVID LAZA GAVILANEZ**, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Jonathan Santiago Vargas Nilve

Roger David Laza Gavilanez

DEDICATORIA JONATHAN VARGAS

Quiero agradecer a mis padres Rogelio y Esthela por haberme formado como una persona de valores, de principios, capaz de tomar buenas decisiones y por esta razón este logro se los dedico de todo corazón a ellos, quienes son el pilar fundamental en mi vida profesional y personal.

DEDICATORIA ROGER LAZA

El presente trabajo lo dedico principalmente a Dios, el forjador de mi camino, por darme fuerzas en el transcurso de mi carrera, a mi madre y mis hermanos que me ayudaron incondicionalmente, me motivaron constantemente para alcanzar mis anhelos dándome la fuerza necesaria para seguir adelante y lograr mi objetivo.

AGRADECIMIENTOS JONATHAN VARGAS

Agradezco a Dios, a mis padres por el apoyo incondicional, a mis hermanos que me han alegrado la vida, a mis amigos por compartir conocimientos, a mis profesores por capacitarme con los conocimientos necesarios para el mundo laboral MSc. Regina Maritzol Tenemaza por ser nuestra directora de tesis y a la Escuela Politécnica Nacional por permitirme ser parte de esta grandiosa comunidad politécnica.

AGRADECIMIENTOS ROGER LAZA

Mis agradecimientos se dirigen a Dios que ayudo en mi camino gracias por tu amor y tu bondad que no tiene fin, a mi madre quien me enseño el valor del esfuerzo y dedicación, a mis hermanos que me apoyaron incondicionalmente, a todas las personas que compartieron sus conocimientos, la Escuela Nacional Politécnica que me formo como profesional y la MSc. Regina Maritzol Tenemaza por su ayuda en la conclusión del presente trabajo.

ÍNDICE DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

Los sistemas de recomendación son una herramienta que establecen un conjunto de métricas y valoraciones en los datos del usuario para realizar predicciones sobre recomendaciones de elementos que ofrezcan un valor y utilidad permitiendo emitir una experiencia personalizada basada en los gustos y preferencia del usuario.

Actualmente las tecnologías de la información y la comunicación se ha convertido en un campo que ofrece un valor añadido al turista como lo son páginas web, blogs, videos que cubren las necesidades básicas del turista, con la evolución de los dispositivos móviles se ha provocado un cambio en la que los turistas interactúan y obtiene información. El presente trabajo de titulación propone una solución novedosa que consiste en desarrollar un sistema de recomendaciones en una aplicación móvil basados en los posts de Facebook que contenga información sobre los gustos y preferencias positivas acerca de sitios visitados por el usuario, para luego analizar los intereses descubiertos y como resultados obtener una ruta con puntos de interés identificados mediante los servicios de Google Maps.

Palabras claves: Procesamiento de lenguaje natural, puntos de interés, sistemas de recomendaciones, turismo, publicación

ABSTRACT

Recommendation systems are a tool that establishes a set of metrics and valuations on user data to make predictions on recommendations of elements that offer value and utility allowing to issue a personalized experience based on the tastes and preferences of the user.

Currently, information and communication technologies have become a field that offers added value to the tourist, such as web pages, blogs, videos that cover the basic needs of the tourist, with the evolution of mobile devices has caused a change in which tourists interact and obtain information. The present work proposes a novel solution that consists of developing a system of recommendations in a mobile application based on Facebook posts that contains information about the tastes and positive preferences about sites visited by the user, and then analyze the interests discovered and as a result obtain a route with points of interest identified through the services of Google Maps.

Keywords: Natural language processing, points of interest, recommender systems, tourism, publication.

1 INTRODUCCION

1.1 Planteamiento del problema

Facebook es una red social que permite conectarse, compartir información sobre lugares visitados, fotos y videos, publicar actualizaciones de estado y chatear con familiares y amigos a través de Internet. Facebook fue creado en el 2004 por Mark Zuckerberg, cualquier persona mayor a 13 años con un correo electrónico válido podría unirse y formar parte de Facebook. En la actualidad, Facebook es la red social más grande del mundo con más de 2700 millones de usuarios activos [1].

Facebook recopila información de los usuarios cuando interactúan con sus servicios como, por ejemplo: los datos personales, el contenido que se publica, los mensajes enviados a otras personas, ubicación de una foto, fecha de creación de un archivo, los hashtags, los grupos a lo que un usuario está conectado y como interactúa con ellos, la información de contacto, información del sistema operativo, las versiones del hardware y software del dispositivo móvil cuando se conecta a la aplicación Facebook, GPS, la cantidad de reacciones que un usuario hace a una publicación, a las páginas a las que dio la reacción, Me gusta, etc [2]. Toda la información almacenada por Facebook, mediante permisos de los propietarios de la información, podrían ser analizados con fines específicos, para ofrecer contenido personalizado.

Una reacción a una publicación en Facebook es una respuesta para expresar el sentimiento que produce dicha publicación. Se permite reaccionar a diferentes publicaciones y anuncios mediante el uso de los botones de reacciones en las que se tiene [3]:

- Me gusta: clásico botón de Facebook y la primera opción. Si no se desea expresar ninguna carita se puede elegir esta reacción.
- Me encanta: se utiliza cuando una publicación te ha gustado demasiado.
- Me importa: se utiliza cuando una publicación te demuestra simpatía, se utiliza más en las publicaciones que son dignas de compartir.
- Me divierte: se utiliza cuando una publicación te causo risa, gracia puedes optar por esta reacción sin la necesidad de escribir un jaja.
- Me asombra: se utiliza cuando una publicación te genero mucho impacto
- Me entristece: se utiliza cuando una publicación te hizo recordar algo triste o viste algo que realmente te conmovió.

- Me enoja: se utiliza cuando una publicación no te gusto y en vez de denunciar el anuncio se opta por esta reacción.

Las reacciones indican que los anuncios o publicaciones son relevantes para el público lo que conlleva a que se realicen comentarios y reacciones posteriores [4].

Debido al crecimiento exponencial de datos generados por los usuarios en una red social [5], se puede generar un modelo de comportamiento del usuario permitiendo observar los gustos y disgustos respecto a cualquier tema que se desee analizar. Y mediante minería de datos y filtros se determinarán patrones de interés respecto al área de turismo.

Si bien existen diversos lugares turísticos creados con la finalidad de impulsar este sector existe una gran cantidad de lugares que no toman relevancia ya sea por su ubicación, o por falta de fomentación, puesto que, el principal problema es que los residentes desconocen estos lugares que son destinados para el desarrollo del turismo. Pueden ser: parques, montañas, playas, iglesias, museos, lagunas, monumentos, miradores, etc. Para visitar cualquiera de estos lugares el usuario debe averiguar la ubicación del lugar, formas de llegar, comentarios positivos o negativos, el rating que es la calificación cuantitativa en la mayoría de las veces sobre cinco puntos que otros usuarios calificaron dependiendo si les agradó el lugar.

El problema principal es que las personas no están dispuestas a responder cuestionarios con el fin de exponer sus intereses, por ello las gigantes como Google, Amazon, Microsoft, YouTube entre otras siguen la huella digital del usuario. Para desarrolladores comunes, que no pueden competir con estas grandes empresas deben investigar formas de recoger información para identificar intereses de usuario.

Por lo expuesto, se propone desarrollar un filtro de información con las reacciones que reflejen un sentimiento positivo sobre páginas, post, publicaciones o álbumes expuestos en Facebook con el fin de desarrollar un sistema de recomendación turístico permitiendo una experiencia personalizada al usuario.

Para demostrar los resultados obtenidos se desarrollará una aplicación móvil que pedirá permisos a la red social Facebook y analizará la información generada por el usuario. Luego, mediante los servicios de Google Maps generará una ruta con los puntos de interés identificados.

1.2 OBJETIVOS

1.2.1 Objetivo General

Identificar intereses turísticos de un usuario, basado en la identificación de sentimientos expresados por el usuario en reacciones sobre imágenes compartidas en Facebook.

1.2.2 Objetivos Específicos

a. Identificar intereses del usuario mediante el texto comentado en imágenes desplegadas por el usuario con sus respectivas reacciones.
b. Analizar intereses de usuario, mediante la aplicación de una herramienta de clasificación de texto y análisis de sentimientos.
c. Desarrollar una aplicación para recomendar sitios basados en los resultados de los intereses descubiertos para el usuario en estudio.
d. Evaluar el descubrimiento de intereses mediante la aplicación.

2 MARCO TEÓRICO

2.1 Estado del arte

2.2 Sistemas de recomendaciones

Los sistemas de recomendaciones son herramientas y técnicas de software capaces de proporcionar sugerencias útiles y eficaces acerca de artículos o lugares que sean probablemente de gran interés para un usuario en específico [6]. Estos sistemas pueden ser aplicados a una gama amplia de aplicaciones como turismo, contenido multimedia, viajes, ventas, redes sociales, etc.

Los sistemas de recomendación tomaron relevancia en el campo de la investigación a partir de los años 90 desde la aparición de las primeras investigaciones sobre filtrado colaborativo. Desde entonces se ha estado trabajando en el desarrollo de nuevos enfoques para los sistemas de recomendación, aunque todavía siguen necesitando mejoras en analizar el comportamiento del usuario, la información sobre los artículos o lugares, métodos más avanzados de modelado de recomendación, incorporación de diversa información contextual, desarrollo de métodos de recomendación menos intrusivos y más flexibles [7].

Tipos de sistemas de recomendación

Sistemas de recomendación colaborativos: es el sistema más popular y el más implementado en los sistemas de recomendación. Este sistema hace recomendaciones personalizadas al usuario final basándose en las valoraciones de los productos que han gustado a otros usuarios en el pasado [6].

Sistemas de recomendación basados en el contexto: se centran en recopilar la información general del usuario (nombres, edad, sexo, gustos generales, preferencias, ubicación, etc.) previamente con los permisos y accesos otorgados por el usuario para después procesar esta información y entregarle como resultado final contenido y servicios personalizados [8].

Sistemas de recomendación basados en el contenido: estos sistemas utilizan las características y descripciones de los artículos [6] que el usuario ha visitado, comprado o calificado anteriormente y mediante una comparación entre estas características y las preferencias del usuario activo se recomienda los artículos que satisfacen sus necesidades. Es muy importante que los sistemas de recomendación estén constantemente actualizados con nueva información [9].

Para este proyecto se utilizará el sistema de recomendación basado en el contexto, debido que se realizará una recolección de datos en la red social de Facebook para que esta sea analizada, obteniendo como resultado los gustos y preferencias del usuario para entregarle al usuario sitios de interés y lugares turísticos que pueda visitar.

2.3 Machine Learning

Machine Learning (ML) o en español aprendizaje automático, es uno de los enfoques principales de la inteligencia artificial (IA). Formando parte de la informática donde los ordenadores son dotados con algoritmos para identificar patrones en datos masivos, este aprendizaje permite a los computadores realizar tareas específicas de forma autónoma, es decir, sin la necesidad de estar programados. Un resultado típico serían las sugerencias o predicciones en una situación particular.

Gracias al aprendizaje automático, en un futuro muchos de los dispositivos obtendrán experiencia y conocimientos a partir de la forma en que estos son utilizados, para poder ofrecer una experiencia personalizada al usuario. Se puede encontrar aplicaciones de esto en la personalización de los sitios de medios sociales como Facebook o los resultados del motor de búsqueda de Google, los algoritmos aplicados pueden aprender de los patrones y utilizar el conocimiento adquirido para tomar decisiones como los filtros de spam en el correo electrónico este tipo de aprendizaje es utilizado con el fin de detectar qué mensajes son correo basura y separarlos de aquellos que no lo son. La Figura 1, muestra tres subconjuntos del aprendizaje automático que pueden utilizarse: aprendizaje supervisado, no supervisado y de refuerzo.

Tipos de aprendizaje automático

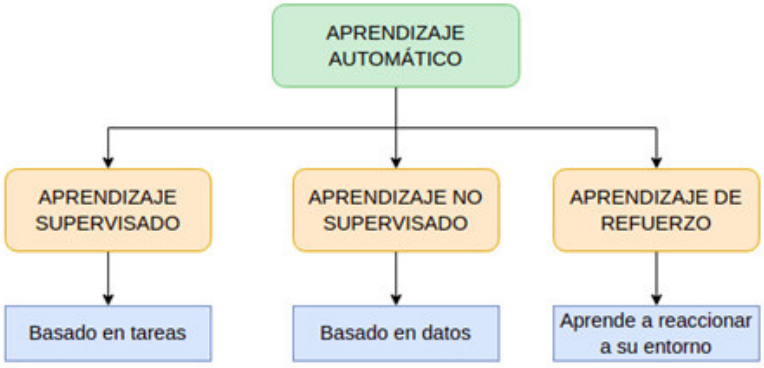

Figura 1. Tipos de aprendizaje automático

Fuente: elaboración propia

Como se puede apreciar en la Figura 1, la diferencia entre estas tres clases de aprendizaje automático es la forma en que maneja los algoritmos para la identificación de patrones. Por ejemplo: Existen 100.000 fotografías y los algoritmos deben identificar las fotos en las que aparezca un gato.

2.3.1 Aprendizaje supervisado, los algoritmos usan datos que ya han sido etiquetados u organizados previamente para indicar cómo tendría que ser categorizada la nueva información. Con este método, se requiere la intervención humana para proporcionar retroalimentación [10]. En el ejemplo indicado se debería enseñar al algoritmo fotos donde apareciera un gato para que luego pudiera identificar imágenes similares.

2.3.2 Aprendizaje no supervisado, los algoritmos no usan ningún dato etiquetado u organizado previamente para indicar cómo tendría que ser categorizada la nueva información, sino que tienen que encontrar la manera de clasificarlas por ellos mismos. Por tanto, este método no requiere la intervención humana [10]. En el ejemplo indicado los algoritmos deberían autoclasificar todas las fotos en las que apareciera un gato en una categoría.

2.3.3 Aprendizaje por refuerzo, los algoritmos aprenden de la experiencia. En otras palabras, tenemos que darles "un refuerzo positivo" cada vez que aciertan [10]. En el ejemplo indicado al algoritmo se le otorgará "recompensas" cada vez que este encuentre un gato.

2.3.4 Auto Machine Learning

Comúnmente conocido como AutoML es el proceso que automatiza cada paso del proceso de ML, desde el preprocesamiento de datos hasta la implementación del modelo ML, utilizando algoritmos de la IA. Esto permite a investigadores apartados del análisis de datos (como los desarrolladores de software) utilicen técnicas de ML sin necesidad de experiencia en el campo [11], en la Figura 2, podemos ver una simple representación de las entradas y salidas de un sistema AutoML.

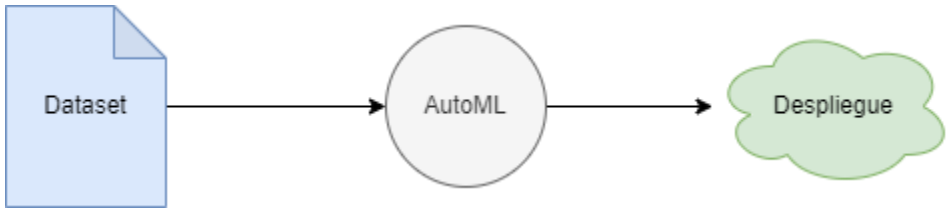

Figura 2. Esquema general del AutoML

Fuente: elaboración propia

Para el desarrollo de este proyecto se usó el servicio clasificación de documentos y textos AutoML de Google Cloud Platform para crear y entrenar un modelo de aprendizaje automático personalizado para predecir la categoría de una frase.

2.4 Productos de Lenguage Natural (Natural Language)

2.4.1 Procesamiento de Lenguaje Natural (PLN)

Según Alexander Gelbukh el PLN es un área de las ciencias de la computación, IA y la lingüística que estudia las interacciones entre las computadoras y el lenguaje humano, por medio del análisis sintáctico, semántico, pragmático y la estructura interna de las palabras, con la finalidad de escribir reglas para el reconocimiento de patrones estructurales, empleando un formalismo gramatical concreto [12]. Con la información de las palabras y la combinación de las reglas generadas, definen los patrones que hay que reconocer en una letra, palabra u oración. Se puede procesar el lenguaje natural por medio de reconocimiento de imágenes, voz o texto.

2.4.2 Clasificación de Contenido

La clasificación de textos o de contenido es una tarea de aprendizaje automático que clasifica los textos en base a una categorización. La clasificación se realiza sobre la base de palabras significativas extraídas del documento de texto. Debido que las categorías están predefinidas, es un proceso de aprendizaje automático supervisado. La clasificación automática de texto tiene muchas aplicaciones como clasificación de documentos de texto en formato electrónico, mejorar resultados de búsqueda en los motores de búsqueda, filtrado de spam, minería de sentimientos, extracción de opiniones, etc. [13].

Para el desarrollo de este proyecto se utilizó el servicio Clasificación de documentos y textos de AutoML para entrenar y compilar un modelo de aprendizaje automático para clasificar contenido de texto en un conjunto de categorías (church, museum, restaurant, park).

2.5 Herramientas, Frameworks y servicios utilizados en el proyecto

Herramientas

Nombre	Uso	Descripción
 Visual Studio Code	Desarrollo del Backend	Visual Studio Code es un editor libre que se ejecuta en el escritorio y está disponible para diferentes sistemas operativos. Permite construir aplicaciones web y cloud modernas [14].
 WebStorm	Desarrollo del Frontend	WebStorm es un ambiente de desarrollo integrado (IDE) para JavaScript y otras tecnologías relacionadas con el desarrollo de aplicaciones web [15].
 GitKraken	Cliente Git	Gitkraken es un cliente Git multiplataforma que simplifica los comandos de Git en simples acciones como arrastrar y soltar. Además, que resuelve los conflictos de combinación de una manera simple [16].
 Git	Control de versiones del código del Frontend y del Backend	Git es un sistema de control de versiones distribuido de código abierto y gratuito. Se usa para detectar y manejar los cambios realizados en cualquier proyecto grande o pequeño con rapidez y eficiencia. Generalmente Git es usado por desarrolladores de software [17].
 GitHub	Alojamiento del proyecto para el control de versiones	GitHub es una plataforma de alojamiento de código fuente que se utiliza para el control de versiones y la colaboración de equipos de trabajo desde cualquier lugar. GitHub permite crear repositorios, ramas, solicitudes de extracción, etc [18].

 Android Studio	Desarrollo de la aplicación móvil	Android Studio es un software multiplataforma que proporciona herramientas rápidas para crear aplicaciones para cualquier dispositivo con un sistema operativo Android [19].
 Postman	Prueba de Peticiones HTTP REST.	Postman es una plataforma API que se utiliza para construir, usar y probar APIs mediante solicitudes HTTP. Postman dispone de una interfaz gráfica en la que se obtiene diferentes tipos de respuesta y además que ayuda a construir mejores APIs [20].
 NPM	Instalar paquetes y dependencias	NPM (Node Package Manager) es un administrador de paquetes extremadamente configurable para la plataforma Node.Js. Se usa para publicar, instalar y desarrollar paquetes. NPM consta de: el sitio web para descubrir paquetes, la interfaz de línea de comandos (CLI) que es la terminal y el registro que es una gran base de datos pública de software [21].
 NinjaMock	Creación de mockups	NinjaMock es una herramienta basada en la nube que se utiliza para crear esquemas, mockups, maquetas, diseños para páginas web y aplicativos móviles con un aspecto profesional. NinjaMock permite administrar, exportar y compartir los diseños [22].
 Trello	Gestionar las tareas del equipo de desarrollo	Trello es una plataforma web que se utiliza para colaborar en proyectos, organizar tareas, gestionar proyectos y fomentar el trabajo en equipo desde un mismo lugar. Los pilares

		fundamentales de Trello son las listas y tarjetas que se sustentan en tableros. Las tarjetas de Trello permiten organizar, gestionar, supervisar y compartir tareas [23].
 Google Forms	Creacion del formulario para obtener las frases	Google Forms es una plataforma web gratuito de administración de formularios que permite crear formularios y encuestas con diferentes tipos de preguntas. Google Forms permite analizar los resultados obtenidos en tiempo real y desde cualquier dispositivo físico [24].

Tabla 1. Herramientas para el desarrollo del proyecto

Fuente: elaboración propia

Frameworks

Nombre	Uso	Descripción
 Ionic	Desarrollo de la aplicación móvil	Ionic es un framework de código abierto utilizado para crear aplicaciones móviles (Android, IOS) y de escritorio rápidas y altamente interactivas utilizando las tecnologías web como HTML, CSS y Javascript. Permite integraciones con otros framewoks como Angular, React y Vue [25].
 Angular	Desarrollo de la aplicación móvil	Angular es una plataforma para el desarrollo web moderno que permite crear aplicaciones web, móvil, móvil nativo y de escritorio utilizando HTML, CSS y Typescript [26].
 Node.js	Base de los Frameworks	Node.js es un entorno de tiempo de ejecución para JavaScript. Es multiplataforma y de código abierto que se ejecuta en el motor V8 Chrome [27].

 Capacitor	Transformar una aplicación web a un aplicativo móvil Android	Capacitor es un tiempo de ejecución nativo multiplataforma y de código abierto que permite crear aplicaciones web nativas Android, iOS y aplicaciones web progresivas multiplataformas utilizando HTML, CSS y Javascript [28].
 Flask	Implementación servidor (lógica del negocio)	Flask es un framework de aplicaciones web WSGI (Web Server Gateway Interface) ligero escrito en Python que permite desarrollar aplicaciones web de una manera sencilla [29].

Tabla 2. Frameworks para el Frontend y Backend

Fuente: elaboración propia

Lenguajes de programación y otros lenguajes

Nombre	Uso	Descripción
 TypeScript	Desarrollo del FrondEnd	TypeScript es un lenguaje de programación estrictamente tipado de alto nivel basado en JavaScript desarrollado y mantenido por Microsoft [30].
 Python	Desarrollo del Backend	Python es un lenguaje de programación de alto nivel interpretado. Python es usado para el desarrollo web, desarrollo de GUI, desarrollo de software, administración de sistemas, inteligencia artificial y análisis de datos [31].
 HTML	Desarrollo del Frontend	HTML (HyperText Markup Language) es el lenguaje de marcado estándar que define el significado y la estructura del contenido web. Un elemento HTML se encapsula mediante los símbolos < y > [32].

 CSS	Desarrollo del Frontend	CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizados para describir la presentación de un documento HTML o XML. CSS permite agregar estilos como colores, tamaño, forma, fuente, espaciado, etc. a los documentos web [33].

Tabla 3. Lenguajes de programación y otros lenguajes

Fuente: elaboración propia

Servicios

Nombre	Uso	Descripción
 Google Cloud Platform	Servicio en consola	Google Cloud Platform es un conjunto de servicios para computación, almacenamiento, redes, Big Data, Machine Learning, Internet de las Cosas (IoT), Inteligencia Artificial [34].
 APIs de Google Cloud	Comunicación entre cliente servidor	Son interfaces programáticas para los servicios de Google Cloud Platform. Permite agregar a las aplicaciones el procesamiento, el almacenamiento y el análisis de datos basado en el aprendizaje automático [35].
 AutoML Natural Language	Entrenamiento de modelo	Utiliza la tecnología de AutoML, para entrenar modelos de aprendizaje automático personalizados y de alta calidad para clasificar, extraer y detectar opiniones [36][37]. • Un modelo de clasificación analiza los datos del texto y muestra una lista de las categorías que se aplican al texto encontrado en los datos.

		• Un modelo de análisis de opiniones inspecciona datos de texto para identificar la opinión emocional si la actitud del escritor es positiva, negativa o neutral.
 API de Cloud Natural Language	Análisis de sentimientos y clasificación de contenidos	Permite extraer información sobre personas, lugares, eventos y muchos otros datos, también es usado para comprender opiniones sobre productos. Incluso puede subir documentos de texto para analizarlos. Funciones de la API de Cloud Natural Language: • **Análisis de opiniones:** Comprende la opinión general expresada en un bloque de texto. • **Clasificación de contenido:** Clasifica documentos en más de 700 categorías predefinidas. • **Análisis multilingüe:** Permite analizar fácilmente textos en varios idiomas como inglés, español, japonés, francés, alemán, italiano, coreano, portugués y chino (simplificado y tradicional). API de REST integrada: Permite acceder a través de la API de REST. El texto puede subirse en la solicitud o integrarse en Cloud Storage [38].

 Google Maps Platform	Ubicación y extracción de información de los lugares dependiendo las categorías	Google Maps Platform es un conjunto de APIs y SDK que permite insertar Google Maps en diferentes aplicaciones web y móviles. También permite extraer y recuperar datos de Google Maps. Se puede usar solo una API o SDK dependiendo de las necesidades del proyecto [39].
 API Graph	Extracción de datos del usuario de Facebook	API Graph es una API de Facebook basada en HTTP que permite a otras aplicaciones consultar datos, extraer datos, publicar historias, administrar anuncios, subir fotos entre otras acciones directamente desde la plataforma Facebook [40].
 Amazon Elastic Compute Cloud (Amazon EC2)	Para la creación de la Instancia Linux, Ubuntu Server	Amazon EC2 ofrece la plataforma de computación más amplia y profunda, con más de 475 instancias y la posibilidad de elegir el procesador, almacenamiento, redes, sistema operativo y modelo de compra más reciente [41].

Tabla 4. Servicios de Google Cloud, Facebook y AWS

Fuente: elaboración propia

2.6 Scrum

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria de software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Actualmente, gracias a las bases de la organización, sin ánimo de lucro, The Agile Alliance [42], las metodologías agiles representan el proceso más utilizado en el desarrollo de software [43], Los principales modelos de proceso ágiles son Extreme Programming (XP) y Scrum [43]. Los requisitos de flexibilidad, tiempo y costos por parte de los clientes de un determinado

software hacen que las metodologías ágiles se conviertan en la principal alternativa de desarrollo frente a procesos tradicionales [43].

Scrum es una metodología ágil, ligera y simple que ayuda a las personas, a los equipos y a las organizaciones a generar valor a través de soluciones adaptables. Scrum se basa en el empirismo y el pensamiento Lean (identificar y eliminar los desperdicios en los procesos productivos o de servicios) [44], además emplea un enfoque iterativo e incremental el cual permite optimizar la previsibilidad y controlar el riesgo. Scrum involucra a grupos de personas que tienen todas las habilidades y experiencia para hacer el trabajo y compartir dichas habilidades [45].

Para el desarrollo de este proyecto se ha decidido que la metodología que se usará de guía será Scrum considerando artefactos XP debido a las características que Scrum posee: simple, adaptable y rápido, por lo que será ideal para levantar requerimientos debido a que estos no están bien definidos y son numerosos

A continuación, se muestra el modelo general de la metodología Scrum en la Figura 3, llevando a cabo las distintas actividades separadas en diferentes etapas:

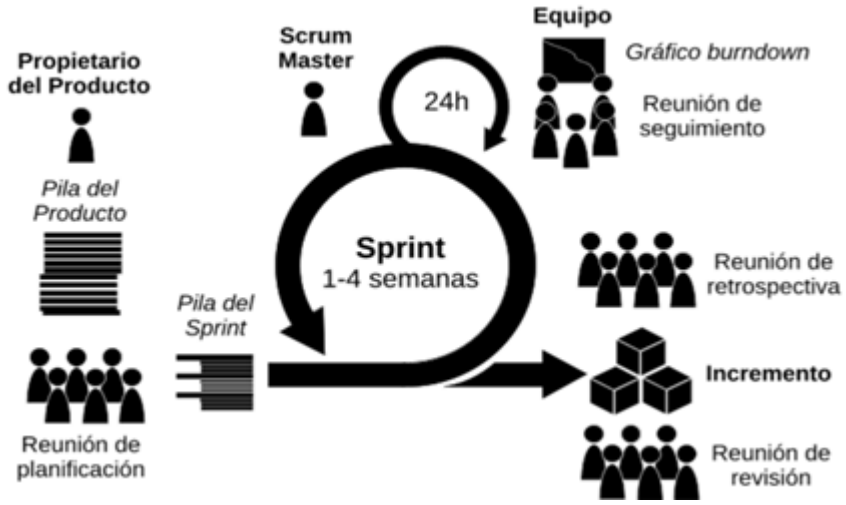

Figura 3. Modelo general de la metodología Scrum

Fuente: [46]

Equipo Scrum es la unidad fundamental de Scrum en las que constan un Scrum Master, el Product Owner y los desarrolladores. Los equipos Scrum son grupo de personas de máximo 10 personas que cuentan con todas las habilidades necesarias para entregar valor en cada entrega. Estos equipos deben ser multifuncionales y autogestionados [45].

Product Owner es el responsable de maximizar el valor del producto final del trabajo hecho por el equipo Scrum. También es el responsable de la buena gestión del Product Backlog. Es muy importante tener en cuenta que el Product Owner es una persona y no un comité [45].

Scrum Master es el responsable de hacer comprender la teoría y práctica de Scrum a todo el equipo scrum y también a toda la organización. El Scrum Master ayuda al equipo Scrum a centrarse en la creación de incrementos funcionales de alto valor, asegura que todos los eventos Scrum se cumplan en el tiempo establecido. El Scrum Master ayuda al Product Owner a encontrar técnicas para una definición de los objetivos del producto y la gestión de los retrasos [45].

Equipo de Desarrollo son las personas del equipo Scrum encargados de crear un incremento útil en cada Sprint [45].

Product Backlog consiste en una lista ordenada de lo que se necesita para mejorar el producto [45].

Sprint es la base fundamental del Scrum, donde las ideas se conviertan en valor. Son eventos que duran un mes o menos para crear consistencia. Un nuevo sprint comienza inmediatamente después de que terminó el Sprint anterior [45].

Sprint Retrospective es una reunión que se realiza al final de cada sprint y sirve para que el equipo scrum analice que fue bien, que problemas se encontró y como se resolvieron estos problemas durante el último sprint [45].

Sprint Backlog es un conjunto de tareas pendientes que se escogen para ser desarrolladas durante la ejecución del sprint [45].

Sprint Burndown Chart es una gráfica que se traza al finalizar el Sprint y muestra el trabajo completado por día frente a la tasa de finalización proyectada en la versión actual del proyecto [47].

3. METODOLOGÍA

3.1 Sprint 0 (18/10/2021)

Este sprint tuvo alta prioridad en el desarrollo del proyecto tanto de la parte lógica como de la parte de desarrollo, tuvo una duración de aproximadamente 2 semanas. Se definieron las herramientas de desarrollo, se realizó un estudio de las herramientas para el desarrollo del proyecto, preparación para la recolección de datos, se diseñó la arquitectura global que tendrá el proyecto, se definieron los roles del equipo Scrum, definición de las historias épicas, el Product Backlog y la cantidad de Sprints con su respectiva duración.

3.1.1 Fase de recolección de datos

Durante esta fase se recolectaron datos con ayuda de un formulario creado en Google Forms y se recogieron comentarios positivos de publicaciones de lugares turísticos de grupos de Facebook. Con estos datos se creó un archivo CSV y el producto Owner definió las categorías que permiten clasificar los datos obtenidos para su respectivo procesamiento. Después con este archivo CSV se creó y entrenó un modelo con el servicio Natural Language Classifier (NLC).

3.1.1.1 Recolección de Conjunto de datos

Con el fin de contar con datos iniciales para realizar el entrenamiento de un modelo para el Natural Language Classifier se procedió a crear un formulario utilizando Google Forms para extraer la mayor información posible sobre las frases positivas que utilizan los usuarios al momento de realizar una publicación en Facebook. El formulario constó de un total de 15 preguntas que se dividieron en categorías (Restaurantes, Playas, Parques, Iglesias o Museos, Montañas).

Las preguntas del formulario se enfocaron en las categorías anteriormente mencionadas, es decir, si la categoría parecía a **Restaurantes** la estructura es "**A continuación, deberá escribir un mensaje o frase positiva que utilizaría al hacer una publicación en Facebook**" y para más compresión del lector se colocó un ejemplo "**Una excelente comida en el mejor restaurante de comida rápida de la ciudad**", por cada categoría la persona respondió a publicación 1, publicación 2 y publicación 3 de maneras distintas.

El conjunto total de datos recolectados con ayuda de este formulario constó de 750 registros realizadas a 50 personas.

3.1.1.2 Estructura de los datos de entrenamiento

La estructura de datos válidos debe estar en formato CSV (comma-separated values) [48]. En este archivo CSV cada fila representa un registro y cada registro tiene dos columnas. La primera columna representa la frase de la publicación y la segunda columna representa su categoría. En la Figura 4, se puede observar 3 registros cada frase y su respectiva categoría.

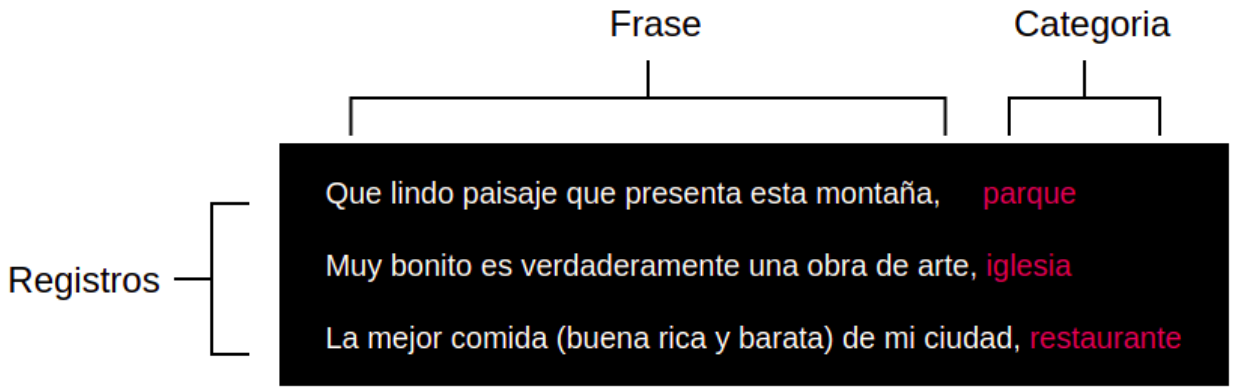

Figura 4. Estructura de un archivo CSV válido

Fuente: elaboración propia

3.1.1.3 Preparación y limpieza de datos

La cantidad de encuestados fueron un total de 50 personas y se obtuvo una cantidad de 750 registros. Se exportó los resultados de la encuesta desde Google Forms con la extensión CSV después, se limpiaron y se prepararon los datos y se cargó el archivo final al servicio Cloud Storage para su posterior entrenamiento con el servicio NLC. El archivo CSV final se lo puede observar en los Anexo I.

3.1.2 Definición de roles

Para la fase de implementación se definieron los siguientes roles, según lo descrito en el marco de trabajo Scrum. Este proyecto está conformado de tres integrantes que cumplen las funciones especificadas en la Tabla 5.

Rol Scrum	Responsable
Product Owner	Maritzol Tenemaza
Scrum Master	Maritzol Tenemaza
Development team	Jonathan Vargas, Roger Laza

Tabla 5. Definición de roles del equipo Scrum

Fuente: elaboración propia

3.1.3 Establecer la arquitectura del proyecto

La arquitectura del proyecto se basa en el patrón de diseño de software MVC (Modelo Vista Controlador) y se lo puede observar en la Figura 5. La lógica empieza cuando el usuario accede a la aplicación móvil, el aplicativo verifica y solicita activar el GPS para identificar la ubicación actual del usuario (latitud y longitud). Luego el usuario debe iniciar sesión con Facebook, existen dos opciones, 1) otorgar los permisos de acceso a Facebook [49], 2) restringir el acceso a sus datos en Facebook en cuyo caso si cierra la aplicación Facebook y el control de la aplicación retorna a login:

Permiso	Descripción
user_hometown	Permite leer la ciudad de origen del usuario.
user_location	Permite leer el nombre de la ciudad según la información que está en el campo de ubicación de su perfil
user_likes	Permite leer una lista que contiene las páginas a las que el usuario indicó que le gustan.
user_photos	Permite leer las fotos que el usuario subió a su cuenta.
user_posts	Permite acceder a las publicaciones que el usuario hizo en su biografía.
email	Permite leer la dirección de correo electrónico principal del usuario.
public_profile	Permite leer los campos de perfil público predeterminados en el nodo User. Campos de perfil público predeterminados: • id: identificador único para el usuario en Facebook • first_name: primer nombre del usuario

	• last_name: apellido del usuario • middle_name: segundo nombre del usuario • name: nombre completo del usuario • name_format: nombre del usuario formateado para manejar de manera correcta en los pedidos de coreano, chino o japones. • picture: Url de la foto del perfil del usuario • short_name: nombre abreviado del usuario

Tabla 6. Permisos de Facebook

Fuente: elaboración propia

Después se extrae su información personal (nombres, correo electrónico, foto de perfil y ciudad) y las publicaciones que pueden tener la siguiente información.

- Identificador
- Imagen
- Frase
- Ubicación del lugar
    - ❖ Nombre, latitud, longitud, código postal, ciudad, país, nombre de las calles.
- Reacción (Me gusta, Me encanta, Me importa, Me divierte, Me asombra, Me entristece, Me enoja)

Luego se hace una limpieza de datos para extraer solo las publicaciones que cumplen con los siguientes requisitos establecidos por el Product Owner:

- Imagen
- Frase
- Ubicación del lugar
    - ❖ Nombre, latitud, longitud, código postal, ciudad, país, nombre de las calles.
- Reacción (Me gusta o Me encanta o Me importa o ninguna)

Se realiza un trabajo de filtrado de publicaciones que cumplen con las condiciones especificadas. Se agrega la ubicación actual del usuario y se envían en formato JSON al servidor. El servidor comprueba que los posts no sean nulos para transformar los mensajes de los posts en oraciones concatenadas con el nombre del lugar donde de la imagen. Estas oraciones son enviadas al analizador de sentimientos de Google y este regresa un arreglo de

oraciones con su respectivo score para identificar si la oración es positiva o negativa. Si los posts son nulos entonces se utiliza la etiqueta tourist_attraction y se hace una petición al servicio de Google Maps para obtener los lugares turísticos que estén alrededor del usuario y enviarlos al aplicativo móvil y terminar con el proceso.

El score esta entre la magnitud de –1 y 1, donde el –1 indica un sentimiento negativo y el 1 un sentimiento positivo, para el desarrollo de este proyecto se toma el score superior o igual 0.7 considerado un valor adecuado, si la oración cumple con esta condición es enviada al Natural Lenguaje Classifier, por el contrario, si no cumple la oración es rechazada.

El clasificador realiza una predicción del mensaje como resultado se obtendrá un score del 0 a 1 de probabilidad en cada categoría, tomando el valor más alto como la categoría que pertenece el mensaje analizado, adicionalmente con los scores obtenidos se los ordenará de mayor a menor y se trabajará con los 4 primeros lugares, agregándole un peso al tag de acuerdo a su posición:

- Primera posición no se agrega peso debido al alto valor que tiene el score.
- Segunda posición se agrega un peso de 0.5,
- Tercera posición se agrega un peso de 0.3.
- Cuarta posición se agrega un peso de 0.1.

La categoría con mayor puntuación es enviada a Google Maps, el cual retorna los lugares mejor puntuados en un radio de 1.5 kilómetros, nuevamente se realiza una limpieza de datos para que sea enviado a la aplicación móvil.

Finalmente, se reciben los datos en formato JSON para mostrarlos al usuario. La información que el aplicativo móvil recibe se detalla a continuación:

- Nombre del lugar
- Ubicación del lugar (latitud y longitud)
- Icono
- Rating: calificación sobre 5 puntos
- Tipo de lugar (restaurant, iglesia, museo, parque)
- User ratings: cantidad de usuarios que visitaron y calificaron el lugar
- Rating place: operación matemática para calcular el mejor lugar. Esta variable consiste en una división entre total de usuarios y rating.

Para que el sistema de recomendación muestre más lugares de posible interés del usuario se agregó en las siguientes etiquetas: acuario, bar, cafetería, estadio, teatro, zoológico.

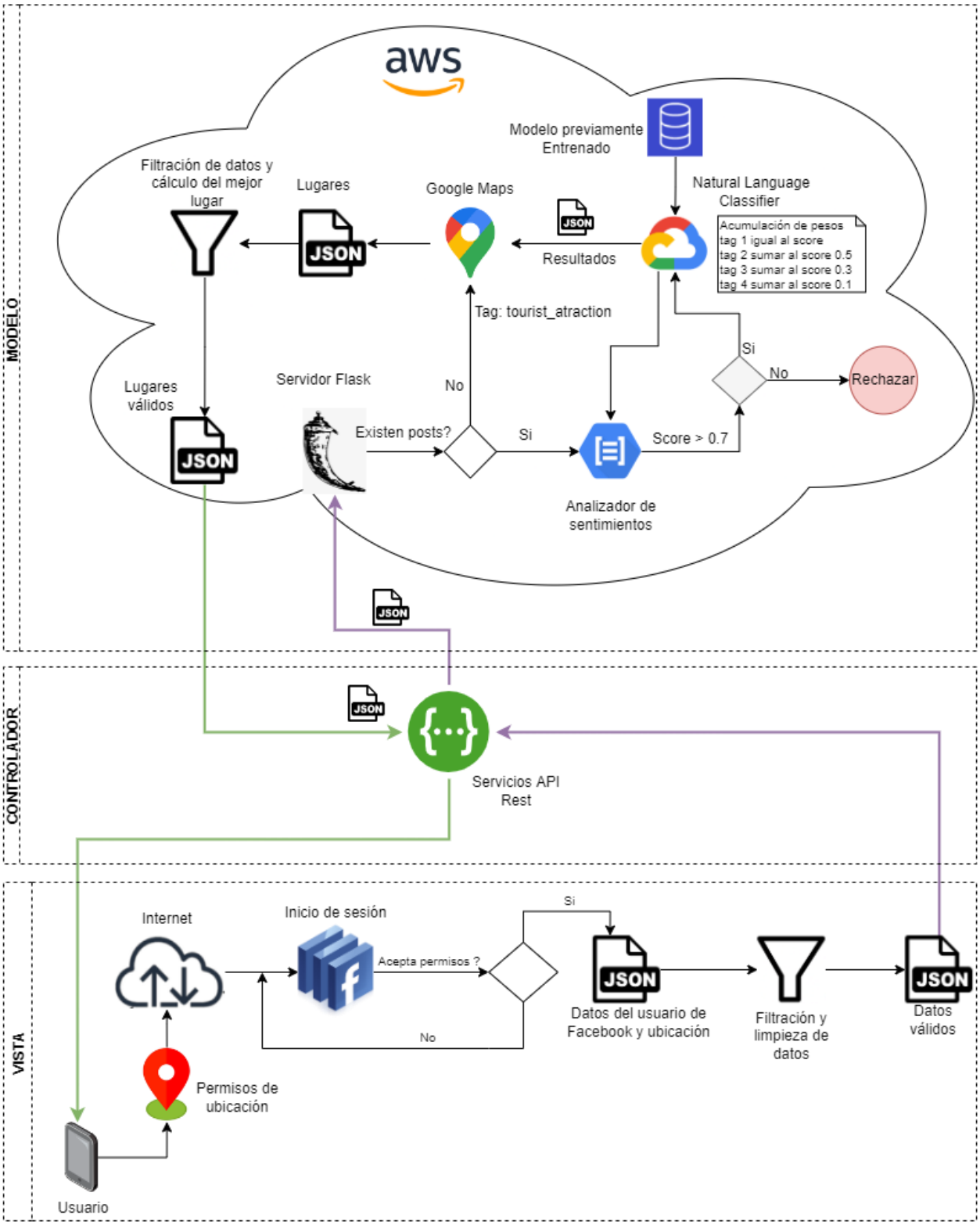

Figura 5. Arquitectura del proyecto

Fuente: elaboración propia

3.1.4 Product Backlog

Por otro lado, en la Tabla 7, se detallan las historias épicas definidas en el Sprint 0 para su posterior desarrollo.

Código	Descripción	Prioridad
TA01	Obtener la información de imágenes publicadas de un Usuario de Facebook	Alta
TA02	Implementar un servidor localizado en Amazon Web Services.	Alta
TA03	Integrar los servicios de Google Cloud en el servidor	Alta
TA04	Implementar los servicios de Google Cloud en el servidor de AWS	Alta
TA05	Desarrollar las interfaces de usuario de la aplicación móvil	Media
TA06	Integrar la API de Google Maps	Media
TA07	Realizar la comunicación entre el Frontend y el Backend	Alta
TA08	Desarrollar la aplicación móvil para probar el sistema de recomendaciones por parte del usuario final	Alta

Tabla 7. Historias épicas

Fuente: elaboración propia

La Tabla 8, muestra el desglose de las historias épicas.

Product Backlog				
Historia Épica	Historias de Usuario			
	ID	Descripción	Estimación (días)	Prioridad
TA01	TA01-01	Creación del repositorio en Github.	1	Media
	TA01-02	Creación del proyecto con Ionic	2	Media
	TA01-03	Creación de la aplicación en Facebook Developers	2	Alta
	TA01-04	Implementación de la API de Facebook.	7	Alta
TA02	TA02-01	Selección del servicio AWS	8	Alta
	TA02-02	Implementación de Framework Flask	3	Alta

TA03	TA03-01	Estudiar las categorías aceptadas para realizar la búsqueda en Google Maps	1	Media
	TA03-02	Preparar los datos obtenidos con las categorías en la parte de recolección de datos para el NLC	2	Media
	TA03-03	Entrenar los datos en el NLC de Google obtenidos en la historia de usuario TA03-02	1	Media
	TA03-04	Realizar pruebas en el modelo generado por NLC	0.5	Media
TA04	TA04-01	Implementar el analizador de sentimientos	1.5	Alta
	TA04-02	Implementar el Natural Language Classifier en el servidor de AWS	1.5	Alta
	TA04-03	Realizar peticiones HTTP al servidor de AWS para verificar los resultados de los servicios implementados	1	Alta
TA05	TA05-01	Diseñar los mockups para la aplicación móvil	1	Media
	TA05-02	Desarrollar las interfaces de usuario en base a los mockups	1.5	Alta
	TA05-03	Filtración y limpieza de los datos extraídos del usuario cuando inicia sesión con Facebook	1	Alta
TA06	TA06-01	Integración de la API de Google Maps	1	Alta
	TA06-02	Obtener los mejores lugares turísticos dependiendo la categoría analizada por el NLC	3	Alta
TA07	TA07-01	Crear servicios para enviar los datos del usuario al BackEnd mediante HTTP	2	Alta
	TA07-02	Configurar el servidor para procesar solicitudes con contenido JSON	3	Media
TA08	TA08-01	Solicitar la ubicación actual del usuario antes de iniciar sesión	4	Alta
	TA08-02	Desarrollar la interfaz de usuario para mostrar resultados	9	Media

Tabla 8. Product Backlog del proyecto

Fuente: elaboración propia

3.1.5 Release Planning

Script	Objetivo	Historias de usuario	Tiempo estimado (días)
Script 1	Construir una aplicativo Android que recopile los datos de un usuario cuando inicie sesión con Facebook	6	24
	Levantar el servidor en AWS con reglas de transferencia de archivos y tráfico HTTP		
Sprint 2	Entrenar el modelo de categorías de lugares turísticos en NLC	8	12
	Implementar los servicios de Google en el servidor AWS		
	Desarrollar las interfaces de usuario		
	Filtrar y limpiar los datos extraídos del usuario		
Sprint 3	Integrar Google Maps en el Frontend y Backend	6	22
	Realizar la comunicación entre el Frontend y el Backend		
	Obtener los mejores lugares turísticos y enviarlos al aplicativo móvil		
	Desarrollar la aplicación móvil para mostrar los puntos de interés		

Tabla 9. Definición de los Sprints

Fuente: elaboración propia

3.2 Sprint 1 (07/11/2021) dos reuniones

3.2.1 Sprint Planning

Objetivo del Sprint

- Construir una aplicación nativa para Android que recopile la información personal y las publicaciones de un usuario cuando inicie sesión con Facebook.
- Realizar el levantamiento del servidor en AWS que contenga reglas de transferencia de archivos y tráfico HTTP.

Historias de Usuario

En la Tabla 10, se encuentran listadas las historias de usuario que serán desarrolladas para el Sprint 1 indicando la estimación en días y la prioridad de cada historia.

ID	Descripción	Estimación (días)	Prioridad
TA01-01	Creación del repositorio en Github.	1	Media
TA01-02	Creación del proyecto con Ionic	2	Media
TA01-03	Creación de la aplicación en Facebook Developers.	2	Alta
TA01-04	Implementación de la API de Facebook.	7	Alta
TA02-01	Selección del servicio AWS	8	Alta
TA02-02	Implementación de Framework Flask	3	Alta

Tabla 10. Historias de Usuario del Sprint 1

Fuente: elaboración propia

Historia de Usuario		TA01-01
Nombre: Creación del repositorio en Github		
Usuario: Usuario en general		
Prioridad en negocio: Media		
Estimación: 2		Iteración asignada: 1
Programador Responsable: Jonathan Vargas, Roger Laza		
Descripción: Como desarrollador quiero crear un repositorio en Github para subir los archivos del proyecto		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe tener un control de versiones del código fuente	Se escriba el código fuente del Backend y del Frontend	Se crea un repositorio en Github con las carpetas para alojar el código fuente del FrontEnd y del BackEnd

Tabla 11. Historia de Usuario, requisito TA01-01.

Fuente: elaboración propia

Historia de Usuario		TA01-02
Nombre: Creación del proyecto con Ionic		
Usuario: Usuario en general		
Prioridad en negocio: Media		
Estimación: 3		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador quiero crear un proyecto para el Frontend utilizando el Framework Ionic.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
El usuario debe tener una aplicación móvil instalada en su dispositivo	El usuario quiera obtener sus puntos de interés	Se crea una aplicación nativa con Ionic, Angular y Capacitor

Tabla 12. Historia de Usuario, requisito TA01-02.

Fuente: elaboración propia

Historia de Usuario		TA01-03
Nombre: Creación de la aplicación en Facebook Developers		
Usuario: Usuario en general		
Prioridad en negocio: Media		
Estimación: 3		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador quiero crear una aplicación en Facebook Developers para el inicio de sesión.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
El usuario debe iniciar sesión con Facebook	El usuario ingresa a la aplicación y se muestra la pantalla de login	Se crea una aplicación de Facebook, se realiza la configuración en la aplicación móvil para el inicio de sesión

Tabla 13. Historia de Usuario, requisito TA01-03.

Fuente: elaboración propia

Historia de Usuario		TA01-04
Nombre: Implementación de la API de Facebook.		
Usuario: Usuario en general		
Prioridad en negocio: Media		
Estimación: 8		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador quiero implementar la API de Facebook para que el usuario pueda iniciar sesión.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se necesita la información personal del usuario	El usuario ingresa a la aplicación e inicia sesión con Facebook	Se obtiene como resultado un objeto JSON con los datos personales y las publicaciones del usuario de Facebook.

Tabla 14. Historia de Usuario, requisito TA01-04.

Fuente: elaboración propia

Historia de Usuario		TA02-01
Nombre: Selección del servicio AWS.		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 21		Iteración asignada: 1
Programador Responsable: Roger Laza		
Descripción: Como desarrollador necesito un servicio de AWS que me permita tener un servidor virtual en la nube		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe contratar los productos de AWS	La aplicación necesite realizar la lógica de negocio	Se obtiene un producto de AWS que permita realizar instalaciones, configuraciones y un entorno para desarrollar scripts

Tabla 15. Historia de Usuario, requisito TA02-01.

Fuente: elaboración propia

Historia de Usuario		TA02-02
Nombre: Implementación de Framework Flask.		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 3		Iteración asignada: 1
Programador Responsable: Roger Laza		
Descripción: Como desarrollador quiero realizar peticiones HTTP al servidor		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe integrar el Framework Flask	La aplicación móvil necesite realizar peticiones HTTP	Se implementa el servidor Flask que permita manejar el tráfico HTTP

Tabla 16. Historia de Usuario, requisito TA02-02.

Fuente: elaboración propia

Sprint Backlog

De acuerdo con la explicación del procedimiento, en la Tabla 17, se definen las tareas asignadas a cada historia de usuario para el Sprint 1.

Sprint Backlog	
Historia de usuario	**Tareas**
TA01-01	Creación de un repositorio de carácter privado y con el nombre TouristApp en GitHub.
	Creación de la carpeta TouristFront para la lógica del cliente y la carpeta TouristBack para la lógica del negocio
	Clonación del repositorio en el equipo local mediante HTTPS
TA01-02	Instalación de NodeJs, NPM y el Framework Ionic
	Creación del proyecto con el Framework Ionic utilizando como Framework de desarrollo Angular y la integración de Capacitor.
	Creación de la aplicación nativa Android
TA01-03	Creación de la aplicación de Facebook para el inicio de sesión y extracción de datos
	Configuración de la aplicación Android para el inicio de sesión con Facebook
TA01-04	Instalación de un plugin de la comunidad de capacitor para el inicio de sesión nativo de Facebook.
	Definición los permisos de Facebook
	Extracción de datos personales y publicaciones del usuario que inicio sesión.
TA02-01	Selección del producto Amazon Web Service, en base al presupuesto realizado en la Anexo IV
	Selección del Sistema Operativo.
	Generación de reglas para la conexión local y transferencia de archivos
TA02-02	Instalación del Framework Flask para la creación del servidor
	Creación del script para la ejecución del servidor
	Definir el proceso de respuestas a una solicitud HTTP

Tabla 17. Sprint Backlog del Sprint 1

Fuente: elaboración propia

3.2.2 Ejecución del Sprint

A continuación, se muestra la implementación de cada una de las historias de usuario y se detallan las más relevantes realizadas en Sprint 1.

Generación de repositorio en Github

Para hacer uso de GitHub es necesario crear una cuenta con un nombre de usuario, el correo electrónico y una contraseña. Se puede hacer uso de un correo personal o un institucional.

En GitHub [18] se procede a crear un repositorio privado con el nombre TouristApp con la licencia Apache-2.0 y con el archivo README en la raíz. Dentro del repositorio se crea las siguientes carpetas:

- TouristFront: para la lógica del cliente
- TouristBack: para la lógica de negocio.

Se debe invitar a los desarrolladores que colaboran en el desarrollo del proyecto por medio de una invitación por correo electrónico. Es necesario clonar el repositorio creado en su equipo personal mediante HTTPS (Hypertext Transfer Protocol Secure) y con la herramienta Gitkraken [16].

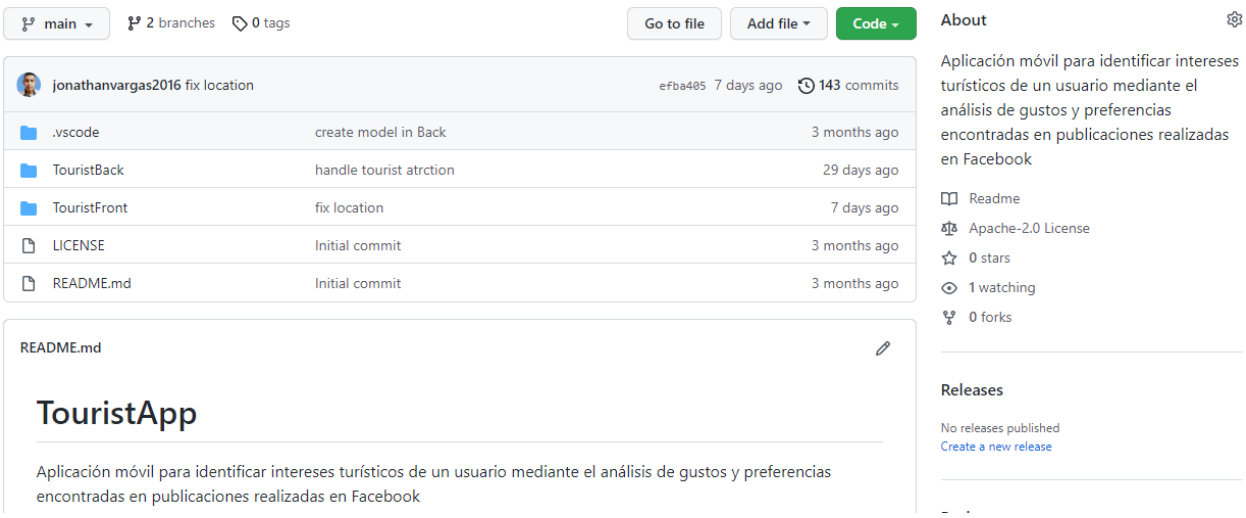

Figura 6. Repositorio TouristApp

Fuente: elaboración propia

Creación del proyecto con Ionic

Ionic [25] es un framework que permite crear aplicaciones nativas para web, móviles y de escritorio multiplataforma. Para crear el proyecto del Frontend se procede a instalar Node.js [27] descargando el ejecutable LTS (Long-term support) desde su página oficial, se instala también el NPM de manera global ejecutando el siguiente comando **npm install -g npm** en el Command Prompt (CMD) de Windows y finalmente se instala Ionic de manera global ejecutando el comando **npm install -g @ionic/cli** en el CMD.

Para crear el proyecto es necesario revisar la documentación oficial de Ionic [50] ya que muestra de manera detallada y continua los pasos necesarios para la creación. En el CMD se procede a crear un proyecto ejecutando el comando **ionic start**, se mostrará opciones como: el tipo de plantilla, el Framework que usará y si desea integrar capacitor. Para la plantilla se escoge Side Menu, para el Framework de desarrollo se selecciona Angular y se confirma la integración del capacitor. Por defecto Ionic corre el puerto 8100 en el navegador.

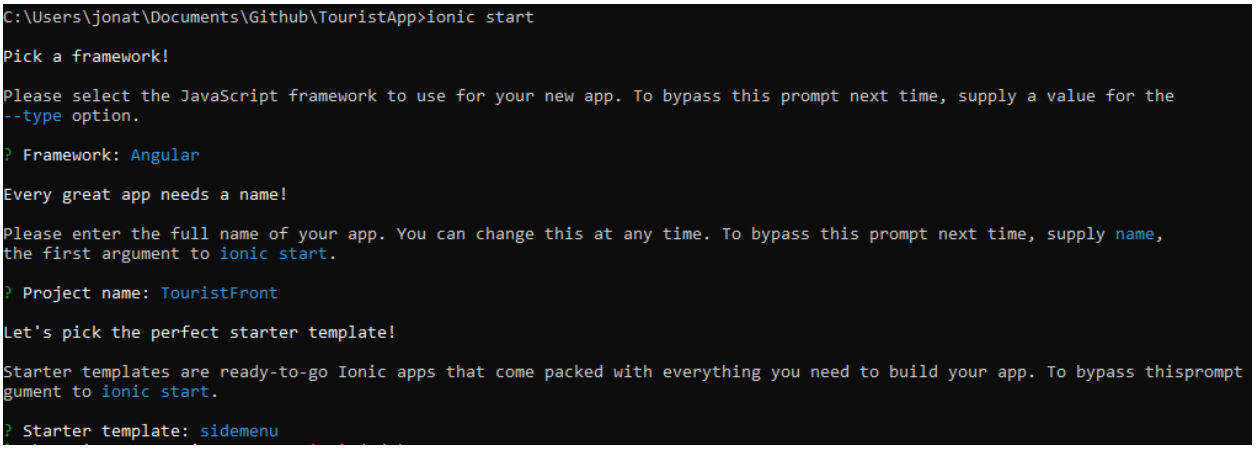

Figura 7. Creación del proyecto con Ionic

Fuente: elaboración propia

Para el desarrollo del código del Frontend se utiliza la herramienta WebStorm [15] porque está más enfocado al desarrollo de aplicaciones móviles y web. Entonces se procede a crear una página con el nombre Home y un componente con el nombre Login siguiendo la documentación oficial de Ionic [51] y escribir el código HTML y darle los respectivos estilos para tener un componente como se muestra en la Figura 8. Finamente, se crea la aplicación nativa Android y todos estos cambios deben ser subidos al repositorio utilizando los comandos Git [17] con la Herramienta Gitkraken [16].

Figura 8. Componente Login

Fuente: elaboración propia

Creación de la aplicación de Facebook para el inicio de sesión.

Para crear una aplicación de Facebook es necesario revisar la documentación oficial [52]. Se debe acceder a la página web de Facebook Developers e iniciar sesión con las credenciales de la cuenta de Facebook. En la parte de **Mis apps** se debe crear una nueva aplicación con el nombre TouristApp y se selecciona el Tipo Negocios. Una vez creado la aplicación se debe ir llenado los datos solicitados como el correo electrónico, el enlace de la política de privacidad, el icono, la categoría y la Información de contacto del delegado de protección de datos. También se debe agregar la plataforma Android y colocar el nombre de paquetes com.touristapp.ec y el nombre de la clase definidos en la aplicación Android. En la Figura 9, se puede observar la aplicación de Facebook creada.

Figura 9. Creación de Aplicación de Facebook

Fuente: Facebook Developer [52]

Integración de la API de Facebook

En el proyecto Frontend se procede a instalar la siguiente librería **capacitor-community/facebook-login** que nos permite declarar los permisos de Facebook, extraer los datos del usuario, el inicio y cierre de sesión [53]. Se empieza definiendo el identificador de la aplicación y el token del cliente de Facebook en la aplicación de Android. El script que se muestra en la Figura 10, permite que el usuario pueda iniciar sesión y aceptar los siguientes permisos:

- user_hometown
- user_location
- user_likes
- user_photos
- user_posts
- email

```
async signIn(): Promise<void> {
  const FACEBOOK_PERMISSIONS = [
    'user_hometown',
    'user_location',
    'user_likes',
    'user_photos',
    'user_posts',
    'email',
    'user_birthday'
  ];
  const result: FacebookLoginResponse = await FacebookLogin.login({
    permissions: FACEBOOK_PERMISSIONS,
  });
  if (result && result.accessToken) {
    await this.getProfile();
    await this.router.navigate(['/']);
  }
}
```

Figura 10. Código para el inicio de sesión y declaración de permisos

Fuente: elaboración propia

El script que se muestra en la Figura 11, permite extraer la información personal del usuario cuando inicia sesión.

```
async getProfile(){
    const respPost = await this.getDataPost()
    const respUser = await FacebookLogin.getProfile<{
      email: string;
      location: any;
      name: any;
      picture: any;
    }>({
      fields: ['email', 'location', 'name', 'picture{url}'],
    }).catch(() => undefined);
    if (respUser === undefined) {
      return null;
    }
    this.user_profile = {
      id: respUser.id,
      name: respUser.name,
      email: respUser.email,
      imagen: respUser.picture.data.url,
      location: respUser.location.name,
      posts: respPost,
      locationCurrent: {
        latitude: this.lat_user,
        longitude: this.lng_user,
      }
    }
  }
```

Figura 11. Código para obtener los datos personales

Fuente: elaboración propia

El script que se muestra en la Figura 12, permite extraer las publicaciones del usuario

```
async getDataPost(){
  this.user_post = []
  const respDataPost = await FacebookLogin.getProfile<{
    posts: any;
  }>({
    fields: ['posts{full_picture,reactions,message,place}'],
    }).catch(() => undefined);
  if (respDataPost === undefined) {
    return null;
  }
  respDataPost.posts.data.forEach( (post) => {
    const dataTransform = this.transformDataPost(post)
    if(dataTransform){
      this.user_post.push(dataTransform)
    }
  })
  return this.user_post;
}
```

Figura 12. Código para extraer las publicaciones

Fuente: elaboración propia

Finalmente, cuando el usuario inicie sesión desde el aplicativo los datos que se deben extraer en formato JSON son los siguientes:

- nombres completos
- enlace de la foto de perfil
- correo electrónico
- ciudad de origen
- las publicaciones que compartió o subió a Facebook

Figura 13. Datos del usuario de Facebook

Fuente: elaboración propia

Para probar el funcionamiento del inicio de sesión con Facebook en la aplicación Android/iOS se debe ejecutar los siguientes comandos de manera secuencial en la terminal del proyecto [54]:

Comando	Descripción
ionic build	permite construir los webs assets
npx cap add android/ios	permite construir la plataforma deseada
ionic cap build android/ios	permite crear la aplicación nativa para la plataforma deseada.

Tabla 18. Comandos para crear la aplicación móvil nativa

Fuente: elaboración propia

Para correr el proyecto en Android se debe utilizar la herramienta Android Studio [55], es necesario que se cree un Android Virtual Device (AVD), se instale el SDK Platform con el nivel de API 28 o superior. Otra forma de instalar la aplicación es mediante la activación de la depuración Universal Serial Bus (depuración USB) de un dispositivo Android físico. En la Figura 14, se puede observar proyecto Android creado para ser ejecutado desde Android Studio.

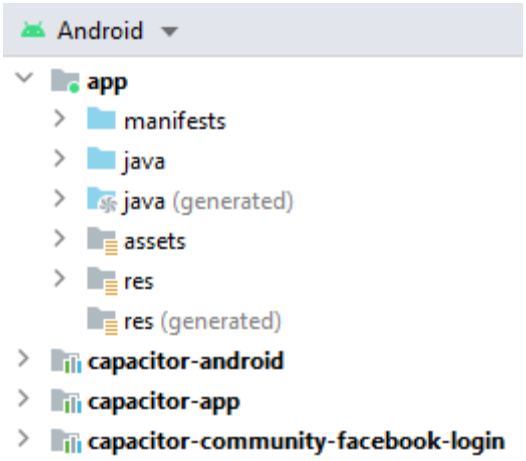

Figura 14. Proyecto Android

Fuente: elaboración propia

Amazon Web Service (AWS)

Amazon Web Service es una colección de servicios de computación en la nube pública formando una plataforma de computación en la nube que ofrece soluciones de alojamiento web en la nube a negocios, organizaciones sin fines de lucros y organismos gubernamentales, manteniendo estos sitios y aplicaciones web a bajo costo [56].

Es importante conocer el nivel gratuito de AWS para que el desarrollo del proyecto esté dentro del presupuesto indicado en el Anexo IV, dentro de los productos que ofrece AWS [57].

Producto Amazon Elastic Compute Cloud (Amazon EC2) tiene la capacidad de cómputo variable, y esta ofrece más de 475 instancias y la posibilidad de escoger el procesador, almacenamiento, redes, sistema operativo y modelo de compra más reciente para que se ajuste a las necesidades de la carga de trabajo [58].

Selección del Sistema Operativo

Como se ha indicado, Amazon EC2 tiene un producto importante para este proyecto el cual se lo denomina instancia. El cual permite escoger en una amplia variedad de Imágenes, las cuales incluye sistemas operativos como Amazon Linux, Red Hat Enterprise Linux, Ubuntu, Microsoft Server.

En este caso para mantenerse en la capa gratuita y por tener más experiencia con distribuciones de Linux por parte del equipo de desarrollo Scrum se ha optado por utilizar el sistema operativo Ubuntu Server 20.04 [59].

Lanzamiento de la instancia

Para el lanzamiento de la instancia se debe realizar los siguientes pasos indicados en la documentación oficial [60].

Una vez realizados estos pasos aparecerán detalles que indicarán cómo configurar un par de claves para la instancia y se usará el par de claves para conectarse mediante Secure Shell (SSH) con la instancia, de forma que pueda ejecutar comandos en el servidor.

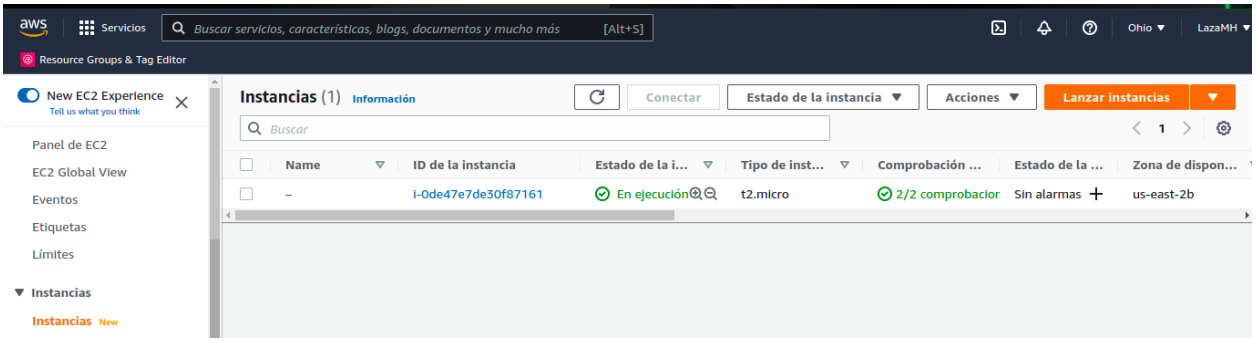

Figura 15. Lanzamiento de instancia

Fuente: instancia EC2 [60]

Generación de grupos de seguridad (Reglas de entrada y salida)

Este apartado es muy importante para definir un grado de seguridad en el servidor, las reglas de entrada y salida deben contener lo necesario para que el servidor reciba y envíe solo información necesaria.

Reglas de Entrada

- SSH: El protocolo Secure Shell permite el inicio de sesión remoto seguro desde una computadora a otra, además protege las comunicaciones con un cifrado fuerte y proporciona métodos de transferencia de archivos inseguros [61].
- HTTPS: para una conexión segura entre el servidor y el cliente

Reglas de Salida

- SSH: El protocolo Secure Shell permite el inicio de sesión remoto seguro desde una computadora a otra, además protege las comunicaciones con un cifrado fuerte y proporciona métodos de transferencia de archivos inseguros [61].
- HTTPS: para una conexión segura entre el cliente y el servidor

Conexión a la instancia

En cualquier línea de comando ya sea en una terminal o una consola de Windows se localizará el archivo touristapp.pem, el cual es la llave que permitirá realizar la conexión a la instancia y mediante el siguiente comando.

ssh -i "touristapp.pem" ubuntu@ec2-18-191-244-36.us-east-2.compute.amazonaws.com

Tabla 19. Comando para conectarse a AWS

Fuente: elaboración propia

Una vez en la línea de comando se utilizará el comando neofetch como se muestra en la Figura 16, para comprobar su funcionamiento y conocer las características del sistema operativo instalado.

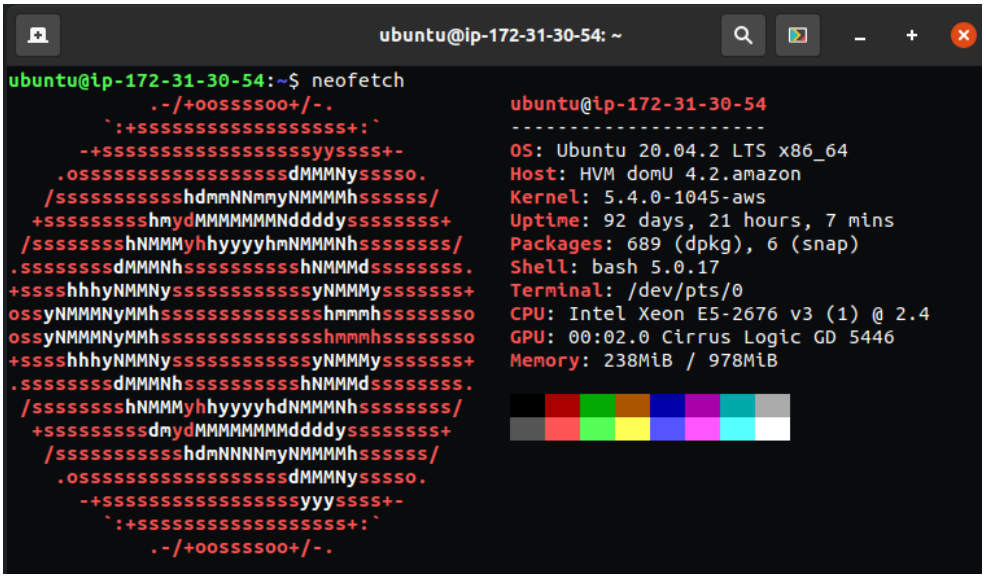

Figura 16. Línea de comando Sistema operativo en Amazon Web Service

Fuente: elaboración propia

Instalación del Framework Flask

Se procede a realizar la instalación del Framework Flask pero, para no influir en la arquitectura del servidor se aislará estas herramientas en un entorno virtual el cual llevará por nombre flask, para esto, utilizará la librería virtualenv [62]. Una vez creado el entorno virtual se lo activa y se realiza la instalación de la librería flask [63].

Para facilitar la edición del código se utilizará Visual Studio Code en una máquina local y por medio del The Secure Copy Protocol, o SCP un protocolo de red de transferencia de archivos se trasladará los archivos al servidor de AWS con el siguiente comando.

scp -i {file.pem} {path/file} ubuntu@ec2-18-117-129-130.us-east-2.compute.amazonaws.com: {path}

Tabla 20. Comando para transferir archivos a AWS

Fuente: elaboración propia

Para subir el script realizado en Python al servidor, en la documentación de Flask se indica la manera en la cual el script debe estar desarrollado como se muestra en la Figura 17.

```python
from flask import Flask, jsonify, request, Response, json
from bson import json_util
from bson.objectid import ObjectId
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
app.secret_key = 'myawesomesecretkey'

@app.errorhandler(404)
def not_found(error=None):
    message = {
        'message': 'Resource Not Found ' + request.url,
        'status': 404
    }
    response = jsonify(message)
    response.status_code = 404
    return response

@app.route("/")
def home_page():
    return '<h1> Bienvenido </h1>'

if __name__ == "__main__":
    app.run(host='0.0.0.0',debug=True, port=3000)
```

Figura 17. Script del servidor Flask app.py

Fuente: elaboración propia

Mediante la herramienta Postman se realiza una petición y se comprueba que la respuesta otorgada por el servidor sea recibida por el cliente.

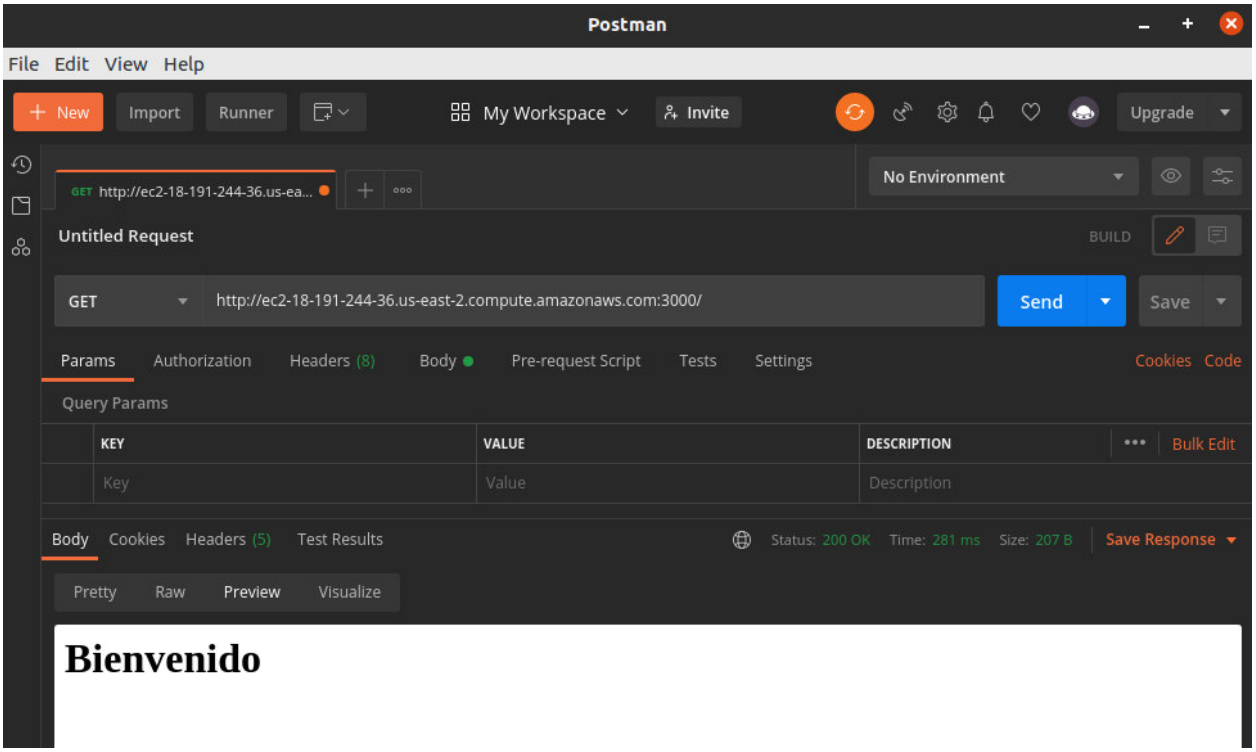

Figura 18. Petición mediante Postman

Fuente: Postman [20]

3.2.3 Sprint Review

El objetivo de este primer sprint se cumplió, aunque se tuvo ciertos obstáculos que se detallan a continuación:

- Inicio de sesión con Facebook: este tipo de error Invalid key hash como se visualiza en la Figura 19, aparece cuando la aplicación de Facebook está en modo desarrollo y el usuario inicia sesión desde un aplicativo móvil. Para resolver este inconveniente se debe agregar la key hash que aparece en la pantalla en la aplicación de Facebook en la plataforma Android en la parte de Hashes de clave.

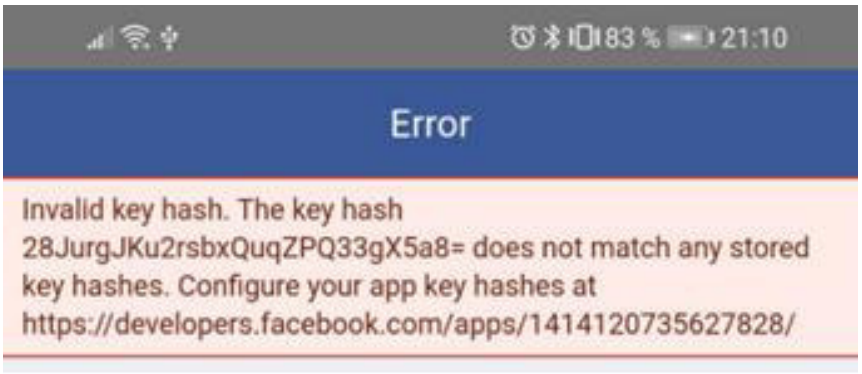

Figura 19. Error Hash key de inicio de sesión

Fuente: elaboración propia

- Uno de los principales problemas para la contratación de productos de AWS fue la obtención de una tarjeta de crédito debido a que AWS, aunque tenga plan gratuito, es obligatorio enlazar la cuenta con una tarjeta de crédito.

Pruebas de aceptación

Luego de realizar el trabajo del primer Sprint, en la Tabla 21, se puede constatar la verificación de cumplimiento con respecto a todos los criterios de aceptación de las historias de usuario.

Historia de usuario	Dado que	Cuando	Entonces	Estado
TA01-01	Se debe tener un control de versiones del código fuente	Se escriba el código fuente del Backend y del Frontend	Se crea un repositorio en Github con las carpetas para alojar el código fuente del Frontend y del Backend	Terminado
TA01-02	El usuario debe tener una aplicación móvil instalada en su dispositivo	El usuario quiera obtener sus puntos de interés	Se crea una aplicación nativa con Ionic, Angular y capacitor	Terminado
TA01-03	El usuario debe iniciar sesión con Facebook	El usuario ingresa a la aplicación y se muestra la pantalla de login	Se crea una aplicación de Facebook, se realiza la configuración en la aplicación móvil para el inicio de sesión	Terminado
TA01-04	Se necesita la información personal del usuario	El usuario ingresa a la aplicación e inicia sesión con Facebook	Se obtiene como resultado un objeto JSON con los datos personales y las publicaciones del usuario de Facebook.	Terminado
TA02-01	Se debe contratar los productos de AWS	La aplicación necesite realizar la lógica de negocio	Se obtiene un producto de AWS que permita realizar instalaciones, configuraciones y un entorno para desarrollar scripts	Terminado
TA02-02	Se debe integrar el Framework Flask	La aplicación móvil necesite realizar peticiones HTTP.	Se implementa el servidor Flask que permita manejar el tráfico HTTP.	Terminado

Tabla 21. Pruebas de aceptación del Sprint 1

Fuente: elaboración propia

3.2.4 Sprint Retrospective

En el Sprint 1 se indicó 17 tareas de las cuales los desarrolladores trabajaron un total de 12 días de forma paralela excepto en la primera tarea TA01-01 en la que ambos tuvieron que discutir la estructura del proyecto (Frontend y Backend), aunque los errores fueron solventados se observa en la línea Actual un detenimiento por las dificultades presentadas en el Sprint Review 3.2.3. El Sprint 1 se pudo completar en el tiempo estimado.

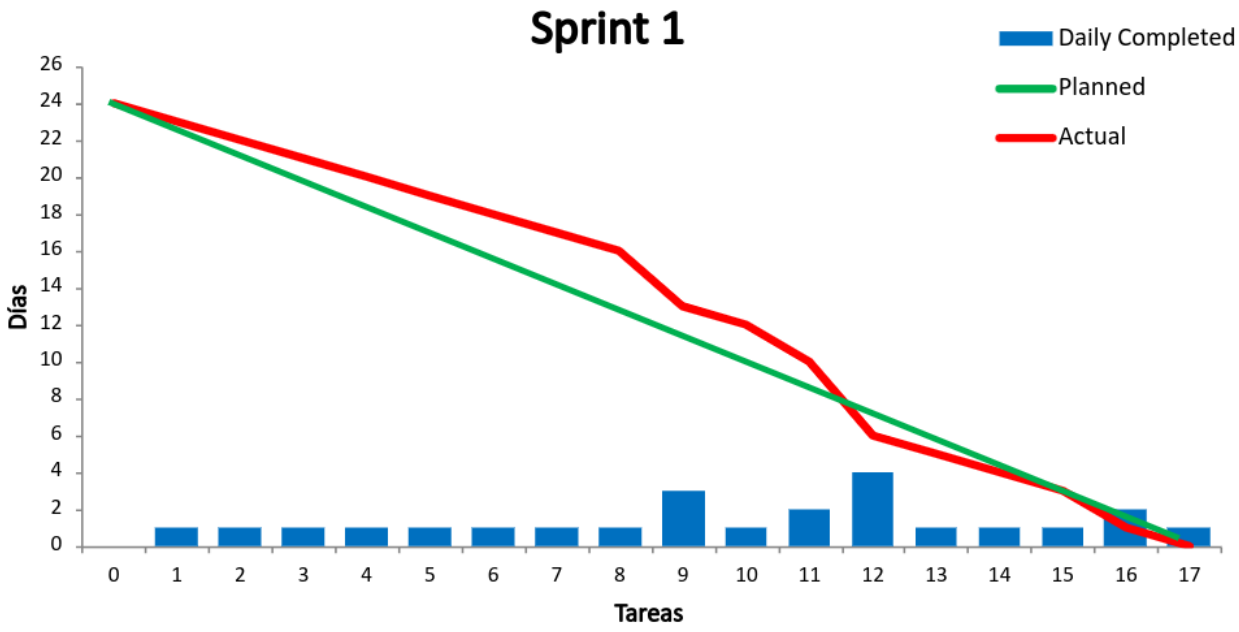

Figura 20. Burndown Chart del Sprint 1

Fuente: elaboración propia

3.3 Sprint 2 (23/11/2021)

3.3.1 Sprint Planning

Objetivo del Sprint

- Realizar el entrenamiento del modelo de categorías de lugares turísticos en Google Cloud.
- Implementar los servicios de Google en el servidor de AWS.
- Desarrollar la interfaz de usuario en base a los mockups.
- Filtrar y limpiar los datos extraídos del usuario al momento de iniciar sesión con Facebook.

Historias de Usuario

En la Tabla 22, se encuentran listadas las historias de usuario que serán desarrolladas para el Sprint 2 indicando la estimación en días y la prioridad de cada historia.

ID	Descripción	Estimación (días)	Prioridad
TA03-01	Estudiar las categorías aceptadas para realizar la búsqueda en Google Maps.	1	Media
TA03-02	Preparar los datos obtenidos con las categorías en la parte de recolección de datos para el NLC.	2	Media
TA03-03	Entrenar los datos en el NLC de Google obtenidos en la historia anterior.	1	Media
TA03-04	Realizar pruebas en el modelo generado por NLC	0.5	Media
TA04-01	Implementar el analizador de sentimientos.	1.5	Alta
TA04-02	Implementar el Natural Language Classifier en el servidor de AWS.	1.5	Alta
TA04-03	Realizar peticiones HTTP al servidor de AWS para verificar los resultados de los servicios implementados.	1	Alta
TA05-01	Diseñar los mockups para la aplicación móvil.	1	Media

TA05-02	Desarrollar las interfaces de usuario en base a los mockups.	1.5	Alta
TA05-03	Filtración y limpieza de los datos extraídos del usuario cuando inicia sesión con Facebook.	1	Alta

Tabla 22. Historias de usuario del Sprint 2

Fuente: elaboración propia

Historia de Usuario		**TA03-01**
Nombre: Estudiar las categorías aceptadas para realizar la búsqueda en Google Maps		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 3		Iteración asignada: 1
Programador Responsable: Jonathan Vargas, Roger Laza		
Descripción: Como desarrollador quiero estudiar la documentación de Google Maps Platform para entender las categorías que serán aceptadas en la búsqueda de Google Maps.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se necesita de categorías específicas que acepte Google Maps	El sistema realice la búsqueda de lugares cercanos al usuario	Se procede a realizar un estudio minucioso en la documentación oficial de Google Maps

Tabla 23. Historia de Usuario, requisito TA03-01.

Fuente: elaboración propia

Historia de Usuario	TA03-02	
Nombre: Preparar los datos obtenidos con las categorías en la parte de recolección de datos para el NLC		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 21	Iteración asignada: 1	
Programador Responsable: Jonathan Vargas, Roger Laza		
Descripción: Como desarrollador quiero preparar las frases de las publicaciones con las categorías church, museum, restaurant, park y tourist_attraction en un archivo CSV.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Recuperando datos. Espere unos segundos e intente cortar o copiar de nuevo.	Se tenga las categorías que serán aceptadas en la búsqueda de lugares por Google Maps	Se procede a realizar un estudio minucioso en la documentación oficial de Google Maps

Tabla 24. Historia de Usuario, requisito TA03-02.

Fuente: elaboración propia

Historia de Usuario	TA03-03	
Nombre: Entrenar los datos en el NLC de Google		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 21	Iteración asignada: 1	
Programador Responsable: Jonathan Vargas, Roger Laza		
Descripción: Como desarrollador quiero entrenar un modelo en NLC para clasificar contenido en un conjunto de categorías personalizado.		
Criterios de Aceptación:		
Dado que	**Cuando**	**Entonces**
Se debe entrenar el conjunto de Datos en el servicio NLC de Google	El sistema necesite realizar una predicción de una categoría según la frase que reciba	Se realiza el entrenamiento de un conjunto de datos en NLC

Tabla 25. Historia de Usuario, requisito TA03-03.

Fuente: elaboración propia

Historia de Usuario	TA03-04	
Nombre: Realizar pruebas en el modelo generado por NLC		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 21	Iteración asignada: 1	
Programador Responsable: Jonathan Vargas, Roger Laza		
Descripción: Como desarrollador quiero probar el modelo entrenado anteriormente para comprobar la predicción correcta de las categorías.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe verificar que el modelo entrenado realice la predicción correcta de la categoría	Se ingrese una frase en la consola de Natural Language Clasiffier	Entrega un objeto JSON con las categorías y sus respectivos puntajes ordenados de mayor a menor

Tabla 26. Historia de Usuario, requisito TA03-04.

Fuente: elaboración propia

Historia de Usuario	TA04-01	
Nombre: Implementar el analizador de sentimientos.		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 21	Iteración asignada: 1	
Programador Responsable: Roger Laza		
Descripción: Como desarrollador, necesito Implementar el analizador de sentimientos		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe integrar el análisis de sentimiento de los servicios de Google Cloud	El sistema necesite enviar un mensaje del post de un usuario obtenido de Facebook	Se recibe una respuesta con un score indicando si el mensaje fue positivo o negativo, entonces si el score es menor a 0.7 se rechaza el post

Tabla 27. Historia de Usuario, requisito TA04-01.

Fuente: elaboración propia

Historia de Usuario	TA04-02
Nombre: Implementar el Natural Language Classifier en el servidor de AWS	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 21	Iteración asignada: 1
Programador Responsable: Roger Laza	
Descripción: Como desarrollador, necesito implementar el Natural Language Classifier en el servidor de AWS	
Criterios de Aceptación	

Dado que	Cuando	Entonces
Se debe integrar el Natural Language Classifier de Google Cloud	El sistema necesite enviar un mensaje con el nombre del lugar obtenido del post del usuario de Facebook.	Se recibe una respuesta con la magnitud de predicción de la categoría del lugar acumulando el valor para obtener la preferencia de usuario

Tabla 28. Historia de Usuario, requisito TA04-02.

Fuente: elaboración propia

Historia de Usuario	TA04-03
Nombre: Realizar peticiones http al servidor de AWS para verificar los resultados de los servicios implementados	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 3	Iteración asignada: 1
Programador Responsable: Roger Laza	
Descripción: Como desarrollador, necesito realizar peticiones HTTP al servidor de AWS para verificar los resultados de los servicios implementados	
Criterios de Aceptación	

Dado que	Cuando	Entonces
El servidor debe procesar peticiones HTTP	El sistema debe solicitar peticiones a los servicios de Google Cloud	Se recibe las respuestas de Google Cloud y son configuradas para ser enviadas a una solicitud al Servidor

Tabla 29. Historia de Usuario, requisito TA04-03.

Fuente: elaboración propia

Historia de Usuario		TA05-01
Nombre: Diseño de los mockups para la aplicación móvil		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 5		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador quiero diseñar los mockups para la aplicación móvil		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe desarrollar los mockups de la aplicación móvil	Se necesite un prototipo de la aplicación móvil	Se desarrolla los mockups utilizando NinjaMock.

Tabla 30. Historia de Usuario, requisito TA05-01.

Fuente: elaboración propia

Historia de Usuario		TA05-02
Nombre: Desarrollar las interfaces de usuario en base a los mockups		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 8		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador necesito desarrollar las interfaces de usuario en base a los mockups creados anteriormente		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe desarrollar las interfaces de usuario en base a los mockups	El usuario ingrese al sistema y navega por los diferentes componentes y paginas	Se desarrolla las interfaces de usuario en Ionic en base a los mockups desarrollados

Tabla 31. Historia de Usuario, requisito TA05-02.

Fuente: elaboración propia

Historia de Usuario		TA05-03
Nombre: Filtración y limpieza de los datos extraídos del usuario cuando inicia sesión con Facebook		
Usuario: Usuario en general		
Prioridad en negocio: Alta		
Estimación: 8		Iteración asignada: 1
Programador Responsable: Jonathan Vargas		
Descripción: Como desarrollador necesito filtrar y limpiar los datos extraídos del usuario cuando inicia sesión con Facebook en base a los criterios especificados por el Product Owner.		
Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
Se debe probar el funcionamiento de las interfaces de usuario para la plataforma Android	El usuario instala la aplicación en su dispositivo Android y comprueba su funcionalidad y usabilidad	Se construye la aplicación para Android y se la ejecuta en Android Studio y se comprueba que las interfaces funcionan según lo esperado

Tabla 32. Historia de Usuario, requisito TA05-03.

Fuente: elaboración propia

Sprint Backlog

De acuerdo con la explicación del procedimiento, en la Tabla 33, se definen las tareas asignadas a cada historia de usuario para el Sprint 2.

Sprint Backlog	
Historia de usuario	**Tareas**
TA03-01	Escoger las categorías las cuales serán utilizadas para predecir los mensajes de los posts de usuario.
TA03-02	Cambiar las etiquetas anteriores con las aceptadas por Google Map.
	Obtener nuevos datos para las etiquetas nuevas.
	Escribir el archivo CSV para el entrenamiento.
TA03-03	Subir el archivo CSV en el servicio de Google NLC y crear las configuraciones necesarias.
	Realizar el entrenamiento y realizar una evaluación del modelo obtenido.
TA03-04	Mediante la herramienta de NLC que proporciona una consola, realizar pruebas al modelo y verificar que realice correctamente la predicción de categoría.

TA04-01	Escribir el script para analizar los sentimientos de los mensajes obtenidos de los posts de Facebook del usuario.
	Mediante el score proporcionado por el analizador de sentimientos separar los posts positivos y descartar los negativos.
	Preparar el mensaje para que el Natural Language Classifier sea más preciso con el resultado.
TA04-02	Implementación del Natural Language Classifier en el servidor de AWS.
	Escribir el script para realizar la clasificación del lugar mediante peticiones realizadas al modelo entrenado de Google Cloud.
	Implementar una respuesta con los lugares turísticos preferidos por el usuario.
TA04-03	Configurar el servidor para que pueda procesar solicitudes HTTP.
	Configurar la respuesta para la solicitud http para que sea procesada.
TA05-01	Registrarse en NinjaMock.
	Diseñar los mockups para la aplicación móvil en NinjaMock.
TA05-02	Desarrollar las interfaces de usuario en base a los mockups.
	Probar el funcionamiento de las interfaces de usuario en Android Studio.
TA05-03	Filtración de los datos de usuario.
	Limpieza de los datos de usuario.

Tabla 33. Sprint Backlog del Sprint 2

Fuente: elaboración propia

3.3.2 Ejecución del Sprint

A continuación, se muestra la implementación de cada una de las historias de usuario y se detallan las más relevantes realizadas en Sprint 2.

Estudiar las categorías aceptadas para realizar la búsqueda en Google Maps.

Para que los puntos de interés sean procesados con facilidad por medio del servicio de Google Maps es necesario estudiar la documentación oficial de Google Cloud Platform en la parte de Tipos de lugares [64] se puede observar una gran variedad de etiquetas que pueden ser analizadas. Para el desarrollo de este proyecto se tomará en cuenta las siguientes categorías indicados por el Product Owner:

- church,
- park,
- museum,

- restaurant
- tourist_attraction

La etiqueta tourist_atraction es utilizada para cuando los puntos de interés no tienen ningún resultado. Esta etiqueta será utilizada para mostrar lugares turísticos que estén alrededor del usuario.

Preparar los datos obtenidos con las categorías en la parte de recolección de datos para el NLC.

Una vez estudiado las categorías se procederá a clasificar los datos con las nuevas etiquetas, pero como se ha agregado algunas etiquetas, se ha decidido obtener oraciones que no contenga los puntos interés predefinidos, esto servirá para realizar una predicción en tourist_attraction, también se ha aumentado un nuevo conjunto de datos, revisando y extrayendo frases positivas de publicaciones de grupos en Facebook acerca de lugares turísticos.

Con los datos obtenidos de Google Form y de los grupos de lugares turísticos de Facebook es necesario clasificarlos con las nuevas etiquetas definidas, para esto se sigue la estructura definida en la Figura 4. El resultado del proceso descrito es un archivo CSV con las etiquetas definidas y con las frases que serán preparadas para ser entrenadas en el NLC.

Entrenar los datos en el NLC de Google

Para realizar el entrenamiento se necesita el archivo CSV final. Para el entrenamiento se debe ir a la consola de Google Cloud Platform y seleccionar Natural Language -> Clasificación de documentos y textos de AutoML. Con el fin de crear el modelo, se crea un nuevo conjunto de datos añadiéndole el nombre **touristapp_v3**, escogiendo la ubicación Global y seleccionando el objetivo del modelo Clasificación con una sola etiqueta.

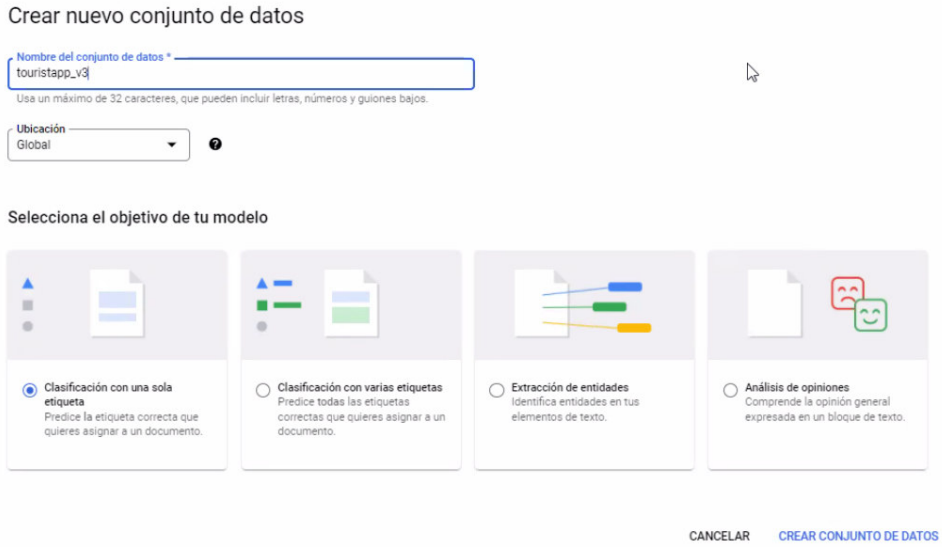

Figura 21. Creación del conjunto de datos

Fuente: Google Cloud Platform – Natural Language [65]

Una vez creado el conjunto de datos se debe importar el archivo CSV y almacenarlo en el Cloud Storage creando una nueva carpeta llamada touristapp.

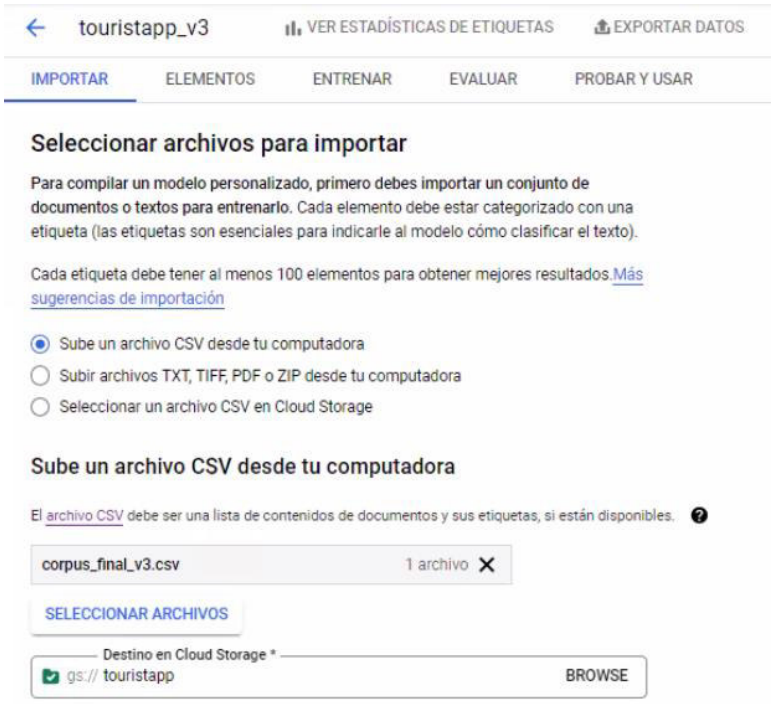

Figura 22. Importación del archivo CSV

Fuente: Google Cloud Platform – Natural Language [65]

Una vez importado el archivo CSV se verifica que no exista errores. Si existe errores en la misma consola se los corrige. Si los errores no se pueden arreglar se deberá cargar un nuevo archivo CSV.

El resultado es la cantidad total de datos, las etiquetas, la cantidad de datos para el entrenamiento, para la validación, para las pruebas y las funciones para eliminar alguna frase, asignar etiquetas o agregar nuevas etiquetas. En el conjunto de datos se obtuvo una alerta debido a que dos frases estaban repetidas, se procedió a eliminar una de ellas desde la consola.

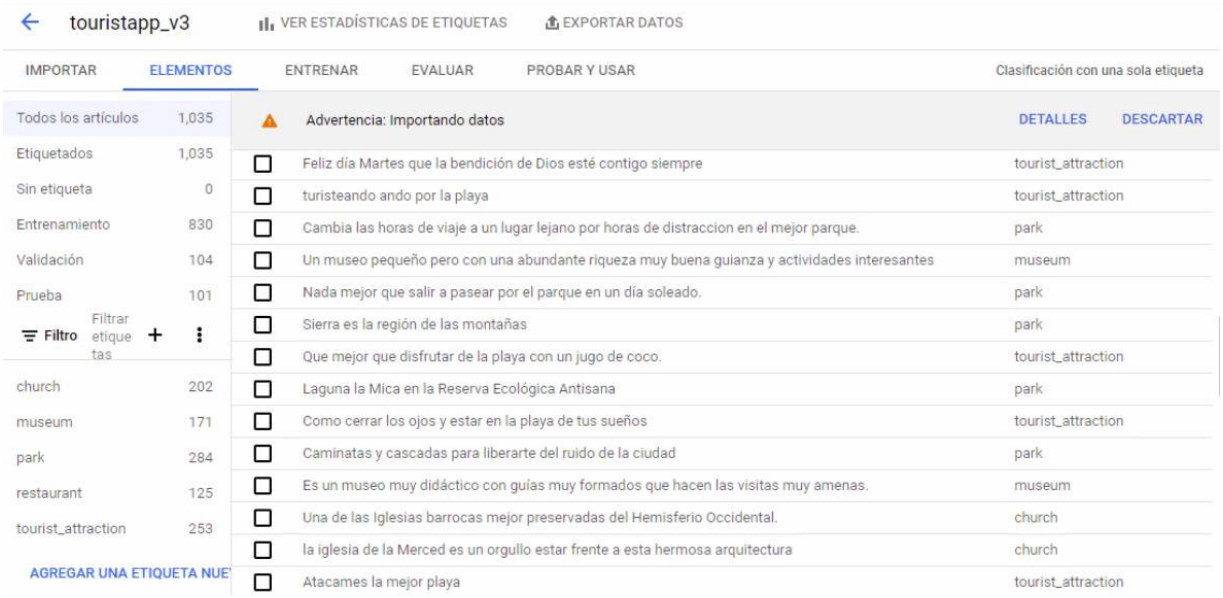

Figura 23. Verificación de errores en el modelo de datos

Fuente: Google Cloud Platform – Natural Language [65]

Una vez con el modelo verificado, se inició con el entrenamiento del modelo. Este proceso durará ciertas horas dependiendo de la cantidad de datos del archivo CSV. En nuestro modelo, el entrenamiento tuvo una duración de 8 horas.

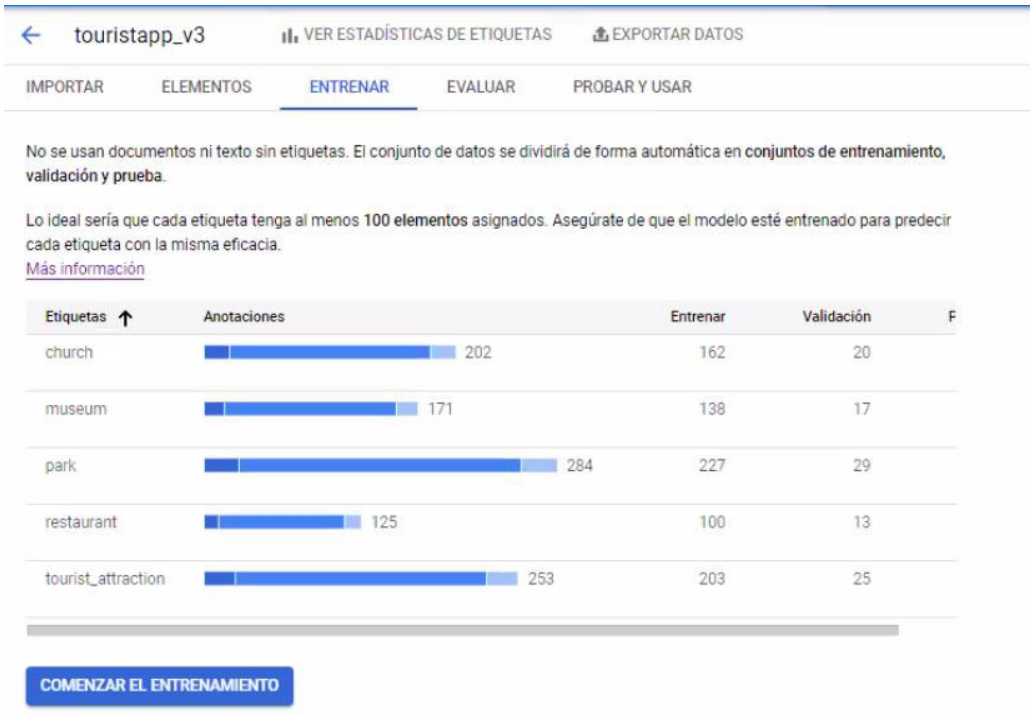

Figura 24. Comenzar el entrenamiento

Fuente: Google Cloud Platform – Natural Language [65]

Realizar pruebas en el modelo generado por NLC

Una vez finalizado el proceso de entrenamiento se ingresará en la pestaña de PROBAR Y USAR, la cual tiene un campo para ingresar texto o cargar documentos. Se ingresó el texto para realizar pruebas y como se puede ver en la Figura 25, muestra los resultados de la predicción concluyendo que la categoría “museum” tiene una probabilidad de 97.78% de pertenecer a la frase ingresada.

Figura 25. Pruebas en el modelo de datos

Fuente: Google Cloud Platform – Natural Language [65]

Implementación del analizador de sentimientos en el servidor de AWS para el analizador de sentimientos es necesario.

EL analizador de sentimientos conocido en Google como Análisis de opiniones, inspecciona el texto enviado e identifica la opinión emocional que predomina dentro del texto estas pueden ser positiva, negativa, mixta o neutra [66]. Un ejemplo de los valores obtenidos al realizar la solicitud al Analizador de opiniones genera el score presentado en la Tabla 34, donde se muestra algunos valores de muestra y la forma de cómo interpretarlos [67].

Opinión	Valores de muestra
Claramente positiva	"score": 0.8
Claramente negativa	"score": -0.6
Neutral	"score": 0.1
Mixto	"score": 0.0

Tabla 34. Valores de muestra según opiniones

Fuente: elaboración propia

Las opiniones claramente positivas y claramente negativas varían dependiendo de cada mensaje analizado y para obtener resultados más acordes al presente trabajo se ha indicado que se aceptarán los mensajes claramente positivos y se descartará el resto, para cumplir con el objetivo del proyecto.

Una parte importante acerca del analizador de opiniones es el idioma, en la solicitud de analizador de opiniones se puede enviar el parámetro, “lenguaje” el cual será obviado para que Google identifique el idioma para optimizar las respuestas, los idiomas soportados por Google [68], para este proyecto se usó el idioma en español, debido a la información recolectada en este idioma.

Para escribir el script y realizar peticiones sobre el analizador de sentimientos de Google es necesario seguir la documentación indicada en [66], para este proyecto fue necesario escribir el código en Python debido que el servidor Flask utiliza este lenguaje.

Una vez preparado los datos que serán analizados por el Analizador de sentimientos se ha implementado un umbral que será ajustado para guardar los posts con el mensaje positivo, para esto se ha indicado que en el valor en el score mayor a 0.7 que se considera un sentimiento positivo y descartando los valores que no cumplen con esta condición.

Una vez escrito el script indicado en la Figura 26, de manera local se utiliza el protocolo SCP para que el archivo sea transferido al servidor localizado en AWS con el comando descrito en la Tabla 20.

```
from google.cloud import language_v1
from natural_language.category import get_prediction
def sample_analyze_sentiment(text_content):
    client = language_v1.LanguageServiceClient()
    type_ = language_v1.Document.Type.PLAIN_TEXT

    document = {"content": text_content, "type_": type_}
    encoding_type = language_v1.EncodingType.UTF8
    response = client.analyze_sentiment(request = {'document': document, 'encoding_type': encoding_type})


    result = categories()

    for sentence in response.sentences:
        if( sentence.sentiment.score > 0.7 ):
            result = handle_category( sentence.text.content , result )

    tags = ['church', 'park' , 'museum' , 'tourist_attraction' , 'restaurant']
    new_result = {}

    for tag in tags:
        repeat = result[tag]['repeat']
        if repeat == 0 :
            repeat = 1
        new_result[tag] =  result[tag]['score'] / repeat  * 100

    return new_result
```

Figura 26. Script del analizador de sentimientos

Fuente: elaboración propia

Una vez que el script se encuentre en el servidor se procede a realizar la ejecución y realización de pruebas. En la Figura 27, se puede verificar que el umbral está funcionando correctamente tomando solo los scores positivos y descartando los demás, también agregando el idioma al resultado. Esto permitirá guardar los posts con los mensajes positivos para que sean procesado por el NLC.

String de pruebas

Comiendo hamburguesa. Odio esta iglesia muy fea. Hermoso lago que existe en el parque. Elegante restaurante Tienen que conocer. Un restaurante de baja categoría poco recomendable. Que increíble museo con mucha historia.

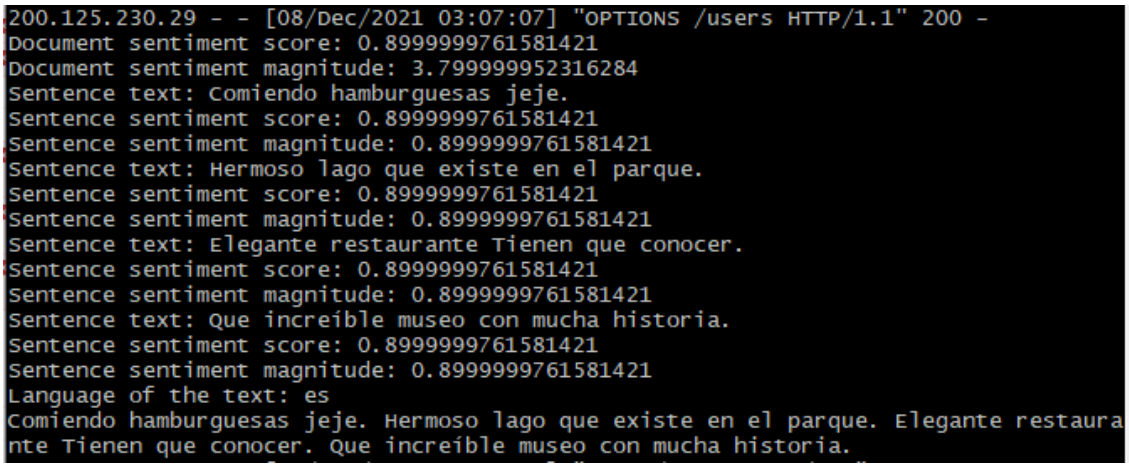

Figura 27. Resultados del analizador de sentimientos

Fuente: elaboración propia

Implementación del Natural Language Classifier (NLC)

Una vez realizado el analizador de sentimientos es momento de desarrollar el script para la integración del servicio de NLC, se deberá usar el script generado en el entrenamiento de conjuntos de datos, este script facilitado por Google esta realizado en Python.

La falencia más importante que fue encontrada al momento de implementar el script es que no existía la manera de enviar un conjunto de datos para ser analizado, es decir, solo se puede enviar un post a la vez, para solucionar este problema fue necesario crear una petición por cada post, para esto, se modificó el script de análisis de sentimiento, enviando un mensaje positivo al NLC.

Se modifica la clase de análisis de sentimiento, para agregar las funciones “handle_category” y “get_prediction”. Con la función “get_prediction” se obtiene la predicción por cada mensaje y mediante la función handle_category se implementará los pesos para que sean agregados al score de predicción y adicionalmente se agregará un contador “repeat” para colocar el número de veces que la etiqueta aparece como resultado de la predicción esto permitirá realizar una media de los datos analizados, obteniendo como resultado un porcentaje de interés por cada

categoría, estos métodos serán llamados en la función “sample_analyze_sentiment” dentro de la condición de los mensajes positivos, en la Figura 28, se indica las funciones agregadas en el analizador de sentimientos.

```
def handle_category( content  , result ):
    response = get_prediction(content)
    for i  in range(4):
        score = response.payload[i].classification.score
        result[response.payload[i].display_name]['score']  += score
        result[response.payload[i].display_name]['repeat'] += 1

        if(i==1):
            result[response.payload[i].display_name]['score'] +=0.5
        if(i==2):
            result[response.payload[i].display_name]['score'] +=0.3
        if(i==3):
            result[response.payload[i].display_name]['score'] +=0.1
    return result
```

Figura 28. Definición de las funciones para el manejo del NLC

Fuente: elaboración propia

Una vez escrito el archivo indicado en la Figura 29, de manera local se utiliza el protocolo SCP para que el archivo sea transferido al servidor localizado en AWS con el comando descrito en la Tabla 20.

```
import os
from google.api_core.client_options import ClientOptions
from google.cloud import automl_v1

def get_prediction( content ):
    options = ClientOptions(api_endpoint='automl.googleapis.com')
    prediction_client = automl_v1.PredictionServiceClient(client_options=options)
    payload = create_request( content )

    params = {}
    request = prediction_client.predict(
          name='projects/353598367974/locations/us-central1/models/TCN6317615692317196288',

          payload=payload
       )

    return request

def create_request(content):
    return {'text_snippet': {'content': content, 'mime_type': 'text/plain'} }

os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = {key/tourist-app}
```

Figura 29. Script del NLC

Fuente: elaboración propia

Con la ejecución de este script se obtiene los resultados indicados en la Figura 30, se indica los siguientes bloques de resultados, el bloque “Message” indica el mensaje que será analizado, el bloque “Sentiment” se indica el score y magnitud que tiene el mensaje es dedir si es un mensaje con sentimiento positivo o negativo, en el bloque “Category Taste Prediction” indica los tags ordenados de mayor a menor hasta la cuarta posición, en el bloque “Interested Point” se puede observar el porcentaje de cada tag analizada, es decir, el usuario tiene mayor preferencia por la categoría “restaurant” debido al valor del porcentaje, este resultado se enviará al servicio a la api de Google Map.

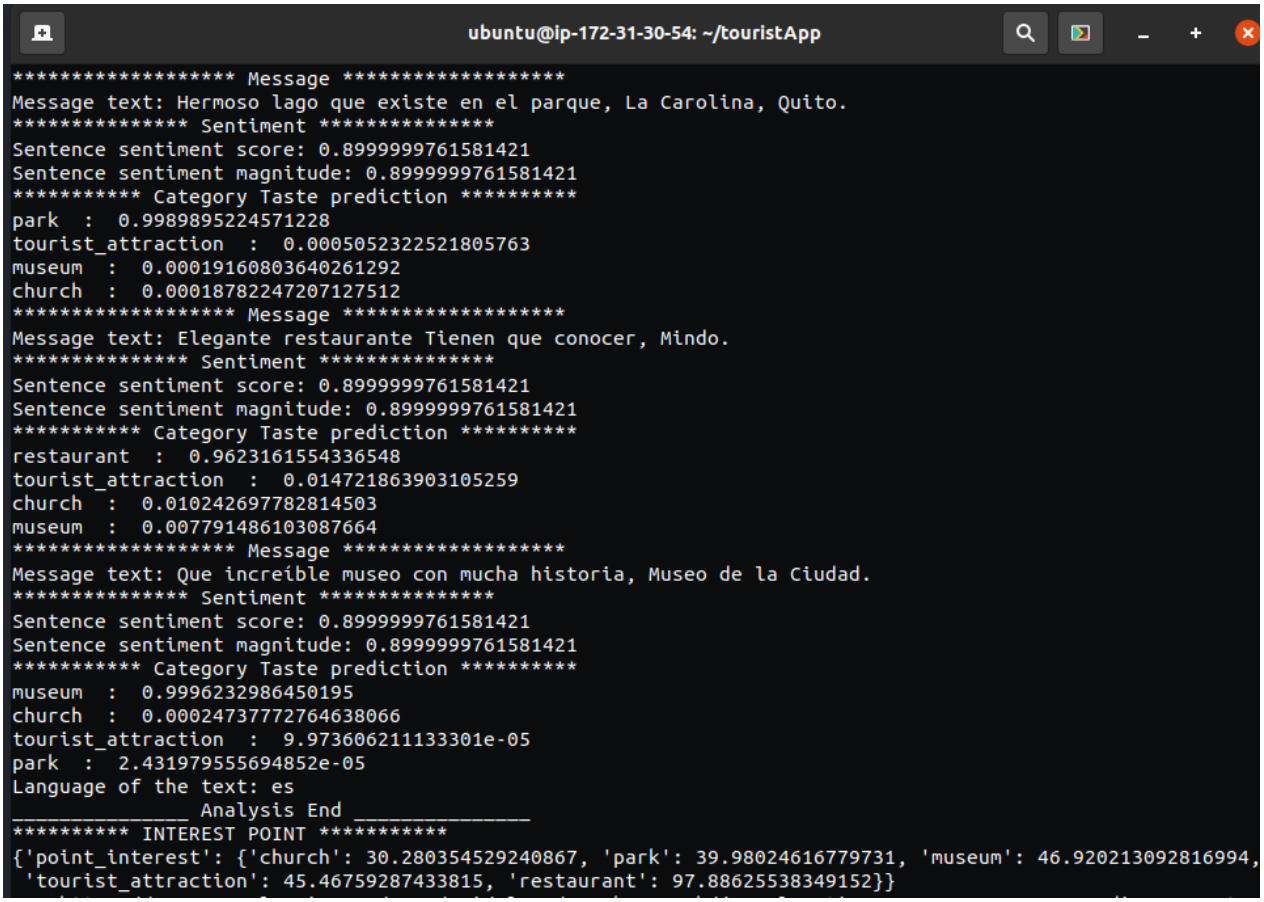

Figura 30. Resultado de la integración del script NLC

Fuente: elaboración propia

Diseño de los mockups

Para el diseño de los mockups se utilizó NinjaMock porque es una herramienta que permite crear maquetas de una manera rápida y sencilla. El nombre del proyecto es TouristApp y consta de 4 páginas (Login, Home, Menu, Ubicacion) que se pueden observar más detalladamente en el Anexo II.

Desarrollo de las interfaces de usuario

Para el desarrollo de las interfaces de usuario se utilizó los mockups (Ver Anexo II). Para crear componentes, páginas y más funciones de Angular es necesario revisar la documentación oficial de Ionic [51]. En la Figura 8, se puede observar el componente Login que tiene un botón en el centro de la pantalla que permite al usuario iniciar sesión con Facebook y recibir los datos en la aplicación que serán almacenado en el Local Storage y deberá ser redirigido a la página Home. Para el fondo de pantalla se utilizó una imagen de un lugar turístico.

En la Figura 31, se puede observar los datos del usuario almacenados en el Local Storage. Se decidió almacenar los datos en el local Storage para cuando el usuario inicia sesión y se mantiene en la aplicación. Si el usuario cierra la aplicación y no cierra sesión y después de un tiempo vuelve a abrir la aplicación el sistema verificará si los datos existen en el Local Storage para que pueda ser redirigido a la página Home caso contrario será redirigido al componente Login.

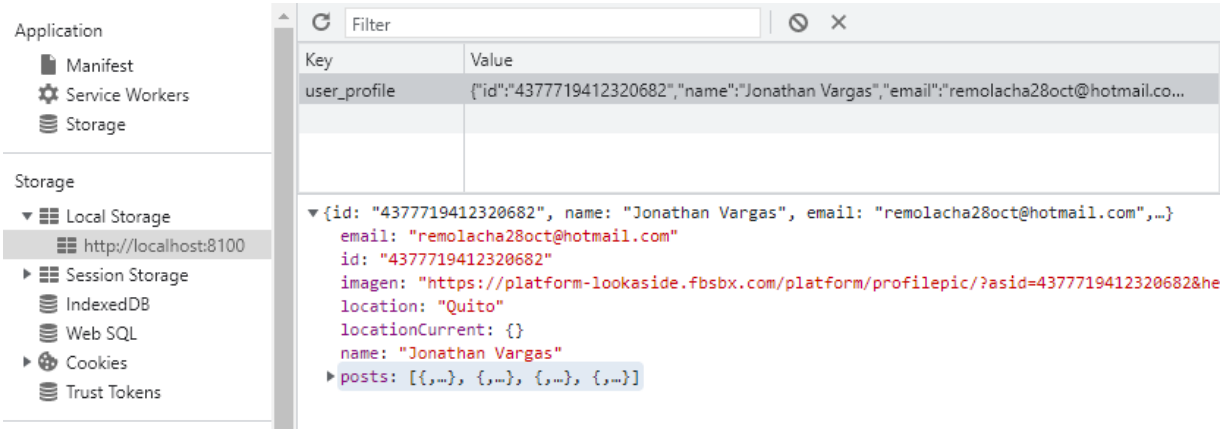

Figura 31. Datos del usuario almacenados en el Local Storage

Fuente: elaboración propia

En la Figura 32 se puede observar la página Home con un icono en la parte superior izquierda que permite abrir el Side Menu y el Toolbar con la descripción “Bienvenido a TouristApp”.

Figura 32. Interfaz de usuario Home

Fuente: elaboración propia

En la Figura 33, se puede observar la página Menu que muestra el Side Menu con la foto del usuario, sus nombres completos y el correo electrónico. Además, se implementó la opción que permite al usuario cerrar sesión.

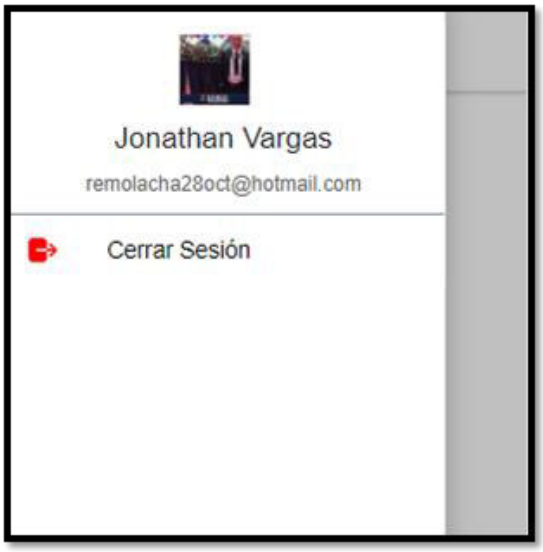

Figura 33. Interfaz de usuario Menu

Fuente: elaboración propia

En la Figura 34, se puede observar el código que permite al usuario cerrar sesión, eliminar sus datos almacenados en el Local Storage y ser redirigido al componente Login.

```
async signOut() {
  localStorage.clear();
  await this.router.navigate(['/login']);
  await FacebookLogin.logout();
}
```

Figura 34. Script para cerrar sesión

Fuente: elaboración propia

En la Figura 35, se puede observar el Modal que muestra el Toolbar con el nombre del punto de interés y un icono en la parte superior izquierda que permite al usuario regresar a la página Home.

Figura 35. Interfaz de usuario Modal

Fuente: elaboración propia

Para correr la aplicación en Android Studio [55] se debe ejecutar los comandos que se pueden observar en la Tabla 18.

Filtración y limpieza de los datos del usuario

Para el proceso de filtración y limpieza de los datos se debe extraer los datos indispensables que cumplan con los siguientes requisitos establecidos por el Product Owner:

- Frase de la publicación.
- Reacción Love o Care o Like
- Enlace de la imagen de la publicación.
- Lugar de la publicación tiene los siguientes atributos:
- Nombre del lugar
- País de donde se encuentra el lugar
- Ciudad de donde se encuentra el lugar
- Coordenadas de latitud y longitud del lugar
- Código postal del lugar
- Calle principal y secundaria del lugar

En la Figura 36, se puede observar el código que muestra el proceso de filtración de datos.

```
transformDataPost(post: any){
  if(post.place && post.message && post.full_picture){
    if(post.reactions &&
      (post.reactions.data[0].type == "LIKE" ||
        post.reactions.data[0].type == "CARE" ||
        post.reactions.data[0].type == "LOVE")){
      return {
        picture: post.full_picture,
        reactions: post.reactions.data[0],
        messages: post.message,
        place: {
          name: post.place.name? post.place.name: null,
          location:  {
            country: post.place.location.country,
            city: post.place.location.city,
            latitude: post.place.location.latitude,
            longitude: post.place.location.longitude,
            street: post.place.location.street ? post.place.location.street : null,
            zip: post.place.location.zip ? post.place.location.zip : null
          }
        }
      }
    }
  }
```

Figura 36. Script para filtrar los datos

Fuente: elaboración propia

Algunos datos extraídos pueden no tener un valor, por ello es necesario que se le agregue el valor de Null o Undefined:

- Reacción puede ser Like, Care, Love, Null o Undefined
- Nombre del lugar de la publicación puede ser Null
- Las calles del lugar de la publicación pueden ser Null
- El código postal de lugar de la publicación puede ser Null

En la Figura 37, se puede observar el código que permite agregar los valores de Null o Undefined solo a los datos mencionados anteriormente.

```
return {
        picture: post.full_picture,
        reactions: null || undefined,
        messages: post.message,
        place: {
          name: post.place.name? post.place.name: null,
          location:  {
            country: post.place.location.country,
            city: post.place.location.city,
            latitude: post.place.location.latitude,
            longitude: post.place.location.longitude,
            street: post.place.location.street ? post.place.location.street : null,
            zip: post.place.location.zip ? post.place.location.zip : null
          }
        }
      }
    }
```

Figura 37. Script para agregar valores Null o Undefined

Fuente: elaboración propia

3.3.3 Sprint Review

Los objetivos del sprint 2 se cumplió según lo esperado. No se encontraron obstáculos en el entrenamiento del NLC.

Pruebas de aceptación

Luego de realizar el trabajo del Sprint 2, en la Tabla 38, se puede constatar la verificación de cumplimiento con respecto a todos los criterios de aceptación de las historias de usuario.

Historia de usuario.	Dado que	Cuando	Entonces	Estado
TA03-01	Se necesita de categorías específicas que acepte Google Maps	El sistema realice la búsqueda de lugares cercanos al usuario	Se procede a realizar un estudio minucioso en la documentación oficial de Google Maps	Terminado
TA03-02	Se debe preparar los datos con las categorías que serán aceptadas por Google Maps	Se tenga las categorías que serán aceptadas en la búsqueda de lugares por Google Maps	Se modifica el archivo CSV con las nuevas categorías añadiendo la categoría correspondiente a cada frase.	Terminado
TA03-03	Se debe entrenar el conjunto de Datos en el servicio NLC de Google	El sistema necesite realizar una predicción de una categoría según la frase que reciba.	Se realiza el entrenamiento de un conjunto de datos en NLC.	Terminado
TA04-01	Se debe integrar el análisis de sentimiento de los servicios de Google Cloud	El sistema necesite enviar un mensaje del post de un usuario obtenido de Facebook.	Se recibe una respuesta con un score indicando si el mensaje fue positivo o negativo, entonces si el score es menor a 0.7 se rechaza el post	Terminado
TA04-02	Se debe integrar el Natural Language Classifier de Google Cloud	El sistema necesite enviar un mensaje con el nombre del lugar obtenido del post del usuario de Facebook.	Se recibe una respuesta con la magnitud de predicción de la categoría del lugar acumulando el valor para obtener la preferencia de usuario	Terminado
TA04-03	El servidor debe procesar peticiones http	El sistema debe solicitar peticiones a los servicios de Google Cloud	Se recibe las respuestas de Google Cloud y son configuradas para ser enviadas a una solicitud al Servidor	Terminado
TA05-01	Se debe desarrollar los mockups de la aplicación móvil	Se necesite un prototipo de la aplicación móvil	Se desarrolla los mockups utilizando NinjaMock.	Terminado
TA05-02	Se debe desarrollar las interfaces de	El usuario ingrese al sistema y navega por los diferentes	Se desarrolla las interfaces de usuario en	Terminado

	usuario en base a los mockups	componentes y paginas	Ionic en base a los mockups desarrollados.	
TA05-03	Se debe probar el funcionamiento de las interfaces de usuario para la plataforma Android	El usuario instala la aplicación en su dispositivo Android y comprueba su funcionalidad y usabilidad.	Se construye la aplicación para Android y se la ejecuta en Android Studio y se comprueba que las interfaces funcionan según lo esperado	Terminado

Figura 38. Pruebas de aceptación del Sprint 2

Fuente: elaboración propia

3.3.4 Sprint Retrospective

En el Sprint 2 se indicó 22 tareas desarrolladas en 12 días de forma paralela, excepto para la historia de usuario TA03 en la que el equipo Scrum tuvo que realizar el entrenamiento de modelo en el Natural Language Classifier.

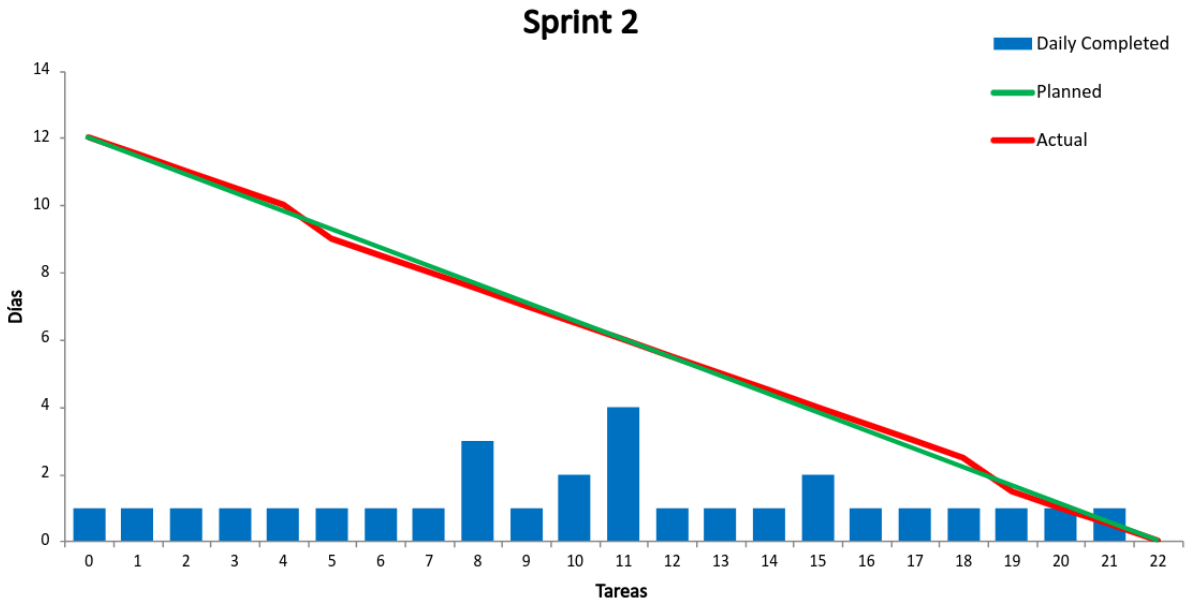

Figura 39. Burndown Chart del Sprint 2

Fuente: elaboración propia

3.4 Sprint 3

3.4.1 Sprint Planning

Objetivo del Sprint

- Integrar la API de Google Maps en el Frontend y el Backend.
- Realizar la comunicación entre el Frontend y el Backend para que puedan procesar solicitudes HTTP.
- Obtener los mejores lugares turísticos mediante los servicios de Google Maps Platform y enviarlos al aplicativo móvil.
- Desarrollar una aplicación nativa para Android que muestre los puntos de interés.

Historias de Usuario

En la Tabla 35, se encuentran listadas las historias de usuario que serán desarrolladas para el Sprint 3 indicando la estimación en días y la prioridad de cada historia.

ID	Descripción	Estimación (días)	Prioridad
TA06-01	Integración de la API de Google Maps	1	Alta
TA06-02	Obtener los mejores lugares turísticos dependiendo la categoría analizada por el NLC	3	Alta
TA07-01	Crear los servicios para enviar los datos del usuario al Backend mediante HTTP	2	Alta
TA07-02	Configurar el servidor para procesar solicitudes con contenido JSON	3	Media
TA08-01	Solicitar la ubicación actual del usuario antes de iniciar sesión	4	Alta
TA08-02	Desarrollar la interfaz de usuario para mostrar resultados	9	Media

Tabla 35. Historias de Usuario del Sprint 3

Fuente: elaboración propia

Historia de Usuario	TA06-01
Nombre: Integrar la API de Google Maps	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Jonathan Vargas	
Descripción: Como desarrollador quiero Integrar la API de Google Maps generando una API Key y realizando la configuración necesaria para implementarlo en Android	
Criterios de Aceptación	

Dado que	Cuando	Entonces
El sistema debe integrar Google Maps	El usuario ingrese a la pantalla principal (componente Home) y desee ver el mapa de Google	Se genera una API key en la Google Maps Platform y se realiza la configuración necesaria para mostrar el mapa de Google en la aplicación

Tabla 36. Historia de usuario, requisito TA06-01

Fuente: elaboración propia

Historia de Usuario	TA06-02
Nombre: Obtener los mejores lugares turísticos dependiendo la categoría analizada por el NLC	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Roger Laza	
Descripción: Como desarrollador quiero que el servidor Flask realice la predicción de los gustos de un usuario	
Criterios de Aceptación	

Dado que	Cuando	Entonces
El sistema debe obtener los mejores lugares turísticos dado una categoría	El usuario ingrese a la aplicación TouristApp y desee ver lugares turísticos dependiendo sus gustos	El sistema procesa la información obtenida por el usuario y predice cual es el punto de interés más prioritario obteniendo los mejores lugares a su alrededor

Tabla 37. Historia de usuario, requisito TA06-02

Fuente: elaboración propia

Historia de Usuario	TA07-01
Nombre: Crear los servicios para enviar los datos del usuario al Backend mediante HTTP	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Jonathan Vargas	
Descripción: Como desarrollador quiero implementar un servicio en el Frontend que permita gestionar el protocolo HTTP y enviar los datos del usuario al Backend.	

Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
El sistema debe crear un servicio para gestionar el protocolo HTTP y utilizar sus métodos.	El sistema necesite enviar los datos del usuario al Backend utilizando los métodos del protocolo HTTP	El sistema recibe los datos de Google Maps que son los puntos de interés en formato JSON.

Tabla 38. Historia de usuario, requisito TA07-01

Fuente: elaboración propia

Historia de Usuario	TA07-02
Nombre: Configurar el servidor para procesar solicitudes con contenido JSON	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Roger Laza	
Descripción: El servidor debe procesar solicitudes http con contenido JSON	

Criterios de Aceptación		
Dado que	**Cuando**	**Entonces**
El sistema debe integrar librerías para el manejo de solicitudes HTTP	EL usuario envié sus datos y los posts de Facebook	EL sistema procesa los datos en los servicios integrados de Google Cloud emitiendo una repuesta de los mejores lugares encontrados en un radio de 1.5 km

Tabla 39. Historia de usuario, requisito TA07-02

Fuente: elaboración propia

Historia de Usuario	TA08-01
Nombre: Solicitar la ubicación actual del usuario antes de iniciar sesión	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Jonathan Vargas	
Descripción: Como desarrollador quiero obtener las coordenadas latitud y longitud de la ubicación actual del usuario	
Criterios de Aceptación	

Dado que	Cuando	Entonces
El sistema debe solicitar permisos para acceder al GPS	El sistema necesite obtener la ubicación actual del dispositivo físico para agregarlos a los datos del usuario	El sistema obtiene las coordenadas de latitud y longitud del dispositivo físico cuando el usuario otorgo permisos y el GPS esta activado

Tabla 40. Historia de usuario, requisito TA08-01

Fuente: elaboración propia

Historia de Usuario	TA08-02
Nombre: Desarrollar la interfaz de usuario para mostrar resultados	
Usuario: Usuario en general	
Prioridad en negocio: Alta	
Estimación: 5	Iteración asignada: 1
Programador Responsable: Jonathan Vargas	
Descripción: Como desarrollador quiero desarrollar las interfaces de usuario para mostrar los resultados finales mediante la aplicación móvil.	
Criterios de Aceptación	

Dado que	Cuando	Entonces
El usuario necesita visualizar los puntos de interés y el mapa de Google	El usuario ingrese a la aplicación y observe los puntos de interés dependiendo sus gustos y preferencias de Facebook	El sistema muestra un listado con los puntos de interés ordenados de mayor a menor dependiendo el mejor lugar. En el mapa de Google se agrega los marcadores de los puntos de interés

Tabla 41. Historia de usuario, requisito TA08-02

Fuente: elaboración propia

Sprint Backlog

De acuerdo con la explicación del procedimiento, en la Tabla 42, se definen las tareas asignadas a cada historia de usuario para el Sprint 3.

Sprint Backlog	
Historia de usuario	**Tareas**
TA06-01	Generar una API Key en Google Maps Platform.
	Configurar la API Key para integrar Google Maps en Android.
TA06-02	Escribir el script en el servidor Flask de la API de Google para obtener los lugares turísticos dependiendo la categoría obtenida con el NLC.
	Realizar una fórmula matemática para calcular los mejores lugares turísticos.
	Limpiar los datos que serán enviados desde el servidor Flask a la aplicación móvil.
TA07-01	Crear el servicio para gestionar el protocolo HTTP.
	Crear un script para enviar datos del usuario al Backend mediante el protocolo HTTP.
TA07-02	Agregar las librerías para procesar solicitudes HTTP.
	Realizar pruebas emitiendo peticiones desde el Frontend con respuesta del Backend.
TA08-01	Instalar una librería de capacitor que permita extraer la ubicación actual del usuario.
	Crear un script para extraer las coordenadas de la ubicación actual del usuario.
	Modificar el script para agregar la ubicación actual del usuario.
TA08-02	Instalar la librería Angular Google Maps (AGM) para mostrar el mapa de Google.
	Ubicar los puntos de interés en el mapa de Google utilizando sus coordenadas de longitud y latitud. El mapa debe mostrar el icono y el nombre de cada punto de interés.
	Mostrar la ubicación actual del usuario en el mapa de Google con su foto de perfil como icono.
	Mostrar un listado con los puntos de interés ordenados dependiendo el mejor lugar.
	Dibujar el trazo desde la ubicación actual del usuario hasta el punto de interés seleccionado por el usuario.

Tabla 42. Sprint Backlog del Sprint 3

Fuente: elaboración propia

3.4.2 Ejecución del Sprint.

A continuación, se muestra la implementación de cada una de las historias de usuario y se detallan las más relevantes realizadas en Sprint 3.

Integración de la API de Google Maps

Para realizar la integración de Google Maps primero se debe generar una API key, por ello es necesario que se revise la documentación oficial de Google Maps Platform [69]. Ingresamos a Google Maps Platform y se debe seleccionar el proyecto creado en este caso **touristapp** y dar clic en crear. Una vez que sea redirigido, en la parte de **Crear credenciales** se debe seleccionar **CLAVE DE API**, le asignamos un nombre y aplicamos ciertas restricciones de clave que ayudaran a evitar el uso no autorizado y el robo de cuotas. Para las restricciones de aplicaciones se debe seleccionar ninguno y para las restricciones de API se debe seleccionar Restringir clave y solo habilitar las siguientes APIs [70]:

- **Directions API**: permite obtener la información acerca de una ruta desde un punto inicial hasta un punto destino. Además, proporciona indicaciones para desplazarse para las diferentes formas de transporte (conducir algún vehículo, andar en bicicleta, transporte público o caminar).
- **Maps JavaScript API**: permite personalizar el mapa con su propio contenido e imágenes para que pueda ser mostrado de una mejor manera dependiendo la plataforma web o móvil. Además, proporciona cuatro tipos de mapas básicos que son: hoja de ruta, satélite, hibrido y terreno y que se los puede modificar.
- **Maps SDK for Android**: SDK que permite agregar el mapa a la plataforma Android y ofrecer información adicional al usuario final. Este SDK es compatible con los lenguajes de programación Java y Kotlin.
- **Places API**: servicio que permite devolver información acerca de establecimientos, ubicaciones geográfica o puntos de interés destacados mediante el uso de solicitudes HTTP.

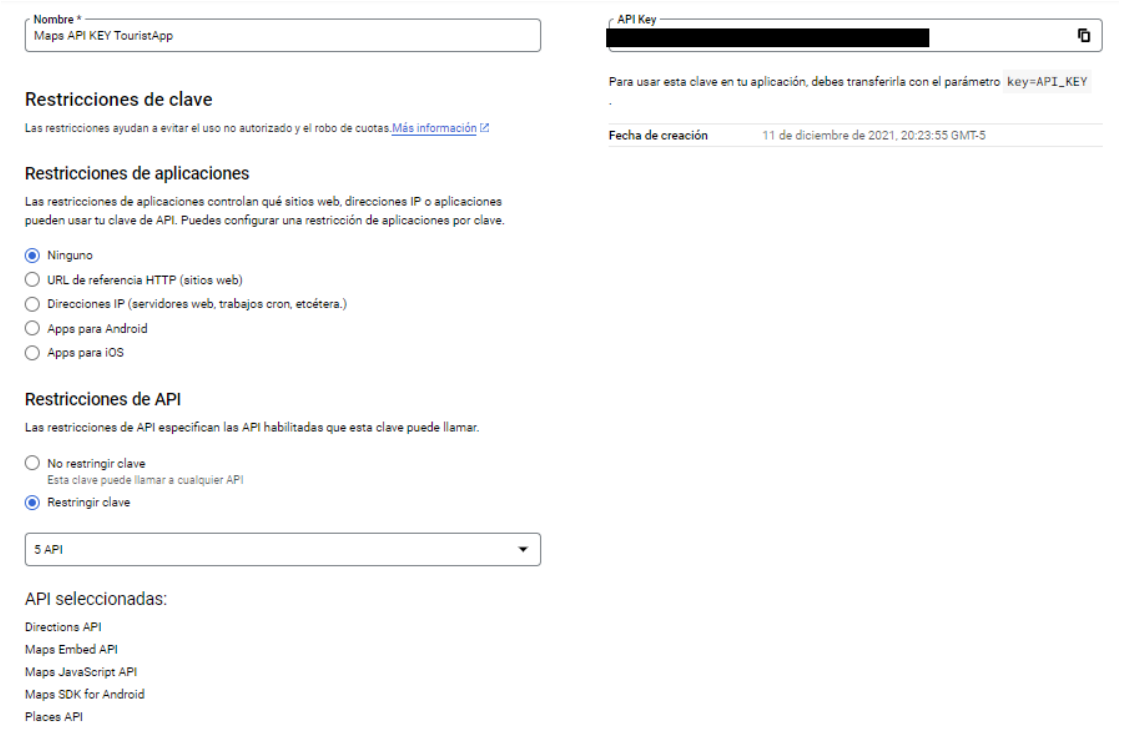

Figura 40. Creación de API Key de Google Maps

Fuente: Google Cloud Platform – Api y servicios [71]

Creación de servicios para gestionar el protocolo HTTP

En el Frontend se crea el servicio con el nombre **UsersService** que permite gestionar el protocolo HTTP y utilizar el método de peticiones POST para enviar una entidad a un recurso en específico [72] (enviar los datos del usuario al Backend).

Para utilizar el método POST es necesario disponer de la ruta del servidor de AWS, para ello se crea un nuevo servicio con el nombre **ConfigService** para declarar la ruta que permitirá conectarse con el Backend.

```
export class ConfigService {
  url_aws: string = 'http://ec2-18-191-244-36.us-east-2.compute.amazonaws.com:3000'
  constructor() { }

  getUrl(){
    return this.url_aws;
  }
}
```

Figura 41. Servicio para declarar la ruta del AWS

Fuente: elaboración propia

Para utilizar el protocolo HTTP en Angular, es necesario importar el servicio **HTTPClient**. Este servicio está disponible como una clase inyectable, con métodos para realizar solicitudes HTTP [73]. El método createOne recibirá el parámetro user_profile que consiste en los datos del usuario y los enviará al Backend mediante el método POST.

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { User } from 'src/app/model/user';
import { ConfigService } from '../utils/config.service';

@Injectable({
  providedIn: 'root',
})
export class UsersService {
  constructor(
    private readonly http: HttpClient,
    private urlService: ConfigService,
  ) { }

  createOne( user_profile : User ) {
    console.log("usuario post", user_profile)
    return this.http.post(this.urlService.getUrl() + '/users', user_profile)
  }
}
```

Figura 42. Servicio para gestionar el protocolo HTTP

Fuente: elaboración propia

Solicitar la ubicación actual del usuario

Para obtener la ubicación actual del usuario es necesario que la aplicación solicite permisos de acceso al GPS y verificar que se encuentre activado. Por ello debe instalar el plugin de capacitor **@capacitor/geolocation** [74] que es una API de geolocalización que proporciona métodos para extraer y rastrear la posición actual del dispositivo físico mediante el uso del GPS, junto con la información de altitud, rumbo y velocidad. Después de instalar el plugin se debe realizar la configuración adicional en el archivo **Manifest** de la aplicación Android. Para realizar esta configuración es necesario revisar la documentación oficial de este plugin [74]. La aplicación debe solicitar permisos para acceder a la ubicación actual del dispositivo físico. En el componente Login se implementará este plugin que solicitará permisos para acceder al Global Positioning System (GPS).

Figura 43. Permiso de acceso al GPS

Fuente: elaboración propia

En la Figura 44, se puede observar el script que permite solicitar permisos para acceder al GPS. Este método se activará cuando la aplicación inicie y cargue el componente Login. Inmediatamente se verificara el estado del permiso (prompt o prompt-with-rationale o denied o granted) con el método **checkPermissions()**. Si el estado del permiso es prompt o prompt-with-rationale o denied (la aplicación no tiene acceso al GPS) se solicitará permisos de acceso al GPS con el método **requestPermissions()**. Si el usuario decide denegar el permiso entonces se mostrará una alerta indicándole que es necesario que otorgue permisos de acceso al GPS desde la configuración de la aplicación.

```
async ngOnInit() {
  let status: any
  let request: any
  try {
   status =  await Geolocation.checkPermissions()
   if (status.location == "prompt" || status.location == "prompt-with-rationale" || status.location == "denied") {

    request = await Geolocation.requestPermissions()
    if(request.location == "prompt" || request.location == "prompt-with-rationale" || request.location == "denied"){

     await this.showAlertgetPermissionGps()
    }else{
     this.turnOnGPS()
    }
   }else{
     this.turnOnGPS()
   }
  }catch (e) {
   console.error("error geolocation ",e.message)
  }
 }
```

Figura 44. Script para solicitar permiso de acceso al GPS

Fuente: elaboración propia

Si el estado del permiso es Granted (el usuario otorgo el permiso de acceso al GPS) se solicitará la activación del GPS mediante el plugin **Location Accuracy** [75] que permite solicitar la activación del GPS mediante un cuadro de dialogo nativo, lo que evita que el usuario tenga que salir de la aplicación. Los plugins que se deben instalar [76] son:

- Npm install ionic-native/location-accuracy
- Npm install cordova-plugin-request-location-accuracy
- Npm install @ionic-native/core

Figura 45. Permiso de activación del GPS

Fuente: elaboración propia

En la Figura 46, se puede observar el script que permite activar el GPS. Si el dispositivo físico no dispone del GPS se mostrará una alerta.

```
turnOnGPS(){
 LocationAccuracy.request(LocationAccuracy.REQUEST_PRIORITY_HIGH_ACCURACY).then(
  () => {
   console.log("GPS Activado")
  },
  error => alert("No se pudo activar el GPS")
 );
}
```

Figura 46. Script para solicitar permiso de activación del GPS

Fuente: elaboración propia

Obtener las coordenadas de la ubicación actual del usuario

Para obtener las coordenadas latitud y longitud se utiliza el plugin @capacitor/geolocation con el método **getCurrentPosition()** [74]. Una vez que se obtienen estas coordenadas se las debe agregar a los datos del usuario. Si el usuario decidió no otorgarle acceso al GPS o no tiene activado el GPS entonces se enviará las coordenadas por defecto de la iglesia San Francisco de Quito que está ubicada en el centro Histórico de Quito.

```
async loadingDataUser() {
  let coordenadas: any
  try{
    coordenadas =  await Geolocation.getCurrentPosition()

  }catch (e) {
    console.error(e.message)
  }
  if(coordenadas){
    this.origin.lat = coordenadas.coords.latitude
    this.origin.lng = coordenadas.coords.longitude
  }else{
    this.origin.lat = -0.220641
    this.origin.lng = -78.515039
  }
  this.user_profile = JSON.parse(localStorage.getItem('user_profile'))
  this.user_profile.locationCurrent.latitude = this.origin.lat
  this.user_profile.locationCurrent.longitude = this.origin.lng
```

Figura 47. Script para obtener las coordenadas de ubicación

Fuente: elaboración propia

Enviar datos del usuario al Backend y obtener los puntos de interés

El script de la Figura 48, permite enviar los datos del usuario al Backend utilizando el método **createOne** creado en el servicio, y obtener como respuesta los puntos de interés en formato JSON proporcionados por el servicio de Google Maps.

```
this.userService.createOne(this.user_profile).subscribe((resp: any) => {
    console.log(resp)
})
```

Figura 48. Script para enviar datos al Backend

Fuente: elaboración propia

En **point_interest** se puede observar las categorías con sus respectivos puntajes. Como resultado se obtendrán solo los lugares que pertenezcan a la categoría con mayor puntaje y estén cerca a la ubicación del usuario.

Figura 49. Puntos de interés

Fuente: elaboración propia

Desarrollo la interfaz de usuario para mostrar resultados finales

Para mostrar el mapa de Google se instaló la librería Angular Google Maps (AGM) [77][78] que es un componente de Angular para Google Maps además, se utilizó las etiquetas descritas en la Tabla 43.

Etiqueta	Descripción
agm-map	Renderiza Google Map
agm-marker	Renderiza un marcador de mapa dentro de AgmMap.
agm-info-window	Renderiza una ventana de información dentro de un AgmMarker
agm-direction	Dibuja la ruta desde el punto de origen hasta el punto de destino

Tabla 43. Etiquetas de AGM

Fuente: elaboración propia

Para instalar esta librería se debe utilizar el comando **npm install @agm/core** y revisar la documentación oficial [79] para importar AGM en Angular y agregar la API Key que se obtuvo en la Figura 40.

```
import { AgmCoreModule } from '@agm/core';
import {environment} from "../../../environments/environment";
import {AgmDirectionModule} from "agm-direction";
import {ModalMapComponent} from "../modal-map/modal-map.component";

@NgModule({
 imports: [
  CommonModule,
  FormsModule,
  IonicModule,
  HomePageRoutingModule,
  BlockUIModule.forRoot(),
  AgmCoreModule.forRoot({
   apiKey: environment.apiKeyMap
  }),
  AgmDirectionModule,
 ],
 declarations: [HomePage, ModalMapComponent]
})
export class HomePageModule {}
```

Figura 50. Importación de AGM en Angular

Fuente: elaboración propia

En el componente Home se debe mostrar el mapa de Google y agregar un marcador con la ubicación actual del usuario y un icono con su foto de perfil.

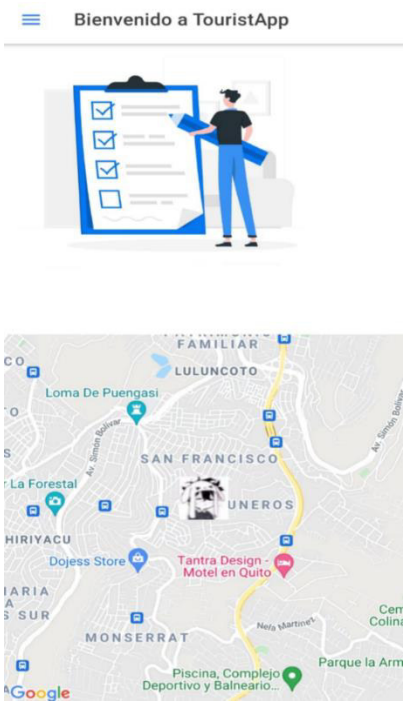

Figura 51. Visualización del mapa de Google y ubicación del usuario

Fuente: elaboración propia

En la Figura 52, se puede observar el menú lateral con los valores de los puntos de interés analizados. Para realizar el cálculo de porcentajes se utilizó la Fórmula 1, con el score de predicción más los pesos divida entre el número de veces que el tag estuvo entre las cuatro primeras posiciones, multiplicada por cien se obtiene los porcentajes de interés

$$interest\_percent = \frac{score}{repeat} * 100 \quad 1)$$

Se realiza el cálculo del porcentaje de interés se realiza una media con los datos del score de clasificación, el peso de la etiqueta dependiendo su posición el resultado de la predicción y el número de veces que se repite esta etiqueta. La etiqueta tourist_attraction fue obviada por ser muy general.

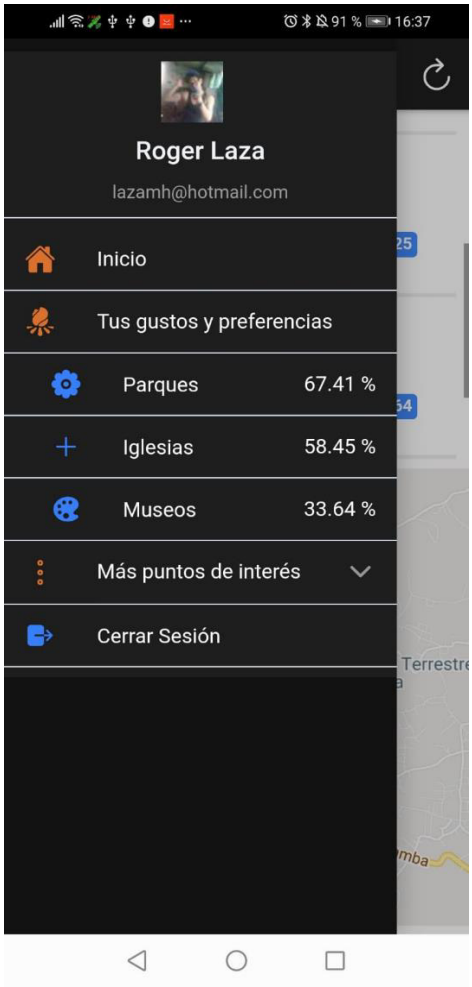

Figura 52. Visualización del menú

Fuente: elaboración propia

En la Figura 53, se puede observar el script que permite mostrar los porcentajes de interés de cada etiqueta correspondiente a los gustos y preferencias del usuario.

```
arrPointInterest.sort((a, b) => {
      return b.percent - a.percent
    })
    arrPointInterest.forEach((point, index) => {
     if (point.percent == 0) {
      this.otherTags.push({
       titulo: point.titulo,
       url: `/punto-interes/${point.tag}`,
       icon: point.icon
      })
     }else {
      if (index < 3) {
       this.appPages.push({
        titulo: point.titulo,
        icon: point.icon,
        url: point.url,
        percent: point.percent.toFixed(2)
       })
      } else {
       this.otherTags.push({
        titulo: point.titulo,
        url: `/punto-interes/${point.tag}`,
        icon: point.icon
       })
      }
     }
```

Figura 53. Script para el cálculo de porcentajes

Fuente: elaboración propia

Los puntos de interés obtenidos se los mostrará en el componente Home, con un máximo de tres etiquetas, en un listado y ordenados de mayor a menor según la variable **rating_place**. Los datos que se mostrarán de los puntos de interés son: tipo (resturant, church, museum, park), nombre del lugar, rating y el número de visitas. Además, cuando el usuario haga touch en cualquier marcador del mapa de Google aparecerá un cuadro de información mostrando el nombre del sitio. En la Figura 54, se indica los puntos de interés en la aplicación.

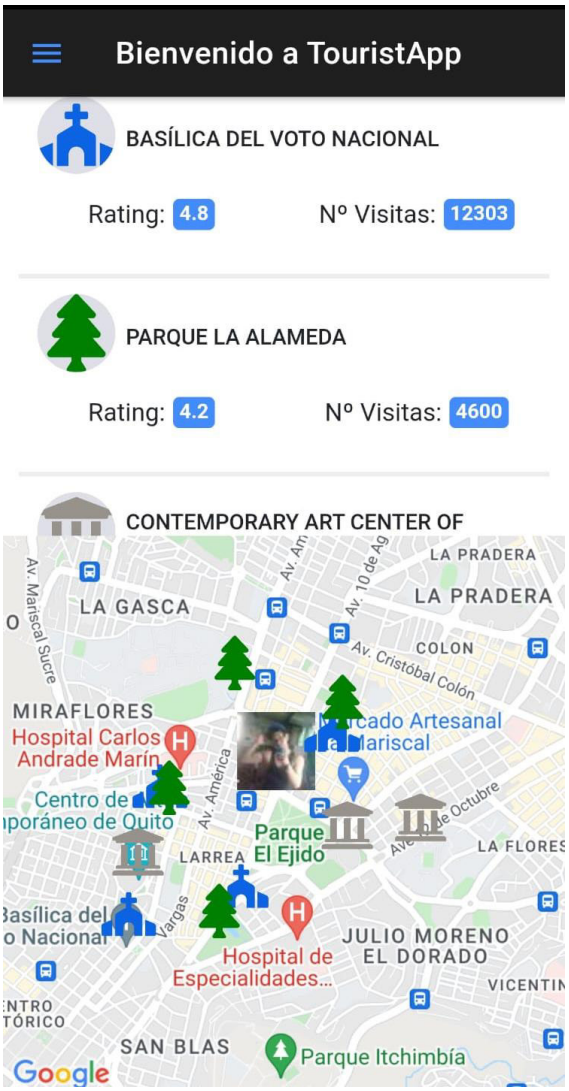

Figura 54. Puntos de interés en el mapa de Google

Fuente: elaboración propia

En la Figura 55, se puede observar el script que permite agregar los marcadores, mostrar el listado de los puntos de interés y mostrar el cuadro de información de cada marcador.

```
this.userService.createOne(this.user_profile).subscribe((resp: any) => {
    console.log(resp)
    resp.google_maps.sort((a, b)=>{
      return b.rating_place - a.rating_place
    })
    console.log("order by rating_place", resp.google_maps)

    resp.google_maps.forEach((place) => {
      let respPlaceFind = this.categoriesPlaces.find((value)=>{
        return value == place.types[0]
      })
      if(!respPlaceFind){
        respPlaceFind = "tourist_attraction"
      }

      this.placesDestination.push({
        place: {
          type: respPlaceFind,
          name: place.name,
          rating: place.rating,
          user_rating: place.user_ratings_total
        },
        destination: {
          lat: place.location.lat,
          lng: place.location.lng
        },
        markerOptions: {
          origin: {
            icon: this.user_profile.imagen,
          },
          destination: {
            icon: `assets/svg/${respPlaceFind}.svg`,
            infoWindow: `<h4> ${place.name} </h4>`,

          },
        }
      })
    })
  })
}
```

Figura 55. Script para agregar los puntos de interés en el mapa

Fuente: elaboración propia

Cuando el usuario haga touch en cualquier punto de interés del listado se activará un modal que mostrará el mapa de Google con dos marcadores. El primer marcador será la ubicación del usuario como punto de origen y el segundo marcador será la ubicación del punto de interés como destino. Entre estos dos marcadores se dibujará la ruta y cuando se haga touch en el punto de interés se mostrará el cuadro de información con sus respectivos datos.

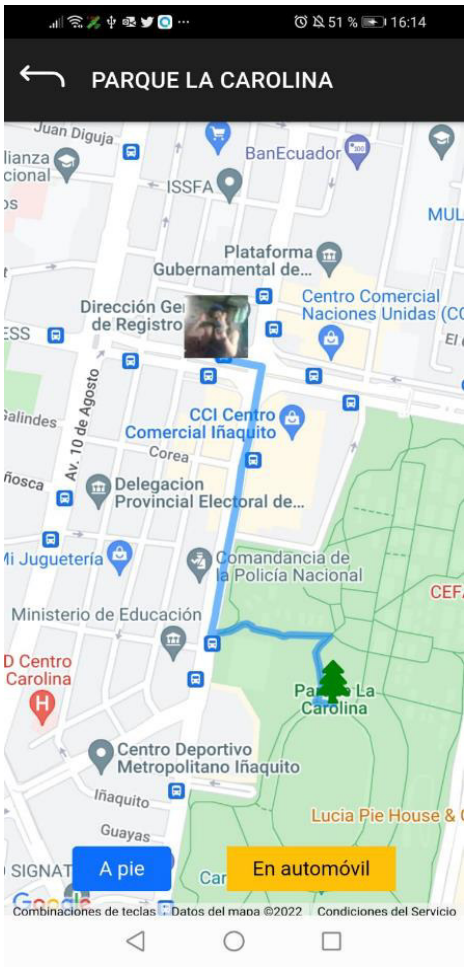

Figura 56. Ruta entre el usuario y el punto de interés

Fuente: elaboración propia

En la Figura 57, se puede observar el script que permite mostrar el modal con los datos del punto de interés y la ubicación del usuario.

```
async openModalMap(destination: DestinationLocation){
 const modal = await this.modalMap.create({
  component: ModalMapComponent,
  componentProps: {origin: this.origin, destination: destination}
 })
 return await modal.present();
}
```

Figura 57. Script para mostrar la ruta entre dos puntos

Fuente: elaboración propia

Construir un APK

Un archivo con la extensión APK (Android Package Kit) es un formato de archivos que utilizan las aplicaciones para que puedan ser instaladas en dispositivos con el sistema operativo Android [80].

Para construir un APK en modo debug de la aplicación TouristApp es necesario revisar la documentación oficial de Android Studio [81]. Este APK puede ser enviado a los usuarios para que la instalen en su dispositivo Android y pueden probar el funcionamiento de la aplicación.

Integración de la API de Google Maps en el Backend

Para realizar la integración del API de Google Maps en el Backend se ha estudiado varios productos que se muestra en los servicios de Google Cloud, para lo cual se ha decidido utilizar el servicio de Nearby Search el cual se describe a continuación:

- **Nearby Search:** consta con una búsqueda por proximidad que permite buscar lugares dentro de una zona determinada. Para afinar la solicitud de búsqueda se debe proporcionar parámetros que contenga palabras clave o especificando el tipo de lugar que está buscando. A continuación, se indica los parámetros que son obligatorios y cuales opcionales:
  - **Required parameters**: a continuación, se indica cuáles son los parámetros obligatorios de enviar.
    - **Location**: el punto alrededor del cual se recupera la información del lugar. Debe especificarse como latitud y longitud.
    - **Key**: credenciales de autenticación adecuadas que tienen la forma de una clave de API, una cadena alfanumérica única que asocia la cuenta de facturación de Google con el proyecto y con la API o el SDK específicos.
- **Optional parameters** a continuación, se indica algunos parámetros que son de opcionales de enviar, en este caso se indicara solo los parámetros relevantes para este proyecto, para ver todos los parámetros revisar la referencia:
  - **Radius**: define la distancia (en metros) dentro de la cual se devolverán los resultados del lugar, los resultados obtenidos estarán contenidos en un círculo por el lugar y el radio. Al hacerlo, el servicio de lugares prefiere mostrar los resultados dentro de ese círculo.

- **Type**: restringe los resultados a los lugares que coinciden con el tipo especificado. Sólo se puede especificar un tipo. Si se proporciona más de un tipo, se ignoran todos los tipos que siguen a la primera entrada.
  - type=hospital|farmacia|médico: se convierte en type=hospital
  - type=hospital,farmacia,médico: se ignora por completo

Una vez definido los parámetros que se usarán para realizar peticiones a la API de Google Maps se procede a escribir el script, véase la documentación [82], se integra al script los parámetros de location, radius, type y key y para parámetros lo cuales son definidos en el método call_api_google como resultado se obtiene el script indicado en la Figura 58.

```
import requests

def call_api_google(latitude, longitude, radio, point_of_interest):
    location = str(latitude) + "," + str(longitude)
    url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json?"
    url += "location=" + location
    url += "&radius=" + radio
    url += "&type=" + point_of_interest
    url += "&key=" + '{secretkey}'

    payload = {}
    headers = {}

    response = requests.request("GET", url, headers=headers, data=payload)
    gps = response.json()
    return gps
```

Figura 58. Script con el API de Google Maps

Fuente: elaboración propia

Cuando se realice la ejecución del script se obtiene un array JSON. En la Figura 59, se puede observar información relevante como: lat, long, rating, user_rating_totals, types name estos parámetros son los más importantes para definir los lugares que serán enviados a la aplicación móvil para que sean procesados.

```
{
    business_status: "OPERATIONAL",
  - geometry: {
      - location: {
            lat: -0.186365,
            lng: -78.48552939999999
        },
      - viewport: {
          - northeast: {
                lat: -0.184609369708498,
                lng: -78.48394121970848
            },
          - southwest: {
                lat: -0.187307330291502,
                lng: -78.48663918029149
            }
        }
    },
    icon: "https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/park-71.png",
    icon_background_color: "#4DB546",
    icon_mask_base_uri: "https://maps.gstatic.com/mapfiles/place_api/icons/v2/tree_pinlet",
    name: "Jardín Botánico de Quito",
  - opening_hours: {
        open_now: false
    },
  + photos: [ … ],
    place_id: "ChIJrTIW7nya1ZER_HZpDAlxJyY",
  + plus_code: { … },
    rating: 4.6,
    reference: "ChIJrTIW7nya1ZER_HZpDAlxJyY",
    scope: "GOOGLE",
  - types: [
        "tourist_attraction",
        "park",
        "point_of_interest",
        "establishment"
    ],
    user_ratings_total: 6756,
    vicinity: "INTERIOR PARQUE, PASAJE # 34, Rumipamba E6-264 Y, Quito"
},
```


Figura 59. Resultados del llamado de la API de Google Maps

Fuente: elaboración propia

Una vez que escrito el archivo indicado en la Figura 58, se utiliza el protocolo SCP para que el archivo sea transferido al servidor localizado en AWS con el comando descrito en la Tabla 20.

Una vez estudiado los parámetros que serán utilizados para tener los mejores lugares turísticos se prepara la respuesta y se envía en la petición hacia Google Maps el cual nos retorna un JSON con los lugares especificados por el punto de interés del usuario.

Mediante el estudio de los trabajos previos en lugares turísticos se han considerado los datos más importantes para desarrollar una fórmula matemática que permita realizar el filtrado de datos, para esto se ha tomado así: se considera que el lugar sea visitado por más de 100 personas para considerarlo recomendable y que su rating este sobre 4 puntos que también es aceptable, la Formula 2 muestra la fórmula matemática que permite recomendar lugares de mayor a menor importancia, estos datos son enviados hacia la aplicación móvil para su procesamiento.

$$Calculate\ the\ best\ places = \frac{user\ ratings\ total}{rating} \quad 2)$$

Realizar la comunicación entre el Frontend y el Backend

Para realizar la comunicación entre el Frontend y el Backend el equipo de desarrollo estudió la estructura de datos que tendrá el contenido de los archivos, para eso, se ha diseñado modelos para que sea más sencillo el manejo de estas.

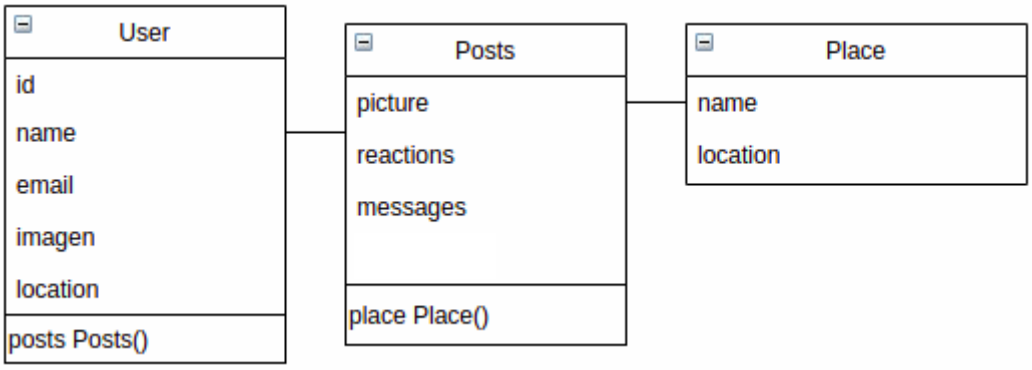

Figura 60. Estructura de datos del JSON de respuesta y envió

Fuente: elaboración propia

Comunicación desde el Backend

Para implementar la comunicación en el servidor Backend se ha optado por importar la librería Cross Origin Resource Sharing (CORS), que permite recibir solicitudes HTTP complejas de una API REST [83].

Como se ha definido antes, el servidor Backend recibirá un JSON con la estructura de la Figura 60, esto permitirá que los datos recibidos sean manipulables y antes de que sean procesados por los servicios de Google Cloud se ha decidido por concatenar la variable message del Post

con la variable namePlace, con el fin de que la predicción de la categoría sea más precisa al momento de identificar los gustos del usuario. Cabe recalcar que los posts que no posean esta información no serán tomados en cuenta.

Una vez procesada la información por los servicios de Google Cloud se obtendrá un JSON de resultados con lugares mejores puntuados de Google Map, se ha implementado el método descrito en la Figura 61, para filtrar la siguiente información: name, location, icon, rating, user rating total rating place además los lugares filtrados serán con un rating mayores a 4 y la cantidad de usuarios que visitaron el lugar será mayor a 100, esto permitirá tener más opciones al usuario en visitar lugares.

```
def filterData( google_maps , counter : int ):
    result = []
    i = 0
    for values in google_maps['results']:
        if( 'rating' in values and values['rating'] >= 4.2 and values['user_ratings_total'] > 100 ):

            rating_place = calculateBestPlaces( values['rating'] , values['user_ratings_total'] )

            result.append({
                'name' : values['name'],
                'location' : values['geometry']['location'],
                'icon' : values['icon'],
                'rating' : values['rating'],
                'types':values['types'],
                'rating':values['rating'],
                'user_ratings_total':values['user_ratings_total'],
                'rating_place': rating_place
            })
            i += 1

        if( counter == i ):
            break

    return result
```

Figura 61. Filtración de tipo de datos

Fuente: elaboración propia

Pruebas de aceptación

Luego de realizar el trabajo del Sprint 3, en la Tabla 44, se puede constatar la verificación de cumplimiento con respecto a todos los criterios de aceptación de las historias de usuario.

Historia de usuario	Dado que	Cuando	Entonces	Estado
TA06-01	El sistema debe integrar Google Maps.	El usuario ingrese a la pantalla principal (componente Home) y desee ver el mapa de Google	Se genera una API key en la Google Maps Platform y se realiza la configuración necesaria para mostrar el mapa de Google en la aplicación.	Terminado
TA06-02	El sistema debe obtener los mejores lugares turísticos dado una categoría	El usuario ingrese a la aplicación TouristApp y desee ver lugares turísticos dependiendo sus gustos	El sistema procesa la información obtenida por el usuario y predice cual es el punto de interés más prioritario obtenido los mejores lugares a su alrededor	Terminado
TA07-01	El sistema debe crear un servicio para gestionar el protocolo HTTP y utilizar sus métodos.	El sistema necesite enviar los datos del usuario al Backend utilizando los métodos del protocolo HTTP	El sistema recibe los datos de Google Maps que son los puntos de interés en formato JSON.	Terminado
TA01-04	El sistema debe integrar librerías para el manejo de solicitudes HTTP	EL usuario envié sus datos y los posts de Facebook	EL sistema procesa los datos en los servicios integrados de Google Cloud emitiendo una repuesta de los mejores lugares encontrados en un radio de 1.5 km	Terminado
TA02-01	El sistema debe solicitar permisos para acceder al GPS	El sistema necesite obtener la ubicación actual del dispositivo físico para agregarlos a los datos del usuario	El sistema obtiene las coordenadas de latitud y longitud del dispositivo físico cuando el usuario otorgo permisos y el GPS esta activado	Terminado
TA02-02	El usuario necesita visualizar los puntos de interés y el	El usuario ingrese a la aplicación y observe los puntos de interés dependiendo sus gustos y	El sistema muestra un listado con los puntos de interés. En el mapa de Google se agrega los marcadores de los puntos de interés	Terminado

	mapa de Google	preferencias de Facebook		

Tabla 44. Pruebas de aceptación del Sprint 3

Fuente: elaboración propia

3.4.3 Sprint Review

Los objetivos del Sprint 3 se cumplieron, aunque se tuvo ciertos obstáculos y se los detallan a continuación:

La aplicación se instaló y se probó en dispositivos con versiones de Android 5, 9, 10 y 11. En la versión de Android 5 no funciona el plugin de facebook-login, en la versión de Android 9 funciona según lo esperado, pero el problema que se observó fue que en las versiones de Android 10 y 11 los datos del usuario que la aplicación TouristApp obtenía cuando el usuario iniciaba sesión no se enviaban al Backend.

El problema es porque los datos del usuario que la aplicación recopila se envían sin cifrar a través del protocolo HTTP. Para solucionar este problema se debe especificar el atributo android: usesCleartextTraffic en true en el archivo Manifest de la aplicación Android [84]. Este atributo tiene el valor predeterminado de true para las versiones de API 27 o inferiores.

3.4.4 Sprint Retrospective

Como se muestra en el gráfico de Sprint Burn Down chart existió un leve desface al verificar que la aplicación móvil funcione de una manera correcta, este problema se pudo solventar en los demás días y se hizo un ajuste para completar correctamente el sprint.

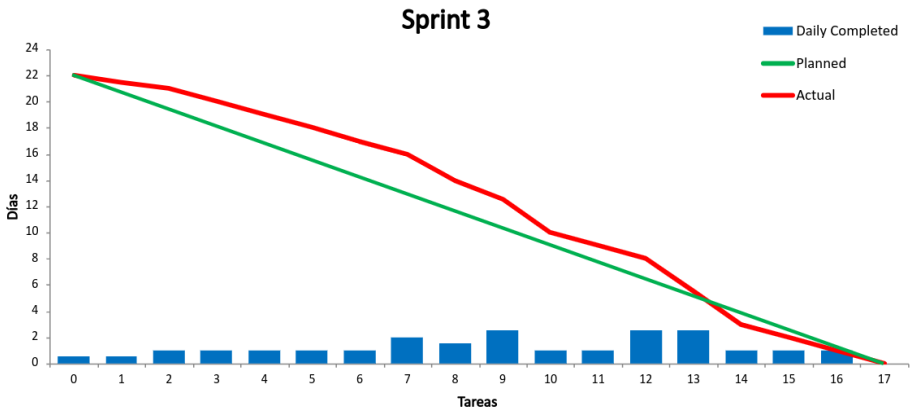

Figura 62. Burndown Chart del Sprint 3

Fuente: elaboración propia

4. RESULTADOS Y DISCUSIÓN

En esta sección se presenta los resultados que fueron obtenidos en base al desarrollo de los objetivos específicos lo cuales fueron cumplidos a continuación, se discute los resultados.

4.1 Analizador de sentimientos

Para el analizador de sentimientos se tuvo que implementar una API la cual se encuentra en el servidor de AWS, por lo cual se envía un conjunto de oraciones con una separación por punto, el resultado que se observa en la Figura 63, muestra el score que oscila entre –1 negativo y 1 positivo que corresponde a la inclinación emocional, se debe tomar en cuenta que no existe valores negativos y menores de 0.7 debido a que son considerados como sentimientos negativos y estos son descartados.

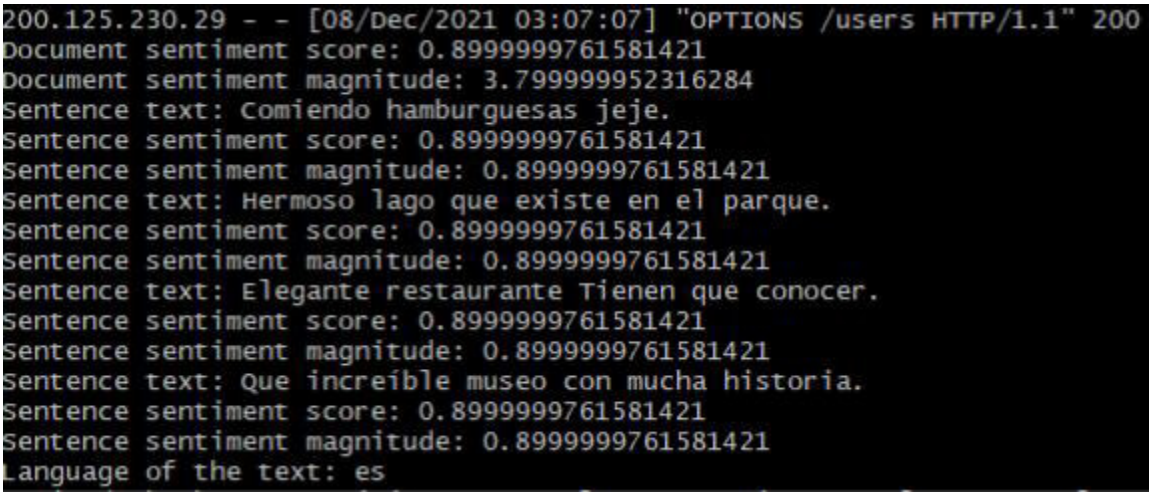

Figura 63. Resultados del analizador de sentimientos

Fuente: elaboración propia

4.2 Natural Language Classifier

Para el NLC se utilizó el servicio de Google Cloud, este fue entrenado por un conjunto de datos que se obtuvieron en grupos turísticos y encuestas, algo importante de recalcar fueron las etiquetas que tienen Google Maps, esto se debe que el NLC devuelve la etiqueta con la que fue entrenado el modelo, esto permitirá una mejor comunicación entre la API de Google Maps y la API de NLC, con los datos y las etiquetas definidas se procedió a realizar un archivo CSV con un total de 1035 datos, el cual fue subido en el NLC de Google. Una vez ejecutado el entrenamiento se obtuvo que los datos entrenados fueron 830 apartando 101 para pruebas y 104 para validación.

La Tabla 45, se precisa las etiquetas definidas con la cantidad de datos que fueron entrenados para su posterior prueba y despliegue.

Etiqueta	Número de elementos
church	202
museum	171
park	284
restaurant	125
tourist_attraction	253

Tabla 45. Etiquetas utilizadas con el número de elementos

Fuente: elaboración propia

Cuando finaliza la etapa de entrenamiento el NLC de Google muestra los resultados que se obtuvo. La precisión fue de un 91.75 % y la Recuperación de 88.12 %. En la Matriz de confusión arrojada por el NLC indicada en la Figura 64, se muestra la frecuencia con la que el modelo clasificó cada etiqueta de manera correcta (en color azul) y qué etiquetas se confundieron más frecuentemente con ella (en color gris).

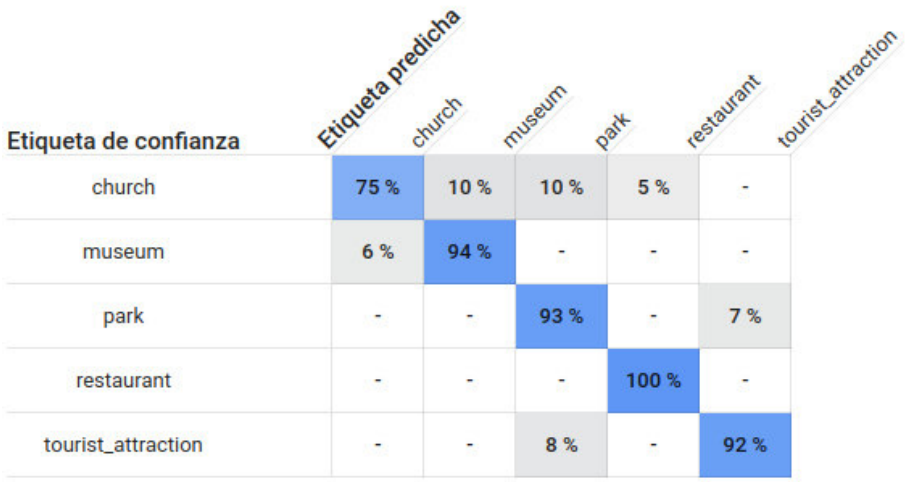

Figura 64. Matriz de confusión

Fuente: Google Cloud Platform – Natural Language [65]

Para probar y usar el modelo se tomó uno de los elementos de prueba y se procedió a usar la herramienta que nos facilita NCL, se ingresa el texto **“Deliciosos y sabrosos encebollados no te lo pierdas."** El resultado de la predicción se puede apreciar en la Figura 65, muestra el score entre más cerca se encuentra del 1 será la predicción de la etiqueta acerca del texto ingresado, en este ejemplo la categoría que pertenece el texto ingresado es de “restaurant”.

Figura 65. Resultado de la predicción

Fuente: Google Cloud Platform - Natural Language [65]

4.3 Producto final

La aplicación móvil consta de tres pantallas, una pantalla de Login para acceder a la aplicación, la pantalla Home que muestra los lugares turísticos cercanos y por último la pantalla Modal que muestra la ubicación actual y al punto donde se desea llegar.

En la pantalla de Login, para ingresar a la aplicación es necesario los datos de ubicación que son otorgados por la geolocalización del usuario. Para iniciar sesión con Facebook se pedirá permisos para acceder a la información y los posts de usuario. A continuación, se muestra el proceso de inicio de sesión del usuario.

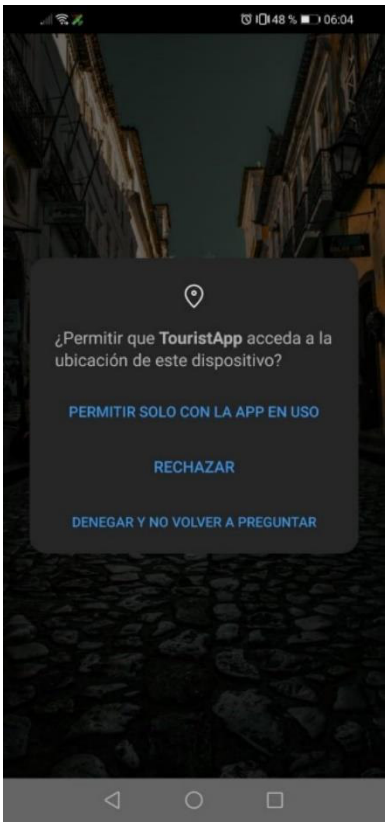

Figura 66. Prueba de permisos de ubicación

Fuente: elaboración propia

Figura 67. Prueba de inicio de sesión con Facebook

Fuente: elaboración propia

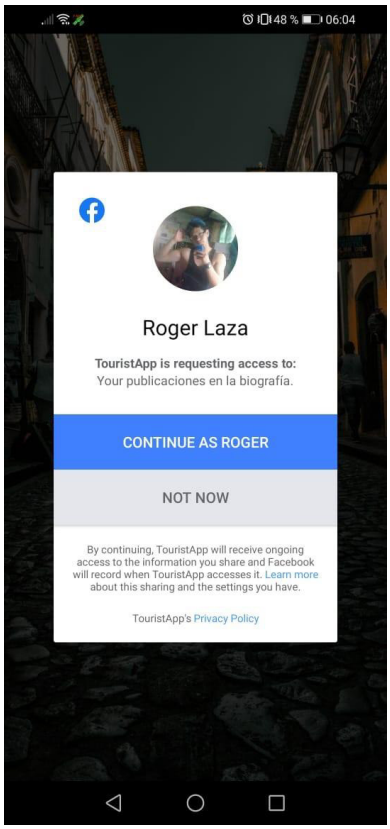

Figura 68. Prueba de permisos de Facebook

Fuente: elaboración propia

En la Figura 69, se puede observar el menú lateral con el ítem de “Tus gustos y preferencias” que contiene los puntos de interés y su respectivo porcentaje de interés. Se añadió otro ítem llamado “Más puntos de interés” el cual contiene una variedad de etiquetas (Aquarium, night_club, café, stadium, movie_theater, zoo) apartados de los intereses de usuario para agregar más valor a la aplicación como se indica en la Figura 70.

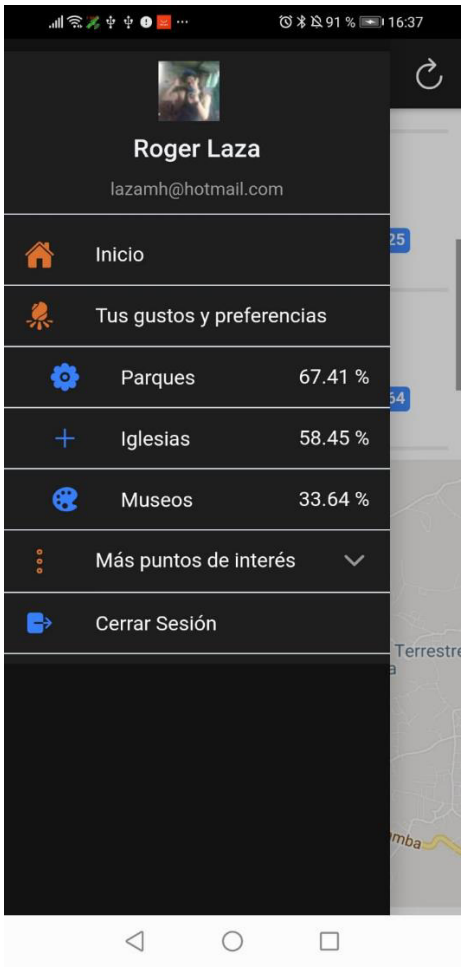

Figura 69. Resultados de análisis de interés

Fuente: elaboración propia

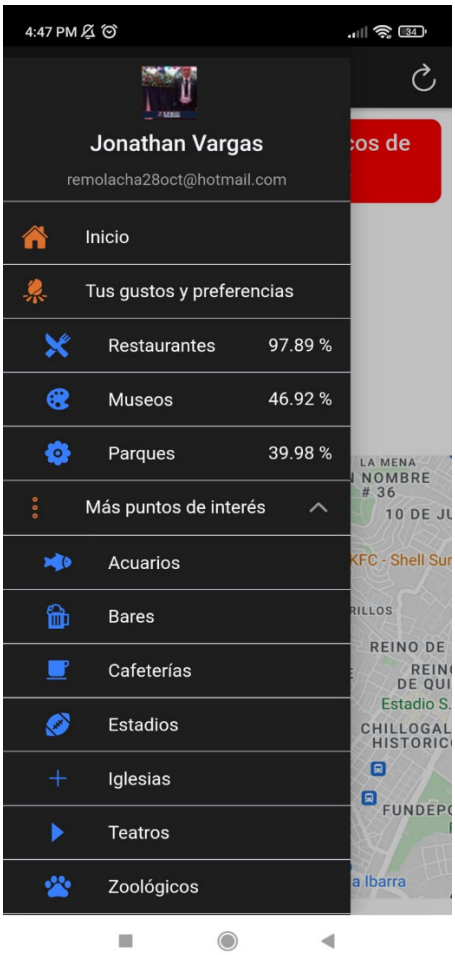

Figura 70. Menú más puntos de interés

Fuente: elaboración propia

Para la pantalla Home consta de una lista de lugares turísticos y un mapa que indica los sitios cercanos, algo que se debe recalcar, es que, los puntos generados consideran los intereses detectados del usuario, cuyo proceso se explicó en la unidad previa. A continuación, se muestra con tres usuarios ubicados en puntos estratégicos de la ciudad de Quito, para el primer usuario Roger Laza su punto de interés es de parques, Para el segundo usuario se le ubicó en el centro histórico de Quito el cual muestra claramente una fascinación por las iglesias. Para el tercer usuario se le ubicó en el norte de Quito el cual muestra una fascinación por los restaurantes. Como información adicional si el usuario no tiene posts en Facebook se realiza una búsqueda con la etiqueta tourist_attraction mostrando estos sitios en la aplicación

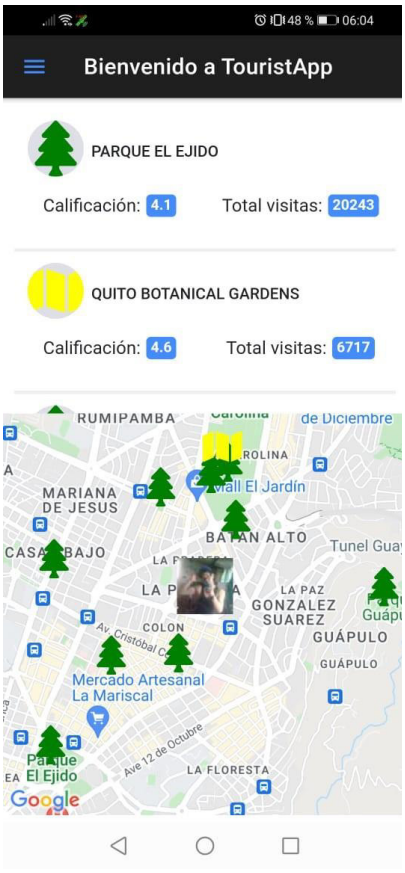

Figura 71. Puntos de interés usuario1

Fuente: elaboración propia

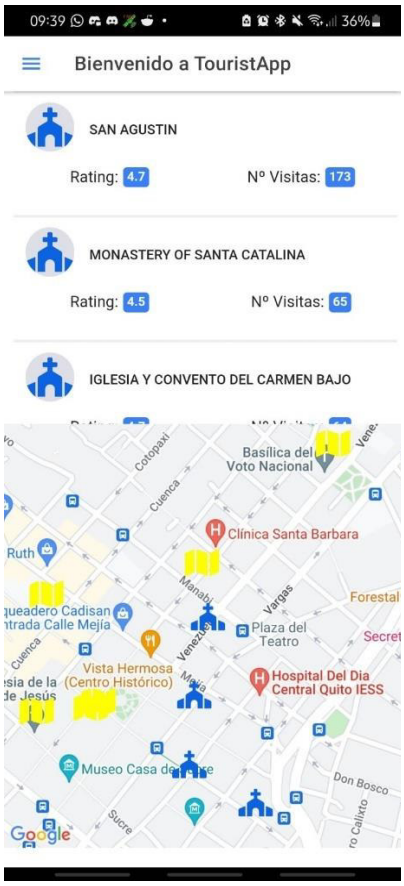

Figura 72. Puntos de interés usuario2

Fuente: elaboración propia

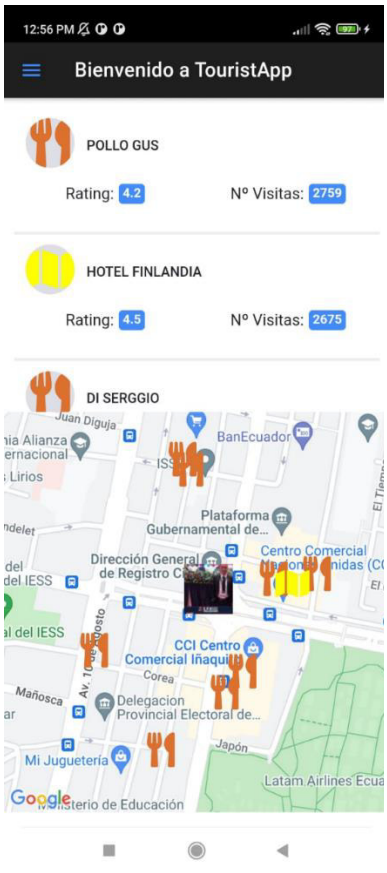

Figura 73. Puntos de interés usuario3

Fuente: elaboración propia

La pantalla Modal indica la ruta para llegar al lugar seleccionado. Se muestra dos modos de viaje que son: **A pie** y **En automóvil**. El modo de viaje **A pie** mostrará la ruta para que el usuario pueda llegar caminando a su destino. Y en el modo de viaje **En automóvil** mostrará la ruta para que el usuario puede llegar conduciendo un vehículo a hacia su destino.

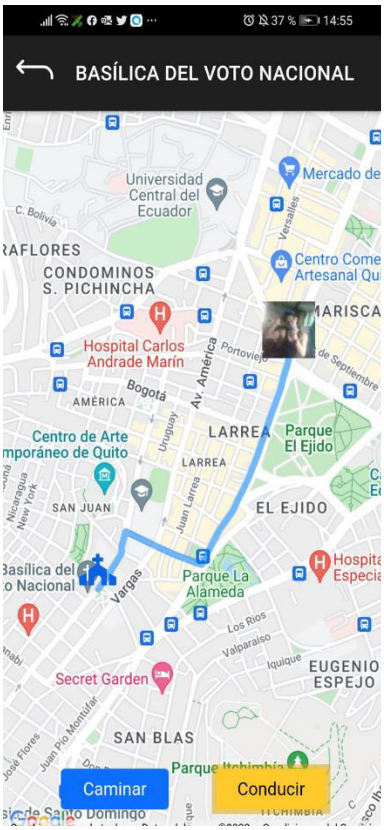

Figura 74. Ruta para usuario1

Fuente: elaboración propia

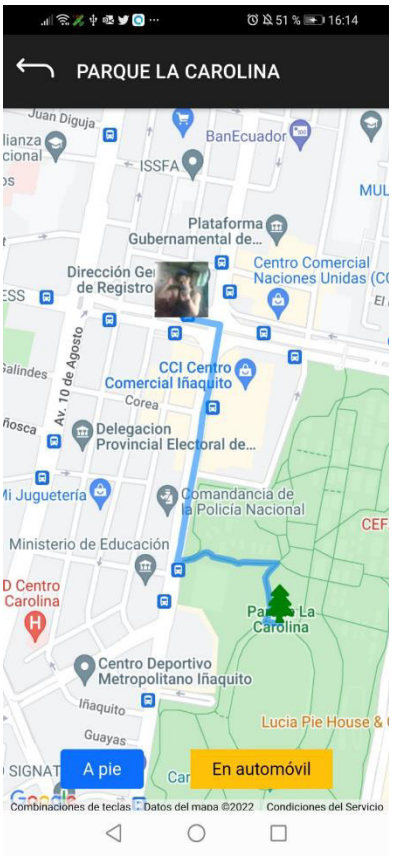

Figura 75. Ruta para usuario2

Fuente: elaboración propia

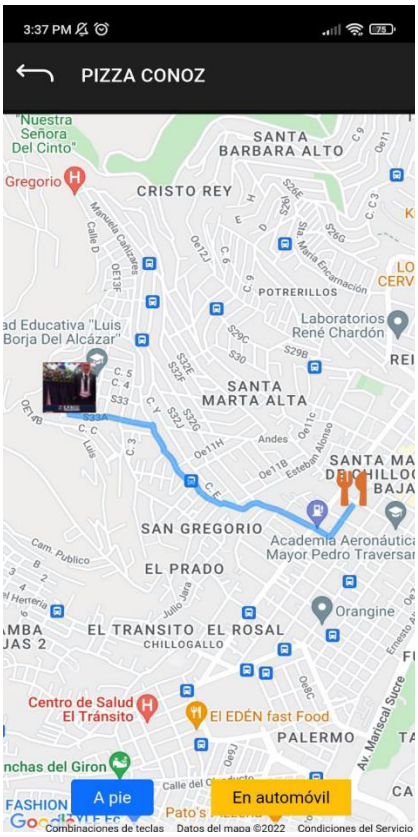

Figura 76. Ruta para usuario3

Fuente: elaboración propia

Si en un caso no existe puntos de interés alrededor de 1.5 km, en la pantalla Home se mostrará un mensaje indicando que no existe lugares cercanos como se muestra en la Figura 77.

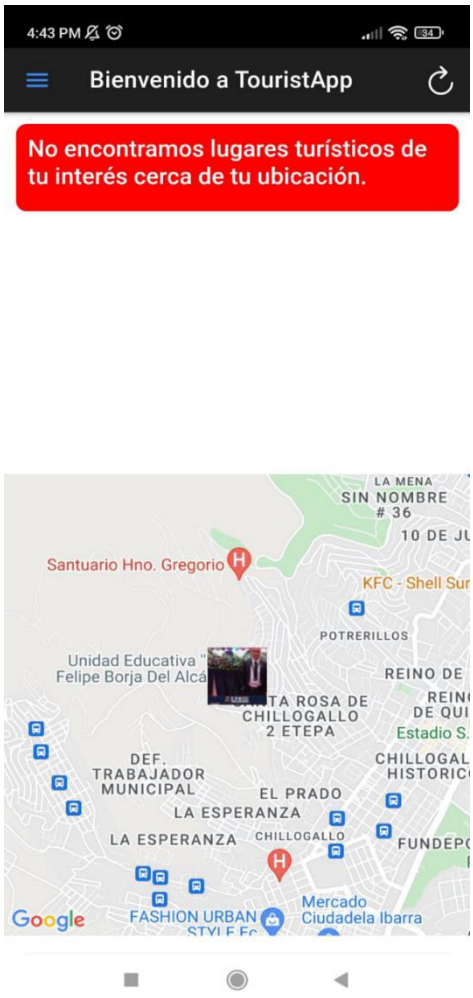

Figura 77. Ruta para usuario2

Fuente: elaboración propia

4.4 Pruebas con usuario finales

Las pruebas con los usuarios finales se realizaron al producto final con un total de 20 personas con el rol de usuario de prueba en la aplicación móvil de Facebook TourisApp, entre el rango de las edades: 18 – 65 años mostrada en la Figura 78. La encuesta permitió determinar la precisión del interés y la usabilidad de la aplicación.

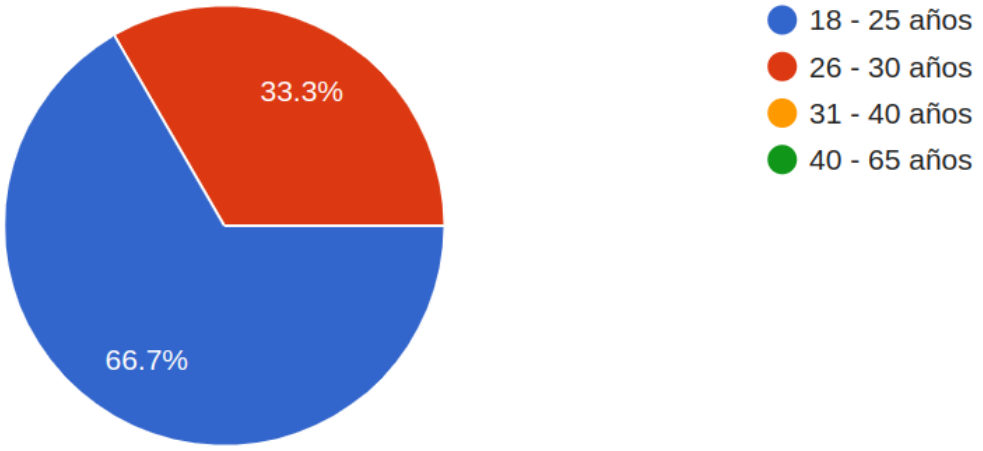

Figura 78. Distribución de edades en pruebas de usabilidad de autores.

Fuente: elaboración propia

Se efectuaron un total de 6 preguntas de las cuales se distribuyeron en 2 preguntas para precisión del interés aquí se evaluará a los implicados si los resultados obtenidos son acordes al punto **Entrenar los datos en el NLC de Google** y 4 preguntas para la usabilidad desarrollado en el punto **Desarrollo la interfaz de usuario para mostrar resultados finales**, cada pregunta se ajustó a una escala de respuestas en un rango de valores del 1 al 5 donde el 1 es Muy difícil y 5 es Muy fácil. Las preguntas realizadas y sus resultados se encuentran en el Anexo III.

4.4.1 Análisis de precisión de intereses

En la Figura 79, se puede observar el promedio de respuestas para cada pregunta con respecto a la utilidad percibida. La pregunta 1 se refiere sobre las recomendaciones dadas por la aplicación son acordes a los intereses del usuario, y obtuvo el promedio de (4.48/5) lo que significa que los usuarios estuvieron conformes con los resultados otorgados por la aplicación móvil ya que estuvieron basados en sus gustos y preferencias, cabe recalcar que los usuarios querían ver más variedad en los resultados.

La pregunta 2 se refiere, si el usuario utilizaría las recomendaciones otorgadas por la aplicación móvil para visitar cualquiera de estos lugares, y obtuvo un promedio de (4.71/5) por lo tanto se deduce que el usuario tiene intención de usar los resultados mostrados por la aplicación y probablemente visitar los lugares turísticos.

Por lo tanto, se deduce que los usuarios percibieron un alto grado de satisfacción con los resultados obtenidos que están acordes a sus gustos e intereses.

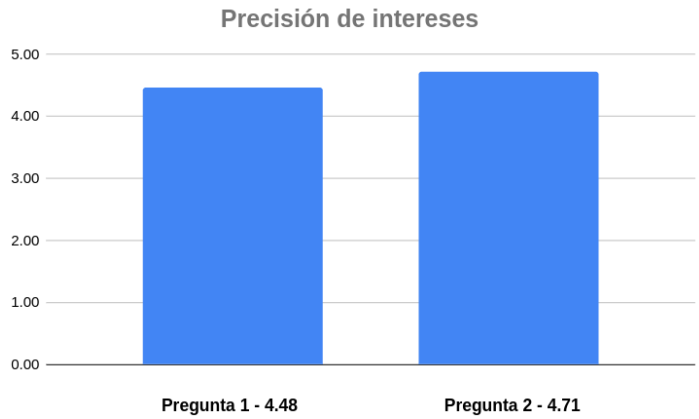

Figura 79. Promedio por pregunta de la utilidad percibida en el producto final

Fuente: elaboración propia

4.4.2 Análisis usabilidad de la aplicación

En la Figura 80, se puede observar el promedio de respuestas para cada pregunta con respecto a la usabilidad de la aplicación móvil. La pregunta 3 se refiere a la facilidad de uso de la aplicación, y obtuvo un promedio alto de (4.81/5) lo que significa que el usuario puede iniciar sesión en la aplicación de una manera sencilla y simple con solo presionar el botón de iniciar sesión y otorgar los permisos necesarios.

La pregunta 4 se refiere a la facilidad de navegación en el dashboard principal y obtuvo un promedio de (4.67/5) lo que significa que el usuario pudo navegar sin complicaciones por el dashboard.

La pregunta 5 se refiere a la facilidad de escoger una localización y obtuvo un promedio de (4.67/5) lo que significa que el usuario no tuvo dificultades al escoger un punto de interés del listado y visualizar la ubicación del lugar ya que la aplicación muestra la ruta desde la ubicación del usuario hasta la ubicación del punto de interés seleccionado.

La pregunta 6 se refiere a la facilidad para cerrar sesión y obtuvo un promedio de (4.76/5) lo que significa que el usuario pudo cerrar sesión de una manera muy simple y sencilla debido a que la aplicación tiene una interfaz de usuario intuitiva.

De lo anterior se evidencia que con estas cifras los usuarios no tuvieron complicaciones al hacer uso de la aplicación

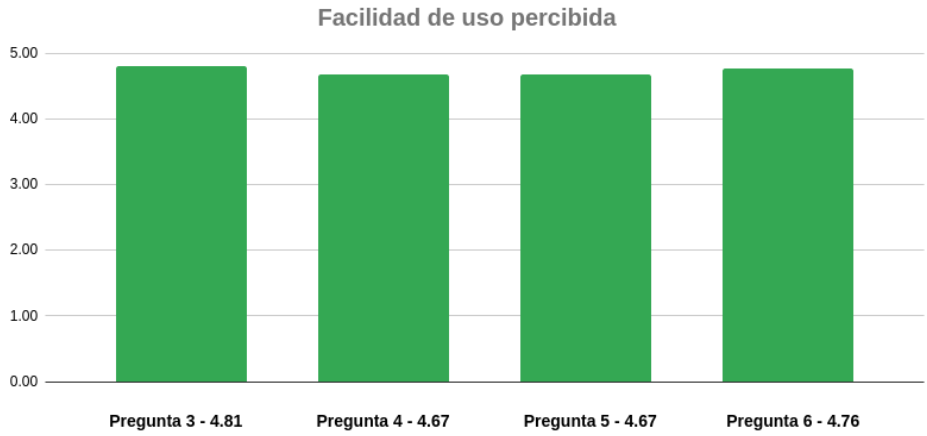

Figura 80. Promedio por pregunta de la facilidad de uso percibida en el producto final

Fuente: elaboración propia

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Mediante la red social Facebook y la otorgación de permisos se pudo acceder a las publicaciones del usuario, la cual consta de la información básica del usuario y los posts que tiene la información de la geolocalización (latitud y longitud), nombre del lugar que ha visitado, foto del lugar visitado, la reacción que ha dado a la publicación y el mensaje de la publicación. La API que permite acceder y extraer esta información fue implementado en la parte del Frontend. Los posts analizados son aquellos que cumplen con la condición de tener una imagen, la frase y la ubicación por lo tanto las que no cumplen con esta condición no son procesadas.
- Con el analizador de sentimientos de Google, las frases de los posts obtenidos muestran un score mayor a 0.7 (este umbral fue estudiado por el equipo de desarrollo y el scrum master llegando a la conclusión que este número puede ser considerado como positivo) lo cual indica que estos posts tienen una frase positiva que indica un gusto e interés del usuario, mientras los que no cumplen con esta condición son desechados.
- Las frases positivas obtenidas con el analizador de sentimientos son enviadas al servicio Natural Language Classifier de Google para predecir la categoría correspondiente a cada frase, con los resultados se obtiene un score de predicción por cada etiqueta, se los ordena de mayor a menor con de los cuales la primera posición por tener un score muy alto no se le agrega ningún peso, pero para la segunda, tercera y cuarta posición se le agrego los pesos de 0.5, 0.3 y 01 respectivamente, estos pesos fueron escogidos porque representan un valor neutro en las predicciones.
- Los resultados del análisis de predicciones indica la acumulación del score de predicción y los pesos entre el número de veces que la etiqueta estuvo entre la primera y cuarta posición, permitiendo dar un porcentaje a los lugares que le gusta al usuario.
- Mediante el uso de la herramienta de diseño de mockups se diseñó el prototipo de la aplicación móvil que consta de un total de 4 pantallas: Login, Home, Menu, Modal. Con ayuda del prototipo se desarrolló las interfaces de usuario, se implementó los diferentes servicios y se instaló los plugins para que la aplicación móvil pueda enviar los datos del usuario en formato JSON al Backend y recibir como respuesta los datos en formato JSON sobre los lugares de interés del usuario.

- Los puntos de interés analizados contienen una etiqueta que es enviada a la API de Google Maps para obtener como respuesta los sitios de interés del usuario dentro de un rango 1.5km. Los cuales son mostrados en la aplicación móvil para que el usuario pueda interactuar con estas.
- Para la evaluación de la aplicación móvil se generó un APK y se envió a los usuarios de prueba para que lo instalen en su dispositivo Android y prueben el funcionamiento de la aplicación TouristApp. Con ayuda de la encuesta de Usabilidad adjuntada en el Anexo III se determinó que la aplicación es fácil de usar ya que posee una interfaz amigable para el usuario. Así quedando muy satisfechos con los resultados y dando feedback para una actualización.

5.2 RECOMENDACIONES

Para trabajos futuros se recomienda

- Para que la aplicación ofrezca una mayor cantidad de lugares por visitar se recomienda la ampliación de los lugares turísticos con las etiquetas que ofrece Google Maps.
- Dar la facilidad al usuario en escoger una etiqueta, esto permitirá que el usuario tenga una mayor variedad de lugares que pueda visitar, aunque no sea de su interés, esto ayudará a que la aplicación móvil pueda mantenerse
- Ofrecer un sistema propio de calificación y comentarios del lugar, a pesar de que Google presenta estos valores, uno de los mayores problemas es que no son calificados de forma correcta y una característica así ayudará a agregar valor a la aplicación.
- Mostrar información del lugar como la descripción y los comentarios para que el usuario puede decidir de una manera más rápida y sencilla visitar el lugar.
- Agregar tiempo de visita, esto permitirá conocer si el usuario disfruto su instancia en el lugar.
- Se debe agregar el presupuesto, esto permitirá que los usuarios que visiten el lugar conocer si su presupuesto se ajusta al del lugar.
- Implementar el servicio de iniciar navegación en tiempo real de Google Maps en el componente Modal para que muestre las instrucciones necesarias al usuario sobre cómo llegar y encontrar la mejor ruta para su lugar de destino.
- Construir la aplicación móvil para producción y subirla a Google Play Console para que cualquier usuario pueda descargarla e instalarla en su dispositivo Android. Esto ayudaría a que la aplicación sea más conocida y llegue a muchos lugares.
- Realizar la configuración de los plugin de capacitor (Facebook login, geolocation, Location Accuracy) para construir la aplicación para el sistema operativo iOS. La aplicación debería ser cargada a la App Store para así llegar y dar a conocer TouristApp en nuevas plataformas.
- Implementar una base de datos que almacene los datos del usuario, las predicciones de las categorías y los puntos de interés para realizar un análisis y determinar patrones que ayudarían a mejorar la aplicación para que pueda ofrecer recomendaciones de lugares turísticos al usuario.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] "Facebook: What is Facebook?" https://edu.gcfglobal.org/en/facebook101/what-is-facebook/1/ (accessed Sep. 22, 2021).

[2] "Facebook: Política de datos." https://www.facebook.com/about/privacy/update (accessed Sep. 22, 2021).

[3] "Facebook Reactions: qué significa cada emoji y para qué usarlos | FOTOS | Foto 1 de 8 | EPIC Mobile | Epic | Peru.com," Feb. 24, 2016. https://peru.com/epic/epic-mobile/facebook-reactions-que-significa-cada-emoji-y-que-usarlos-fotos-noticia-441149] (accessed Sep. 22, 2021).

[4] "Reacciones a publicaciones | Servicio de ayuda de Facebook para empresas." https://www.facebook.com/business/help/118654155244100 (accessed Sep. 22, 2021).

[5] Alvino Clay, "Estadísticas de la situación digital de Ecuador en el 2020-2021 | Branch," May 05, 2021. https://branch.com.co/marketing-digital/estadisticas-de-la-situacion-digital-de-ecuador-en-el-2020-2021/ (accessed Sep. 22, 2021).

[6] F. Ricci, B. Shapira, and L. Rokach, *Recommender systems handbook, Second edition*. 2015.

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005, doi: 10.1109/TKDE.2005.99.

[8] G. G. I, T. D. Ii, and J. L. C. I, "Método para sistemas de recomendaciones sensibles al contexto en entornos turísticos," *Ing. Ind.*, vol. 38, no. 1, pp. 68–80, 2017.

[9] J. P. Lucas, "Métodos de clasificación basados en asociacion aplicados a sistemas de recomendación," *Tesis Dr. Univ. Salamanca*, 2010, [Online]. Available: https://www.mendeley.com/catalog/métodos-clasificación-basados-en-asociacion-aplicados-sistemas-recomendación/?utm_source=desktop&utm_medium=1.14&utm_campaign=open_catalog&userDocumentId=%7B0266e5d0-3c99-48c8-aa76-fda6ff9bbd18%7D.

[10] L. Rouhiainen, "Inteligencia artificial 101 cosas que debes saber," *Alienta Editor.*, p. 352, 2018, [Online]. Available: https://planetadelibrosar0.cdnstatics.com/libros_contenido_extra/40/39307_Inteligencia_artificial.pdf.

[11] C. C. Gil, "Trabajo de Fin de Grado Identificación y análisis de las principales tecnologías de AutoML," 2021.

[12] A. Gelbukh, "Procesamiento de Lenguaje Natural y sus Aplicaciones," vol. I, 2010.

[13] M. K. Dalal and M. A. Zaveri, "Automatic Text Classification: A Technical Review," *Int. J. Comput. Appl.*, vol. 28, no. 2, pp. 37–40, 2011, doi: 10.5120/3358-4633.

[14] "Visual Studio Code - Code Editing. Redefined." https://code.visualstudio.com/ (accessed Feb. 06, 2022).

[15] "WebStorm: The Smartest JavaScript IDE, by JetBrains."

https://www.jetbrains.com/webstorm/ (accessed Feb. 06, 2022).

[16] “GitKraken | Legendary Git Tools.” https://www.gitkraken.com/ (accessed Feb. 06, 2022).

[17] “Git.” https://git-scm.com/ (accessed Feb. 06, 2022).

[18] “GitHub Docs.” https://docs.github.com/en/get-started/quickstart/hello-world (accessed Feb. 06, 2022).

[19] “Android Studio and SDK tools | Android Developers.” https://developer.android.com/studio (accessed Feb. 06, 2022).

[20] “Postman API Platform.” https://www.postman.com/product/what-is-postman/ (accessed Feb. 06, 2022).

[21] “npm | javascript package manager.” https://docs.npmjs.com/cli/v6/commands/npm (accessed Feb. 06, 2022).

[22] “NinjaMock online wireframe and mockup tool.” https://ninjamock.com/home/index (accessed Feb. 06, 2022).

[23] “Trello.” https://trello.com/en (accessed Feb. 06, 2022).

[24] “Google Forms – create and analyse surveys, for free.” https://www.google.com/intl/en-GB/forms/about/ (accessed Feb. 06, 2022).

[25] “Cross-Platform Mobile App Development: Ionic Framework.” https://ionicframework.com/ (accessed Feb. 06, 2022).

[26] “Angular.” https://angular.io/ (accessed Feb. 06, 2022).

[27] “Node.js.” https://nodejs.org/en/ (accessed Feb. 06, 2022).

[28] “Capacitor by Ionic - Cross-platform apps with web technology.” https://capacitorjs.com/ (accessed Feb. 06, 2022).

[29] “Welcome to Flask — Flask Documentation (2.0.x).” https://flask.palletsprojects.com/en/2.0.x/ (accessed Feb. 06, 2022).

[30] “TypeScript: JavaScript With Syntax For Types.” https://www.typescriptlang.org/ (accessed Feb. 06, 2022).

[31] “Welcome to Python.org.” https://www.python.org/ (accessed Feb. 06, 2022).

[32] “HTML: HyperText Markup Language | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Feb. 06, 2022).

[33] “CSS: Cascading Style Sheets | MDN.” https://developer.mozilla.org/en-US/docs/Web/CSS (accessed Feb. 06, 2022).

[34] “Servicios de computación en la nube | Google Cloud.” https://cloud.google.com/ (accessed Feb. 06, 2022).

[35] “API de Google Cloud.” https://cloud.google.com/apis/docs/overview (accessed Feb. 06, 2022).

[36] “Cloud Natural Language | Google Cloud.” https://cloud.google.com/natural-

language?hl=es (accessed Feb. 06, 2022).

[37] “Guía para principiantes de AutoML .” https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide?hl=es#text_5 (accessed Feb. 06, 2022).

[38] “API de Cloud Natural Language: Qwik Start.” https://www.cloudskillsboost.google/focuses/582?locale=es&parent=catalog (accessed Feb. 06, 2022).

[39] “Google Maps Platform.” https://developers.google.com/maps/faq#whatis (accessed Feb. 06, 2022).

[40] “Información general - API Graph.” https://developers.facebook.com/docs/graph-api/overview (accessed Sep. 22, 2021).

[41] “Características de Amazon EC2 – Amazon Web Services.” https://aws.amazon.com/es/ec2/features/?trk=ec2_landing (accessed Feb. 06, 2022).

[42] P. Letelier and M. C. Penadés, “Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP),” *www.cyta.com.ar/ta0502/v5n2a1.htm*, Apr. 2006.

[43] D. E. Sistemas, “Escuela politécnica nacional,” 2020.

[44] “Lograr eficiencia con el pensamiento ‘Lean’ | Conexión ESAN.” https://www.esan.edu.pe/conexion-esan/lograr-eficiencia-con-el-pensamiento-lean (accessed Mar. 14, 2022).

[45] K. Schwaber, J. Sutherland, L. Guía, D. De Scrum, and L. Reglas, “La Guía Scrum,” 2020.

[46] E. J. Torres Martínez, E. C. Arzuza Agudelo, and O. F. Becerra Uribe, “Aplicación de la metodología SCRUM para la optimización de procesos académicos en la Universidad de San Buenaventura, Cartagena,” 2012, [Online]. Available: http://hdl.handle.net/10819/2353.

[47] “Scrum Burndown Chart - International Scrum Institute.” https://www.scrum-institute.org/Burndown_Chart.php (accessed Feb. 06, 2022).

[48] “Prepara tus datos de entrenamiento | Documentación de AutoML Natural Language .” https://cloud.google.com/natural-language/automl/docs/prepare (accessed Feb. 07, 2022).

[49] “Referencia de permisos - API Graph.” https://developers.facebook.com/docs/permissions/reference (accessed Feb. 07, 2022).

[50] “How to Install The Ionic Framework CLI to Build Mobile Apps.” https://ionicframework.com/docs/intro/cli (accessed Feb. 11, 2022).

[51] “Ionic Generate: Create Pages, Components, & Angular Features.” https://ionicframework.com/docs/cli/commands/generate (accessed Feb. 18, 2022).

[52] “Crear una app - Desarrollo de apps.” https://developers.facebook.com/docs/development/create-an-app/ (accessed Feb. 18, 2022).

[53] “capacitor-community/facebook-login: Facebook Login support.” https://github.com/capacitor-community/facebook-login (accessed Feb. 18, 2022).

[54] “Angular NG Build Configuration to Build Ionic Apps | ionic build.” https://ionicframework.com/docs/cli/commands/build (accessed Feb. 18, 2022).

[55] “Run your app | Android Developers.” https://developer.android.com/training/basics/firstapp/running-app (accessed Feb. 11, 2022).

[56] “Cree aplicaciones web y sitios web de manera gratuita.” https://aws.amazon.com/es/free/webapps (accessed Feb. 08, 2022).

[57] “Capa gratuita de AWS.” https://aws.amazon.com/es/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free Tier Types=*all&awsf.Free Tier Categories=*all (accessed Feb. 08, 2022).

[58] “AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube.” https://aws.amazon.com/es/ec2/?p=ft&c=wa&z=2 (accessed Feb. 08, 2022).

[59] “Installation | Ubuntu.” https://ubuntu.com/server/docs/installation (accessed Mar. 08, 2022).

[60] “Módulo 2: Crear una instancia EC2.” https://aws.amazon.com/es/getting-started/hands-on/deploy-wordpress-with-amazon-rds/3/ (accessed Feb. 08, 2022).

[61] “SSH protocol is the standard for strong authentication, secure connection, and encrypted file transfers. We developed it.” https://www.ssh.com/academy/ssh/protocol (accessed Mar. 08, 2022).

[62] “virtualenv · PyPI.” https://pypi.org/project/virtualenv/ (accessed Feb. 08, 2022).

[63] “Flask · PyPI.” https://pypi.org/project/Flask/ (accessed Feb. 08, 2022).

[64] “Place Types | Google Developers.” https://developers.google.com/maps/documentation/places/web-service/supported_types?fbclid=IwAR3yJkxB4MjwyAkmEMuZHg1DrB-MS4_lsoGKK-o9XTr5du-nmiXEokLtDqQ#table1 (accessed Feb. 13, 2022).

[65] “Natural Language – Tourist-App – Google Cloud Platform.” https://console.cloud.google.com/natural-language/locations/us-central1/datasets/TCN5534585842212798464;modelId=TCN6317615692317196288/evaluate?project=tourist-app-333203 (accessed Apr. 24, 2022).

[66] “Analizar opiniones | API de Cloud Natural Language.” https://cloud.google.com/natural-language/docs/analyzing-sentiment (accessed Feb. 16, 2022).

[67] “Conceptos básicos de la API de Natural Language | API de Cloud Natural Language.” https://cloud.google.com/natural-language/docs/basics#interpreting_sentiment_analysis_values (accessed Feb. 16, 2022).

[68] “Idiomas compatibles | API de Cloud Natural Language.” https://cloud.google.com/natural-language/docs/languages (accessed Feb. 16, 2022).

[69] “Use API Keys | Maps Static API.” https://developers.google.com/maps/documentation/maps-static/get-api-key (accessed Feb. 28, 2022).

[70] “Google Maps Platform Documentation | Places | Routes.” https://developers.google.com/maps/documentation (accessed Feb. 28, 2022).

[71] “API y servicios – Google Cloud Platform.” https://console.cloud.google.com/apis/credentials/key/b6dbc0d2-67e0-400e-a006-4e8588a35836?project=tourist-app-333203 (accessed Apr. 24, 2022).

[72] “Métodos de petición HTTP - HTTP.” https://developer.mozilla.org/es/docs/Web/HTTP/Methods (accessed Feb. 28, 2022).

[73] “Angular - HttpClient.” https://angular.io/api/common/http/HttpClient#description (accessed Feb. 28, 2022).

[74] “API del complemento Capacitor de geolocalización - Capacitor.” https://capacitorjs.com/docs/apis/geolocation (accessed Feb. 23, 2022).

[75] “Location Accuracy | Ionic Documentation.” https://ionicframework.com/docs/native/location-accuracy (accessed Feb. 28, 2022).

[76] “How to Switch on Device GPS in Ionic 5 without Leaving App - positronX.io.” https://www.positronx.io/switch-on-device-gps-in-ionic-without-leaving-application/ (accessed Feb. 28, 2022).

[77] “@agm/core.” https://angular-maps.com/api-docs/agm-core/ (accessed Feb. 28, 2022).

[78] “Direction | Agm-Direction-Docs.” https://robby570.tw/Agm-Direction-Docs/source/featured/direction.html (accessed Feb. 28, 2022).

[79] “Getting started with Angular Google Maps (AGM).” https://angular-maps.com/guides/getting-started/ (accessed Feb. 28, 2022).

[80] “What is APK file (Android Package Kit file format)? - Definition from WhatIs.com.” https://whatis.techtarget.com/definition/APK-file-Android-Package-Kit-file-format (accessed Feb. 28, 2022).

[81] “Build and run your app | Android Developers.” https://developer.android.com/studio/run (accessed Feb. 28, 2022).

[82] “Nearby Search | Places API.” https://developers.google.com/maps/documentation/places/web-service/search-nearby#maps_http_places_nearbysearch-py (accessed Feb. 28, 2022).

[83] “Habilitación de CORS para un recurso de la API de REST - Amazon API Gateway.” https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/how-to-cors.html (accessed Feb. 28, 2022).

[84] “usesCleartextTraffic.” https://developer.android.com/guide/topics/manifest/application-element#usesCleartextTraffic (accessed Feb. 28, 2022).

7. ANEXOS

7.1 Anexo I: Archivo CSV con las frases y sus respectivas categorías

Que lindo paisaje que presenta esta montaña,park
Déjate deslumbrar por su cultura expresada majestuosamente en sus Playas,tourist_attraction
Playa San Lorenzo en Manta. Provincia de Manabí,tourist_attraction
Su gran diversidad natural y cultural abarca páramos y bosques nublados,park
Muy bonito paisaje en esta montaña de nuestro Ecuador,park
Bella vista del volcán,park
Bellísimo San Andrés imponente por su templo y su nevado.,park
LINDO MI PUEBLITO SAN ANDRES CON SU MAJESTUOSO CHIMBORAZO,park
El Volcan Carihuairazo,park
El volcán más hermoso del Ecuador,park
Nombrado como Patrimonio Cultural del Estado en Tulcán,church
Es un lugar espectacular. La hermosura de las figuras es algo increíble.,museum
Es único. Bella obra de arte,church
Es una verdadera obra de Arte ,church
Muy bonito es verdaderamente una obra de arte,church
Galápagos en la Isla San Cristóbal,park
Vista del islote Leon Dormido en Galápagos,park
Una de las mejores playas de Manabí,tourist_attraction
Hermoso bello Manabi kilometros de playas.,tourist_attraction
Reserva de Producción de Fauna Chimborazo,park
Vista de los Paisajes en Sozoranga,park
Vista de los Paisajes en Sibambe,park
Playa Los Frailes,tourist_attraction
Parque Nacional Machalilla,park
Una piña colada a orillas del mar ,tourist_attraction
Déjate deslumbrar por la hermosura de estas playas,tourist_attraction
Hermosa Playa en nuestro pais,tourist_attraction
Playa Rosada en Santa Elena ,tourist_attraction
Vista de la Laguna de Busa,park
Vista de la Lagunas De Atillo en el Parque Nacional Sangay ,park
Unas lagunas hermosas y con mucha historia,park
Muy hermosas ese conjunto de lagunas,park
Las playas de Pedernales en Provincia de Manabi ,tourist_attraction
Precioso volcán de mi lindo Ecuador,park
Ven a Loja y disfruta las Fiestas de Las Artes Vivas,museum

Figura 81. Archivo CSV final

Fuente: elaboración propia

7.2 Anexo II: Mockups de la aplicación TouristApp

Figura 82. Mockup Login

Fuente: elaboración propia

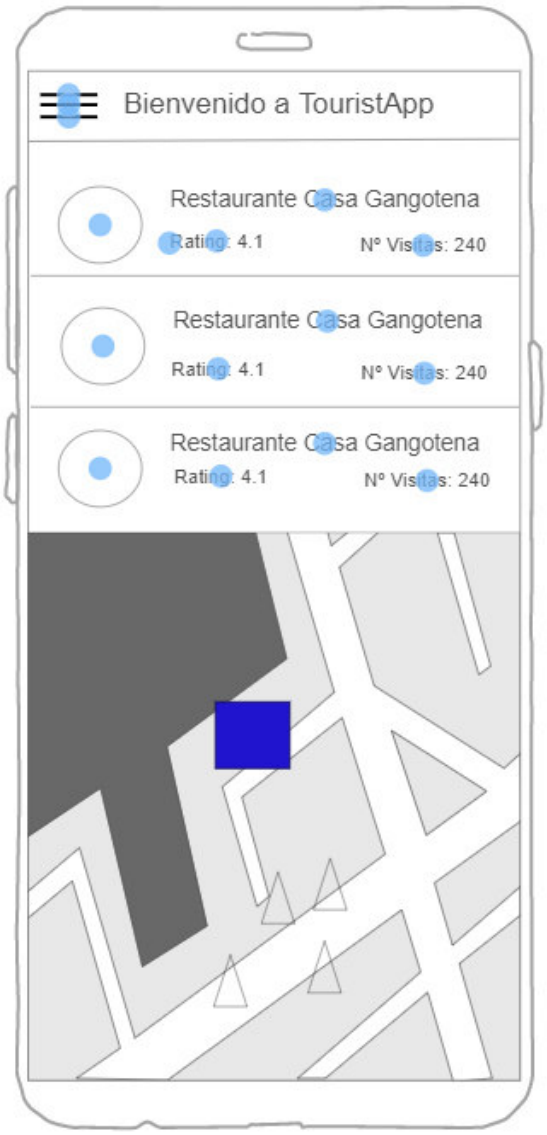

Figura 83. Mockup Home

Fuente: elaboración propia

Figura 84. Mockup Menu

Fuente: elaboración propia

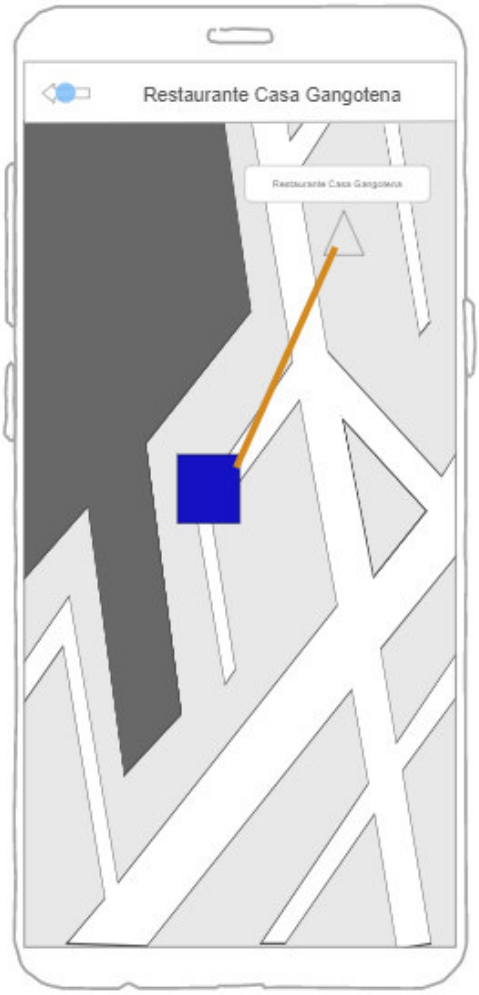

Figura 85. Mockup Ubicacion

Fuente: elaboración propia

7.3 Anexo III: Encuesta de precisión del interés y la usabilidad de la aplicación

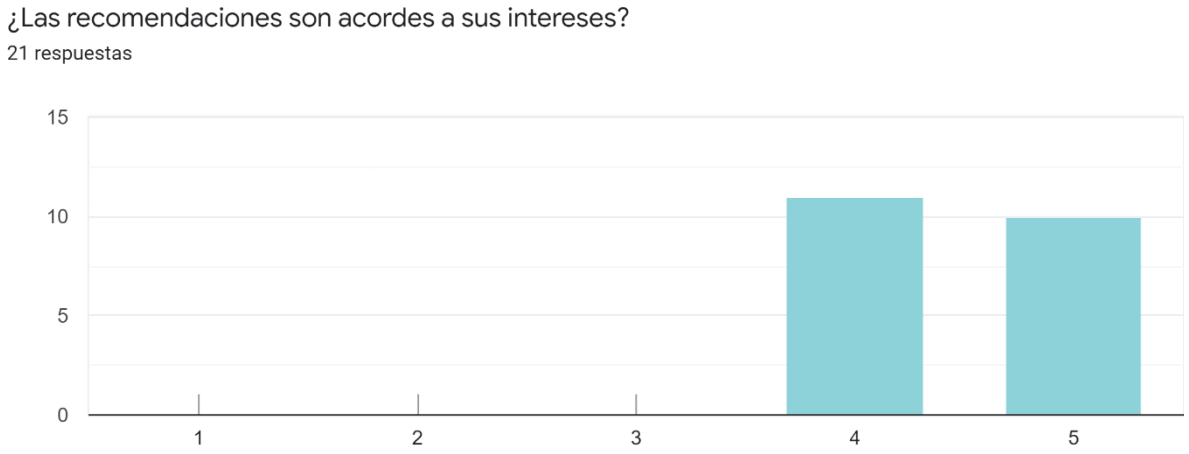

Figura 86. Encuesta precisión de interés pregunta 1

Fuente: Google Forms [24]

Figura 87. Encuesta precisión de interés pregunta 2

Fuente: Google Forms [24]

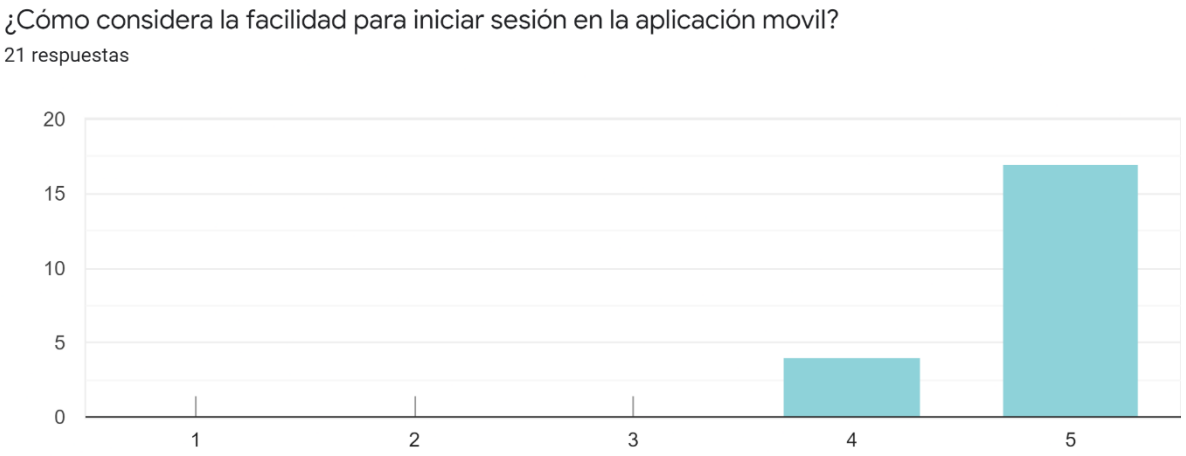

Figura 88. Encuesta de Usabilidad pregunta 1

Fuente: Google Forms [24]

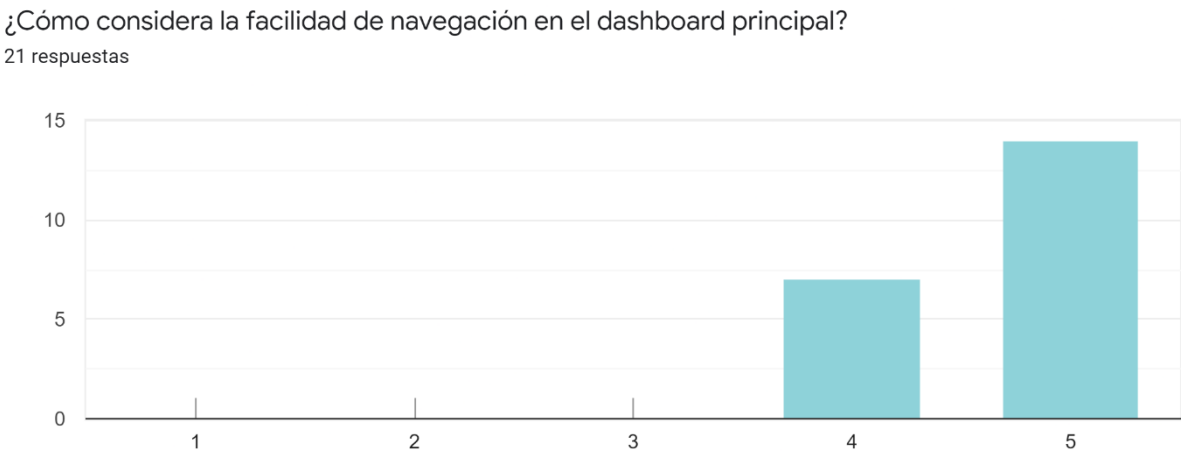

Figura 89. Encuesta de Usabilidad pregunta 2

Fuente: Google Forms [24]

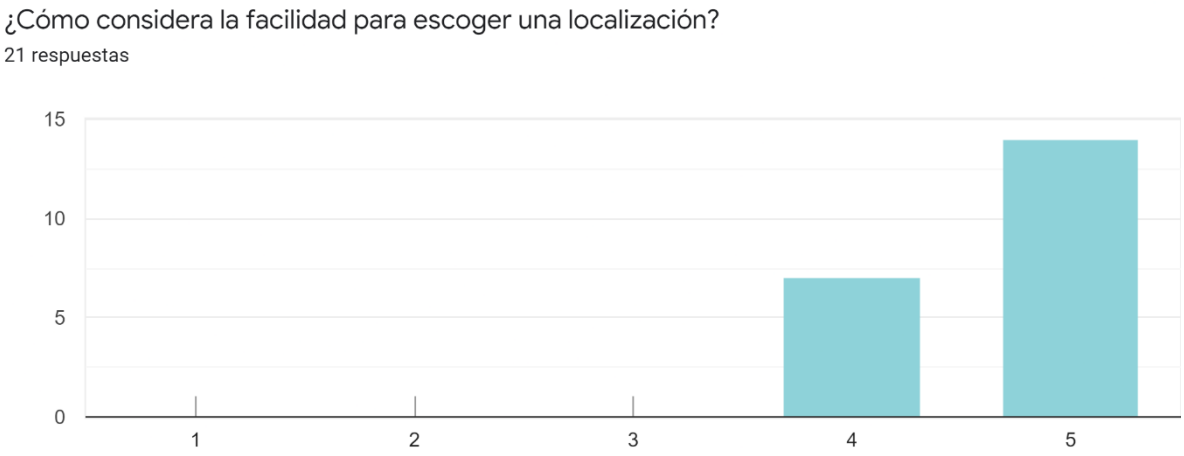

Figura 90. Encuesta de Usabilidad pregunta 3

Fuente: Google Forms [24]

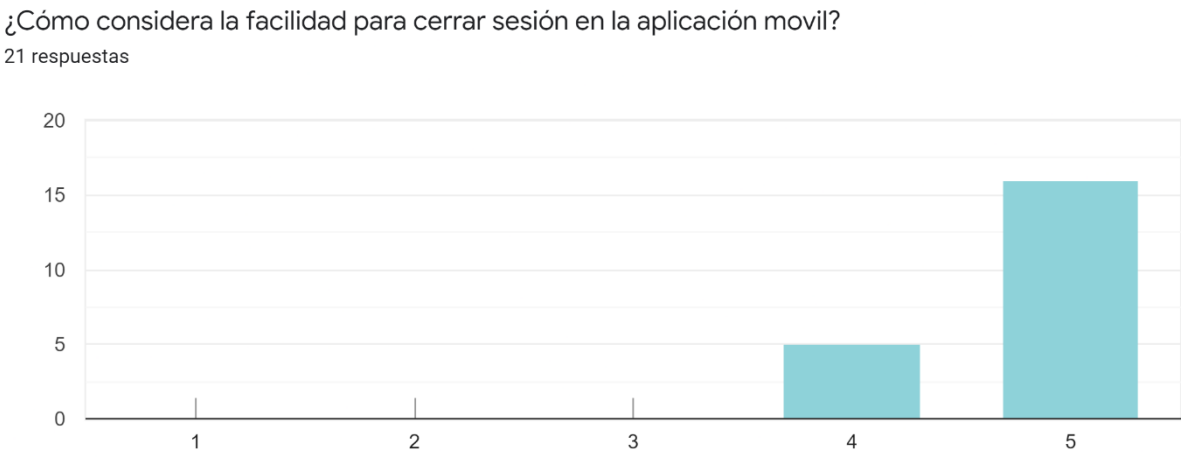

Figura 91. Encuesta de Usabilidad pregunta 4

Fuente: Google Forms [24]

7.4 Anexo IV: Costos del proyecto

1. Datos previos

Para realizar los cálculos del costo del proyecto se deberá utilizar los siguientes datos:

- Scrum Master: 1
- Desarrollador Backend: 1
- Desarrollador Frontend: 1
- Costo por hora: $ 3.125 (160 horas laborables al mes)
- Diario: 8 horas solo días laborables
- Inicio: octubre del 2021
- Fin: diciembre del 2021

2. Cálculo del costo por sprint

En la Tabla 46, se detalla el tiempo aproximado que se ocupó para el desarrollo de cada Sprint y posteriormente se realizó el cálculo del costo para cada Sprint en base a la información detallada en la sección datos previos.

Sprint	Historia de usuario	Descripción	Tiempo (días)	Costo
Sprint 1	TA01-01	Creación del repositorio en Github.	1	$25,00
	TA01-02	Creación del proyecto con Ionic	2	$50,00
	TA01-03	Creación de la aplicación en Facebook Developers	2	$50,00
	TA01-04	Implementación de la API de Facebook.	7	$175,00
	TA02-01	Selección del servicio AWS	8	$200,00
	TA02-02	Implementación de Framework Flask	3	$75,00
Sprint 2	TA03-01	Estudiar las categorías aceptadas para realizar la búsqueda en Google Maps	1	$25,00

	TA03-02	Preparar los datos obtenidos con las categorías en la parte de recolección de datos para el NLC	2	$50,00
	TA03-03	Entrenar los datos en el NLC de Google obtenidos en la anterior tarea	1	$25,00
	TA03-04	Realizar pruebas en el modelo generado por NLC	0,5	$12,50
	TA04-01	Implementar el analizador de sentimientos	1,5	$37,50
	TA04-02	Implementar el Natural Language Classifier en el servidor de AWS	1,5	$37,50
	TA04-03	Realizar peticiones http al servidor de AWS para verificar los resultados de los servicios implementados	1	$25,00
	TA05-01	Diseñar los mockups para la aplicación móvil	1	$25,00
	TA05-02	Desarrollar las interfaces de usuario en base a los mockups	1,5	$50,00
	TA05-03	Filtración y limpieza de los datos extraídos del usuario cuando inicia sesión con Facebook	1	$25,00
Sprint 3	TA06-01	Integración de la API de Google Maps	1	$25,00
	TA06-02	Obtener los mejores lugares turísticos dependiendo la categoría analizada por el NLC	3	$75,00
	TA07-01	Crear servicios para enviar los datos del usuario al BackEnd mediante HTTP	2	$50,00
	TA07-02	Configurar el servidor para procesar solicitudes con contenido JSON	3	$75,00
	TA08-01	Solicitar la ubicación actual del usuario antes de iniciar sesión	4	$100,00

	TA08-02	Desarrollar la interfaz de usuario para mostrar resultados	9	$225,00
			Subtotal	**$1.437,50**

Tabla 46. Cálculo del costo por sprint

Fuente: elaboración propia

2. Cálculo de costos de soporte y mantenimiento

En la Tabla 47, se detallan los costos correspondientes al soporte y mantenimiento del proyecto.

Alojamiento	**Observación**	**Costo**
Google Play	Pago único	$ 25
App Store	Pago anual	$ 99
AWS	Pago mensual	$ 20
AutoML	Pago mensual	$ 33,60
Google Maps	Pago mensual	$ 21,70
	Subtotal	**$199,30**

Tabla 47. Detalle de costos estimados de soporte y mantenimiento del sistema

Fuente: elaboración propia

3. Cálculo de sueldos

En la Tabla 48, se detallan los costos correspondientes a los sueldos de las personas comprometidas a llevar a cabo el proyecto.

Empleado	**Sueldo mensual**	**Personas**	**Costo mensual**
Scrum master	$ 600.00	1	$ 600.00
Desarrollador Backend	$ 500.00	1	$ 500.00
Desarrollador Frontend	$ 500.00	1	$ 500.00
Tester	$ 100.00	1	$ 100.00
		Subtotal	**$ 1.700,00**

Tabla 48. Detalle de costos estimados de sueldos

Fuente: elaboración propia

4. Cálculo de servicios básicos

En la Tabla 49, se detallan los costos de los servicios básicos.

Servicio	Costo Mensual
Agua	$ 15.00
Luz	$ 30.00
Internet Plan 45 mbps	$ 36.96
Subtotal	**$81.96**

Tabla 49. Detalle de costos estimados de servicios básicos

Fuente: elaboración propia

5. Cálculo de equipos

En la Tabla 50, se detallan los costos referentes a los equipos utilizados para el desarrollo del proyecto.

Equipo	Cantidad	Costo
Laptop MSI Core I7 9va 16gb 500 SSD Linux Ubuntu 20.0.4	1	$ 1625.00
Laptop Dell Core I7 11va 8gb 256g SSD Windows 10	1	$ 899.00
Escritorio	2	$ 55.00
Silla oficina	2	$ 64.99
	Subtotal	**$ 2643.99**

Tabla 50. Cálculo del costo total de equipos

Fuente: elaboración propia

6. Costo total del proyecto

En la Tabla 51, se procedió a determinar el costo total del proyecto, en base a los subtotales obtenidos en las secciones anteriores. Obteniendo un costo final de $10025.27.

Tipo de costo	Costo	Cantidad	Costo Total
Cálculo del costo por sprint	$ 1.437,50	1	$ 1.437,50
Alojamientos	$ 199,30	3	$ 597.90
Sueldos	$ 1.700,00	3	$ 5100
Servicios básicos	$ 81.96	3	$245.88
Equipos	$ 2643.99	1	$ 2643.99
	TOTAL		**$ 10025.27**

Tabla 51. Cálculo del costo total del proyecto

Fuente: elaboración propia