ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA RECOMENDADOR DE UBICACIONES PARA SITIOS TURÍSTICOS DEL CENTRO HISTÓRICO DE QUITO BASADO EN PROCESAMIENTO DE LENGUAJE NATURAL

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

LIMAICO MERA JOSÉ FRANCISCO
jose.limaico@epn.edu.ec

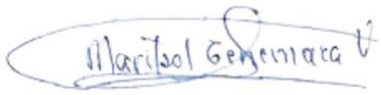
DIRECTORA: Ing. Regina Maritzol Tenemaza Vera, MSc.
maritzol.tenemaza@epn.edu.ec

CODIRECTOR: Ing. Vicente Adrián Eguez Zarsoza, MSc.
adrian.eguez@epn.edu.ec

QUITO, AGOSTO 2020

AVAL

Certificamos que el presente trabajo fue desarrollado por José Francisco Limaico Mera, bajo nuestra supervisión.



MSc. Maritzol Tenemaza

DIRECTORA

MSc. Adrián Eguez

CODIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, José Francisco Limaico Mera, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentada para ningún grado o calificación profesional; y, además, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis 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.

José Francisco Limaico Mera

DEDICATORIA

A mis padres por siempre apoyarme en todas las decisiones, a pesar de que estas parecieran descabelladas.

A mi familia por brindarme su apoyo incondicional y por preocuparse por mí.

A mis amigos de la Escuela Politécnica Nacional por su amistad durante esta etapa universitaria.

AGRADECIMIENTO

Un sincero agradecimiento a mi directora del proyecto MSc. Maritzol Tenemaza por brindarme su apoyo y confianza para desarrollar este proyecto y a mi codirector MSc. Adrián Eguez por guiarme en la implementación práctica de este proyecto.

ÍNDICE DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Resumen

Los sistemas de recomendación que utilizan métodos tradicionales para la clasificación y generación de recomendaciones pueden ser mejorados aplicando técnicas de aprendizaje profundo con técnicas de aprendizaje automático. El propósito del presente trabajo es brindar una solución para mejorar la eficacia de las recomendaciones de estos sistemas, mediante el procesamiento de lenguaje natural. Para lograr este objetivo, se entrenó un modelo de procesamiento de lenguaje natural que usa enfoques de aprendizaje profundo. El desarrollo del modelo está compuesto por dos fases: a) la fase de recolección de datos y b) la fase de desarrollo.

En primer lugar, en la fase de recolección de datos se obtuvo el conjunto de datos de entrenamiento a partir de encuestas realizadas sobre consultas de sitios turísticos con su respectiva categoría. En contraste, en la fase de desarrollo se entrenó el modelo empleando el conjunto de datos recopilados por medio de la implementación de un evento de entrenamiento ofrecido por el servicio de IBM, denominado Watson Natural Language Classifier (NLC). El modelo desarrollado permitió clasificar sentencias de texto en 62 categorías producidas por medio del evento de entrenamiento.

Para probar el modelo mencionado se desarrolló un sistema que brinda recomendaciones de sitios turísticos para el centro histórico de Quito. De acuerdo con esto, el sistema emplea el servicio NLC como su principal fuente de procesamiento de sentencias de texto. Como resultado se compilaron recomendaciones de puntos de interés asociados a sitios turísticos que se ajustaban a las preferencias del usuario.

Palabras clave: procesamiento de lenguaje natural, aprendizaje automático, aprendizaje profundo, puntos de interés, sistemas de recomendaciones.

ABSTRACT

Recommendation systems that use traditional methods for classifying and generating recommendations can be improved by using deep learning techniques in conjunction with machine learning techniques. The purpose of this work is to provide a solution to improve the effectiveness of the recommendations of these systems, using natural language processing. To achieve this goal, a natural language processing model using deep learning approaches was trained.

The development of the model is made up of two phases: a) the data collection phase and b) the development phase. In the data collection phase, the training data set was obtained from surveys carried out on consultations of tourist sites with their respective category. In the development phase, the model was trained using the data set obtained through the implementation of a training event offered by the IBM service called Watson Natural Language Classifier (NLC). The developed model allows classifying text sentences in 62 categories obtained through the training event.

To test the aforementioned model, a system was developed that provides recommendations for tourist sites for the Historic Center of Quito. This system uses the NLC service as its main source for processing text statements. As a result, recommendations of points of interest associated with tourist sites that fit the user's preferences were obtained.

Keywords: natural language processing, machine learning, deep learning, points of interest, recommendation systems.

1 INTRODUCCIÓN

1.1 Problema

El Procesamiento de Lenguaje Natural (PLN) es una gama de técnicas computacionales motivadas por la teoría para el análisis automático y la representación del lenguaje humano [1]. Durante décadas, los enfoques de aprendizaje automático, los cuales apuntan a problemas de PLN, se han basado en modelos poco profundos como máquinas de vectores de soporte o Support Vector Machines (SVM) [2]. Por otro lado, las arquitecturas y algoritmos de aprendizaje profundo han logrado avances impresionantes en campos como la visión por computadora y el reconocimiento de patrones, utilizando modelos de aprendizaje profundo como Redes Neuronales Convolucionales (RNC), siendo estos últimos también aplicables a tareas de PLN [3].

Recientemente, el aprendizaje profundo ha estado cambiando la arquitectura para las recomendaciones dramáticamente y brindando más oportunidades para mejorar el rendimiento (recuperación, precisión, etc.) de los sistemas de recomendación [4]. Collobert y Weston [5] demostraron avances recientes en sistemas de recomendación basados en aprendizaje profundo, los cuales han ganado una atención significativa al superar los obstáculos de los modelos convencionales y lograr recomendaciones de alta calidad. Además, este tipo de aprendizaje capta las intrincadas relaciones dentro de los datos en sí, mostrando mejoras significativas en modelos que manejan abundantes fuentes de datos como información contextual, textual y visual [4].

En la industria turística, los sistemas de recomendación son herramientas esenciales para mejorar la experiencia del usuario y promover las ventas o servicios para muchos sitios turísticos. Cabe destacar que el turismo aporta una contribución directa del 2,2 % al PIB del Ecuador [6]. Según cifras oficiales obtenidas [6], el centro histórico de Quito es uno de los sitios más visitados por los turistas con un 67 % de visitas. De esto también se concluye que Quito es la primera ciudad declarada Patrimonio Cultural de la Humanidad por la Unesco en 1978. De igual manera, en el 2018 se volvió a ratificar como líder del turismo sudamericano al ganar por sexta vez el premio World Travel Awards (WTA) 2018 considerado como el Óscar del turismo [7].

En este estudio se desarrolló un modelo de PLN que emplea enfoques de aprendizaje profundo por medio del servicio de IBM Watson Natural Language Classifier (NLC). Para probar su funcionamiento, se desarrolló un sistema de recomendaciones de sitios turísticos para el centro histórico de la ciudad de Quito, donde se usa este servicio como principal fuente de PLN.

1.2 Pregunta de investigación

¿Cuál es el modelo de PLN que permite recomendar sitios turísticos para el centro histórico de Quito, a partir del análisis del lenguaje natural sobre rastros de contenido dejado por el usuario?

1.3 Objetivo general

- Desarrollar un sistema de recomendación de sitios turísticos basado en el análisis de intereses del turista a través del PLN.

1.4 Objetivos específicos

- Utilizar el servicio de PLN de IBM NLC para clasificar los intereses del usuario a partir de encuestas realizadas.
- Determinar puntos de interés (POIs, por su nombre en inglés Points Of Interest) a partir de los gustos del turista, con el fin de presentarlos en un sistema de recomendaciones.
- Optimizar los POIs con rutas ajustadas a intereses del turista y su posición.
- Analizar, diseñar y construir el sistema de recomendaciones para presentar al turista los itinerarios generados.

1.5 Hipótesis

El conocimiento del usuario permite adaptar información para diferentes realidades. Con la aplicación del análisis del lenguaje natural sobre rastros de contenido dejados por el usuario, se pueden identificar intereses del usuario para proponer itinerarios de visita al centro histórico de Quito.

2 MARCO TEÓRICO

2.1 Estado del arte

En los últimos años, una serie de investigaciones acerca de sistemas de recomendación han sido presentadas. Por ejemplo, Su y Khoshgoftaar [8] expusieron una revisión exhaustiva sobre técnicas de filtrado colaborativo enfocándose en los sistemas de recomendación tradicionales. Por su parte, Burke [9] propuso una revisión sobre el sistema de recomendación híbrido que consiste en una combinación de los sistemas basados en filtrado colaborativo y los sistemas basados en contenido; Fernández-Tobías et al. [10] y Khan et al. [11] revisaron los modelos de recomendación entre dominios, los cuales buscan acoplar recomendaciones personalizadas de artículos entre diferentes dominios o categorías, por ejemplo, sugerir una película y una canción en particular al mismo usuario, por nombrar algunos.

Por otro lado, las investigaciones acerca de los sistemas de recomendación basados en el aprendizaje profundo son escasas. En el momento, solo existen dos investigaciones relacionadas que se han publicado formalmente. Tilahun et al. [12] introdujeron tres modelos de recomendación basados en el aprendizaje profundo y, aunque los tres modelos son influyentes en esta área de investigación, solo se enfocan en su mayoría en el filtrado colaborativo [13] [14] [15]. Liu y Caihua [16] revisaron 13 documentos sobre aprendizaje profundo aplicado a los sistemas de recomendaciones y propusieron clasificar estos modelos con base en la forma de entradas (enfoques que usan información de contenido y enfoques sin información de contenido) y productos (calificación y clasificación).

En la revisión realizada por Zhang et al. [4] se tuvo en cuenta la influencia del aprendizaje profundo y se demostró su eficacia cuando se aplica a la investigación de sistemas de recomendación. Esta proporciona el diseño de una taxonomía de modelos de recomendación basados en aprendizaje profundo junto con un resumen exhaustivo del estado del arte. De este modo, se ampliaron las tendencias actuales y se brindaron nuevas perspectivas que sirvieron como base para el desarrollo del presente proyecto.

De acuerdo con la creciente popularidad, el potencial del aprendizaje profundo aplicado en los sistemas de recomendación y, a su vez, los resultados de investigaciones anteriores, se optó por el desarrollo de un modelo de clasificación de lenguaje natural que maneja técnicas de aprendizaje profundo como servicio. Esta herramienta se utilizó para procesar las consultas realizadas por los usuarios en un sistema de recomendaciones de sitios turísticos para el centro histórico de la ciudad de Quito. Con su uso se comprobó su

funcionamiento y se evidenciaron los resultados del aprendizaje profundo en los sistemas de recomendación, como se indica en las investigaciones mencionadas.

2.2 Tipos de sistemas de recomendaciones

Los sistemas de recomendación consideran la preferencia de los usuarios sobre los elementos y, asimismo, recomiendan de forma proactiva los elementos que les gustan a los usuarios. Los modelos de recomendación se han desarrollado a partir de varios enfoques como son: calificación, contenido u otro conocimiento. Además, estos enfoques se clasifican generalmente en tres categorías: filtrado colaborativo, basado en contenido e híbrido [17].

2.2.1 Sistemas de recomendaciones basados en filtrado colaborativo.

Los sistemas de filtrado colaborativo proporcionan al usuario recomendaciones de artículos basados en sus gustos anteriores y en las preferencias de otros consumidores, quienes tenían gustos similares, según lo medido por las calificaciones. Por consiguiente, al usuario se le recomendarán artículos que eran predilectos para personas con preferencias similares en el pasado [18].

2.2.2 Sistemas de recomendaciones basados en contenido.

En cambio, los sistemas basados en contenido no utilizan ningún dato de preferencia y proporcionan recomendaciones directamente basadas en la similitud de elementos. Al usuario se le recomendarán elementos similares a los que este prefirió en el pasado. Estos sistemas recomiendan un artículo a un consumidor en función de una descripción del artículo y un perfil de los intereses de este [15].

2.2.3 Métodos híbridos.

Varios sistemas de recomendación emplean un enfoque híbrido al combinar métodos colaborativos y basados en contenido, lo que ayuda a evitar ciertas limitaciones de los sistemas mencionados. Las diferentes formas de combinar estos métodos se pueden clasificar de la siguiente manera [19]:

- Implementar métodos colaborativos y basados en contenido por separado y combinar sus predicciones;
- incorporar algunas características basadas en contenido en un enfoque colaborativo;

- incorporar algunas características de colaboración en un enfoque basado en contenido; y
- construir un modelo unificador general que incorpore características basadas en contenido y colaborativas.

2.3 Aprendizaje profundo

El aprendizaje profundo es un subconjunto del aprendizaje automático o aprendizaje de máquina en el que las redes neuronales (algoritmos inspirados en el cerebro humano) aprenden de grandes cantidades de datos. Los algoritmos de aprendizaje profundo realizan tareas con frecuencia y mejoran gradualmente el resultado, gracias a capas profundas que permiten el aprendizaje progresivo. Cabe resaltar que el aprendizaje profundo forma parte de una familia más amplia de métodos de aprendizaje automático basados en redes neuronales [20].

2.3.1 Arquitecturas que utilizan aprendizaje profundo.

La cantidad de arquitecturas y algoritmos que se manejan en el aprendizaje profundo es amplia y variada. A continuación, se describen las aplicaciones de cinco de las arquitecturas de aprendizaje profundo más importantes, desarrolladas en los últimos veinte años (ver Figura 1). Conviene destacar que LSTM (Long Short-Term Memory) y CNN (Convolutional Neural Network) (ver Tabla 1) son dos de las orientaciones más antiguas de esta lista, pero también dos de los más usadas en diversas aplicaciones.

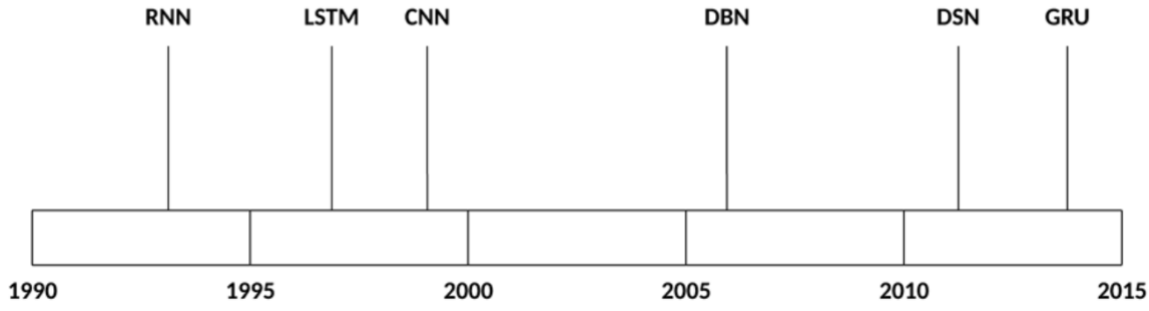


Figura 1. Línea del tiempo de creación de las arquitecturas basadas en aprendizaje profundo

Fuente: «IBM Developer, Deep learning architectures- IBM Developer», [En línea]. Available: https://developer.ibm.com/technologies/deep-learning/articles/cc-machine-learning-deep-learning-architectures. [Último acceso: junio 13 2020].

Para más detalle, en la Tabla 1 se presenta un resumen de las arquitecturas de aprendizaje profundo más empleadas, las cuales han sido desarrolladas en los últimos veinte años.

Dentro de estas se encuentra la arquitectura aplicada a redes neuronales convolucionales, debido a que su enfoque se puede aplicar al procesamiento de imágenes y, según recientes estudios [1] [12], al PLN. En correspondencia con ello, en este proyecto se utiliza el enfoque de aprendizaje profundo en su algoritmo de procesamiento.

Arquitectura	Aplicación
Redes Neuronales Recurrentes (Recurrent Neural Networks-RNN)	Reconocimiento de voz, reconocimiento de escritura.
Redes LSTM/GRU	Comprensión de texto en lenguaje natural y reconocimiento de escritura a mano, de voz, de gestos y subtítulos de imágenes.
Redes Neuronales Convolucionales (Convolutional Neural Networks-CNN)	Reconocimiento de imágenes, análisis de video y PLN.
Redes de Creencias Profundas (Deep Belief Networks-DBM)	Reconocimiento de imágenes, recuperación de información, comprensión del lenguaje natural, predicción de fallas.
Redes de Apilamiento Profundo (Deep Stacking Networks-DSN)	Recuperación de información, reconocimiento continuo de voz.

Tabla 1. Arquitecturas utilizadas en aprendizaje profundo y su aplicación

Fuente: IBM Developer, «Deep learning architectures-IBM Developer», [En línea]. Available: https://developer.ibm.com/technologies/deep-learning/articles/cc-machine-learning-deep-learning-architectures. [Último acceso: junio 13 2020].

2.4 Servicios de procesamiento de lenguaje natural

Debido a la creciente demanda de PLN enfocado en modelos de aprendizaje profundo, algunas empresas de servicios tecnológicos han desarrollado y ofrecido como servicio sus propias aplicaciones de clasificación y PLN. La principal característica de estos radica en que manejan técnicas de aprendizaje profundo, como las mencionadas en la Tabla 1, junto con técnicas de aprendizaje automático en sus algoritmos de procesamiento para realizar los eventos de clasificación. A continuación, se explican los más conocidos actualmente.

2.4.1 IBM Watson Natural Language Classifier.

En el núcleo del PLN se encuentra la clasificación de texto. En contraste, el NLC tiene las siguientes características: permite a los usuarios clasificar el texto en categorías personalizadas; confiere crear modelos personalizados a partir de un conjunto de datos de entrenamiento, para obtener categorías a escala; y combina varias técnicas avanzadas de aprendizaje de máquina con el fin de proporcionar la mayor precisión posible, sin requerir muchos datos de entrenamiento.

Detrás de escena, el NLC utiliza un conjunto de modelos de clasificación, junto con técnicas de aprendizaje supervisadas y sin supervisión, para lograr sus niveles de precisión.

Después de que se reúnen datos de entrenamiento, NLC evalúa estos datos contra múltiples SVM y una CNN mediante el aprendizaje profundo como servicio (DLaaS) provisto por IBM [22].

Se puede utilizar el NLC en muchas aplicaciones y sectores diferentes; estos son algunos ejemplos [23]:

- Banca y finanzas: clasifica inversiones, riesgos y transacciones.
- Educación superior y gobierno: ordena texto o documentos en categorías. Es útil para entornos académicos, legales, organizaciones sin ánimo de lucro y otras organizaciones que requieren clasificación.
- Comercio electrónico y venta al por menor: ayuda a los usuarios a elegir productos mediante la reducción de las opciones por tema. Permite etiquetar productos o identificar artículos fraudulentos.
- Servicios: clasifica consultas de servicio, mensajes y respuestas para ayudar a resolver problemas y a desplegar soluciones de forma más rápida.
- Redes sociales: organizar tweets, correo electrónico, publicaciones y comparticiones en categorías o temas.

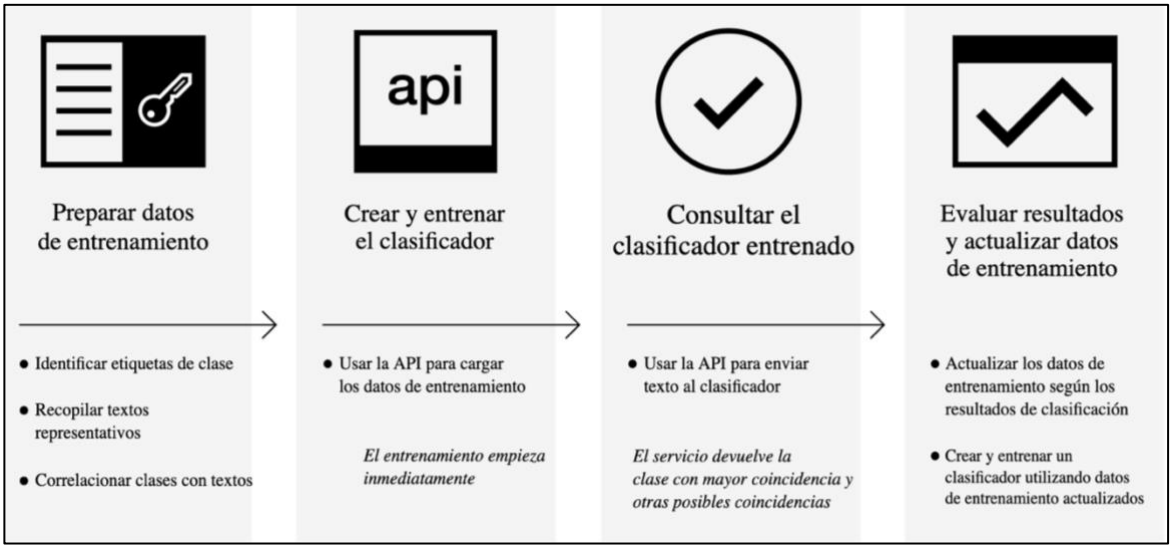


Figura 2. Uso del servicio NLC

*Fuente: IBM Cloud, «About» 2020. [En línea]. Available: https://cloud.ibm.com/docs/*natural-language-classifier?topic=natural-language-classifier-about. [Último acceso: 2 junio 2020].

2.4.2 Amazon Comprehend.

Amazon Comprehend es un servicio de PLN que emplea el aprendizaje automático para encontrar información y relaciones en textos.

Amazon Comprehend utiliza el aprendizaje automático para descubrir la información y las relaciones en datos no estructurados. El servicio identifica el idioma del texto; extrae frases, nombres de lugares, personas, marcas o eventos clave; comprende el grado de positividad o negatividad del texto; analiza el texto mediante tokenización y categorías gramaticales; y organiza automáticamente una colección de archivos de texto por tema. [23]

2.4.3 Google Cloud Natural Language.

Natural Language es un servicio de PLN que aplica el aprendizaje automático para mostrar la estructura y el significado de los textos. En el mismo sentido, ayuda a extraer información sobre personas, lugares o eventos, al igual que permite comprender mejor las opiniones en las redes sociales. Esta herramienta facilita analizar textos e integrarlos en el almacenamiento de documentos de Google denominado Cloud Storage [25].

2.4.4 Comparación entre los servicios de Procesamiento de Lenguaje Natural.

En la Tabla 2 se evidencia una comparación de las características principales entre los tres servicios más importantes de procesamiento natural disponibles. Cabe recalcar que, con base en las características presentadas, se escogió al servicio de IBM NLC por su fácil implementación y escalabilidad al momento de integrarlo en el sistema de recomendaciones.

	Google Natural Language	**Amazon Comprehend**	**Watson Natural Language Classifier**
API REST integrada: accede a los servicios de clasificación desde el API REST integrado.	SÍ	NO	SÍ
Análisis sintáctico: extrae tokens y frases; identifica categorías gramaticales y crea árboles de análisis de dependencias de cada frase.	SÍ	SÍ	SÍ

Tabla 2. Comparación de características provistas por los servicios de PLN

Fuente: elaboración propia

Clasificación de contenido personalizado: crea etiquetas personalizadas a partir de datos de entrenamiento para definir los modelos de clasificación.	**NO**	**SÍ**	**SÍ**
Varios idiomas: analiza texto fácilmente en varios idiomas, por ejemplo: alemán, chino (tanto simplificado como tradicional), coreano, español, francés, inglés, italiano, japonés, portugués y ruso.	SÍ	SÍ	SÍ
Compatibilidad con conjuntos de datos de gran tamaño: como admite 5000 etiquetas de clasificación y 1 millón de documentos de hasta 10 MB cada uno, se puede acceder a casos prácticos complejos.	SÍ	SÍ	NO

Tabla 3. Comparación de características provistas por los servicios de PLN – continuación

Fuente: elaboración propia

2.5 Herramientas y *frameworks* utilizados en el proyecto

2.5.1 Express.js.

Express.js es un *framework* minimalista y flexible que sirve para desarrollar aplicaciones web de Node.js; además, proporciona un conjunto sólido de características con el objetivo de crear una API de manera rápida y fácil [26]. Este *framework* fue empleado para construir el servidor RESTful que ejecuta llamadas hacia el servicio NLC dentro del sistema de recomendaciones.

2.5.2 Angular.

Angular (comúnmente llamado Angular 2+o Angular 2) es un *framework* para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza

para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. [27]
Este *framework* funcionó para desarrollar la aplicación web a través de la cual interactúa el usuario al momento de buscar recomendaciones de sitios turísticos dentro del sistema de recomendaciones [27].

2.5.3 MongoDB.

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos sea más fácil y rápida. [28]
En pro de la investigación, MongoDB se usó para almacenar la información correspondiente a los puntos de interés asociados a los sitios turísticos del centro histórico de Quito [28].

2.5.4 Webstorm.

Webstorm es un IDE (Integrated Development Environment) robusto de la compañía JetBrains; es empleado para el desarrollo de modernos ecosistemas en Javascript y se acopla a los *frameworks* y librerías más actuales para proyectos web, móviles y de escritorio [29]. Este IDE fue utilizado para desarrollar la aplicación web que usa el servicio NLC dentro del sistema de recomendaciones.

2.5.5 Google Maps.

Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. (filial de Google). Este ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo. Para este estudio, sirvió principalmente en la aplicación web y para presentar los puntos de interés de los sitios turísticos en un mapa interactivo [30]. De igual manera, se hizo necesario el servicio de geolocalización provisto por Google Maps para obtener la posición actual del usuario.

2.5.6 Google Analytics.

Google Analytics es una herramienta analítica web de la empresa Google lanzada el 14 de noviembre de 2005. Este servicio se encarga de ofrecer información agrupada del tráfico

que llega a los sitios web según audiencia, adquisición, comportamiento y conversiones que se llevan a cabo en el sitio web [31]. En este sentido, se trabajó con esta herramienta para analizar la cantidad de consultas enviadas por los usuarios en la aplicación web.

2.5.7 SCRUM.

En el desarrollo de la segunda fase de la metodología se utilizó el marco de trabajo Scrum, debido a que, a través de un enfoque ágil, se puede cumplir con los objetivos planteados dentro del tiempo establecido con la asimilación de cualquier cambio de requerimiento. Justo es decir que Scrum es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. De acuerdo con ello, no es un proceso o una técnica para construir productos; en lugar de eso, es un *framework* dentro del cual se pueden emplear varias técnicas y procesos. En suma, Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo [32].

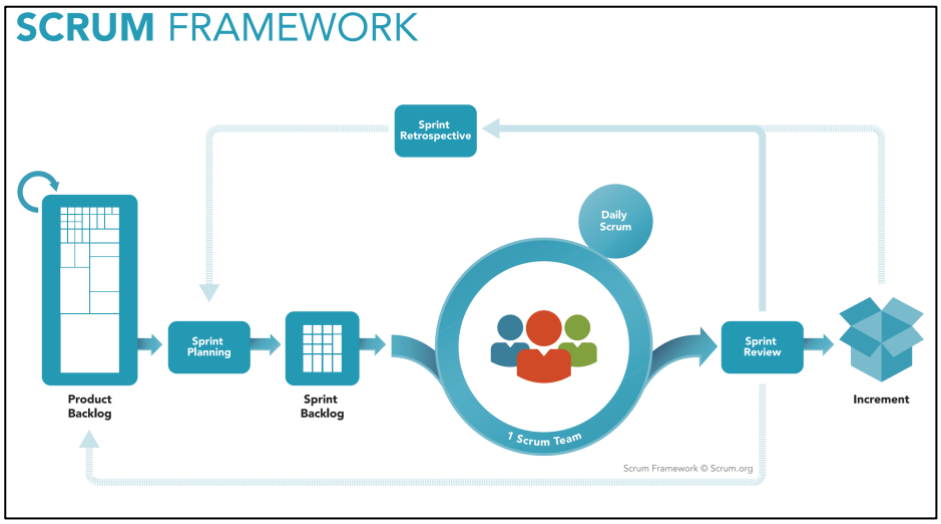


Figura 3. Descripción del marco de trabajo Scrum

Fuente: K. Schwaber y J. Sutherland, «The Scrum Guide: The Definitive The Rules of the Game», Scrum.Org and ScrumInc, p. 19, 2017.

3 METODOLOGÍA

Se utilizó un enfoque exploratorio para el desarrollo del presente proyecto. A partir de este se determinaron las características principales del problema para su búsqueda, análisis, uso de herramientas y lenguajes de programación que mejor se adaptaron a las necesidades halladas. En consonancia con ello, la metodología aplicada se dividió en dos fases clave que son: fase de recolección de datos y fase de desarrollo.

3.1 Fase de recolección de datos

Esta fase facilitó la obtención de datos necesarios para el entrenamiento y prueba del modelo de PLN que utiliza el servicio NLC. A partir de la creación de este prototipo se pudieron obtener las diferentes categorías que clasificaron las sentencias de texto remitidas al sistema de recomendaciones.

3.1.1 Conjunto de datos.

El conjunto de datos usados para el entrenamiento del modelo utilizado por el servicio NLC constó de un total de 685 registros y, a su vez, estos se obtuvieron a partir de encuestas realizadas a 60 usuarios. Basándose en lo anterior, la encuesta se estructuró en dos partes: la consulta que el turista realizaría a Google si visitara la ciudad y la tipificación de la pregunta (esto último para obtener la categoría que se asociaría a la respectiva consulta). Estos registros se almacenaron en un archivo que posteriormente fue trasladado al formato Comma Separated Values (CSV) del cual el 80 % se utilizó para el evento de entrenamiento y el restante 20 % se empleó con fines de prueba del modelo.

3.1.2 Preparación de datos de entrenamiento.

El aprendizaje automático describe el proceso de aprender algunas propiedades a partir de un conjunto de datos, para luego aplicar las propiedades a datos nuevos. Adviértase que, a pesar de todo, el servicio NLC sigue este proceso. Este servicio debe ser entrenado para conectar clases predefinidas a textos de ejemplo y después implementar dichas clases a entradas nuevas. Por lo tanto, el clasificador entrenado es tan bueno como los datos de entrenamiento. Cada valor de texto de los datos debe representar los tipos de textos sobre los que se desea que el clasificador realice su predicción. Asimismo, cada clase que se espera devolver debe estar en los datos de entrenamiento, y las clases que se asocien a cada texto deben ser las correctas [33].

3.1.3 Estructura de los datos de entrenamiento.

Se puede proporcionar los datos para entrenar el servicio NLC en formato CSV (...). En el formato CSV, una fila del archivo representa un registro del ejemplo. Cada registro tiene dos o más columnas. La primera columna es el texto representativo que hay que clasificar. Las columnas adicionales son clases que se aplican a dicho texto. [En la Figura 4] se muestra la representación de un archivo CSV que contiene cuatro registros. Cada registro del ejemplo incluye la entrada de texto y una clase, separados por una coma [34]:

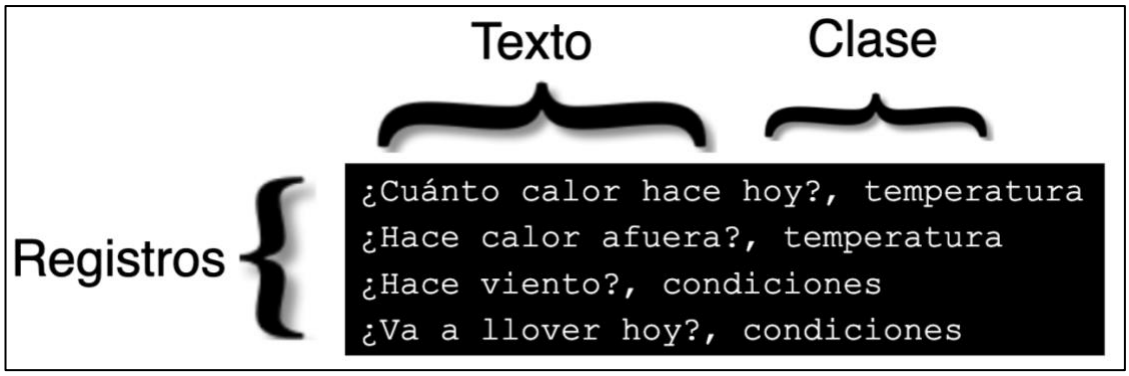


Figura 4. Representación de contenido de un archivo CSV válido

Fuente: IBM Cloud, «Preparación de los datos,» 2020. [En línea]. Available: https://cloud.ibm.com/docs/natural-language-classifier?topic=natural-language-classifier-using-your-data&locale=es. [Último acceso: 1 agosto 2020].

3.1.4 Identificación de etiquetas de clase.

El archivo corpus.csv se utilizó para identificar las diferentes clases que fueron obtenidas según lo explicado en el apartado 3.1.1. Siguiendo la preparación de datos, se pudieron identificar un total de sesenta y dos categorías o clases. La Tabla 3 muestra parte de la información que contiene el archivo corpus.csv, en el que se puede apreciar la consulta realizada por los usuarios y la categoría correspondiente a esa pregunta.

Pregunta	**Categoría**
Precios museos.	Museo
Restaurantes en el centro.	Comida
Cómo llegar a El Panecillo.	El Panecillo
Lugares por visitar.	Información
Hoteles Quito.	Hoteles
Agenda cultural.	Centro de información.
Bancos	Banco

Fiestas	Centro de información.
Cómo llegar a La Ronda.	Plazas
Sitios turísticos.	Información
Ropa	Centro comercial.
Lugares donde haya buena música.	Bar
Los museos más importantes.	Museo
Iglesias	Iglesia
Tomar buen desayuno.	Comida
¿Cuál es la principal iglesia en el centro de Quito?	Iglesia
Iglesias más representativas.	Iglesia
Lugares para visitar en el centro histórico.	Museo
¿Cuál es la iglesia más hermosa del centro histórico de Quito?	Iglesia
Lugares turísticos en el centro histórico.	Museo
Cómo llegar en transporte público.	Parada de bus
Calle de las iglesias.	Iglesia
Restaurantes	Comida
Iglesias icónicas.	Iglesia
Plazas e iglesias	Plazas
¿Dónde hay buena seguridad?	Seguridad
¿Dónde está la Plaza Grande?	Plazas
Fantasmas del centro de Quito.	Paranormal
Ubicación de museos.	Museo
Sugerencias de restaurantes de comida típica.	Comida
Ubicación de restaurantes.	Museos o iglesias.
Hoteles en el centro histórico de Quito.	Hoteles
El Panecillo.	Miradores
Cines en el centro histórico de Quito.	Cine
Bancos en el centro histórico de Quito.	Bancos
Hoteles en Quito.	Hoteles
Cajeros automáticos en el centro histórico de Quito.	Banco
Rutas de los buses para llegar al centro histórico.	Parada de bus.
Ruta de iglesias.	Iglesias
Museos	Museo
Comida típica.	Comida

Tabla 4. Contenido del archivo corpus.csv

Fuente: elaboración propia

3.2 Fase de desarrollo

En esta fase se manejó un enfoque ágil para lo cual se necesitó el marco de trabajo Scrum, el cual permite separar al producto en incrementos de software; estos se entregan al final

de cada Sprint. La duración de los Sprints se definió en el Sprint 0, de acuerdo con lo que establece el *framework*. Asimismo, se definió el contenido de las historias de usuario y el Product Backlog para toda la fase de desarrollo.

3.2.1 Sprint 0.

Este Sprint fue muy importante para el desarrollo de la parte práctica del proyecto, puesto que durante este periodo se contempló la recopilación de la información necesaria para su implementación. De igual manera, se desarrolló la arquitectura global para el proyecto, se definieron los roles del equipo Scrum, de las historias épicas, el Product Backlog y el número de Sprints con su respectiva duración.

3.2.2 Arquitectura.

La Figura 5 representa la arquitectura utilizada para la implementación del sistema. La lógica comienza con la obtención de la frase de consulta que el usuario desea buscar, según sus preferencias. Luego, el servicio NLC se encarga de procesar esta expresión a través del modelo de clasificación y retorna todas las categorías correspondientes al criterio de búsqueda. El servidor de Backend escoge la categoría principal, resultado del procesamiento anterior, y busca en la base de datos todos los puntos de interés que coincidan con esta categoría. Finalmente, se devuelven todos los puntos de interés que cumplan con esta condición en formato JSON hacia el servidor de aplicaciones para su visualización en la aplicación web.

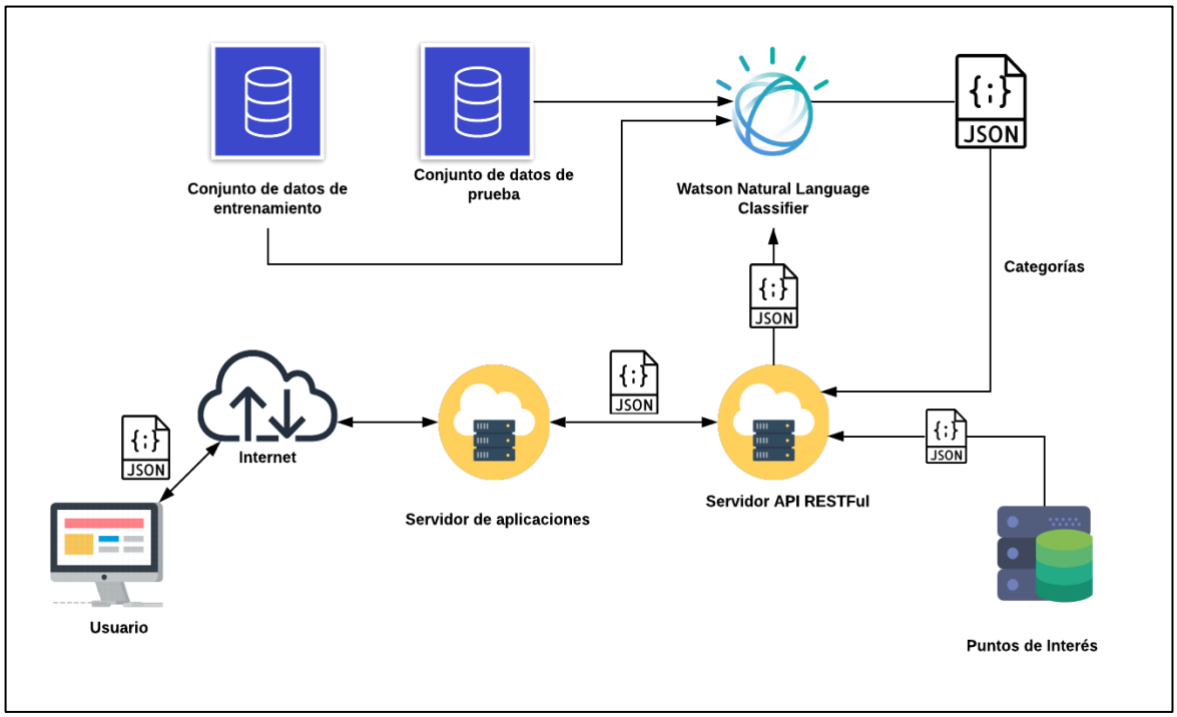


Figura 5. Arquitectura del sistema de recomendaciones

Fuente: elaboración propia

3.2.3 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 4.

Rol Scrum	Responsable
Product owner	Maritzol Tenemaza
Scrum master	Adrián Eguez, Maritzol Tenemaza
Development team	José Limaico

Tabla 5. Definición de roles del equipo Scrum

Fuente: elaboración propia

3.2.4 Definición de historias épicas.

Por otro lado, en la Tabla 5 se detallan las historias épicas definidas en el Sprint 0 para su posterior desarrollo.

Código	Descripción	Prioridad
ES01	Entrenamiento e implementación del clasificador de lenguaje natural.	Alta
ES02	Desarrollo del API RESTful para obtener los puntos de interés.	Alta
ES03	Desarrollo de la aplicación web para probar el sistema de recomendaciones por parte del usuario final.	Media

Tabla 6. Definición de historias épicas

Fuente: elaboración propia

3.2.5 Product Backlog.

La Tabla 6 detalla el Product Backlog obtenido a partir del desarrollo de las historias épicas especificadas en el Sprint 0.

Product Backlog				
Historia épica.	**Historia de usuario**			
	ID	**Título**	**Estimación (días).**	**Prioridad**
ES01	ES01-01	Construcción del modelo de clasificación para el servicio NLC.	7	Alta
	ES01-02	Implementación del clasificador de lenguaje natural.	1	Alta
	ES01-03	Consumir servicio de IBM NLC.	1	Media
ES02	ES02-01	Creación de API RESTful para consumir peticiones HTTP.	6	Alta
	ES02-02	Conexión con la base de datos no relacional MongoDB para almacenar información de sitios turísticos.	3	Alta
ES03	ES03-01	Desarrollo de la interfaz de inicio.	3	Media
	ES03-02	Desarrollo de la interfaz de búsqueda.	4	Media
	ES03-03	Desarrollo de la interfaz para mostrar resultados.	2	Media

Tabla 7. Product Backlog definido para el proyecto

Fuente: elaboración propia

3.3 Sprint 1

3.3.1 Sprint Planning.

- **Objetivo del Sprint**

Construir un clasificador de lenguaje natural mediante el entrenamiento de un modelo de clasificación, el cual fue empleado por el servicio NLC.

- **Historias de usuario**

En la Tabla 7 se exhibe la lista de las historias de usuario que se tomaron para este primer Sprint con su respectiva estimación; en el Anexo II se detallan sus características.

ID	Titulo	Estimación (días).	Estado
ES01-01	Construcción del modelo de clasificación para el servicio NLC.	7	Por Implementar.
ES01-02	Implementación del clasificador de lenguaje natural.	1	Por implementar.
ES01-03	Consumir servicio de IBM NLC.	1	Por implementar.

Tabla 8. Historias de usuario seleccionadas para el Sprint 1

Fuente: elaboración propia

- **Sprint Backlog**

De acuerdo con la explicación del procedimiento, en la Tabla 8 se definen las tareas asignadas a cada historia de usuario para el Sprint 1.

Sprint Backlog	
Historia de usuario.	**Tareas**
ES01-01	Construir el modelo de clasificación de lenguaje natural, utilizando el 80 % del conjunto de datos obtenidos anteriormente.
	Probar el modelo de clasificación, pero solo con el uso del 20 % del conjunto de datos recopilados con anterioridad.
ES01-02	Implementar el servicio NLC empleando el modelo de clasificación –entrenado previamente– a través de la API ofrecida por el servicio.
	Usar la API para enviar textos al clasificador y así obtener la categoría con más –y otras– coincidencias.
ES01-03	Consumir servicio de IBM NLC.
	Evaluar los resultados obtenidos y actualizar los datos de entrenamiento según sea el caso.

Tabla 9. Sprint Backlog para el Sprint 1

Fuente: elaboración propia

3.3.2 Ejecución del Sprint.

- **Carga de los datos de entrenamiento**

El servicio NLC se entrenó con solo el 80 % del conjunto de datos obtenido; este puede ser cargado como un Asset dentro del proyecto, utilizando el API provisto por el entorno de desarrollo Watson Studio. En la Figura 6 se indica el entorno donde se realizó la carga del archivo, el cual contiene el grupo de datos de entrenamiento para iniciar la creación del modelo de clasificación a partir del archivo corpus_limpio.csv.

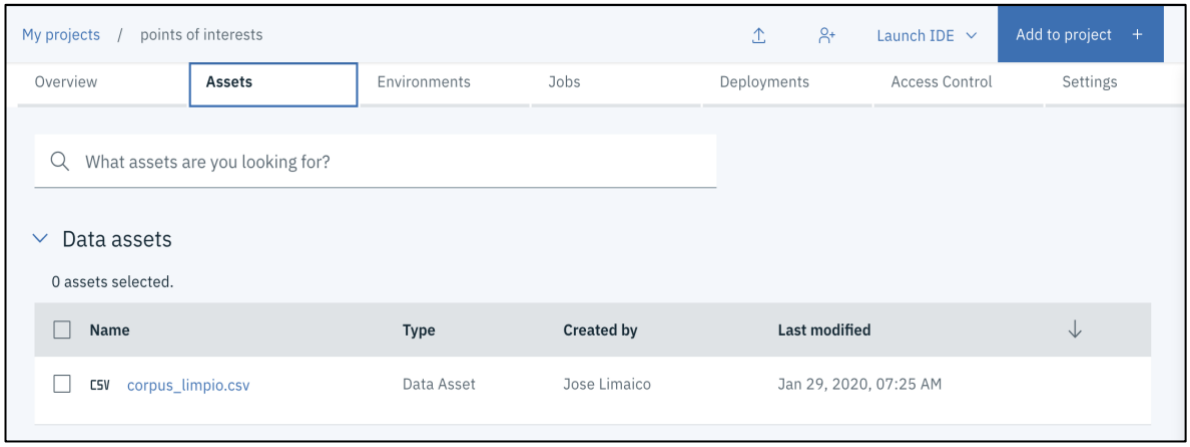


Figura 6. Listado de Assets que contiene el proyecto asociado al servicio NLC

Fuente: elaboración propia

- **Evento de entrenamiento**

Se puede iniciar un evento de entrenamiento posterior a la carga de archivos al proyecto creado previamente en Watson Studio y, en este mismo sentido, vincular el servicio deseado (en este caso NLC) al proyecto. Una vez que se ha subido el archivo que contiene el conjunto de datos de entrenamiento, se procede a crear un evento de entrenamiento, el cual empieza inmediatamente. La Figura 7 indica un resumen del resultado del evento de entrenamiento, en el que se aprecian algunas características: estado del modelo, fecha de creación, lenguaje empleado, número de clases y número de ejemplos de texto obtenidos.

Figura 7. Resumen del evento de entrenamiento y creación del modelo de clasificación

Fuente: elaboración propia

- **Consumo del servicio NLC**

En la Figura 8 se observa el cuerpo de la petición utilizado para consumir el API del servicio, este se crea automáticamente luego de que se ha completado dicho evento satisfactoriamente. Ahora bien, para la prueba se realizó la siguiente búsqueda: “iglesias del Centro Histórico de Quito”. El endpoint de la API utilizado para probar el servicio fue el siguiente:
https://gateway.watsonplatform.net/natural-language-classifier/api/v1/classifiers/dc534bx708-nlc-22/classify

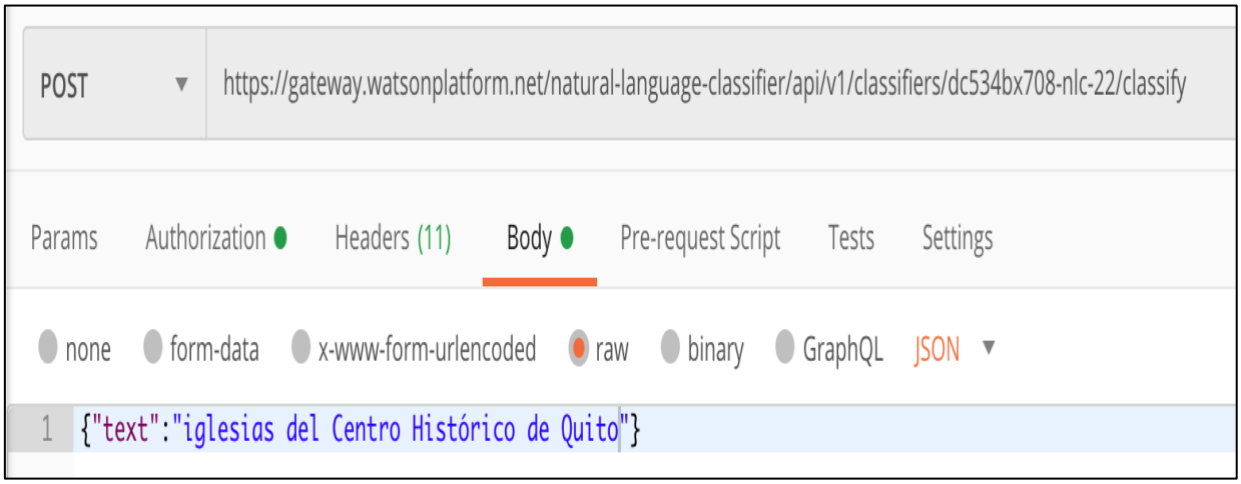


Figura 8. Consumo del servicio NLC empleando Postman

Fuente: elaboración propia

La respuesta del servidor envía un atributo “top_class” que contiene el nombre de la categoría con mayor nivel de confianza, es decir, aquella que más coincide con el criterio de búsqueda. Paralelamente, surge otro componente denominado “classes” que es un arreglo de objetos, el cual contiene todas las categorías que podrían coincidir con el criterio de búsqueda, como se puede apreciar en la Figura 9.

Figura 9. Respuesta del servicio NLC

Fuente: elaboración propia

3.3.3 Sprint Review.

El objetivo del primer Sprint se alcanzó con facilidad, puesto que no apareció ningún obstáculo durante el desarrollo, gracias a la fácil implementación del modelo de clasificación de lenguaje natural ofrecido por IBM y su servicio NLC provisto por Watson Studio.

- **Pruebas de aceptación**

Luego de realizar el trabajo del primer Sprint, en la Tabla 9 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	Cumplido
ES01-01	Se crea un nuevo Asset con el conjunto de datos de entrenamiento.	Se entrena el modelo de clasificación de lenguaje natural.	Se muestra una tabla de resumen con la información del modelo entrenado satisfactoriamente.	SÍ
ES01-02	Se crea un nuevo Asset con el conjunto de datos de prueba.	Se utiliza este Asset para probar el funcionamiento del modelo.	El modelo de clasificación devuelve las categorías correctas para cada registro del conjunto de datos de prueba.	SÍ
ES01-03	Se implementa el modelo de clasificación de lenguaje natural.	El usuario consume el modelo de clasificación –por medio del servicio NLC–, enviando un texto de consulta.	Se obtienen todas las categorías relacionadas al texto de consulta con su respectivo nivel de coincidencia.	Sí

Tabla 10. Criterios de aceptación definidos para el Sprint 1

Fuente: elaboración propia

3.3.4 Sprint Retrospective.

En la Figura 10 (gráfico de *burn down*) se observa el progreso de las tareas trabajadas durante todo el Sprint. Como se puede apreciar, el desarrollo del primer Sprint se pudo completar en el tiempo estimado.

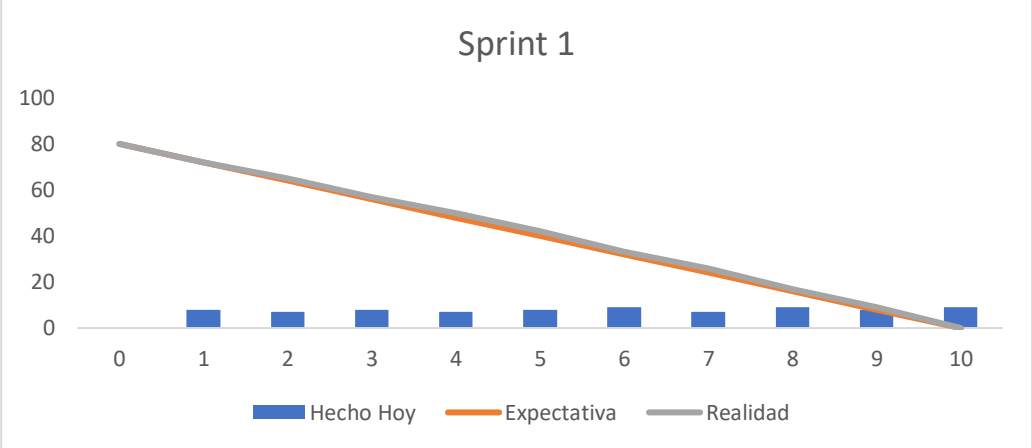


Figura 10. Gráfico *burn down* para el Sprint 1

Fuente: elaboración propia

3.4 Sprint 2

3.4.1 Sprint Planning.

- **Objetivo del Sprint**

Implementar un servidor RESTful HTTP que consuma el servicio de procesamiento de lenguaje natural NLC para obtener los puntos de interés requeridos por el usuario.

- **Historias de usuario**

En la Tabla 10 se indica la lista de las historias de usuario que se tomaron para el segundo Sprint con su respectiva estimación; en el Anexo III se explican de forma detallada.

ID	Titulo	Estimación (días)	Estado
ES02-01	Creación de API RESTful para consumir peticiones HTTP.	6	Por Implementar.
ES02-02	Conexión con la base de datos MongoDB.	3	Por Implementar.

Tabla 11. Historias de usuario seleccionadas para el Sprint 2

Fuente: elaboración propia

- **Sprint Backlog**

En la Tabla 11 se detallan las tareas definidas para cada historia que se implementa en el Sprint 2.

Sprint Backlog	
Historia de usuario.	**Tareas**
ES02-01	Crear la configuración del proyecto base en Nodejs con el gestor de NPM (Node Package Manager). Enseguida, instalar Express.js y ODM para conexión con la base de datos MongoDB.
	Levantar el servidor y verificar que escuche peticiones HTTP.
	Crear una función que atienda solicitudes HTTP POST a través del endpoint/categories y que retorne una respuesta de tipo JSON. Esta respuesta deberá restituir todos los puntos de interés que coincidan con la búsqueda realizada por el usuario.

Tabla 12. Sprint Backlog para el Sprint 2

Fuente: elaboración propia

	Diseñar el cuerpo de petición que se recibirá en la solicitud del cliente para aceptar sus preferencias.
	Validar que el cuerpo de petición coincida con el esquema diseñado.
ES02-02	Crear el esquema de la base de datos e implementar su conexión.
	Crear una función que escuche solicitudes HTTP POST a través del endpoint /create-category. Esta función debe retornar una respuesta de tipo JSON.
	Insertar registros en la base de datos utilizando el endpoint/create-category.

Tabla 13. Sprint Backlog para el Sprint 2 - continuación

Fuente: elaboración propia

3.4.2 Ejecución del Sprint.

- **Creación de API RESTful para consumir peticiones HTTP**

Se creó un servidor en el entorno Node.js utilizando el *framework* Express.js, el cual permitirá escuchar peticiones HTTP enviadas desde la aplicación web del sistema de recomendaciones. En la Figura 11 se enuncia la estructura del Backend de dicho sistema. Cierto es que también se observan las carpetas como constants, models, services y el controlador principal main. controller. Esta arquitectura permite recibir peticiones HTTP desde la aplicación web, las cuales son enviadas hacia el servicio NLC. Desde esta perspectiva, este servicio se encarga de procesar las peticiones recibidas y, a su vez, procede a enviar la categoría con más coincidencias junto con un listado de posibles expresiones que tengan relación con la frase recibida.

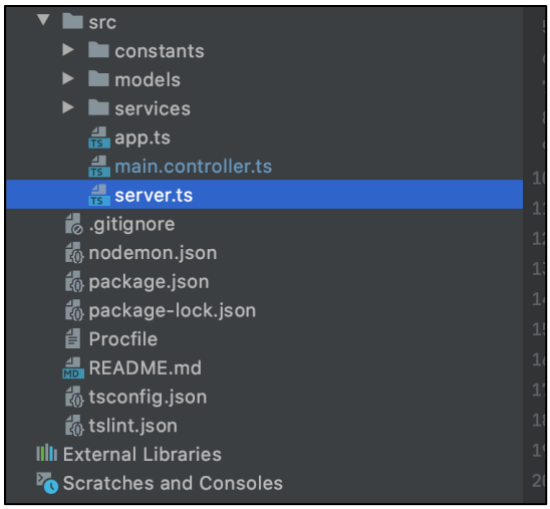


Figura 11. Estructura del Backend utilizado por el sistema de recomendaciones

Fuente: elaboración propia

Para la configuración del ODM se utilizó la librería Mongoose [27], a la cual se le instaló el proyecto empleando el administrador de paquetes de Node.js NPM. Este atributo permite gestionar la conexión con la base de datos MongoDB por medio de una solución directa, basada en esquemas para modelar los datos de la aplicación.

Para el consumo del servicio de IBM NLC se creó la siguiente ruta denominada “categories” en el controlador principal. Al mismo tiempo, esta ruta conlleva un servicio denominado LocationsService que ejecuta un método conocido como getCategories (). Este método se encarga de procesar la consulta que envía la aplicación web y la transforma en una petición HTTP para enviarla hacia el API creado en la implementación del servicio NLC.

```
public getCategories(req: Request, res: Response) {
    const text_query: string = req.body.text;
    const naturalLanguageClassifier = new NaturalLanguageClassifierV1( options: {
        iam_apikey: `${API_KEY}`
    });
    naturalLanguageClassifier.classify(
        params: {
            text: `${text_query}`,
            classifier_id: `${CLASSIFIER_ID}`
        },
        callback: (error: Error, response: any) => {
            if (error)
                res.send(error);
            else {
                Location.find( conditions: {ibm_type: _.defaultTo(response.top_class,  defaultValue: '')},
                    projection: (error: Error, location: MongooseDocument) => {
                    if (error)
                        res.send(error);

                    res.json(location)

                });
            }

        }
    )

}
```

Figura 12. Método getCategories que realiza la petición HTTP hacia el servicio de IBM NLC

Fuente: elaboración propia

La Figura 13 detalla el archivo main. controller; en este se ejecuta la llamada hacia la ruta especificada “/categories” la cual recibe como parámetro el texto de consulta ingresado por el usuario como parte del cuerpo de la petición.

```
public routes() {
    this.app.route( prefix: '/').get(this.locationsService.welcomeMessage);
    this.app.route( prefix: '/get-locations').get(this.locationsService.getAllLocations);
    this.app.route( prefix: "/categories").post(this.locationsService.getCategories);
    this.app.route( prefix: "/create-location").post(this.locationsService.addNewLocation);
}
```

Figura 13. Método routes () que instancia las rutas del servidor Backend

Fuente: elaboración propia

- **Conexión con la base de datos MongoDB**

Para la conexión con la base de datos se implementó el método connect proporcionado por la librería Mongoose, el cual permite conectar el servidor Backend con la base de datos no relacional MongoDB. La Figura 14 manifiesta el método empleado para realizar la conexión con la base de datos a través de la URI (Uniform Resource Identifier) proporcionada por esta.

```
private setMongoConfig() {
    const dbu_uri:string = process.env.MONGODB_URI || 'mongodb://localhost:27017/locations';
    mongoose.Promise = global.Promise;
    mongoose.connect( uris: `${dbu_uri}`, options: {
        useNewUrlParser: true,
        useUnifiedTopology: true
    });
}
```

Figura 14. Conexión con la base de datos

Fuente: elaboración propia

Para la creación del esquema de la base de datos se aplicaron los modelos de Mongoose. Estos son constructores compilados a partir de definiciones de Schema. Hay que recalcar que una instancia de un modelo se llama documento. Estos son responsables de crear y leer documentos de la base de datos subyacente de MongoDB [35]. La definición del esquema se representa en la Tabla 12.

Nombre	Tipo de dato	Descripción
Geometry	Object	(latitude, longitude)
icon	String	URL del ícono del punto de interés.
name	String	Nombre del punto de interés.

Tabla 14. Descripción del esquema de la tabla Location

Fuente: elaboración propia

Photos	Object	Información de la imagen del punto de interés.
rating	Number	Valoración del punto de interés.
Vicinity	String	Dirección del punto de interés.
ibm_type	Array [strings]	Arreglo de categorías que representan al punto de interés.

Tabla 15. Descripción del esquema de la tabla Location – continuación

Fuente: elaboración propia

Para insertar datos en la base se creó un endpoint de tipo POST asociado a la ruta /create-location. Es prudente advertir que esta ruta se encarga de llamar al método addNewLocation () y, asimismo, inserta los registros en la base de datos como se demuestra en la Figura 15.

```
public addNewLocation(req: Request, res: Response) {
    const newLocation = new Location(req.body);
    newLocation.save( fn: (error: Error, location: MongooseDocument) => {
        if (error) {
            res.send(error);
        }
        res.json(location);
    });
}
```

Figura 15. Método addNewLocation () utilizado para insertar registros en la base de datos

Fuente: elaboración propia

3.4.3 Sprint Review.

El objetivo del segundo Sprint se ha cumplido con éxito en el tiempo acordado, debido a la experiencia del equipo de desarrollo con respecto al uso de las herramientas, lenguajes y *frameworks*. No existió ningún retraso con la entrega del Sprint y se pudo obtener un incremento significativo en cuanto a la implementación del sistema de recomendaciones.

- **Pruebas de aceptación**

Luego de realizar el trabajo del segundo Sprint, en la Tabla 12 se puede verificar que se cumplieron todos los criterios de aceptación de las historias de usuario.

Historia de Usuario	Dado que	Cuando	Entonces	Cumplido
ES02-01	Se consume el endpoint/categories.	Se envía como cuerpo de la petición la consulta realizada por el usuario.	Se obtiene como resultado una lista de todos los sitios turísticos que coincidan con las preferencias del usuario en formato JSON.	Sí
	Se consume el endpoint/categories.	No se envía como cuerpo de la petición la consulta realizada por el usuario.	Se muestra un mensaje de error indicando que la solicitud es inválida	Sí
ES02-02	Se consume el endpoint/create-category.	Y se envía la información del registro a insertar en la base de datos.	Se obtiene como resultado un JSON con la información del sitio turístico que ha sido creado en la base de datos.	Sí
	Se consume el endpoint /create-category correctamente.	Se revisa la base de datos.	Se verifica que el registro se ha creado con éxito.	Sí

Tabla 16. Criterios de aceptación del Sprint 2

Fuente: elaboración propia

- **Incremento obtenido**

El incremento obtenido refleja un avance importante con respecto al desarrollo del sistema de recomendaciones. Con el fin de cumplir el objetivo, se generaron puntos de interés que cumplen con las preferencias del turista y estos se reflejan en formato JSON. En la Figura 16 se presentan los resultados obtenidos a través de Postman al consumir el endpoint/categories.

Figura 16. Consumo del endpoint categories

Fuente: elaboración propia

En la Figura 17 se evidencia el consumo del endpoint /create-category y, al mismo tiempo, se detalla la creación de un nuevo registro en la base de datos al momento que el servidor responde con la respuesta en formato JSON.

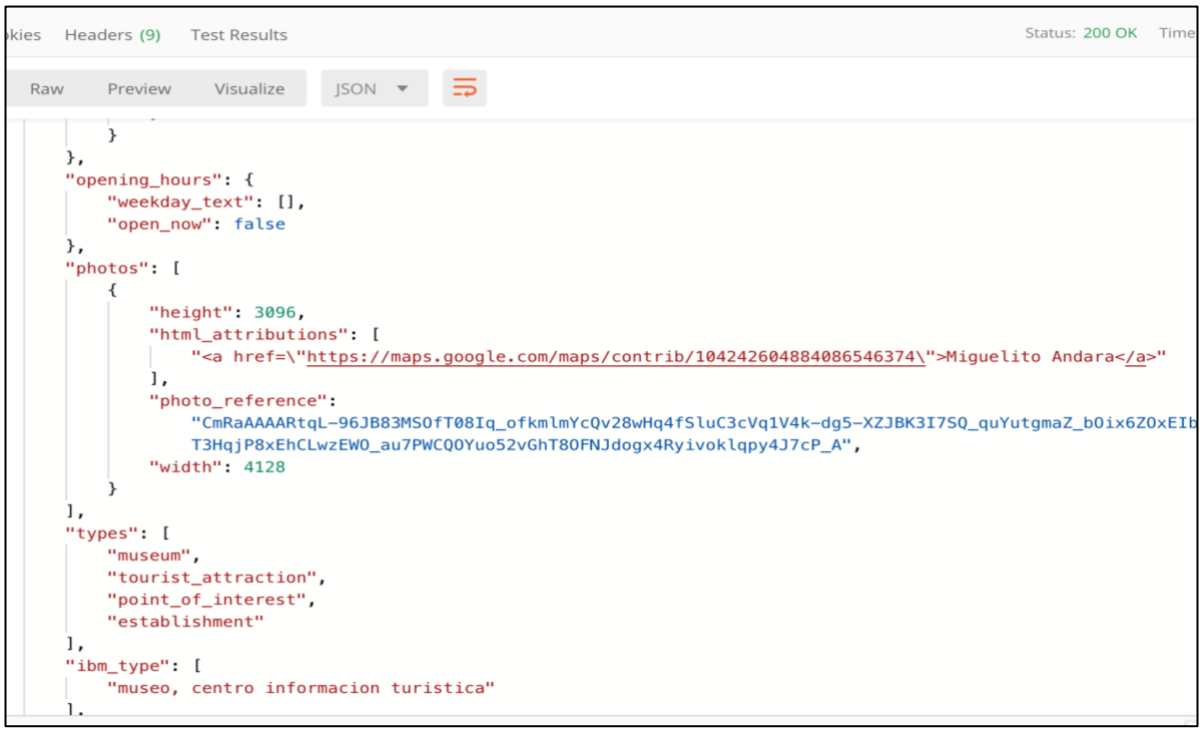


Figura 17. Creación de registro en la base de datos utilizando Postman

Fuente: elaboración propia

3.4.4 Sprint Retrospective.

En la Figura 18 se desarrolla el gráfico de *Burn down* para el Sprint 2. Debido al conocimiento de las herramientas y *frameworks* utilizados, se expone en el gráfico la idea de que se pudo completar el desarrollo sin mayor contratiempo.

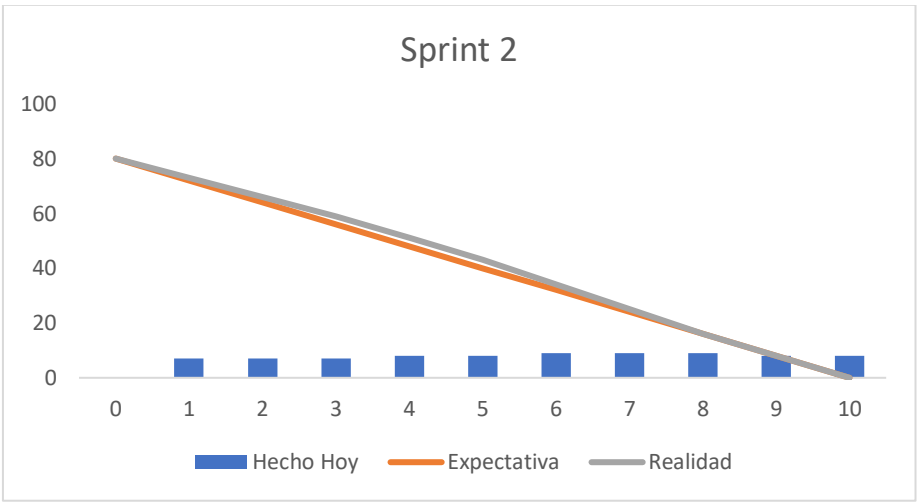


Figura 18. Gráfico *burn down* del Sprint 2

Fuente: elaboración propia

3.5 Sprint 3

3.5.1 Sprint Planning.

- **Objetivo del Sprint**

Desarrollar una aplicación web que permita probar el sistema de recomendaciones al usuario final.

- **Historias de usuario**

En la Tabla 14 se señala la lista de las historias de usuario que se tomaron para el tercer Sprint con su respectiva estimación; estas historias se profundizan en detalle en el Anexo IV.

ID	Titulo	Estimación (días).	Estado
ES03-01	Desarrollo de la interfaz de inicio.	3	Por implementar.
ES03-02	Desarrollo de la interfaz de búsqueda.	4	Por implementar.
ES03-03	Desarrollo de la interfaz para desplegar resultados.	2	Por implementar.

Tabla 17. Historias de usuario seleccionadas para el Sprint 3

Fuente: elaboración propia

- **Sprint Backlog**

Desde otra perspectiva, en la Tabla 15 se expone el Sprint Backlog del Sprint 3 en el que se detallan las tareas asignadas a cada historia de usuario.

Sprint Backlog	
Historia de usuario.	**Tareas**
ES03-01	Inicializar el proyecto en angular.
	Crear el componente de la pantalla de Inicio que contiene la información de uso de la aplicación.
	Implementar la vista de inicio de acuerdo con el diseño.
	Crear las interfaces para la respuesta que envía el consumo del endpoint/categories.

Tabla 18. Sprint Backlog para el Sprint 3

Fuente: elaboración propia

ES03-02	Habilitar la API Directions de Google Maps para obtener la ruta más optima entre la posición actual del usuario y el punto de interés en específico.
	Instalar la librería angular-google-maps, atributo que permite desplegar el mapa de Google Maps y seleccionar un punto en el mapa.
	Crear el componente para la vista de despliegue, el cual deja ver los resultados de la búsqueda en el mapa.
	Implementar la vista de consulta de los sitios turísticos de acuerdo con el diseño.
	Utilizar enrutamiento para cargar los nuevos componentes.
	Validar que el campo de búsqueda del formulario sea requerido.
	Consumir el endpoint/categories con el texto de consulta obtenido en el formulario.
ES03-03	Presentar los resultados obtenidos en el mapa, exhibidos como marcadores de los puntos de interés.
	Agregar ventanas de información a los marcadores del mapa para mostrar información detallada del sitio turístico.
	Mostrar los detalles de las rutas para cada sitio turístico en una tabla.

Tabla 19. Sprint Backlog para el Sprint 3 – continuación

Fuente: elaboración propia

3.5.2 Desarrollo del Sprint.

- **Desarrollo de la interfaz de inicio**

Para el desarrollo de la interfaz de inicio se sirvió del diseño del Mockup 1 (ver Anexo V) como base. En el desarrollo de esta vista se usó la librería Angular Material, la cual contiene componentes que se pueden utilizar junto con la Angular para su desarrollo. Con el fin de detallar lo mencionado, en la Figura 19 se señala la vista diseñada en Angular con la implementación de la imagen principal y la información de uso de la aplicación.

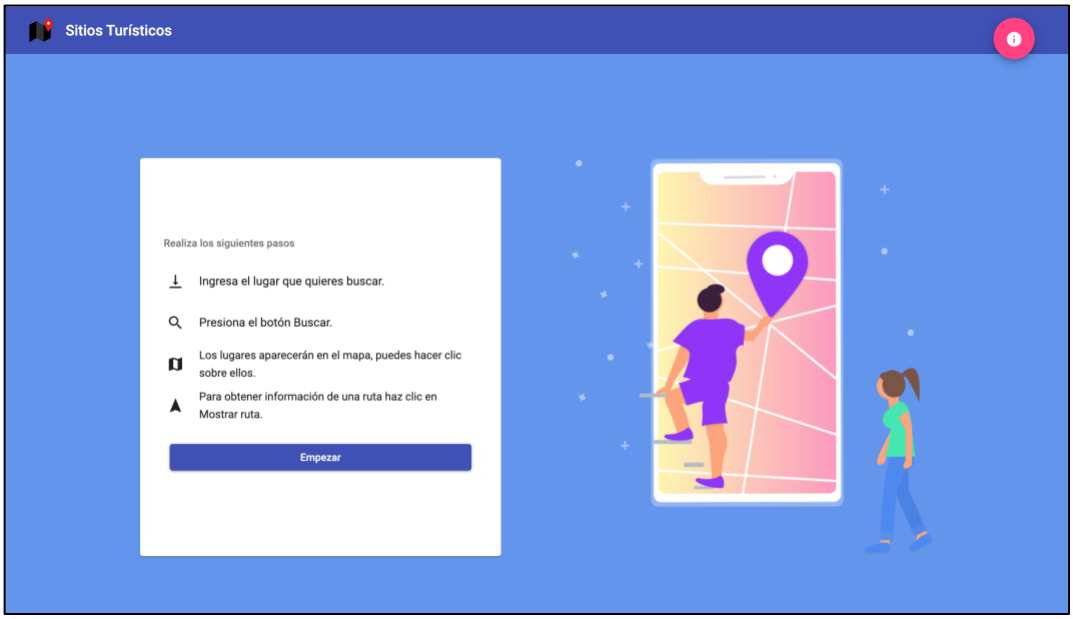


Figura 19. Interfaz de inicio de la aplicación

Fuente: elaboración propia

- **Desarrollo de la interfaz de búsqueda**

El desarrollo de esta interfaz implicó la integración de la librería angular google maps (agm) que gestiona de manera optimizada el uso del SDK (Software Development Kit) de Mapas para Javascript provisto por Google, permitiendo una implementación más sencilla de mapas en Angular. En la Figura 20 se manifiesta la interfaz de búsqueda a través de la cual los usuarios pueden indagar los sitios turísticos de su preferencia. Desde otro ángulo, un campo de texto permite recolectar el contenido de la consulta que a su vez será enviada hacia el servidor, para obtener los puntos de interés que más se acerquen a lo solicitado por el turista.

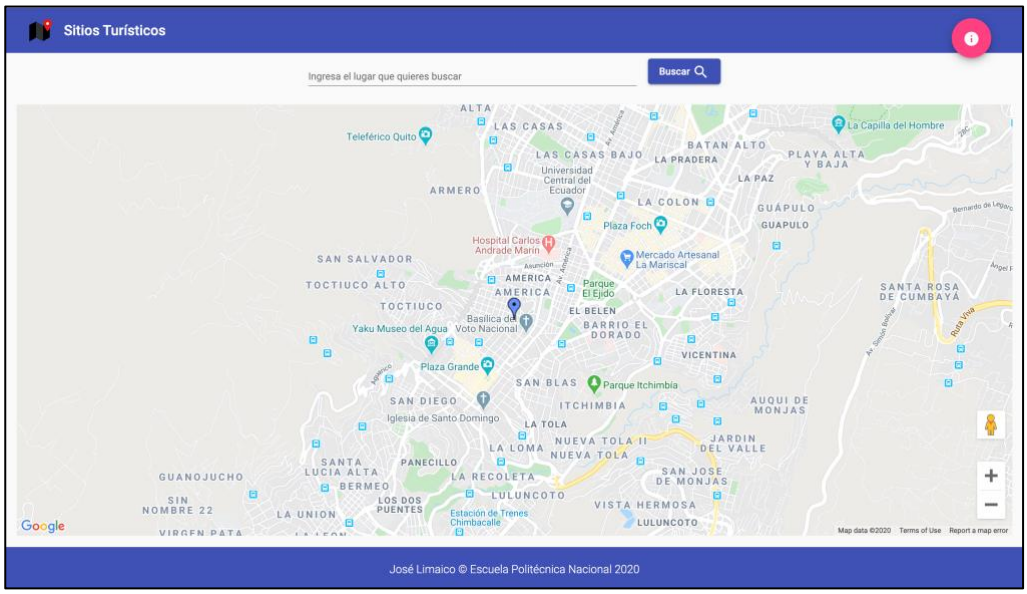


Figura 20. Interfaz de búsqueda de sitios turísticos

Fuente: elaboración propia

3.6 Interfaz para presentación de resultados

De acuerdo con esto, la Figura 21 enseña la interfaz de despliegue de resultado donde los puntos de interés obtenidos se representan con un marcador de color rojo, mientras que la ubicación actual del usuario se especifica con un resaltador de color azul. En esta imagen también se observan las ventanas informativas que contiene cada elemento.

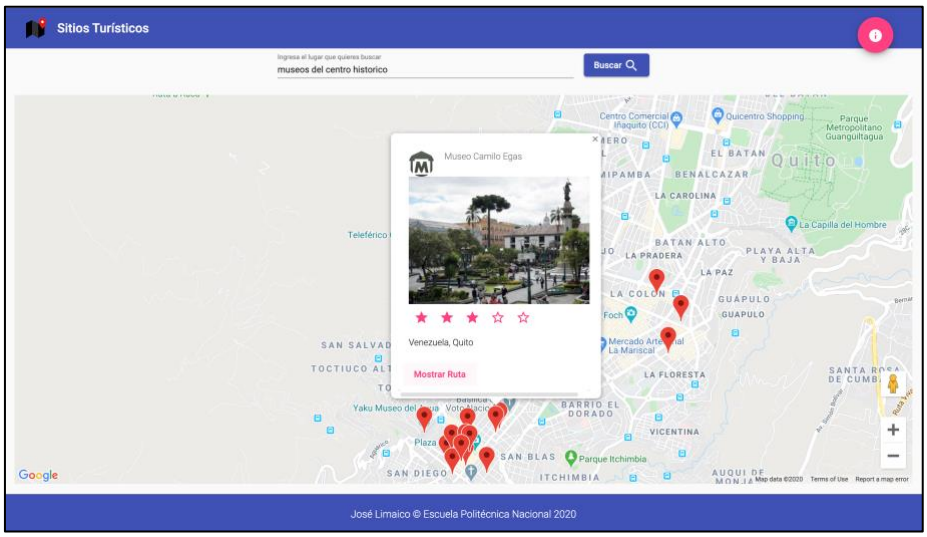


Figura 21. Interfaz de presentación de resultados

Fuente: elaboración propia

De la misma manera, la Figura 22 indica la ruta obtenida entre la posición actual del usuario y un sitio turístico en específico (Museo del Agua). De otro lado, en el lado izquierdo de la aplicación se precisan las direcciones que el usuario debe seguir para llegar a su destino, partiendo desde su posición actual.

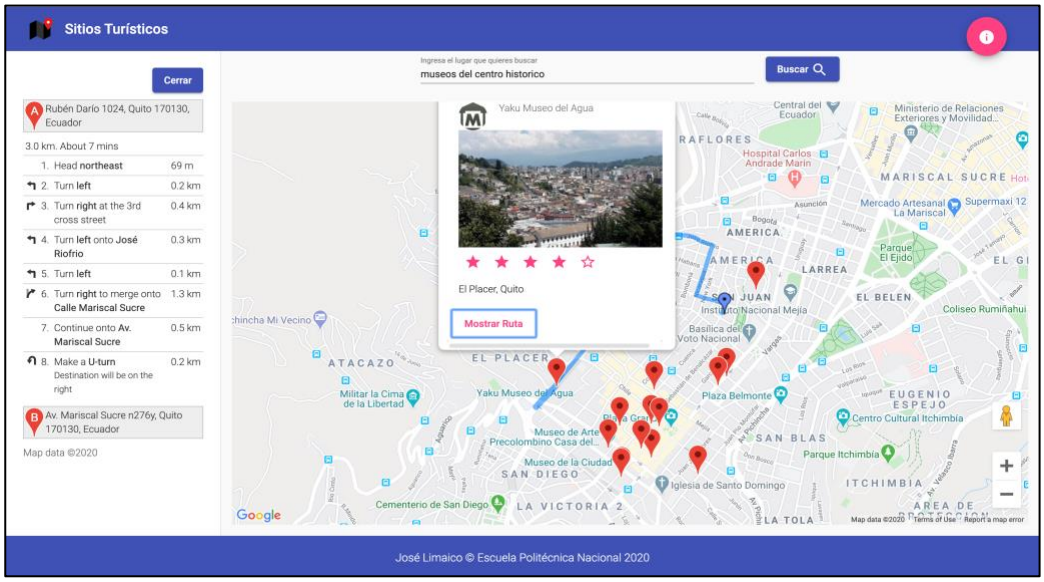


Figura 22. Ruta optimizada entre la posición actual del usuario y el sitio turístico

Fuente: elaboración propia

3.6.1 Sprint Review.

El objetivo de este Sprint se pudo completar en el tiempo acordado, pero con cierta demora del equipo de desarrollo debido a la falta de conocimiento de las nuevas librerías que se tuvieron que utilizar para desarrollar el aplicativo. Sin embargo, se pudo completar el objetivo del Sprint de acuerdo con lo planificado previamente y, de la misma manera, se logró finalizar el desarrollo del sistema de recomendaciones en el tiempo estimado.

- **Pruebas de aceptación**

Posterior a ejecutado el trabajo del tercer sprint, en la Tabla 16 se verifica el cumplimiento de todos los criterios de aceptación de las historias de usuario.

Historia de usuario.	Dado que	Cuando	Entonces	Cumplido
ES3-01	El usuario desee consultar sitios turísticos del centro histórico.	El usuario acceda a la pantalla de inicio.	Presentar una vista informativa que indique la manera de usar la aplicación.	SÍ
	El usuario accede a la pantalla de inicio.	El usuario necesita navegar hacia la vista de consulta.	Tener un botón que permita navegar hacia la vista de consulta.	SÍ
ES03-02	El usuario accede a la pantalla de búsqueda de resultados.	El usuario quiera consultar información de sitios turísticos de su preferencia.	Se debe implementar un input de entrada de texto que permita al usuario ingresar la consulta de los sitios turísticos.	SÍ
ES3-03	El usuario accede a la pantalla de búsqueda de resultados.	El usuario realiza la consulta.	Se deben mostrar en un mapa interactivo todos los sitios turísticos, luego de ser consultados y, a la vez, representados por un marcador de color rojo.	SÍ
	El usuario realiza la consulta.	Se muestran los marcadores que representan la ubicación de los sitios turísticos en el mapa.	Aplicar ventanas informativas que se despliegan al hacer clic sobre los marcadores.	SÍ
	Se muestran las ventanas informativas sobre los marcadores.	El usuario haga clic en el botón “Mostrar Ruta”.	Se debe especificar la ruta más corta en el mapa entre la ubicación actual del usuario y el sitio turístico seleccionado.	SÍ

Tabla 20. Pruebas de aceptación para el Sprint 3

Fuente: elaboración propia

3.6.2 Sprint Retrospective.

A continuación, en la Figura 24 se ejemplifica el Sprint 3 mediante el gráfico de *burn down*. En la línea del progreso se observa un detenimiento por las dificultades al momento de integrar el sistema de análisis de lenguaje natural con el sistema de recomendaciones.

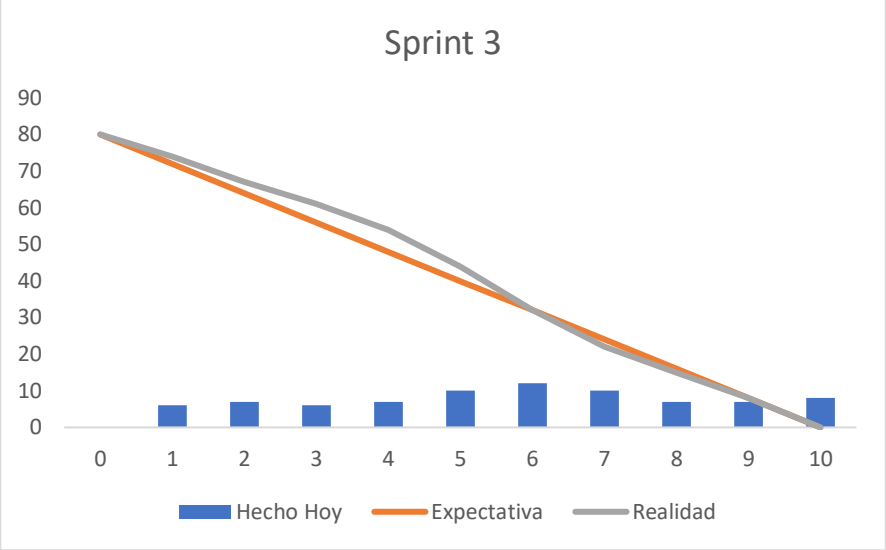


Figura 23. Gráfico *burn down* del Sprint 3

Fuente: elaboración propia

4 RESULTADOS Y DISCUSIÓN

De los resultados obtenidos se estableció que el modelo de clasificación de lenguaje natural implementado generó recomendaciones de sitios turísticos. De esta forma, se cumplió con el desarrollo de la pregunta de investigación planteada, cuestionamiento base del desarrollo de este proyecto.

Por otro lado, la implementación del sistema de recomendaciones evidenció que la construcción y entrenamiento de un modelo de clasificación, que utiliza enfoques de aprendizaje profundo y aprendizaje automático para su entrenamiento, permite la generación de recomendaciones de sitios turísticos. Estas recomendaciones se obtienen al clasificar las sentencias de texto en categorías definidas por los resultados del evento de entrenamiento. A continuación, se discute el producto final.

4.1 Clasificador de lenguaje natural

Para el PLN se utilizó el servicio de IBM Watson NLC, este fue entrenado con un conjunto de datos obtenidos de un archivo CSV. Los datos se obtuvieron a partir de encuestas realizadas con un total de 685 registros. De lo anterior se destaca que el 80 % de los registros de este archivo se utilizaron como un activo de datos para el entrenamiento del servicio NLC. Luego de ejecutar el evento de entrenamiento, se obtuvo un total de 62 categorías o clases que abarcaron un total de 650 ejemplos de texto. Para probar el modelo entrenado previamente, perteneciente al servicio NLC, se utilizó el 20 % restante de los registros del archivo CSV.

La Tabla 17 precisa las clases principales con más de diez ejemplos y, asimismo, demuestra la cantidad de pautas obtenidas al finalizar el evento de entrenamiento para su posterior despliegue y prueba.

Clase	Número de ejemplos
Bar	45
Restaurantes	25
Centro de información turística	51
Comida	88
Edificio civil	22
Galería de arte	10
Hoteles	47
Iglesias	68
Museos	70
Platos típicos	13
Plazas	34
Restaurantes	12

Tabla 21. Categorías principales obtenidas luego del evento de entrenamiento

Fuente: elaboración propia

Al momento de ejecutar el evento de prueba del modelo de clasificación se utilizó el 20 % de los datos del conjunto de datos principales. Estos registros se trasladaron a un archivo en formato TXT y fueron cargados como un activo a través del apartado de prueba provisto por el Entorno Watson. Luego de realizar la respectiva carga de la información, se procedió a iniciar el evento de prueba del modelo. Durante dicho evento, este analizó cada ejemplo del archivo y lo clasificó retornando las categorías que más coincidían con la frase evaluada, incluyendo la categoría principal como se puede apreciar en la Figura 24.

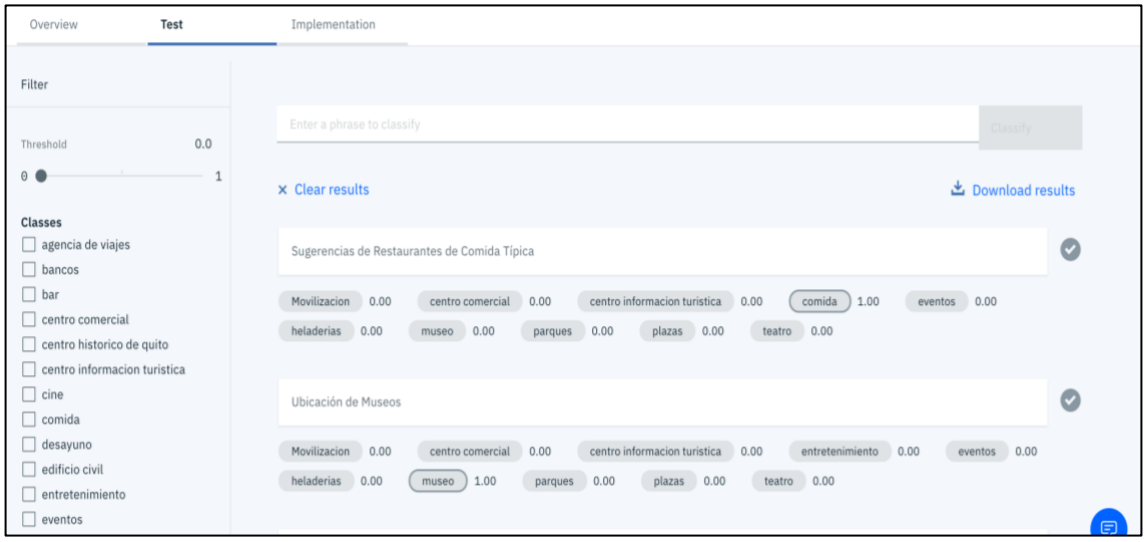


Figura 24. Evento de prueba del modelo

Fuente: elaboración propia

4.2 Producto final

La aplicación web consta de dos pantallas: una pantalla principal que presenta información de uso de la aplicación; y la pantalla de búsqueda y presentación de resultados. Esta última consta de un buscador que permite a los usuarios consultar los sitios turísticos de su preferencia. Además, en esta se despliegan en un mapa las ubicaciones de los sitios turísticos obtenidos, los cuales son representados con marcadores de color rojo, mientras que la posición actual del usuario es de color azul.

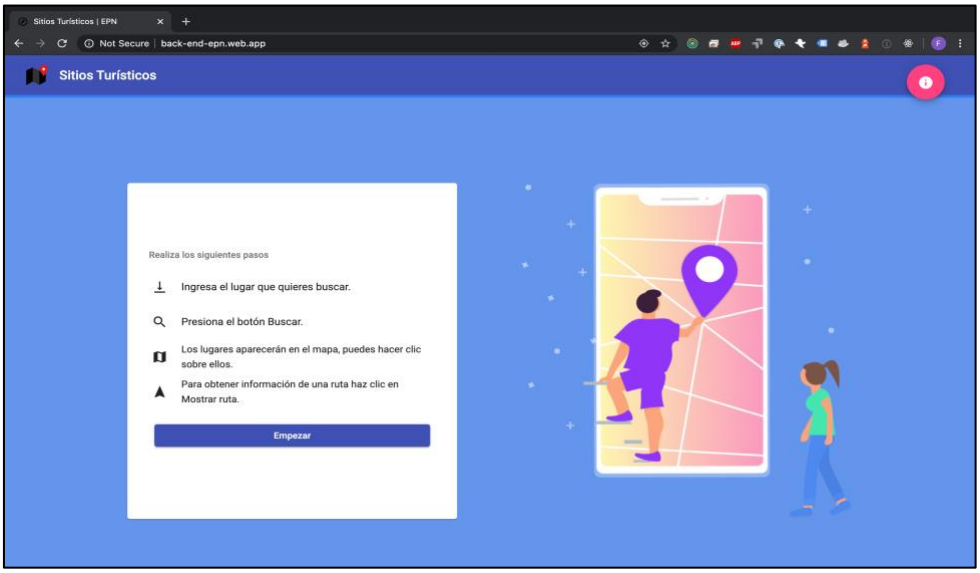


Figura 25. Pantalla Inicial de la aplicación

Fuente: elaboración propia

En la pantalla de búsqueda y presentación de resultados también se despliega la ruta más corta entre la posición actual del usuario y un sitio turístico en específico. Esta ruta puede ser ejemplificada en el mapa por medio de un botón que se encuentra dentro de las ventanas informativas de los marcadores de los sitios turísticos (ver figuras 25 y 26).

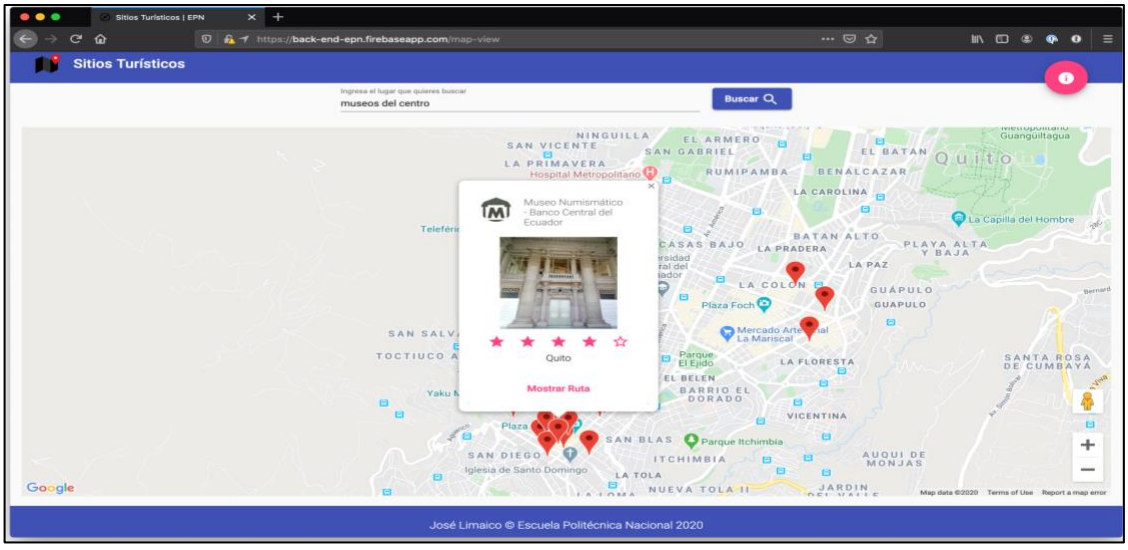


Figura 26. Pantalla de búsqueda y presentación de resultados

Fuente: elaboración propia

- **Toma de datos**

La imagen de consulta autoriza al usuario a ingresar a una frase a través de un campo de texto. La Figura 27 corresponde al campo de texto empleado para recibir el criterio de

búsqueda que va a ser enviado hacia el servicio de PLN NLC. Todo esto, para posteriormente obtener los sitios turísticos que serán remitidos mediante el servidor RESTful.

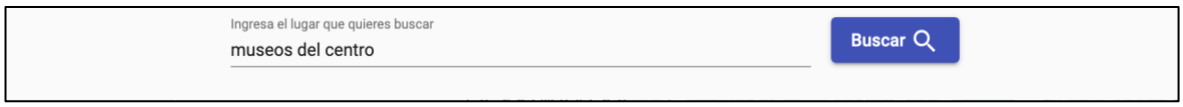


Figura 27. Campo de texto utilizado para obtener el criterio de búsqueda de consulta

Fuente: elaboración propia

- **Sitios turísticos obtenidos**

La Figura 28 corresponde al resultado global de todos los sitios turísticos obtenidos luego de realizar una consulta en particular, en este caso: “museos del centro”. Cada sitio turístico, representado por un marcador de color rojo, está colocado en el mapa en la ubicación actual donde se encuentra el sitio turístico, tal como se aprecia en la Figura 28.

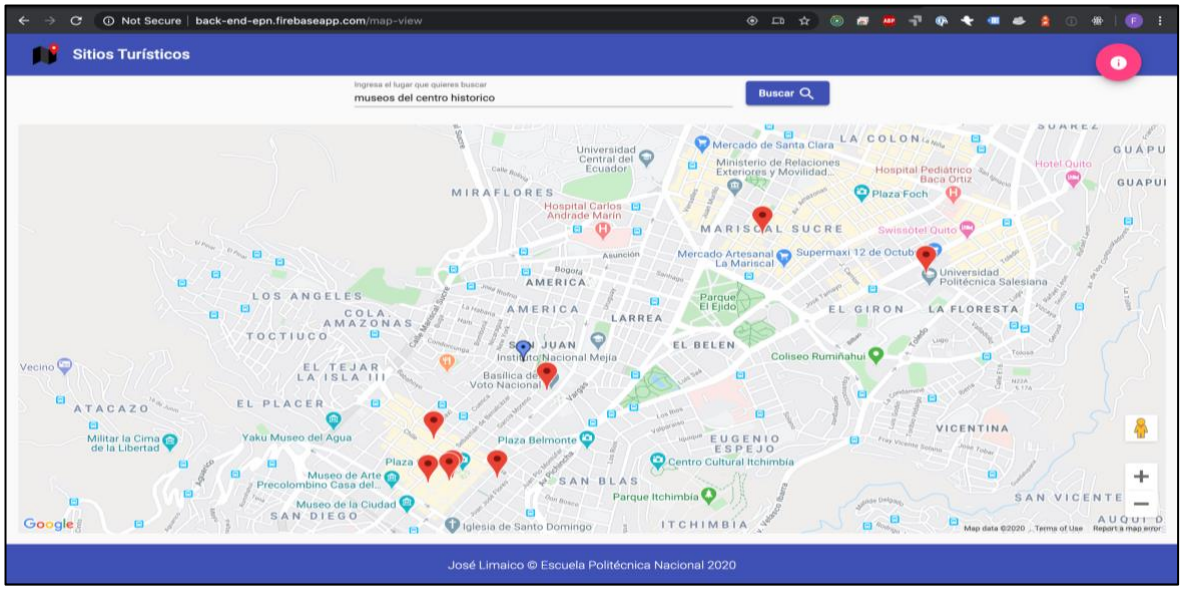


Figura 28. Ubicación de los sitios turísticos representados en el mapa

Fuente: elaboración propia

- **Presentación de información**

Luego de seleccionar cualquier marcador en el mapa, se despliega una tarjeta informativa que contiene información acerca del sitio turístico seleccionado, por ejemplo: nombre, imagen, ícono de referencia, calificación obtenida y un botón para mostrar la ruta más cercana desde la posición actual del usuario hacia el sitio turístico (ver Figura 29).

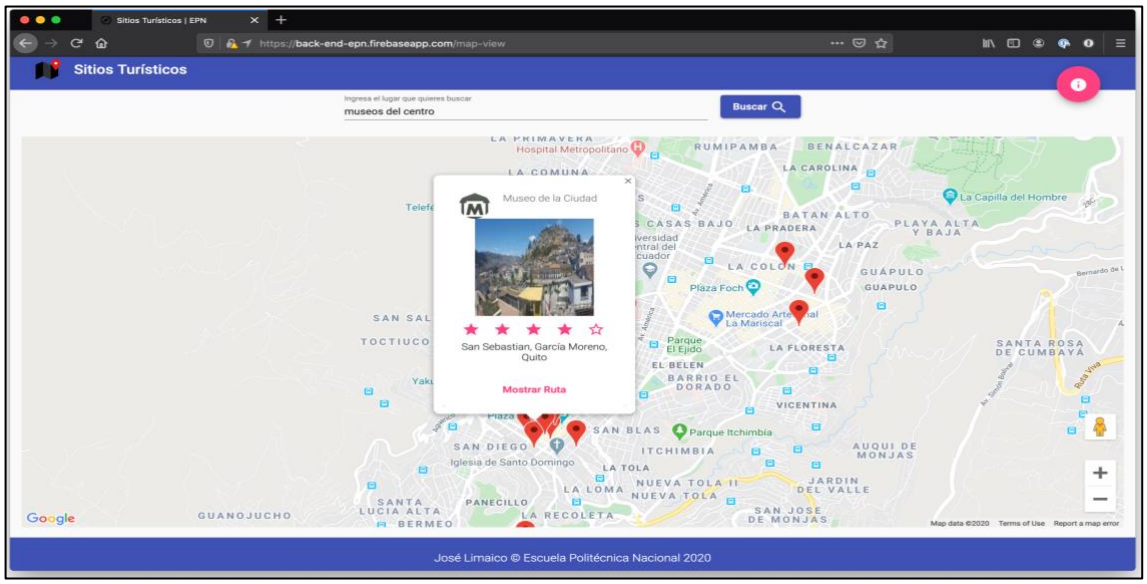


Figura 29. Tarjeta informativa del sitio turístico "museo de la ciudad"

Fuente: elaboración propia

Al momento de hacer clic en el botón “Mostrar Ruta” se despliega un panel lateral que señala al usuario el itinerario de direcciones que debe seguir para llegar desde su posición actual hacia la ubicación del sitio turístico seleccionado. De acuerdo con ello, en la Figura 30 se detalla la representación en el mapa de la ruta más corta entre la posición del usuario y la ubicación del sitio turístico (color azul).

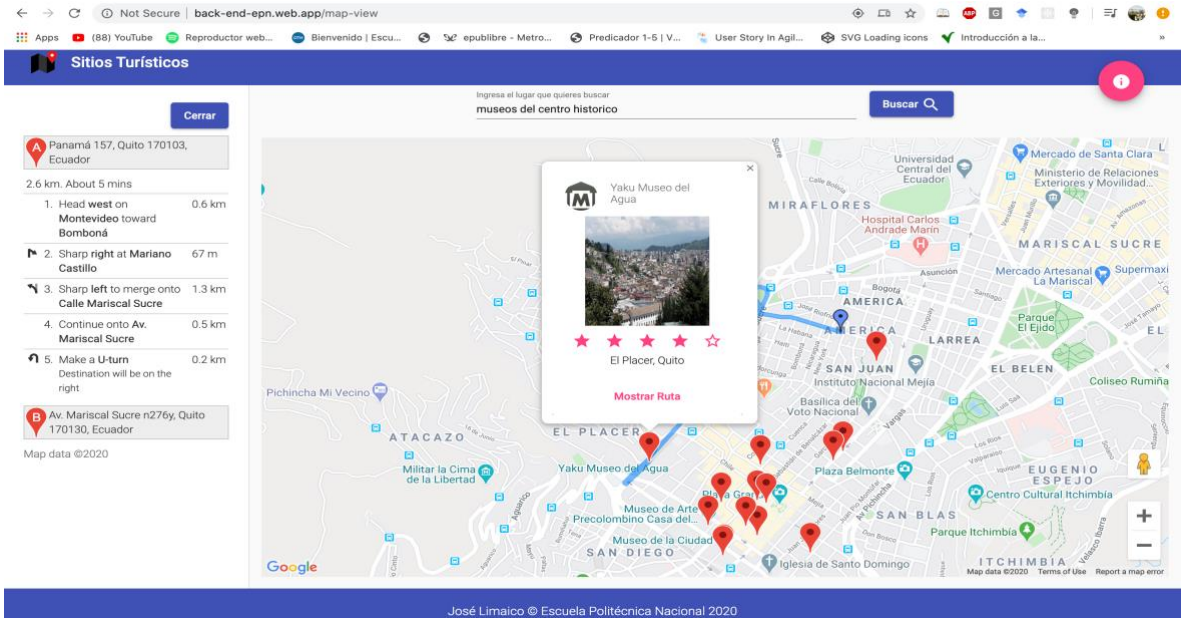


Figura 30. Representación de la ruta más corta entre la ubicación del sitio turístico y la posición actual del usuario

Fuente: elaboración propia

4.3 Análisis de consultas realizadas

En la Figura 31 se observa el número de consultas que realizaron los usuarios en la aplicación web y su respectivo porcentaje con relación al total. Con el objetivo de cumplir esto, se hizo uso de Google Analytics para capturar cada sentencia de texto enviada por los usuarios. Este texto está asociado a un evento de búsqueda previamente configurado. A partir de los resultados obtenidos por medio de Google Analytics, se prueba de que existieron un total de noventa consultas (eventos) realizadas por los usuarios. De estas consultas, se determinó lo siguiente: el 22,4 % perteneció a la categoría de iglesias; el 15,56 % a la categoría de museos; el 6,66 % a bares y plazas, respectivamente. De igual manera, el 3,33 % a hospitales y hoteles y, por último, el 1,11 % a restaurantes. Por otro lado, los porcentajes restantes correspondieron a sentencias de texto que derivaron de las categorías principales ya mencionadas, estas fueron: monasterios, comida, puntos de información turística, entre otros (ver Figura 31).

	Etiqueta de evento	Total de eventos	Eventos únicos
		90 % del total: 100,00 % (90)	49 % del total: 100,00 % (49)
1.	iglesias	22 (24,44 %)	4 (8,16 %)
2.	museos	14 (15,56 %)	3 (6,12 %)
3.	bares	6 (6,67 %)	2 (4,08 %)
4.	plazas	6 (6,67 %)	4 (8,16 %)
5.	hospitales	3 (3,33 %)	1 (2,04 %)
6.	hoteles	3 (3,33 %)	2 (4,08 %)
7.	iglesia	2 (2,22 %)	1 (2,04 %)
8.	informacion	2 (2,22 %)	1 (2,04 %)
9.	monasterios	2 (2,22 %)	2 (4,08 %)
10.	museos del centro historico	2 (2,22 %)	2 (4,08 %)
11.	Páramo de el ángel	2 (2,22 %)	1 (2,04 %)
12.	parques	2 (2,22 %)	2 (4,08 %)
13.	arco	1 (1,11 %)	1 (2,04 %)
14.	arco de san diego	1 (1,11 %)	1 (2,04 %)
15.	Basilica	1 (1,11 %)	1 (2,04 %)
16.	casas	1 (1,11 %)	1 (2,04 %)
17.	Centros de información turística	1 (1,11 %)	1 (2,04 %)

Figura 31. Consultas realizadas por los usuarios en el sistema de recomendaciones

Fuente: elaboración propia

En definitiva, Google Analytics permitió registrar la cantidad de eventos únicos enviados por la aplicación web. Cabe resaltar que los eventos únicos son una contabilización de un

evento que tiene asignado una categoría específica, la cual ha aparecido al menos una vez dentro de una sesión. De acuerdo con la figura, los resultados reflejan que existió un total de 49 eventos únicos relacionados a las categorías principales ya mencionadas. Estos eventos aparecieron de la siguiente manera en una misma sesión: iglesias (al menos cuatro veces), museos (aprox. tres veces), bares (aprox. dos veces), plazas (al menos cuatro veces), hospitales (aprox. una vez), hoteles (al menos dos veces), centros de información turística (aprox. una vez) y restaurantes (al menos una vez). Con respecto al resto de eventos, ocurrió lo mismo que con los eventos principales, es decir, las sentencias de texto que se relacionaron a eventos únicos se derivaron de las categorías principales ya mencionadas.

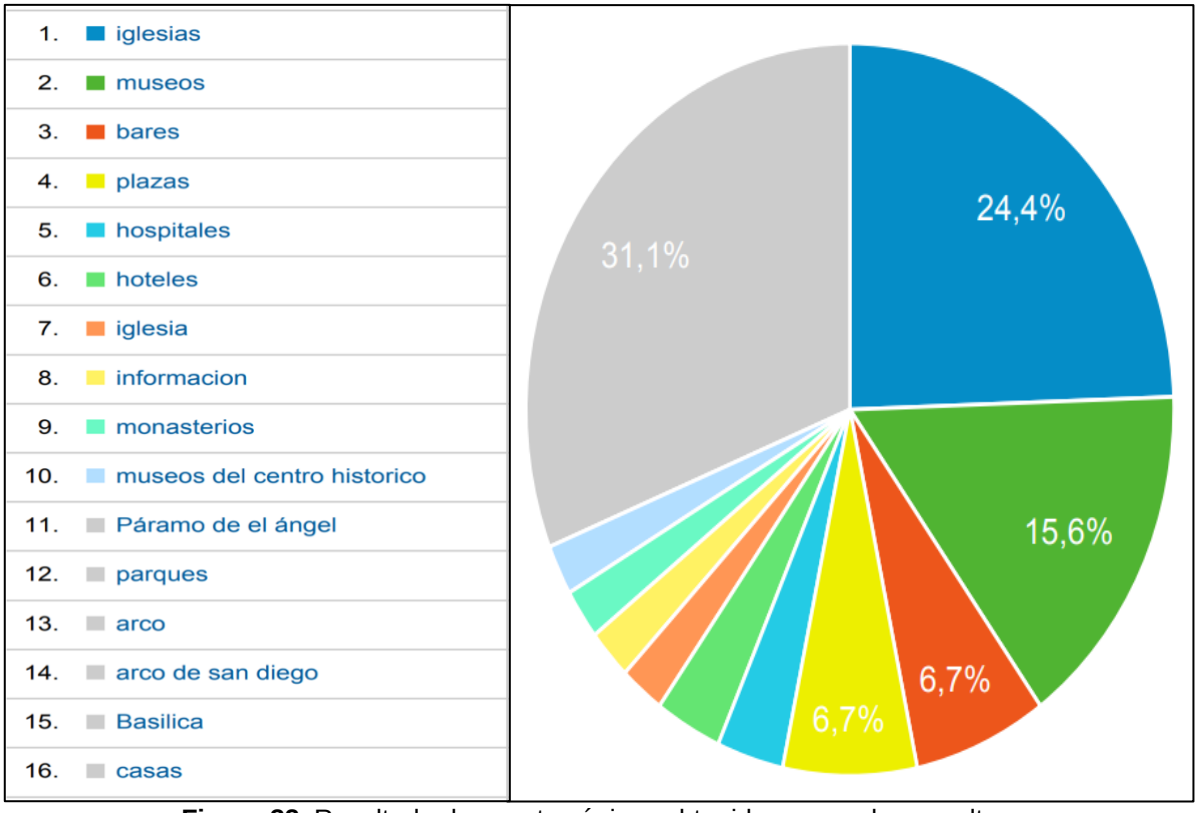


Figura 32. Resultado de eventos únicos obtenidos por cada consulta

Fuente: elaboración propia

4.4 Pruebas con usuarios finales

Se realizaron pruebas del sistema de recomendaciones por medio de la evaluación de la aplicación web. Se evaluaron a un total de 15 personas que trabajaban en Kushki, una empresa de comercio electrónico. En las pruebas realizadas se evaluó la utilidad y facilidad de uso percibidas, así como también la adaptabilidad por parte del usuario hacia el uso de la aplicación y su actitud con respecto al uso de esta. Para la evaluación se implementó el

modelo TAM (Technology Acceptance Model) [36]. Según este modelo, se efectuaron un total de 13 preguntas por cada usuario de acuerdo con su percepción: cinco para la utilidad, cuatro para la facilidad de uso, dos para la actitud con respecto al manejo y dos para la adaptabilidad. La respuesta de cada pregunta se ajustó a una escala del 1 al 7 (donde 1 es la peor calificación y 7 es la mejor). Los cuestionamientos realizados y sus resultados se encuentran adjuntados en el Anexo VI.

4.4.1 Utilidad percibida.

En la Figura 33 se puede advertir el promedio de respuestas para cada pregunta con respecto a la utilidad percibida. Si se tienen en cuenta los resultados obtenidos, se afirma que la pregunta cinco –la cual hace referencia al grado de utilidad que ofrece el sistema para obtener recomendaciones de sitios turísticos del centro histórico–, tuvo el mayor promedio (6,57/7). Por lo tanto, se demuestra que los usuarios percibieron un alto grado de utilidad ofrecida por parte del sistema de recomendaciones mediante el uso de la aplicación web.

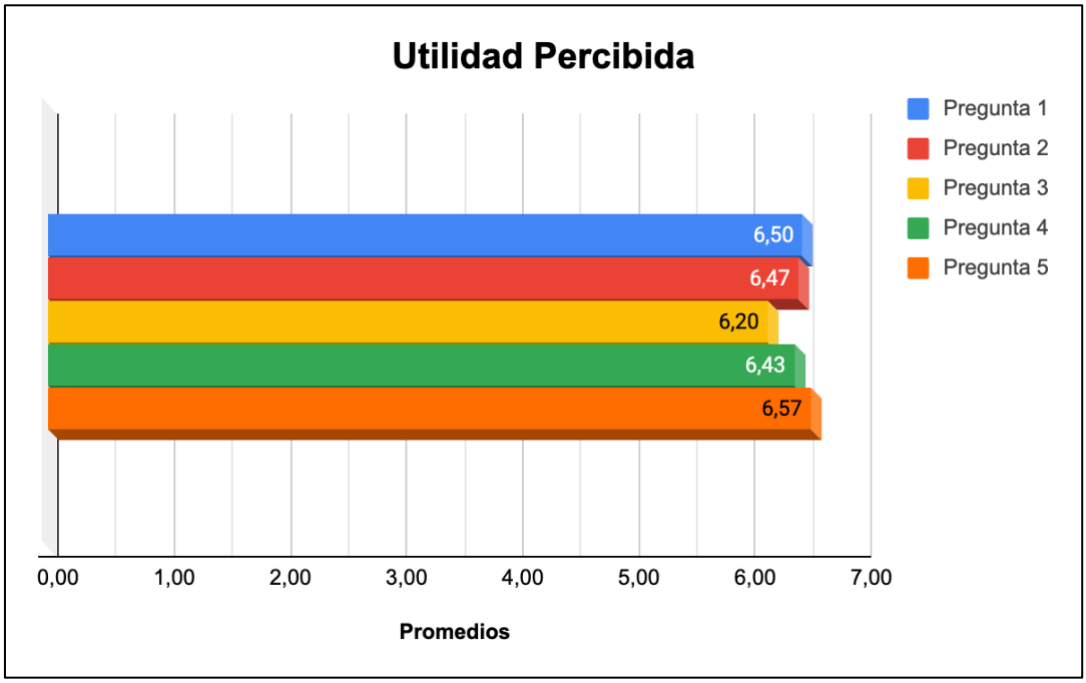


Figura 33. Promedio por pregunta de la utilidad percibida en el sistema de recomendaciones

Fuente: elaboración propia

4.4.2 Facilidad de uso percibida.

La Figura 34 indica los promedios obtenidos por cada pregunta, la cual se relaciona con la facilidad de uso percibida por los usuarios, luego de que utilizaron el sistema de

recomendaciones. En cierto sentido, de los resultados obtenidos se puede demostrar que la pregunta cuatro –esta hace referencia al nivel de dificultad de uso que ofrece el sistema para obtener recomendaciones de sitios turísticos del centro histórico– tuvo el mayor promedio (6,71/7). Esta cifra evidencia que los usuarios no percibieron mayores inconvenientes al momento de usar el sistema de recomendaciones.

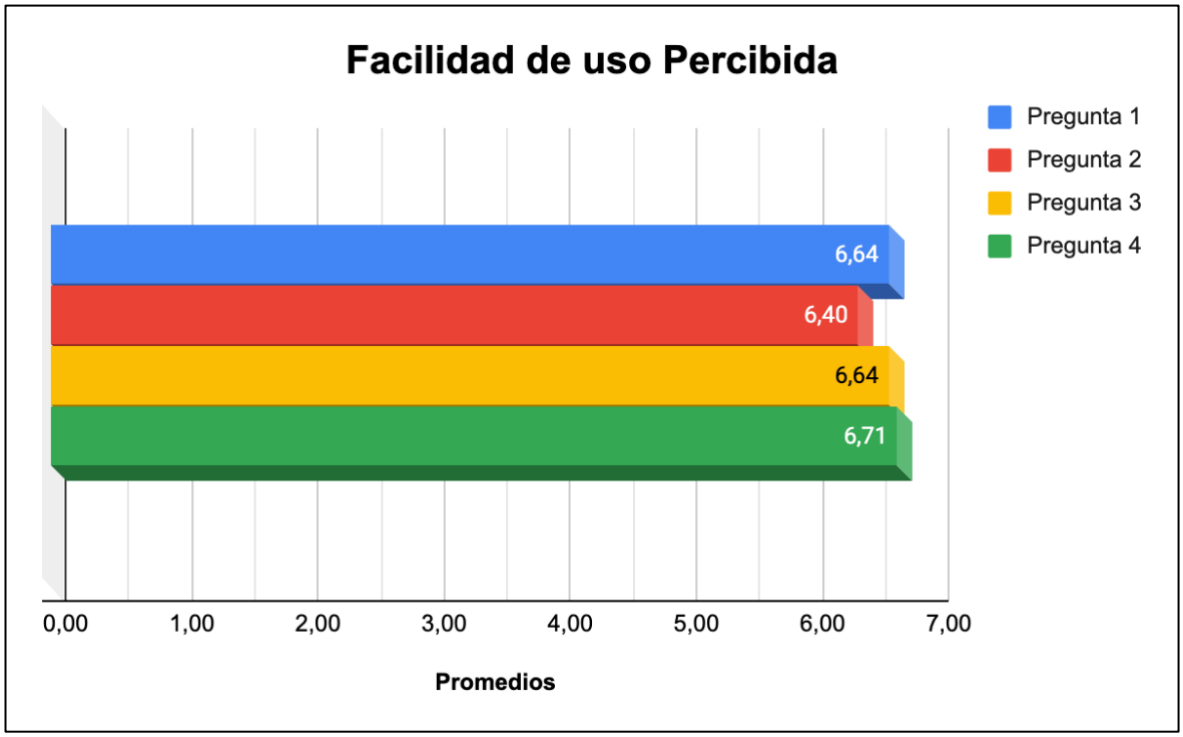


Figura 34. Promedio por pregunta de la facilidad de uso percibida en el sistema de recomendaciones

4.4.3 Actitud con respecto al uso.

En la siguiente figura se señalan los promedios obtenidos por cada pregunta que se relaciona con la actitud respecto al uso de los usuarios, luego de que utilizaron el sistema de recomendaciones. Según los resultados obtenidos, se evidencia que las preguntas uno y dos –las cuales hacen referencia a la intención de uso por parte de los usuarios para obtener recomendaciones– tuvieron resultados de 6,57 y 6,64, respectivamente. De lo anterior se deduce que los usuarios tienen intención de usar el sistema de recomendaciones y lo usarían en ocasiones posteriores.

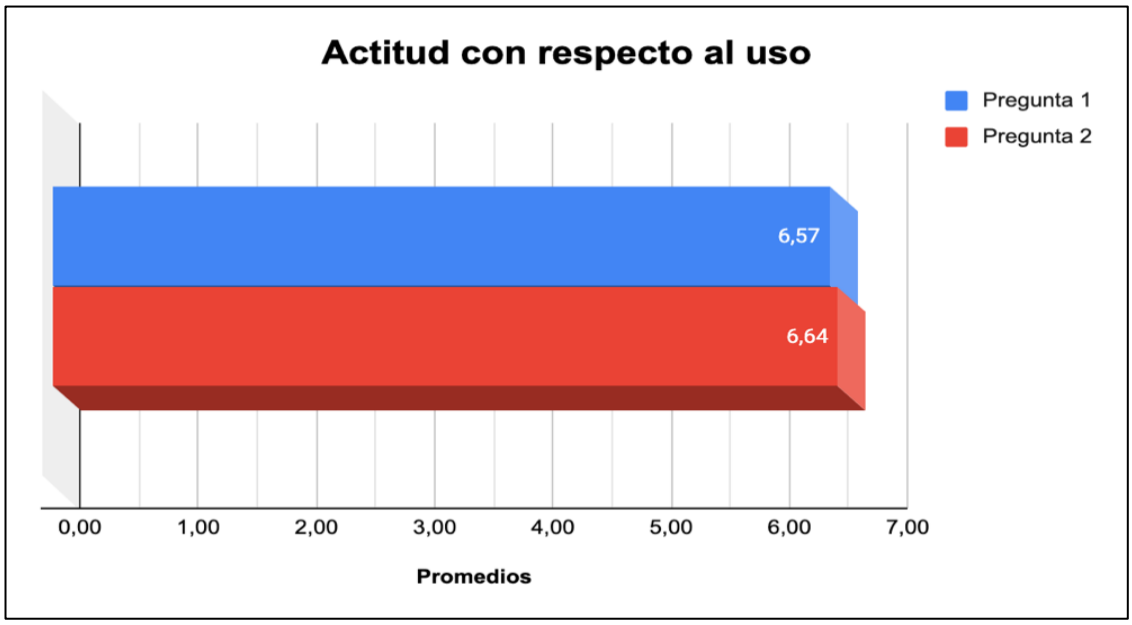


Figura 35. Promedio por pregunta de la actitud con respecto al uso para el sistema de recomendaciones

Fuente: elaboración propia

4.4.4 Adaptabilidad percibida.

La Figura 36 señala los promedios resultantes de las dos preguntas que se relacionan con la adaptabilidad percibida por los usuarios, posterior a su utilización en el sistema de recomendaciones. Según los resultados obtenidos, se concluye que las preguntas uno y dos –estas aluden al conocimiento de cómo funciona el sistema de recomendaciones– tuvieron resultados satisfactorios: 6,07 y 6,29, respectivamente. En concreto, los usuarios tienen conocimiento del funcionamiento del sistema de recomendaciones y también saben que el sistema utiliza PLN para obtener las recomendaciones.

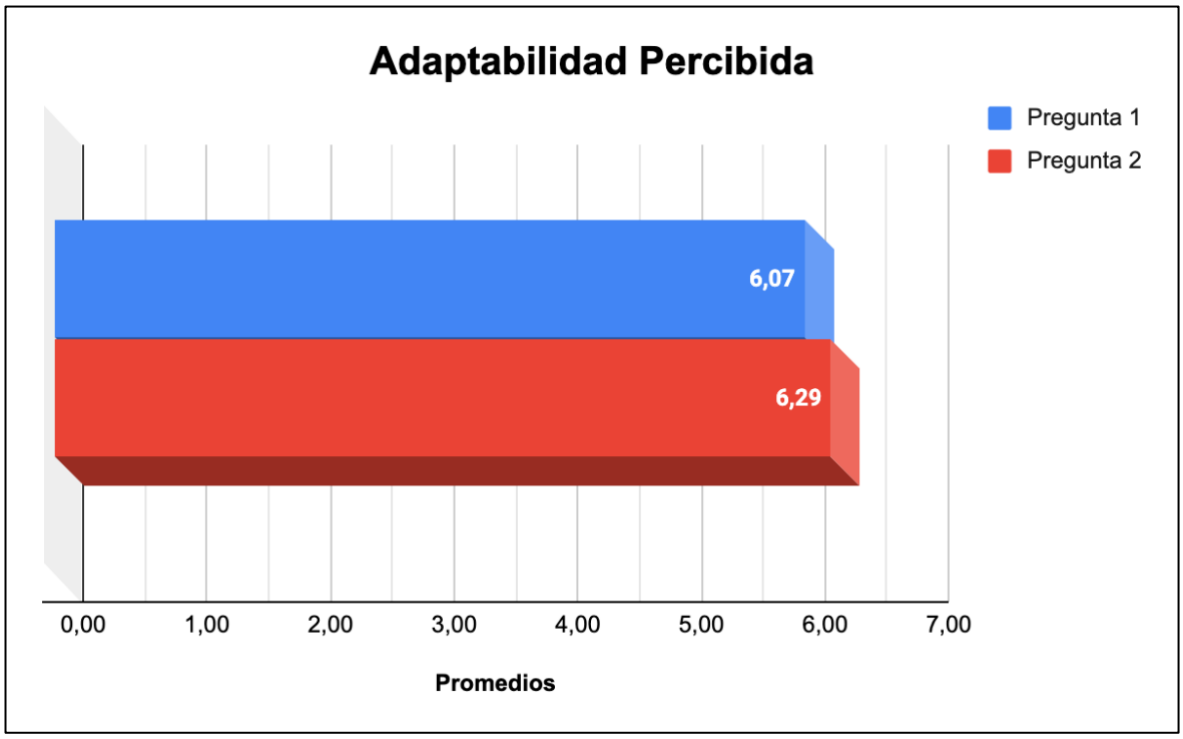


Figura 36. Promedio por pregunta de adaptabilidad percibida en el sistema de recomendaciones

Fuente: elaboración propia

5 CONCLUSIONES Y RECOMENDACIONES

En el presente documento, se explicó un modelo de clasificación de lenguaje natural que permite generar recomendaciones de sitios turísticos para el centro histórico de Quito a través del procesamiento de sentencias de texto ingresadas por los usuarios. Como se señaló, el modelo se entrenó a partir de un conjunto de datos que fue estructurado con base a encuestas realizadas; de esto, se obtuvo un total de 685 registros con sentencias de texto y su respectiva categoría. De este conjunto de datos, se utilizó el 80 % en el evento de entrenamiento y el 20 % de los datos restantes sirvió para probar el prototipo mencionado. Hay que destacar que el modelo de clasificación empleó enfoques de aprendizaje profundo como RNC, facilitando mejorar la eficacia en la clasificación de sentencias de texto con conjuntos de datos de tamaño reducido. Finalmente, este se implementó en un sistema de recomendaciones por medio de una aplicación web, la cual se desarrolló para probar el modelo de clasificación entrenado; esto se desarrolló a través de la llamada al servicio de IBM Watson NLC.

A partir de este estudio, se demostraron resultados satisfactorios con respecto a la consulta de los sitios turísticos por medio de la aplicación web. Los resultados de las encuestas realizadas, para medir el nivel de usabilidad y utilidad percibidas de la aplicación, la evaluación de la adaptabilidad percibida y, asimismo, la actitud de los usuarios, con respecto al uso, demuestra que es posible generar recomendaciones de sitios turísticos, a partir del análisis y procesamiento de sentencias de texto enviadas por los usuarios. Todo esto generó el cumplimiento a la pregunta de investigación y demostró la hipótesis propuesta para el desarrollo del presente proyecto.

Luego de implementar el sistema de recomendaciones, se logró cumplir el objetivo principal y los objetivos específicos planteados en el presente documento. Estos abarcan principalmente la implementación de un servicio de procesamiento y clasificación de lenguaje natural y, a su vez, la adecuación de un sistema de recomendaciones de sitios turísticos que empleó el servicio mencionado como su principal fuente de procesamiento.

El presente proyecto brindó una alternativa para obtener recomendaciones de sitios turísticos a partir del PLN mediante enfoques de aprendizaje profundo y aprendizaje automático implementados en un modelo de clasificación, el cual fue adecuado como servicio. Para trabajos futuros se propone el análisis y comparación de los servicios de PLN que existen actualmente, con el fin de validar las técnicas de aprendizaje automático y enfoques de aprendizaje profundo que se utilizan en los algoritmos de PLN, según la documentación afirma.

6 REFERENCIAS

[1] T. Young, D. Hazarika, S. Poria y E. Cambria, «Recent trends in deep learning based natural language processing,» *IEEE Comput. Intell. Mag.,* vol. 3, nº 3, pp. 55-75, 2018.

[2] K. Leitmeyer y C. Adlhoch, «Jumping NLP Curves: A Review of Natural Language Processing Research,» *IEEE Computational Intelligence Magazine,* vol. 9, nº 2, pp. 48-57, 2014.

[3] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton y J. Leskovec, «Graph convolutional neural networks for web-scale recommender systems,» *KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 974-983, 2018.

[4] S. Zhang, L. Yao, A. Sun y Y. Tay, «Deep learning based recommender system: A survey and new perspectives,» *ACM Computing Surveys,* vol. 52, nº 1, pp. 1-35, 2019.

[5] R. Collobert y J. Weston, «A unified architecture for natural language processing: Deep neural networks with multitask learning,» *ICML '08: Proceedings of the 25th international conference on Machine learning,* pp. 60-167, 2008.

[6] Portal Servicios MINTUR, «Turismo en Cifras,» 24 mayo 2020. [En línea]. Available: https://servicios.turismo.gob.ec/index.php/turismo-cifras.

[7] M. Ávila, «El aporte del Distrito Metropolitano de Quito al desarrollo del turismo en el Ecuador y al crecimiento económico del país en el periodo 2017-2020 [Tesis de maestría],» Universidad Andina Simón Bolívar, Sede Ecuador, Distrito Metropolitano de Quito, 2017.

[8] X. Su y T. M. Khoshgoftaar, «A Survey of Collaborative Filtering Techniques,» *Advances in Artificial Intelligence,* vol. 2009, pp. 1-19, 2009.

[9] R. Burke, «Hybrid recommender systems: Survey and experiments,» *User Modeling and User-Adapted Interaction,* vol. 12, nº 4, pp. 331-370, 2002.

[10] I. Fernández-Tobías, I. Cantador, M. Kaminskas y F. Ricci, «Cross-domain recommender systems : A survey of the State of the Art,» *Spanish Conference on Information Retrieval,* pp. 187-198, 2012.

[11] M. Khan, R. Ibrahim y I. Ghan, «Cross Domain Recommender Systems,» *ACM Computing Surveys,* vol. 50, nº 3, pp. 1-34, 2017.

[12] B. Tilahun, C. Awono y B. Batchakui, «A Survey of State-of-the-art: Deep Learning Methods on Recommender System,» *International Journal of Applied Mathematics and Computer Science,* vol. 162, nº 10, pp. 17-22, 2017.

[13] A. V. d. Oord, S. Dieleman y B. Schrauwen, «Deep content-based music recommendation,» Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 2013.

[14] B. Uria, M. A. Cote, K. Gregor, I. Murray y H. Larochelle, «Neural autoregressive distribution estimation,» *Journal of Machine Learning Research,* vol. 17, pp. 1-37, 2016.

[15] H. Wang, N. Wang y D. Yeung, «Collaborative deep learning for recommender systems,» *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* vol. 2015, pp. 1235-1244, 2015.

[16] Liu y W. Caihua, «Deep Learning Based Recommendation: A Survey,» *Lecture Notes in Electrical Engineering,* vol. 2, pp. 467-475, 2017.

[17] G. Adomavicius y A. Tuzhilin, «Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions Gediminas,» *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, nº 6, p. 734, 2005.

[18] F. Ricci, B. Shapira y L. Rokach, Recommender systems handbook, Second ed., Springer, 2015.

[19] D. Jannach, M. Zanker, A. Felferning y G. Friedrich, Recommender Systems: An Introduction, Cambridge: Cambridge University Press, 2012.

[20] A. Karatzoglou y B. Hidas, «Deep Learning for Recommender Systems,» 30 agosto 2017. [En línea]. Available: https://es.slideshare.net/kerveros99/deep-learning-for-recommender-systems-recsys2017-tutorial.

[21] IBM Developer, «Deep learning architectures – IBM Developer,» [En línea]. Available: https://developer.ibm.com/technologies/deep-learning/articles/cc-machine-learning-deep-learning-architectures. [Último acceso: junio 13 2020].

[22] J. Hurwitz y D. Kirsch, Machine Learning For Dummies®, IBM Limited Edition, Hoboken: John Wiley & Sons, Inc., 2018.

[23] Aws Amazon, «Características de Amazon Comprehend,» 2020. [En línea]. Available: https://aws.amazon.com/es/comprehend/. [Último acceso: 7 junio 2020].

[24] IBM Cloud, «About,» 2020. [En línea]. Available: https://cloud.ibm.com/docs/natural-language-classifier?topic=natural-language-classifier-about. [Último acceso: 2 junio 2020].

[25] G. Cloud, «Cloud Natural Language,» 2020. [En línea]. Available: https://cloud.google.com/natural-language. [Último acceso: 7 junio 2020].

[26] Express, «Node.js web application framework,» 2020. [En línea]. Available: https://expressjs.com/.. [Último acceso: 16 mayo 2020].

[27] Angular, «Angular,» 2020. [En línea]. Available: https://angular.io/. [Último acceso: 16 mayo 2020].

[28] MongoDB, «La base de datos líder del mercado para aplicaciones modernas,» 2020. [En línea]. Available: https://www.mongodb.com/e. [Último acceso: 16 mayo 2020].

[29] JetBrains, «WebStorm: The Smartest JavaScript IDE by JetBrains,» 2020. [En línea]. Available: https://www.jetbrains.com/webstorm/. [Último acceso: 16 mayo 2020].

[30] D. Gavalas y M. Kenteri, «A web-based pervasive recommendation system for mobile tourist guides,» *Personal and Ubiquitous Computing,* vol. 15, nº 17, pp. 759-770, 2011.

[31] Google Marketing Platform, «Herramientas y servicios de análisis para su empresa - Google Analytics,» 2020. [En línea]. Available: https://marketingplatform.google.com/intl/es/about/analytics/. [Último acceso: 18 julio 2020].

[32] K. Schwaber y J. Sutherland, «The Scrum Guide: The Definitive The Rules of the Game,» *Scrum.Org and ScrumInc,* p. 19, 2017.

[33] Data preparation, «Data preparation,» *Lecture Notes in Business Information Processing,* vol. 207, pp. 71-83, 2015.

[34] IBM Cloud, «Preparación de los datos,» 2020. [En línea]. Available: https://cloud.ibm.com/docs/natural-language-classifier?topic=natural-language-classifier-using-your-data&locale=es. [Último acceso: 1 agosto 2020].

[35] Mangoose, «Mongoose ODM v5.9.19,» 2020. [En línea]. Available: https://mongoosejs.com/.. [Último acceso: 21 junio 2020].

[36] F. D. Davis, «A Technology Acceptance Model For Empirically Testing New End-User Information Systems: Theory And Results,» *Science ,* vol. 146, nº 3652, pp. 1648-1655, 1985.

7 ANEXOS

7.1 Anexo I: Historias Épicas

HISTORIA DE USUARIO ÉPICA.	
ID: ES01	**Prioridad:** alta
Título: entrenamiento e implementación del Clasificador de lenguaje natural.	
Descripción: como usuario, necesito obtener recomendaciones de sitios turísticos a partir de consultas en forma de texto realizadas al sistema.	

Tabla 22. Historia Épica ES01

Fuente: elaboración propia

HISTORIA DE USUARIO ÉPICA	
ID: ES02	**Prioridad:** alta
Título: desarrollo del servidor RESTful API.	
Descripción: como usuario, necesito obtener sitios turísticos de acuerdo con las categorías de cada consulta realizada.	

Tabla 23. Historia Épica ES02

Fuente: elaboración propia

HISTORIA DE USUARIO ÉPICA	
ID: ES03	**Prioridad:** media
Título: desarrollo de la aplicación web para probar el sistema de recomendaciones por parte del usuario final.	
Descripción: como usuario, necesito una interfaz amigable, de manera que pueda visualizar los sitios turísticos generados de acuerdo con mis preferencias.	

Tabla 24. Historia Épica ES03

Fuente: elaboración propia

7.2 Anexo II: Historias de usuario del Sprint 1

<table>
<tr><th colspan="4">HISTORIA DE USUARIO</th></tr>
<tr><td colspan="3">ID: ES01-01</td><td>Días estimados: 7</td></tr>
<tr><td colspan="4">Título: construcción del modelo de clasificación para el servicio NLC</td></tr>
<tr><td>Sprint No: 1</td><td colspan="2">Prioridad: alta</td><td>Estado: por implementar.</td></tr>
<tr><td colspan="4">Descripción: como usuario, quiero obtener recomendaciones de sitios turísticos a partir de consultas de texto para conocer lugares turísticos de acuerdo con mis preferencias.</td></tr>
<tr><td colspan="4">Criterios de aceptación</td></tr>
<tr><td>Dado que</td><td colspan="2">Cuando</td><td>Entonces</td></tr>
<tr><td>Se crea un nuevo asset con el conjunto de datos de entrenamiento.</td><td colspan="2">Se entrena el modelo de clasificación de lenguaje natural.</td><td>Se muestra una tabla de resumen con la información de entrenamiento del modelo entrenado satisfactoriamente.</td></tr>
</table>

Tabla 25. Definición de la Historia de usuario ES01-01

Fuente: elaboración propia

<table>
<tr><th colspan="4">HISTORIA DE USUARIO</th></tr>
<tr><td colspan="3">ID: ES01-02</td><td>Días estimados: 1</td></tr>
<tr><td colspan="4">Título: implementación del clasificador de lenguaje natural.</td></tr>
<tr><td>Sprint No: 1</td><td>Prioridad: alta</td><td colspan="2">Estado: por implementar.</td></tr>
<tr><td colspan="4">Descripción: como usuario, quiero obtener recomendaciones de sitios turísticos a partir de consultas de texto para conocer lugares turísticos de acuerdo con mis preferencias.</td></tr>
<tr><td colspan="4">Criterios de aceptación</td></tr>
<tr><td>Dado que</td><td>Cuando</td><td colspan="2">Entonces</td></tr>
<tr><td>Se crea un nuevo asset con el conjunto de datos de prueba.</td><td>Se utiliza este asset para probar el funcionamiento del modelo.</td><td colspan="2">El modelo de clasificación devuelve las categorías correctas para cada registro del conjunto de datos de prueba.</td></tr>
</table>

Tabla 26. Definición de la Historia de Usuario ES01-02

Fuente: elaboración propia

HISTORIA DE USUARIO		
ID: ES01-03		**Días estimados:** 1
Título: consumir servicio de IBM NLC.		
Sprint No: 1	**Prioridad:** alta	**Estado:** por implementar.
Descripción: como usuario, quiero obtener recomendaciones de sitios turísticos a partir de consultas de texto para conocer lugares turísticos de acuerdo con mis preferencias.		
Criterios de aceptación		
Dado que	**Cuando**	**Entonces**
Se implementa el modelo de clasificación de lenguaje natural.	El usuario consume el modelo de clasificación por medio del servicio NLC. Enviando un texto de consulta.	Se obtienen todas las categorías relacionadas al texto de consulta con su respectivo nivel de coincidencia en porcentaje.

Tabla 27. Definición de la Historia de Usuario ES01-03

Fuente: elaboración propia

7.3 Anexo III: Historias de usuario del Sprint 2

HISTORIA DE USUARIO		
ID: ES02-01		**Días estimados:** 6
Título: creación de API RESTful para consumir peticiones HTTP.		
Sprint No: 2	**Prioridad:** alta	**Estado:** por implementar.
Descripción: como usuario, quiero obtener recomendaciones de sitios turísticos de acuerdo con mis intereses.		
Criterios de aceptación		
Dado que	**Cuando**	**Entonces**
Se consume el endpoint/categories.	Se envía la consulta realizada por el usuario.	Se obtiene como resultado una lista de todos los sitios turísticos que coincidan con las preferencias del usuario en formato JSON.
Se consume el endpoint/categories.	**No** se envía la consulta realizada por el usuario.	Se muestra un mensaje de error indicando que la solicitud es inválida.

Tabla 28. Definición de la Historia de Usuario ES02-01

Fuente: elaboración propia

HISTORIA DE USUARIO		
ID: ES02-02		**Días estimados:** 3
Título: conexión con la base de datos no relacional MongoDB para almacenar información de sitios turísticos.		
Sprint No: 2	**Prioridad:** alta	**Estado:** por implementar.
Descripción: como usuario, quiero obtener información de cada sitio turístico de acuerdo con mis intereses para tener más conocimiento de cada sitio turístico.		
Criterios de aceptación		
Dado que	**Cuando**	**Entonces**
Se consume el endpoint/create-category.	Y se envía la información del registro a insertar en la base de datos.	Se obtiene como resultado un JSON con la información del sitio turístico que ha sido creado en la base de datos.
Se consume el endpoint/create-category correctamente.	Se revisa la base de datos.	Se verifica que el registro se ha creado con éxito.

Tabla 29. Definición de la Historia de Usuario ES02-02

Fuente: elaboración propia

7.4 Anexo IV: Historias de usuario del Sprint 3

HISTORIA DE USUARIO		
ID: ES03-01	**Días estimados:** 2	
Título: desarrollo de la interfaz de inicio.		
Sprint No: 3	**Prioridad:** media	**Estado:** por implementar.
Descripción: como turista, quiero saber cómo utilizar la aplicación web para buscar sitios turísticos del centro histórico de Quito.		
Criterios de aceptación		
Dado que	**Cuando**	**Entonces**
El usuario desea consultar sitios turísticos del centro histórico.	El usuario acceda a la pantalla de inicio.	Mostrar una vista informativa que indique cómo usar la aplicación.
El usuario acceda a la pantalla de inicio.	El usuario necesite navegar hacia la vista de consulta.	Mostrar un botón que permita navegar hacia la vista de consulta.

Tabla 30. Definición de la Historia de Usuario ES03-01

Fuente: elaboración propia

HISTORIA DE USUARIO		
ID: ES03-02	**Días estimados:** 3	
Título: desarrollo de la interfaz de búsqueda.		
Sprint No: 3	**Prioridad:** media	**Estado:** por implementar.
Descripción: como usuario, quiero consultar información de sitios turísticos del centro histórico de Quito para tener mayor conocimiento de estos sitios.		

Criterios de aceptación		
Dado que	**Cuando**	**Entonces**
El usuario accede a la pantalla de búsqueda de resultados.	El usuario quiera consultar información de sitios turísticos de su preferencia.	Se debe mostrar un cuadro de texto que permita al usuario ingresar la consulta de los sitios turísticos.
El usuario presiona el botón buscar.	El cuadro de texto está vacío.	Se muestra un mensaje de error indicando que se debe llenar este campo.

Tabla 31. Definición de la Historia ES03-02

Fuente: elaboración propia

<table>
<tr><th colspan="4">HISTORIA DE USUARIO</th></tr>
<tr><td colspan="2">ID: ES03-03</td><td colspan="2">Días estimados: 4</td></tr>
<tr><td colspan="4">Título: desarrollo de la interfaz para mostrar los resultados.</td></tr>
<tr><td>Sprint No: 3</td><td>Prioridad: media</td><td colspan="2">Estado: por implementar.</td></tr>
<tr><td colspan="4">Descripción: como usuario, quiero ver los resultados de la búsqueda realizada para tener conocimiento de la ubicación e información adicional de cada sitio turístico.</td></tr>
<tr><td colspan="4">Criterios de aceptación</td></tr>
<tr><th>Dado que</th><th>Cuando</th><th colspan="2">Entonces</th></tr>
<tr><td>El usuario accede a la pantalla de búsqueda de resultados.</td><td>El usuario realiza la consulta.</td><td colspan="2">Se deben mostrar en un mapa interactivo todos los sitios turísticos representados por un marcador de color rojo.</td></tr>
<tr><td>El usuario realiza la consulta.</td><td>Se muestran los marcadores que representan la ubicación de los sitios turísticos en el mapa.</td><td colspan="2">Mostrar ventanas informativas que se desplegarán al hacer clic sobre los marcadores.</td></tr>
<tr><td>Se muestran las ventanas informativas, sobre los marcadores.</td><td>El usuario hace clic en el botón “Mostrar Ruta”.</td><td colspan="2">Se debe mostrar la ruta más corta en el mapa entre la ubicación actual del usuario y el sitio turístico seleccionado.</td></tr>
</table>

Tabla 32. Definición de la Historia de Usuario ES03-03

Fuente: elaboración propia

7.5 Anexo V: Diseño de las interfaces de usuario final

7.5.1 Mockup 1: pantalla de Información

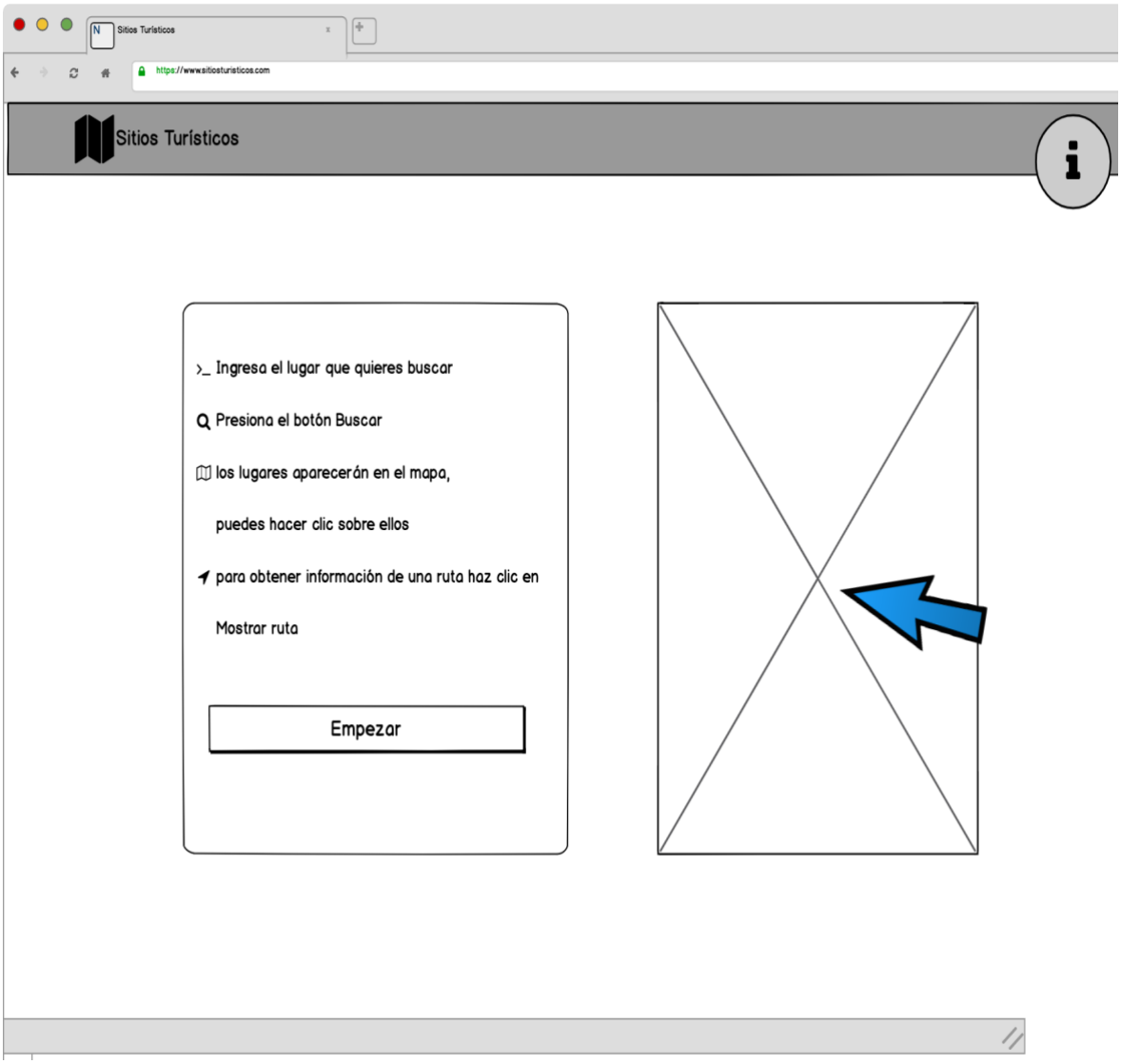


Figura 37. Pantalla informativa de la aplicación

Fuente: elaboración propia

7.5.2 Mockup 2: pantalla de búsqueda y despliegue de resultados

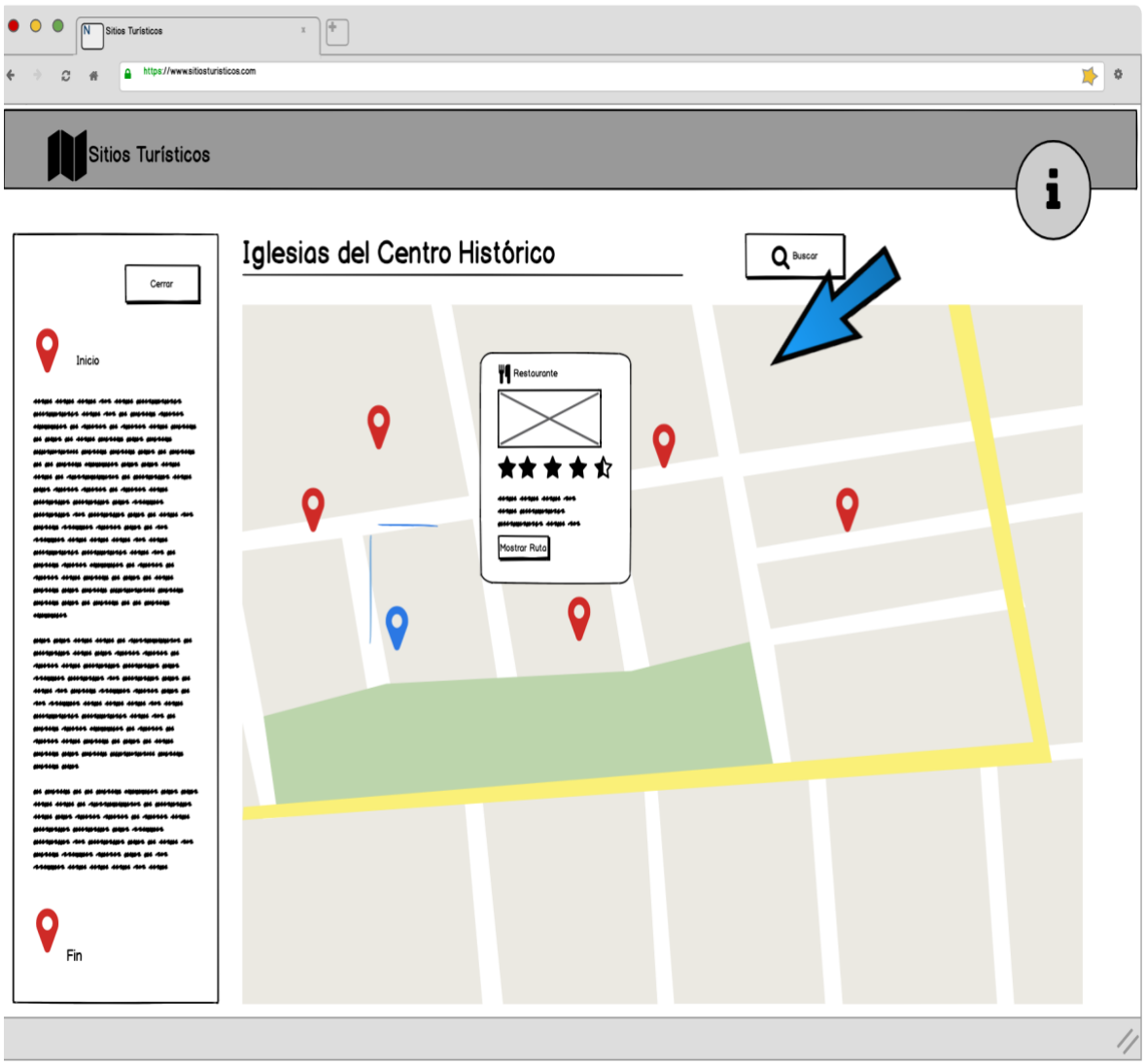


Figura 38. Pantalla de búsqueda y despliegue de resultados

Fuente: elaboración propia

7.5.3 Mockup 3: cuadro de diálogo de información de la arquitectura

Figura 39. Cuadro informativo de la arquitectura del sistema

Fuente: elaboración propia

7.6 ANEXO IV: resultados de las encuestas realizadas

7.6.1 Utilidad percibida

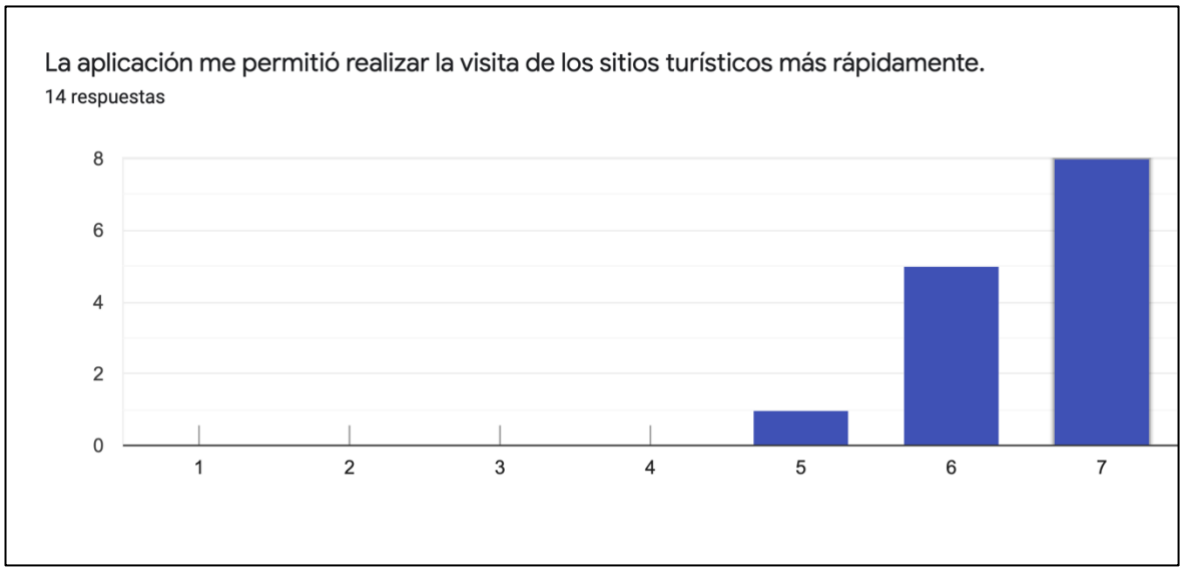


Figura 40. Resultados de la pregunta 1 para la utilidad percibida

Fuente: elaboración propia

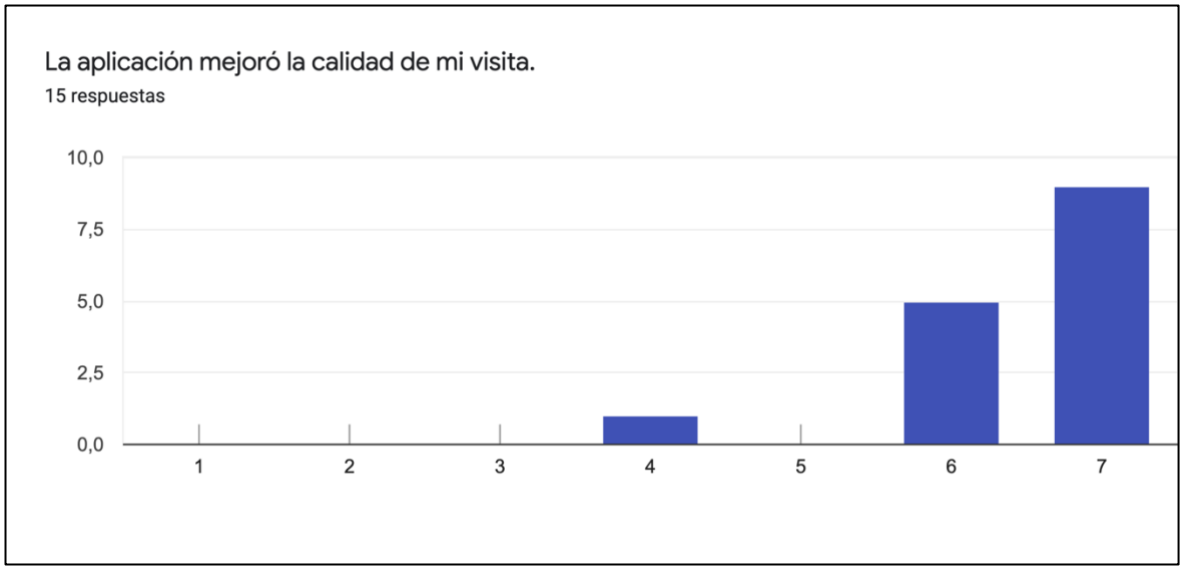


Figura 41. Resultados de la pregunta 2 para la utilidad percibida

Fuente: elaboración propia

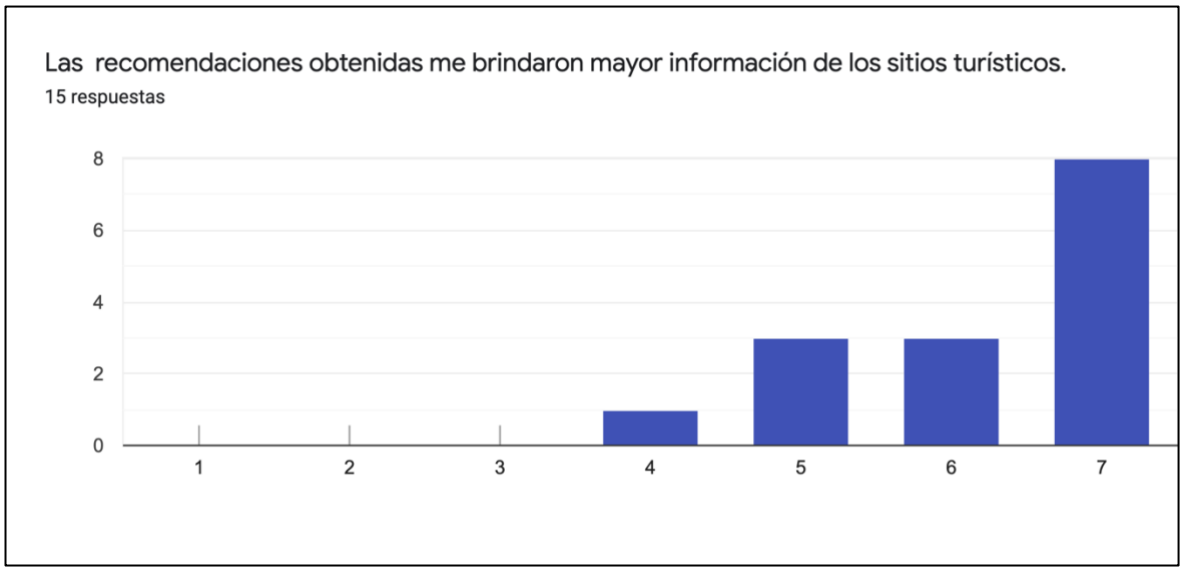


Figura 42. Resultados de la pregunta 3 para la utilidad percibida

Fuente: elaboración propia

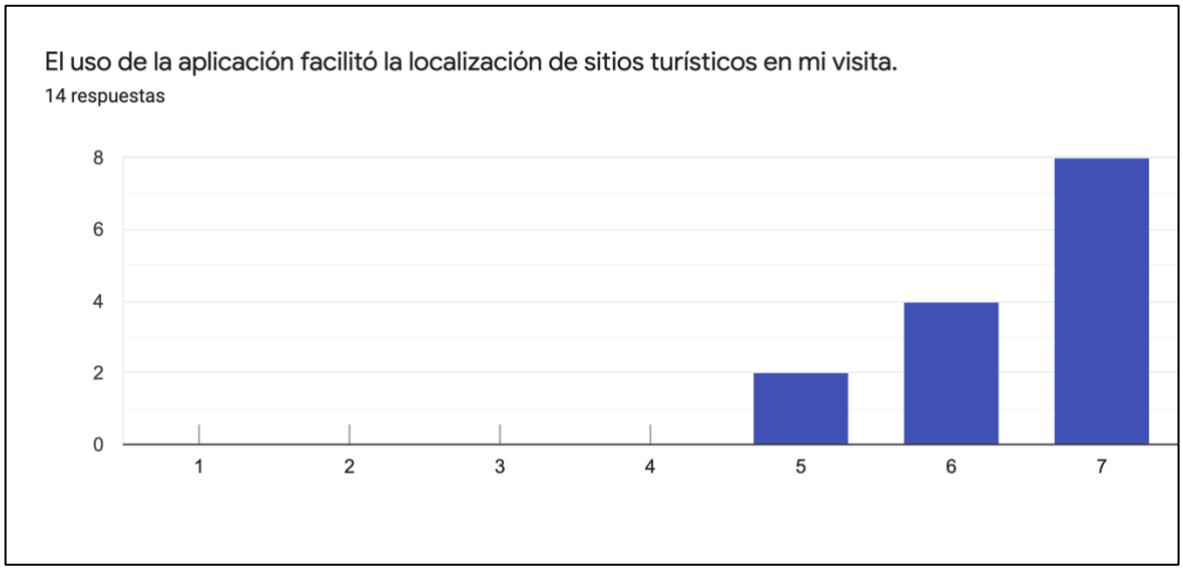


Figura 43. Resultados de la pregunta 4 para la utilidad percibida

Fuente: elaboración propia

Figura 44. Resultados de la pregunta 5 para la utilidad percibida

Fuente: elaboración propia

7.6.2 Facilidad de uso percibida

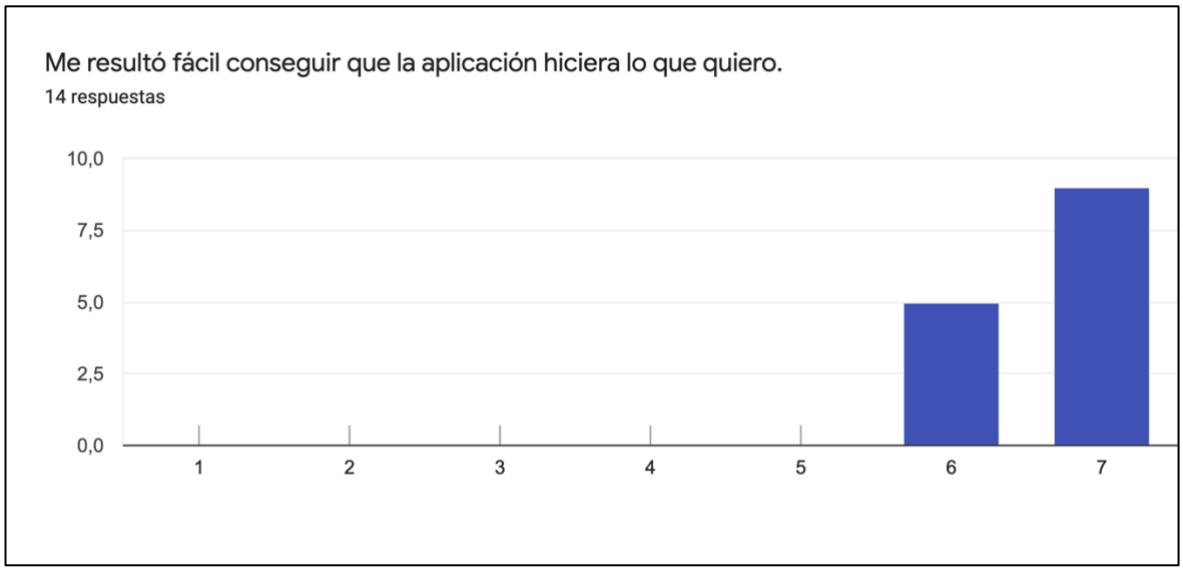


Figura 45. Resultados de la pregunta 1 para la facilidad de uso percibida

Fuente: elaboración propia

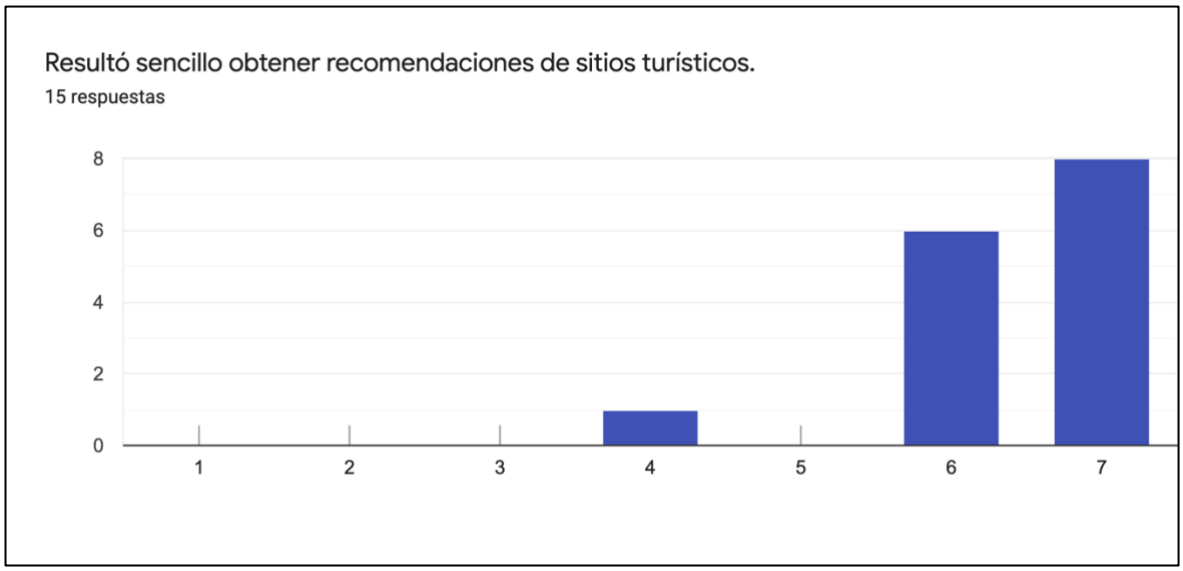


Figura 46. Resultados de la pregunta 2 para la facilidad de uso percibida

Fuente: elaboración propia

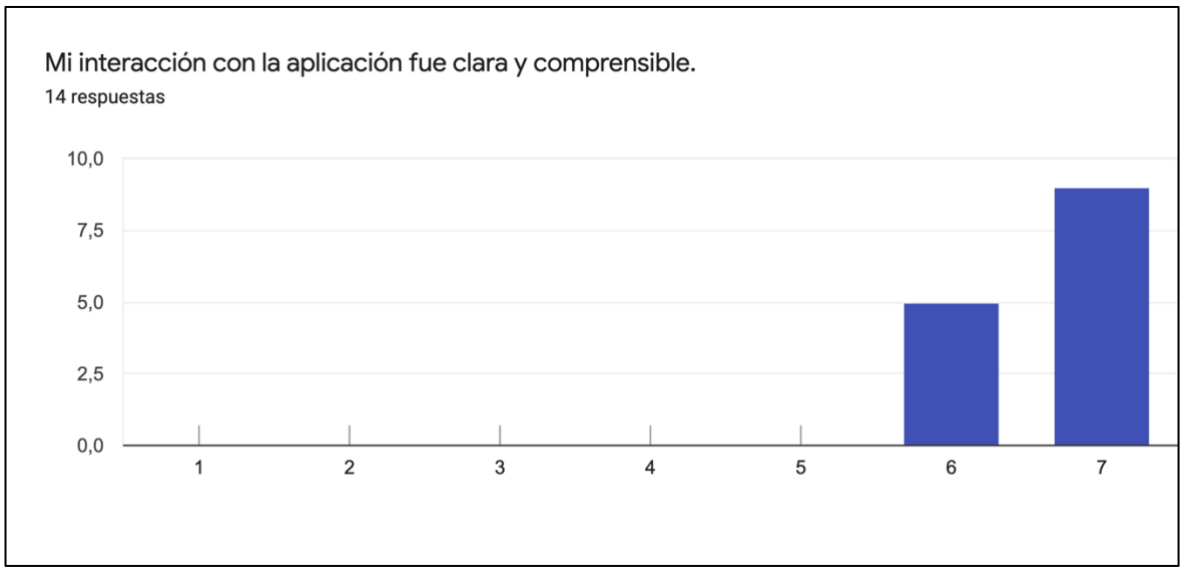


Figura 47. Resultados de la pregunta 3 para la facilidad de uso percibida

Fuente: elaboración propia

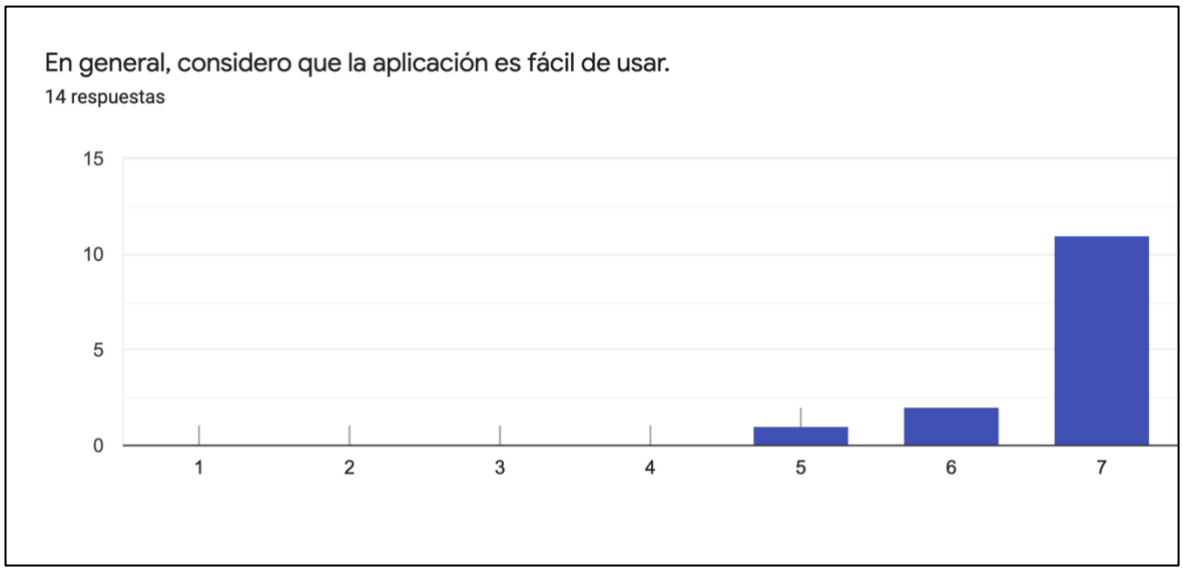


Figura 48. Resultados de la pregunta 4 para la facilidad de uso percibida

Fuente: elaboración propia

7.6.3 Actitud con respecto al uso

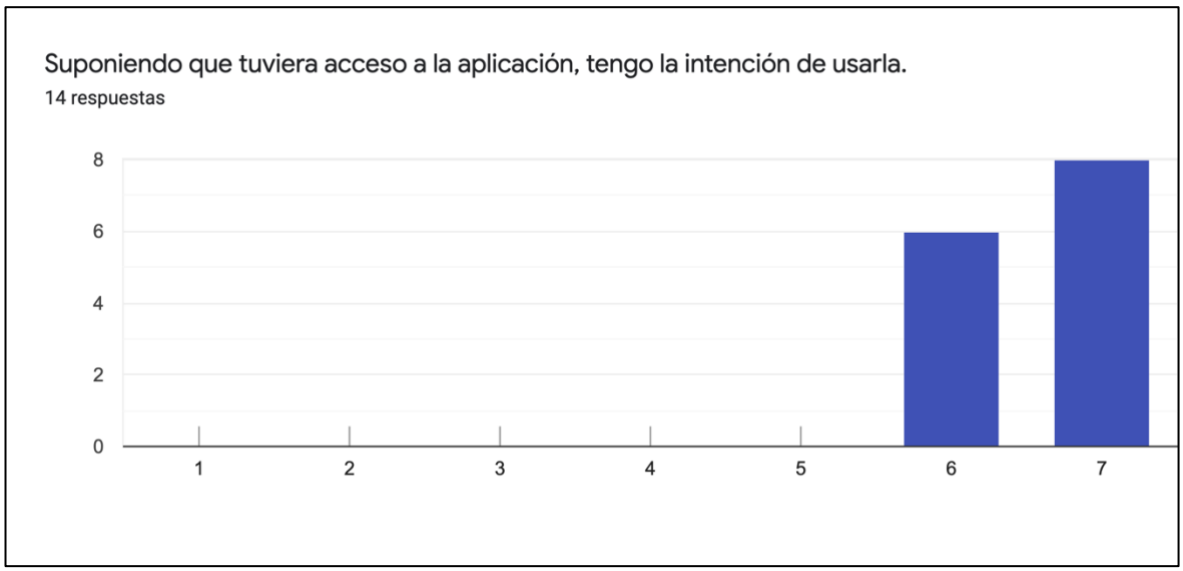


Figura 49. Resultados de la pregunta 1 para la actitud del usuario con respecto al uso

Fuente: elaboración propia

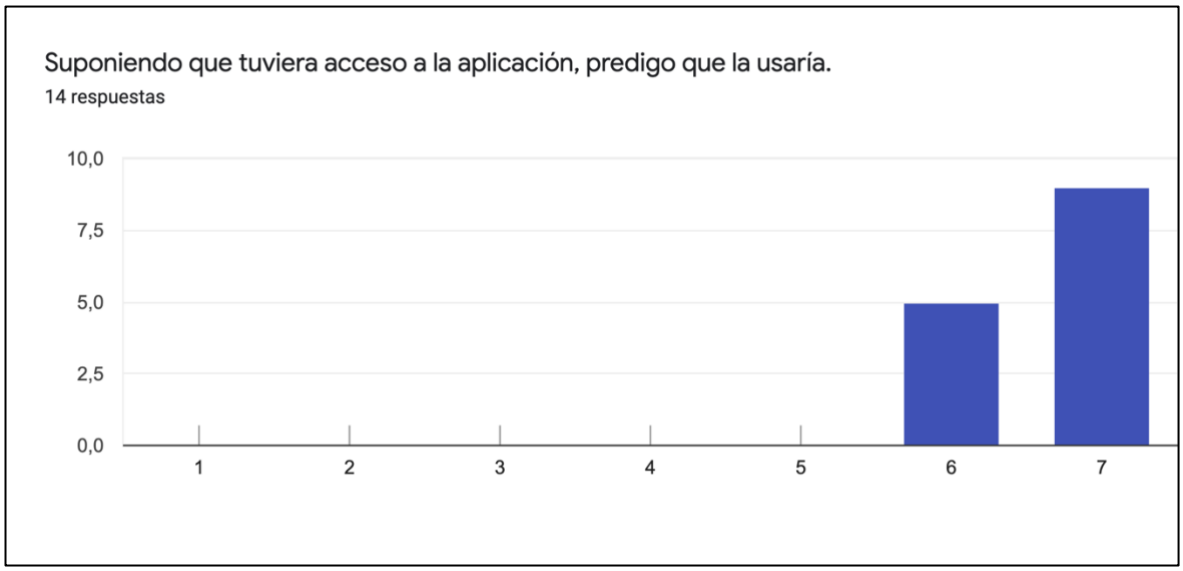


Figura 50. Resultados de la pregunta 2 para la actitud con respecto al uso

Fuente: elaboración propia

7.6.4 Adaptabilidad percibida

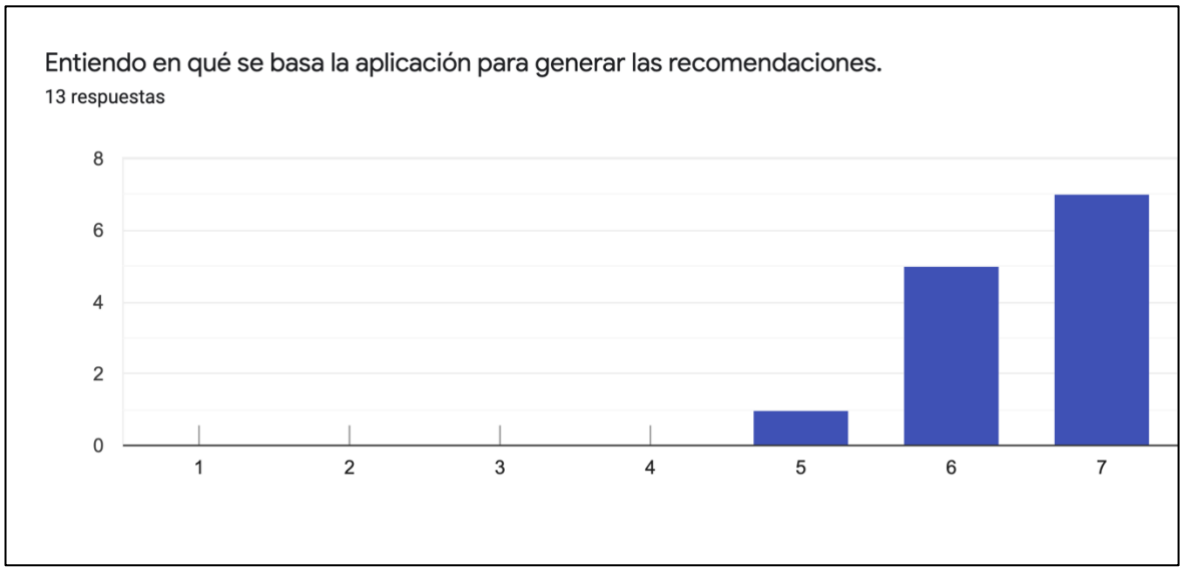


Figura 51. Resultados de la pregunta 1 para la adaptabilidad percibida

Fuente: elaboración propia

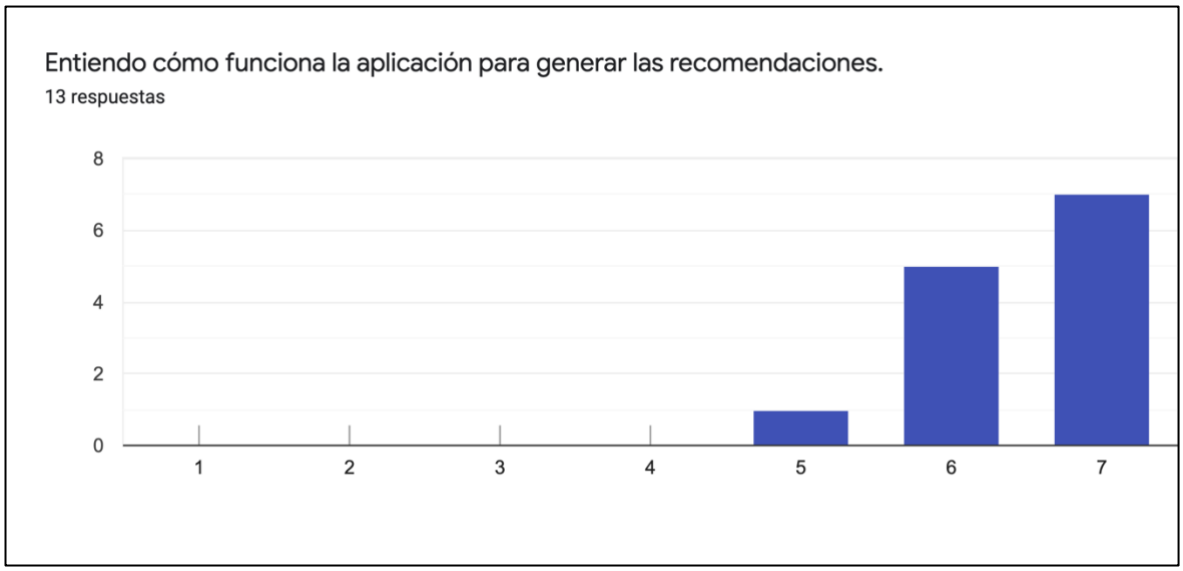


Figura 52. Resultados de la pregunta 2 para la adaptabilidad percibida

Fuente: elaboración propia