UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Desarrollo de una aplicación móvil bajo NativeScript para la recomendación de recursos turísticos en base al algoritmo de K-NN

Tesis para obtener el Título Profesional de Ingeniero de Sistemas

Por:

Félix German Castro Vilchez

Asesor:

Mg. Abel Angel Sullon Macalupu

CoAsesor:

Dra. Nelida Gladys Maquera Sosa

Juliaca, junio del 2019

DECLARACIÓN JURADA DE AUTORÍA DEL INFORME DE TESIS

Mg. Abel Angel Sullon Macalupu, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente informe de investigación titulado: **"Desarrollo de una aplicación móvil bajo NativeScript para la recomendación de recursos turísticos en base al algoritmo de K-NN"** constituye la memoria que presenta el Bachiller **Félix German Castro Vilchez** para obtener el título de Profesional de Ingeniero de Sistemas, cuya tesis ha sido realizada en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Juliaca, a los 11 días del mes de octubre del año 2021

Mg. Abel Angel Sullon Macalupu
Asesor

ACTA DE SUSTENTACIÓN DE TESIS **030**

En Puno, Juliaca, Villa Chullunquiani, a...dos.....día(s) del mes de...julio.....del año 20..... siendo las.....horas, se reunieron en el Salón de Grados y Títulos de la Universidad Peruana Unión, Filial Juliaca, bajo la dirección del Señor Presidente del jurado: ...Mg. Lennin Henry Centurion..... secretario: Dr. Jorge Alejandro Sanchez Garces..... y los demás miembros: Ing. David Mamani Pari.....y el asesor...Mg. Abel Angel Sullon Macalupu..... con el propósito de administrar el acto académico de sustentación de la tesis titulada: Desarrollo de una aplicación móvil bajo NativeScript para la recomendación de recursos turísticos en base al algoritmo de K-NN.....de el(los)/la(las) bachiller/es: a) Felix German Castro Vilchez.....b).....conducente a la obtención del título profesional de .....Ingeniero de Sistemas.....

(Nombre del Título Profesional)

con mención en.....

El Presidente inició el acto académico de sustentación invitando al (los)/a(la)(las) candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por el(los)/la(las) candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Candidato (a): Felix German Castro Vilchez

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	16	B	Bueno	Muy bueno

Candidato (b): .....

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al(los)/a(la)(las) candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente

Secretario

Asesor

Miembro

Miembro

Candidato/a (a)

Candidato/a (b)

DEDICATORIA

El presente trabajo de investigación va dedicado a Dios, quien me guio y estuvo presente en la trayectoria de mi vida, cuidándome, bendiciéndome y dándome fuerzas para cumplir mis metas trazadas. A mis padres que, con apoyo incondicional, amor y confianza permitieron que logre culminar la carrera.

AGRADECIMIENTOS

En primer lugar, mis agradecimientos hacia Dios, por darme la vida y permitirme estudiar en una Universidad Adventista, y gracias al estado peruano por darme la oportunidad de ser parte de Beca 18.

Gracias a la Dra. Nelida Gladys Maquera Sosa por darme la oportunidad de ser parte del proyecto de investigación “Plataforma digital inteligente y Big Data para el turismo rural comunitario en la Región Puno”, de donde surgió esta investigación.

Gracias al Mg. Abel Angel Sullon Macalupu por compartir su experiencia profesional, y por acompañarme en esta investigación en calidad de Asesor.

Gracias a dos grandes amigos y compañeros de clase, Neftalí Ccana y Yuselenin Anquise, por compartir conmigo momentos de estrés, presión, trasnochadas y al final del día alegrías durante el proceso de aprendizaje que nos tocó vivir.

Y gracias a una amiga muy especial a quien estimo tanto Mirian Calcina, por su amistad, ayuda incondicional y apoyo moral que lo necesite en momentos de desánimo.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

SÍMBOLOS USADOS

KNN: Abreviatura en inglés de k vecinos más cercanos

IA: Inteligencia artificial

MICETUR: Abreviatura de El Ministerio de Comercio Exterior y Turismo

SEGITTUR: Abreviatura de Sociedad Mercantil Estatal para la Gestión de la Innovación y las Tecnologías Turísticas.

FONDECYT: Abreviatura de Fondo Nacional de Desarrollo Científico y Tecnológico.

CONCYTEC: Abreviatura de Concejo Nacional de Ciencia, Tecnología e Innovación Tecnológica.

CRISP-DM: Abreviatura en inglés de Proceso estándar de la industria para la minería de datos

UI: Abreviatura inglés de Interfaz de usuario.

UX: Abreviatura en inglés de Experiencia de Usuario.

JSON: Abreviatura en inglés de Notación de Objetos de JavaScript.

BSON: Representa documentos JSON en formato Binario.

XML: Abreviatura en inglés de Lenguaje de marcado extensible.

CSS: Abreviatura en in inglés de hojas en estilo en cascada.

HTML: Abreviatura en inglés de lenguaje de marcas de hipertexto

SQL: Abreviatura en inglés de lenguaje de consulta estructurada.

NO SQL: Hace referencia al lenguaje de consulta no estructurada.

ISO: Abreviatura en inglés de Organización Internacional de Normalización.

WIP: Abreviatura en inglés de Trabajo en progreso.

OEM: Abreviatura en inglés de fabricante de equipo original.

JIT: Abreviatura en inglés de justo a tiempo.

AOT: Abreviatura en inglés de compilación anticipada.

CLI: Abreviatura en inglés de interfaz de línea de comandados.

PWA: Abreviatura en inglés de Aplicación Web Progresiva.

SDK: Abreviatura en inglés de Kit de desarrollo de Software.

UML: Abreviatura en inglés de Lenguaje Unificado de Modelado

TIC: Abreviatura en inglés de Tecnologías de información y comunicación

OMT: Abreviatura de Organización Mundial del Turismo

CB: Hace referencia al filtrado basado en contenido en los sistemas de recomendación.

CF: Hace referencia al filtrado basado colaborativo en los sistemas de recomendación

DF: Hace referencia al filtrado basado demográfico en los sistemas de recomendación

RESUMEN

En la gran cantidad de información disponible en internet, al momento de ser analizada y ver todas las posibles alternativas y elegir puntos de intereses que se ajusten al perfil, a las preferencias y a los recursos disponibles de un turista, puede resultar una tarea tediosa, aburrida e inversión de mucho tiempo al no contar con una herramienta interactiva que facilite dicha tarea. Motivo por el cual se planteó desarrollar una aplicación móvil para la recomendación de recursos turísticos en base al algoritmo de K-NN. Para la investigación se optó por una metodología ágil; Kanban, para la gestión exclusiva de las tareas para el cumplimiento de los objetivos, apoyada a la ISO 12207, ciclo de vida del desarrollo de software. Los datos de los recursos turísticos fueron extraídos de la plataforma de MINCETUR empleando la técnica web Scraping, exclusivamente con el fin para probar los algoritmos con datos reales. Todo este proceso fue realizado bajo el método de CRISP-DM; método para investigaciones de minería de datos. El algoritmo fue programado en NodeJS, posteriormente fue integrado al servidor de la aplicación hecho en LoopBack 4; framework de NodeJS para crear aplicaciones de API REST, justamente para consumirlas desde una aplicación móvil o cualquier otro sistema a nivel cliente. El desarrollo móvil se hizo desde un enfoque multiplataforma, esto por las ventajas que ofrecen tanto a los clientes y desarrolladores. Como resultado final se tuvo la aplicación móvil funcional que realiza recomendaciones de los recursos turísticos según a las calificaciones positivas y a los datos demográficos de los usuarios.

Palabras claves: KNN, Sistemas de recomendación, Kanban, turismo, desarrollo móvil, NativeScript, Flutter, NodeJS, LoopBack.

ABSTRACT

In the large amount of information available on the internet, at the time of being analyzed and see all possible alternatives and choose points of interest that fit the profile, preferences and available resources of a tourist, can be a tedious, boring and investment of a long time to not have an interactive tool to facilitate this task. Reason for which it was proposed to develop a mobile application for the recommendation of tourist resources based on the K-NN algorithm. For the research, an agile methodology was chosen; Kanban, for the exclusive management of the tasks for the fulfillment of the objectives, supported to the ISO 12207, life cycle of the software development. The data of the tourist resources were extracted from the MINCETUR platform using the Scraping web technique, exclusively for the purpose of testing the algorithms with real data. All this process was carried out under the CRISP-DM method; method for data mining investigations. The algorithm was programmed in NodeJS, later it was integrated to the server of the application made in LoopBack 4; NodeJS framework to create REST API applications, just to consume them from a mobile application or any other system at the client level. The mobile development was made from a multiplatform approach, this because of the advantages offered to both customers and developers. The final result was the functional mobile application that makes recommendations of the tourist resources according to the positive ratings and the demographic data of the users.

Keywords: KNN, Recommendation Systems, Kanban, tourism, mobile development, NativeScript, Flutter, NodeJS, LoopBack 4.

CAPÍTULO I. El problema

1.1. Identificación del problema

Según Estébanez (2013) y López de Ávila et al. (2015), en los últimos años hubo un cambio significativo en el comportamiento de los turistas, prefieren opciones más personalizadas, buscan destinos turísticos que se adapten a sus intereses y organizan sus viajes por cuenta propia.

El perfil del turista extranjero, según PromPerú (2018), el 69% de los visitantes señalan a Internet como el medio más influyente en la elección de un destino. Además, el 75% de los turistas organizan su viaje por cuenta propia (PromPerú, 2017). Al igual que los turistas extranjeros, los turistas nacionales también muestran altos porcentajes en el uso del Internet, donde el 82% tienen teléfonos inteligentes con acceso a Internet (PromPerú, 2018), y son usados para buscar información turística durante el viaje y compartir las experiencias vividas mediante las redes sociales (PromPerú, 2017).

En la gran cantidad de información disponible en internet, al momento de ser analizada y ver todas las posibles alternativas y elegir puntos de intereses que se ajusten al perfil, a las preferencias y a los recursos disponibles de un turista, puede resultar tedioso, aburrido e inversión de mucho tiempo al no contar con una herramienta interactiva que facilité dicha tarea.

Internet y la información disponible en ella, facilitan en gran manera la planificación de un viaje. Sin embargo en la gran cantidad de información, la selección de puntos de intereses que se ajusten al perfil, a las preferencias y a los recursos disponibles de un turista, puede resultar una tarea tediosa, aburrida e inversión de mucho tiempo al no contar con una herramienta interactiva que facilite dicha tarea (Kbaier, Masri, y Krichen, 2017, p. 244).

Existen una variedad de plataformas web y aplicaciones móviles, que de alguna manera optimizan y agilizan la planificación de un viaje. Se tiene aplicaciones que ayudan en el alojamiento, transporte y en algunos casos en la alimentación. Mientras que las rutas a los

recursos o atractivos turísticos están definidas de manera estática; carecen de interactividad y personalización (Sp, 2013, pp. 1-2).

Según SEGITTUR y Cámara de Comercio de España (2018), en su guía de aplicaciones móviles para el turismo tiene registrado 708 aplicativos para las siguientes distintas tareas:

Aplicaciones para buscar y reservar transporte, alojamiento o un lugar donde comer, aplicaciones para descubrir un destino, aplicaciones ligadas a la naturaleza, a la cultura, a la gastronomía, aplicaciones que ayudan a elegir compañeros de viaje, a saber, cómo están las olas para surfear o el viento para navegar, a traducir carteles, a obtener descuentos de entradas, etc.

Sin embargo, no existen aplicaciones para hacer un recorrido a partir de puntos de intereses que se ajusten al perfil, a las preferencias y a los recursos de los visitantes. El directorio de aplicaciones de PromPerú (2018), tampoco cuenta con aplicaciones móviles con características de interactividad y personalización.

1.2. Justificación

1.2.1. Social

La aplicación que se desea implementar, en primera instancia será de gran ayuda para los turistas; sean nacionales o extranjeros. Permitirá hacer recorridos en base a recomendaciones o sugerencias a partir del perfil, gustos, preferencias y recursos disponibles del turista. La aplicación agilizará la planificación, antes, durante y después del viaje. Además, siendo el principal contenido de la aplicación los recursos turísticos o emprendimientos, tienen más posibilidad de promocionarse, lo que hará posible un incremento de número de visitantes a los recursos, incluso a los lugares más desconocidos y menos visitadas.

1.2.2. Económico

El recurso más importante de una entidad son los datos, que a partir de ella se puede generar nuevos conocimientos, nuevos modelos de negocio, en busca de la competitividad y sostenibilidad de la empresa. La aplicación será implementada con ese enfoque, en las tendencias tecnológicas, como es el la Ciencia de Datos y el Big Data. Las empresas

necesitan apoyarse a este enfoque para generar competitividad y sostenibilidad económica del país.

Según PromPerú (2018), el turismo en el Perú representa alrededor del 4% del Producto Bruto Interno (PBI), lo que significa, a más turistas en un recurso turístico, muy aparte de generar más ingresos para el emprendedor o dueño del recurso y en efecto tener una mejor calidad de vida, también se tendrá un incremento de porcentaje del PBI.

Para ofrecer información actualizada en una aplicación móvil o cualquier otro medio de información, se necesita personal, muchas veces personal capacitado, y eso, solo a inicios de la aplicación. Si se quiere incorporar nuevas funcionalidades, o también conocido en el desarrollo de software, como el mantenimiento de la aplicación, es necesario expertos del área; la aplicación generará más empleo.

En el mundo de las aplicaciones móviles, existen muchas formas de generar ingresos; incorporando anuncios, limitando funcionalidades a cambio de pagos mensuales o anuales. Considerar una de las formas mencionadas en el sector de turismo, y a inicios de la aplicación, no es una buena idea; la publicidad es muy molestosa, y pagarlo generaría pérdida de la fidelidad del usuario. Lo recomendable es generar ingresos por los recursos turísticos o emprendimientos que están en la aplicación, pero cuando la aplicación esté robusto y dando buenos resultados. O mejor aún, iniciar un emprendimiento para dar continuidad y bridar servicio exclusivo a los turistas.

1.2.3. Ambiental

Teniendo en cuenta más de 4000 recursos turísticos que se tiene en el Perú. Las oficinas o agencias que ofrecen estos recursos, ¿Cuantos afiches, guías, anuncios, etc. piden a imprimir por día? Además de eso, al estar ya en las manos de los visitantes, muchas veces son abandonados al paso, sin considerar las consecuencias al medio ambiente, y muchas veces son hechos por las personas locales, dando una mala imagen a los visitantes extranjeros.

La digitalización de la información ayuda a la fomentación de: “menos papeles, más árboles”. Esta investigación es un claro de ejemplo de cuidar el medio ambiente, ya que tiene la finalidad de eliminar toda información que se ofrece por medios físicos.

1.2.4. Legal

Esta investigación se ajusta a la Ley Nº 30309, Ley que promueve la investigación científica, desarrollo tecnológico e innovación tecnológica.

1.2.5. Tecnológico

La manera de ofrecer recursos turísticos, que se ajusten al perfil del turista, es mediante los sistemas de recomendación, que son muy usados en otros campos, como, por ejemplo: en el comercio electrónico y las páginas de entretenimiento o venta de películas; caso Netflix (Kbaier et al., 2017) y (Isinkaye, Folajimi, y Ojokoh, 2015).

La creciente demanda de los dispositivos móviles y aprovechando la ventaja de su portabilidad, es viable implementar una solución para los turistas que viajan a Perú. Sin embargo, la diversidad de los dispositivos y plataformas existentes en los teléfonos inteligentes, trae muchos desafíos en el proceso de desarrollo (Mehlhorn, 2017).

Las empresas requieren tener presencia de sus productos y servicios en la mayor cantidad de plataformas posibles con el objetivo de llegar a más cantidad de usuarios potenciales (Nitze y Schmietendorf, 2013). Para contemplar esta necesidad, desde un enfoque nativo, un desarrollador debe aprender muchos lenguajes de programación, diseñar interfaces de usuario para cada plataforma, y los costos de implementación y mantenimiento serían muy altas (Charkaoui, Adraoui, y Benlahmar, 2014). La solución a este problema son los marcos de desarrollo multiplataforma, que permiten desarrollar una sola aplicación para varios sistemas operativos móviles (Nitze y Schmietendorf, 2013), (Mehlhorn, 2017) y (Charkaoui, Adraoui, y Benlahmar, 2014).

Los sistemas de recomendación, son considerados inteligentes, por el hecho de tener algoritmos de aprendizaje automático en su funcionamiento. Y teniendo en cuenta el uso de los celulares y la planificación de un viaje por cuenta propia de los turistas, en este proyecto

se plantea desarrollar, desde un enfoque multiplataforma, una aplicación móvil para la recomendación de recursos turísticos a partir del perfil, de las preferencias y de los recursos (tiempo y dinero) disponibles del turista en base al algoritmo de K-NN.

El desarrollo de la aplicación móvil será con el enfoque de multiplataforma compilado y bajo NativeScript, donde el desarrollo es más sencillo, y menos costosa la implementación y el mantenimiento. El producto final será una aplicación con las mismas características que se puede lograr con el desarrollo nativo.

1.3. Presunción filosófica

El presente trabajo se realiza teniendo en mente el libro de Hebreos 13:16 (RVR1960), que dice: “Y de hacer bien y de la ayuda mutua no os olvidéis; porque de tales sacrificios se agrada Dios”. Este versículo nos insta al servicio a los demás, en apoyar a los emprendedores o dueños de los recursos, que muy pocas veces son visitadas de manera constante.

1.4. Objetivos de la investigación

1.4.1. Objetivo general

Desarrollar una aplicación móvil bajo NativeScript para la recomendación de recursos turísticos en base al algoritmo de K-NN

1.4.2. Objetivos específicos

- Analizar los requerimientos funcionales para el desarrollo de la aplicación móvil de recomendación de recursos turísticos.
- Integrar el algoritmo k vecinos más cercanos en los diferentes enfoques de sistemas de recomendación en la aplicación móvil.
- Realizar pruebas de la aplicación móvil.

CAPÍTULO II. Revisión de Literatura

2.1. Antecedentes de la investigación

Investigaciones acerca de sistemas en el sector de turismo

En la investigación de (SILVIA, 2013) titulado "Generación de rutas. comparativa de sistemas de destinos turísticos. características necesarias", tuvo como objetivo obtener las características deseables que tendría que tener una plataforma web o aplicación turística para personalizar una ruta. Se realizó un estudio comparativo de las técnicas de generación de rutas existentes y de algunos proyectos realizados para comprobar si las herramientas y elementos que poseen son los adecuados para ayudar al turista a diseñar un viaje personalizado.

Se realizó una encuesta, a 100 jóvenes universitarios de 20 a 30 años de edad, para obtener información y conocer las opiniones, necesidades y usos sobre los sistemas de creación de rutas turísticas personalizadas.

Tras el estudio de las distintas aplicaciones y modelos existentes, y considerando los resultados obtenidos de la encuesta, los proyectos presentados sobre este tipo de sistemas de ayuda al turista, en la mayoría de los destinos no se aplican.

Además, los sistemas que actualmente están ofreciendo sus servicios no cuentan con todos los elementos necesarios para poder crear una ruta personalizada.

Las características deseables que se encontraron para facilitar y mejorar la satisfacción del turista en la visita a un destino fueron: cubrir las tres etapas de un viaje(antes, durante y después), accesible desde el dispositivo móvil, alta personalización, automaticidad de la ruta, localización, adaptabilidad, disponible en distintos idiomas, funcionamiento sin conexión a internet, orientado a la ruta, ofrecer experiencia de otros usuarios, posibilidad de compartir por redes sociales, posibilidad de almacenar las preferencias y reducir tiempo de búsqueda.

Otra investigación realizada por (Kbaier et al., 2017) titulado: "A Personalized Hybrid Tourism Recommender System", tuvo como objetivo desarrollar un sistema de recomendación híbrida en el campo del turismo, combinando tres métodos más populares

en los sistemas de recomendación, como son: el filtrado colaborativo(CF), filtrado basado en contenido(CB) y el filtrado demográfico(DF).

Las técnicas que se usaron para la implementación fueron, el vecino más cercano(K-NN) tanto para CB como CF, y el árbol de decisiones para DF. Para mejorar la precisión de la recomendación, se usaron dos técnicas de hibridación: conmutación y ponderación. Los datos de experimentación se obtuvieron de TripAdvisor, que fueron pre procesados y limpiados para aplicar las diferentes técnicas.

Se obtuvo una aplicación que recomienda a un turista las mejores atracciones, a partir de la unión de los tres enfoques de sistemas de recomendación sujeta a restricciones de cada método. Los resultados experimentales mostraron claramente que el método híbrido proporciona una predicción más precisa que los otros métodos utilizados por separado.

Investigaciones acerca de aplicaciones móviles con enfoque multiplataforma

En la investigación de (Warén, 2016), titulado "Cross-platform mobile software development with React Native", tuvo como objetivo crear una aplicación móvil totalmente funcional con React Native. En la parte de back-end se usó Ruby on Rails. Sin embargo, la tesis se centra en el front-end y la API se tiene funcionando a la perfección para el consumo de los datos.

La aplicación consiste en un proveedor de empleados, donde las empresas pueden solicitar empleados a otras empresas; préstamo de empleados por tiempo predeterminado. Como en todo proceso de desarrollo de software, primero se define los requerimientos, diseño de la base de datos, diseño de las interfaces de usuario y la implementación.

La aplicación fue implementada en cierta parte, y el autor, concluye diciendo: "es una buena opción para la creación de aplicaciones móviles y especialmente adecuado para los programadores con experiencia en el desarrollo web".

En la investigación de (Olusola Olajide, Omotayo Abiodun, Adebola Okunola, Gabriel Omomule, & Michael Orimoloye, 2018) titulado "Performance Evaluation of Native and Hybrid Android Applications, tuvo como objetivo evaluar el rendimiento de las aplicaciones

móviles nativas e híbridas en Android y explorar la limitación de los marcos de desarrollo para dispositivos móviles.

En un principio, se describen varios casos de prueba para evaluar el rendimiento de las aplicaciones móviles con las que se crearon dos aplicaciones para implementar un cálculo matemático para ambos, nativo e híbrido, respectivamente. Esto se utiliza como punto de referencia debido a la naturaleza recursiva y el uso de la memoria de ambas aplicaciones para el uso de la CPU y la memoria.

La eficacia del rendimiento de las aplicaciones nativas e híbridas se compara en un dispositivo móvil.

La investigación se realizó seleccionando el SUT y el CUS (Sistema bajo estudio y Componente en estudio).

Se realizó un cálculo y se registró el tiempo tomado.

Los siguientes aspectos se utilizaron para la evaluación del rendimiento de la aplicación: tiempo de CPU, espacio de memoria, uso de batería, demanda de comunicación y el tiempo total de ejecución de la aplicación para mostrar páginas web (híbrido).

Los autores concluyen que las aplicaciones híbridas aún sufren problemas de rendimiento en comparación con las aplicaciones nativas, ya que, en todas las pruebas realizadas, las aplicaciones nativas fueron superiores a las aplicaciones híbridas.

2.2. Marco teórico

2.2.1. Turismo

Según López de Ávila et al. (2015, p. 14) una de las actividades económicas que lidera el comercio electrónico a nivel mundial es el turismo, y afirma que: “es un hecho que las tecnologías de la información y la comunicación (TIC), principalmente el acceso casi universal a internet, han significado una innovación disruptiva en las relaciones entre oferta y demanda”.

El entorno digital permite promocionar destinos, vender productos y servicios y, a la vez, conocer a los clientes. La atención a ellos es ahora más personalizada y de mayor calidad. Además, acceder a internet ahora es más fácil e inmediato desde la aparición de los teléfonos inteligentes, y son para gestionar información en tiempo real. A partir de este nuevo escenario, como afirma López de Ávila et al. (2015, p. 15): "el turista tiene en sus manos, al instante, toda la información acerca de productos, servicios, rutas, horarios, precios, disponibilidad, etc. Al mismo tiempo, aparecen destinos hasta la fecha desconocidos, con la consiguiente revitalización turística de zonas menos demandadas".

2.2.1.1. Turismo en el Perú

Según la Organización Mundial del Turismo (2016): "La economía del Perú depende principalmente de la agricultura, la industria minera, los productos primarios y del sector terciario, en el que se incluye el turismo" y según PromPerú (2018), el turismo representa alrededor del 4% del PBI del país, generado como menciona DGIETA y Mincetur (2016): "por actividades económicas como el transporte de pasajeros, provisión de alimentos y bebidas, alojamiento, industria cultural, recreativa y deportiva, agencias de viajes, producción y comercio de artesanía, entre otros".

El perfil del turista extranjero, según PromPerú (2018), el 69% de los visitantes señalan a Internet como el medio más influyente en la elección de un destino. Además, el 75% de los turistas organizan su viaje por cuenta propia (PromPerú, 2017). Al igual que los turistas extranjeros, los turistas nacionales también muestran altos porcentajes en el uso del Internet, donde el 82% tienen teléfonos inteligentes con acceso a Internet (PromPerú, 2018), y son usados para buscar información turística durante el viaje y compartir las experiencias vividas mediante las redes sociales (PromPerú, 2017).

2.2.2. Aprendizaje automático

El aprendizaje automático según Dasgupta (2018, p. 214) no es más que predecir eventos futuros basados en datos históricos. En sus dos ramas principales que son: aprendizaje supervisado y aprendizaje no supervisado, la diferencia está en la estructura de los datos y en la tarea del algoritmo (Dangeti, 2017, p. 359) y (Dasgupta, 2018, p. 220). Para el

aprendizaje automático supervisado, el conjunto de datos de entrenamiento está etiquetada por clases, y el trabajo del algoritmo es determinar la respuesta correcta en función de las características similares que una clase pueda tener. Mientras que, para el aprendizaje automático no supervisado, el conjunto de datos no está etiquetada, y el trabajo del algoritmo es agrupar los datos en función de características similares de los datos.

2.2.2.1. K vecinos más cercanos (KNN)

Es un algoritmo de clasificación, perteneciente al aprendizaje automático supervisado (Kanber, 2018). Se le conoce por ser un algoritmo simple, pero a la vez perezoso por tener un aprendizaje en memoria, es más, no tiene la fase de entrenamiento, ya que su funcionamiento comienza durante la fase de pruebas, por lo tanto a más datos a ser operado, más es el tiempo de respuesta, y esto hace que el algoritmo k-nn no sea recomendable para trabajar con grandes cantidades de datos (Dangeti, 2017, p. 187).

La constante “k” representa el número de vecinos más cercanos que se quiere considerar al clasificar el nuevo punto.

1. Funcionamiento

Como lo describe Kanber (2018) y Natingga (2017), el proceso que sigue el algoritmo k vecinos más cercano para llegar a su objetivo es el siguiente:

Para poner en ejecución el algoritmo KNN, los datos tienen que estar debidamente etiquetados en clases o categorías, de modo que se pueda saber las posibles salidas. Además, si se requiere, el conjunto de datos tiene que estar dividido en dos subconjuntos, que son para el entrenamiento y las pruebas del algoritmo. Como se mencionó anteriormente, KNN no pasa por la fase de entrenamiento, el subconjunto que se divide para las pruebas, son los datos a evaluar sobre el subconjunto de entrenamiento.

El algoritmo se pone en ejecución cuando un nuevo punto es dado para su evaluación, y cuando esto ocurra, básicamente se genera una lista de la distancia entre el punto entrante a todos los datos del subconjunto de entrenamiento. Para calcular la similitud entre un punto y el otro, existen varias técnicas como la distancia euclidiana, similitud de coseno,

coeficiente de tanimoto, la correlación de Pearson, la correlación de Spearman, etc. (Kbaier et al., 2017).

Una vez se tenga la lista de las distancias, el siguiente paso es ordenarla, de lo más cercano a lo más lejano, y luego mostrar los k vecinos más cercanos del punto de entrada. Finalmente, si fuera necesario determinar qué etiqueta representa la mayoría de los k vecinos más cercanos; este es el resultado del algoritmo.

2.2.3. Sistemas de recomendación

Son sistemas que recomiendan o sugieren un producto o un servicio en particular en función de actividades y/o operaciones anteriores, también conocidos como datos históricos (Banik, 2018, p. 7). La idea de los sistemas de recomendación es solucionar el problema de la abundancia de información con la que se encuentre el usuario (Isinkaye et al., 2015). En el fondo no son más que un filtrado dinámico de información para proporcionar a los usuarios contenido y servicio personalizado basados en técnicas o algoritmos de aprendizaje automático.

Según Banik (2018, p. 10) describe que: “en los sistemas de recomendación, como con casi cualquier otro problema de aprendizaje automático, las técnicas y modelos que usa dependen en gran medida de la cantidad y calidad de los datos que posee”. Significa que existen muchos enfoques de sistemas de recomendación, pero cuatro son los más populares, y varían en función de los datos que requieren. Además de los enfoques independientes, existe un llamado “enfoque híbrido”, que básicamente es la unión de dos a más enfoques independientes para cubrir las desventajas de cada enfoque.

2.2.3.1. Filtrado colaborativo

Según Banik (2018) el filtrado colaborativo aprovecha el poder de la comunidad para proporcionar recomendaciones. Esto significa crear una matriz de elementos de las preferencias de los artículos por parte de los usuarios, y sobre esa matriz comparar usuarios con intereses y preferencias relevantes calculando similitudes entre sus perfiles, formando grupos de comunidades. Y las recomendaciones que un usuario recibirá será por aquellos

artículos que no ha calificado anteriormente pero que ya fueron calificados positivamente por los usuarios de su comunidad (Isinkaye et al., 2015).

Para que las recomendaciones mediante este enfoque sean de calidad, y no pierdan la relevancia de su predicción, se necesitan datos históricos del usuario activo además de otros usuarios, lo cual no es posibles cuando el usuario activo es nuevo, y no posee ninguna actividad realizada en el pasado sobre el sistema, a este problema se le conoce como "arranque en frío", y es propio de este enfoque (Isinkaye et al., 2015) y Banik (2018).

2.2.3.2. Filtrado basado en contenido

Este enfoque, a diferencia del filtrado colaborativo no necesita el perfil, las preferencias, ni calificaciones de otros usuarios, se basa sólo en el perfil del usuario activo y en el análisis de los atributos de los elementos evaluados en el pasado para generar las recomendaciones (Isinkaye et al., 2015).

El filtrado basado en contenido supera el problema de "arranque en frío" que presenta el enfoque colaborativo, porque este no necesita datos de otros usuarios. Pero aún sufre un nuevo problema llamado "inicio en frío" del usuario, en el cual el usuario es nuevo y no tiene elementos calificados, y por ende el algoritmo no puede operar sobre ella (Kbaier et al., 2017).

2.2.3.3. Filtrado basado en el conocimiento

El filtrado basado en conocimiento, brinda recomendaciones sujetas a restricciones, y es el recomendador más sencillo de implementar. El flujo que sigue el algoritmo es el siguiente: se solicita cierta información de preferencias al usuario y el sistema proporciona recomendaciones que satisfacen esas condiciones mencionadas anteriormente (Banik, 2018).

2.2.3.4. Filtrado demográfico

Para este enfoque es necesario los datos demográficos del usuario, como pueden ser la edad, género, profesión, lugar de origen, etc. También es necesario los datos demográficos de otros usuarios y sus respectivas preferencias. Una vez se tenga los datos, el proceso que

sigue es el siguiente: se realiza el cálculo de similitud utilizando los datos demográficos de los usuarios para obtener una cantidad de usuarios con datos demográficos similares al usuario activo. Finalmente, se obtiene recomendaciones de los elementos que han sido valorados positivamente por los usuarios similares al usuario activo (Safoury y Salah, 2013).

En este enfoque aún persiste el problema de arranque en frío, cuando no se tiene las preferencias de otros usuarios. También está el problema de falta de información demográfica de los usuarios, ya que este enfoque funciona mejor mientras haya más datos demográficos, y es muy difícil conseguir tales datos (Safoury y Salah, 2013).

2.2.3.5. Filtrado híbrido

La técnica de filtrado híbrido combina diferentes técnicas de recomendación para obtener recomendaciones más precisas y efectivas anulando las limitaciones y problemas de los sistemas de recomendación que trabajan de manera independiente (Banik, 2018, p. 14). La combinación de enfoques se puede realizar de cualquiera de las siguientes maneras: implementación separada de algoritmos y combinación del resultado, utilizando algunos filtros basados en contenido en enfoque colaborativo o viceversa, de tal manera se tiene un sistema de recomendación unificado (Kbaier et al., 2017).

Existen otras técnicas de hibridación, como la hibridación ponderada y la hibridación conmutada. La hibridación ponderada combina los resultados de diferentes recomendadores para generar una lista de recomendación basados en modelos lineales, mientras que la hibridación conmutada se basa en el cambio entre diferentes resultados de recomendación para aprovechar cada tipo en una situación diferente y para obtener el mejor resultado de calificación (Kbaier et al., 2017).

2.2.4. Desarrollo de software

Es la construcción de aplicaciones para resolver problemas específicos de negocio, en algunos casos, sobre todo en los últimos años, se viene implementando algoritmos matemáticos para modelar, predecir y simular sistemas complejos, como es el caso de los sistemas de recomendación (Rivero Cuesta, s. f., p. 5).

2.2.4.1. Ciclo de vida del desarrollo de software

La ISO, International Organization for Standardization, en su norma 12207 define al ciclo de vida de un software como: "un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso".

El ciclo de vida de un software comprende una serie de etapas, donde se pueden establecer una serie de objetivos, tareas y actividades para completar la fase y pasar a la siguiente. Estas fases son: análisis de requerimientos, diseño, implementación, pruebas y mantenimiento (Berzal, 2004) y (Cantone, 2006).

1. Análisis de requerimientos

El primer paso para la implementación de un software es averiguar qué es lo que realmente tiene que hacer el software. En esta etapa es donde se definen los requerimientos funcionales y no funcionales del sistema. Al final se tiene una descripción clara del producto a desarrollar, se conoce las funcionalidades que aportará, y el comportamiento que tendrá. Esta etapa es muy importante, ya que una funcionalidad mal analizada o que no fue detectada, puede generar cambios en las etapas posteriores, en consecuencia, generaría cambios en los costos y en el tiempo, depende del impacto del cambio en el sistema (Berzal, 2004) y (Cantone, 2006).

Para un buen análisis es bueno identificar a los stakeholders, que realmente conozcan las reglas del negocio, estos pueden ser desde el cliente que paga el producto, el equipo de trabajo, hasta los usuarios finales. Cuando un producto es nuevo, esta etapa es aún más complicada, y se requiere el diseño de un prototipo, una técnica que facilita la recolección de requerimientos, mostrando el aspecto que tendrá el producto final (Berzal, 2004).

2. Diseño

Se sabe lo que tiene que hacerse, en esta etapa se define el cómo y el con qué se va a desarrollarse. En esta etapa es común utilizar el Lenguaje Unificado de Modelado (UML), para limitar, definir la estructura y el comportamiento del sistema (Berzal, 2004). Algunos

de los diagramas más usados son, diagrama de entidad relación, diagrama de clases, diagrama de casos de uso, diagrama de actividades, diagrama de secuencia, diagrama de componentes, etc.

3. Desarrollo

Una vez entendido el que hacer y cómo hacerlo de manera clara y precisa es hora de implementarlo, en el lenguaje y entorno de desarrollo elegido. El desarrollo se realiza de acuerdo a la arquitectura elegida, supongamos una arquitectura de tipo cliente servidor, se empezará programando el backend, donde incluye creación de modelos y las APIs. Y luego el lado del frontend con el desarrollo de las interfaces (Berzal, 2004).

4. Pruebas

El objetivo de las pruebas es encontrar errores para mejorar la calidad del sistema, haciendo más robusto a fallos. Estos errores pueden ser de requerimiento, de diseño o de funcionalidad. Detectar errores en el producto antes del lanzamiento es muy importante, ya que una falla en el sistema mientras que esté en funcionamiento puede traer grandes consecuencias (Berzal, 2004).

Existen dos tipos de pruebas, están las manuales y automáticas, dentro de ellas aún existen niveles, como las siguientes: pruebas unitarias, pruebas de integración, pruebas de validación, pruebas del sistema, y pruebas de aceptación (Berzal, 2004).

5. Mantenimiento

Esta es la etapa más larga del ciclo de vida de un software. La naturaleza de un software es que no se desgasta con el pasar de los años, al contrario, se busca mejorar el sistema. Generalmente el mantenimiento cubre tres etapas: eliminar los defectos que se detectan durante su funcionamiento, añadir nuevas funcionalidades al sistema, y por último adaptarlo a nuevas necesidades o contextos; pasar de un marco monolítico a microservicios por ejemplo (Berzal, 2004).

2.2.5. Desarrollo móvil

Para entrar en contexto es necesario saber que una aplicación móvil es un software que se ejecuta en un dispositivo móvil inteligente, se puede acceder a ella desde una tienda de aplicaciones disponible para cada plataforma o sistema operativo. Algunas aplicaciones son de acceso libre, y algunas de pago. Las aplicaciones móviles en un inicio se ofrecieron para la productividad en general, mostrar información y otras tareas básicas, sin embargo, la demanda y la disponibilidad de herramientas de desarrollo impulsaron su crecimiento acelerado, tal fue la demanda que las compañías empezaron a ofrecer sus productos y servicios por medio de las aplicaciones móviles (D'Angelo y Rodríguez, 2015) y (Jiménez y García, 2015).

Sin embargo, la diversidad de los dispositivos y plataformas existentes en los teléfonos inteligentes, trajo muchos desafíos en el proceso de desarrollo (Mehlhorn, 2017). Las empresas requerían y requieren tener presencia de sus productos y servicios en la mayor cantidad de plataformas posibles con el objetivo de llegar a más cantidad de usuarios (Nitze y Schmietendorf, 2013).

Para cubrir esta necesidad, desde un enfoque nativo, un desarrollador debe aprender muchos lenguajes de programación, diseñar interfaces de usuario para cada plataforma, y los costos de implementación y mantenimiento serían muy altos (Charkaoui et al., 2014). En solución a este este problema aparecieron varios enfoques de desarrollo para la construcción de aplicaciones móviles con el objetivo de tener una aplicación en varios sistemas operativos móviles con una sola base de código (Nitze y Schmietendorf, 2013).

La aparición de los enfoques de desarrollo favoreció de alguna manera a los desarrolladores, ya que la mayoría de los enfoques tratan de llevar el desarrollo web para el desarrollo móvil, significa que un desarrollador web, con las mismas tecnologías, ya puede construir una aplicación móvil, con la ventaja de cubrir varias plataformas móviles con una sola base de código. También los favorecidos son las empresas o usuarios que requieren tener presencia de sus servicios o productos en las plataformas principales evitando gastos muy altos en la implementación y mantenimiento. El tiempo de construcción de una aplicación también disminuye en gran manera, favoreciendo tanto a los desarrolladores como a las empresas.

En los enfoques de desarrollo móvil se tiene: el enfoque nativo, enfoque web y multiplataforma. Algunos de los enfoques tienen sub divisiones como el caso de multiplataforma; de tipo Nativo e Híbrido.

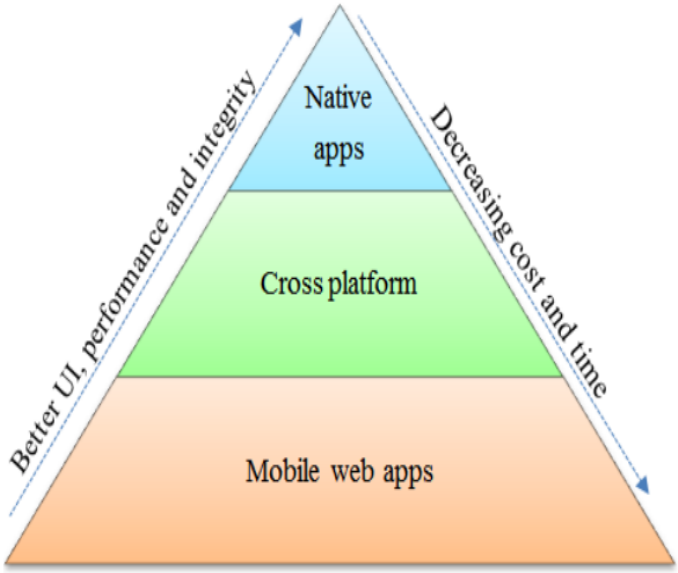


Figura 1. Desarrollo nativo vs. Desarrollo multiplataforma vs. Desarrollo web móvil
Fuente: (Mohamed & Abdali, 2017)

Las diferencias generales entre estos enfoques se muestran en la figura 1, donde más características nativas tenga la aplicación, ésta tendrá una mejor interfaz, mejor rendimiento e integridad y a la vez el tiempo y costo de desarrollo y mantenimiento serán altos. Sin embargo, cuando una aplicación es menos nativa o más aún, sea una aplicación web móvil, es todo lo contrario. Para mantener un equilibrio, se tiene en el término medio al enfoque multiplataforma (Mohamed y Abdali, 2017).

2.2.5.1. Enfoque nativo

Cada plataforma tiene un kit de herramientas para el desarrollo (SDK) que permite la construcción de las aplicaciones (D'Angelo y Rodríguez, 2015). Solo teniendo en cuenta las principales plataformas que existen en el desarrollo móvil como es Android y iOS se ven los grandes desafíos con la que un desarrollador se encuentra. Las aplicaciones de iOS están construidas con Objective-c, y en los últimos años viene sumándose Swift, lenguajes que

únicamente funcionan en un entorno Mac OS. Las aplicaciones para android están codificadas en Java y Kotlin; también impulsado en los últimos años (Nagesh y Caicedo, 2012) y (El-Kassas, Abdullah, Yousef, y Wahba, 2017).

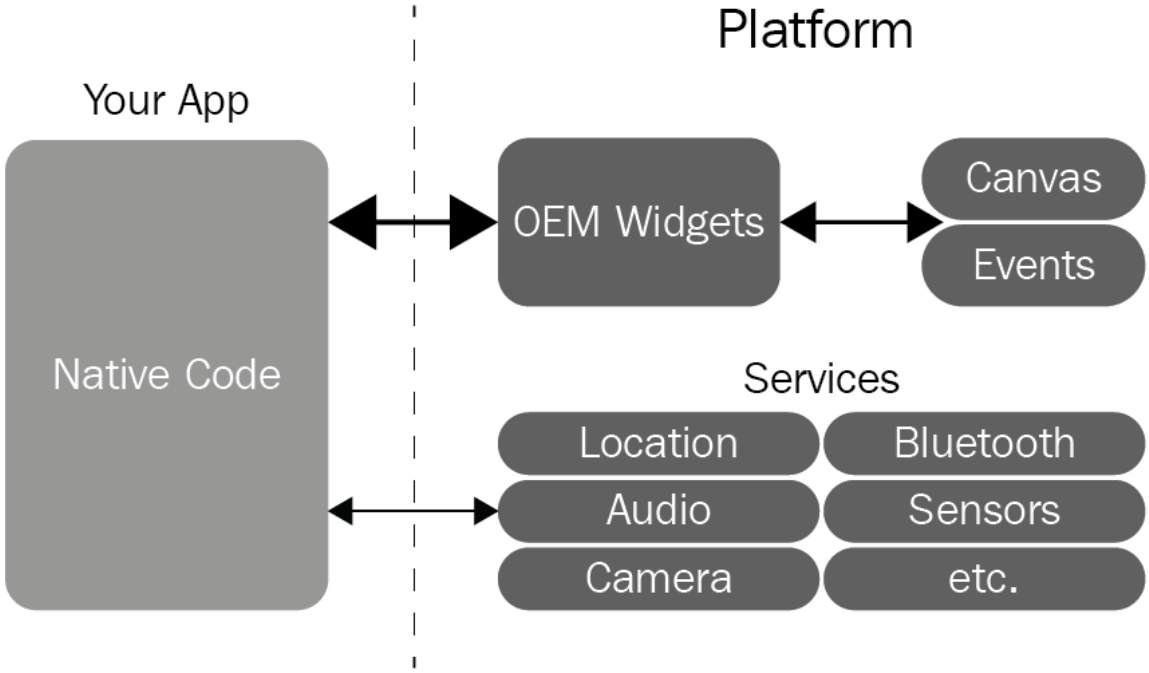


Figura 2. Arquitectura de una aplicación nativa
Fuente: (Mainkar & Giordano, 2019)

Las aplicaciones desarrolladas mediante este enfoque, se los conoce como "aplicación nativa". Una aplicación al ser desarrollada en diferentes plataformas, el funcionamiento muchas veces también es diferente en cada plataforma móvil para el que ha sido desarrollado.

2.2.5.2. Enfoque web

Las aplicaciones web a menudo se describen como multiplataforma. Son accesibles de una multitud de plataformas desde un navegador web. Los desarrolladores han utilizado herramientas para crear aplicaciones web para que también se puedan ver agradables en los dispositivos móviles. Sin embargo, estas aplicaciones carecen de experiencias nativas en cuanto a las interfaces y no pueden acceder a las APIs nativas de los dispositivos móviles

(El-Kassas et al., 2017) y (Nagesh y Caicedo, 2012). Una solución a este problema son las Aplicaciones Webs Progresivas (PWA), impulsando con fuerza en los últimos años, inclusive empresas reconocidas de tecnología ya lo implementaron como es Twitter, con Twitter Lite (Love, 2018).

1. Aplicaciones web progresivas (PWA)

Las PWA son aplicaciones escritas para la web, se ejecutan en un navegador, con la diferencia de ser desarrolladas con algunas técnicas que hacen posible un comportamiento nativo de una aplicación móvil nativa. Estas tienen soporte fuera de línea con ayuda de los Service Workers, son instalables, poseen notificaciones push y tiene acceso a APIs nativas, la cámara y la geolocalización son algunos de ellos (Love, 2018). Al decir que son instalables, no significa que se encuentran en la tienda de alguna de las plataformas móviles. Al ser aplicaciones webs normales, al ingresar en el sitio desde un es dispositivo móvil, el navegador te da la posibilidad de incorporar la aplicación al menú de inicio del móvil, posterior a eso, la aplicación se ejecutará como si fuera nativo.

Las PWA son consideradas multiplataforma, por el hecho de funcionar en cualquier plataforma sea web o móvil con tan solo tener un navegador web, y también en cuestiones de desarrollo, porque solo se tiene una fuente de código, escrito para que una aplicación sea más rápido, de confianza, atractivo e integrado (Love, 2018).

2.2.5.3. Enfoque multiplataforma

El término "desarrollo móvil multiplataforma" se refiere a la creación de aplicaciones a partir de una base de código principal que luego se ejecutara en múltiples sistemas operativos móviles. En este enfoque se tiene dos categorías, uno es el tipo híbrido y el otro nativo, también lo llaman compilado. Ambas categorías son desarrolladas con tecnologías web como es HTML, CSS y JAVASCRIPT.

El desarrollo multiplataforma de tipo híbrido consiste en el encapsulamiento de toda una aplicación web en una “WebView” nativo como es en el caso de Android. Mientras que el desarrollo multiplataforma de tipo nativo, consiste en la compilación a componentes nativos a partir de una base de código escritos en JavaScript.

2.2.5.4. Enfoque multiplataforma de tipo híbrido

Las aplicaciones híbridas son una combinación entre una aplicación web y una aplicación nativa. Todo los sistemas operativos o plataformas móviles tienen una API para mostrar contenidos webs, en caso de Android es el WebView, gracias a esta API, existe la posibilidad de encapsular código html, css, y JavaScript dentro de una aplicación. A diferencia del enfoque web, estas aplicaciones se pueden instalar desde una tienda de aplicaciones. Los principales Frameworks que permiten el desarrollo multiplataforma con enfoque híbrido son: Apache cordova, PhoneGap, Ionic, Framework 7, etc.

Algunos inconvenientes con este enfoque es la experiencia de usuario, que tiene que ver con el diseño de las interfaces y el rendimiento al usar las APIs nativas, como el GPS, cámara, etc, porque para acceder a las APIs nativas existe un intermediario, apache cordova en caso de Ionic por ejemplo (Ravulavaru, 2017, p. 61).

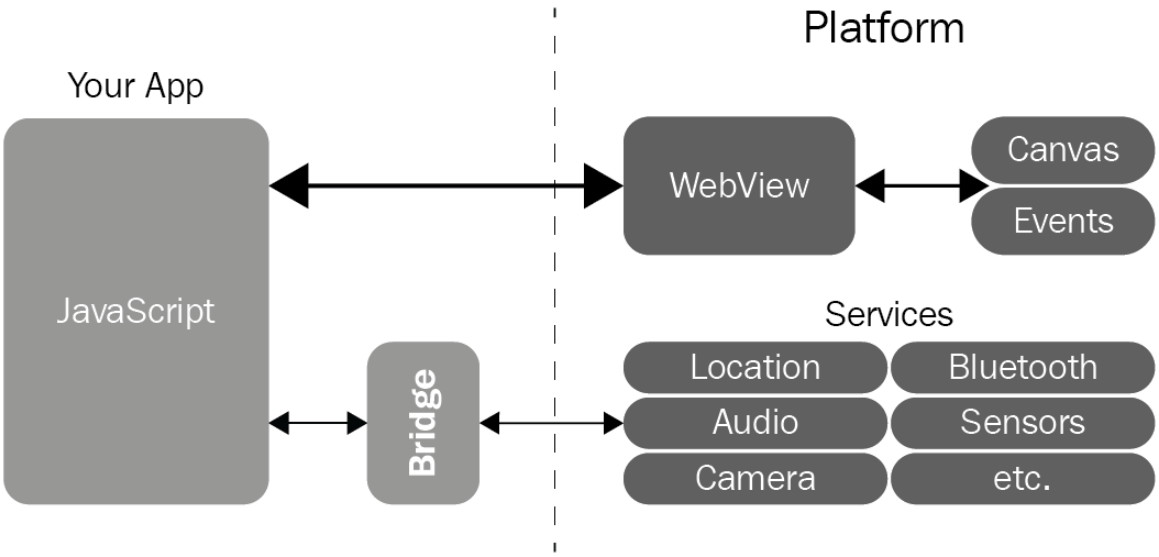


Figura 3. Arquitectura de las aplicaciones móviles de tipo Híbrida
Fuente:(Mainkar & Giordano, 2019)

2.2.5.5. Enfoque multiplataforma de tipo nativo

Este enfoque aparece a raíz de algunas limitaciones y problemas que presenta el enfoque híbrido. También conocido como compilación cruzada, este enfoque, mediante varias

herramientas que aparecieron en los últimos años, hacen posible el desarrollo de una aplicación verdaderamente nativa. A Partir de un código escrito en un lenguaje, estos son transformados a un código nativo de varias plataformas móviles (Latif, Lakhrissi, Nfaoui, & Es-Sbai, 2016). El rendimiento nativo, y todas las características de las aplicaciones nativas, incluido la interfaz nativa es posible con este enfoque.

Existen diferentes frameworks para lograr este propósito, y manejan tecnologías web, incluso haciendo el uso de frameworks avanzados de JAVASCRIPT: Angular, React y Vue. Son los más populares en la actualidad en el desarrollo web (Harryho, 2017). Las tecnologías que tienen un grado de madurez alta en este enfoque son: React Native y NativeScript.

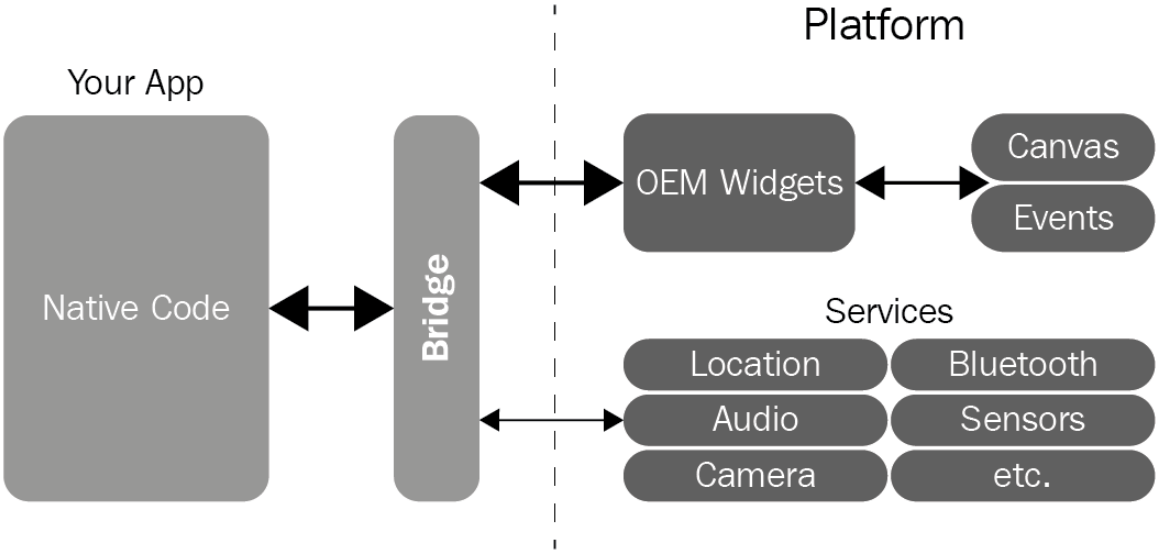


Figura 4. Arquitectura de las aplicaciones móviles de tipo Nativa
Fuente: (Mainkar & Giordano, 2019)

1. React Native

Para implementar aplicaciones móviles con React Native, es necesario conocer React, una biblioteca de JavaScript para crear interfaces de usuario basado en componentes (Facebook Inc., 2016). React Native está basado en el diseño de React, permite la creación de aplicaciones móviles verdaderamente nativas a partir de componentes declarativos que posee, y éstas son compiladas a código nativo de cada plataforma. También tiene la

posibilidad de construir parte de una aplicación en React Native, y parte de la aplicación usando código nativo directamente (Facebook Inc, 2017).

2. NativeScript

NativeScript, a diferencia de React Native, da la posibilidad de implementar aplicaciones para IOs y Android con Angular, Vue, TypeScript y hasta inclusive con el mismo JavaScript (NativeScript community, s. f.).

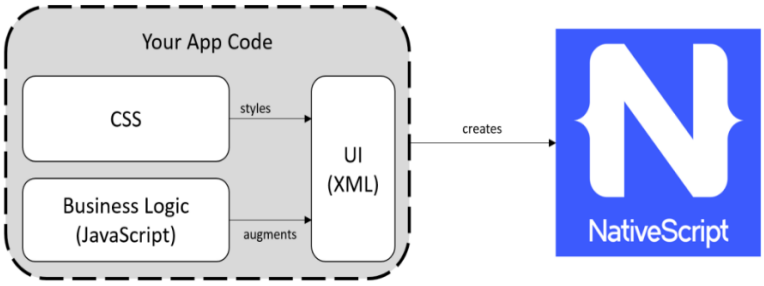


Figura 5. Aplicaciones móviles con JavaScript, CSS, y XML
Fuente: (Branstein & Branstein, 2018)

Como se muestra en la figura 3, NativeScript divide en tres partes su código, JavaScript para el desarrollo de la lógica de negocio, XML para construir las interfaces, y CSS para dar estilos a las interfaces, al igual que desarrollar una aplicación con HTML.

Según Branstein y Branstein (2018) NativeScript es único en el mundo de aplicaciones móviles multiplataforma porque permite escribir código de la UI (XML) una sola vez para ambas plataformas: IOs y Android. Además, los componentes XML, al ser ejecutada la aplicación, terminan siendo elementos de interfaz nativos.

Al igual que otras plataformas multiplataforma, también tiene acceso a las APIs nativas de los dispositivos, sin embargo, la forma que lo hace es distinta, tiene acceso a cada función de la API nativa, características y componentes de hardware del dispositivo.

Para trabajar con otros frameworks, la estructura general de NativeScript se mantiene, pero se aprovecha al máximo las características que poseen estos frameworks, haciendo más fácil y rápido el desarrollo.

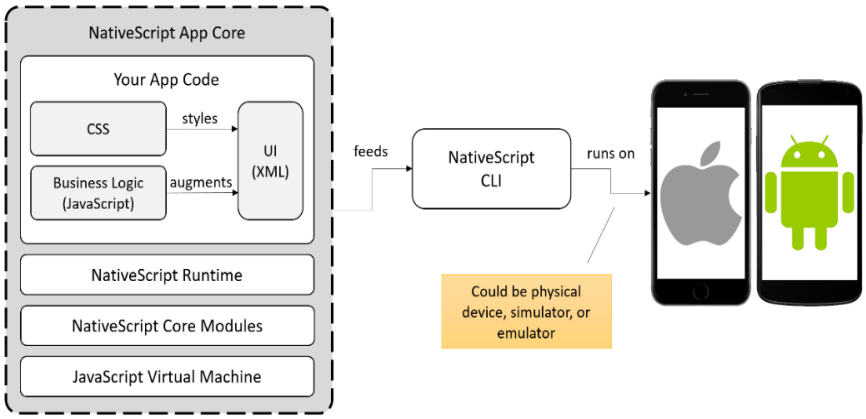


Figura 6. Núcleo de una aplicación NativeScript.
Fuente: (Branstein & Branstein, 2018)

El funcionamiento de NativeScript es mediante varios componentes, como se muestra en la Figura 4. Se tiene el código escrito, éste interactúa con el tiempo de ejecución y los módulos de NativeScript. Luego la herramienta NativeScript CLI agrupa el código, el tiempo de ejecución y los módulos, en una aplicación nativa que contiene una máquina virtual de JavaScript. El tiempo de ejecución y los módulos de NativeScript son bibliotecas que usa NativeScript para crear aplicaciones, y el código más estas bibliotecas se ejecutan dentro de una máquina virtual de JavaScript.

3. Flutter

Flatter; escrito en Dart, es otro de los Framerows hecho en la categoria de multiplataforma de tipo nativo; desarrado por google. Este fue desarrollado con el objetivo de hacer que el desarrollo sea lo más fácil, rápido y productivo posible.

En Flutter el desarrollo es a base a Widgets, desde un simple texto o un botón hasta una animacion o gesto, incluso para acceder a las apis nativas es mediante los widgets (Mainkar & Giordano, 2019). En una aplicación a menudo se ven arboles de widgets compuestos

mediante la herencia. Ademas flutter posee un catálogo completo de widgets en los dos modos más conocidos en el diseño, widgets de material design y cupertino apple, pero eso no quiere decir que los widgets de material solo se pueden usar para Android, sino todo lo contrario, en ambas se ven tal y cual fueron diseñados.

Existen dos tipos principales de widgets en flutter, que son Stateless Widget y Stateful Widget, en la tabla 1 se muestra las diferencias.

Tabla 1. Diferencias entre widget con estado y sin estado.

	Composición dinámica	**Es inmutable**	**Objeto con sub estado mutable**
Widget sin estado	Falso	Verdadero	Falso
Widget con estado	Verdadero	Verdadero	Verdadero

Fuente: Wu, 2018.

El widget de estado viene con un objeto correspondiente que representa el estado. El estado se considera como la capa de presentación de la estructura interna del widget, el estado describe cómo el widget responde a las interacciones del usuario, como cambiar el diseño del widget. Relativamente, el widget sin estado es un widget simple que no responde a la reacción del usuario. Este patrón de diseño permite que el propio widget se mantenga inmutable, lo que evita que el marco vuelva a representar la vista del widget (Wu, 2018).

En comparación de los marcos mencionados anteriormente, Flutter sobre sale frente a todos, inclusive sobre las nativas; muy aparte por la ventaja en el desarrollo de escribir código solo una vez para diferentes marcos. Esto por la forma en la que está diseñado su arquitectura, donde las aplicaciones se compilan en AOT (Ahead Of Time) en lugar de JIT (Just In Time) como las soluciones JavaScript por ejemplo y es a un lenguaje maquina (ARM). Esta solución también elimina el concepto del puente y no se basa en una plataforma OEM (Mainkar & Giordano, 2019), ya que los widgets son de Skia, la cual tambien es el motor de renderizado de Fuchsia; sistema operativo que está siendo creada por Google.

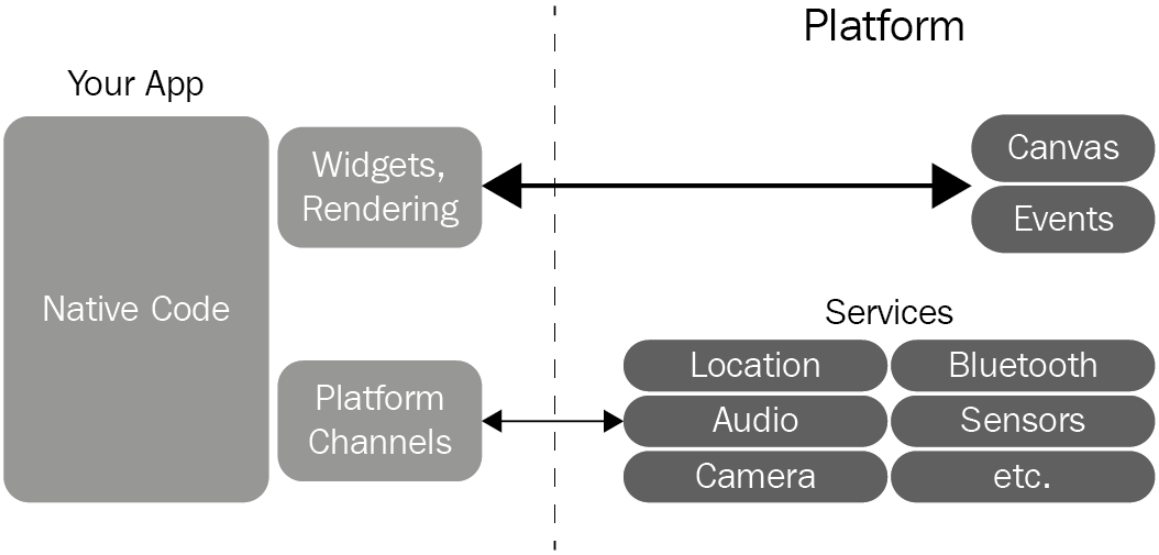


Figura 7. Arquitectura de Flutter
Fuente: (Mainkar & Giordano, 2019)

2.2.6. Kanban

Es una metodología ágil, que se enfoca en definir, gestionar y mejorar el flujo de trabajo eliminando los cuellos de botella (Anderson y Carmichael, 2016). A menudo es comparado con Scrum, también una metodología ágil, que gestiona el trabajo a base de iteraciones o sprints definidos, que además de contar con roles definidos, necesita de un equipo multifuncional (Björkholm y Björkholm, 2015).

Kanban, a diferencias de Scrum, es más flexible, donde las iteraciones o sprint son opcionales y no existen roles definidos (Li, 2016). Se enfoca más en el flujo continuo y tiempo del ciclo en base a políticas definidas, donde la visualización del trabajo en progreso (WIP) es importante para identificar problemas y los cuellos de botella, que serán abordados en su momento para que el proceso fluya sin problemas (Li, 2018), (Ahmad, 2016) y (Anderson & Linden-Reed, s. f.).

Se ha utilizado en las empresas de fábrica y de producción desde hace décadas, sin embargo, es relativamente nuevo en el área de desarrollo de software (Khan, 2014); no tan nuevo, desde el 2005 viene siendo promovido por algunos autores citados en esta

investigación como es David J. Anderson quien se atribuye ser el primer impulsor del proceso kanban para el desarrollo de software (Anderson & Linden-Reed, s. f.).

Según Anderson y Carmichael (2016) y Björkholm y Björkholm (2015), existen tres principios de gestión y tres principios de despliegue de servicios en kanban:

2.2.6.1. Principios de gestión

- Comience con lo que está haciendo ahora.
- Buscar la mejora a través del cambio evolutivo.
- Fomentar el liderazgo en cada nivel de la organización.

2.2.6.2. Principios de despliegue de servicios

- Entender las necesidades y expectativas de los clientes y focalizarse en ellas.
- Gestionar el trabajo, dejar que la gente se auto organice alrededor de las tareas.
- Evolucionar las políticas para mejorar los resultados hacia el cliente y del negocio.

2.2.6.3. Prácticas generales

También están las prácticas generales de Kanban, que definen las actividades fundamentales para el manejo de los sistemas kanban (Anderson y Carmichael, 2016) y (Björkholm y Björkholm, 2015).

1. Visualizar lo que se está haciendo

Esto incluye tanto los pasos en el proceso como el trabajo que se realiza en cada paso que se refleja en un tablero o pizarra, sea física o virtual. Para el diseño del tablero, kanban no restringe como hacerlo, por lo que la personalización es de acuerdo a las necesidades de la organización.

2. Limitar el trabajo en progreso

Limita cuántos elementos de trabajo pueden estar en cada paso del flujo de trabajo a la vez. Los límites son considerados como políticas para el proceso kanban y son establecidas de acuerdo a la capacidad del equipo. Los límites indican que los miembros del equipo pueden ayudar si un miembro se retrasa o pone en espera a otro miembro del equipo.

3. Gestionar el flujo

Consiste en mejorar el flujo de un proceso, y así el tiempo de entrega será más corto. Aquí es donde comienzan a cambiar las organizaciones y la forma en que funcionan; el cambio disminuirá el tiempo de entrega.

4. Hacer explícitas las políticas

Se trata de ser claro sobre el proceso, las políticas y principios que se definen. Estas deben ser simples, bien definidas, visibles, y deben aplicarse siempre y ser modificables cuando sea necesario. El objetivo es que los todo los involucrados conozcan y sigan el proceso y puedan sugerir mejoras. Los límites del trabajo en progreso son un tipo de política, otras pueden ser: asignación de capacidad, definición de Hecho u otras políticas para los entregables de las etapas de un proceso.

5. Implementar ciclos de retroalimentación o feedback

El objetivo es obtener retroalimentación en todas las áreas del proceso, teniendo en cuenta la opinión de los clientes, usuarios finales, y otros interesados externos, de modo que se pueda saber la calidad del servicio que se brinda; no solo importa hacer entregas rápidas. Kanban define siete oportunidades de retroalimentación específicas, o cadencias. Las cadencias son las reuniones y revisiones cíclicas que dirigen la evolución como se muestra algunas de ellas en la figura 5.

6. Mejorar de manera colaborativa, evolucionar experimentalmente

Kanban inicia con el estado actual de la organización, y lo que busca este principio es mejorar evolutivamente el proceso para optimizar el tiempo de entrega. El proceso evolutivo involucra diferenciación, aplicando modelos y métodos científicos sobre el proceso.

2.2.6.4. Roles de Kanban

Kanban no exige ni define roles; solo sugerencias, como mencionan Björkholm y Björkholm (2015): “Solo agregue los roles que realmente necesita y modifíquelos para adaptarlos a su organización y modifíquelos nuevamente cuando la situación cambie”.

CAPÍTULO III. Materiales y Métodos

3.1. Descripción del lugar de ejecución

El proyecto es ejecutado en la oficina de CONCYTEC – UPeU y/o vivienda del investigador. El lugar es elegido porque esta investigación es parte del proyecto "Plataforma digital inteligente y Big Data para el turismo rural comunitario en la Región Puno", proyecto financiado por FONDECYT, y entidad ejecutora y evaluadora CONCYTEC.

3.2. Metodología de la investigación

3.2.1. Tipo de investigación

(CIANCIACTIVA & CONCYTEC, 2017), en las bases integradas de proyectos de investigación básica y aplicada, define una investigación aplicada de la siguiente manera: "Está dirigida a determinar, a través del conocimiento científico, los medios (metodologías, protocolos y/o tecnologías) por los cuales se puede cubrir una necesidad reconocida y específica". Por ende, este proyecto es una investigación aplicada porque cubrirá una necesidad en el sector de turismo con una solución tecnológica.

3.2.2. Materiales e insumos

Los principales materiales e insumos que se emplearon para la ejecución del proyecto fueron los siguientes:

3.2.2.1. Hardware

- Laptop Mac OS
- Dispositivo móvil Android

3.2.2.2. Software

Backend

- TypeScript
- JavaScript

- NodeJs
- Framework LoopBack 4
- MongoDB
- VueJs
- NuxtJs
- Python

Frontend

- Framework Flutter
- Lenguaje de programación Dart
- Editor de texto Visual Studio Code
- Emulador IOs
- Emulador Android
- Redux flutter
- API de Google Map Static
- API de Google Map
- Google cloud platform
- Postman
- Pivotal Web Services
- MongoDB Atlas

Otros

- Adobe XD
- Cacoo
- Sparx Systems Enterprise Architect
- Swift Kanban

3.2.3. Arquitectura de solución

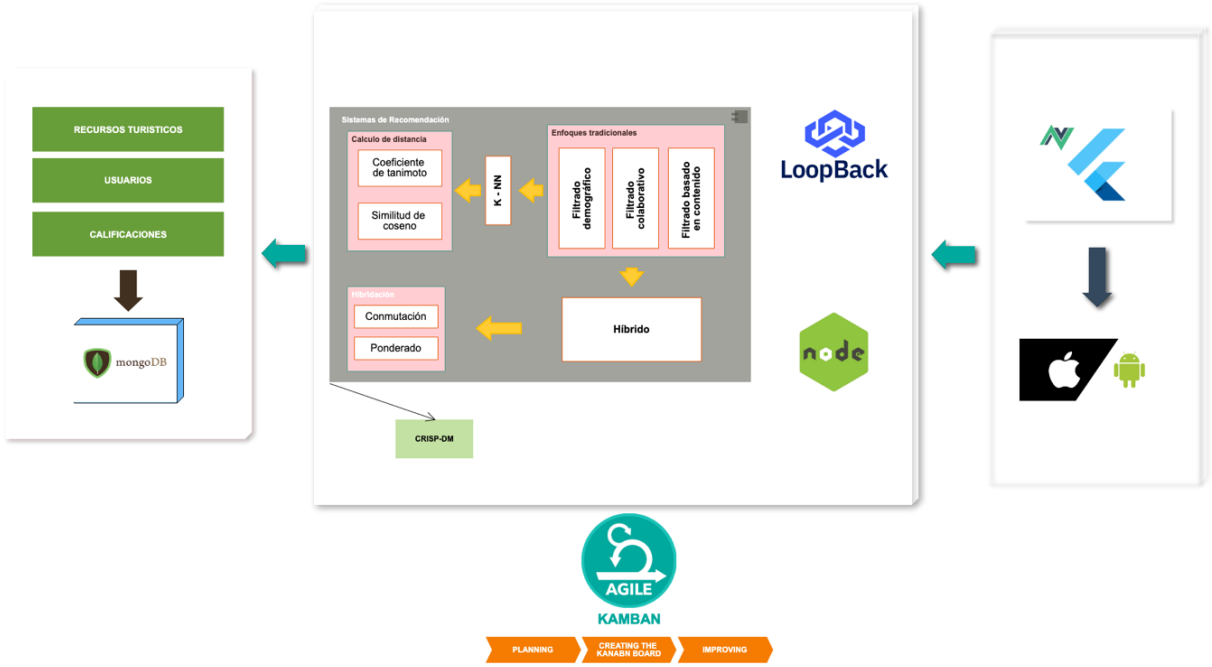


Figura 8. Arquitectura de solución.
Fuente: Elaboración propia.

En la figura 8 se visualiza la arquitectura de solución del proyecto de investigación en 3 apartados. La primera parte representa el almacenamiento de los datos, donde los datos principales para la investigación son los recursos turísticos y los datos demográficos de los usuarios que interactúan con el sistema y que además son los encargados de dar las calificaciones a los recursos turísticos. El segundo apartado representa toda la lógica de los algoritmos, también conocido como el backend. La última parte representa en si la aplicación correspondiste con quien el usuario tendrá interacción.

El flujo correspondiente es el siguiente, un usuario activo, navega en la aplicación realizada con flutter, esta puede ser en Android o IOs, ya que la arquitectura de flutter lo permite, un usuario completa la información demográfica, y mientras va realizando viajes sobre los recursos, y según a cada acción que vaya realizando, por ejemplo, dar una calificación; recibirá recomendaciones. Frente a cada acción que vaya realizando un usuario, inclusive inmediatamente después de su registro, los algoritmos de sistemas de recomendación empiezan a realizar su trabajo, sujeta a ciertos estados y restricciones, y en respuesta a las acciones del usuario, esta propone recomendaciones. Para que los sistemas

de recomendación cumplan con su propósito, estas se alimentan de la base de datos y con los datos adecuados.

3.2.4. Metodología Kanban

Esta metodología se eligió por lo flexible que es, donde las iteraciones o sprint son opcionales y no existen roles definidos.

3.2.4.1. Análisis de la situación actual del proyecto

Se hizo el análisis de la situación actual del proyecto, con los recursos que cuenta, y las necesidades que esta presenta para la creación del tablero y definir el proceso general, y plasmarlo en base a los principios y prácticas generales que la metodología Kanban lo define.

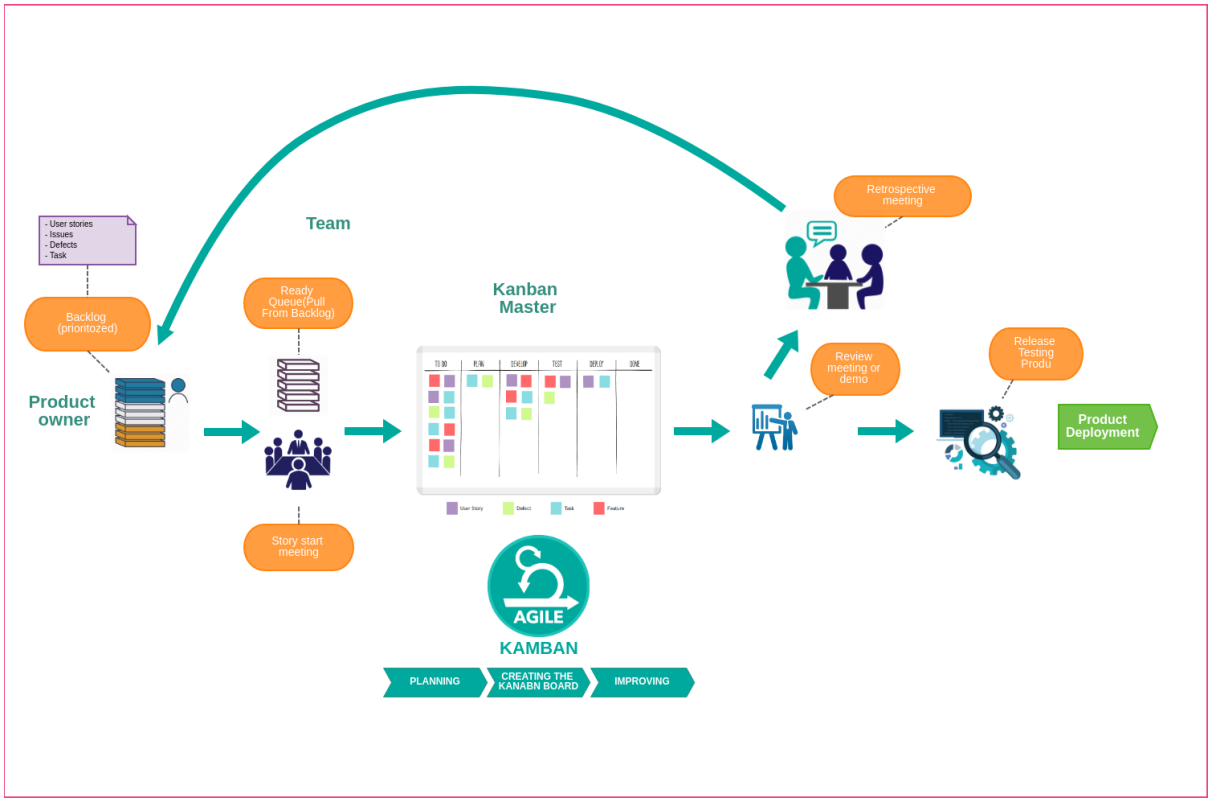


Figura 9. Sistema kanban en una organización
Fuente: Elaboración propia en base a (Björkholm & Björkholm, 2015).

Para la construcción del tablero se tomó en cuenta los objetivos del proyecto, donde cada uno de ellos sigue pasos diferentes para ser completados, inclusive el objetivo 2 sigue un

método en específico. En base al análisis del proyecto, y para tener un control y orden adecuado del flujo de trabajo, se hizo la creación de un tablero por tipo de área. También para organizarlo de esta forma se tomó en cuenta uno de sus principios de la metodología: "**gestionar el trabajo:** dejar que la gente se auto organice alrededor de las tareas".

Tablero para gestionar tareas de análisis y diseño

En la figura 10 se muestra un tablero personalizado para tareas exclusivamente que tengan que ver con el análisis y diseño de la aplicación que se desarrolló. También se definió algunas políticas de límite de trabajo como por ejemplo en la columna "DESARROLLO" solo hasta dos tareas se pueden realizarse en paralelo, sin embargo 5 tareas pueden acumularse en la lista de espera para luego entrar a la columna de validación.

Figura 10. Captura del tablero análisis y diseño.
Fuente: Elaboración propia en Swift Kamban, 2019.

Tablero para gestionar tareas del Backend

La figura 11 muestra un tablero similar al anterior, con una única diferencia, que es exclusivamente para gestionar tareas de tipo backend: creación de APIs, integración de los algoritmos, manejo de datos, etc.

Figura 11. Captura del tablero de tareas del backend
Fuente: Elaboración propia en Swift Kanban, 2019.

Tablero para gestionar tareas de tipo IA

El tablero que de la figura 12 es particularmente diferente a las demás, esto porque la implementación de los algoritmos es un tema relacionado a la minería de datos o inteligencia artificial, lo cual necesitan apoyarse de ciertos métodos para su desarrollo. Para este proyecto se optó por el método de CRISP-DM, únicamente para el cumplimiento del objetivo 2, y de acuerdo a las fases que esta posee se creó las columnas del tablero como se puede observar en la figura.

Figura 12. Captura del tablero de implementación de algoritmos
Fuente: Elaboración propia en Swift Kanban, 2019.

Paso 1: Entendimiento del negocio

Etapa en la cual, se tiene una comprensión profunda del problema que se quiere abordar, como pueden ser, los objetivos del negocio, situación actual, área del problema y soluciones actuales, datos disponibles, requerimientos, restricciones, criterios de evaluación del modelo, etc.

Paso 2: Entendimiento de los datos.

En esta etapa se definen los criterios de selección de los datos, datos que se extraerán, definir fuentes para la extracción de los datos, y una vez extraídos, exploración de los mismos.

Paso 3: Preparación de los datos.

En esta fase se realiza la selección de datos a utilizar en el estudio, y definición de criterios para excluir datos innecesarios para la investigación. Luego se realiza la limpieza de los datos en base a los criterios definidos.

Paso 4: Modelamiento.

Etapa en la cual se implementa el algoritmo k vecinos más cercanos, en los diferentes enfoques de sistemas de recomendación. La implementación del algoritmo se realiza en base a las investigaciones de (Kbaier et al., 2017) y (Safoury & Salah, 2013), donde el proceso del algoritmo de sitemas de recomendación, en sus diferentes enfoques están definidos de manera detallada.

Paso 5: Evaluación.

Finalmente se analizan los resultados del algoritmo implementado, sin embargo, no son sometidos a métodos específicos como para saber la presión o el error del algoritmo.

Tablero para gestionar tareas de UI – movil (Frontend)

Por último, se tiene el tablero para gestionar tareas exclusivas en el desarrollo de la aplicación móvil: como es el desarrollo de las interfaces, consumo de las APIs, etc.

Figura 13. Captura del tablero UI-movil
Fuente: Elaboración propia en Swift Kamban, 2019.

La definición de los límites de trabajo en progreso (WIP), se hizo en base a la capacidad del equipo, en este caso una persona quien lo ejecutará, más otra persona de revisión y aprobación.

La herramienta que se usó para el control del flujo del trabajo es: “swiftkanban”, software kanban para la gestión visual de proyectos Lean, figura 07 (“Kanban Software For Lean Visual Project Management”, 2018). Una de las virtudes de esta herramienta es poder visualizar el trayecto de una tarea desde su creación hasta su finalización, con la peculiar característica de registrar tiempos de espera y trabajo, por ejemplo en la figura 14 se observa en cada columna el tiempo permanecido de dicha tarea, pero hay una diferencia entre ellos, hay columnas de espera y de trabajo, y en base a eso se obtiene los tiempos totales en la gestión de una tarea, lo cual es muy importante y valioso para hacer retroalimentación si se requiere.

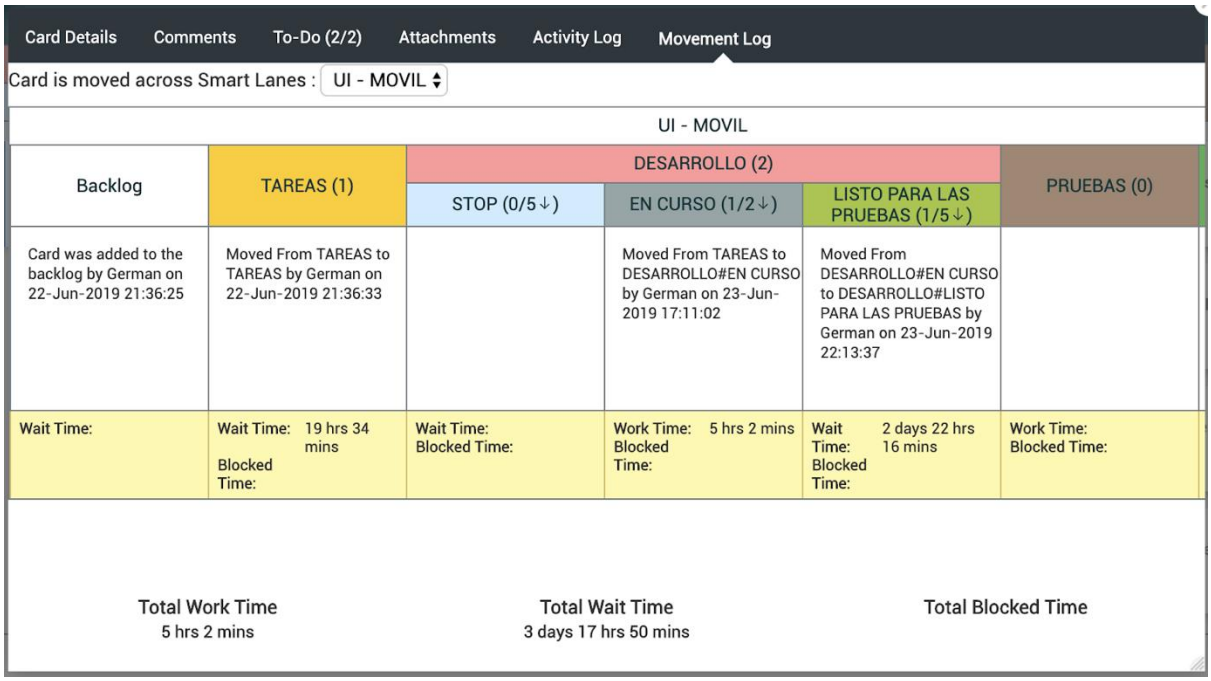


Figura 14. Historial de una tarea en el proceso de su gestión
Fuente: Swift Kamban, 2019.

Para el cumplimiento con los objetivos de la presente investigación, el desarrollo se realiza en base al ciclo de vida del desarrollo del software como lo define la ISO 12207. La metodología kanban propone de manera opcional o cuando el proyecto lo necesite, en realizar iteraciones. Considerando las peculiaridades del proyecto, de tener una parte de inteligencia artificial y la otra de desarrollo de un software, en la esta investigación no se considera necesario realizarlo en iteraciones. Pese a esto el uso de kanban en el proyecto no se anula de ninguna manera, ya que es para la gestión exclusiva de las tareas, en tableros diferentes, según al ciclo de vida del desarrollo del software.

El orden del desarrollo de esta investigación es como se muestra en la figura 15: en primera instancia se realiza el análisis de los requerimientos, luego el diseño, el desarrollo; fase en la cual también se implementan los algoritmos, y finalmente las pruebas.

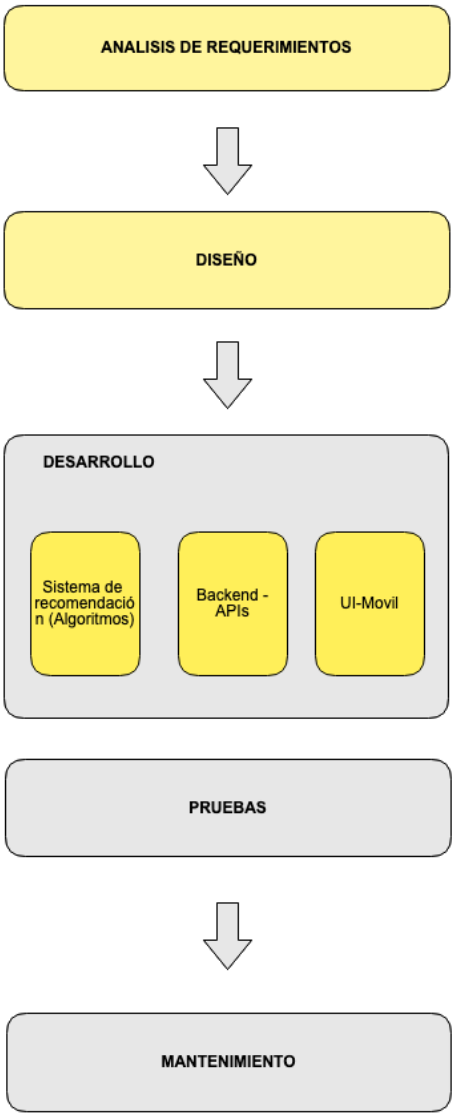


Figura 15. Fases del ciclo de vida del desarrollo de software
Fuente: Elaboración propia, 2019.

CAPÍTULO IV. Desarrollo del aplicativo móvil

4.1. Análisis de requerimientos

La definición de los requerimientos de la aplicación se realizó viendo las necesidades que un turista pueda tener al querer realizar un recorrido de manera interactiva y personalizada durante el viaje.

Para el entendimiento de los requerimientos, estos son plasmados mediante historias de usuario; técnica muy usada en metodologías agiles.

En función de varias investigaciones, y plataformas exploradas previamente, se determinaron los requerimientos funcionales y no funcionales del sistema, donde los requerimientos funcionales fueron desglosados en historias de usuario. Ver todas las historias en Anexo A.

Tabla 2. Requerimientos no funcionales

	Requerimientos
1	Cubrir las tres etapas de un viaje
2	Offline
3	Múltiples idiomas
4	Diseño intuitivo y usabilidad

Fuente: Elaboración propia, 2019.

Tabla 3. Requerimientos funcionales.

	Requerimientos
1	Alta personalización.
2	Automaticidad de una ruta.
3	Ofrecer los puntos más interesantes
4	Ofrecer experiencias de otros turistas
5	Almacenar preferencias y experiencias

6	Reducir tiempos de búsqueda

Fuente: Elaboración propia, 2019.

4.1.1. Definición de roles del sistema

Tabla 4. Roles del aplicativo móvil.

	Rol	Frecuencia de uso de la aplicación	Objetivo
1	Turista	Antes, durante y después de un viaje	Recibir recomendaciones de recursos turísticos según a su perfil, preferencias y recursos que dispone, y en base a las recomendaciones construir una ruta del recorrido, con alta personalización.
2	Emprendedor o dueño de un recurso	Uso frecuente	Recibir informes de posibles visitas de turistas en función de la calificación y elección de rutas. Ver reportes, reacciones, calificaciones sobre un recurso. Mantener actualizada la información de los recursos.
3	Administrador	Poca frecuencia	Registrar emprendimiento y mantener actualizada la información de los recursos.

Fuente: Elaboración propia, 2019.

4.1.2. Historias de usuario

- USHIS001: Acceso mediante redes sociales.
- USHIS002: Contemplar datos demográficos y preferenciales
- USHIS003: Recibir recomendaciones de los recursos turísticos

- USHIS004: Realizar calificaciones a los recursos
- USHIS005: Generar una ruta turística
- USHIS006: Recibir información detallada de los recursos
- USHIS007: Compartir experiencia vivida
- USHIS008: Información en varios idiomas
- USHIS009: Información sin conexión a internet
- USHIS010: Obtener información de servicios cercanos
- USHIS011: Recibir notificaciones
- USHIS012: Información en varios idiomas

4.1.2.1. Descripción de historias de usuario para la aplicación móvil

Tabla 5. Historia de usuario número 1

Enunciado de la historia				
ID: USHIS001		**Rol:** Turista		
Característica / Funcionalidad: Poder acceder a la aplicación móvil mediante redes sociales				
Objetivo: Agilizar y facilitar el acceso				
Prioridad: Baja		**Complejidad:** Muy alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Acceso con una cuenta de Google	Si se tiene con una cuenta de Google	Ninguno	Usuario registrado
2	Acceso con una cuenta de Facebook	Si se cuenta con una cuenta de Facebook	Ninguno	Usuario registrado
3	Acceso con correo electrónico previamente registrado	Si no cuenta con una cuenta de Facebook y Google	Ninguno	Usuario registrado

Fuente: Elaboración propia, 2019.

Tabla 6. Historia de usuario número 2

Enunciado de la historia				
ID: USHIS002	**Rol:** Turista			
Característica / Funcionalidad: Proveer datos demográficos y preferenciales.				
Objetivo: Recibir mejor experiencia en las recomendaciones por parte del sistema.				
Prioridad: Muy alta	**Complejidad:** Media			
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Completar datos demográficos	Ninguno	Ninguno	Tener los datos demográficos en el sistema
2	Completar lista de preferencias	Ninguno	Ninguno	Tener las preferencias en el sistema

Fuente: Elaboración propia, 2019.

Tabla 7. Historia de usuario número 3

Enunciado de la historia				
ID: USHIS003	**Rol:** Turista			
Característica / Funcionalidad: Recibir recomendaciones de los recursos turísticos en base a mis preferencias				
Objetivo: Deducir el tiempo de búsqueda				
Prioridad: Muy alta	**Complejidad:** Muy alta			
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Recomendaciones según datos demográficos	Cuando el usuario activo es nuevo en el	Generar lista de recomendaciones	Presentar al usuario recomendaciones

		sistema y además se tiene datos de otros usuarios en el sistema		en base a sus datos demográficos y de otros usuarios.
2	Recomendaciones según calificaciones y preferencias	Cuando el usuario tenga calificaciones anteriores	Generar lista de recomendaciones	Tener recomendaciones similares a los recursos calificados anteriormente
3	Recomendaciones en base a usuario vecinos	Cuando el usuario tenga calificaciones anteriores y además exista calificaciones de otros usuarios	Generar lista de recomendaciones	Obtener recomendaciones en base a usuarios similares al activo
4	Recomendaciones sujetas a restricciones	Cuando el usuario ingresa alguna restricción o preferencia en particular	Generar lista de recomendaciones	Tener recomendaciones sujetas a las restricciones del usuario
5	Recomendación combinada	Cuando se cumple todas las anteriores	Generar lista de recomendaciones	Tener las mejores recomendaciones de cada enfoque

Fuente: Elaboración propia, 2019.

Tabla 8. Historia de usuario número 4

Enunciado de la historia				
ID: USHIS004	**Rol:** Turista			
Característica / Funcionalidad: Realizar calificaciones sobre los recursos turísticos.				
Objetivo: Obtener recomendaciones en base a las calificaciones y ayudar a otros usuarios				
Prioridad: Muy alta	**Complejidad:** Media			
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Calificar un recurso	Al haber visitado el recurso	Actualiza el promedio de calificación del recurso	Guardar calificaciones

Fuente: Elaboración propia, 2019.

Tabla 9. Historia de usuario número 5

Enunciado de la historia				
ID: USHIS005	**Rol:** Turista			
Característica / Funcionalidad: Debe generar una ruta turística				
Objetivo: Optimizar el viaje				
Prioridad: Baja	**Complejidad:** Muy alta			
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Construir en base a las recomendaciones obtenidas	Cuando el usuario este satisfecho con las recomendaciones	Construir ruta	Ruta construida
2	Descartar recursos de las	Cuando el usuario no quiera visitar	Construir ruta	Ruta construida

	recomendaciones obtenidas y construir la ruta	uno o más recursos		
3	Agregar recursos a la lista de recomendaciones y construir la ruta	Cuando el usuario desea visitar uno o más recursos que no están en la lista de recomendaciones	Construir ruta	Ruta construida
4	Visitar todos los recursos sin repetir	Ninguno	Ninguno	Ruta optima

Fuente: Elaboración propia, 2019.

Tabla 10. Historia de usuario número 6

Enunciado de la historia				
ID: USHIS006		**Rol:** Turista		
Característica / Funcionalidad: Recibir información detallada de los recursos				
Objetivo: Mantener informado sobre un determinado recurso				
Prioridad: Muy alta		**Complejidad:** Alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Información del clima	Ninguno	Presentar advertencias en situaciones negativas	Información detallada
2	Información descrita y fotográfica	Ninguno	Ninguno	Información detallada

Fuente: Elaboración propia, 2019.

Tabla 11. Historia de usuario número 7

Enunciado de la historia				
ID: USHIS007		**Rol:** Turista		
Característica / Funcionalidad: Compartir mis experiencias de los viajes sobre los recursos.				
Objetivo: Interactuar con otros usuarios				
Prioridad: Baja		**Complejidad:** Muy alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Comentar sobre un recurso	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios
2	Comentar sobre una ruta	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios
3	Comentar sobre un servicio	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios
4	Los comentarios incluyen material visual (fotografías y emojis)	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios
5	Compartir o Recomendar rutas o recursos	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios
6	Compartir o Recomendar rutas o recursos por Facebook y WhatsApp	Si el usuario lo requiera	Ninguno	Crear interacción con otros usuarios

Fuente: Elaboración propia, 2019.

Tabla 12. Historia de usuario número 8

Enunciado de la historia	
ID: USHIS008	**Rol:** Turista
Característica / Funcionalidad: Tener la información en varios idiomas	

Objetivo: Mejor comprensión				
Prioridad: Baja		**Complejidad:** Alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Información en español	Si el móvil este en este idioma	Cambio de idioma	Información en el idioma según corresponda
2	Información en inglés	Si el móvil este en este idioma, excepto en español o portugués	Cambio de idioma	Información en el idioma según corresponda
3	Información en portugués	Si el móvil este en este idioma	Cambio de idioma	Información en el idioma según corresponda

Fuente: Elaboración propia, 2019.

Tabla 13. Historia de usuario número 9

Enunciado de la historia				
ID: USHIS009		**Rol:** Turista		
Característica / Funcionalidad: Tener la información sin conexión a internet				
Objetivo: Disponibilidad de la información				
Prioridad: Baja		**Complejidad:** Alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Funcionamiento offline	Si no te cuenta con conexión a internet	Activar modo offline	Información que el usuario necesite

Fuente: Elaboración propia, 2019.

Tabla 14. Historia de usuario número 10

Enunciado de la historia				
ID: USHIS010		**Rol:** Turista		
Característica / Funcionalidad: Obtener información de servicios cercanos				
Objetivo: Facilitar el acceso a los servicios				
Prioridad: Baja		**Complejidad:** Muy alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Transporte	Si el usuario lo requiera	Ninguno	Información de servicios cercanos
2	Alimentación	Si el usuario lo requiera	Ninguno	Información de servicios cercanos
3	Alojamiento	Si el usuario lo requiera	Ninguno	Información de servicios cercanos

Fuente: Elaboración propia, 2019.

Tabla 15. Historia de usuario número 11

Enunciado de la historia				
ID: USHIS011		**Rol:** Emprendedor o dueño del recurso		
Característica / Funcionalidad: Recibir notificaciones sobre calificaciones y posibles visitas				
Objetivo: Mantener informado				
Prioridad: Baja		**Complejidad:** Muy alta		
Criterios de aceptación				
Nro. de escenario	**Título**	**Contexto**	**Evento**	**Comportamiento esperado**
1	Posibles visitas	Cuando un recurso o emprendimiento es incluido en una	Envía notificación	Usuario notificado

		ruta		
2	Comentarios de los recursos o emprendimientos	Cuando un turista realiza un comentario	Envía notificación	Usuario notificado
3	Calificaciones a los recursos o emprendimientos	Cuando un turista realiza una calificación	Envía notificación	Usuario notificado

Fuente: Elaboración propia, 2019.

4.2. Diseño

4.2.1. Diagrama de clases

En la Figura 16 se muestran los modelos principales de la base de datos. Para esta investigación fueron considerados como indispensables los siguientes:

- Resource, donde está la información del recurso turístico.
- User, Información de autentificación del usuario.
- Demographic, datos demográficos del usuario.
- Rating, calificación del usuario sobre un recurso

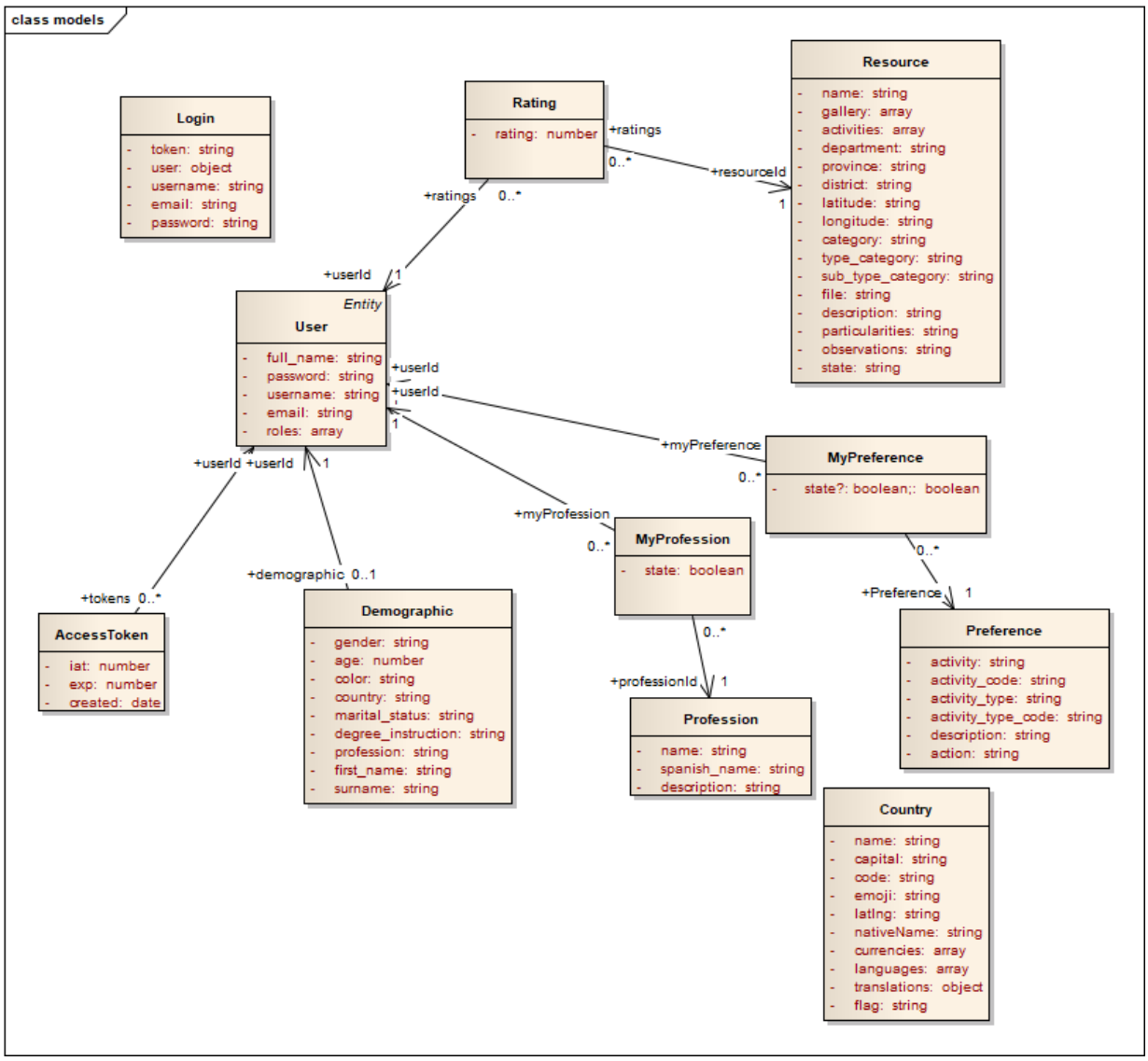


Figura 16. Diagrama de clases
Fuente: Elaboración propia, 2019.

4.2.2. Arquitectura tecnológica

En la Figura 17 se muestra la arquitectura tecnología propuesta para esta investigación, donde se ve es a groso modo que es una arquitectura cliente servidor. Para la construcción del servidor o el Backend se utilizó el Framework Loopback en su versión 4; un Framework escrito sobre NodeJS y apoyado sobre expressJS para la construcción de API REST. La característica peculiar de LoopBack es la creación de modelos, controladores, y todo lo que

conlleva a la arquitectura mediante el uso de la terminal, o su herramienta de cliente, y eso facilita en gran manera el desarrollo.

Otra de las características del Framework empleada en el servidos es su capacidad de conectarse a múltiples recursos de base de datos, y en sus dos tipos, las No SQL y SQL. Tambien tiene la capacidad de conectarse a servicios externos.

En el servidor es donde esta toda la lógica de la aplicación, y son expuestas mediante las APIs REST; obviamente con restricciones de acceso si se requiere, alimentada desde una base de datos, en este caso una base de datos No SQL, como es mongoDB. La elección de esta base de datos es por las características que posee. Muy aparte del código abierto que es, es una base de datos orientada a colecciones y documentos, en otras palabras, tiene una estructura de BSON, una similar a JSON. Se desataca por la facilidad de uso, integración e implementación.

Una vez las APIs son expuestas, se hace la conexión desde cualquier Framework de Frontend, sea para una aplicación web o móvil. Para esta investigación inicialmente se propuso desarrollar la aplicación móvil en NativeScript; recomendado para desarrolladores web, y aplicaciones que no manejan grandes cantidades de datos, sin embargo, esta fue cambiado por cuestiones de rendimiento de la aplicación, por el cual se optó por Flutter por las características que ofrece.

También se hace el uso de Vuejs, para una herramienta web, donde se realiza el consumo de las mismas APIs, pero para un tema de pruebas, y para realizar algunas transformaciones de los datos para ser enviadas al servidor.

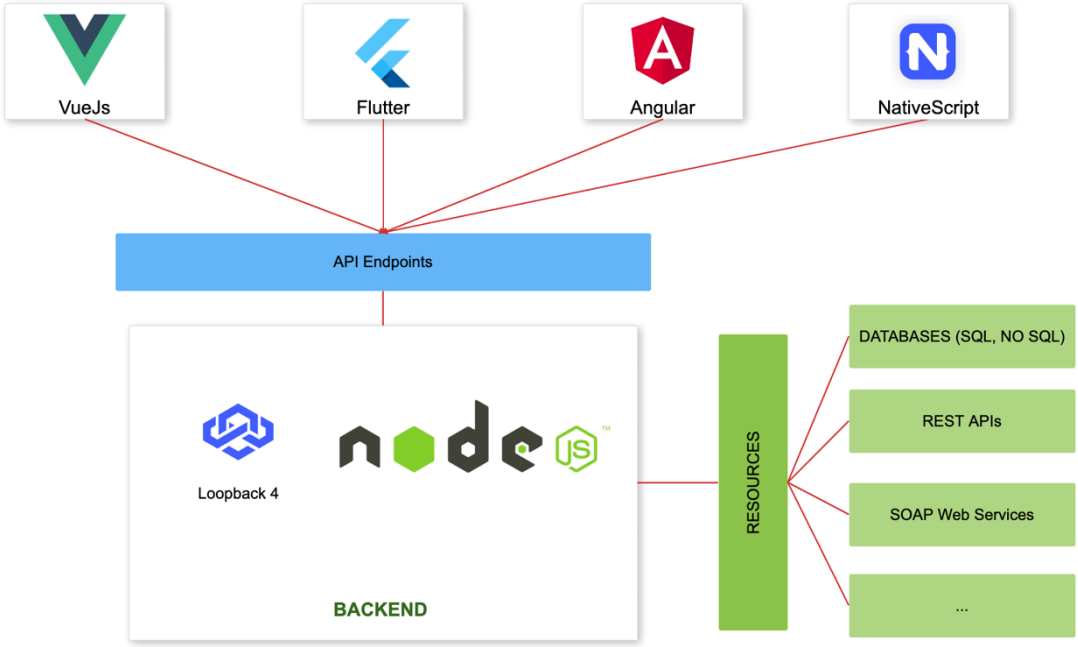


Figura 17. Arquitectura tecnológica.
Fuente: Elaboración propia, 2019.

4.2.3. Prototipo de la aplicación móvil

El la figura 18 se muestra la vista del login y registro de un usuario en la aplicación móvil, pero eso no significa que un usuario necesariamente tiene que autentificarse para acceder a la aplicación, esto solo es necesario cuando un turista desea dar calificaciones o hacer algún comentario, mientras no sea el caso, el turista debería poder explorar los recursos sin ninguna restricción.

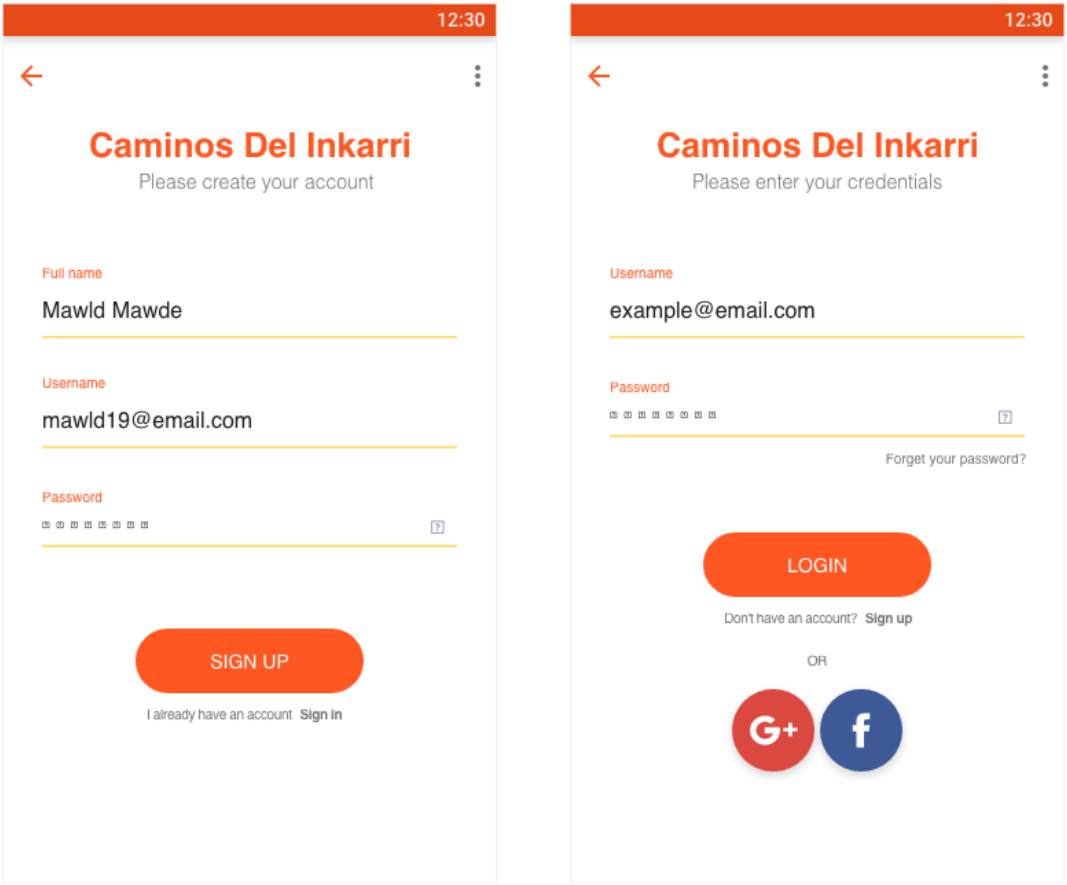


Figura 18. Prototipo de la autentificación
Fuente: Elaboración propia, 2019.

En la figura 19 se observa la vista de la información y las preferencias de un usuario, el llenado de esta información no es necesaria, pero si es recomendada para que el turista pueda recibir recomendaciones en función a la información proporcionada.

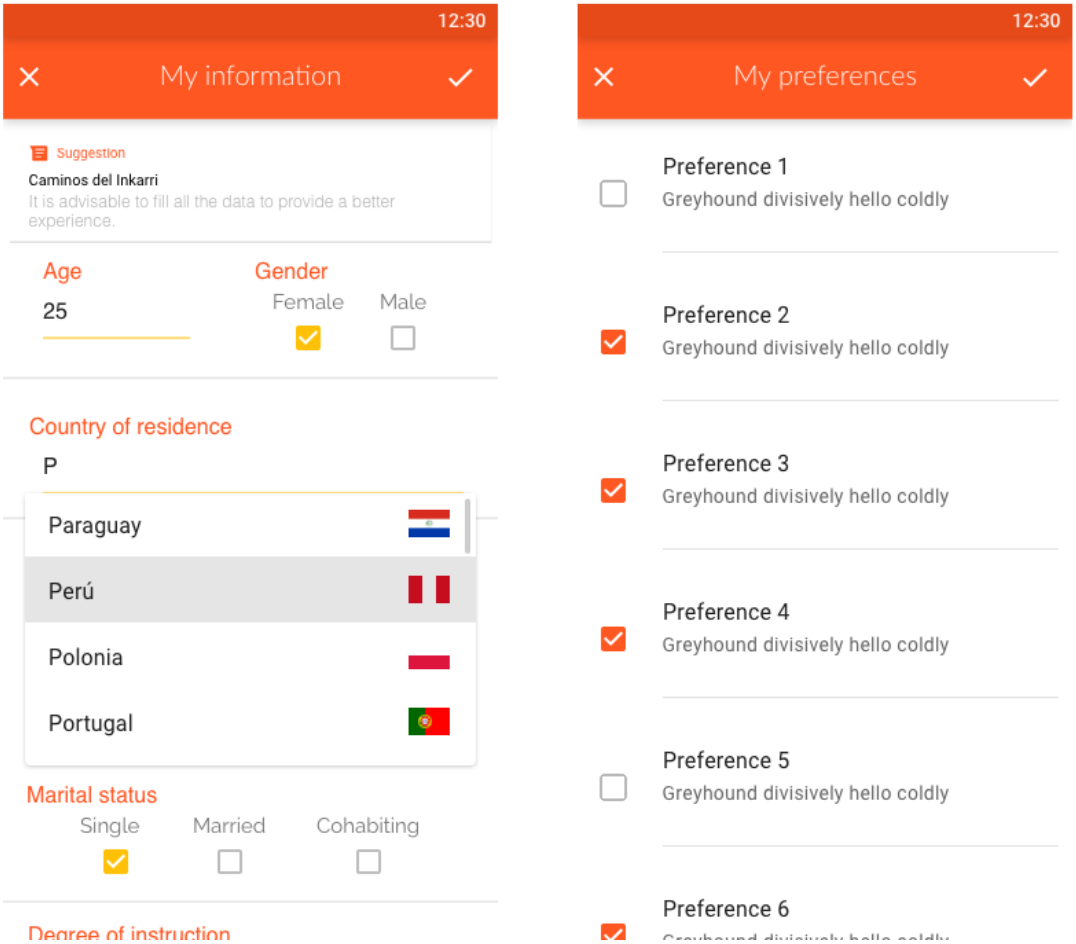


Figura 19. Prototipo de información y preferencia del usuario
Fuente: Elaboración propia, 2019.

La siguiente figura muestra la lista de los recursos y detalles de cada uno, esta vista es la principal de la aplicación, porque para su acceso no se requiere autentificación. Sin embargo, en esta vista el usuario ya puede realizar alguna interacción; calificar el recurso, por ejemplo, pero para ello es necesario autentificarse o crear una cuenta.

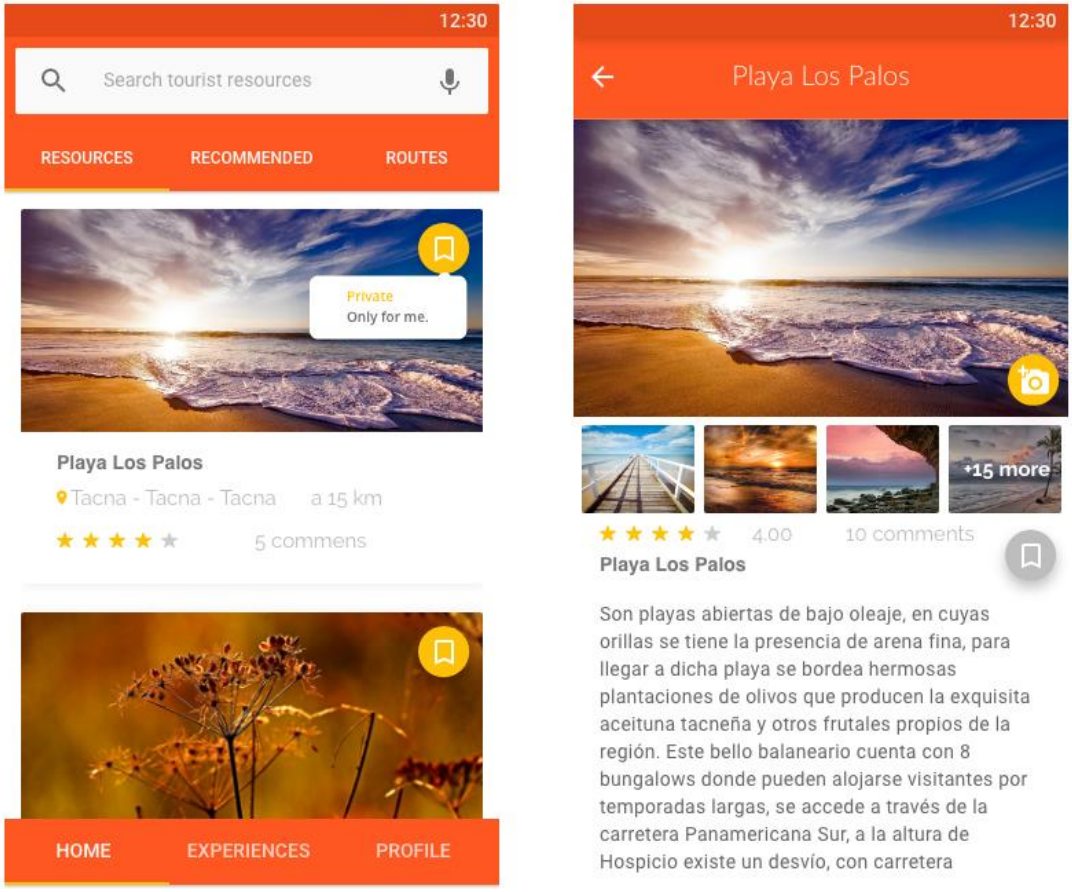


Figura 20. Prototipo de la lista de recursos y su información
Fuente Elaboración propia, 2019.

Los prototipos mostrados anteriormente son las principales que la componen para el cumplimiento de los objetivos de esta investigación, en cuanto a la aplicación propiamente dicho. Ver todo el prototipo en Anexo B.

4.3. Desarrollo

4.3.1. Desarrollo del sistema de recomendación

Se llegó a implementar dos enfoques de los sistemas de recomendación. Para el desarrollo de estos enfoques se aplicó el método de CRISP-DM, que consta de 5 pasos descrita de la siguiente maneara.

Sistema de recomendación basado en contenido y basado en datos demográficos

Es importante mencionar que el objetivo de esta investigación no es modelar el algoritmo como tal, porque ya existen muchas investigaciones sobre el tema. Lo que se buscó es replicar tales soluciones al turismo, con referencia absoluta de dos investigaciones.

Para la recomendación basado en contenido se usó como guía la investigación de (Kbaier et al., 2017) titulado "A Personalized Hybrid Tourism Recommender System". Y para la recomendación basado en datos demográficos se utilizó la investigación de (Safoury & Salah, 2013) titulado "Exploiting User Demographic Attributes for Solving Cold-Start Problem in Recommender System" .

4.3.1.1. Compresión del negocio

Se desea realizar recomendaciones de los recursos turísticos hacia los visitantes en base a sus preferencias. Para ello los sistemas de recomendación son la mejor solución. Dentro de los sistemas de recomendación se encuentran diferentes enfoques, los cuales solucionan problemas de forma diferente y en situaciones distintas.

Uno de los enfoques implementados es el sistema de recomendación basado en contenido. Su función es recomendar recursos a un usuario a partir de sus calificaciones sobre un recurso o actividad.

El problema de este enfoque es cuando un usuario es nuevo y no tiene calificaciones sobre los recursos, a esto se le conoce como: "inicio en frio".

Otro de los enfoques implementados es el filtrado basado en datos demográficos del usuario. La función del algoritmo es recomendar recursos en base a sus datos demográficos, obteniéndolos a partir de usuarios similares.

4.3.1.2. Comprensión de los datos

Para ambos enfoques los datos que se necesitan son:

- Usuarios
- Recursos turísticos
- Calificaciones a los recursos

Los datos se obtuvieron de la plataforma web MINCETUR con la técnica llamada scraping; técnica usada para extraer datos de los sistemas web.

Una vez obtenida los datos, un total de 4833 recursos a nivel nacional, hubo dentro de ella información repetida e incompleta.

En los datos obtenidos se encontró información necesaria y suficiente para el cumplimiento del objetivo, pero en cierta cantidad de recursos. De los datos obtenidos se definió las métricas de selección para solo considerar datos válidos, obteniendo un total de 230 recursos.

4.3.1.3. Selección de características

Las características consideradas para el filtrado basado en contenido son las actividades de los recursos.

Las actividades de los recursos están categorizadas en categorías y sub categorías. Para el algoritmo se consideraron las subcategorías, que son más específicos. No se consideraron sub categorías de tipo “Otros”.

Al final se obtuvo 48 actividades en total, de los cuales, por criterio personal, viendo la disminución de los recursos, se limitó a que por lo menos en un recurso exista 5 actividades y 5 fotografías.

Para el filtrado basado en datos demográficos del usuario se consideraron las siguientes características:

- Edad
- Género
- País
- Color favorito
- Estado social
- Grado de instrucción

4.3.1.4. Modelamiento

Filtrado basado en contenido

Según (Kbaier et al., 2017), los pasos para obtener dichos resultados son los siguientes:

- Identificar actividades no calificadas por el usuario activo.
- Para cada actividad no calificada, calcule su similitud con toda la actividad calificada por el usuario activo utilizando la fórmula de similitud de distancia euclidiana.
- Encuentre para cada actividad no calificada su actividad más cercana y pronostique su tasa (la tasa de la actividad calificada similar más alta).

Para calcular la similitud entre un recurso y otro se utilizó la fórmula de distancia euclidiana de n dimensiones, también es empleada en las investigaciones mencionadas.

Ecuación 1. Distancia euclidiana par espacios bidimensionales

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 2. Espacio euclídeo de n-dimensiones

$$d_E(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Se implementó el enfoque siguiendo los pasos mencionados.

Filtrado demográfico

Según (Safoury & Salah, 2013) y (Kbaier et al., 2017) los pasos principales del algoritmo son los siguientes:

- Limpieza y transformación de datos demográficos del usuario
- Cálculo de similitud del usuario activo entre todos los usuarios del sistema.
- Obtención de k vecinos más cercanos en cuando a la similitud obtenida.
- De los k usuario obtener los recursos calificados positivamente.
- Calcular la frecuencia por recurso

- Obtención de k recursos con frecuencias más altas.

Para el cálculo de similitud entre los usuarios se utilizó la fórmula de coeficiente de tanimoto, que fue empleado en las investigaciones anteriormente mencionadas.

Ecuación 3. Coeficiente de Jaccard / Tanimoto

$$T = \frac{N_c}{N_a + N_b - N_c}$$

Se implementó el enfoque siguiendo los pasos mencionados.

Para la implementación o la programación de estos enfoques se utilizó el lenguaje de NodeJS y ejecutados por terminal, además estas fueron probadas con datos generados aleatoriamente en caso del filtrado basado en contenido, y para el caso de filtrado demográfico los datos se obtuvieron de MovieLens como lo hace la investigación de (Safoury & Salah, 2013).

4.3.1.5. Evaluación

De cierta forma los modelos están probados en las investigaciones ya mencionadas, por lo que no fueron necesarios de realizarlo, ya que tampoco es prioridad en la investigación.

4.3.1.6. Despliegue

Una vez implementado estas fueron incorporadas en las APIs del Backend, lo cual se dallan algunos fragmentos del código en el apartado siguiente.

4.3.2. Desarrollo del Backend – APIs

Para el desarrollo del backend se utilizó el Framework LoopBack 4, esta te provee una arquitectura completa para crear APIs REST. Un proyecto de LoopBack 4, se compone de tres capas principales: modelos, controladores y repositorios.

Básicamente toda la lógica del negocio se realiza en los controladores, y en estas mismas las APIs son expuestas. Para realizar una conexión de los controladores a los modelos están los repositorios.

En la Figura 20 se muestra algunos de los controladores del proyecto y dentro de ellas los métodos específicos acompañados de un endPoint.

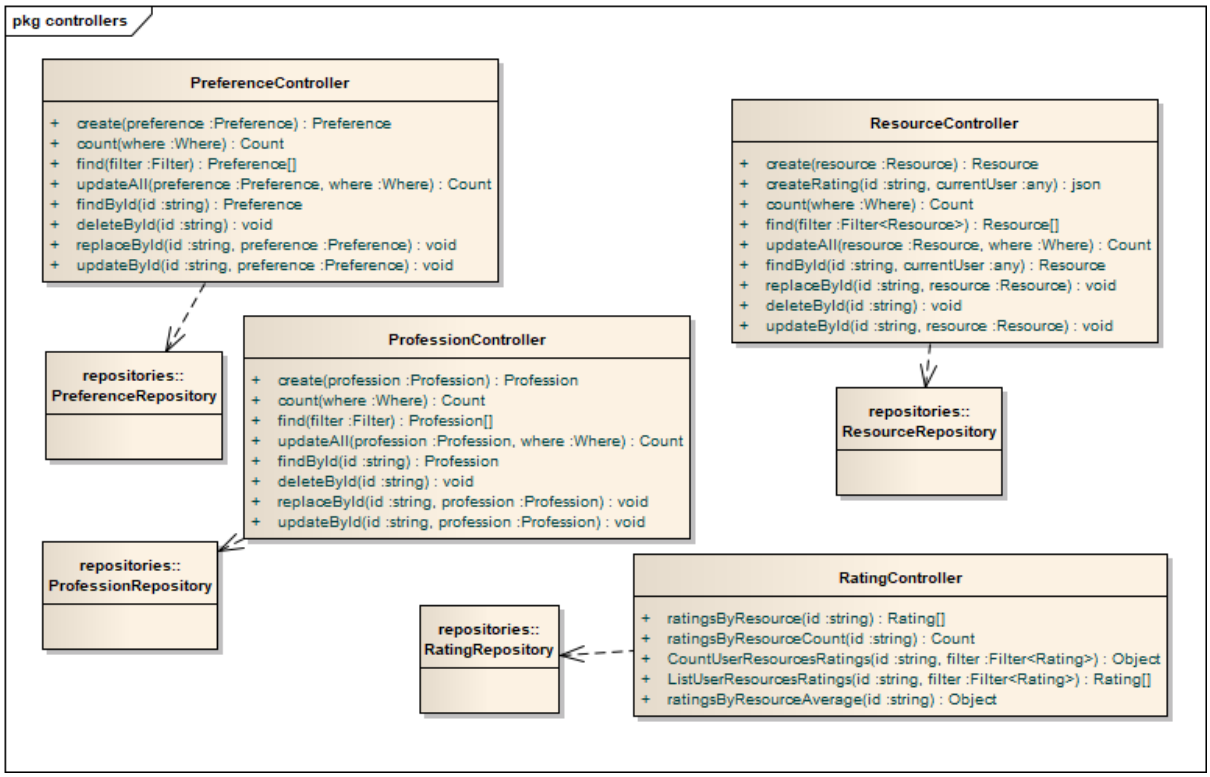


Figura 21. Diagrama de clases.
Fuente: Elaboración propia, 2019.

En la figura 21, se muestra la definición del controlador para realizar la recomendación de filtrado basado en contenido. Este controlador necesita de cuatro modelos, los cuales se conectan mediante los repositorios; estas son definidas en el constructor. También se puede observar el método en cual contiene el endpoint API REST: “content-based-filtering”. Esta ruta necesita autorización, en la cual viene el usuario activo. Al final una vez pasado por todos los pasos del algoritmo la respuesta es un Array de Recursos.

```
export class ContentBasedFilteringController {
  constructor(
    @repository(ResourceRepository)
    public resourceRepository: ResourceRepository,
    @repository(RatingRepository)
    public ratingRepository: RatingRepository,
    @repository(UserRepository)
    public userRepository: UserRepository,
    @repository(PreferenceRepository)
    public preferenceRepository: PreferenceRepository,
  ) { }
  @authenticate('jwt')
  @get('/content-based-filtering', {
    responses: {
      '200': {
        description: 'Array of Resource model instances',
        content: {
          'application/json': {
            schema: { type: 'array', items: { 'x-ts-type': Resource } },
          },
        },
      },
    },
  })
  async find(
    @inject('authentication.currentUser') currentUser: any,
  ): Promise<Resource[]> { }
}

```

Figura 22. Definición de un controlador en LoopBack 4.
Fuente: Elaboración propia, 2019.

Dentro de un controlador se define toda la lógica del algoritmo, por ejemplo, en la siguiente figura se tiene los primeros pasos del algoritmo. Se puede observar el uso de los repositorios para la obtención de los datos del recurso en este caso.

```
1
2  //! PARA ESTE CASO UNA ACTIVIDAD REPRESENTA UN RECURSO
3
4      /**
5       * * PASO 1:
6       * !Identificar actividades no calificadas por el usuario activo.
7
8       */
9
10     /**
11      * Obtener datos de los recursos solo con algunos campos
12      */
13     const resources: Resource[] = await this.resourceRepository.find({
14       //limit: 50,
15       //fields: {_id: true, department: true, name: true, activities: true},
16       //include: [{relation: 'ratings', scope: {fields: {ratings: true}}}],
17       //order: ['name DESC'],
18     });
19
20     /**
21      * Obtengo todas las preferencias para ayudarme a construir una matriz de actividades de los recursos
22      */
23     const preferences: Preference[] = await this.preferenceRepository.find({
24       fields: {activity_code: true},
25       order: ['activity_code ASC'],
26     });
27
```

Figura 23. Uso de los repositorios
Fuente: Elaboración propia, 2019.

Para el cálculo de las distancias se creó una clase, en la cual está definida el método “euclidean” que básicamente determina la similitud entre un recurso y otro con entraras de n-dimensiones y ambas de igual longitud. En nuestro caso el algoritmo recibe 48 dimensiones que representan a las actividades de los recursos. Más detalles del todo el proceso en el anexo C y D.

```
1
2 class Distances {
3   constructor() {}
4
5   euclidean(x: number[], y: number[]) {
6     if (x.length !== y.length) {
7       return 'x has a different size than y';
8     }
9     let accumulator: number = 0;
10    for (let item = 0; item < x.length; item++) {
11      accumulator = accumulator + Math.pow(x[item] - y[item], 2);
12    }
13    return Math.sqrt(accumulator);
14  }
15 }
16
17
```

Figura 24. Distancia euclidiana de n dimensiones.
Fuente: Elaboración propia, 2019.

Finalmente, en la figura 24 se muestra el famoso algoritmo KNN, que está personalizado para obtener k recursos.

```
knn(list: any[], k: number) {
    const list1 = list.sort((x, y) => x.distance - y.distance);
    const temp: any = {};

    for (let index = 0; Object.keys(temp).length < k; index++) {
      if (!temp[list1[index].resourceId]) {
        temp[list1[index].resourceId] = [];
        temp[list1[index].resourceId].push(list1[index]);
      } else {
        temp[list1[index].resourceId].push(list1[index]);
      }
    }
    const result: string[] = [];
    for (const iterator in temp) {
      result.push(iterator);
    }
    return result;
  }

```

Figura 25. KNN para obtener k recursos.
Fuente: Elaboración propia, 2019.

4.3.3. Desarrollo de la aplicación móvil

4.3.3.1. Desarrollo con NativeScript-Vue

En la figura 25, se muestra la estructura típica de un archivo VueJS, significa que, para los desarrolladores web, el Framework es una buena opción, también fue elegido por ese motivo en esta investigación.

```
<template>
  <Page @navigatingTo="navigatingTo">
    <CustomActionBarTrans/>
  </Page>
</template>

<script>
import { mapState } from 'vuex';
const app = require('tns-core-modules/application');
const platform = require('tns-core-modules/platform');
const color = require('tns-core-modules/color');

import CarouselItemCustom from '../components/CarouselItemCustom';

export default {
  components: {
    CustomActionBarTrans,
  },
  data: () => ({
    password: '',
    username: '',
  }),
  mounted() {},
  computed: {
  },
  methods: {
   ---
  },
};
</script>
<style scoped lang="scss">

</style>

```

Figura 26. Estructura básica de un archivo vuejs
Fuente: Elaboración propia, 2019.

Se realizó el desarrollo con dicho Framework algunas pantallas principales de la aplicación inclusive se hizo el consumo de las APIs, sin embargo, en el proceso del desarrollo se experimentó problemas de rendimiento en el renderizado de las vistas, sobre todo cuando se trata de una enorme cantidad de datos. Se buscó la solución correspondiente

sin ningún éxito. Motivo por el cual se optó de cambiar el Framework. En la figura 26 se muestra la estructura del proyecto avanzado y algunas capturas de la aplicación en la figura 27 y 28.

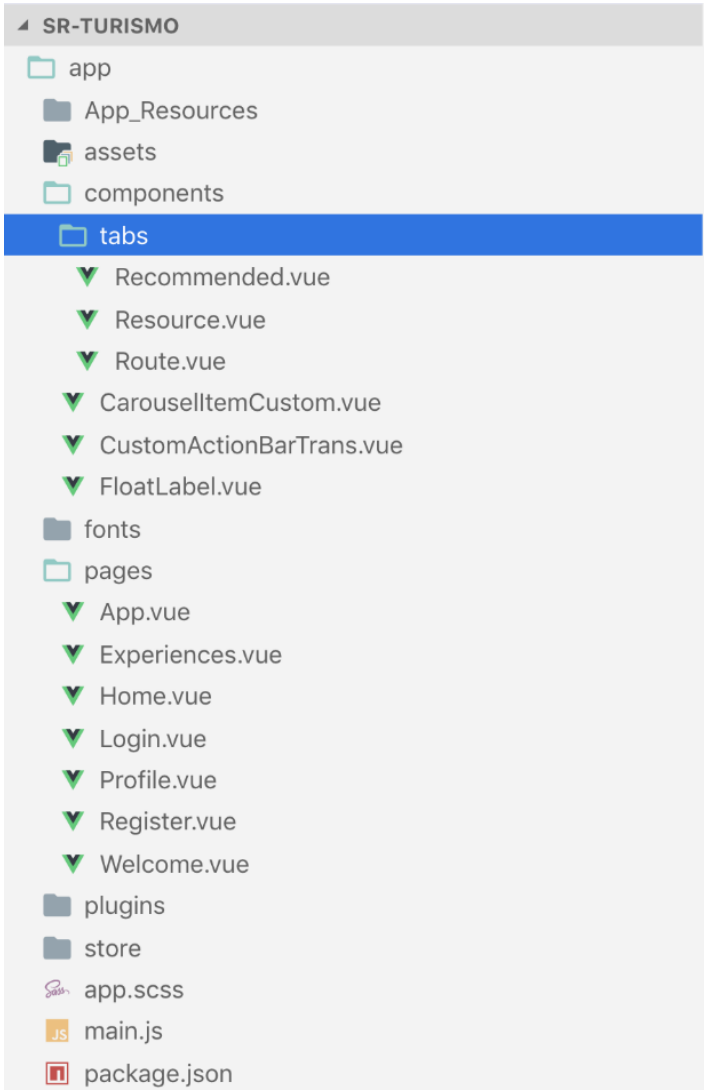


Figura 27. Estructura de un proyecto NativeScript
Fuente: Elaboración propia, 2019.

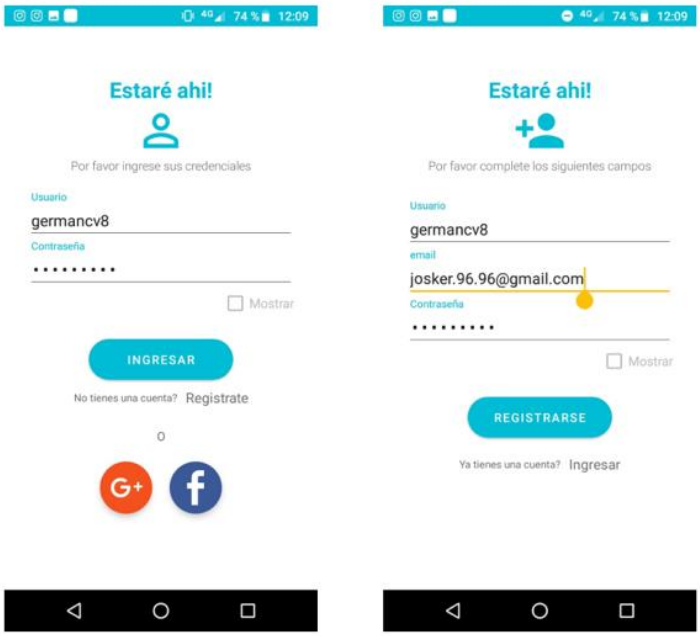


Figura 28. Autentificación con Nativescript-vue
Fuente: Elaboración propia, 2019.

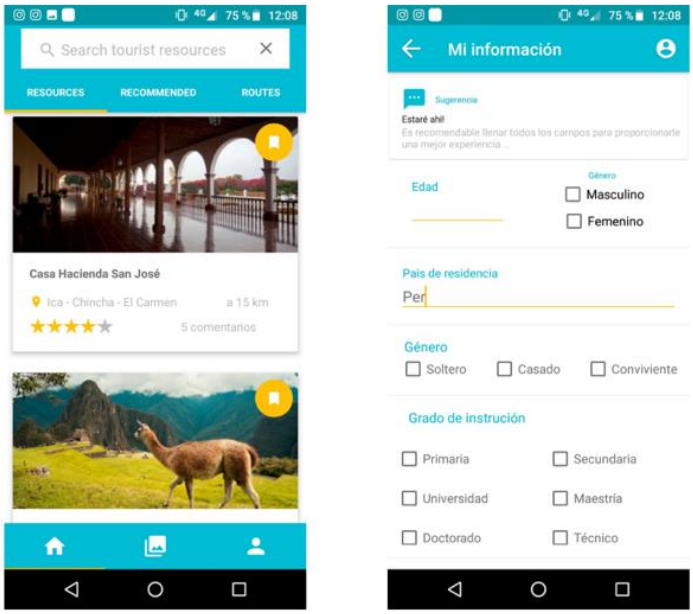


Figura 29. Lista de recursos e información demográfica con Nativescript-vue
Fuente: Elaboración propia, 2019.

4.3.3.2. Desarrollo con Flutter

Como se describió en los capítulos anteriores, este Framework trabaja a base Widgets, y un widget puede componerse de un árbol de widgets como se muestra en la figura 29. También se menciona que los widgets son de dos tipos, en figura se tiene un widget sin estado quien extiende de la clase “StatelessWidget”, esto simplemente porque no habrá cambios sobre un dato en específico.

```
class CardWidget extends StatelessWidget {
  final Resource resource;

  const CardWidget(this.resource);

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.all(10),
      child: InkWell(
        onTap: () {
          Navigator.of(context).push(
            new MaterialPageRoute(
              builder: (BuildContext context) {
                return ResourceDetail(
                  resource: resource,
                ;
              ,
            ,
        child: Column(
          children: <Widget>[
            Container(
              height: 160,
              width: MediaQuery.of(context).size.width,
              child: ClipRRect(
                borderRadius: BorderRadius.only(
                    topLeft: Radius.circular(4), topRight:       Radius.circular(4)),
                child: resource.gallery.length > 0
                    ? CachedNetworkImage(
                        imageUrl: resource.gallery[0],
                        fit: BoxFit
                            .cover, // I thought this would fill up my Container but it doesn't

                    : Text('Sin Imagen'),
              ,
            ,

          ;
        })
);

```

Figura 30. Estructura de un widget sin estado.
Fuente: Elaboración propia, 2019.

La siguiente figura muestra la estructura del proyecto. El desarrollo de la aplicación, tanto de las vistas y consumo de APIs cubren con el objetivo del proyecto.

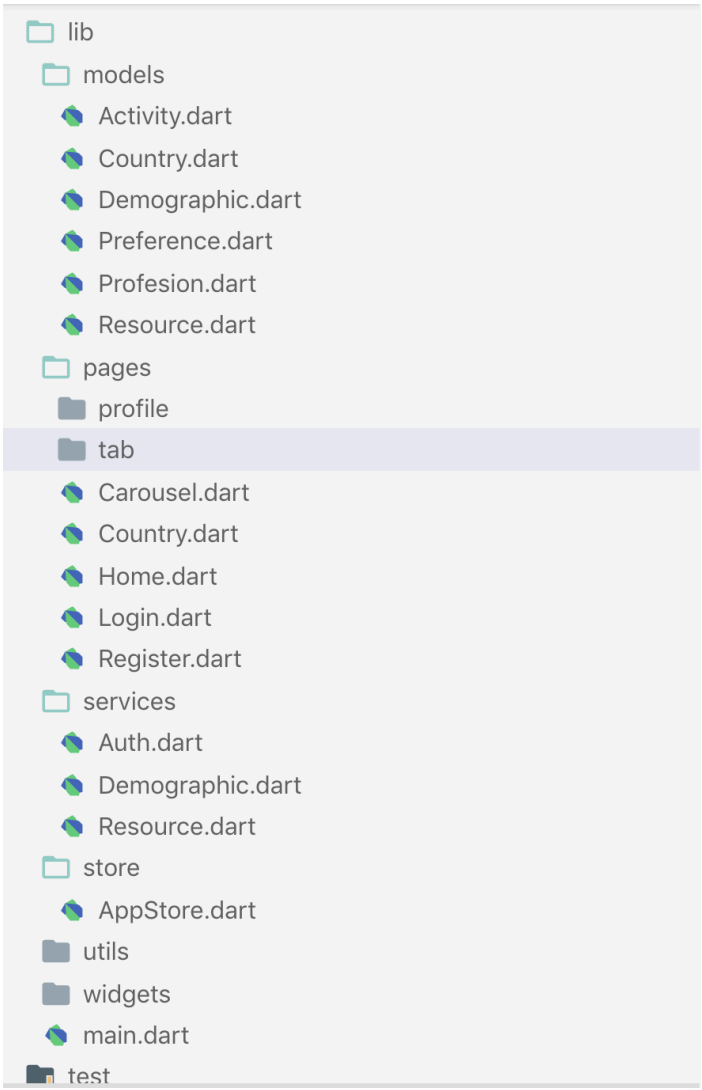


Figura 31. Estructura de un proyecto Flutter
Fuente: Elaboración propia, 2019.

4.4. Pruebas

Para completar los objetivos del proyecto, que son las pruebas de aceptación o pruebas de funcionamiento, en las historias de usuario se definieron criterios de aceptación los cuales incluyen 4 historias de usuario en concreto.

4.4.1. Pruebas de aceptación de la historia USHIS002

Tabla 16. Pruebas de aceptación de la historia número 2

Como turista necesito proveer datos demográficos y preferenciales.		
Criterios de aceptación		**Evaluación**
1	Completar datos demográficos para tenerlos en el sistema.	Satisfactoria
2	Completar lista de preferencias para tenerlos en el sistema	Satisfactoria

Fuente: Elaboración propia, 2019.

4.4.2. Pruebas de aceptación de la historia USHIS003

Tabla 17. Pruebas de aceptación de la historia numero 3

Como turista deseo recibir recomendaciones de los recursos turísticos en base a mis preferencias.		
Criterios de aceptación		**Evaluación**
1	Recomendaciones según datos demográficos para presentar al usuario recomendaciones en base a los datos demográficos de otros usuarios.	Satisfactoria
2	Recomendaciones según calificaciones para presentar a los usuarios recomendaciones similares a los recursos calificados anteriormente.	Satisfactoria

Fuente: Elaboración propia, 2019.

4.4.3. Pruebas de aceptación de la historia USHIS004

Tabla 18. Pruebas de aceptación de la historia numero 4

Como turista deseo realizar calificaciones sobre los recursos turísticos.

Criterios de aceptación		Evaluación
1	Calificar un recurso para proporcionar recomendaciones en base a esas calificaciones.	Satisfactoria

Fuente: Elaboración propia, 2019.

4.4.4. Pruebas de aceptación de la historia USHIS006

Tabla 19. Pruebas de aceptación de la historia número 6

Como turista deseo recibir información detallada de los recursos turísticos.		
Criterios de aceptación		**Evaluación**
1	Información descrita y fotográfica para ver los recursos.	Satisfactoria

Fuente: Elaboración propia, 2019.

CAPÍTULO V. Resultados y discusión

5.1. Resultado del objetivo específico 1

En cumplimiento de este objetivo se obtuvo en primera instancia las historias de usuario más sus especificaciones. Para una mejor comprensión se hizo la creación de los prototipos, esto para un tema de guía de estilo en cuanto al diseño para el programador.

En base a las historias de usuario y los prototipos se realizó la arquitectura tecnológica más los diagramas de clase.

5.1.1. Discusión

La investigación de (SILVIA, 2013), demuestra la importancia que tiene el análisis de los requerimientos antes de realizar un proyecto de este tipo. Tal fue el caso, donde su objetivo principal fue determinar las características deseables en los sistemas de generación de rutas para el turismo. Esta investigación fue un punto de partida para definir los requerimientos para la aplicación móvil. Sin embargo, como el objetivo general es llegar hasta el desarrollo, se vio la necesidad de realizar algunos diseños para un mejor entendimiento o guía para el programador.

5.2. Resultado del objetivo específico 2

Se incorporaron al backend de la aplicación móvil dos enfoques de recomendación, los cuales se basan en KNN; filtrado basado en contenido y filtrado demográfico. Cada una de ellas funcionan en escenarios distintos, sin embargo, pueden unirse ambas para realizar recomendaciones más precisas, pero depende de los escenarios en la que se encuentra un usuario, por ejemplo, si un usuario es nuevo y no tiene calificaciones sobre un recurso, el filtrado basado en contenido no sirve para nada, sin embargo, el usuario por recomendación de la aplicación puede llenar sus datos demográficos y obtener recomendaciones inmediatamente.

Para el cumplimiento de este objetivo también se desarrolló una herramienta de ayuda para la transformación, limpieza de los datos y pruebas correspondientes a la hora de programar los algoritmos, ver anexo C.

5.2.1. Discusión

Existen muchas investigaciones de este tipo, en cada enfoque de los sistemas de recomendación y aplicadas a múltiples áreas, unos intentando solucionar problemas de otros enfoques, pruebas y experimentos con diversos modelos, etc. Sin embargo, muy pocas veces son desplegados o integrados en sistemas funcionales. Solamente a nivel de turismo, de más de 400 aplicaciones solamente para turismo, ninguna de ellas tiene incorporado algoritmos de esta naturaleza (SEGITTUR & Cámara de Comercio de España, 2018). Es por eso que el objetivo no es implementar o diseñar el algoritmo, porque el algoritmo ya está definido, dicho de manera simple, "solo falta programar e integrarlo a un sistema funcional", lo cual se hizo en esta investigación.

Sobre las técnicas y pasos que se emplearon en la implementación del algoritmo no hay mucho que decir ya que son las mismas usadas y recomendadas en las investigaciones de (Kbaier et al., 2017).

Lo único que varía son los escenarios y las características que se consideraron o incorporaron para alimentar el algoritmo, por ejemplo, se consideró la característica de "Color favorito" como dato demográfico de un usuario, pero intentar extraer mucha información de un usuario puede no ser tan fácil, pero si por ejemplo al color le acompañas con un cambio del tema completo de la aplicación puede ser muy atractivo.

El algoritmo KNN es conocido como un algoritmo perezoso, donde la fase de aprendizaje lo realiza en memoria, lo cual genera problemas de rendimiento cuando los datos a procesar son demasiado grandes.

5.3. Resultado del objetivo específico 3

En el desarrollo y al culminar con el desarrollo de la aplicación móvil se realizaron las respectivas pruebas, sobre todo las pruebas funcionales, sujeta a los criterios de aceptación de las historias definidas. Los resultados de dichas pruebas son aceptables y cumplen con lo que se definió, por lo menos para el cumplimiento del objetivo general.

5.4. Discusión

(Cohn, 2009) es claro al decir que las pruebas de aceptación se convierten en pruebas completas que se utilizan para demostrar que la historia se ha codificado de forma correcta y completa. Es por eso que se consideró como un objetivo de la investigación, porque incluye el desarrollo en sí misma, ya que las pruebas de aceptación son definidas al inicio, y demostrados durante y al final del desarrollo, en palabras simple; “si no hay desarrollo, no hay pruebas”.

Para el proyecto no se incluyeron pruebas no funcionales, como es la usabilidad, disponibilidad, rendimiento, etc., porque el objetivo sería muy amplio, sin embargo, fueron consideraros de alguna manera, es por eso el cambio del Framework NativeScrzip a Flutter, ya que un mal rendimiento de la aplicación afecta la experiencia de usuario, más aún cuando la aplicación gestiona mucha información, por más que sea de solo de lectura.

CAPÍTULO VI. Conclusiones y recomendaciones

6.1. Conclusiones

En base a los objetivos, al proceso de ejecución y a los resultados obtenidos de la presente investigación se concluye lo siguiente:

La aplicación de la metodología Kanban para la gestión y control de las tareas apoyada a la ISO 12207, fue lo justo y necesario para esta investigación, por lo flexible que es, ya que dio la facilidad de adaptarlo al proyecto según a las necesidades y limitaciones.

Para el cumplimiento del primer objetivo se definieron los requerimientos respectivos, y estas fueron plasmadas en historias de usuario con los criterios de aceptación, esto, para ver el cumplimiento de las historias tal y como se definió. También se hizo el diseño de los prototipos, de la arquitectura y diagrama de clases de los modelos, esto para ayudar en la comprensión de lo que se quiere en el proceso de desarrollo.

El segundo objetivo fue un tanto especial, porque necesitaba apoyarse de otro método para su cumplimento, se optó por CRISP-DM; método para investigaciones de minería de datos. El cual nos ayudó en la implementación de los sistemas de recomendación donde los algoritmos de aprendizaje automático están presentes, es nuestro caso el KNN. Los datos de los recursos turísticos fueron extraídos de la plataforma de MINCETUR con la técnica llamada “Web Scraping”, exclusivamente con el fin de probar los algoritmos con datos reales.

Respecto al último objetivo específico, considerado la más importante para obtener el resultado esperado. De cierta forma incluye al desarrollo en sí, porque están presentes en la definición de las historias, durante el desarrollo y al hacer las posibles entregas o pruebas funcionales para ver si realmente cumple con lo que se pidió inicialmente, y así fue para esta investigación, se obtuvo lo necesario para cubrir con el objetivo de la presente investigación, según a las prioridades de las historias de usuario.

En general, en cumplimiento del objetivo de la investigación, se tuvo la aplicación móvil funcional que realiza recomendaciones de los recursos turísticos según a las calificaciones

positivas y a los datos demográficos de los usuarios; gracias a la metodología y a las tecnologías empleadas en esta investigación.

6.2. Recomendaciones

Se recomienda realizar proyectos de sistemas de recomendación con enfoques híbridos, utilizando técnicas de conmutación o ponderación, ya que estas aprovechan lo mejor de cada enfoque.

También se recomienda completar lo definido en los requerimientos de esta investigación, incluir algoritmos de optimización para la construcción de rutas personalizadas a partir de las recomendaciones obtenidas.

Se recomienda probar con algoritmos de aprendizaje automático que no son considerados "perezosos", ya que de éstas la fase de entrenamiento es en memoria, y con grandes cantidades de datos el rendimiento se puede ver afectada; y no es recomendable de hecho.

En cuanto a las herramientas para el desarrollo móvil se recomienda realizar proyectos con Flutter, esto por las grandes ventajas que presenta sobre otras herramientas, por el futuro que tiene, y por la empresa que lo respalda.

REFERENCIAS

Ahmad, M. (2016). Exploring Kanban in software engineering. *University of Oulu*. Obtenidode http://jultika.oulu.fi/files/isbn9789526214085.pdf

Anderson, D. J., & Carmichael, A. (2016). Kanban, esencial condensado. En *Lean Kanban, Inc* (First edit). Recuperado de https://leankanban.com/guide/

Anderson, D. J., & Linden-Reed, J. (s. f.). *Getting Started with Kanban for Software Development* (p. 6). p. 6. DZone, Inc.

Banik, R. (2018). *Hands-on recommendation systems with Python : start building powerful and personalized, recommendation engines with Python*. Recuperado de https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788993753

Berzal, F. (2004). *El ciclo de vida de un sistema de información*.

Björkholm, T., & Björkholm, J. (2015). *Kanban in 30 days : modern and efficient organization that delivers results* (U. S. Kadam & V. Phadke, Eds.). Recuperado de https://www.packtpub.com/mapt/book/business/9781783000906

Branstein, M., & Branstein, N. (2018). *The NativeScript Book building mobile apps with skills you already have*. The Brosteins.

Cantone, D. (2006). *Implementacion y debugging : la biblia de la programacion*. Recuperado de https://ingsw.pbworks.com/f/Ciclo+de+Vida+del+Software.pdf

Charkaoui, S., Adraoui, Z., & Benlahmar, E. H. (2014). Cross-platform mobile development approaches. *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, 188-191. https://doi.org/10.1109/CIST.2014.7016616

CIANCIACTIVA, & CONCYTEC. *Bases Integradas PROYECTOS DE INVESTIGACIÓN BÁSICA Y APLICADA*. , Pub. L. No. 2017-02 (2017).

Cohn, M. (2009). *User Stories Applied for Agile Software Development*. Recuperado de

www.wowebook.com

D'Angelo, P., & Rodríguez, M. (2015). *Aplicación móvil para información y ubicación del turista perdido* (San Martin de Porres). Recuperado de http://www.repositorioacademico.usmp.edu.pe/bitstream/usmp/1449/1/rodriguez_dm.pdf

Dangeti, P. (2017). *Statistics For Machine Learning*. Recuperado de https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788295758

Dasgupta, N. (2018). *Practical big data analytics : hands-on techniques to implement enterprise analytics and machine learning using Hadoop, Spark, NoSQL and R*. Recuperado de https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781783554393

DGIETA, & Mincetur. (2016). *Medición Económica del Turismo*. Recuperado de www.mincetur.gob.pe

El-Kassas, W. S., Abdullah, B. A., Yousef, A. H., & Wahba, A. M. (2017). Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal*, *8*(2), 163-190. https://doi.org/10.1016/j.asej.2015.08.004

Facebook Inc. (2016). React - A declarative, efficient, and flexible JavaScript library for building user interfaces. *a Javascript Library for Building User Interfaces*, p. 1. Recuperado de https://reactjs.org/

Facebook Inc. (2017). A framework for building native apps using React. Recuperado 30 de mayo de 2018, de React Native website: https://facebook.github.io/react-native/

Harryho. (2017). Angular vs React vs Vue - codeburst. Recuperado 22 de septiembre de 2018, de https://codeburst.io/angular-vs-react-vs-vue-f470f5b74bf6

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems:

Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261-273. https://doi.org/10.1016/J.EIJ.2015.06.005

Jiménez Cordero, M. A., & García Coello, E. A. (2015). *Aplicación móvil celular para incentivar el turismo urbano en Guayaquil* (UNIVERSIDAD POLITÉCNICA SALESIANA SEDE GUAYAQUIL). Recuperado de https://dspace.ups.edu.ec/bitstream/123456789/10323/1/UPS-GT001230.pdf

Kanber, B. (2018). *Hands-on Machine Learning with JavaScript Solve complex computational web problems using machine learning*. Recuperado de https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788998246

Kbaier, M. E. B. H., Masri, H., & Krichen, S. (2017). A Personalized Hybrid Tourism Recommender System. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 244-250. https://doi.org/10.1109/AICCSA.2017.12

Khan, Z. (2014). *Scrumban-Adaptive Agile Development Process: Using scrumban to improve software development process* (Helsinki Metropolia University of Applied Sciences). Recuperado de https://www.theseus.fi/bitstream/handle/10024/77014/Khan_Zahoor.pdf

Latif, M., Lakhrissi, Y., Nfaoui, E. H., & Es-Sbai, N. (2016). Cross platform approach for mobile application development: A survey. *2016 International Conference on Information Technology for Organizations Development, IT4OD 2016*, 1-5. https://doi.org/10.1109/IT4OD.2016.7479278

Li, P. (2016). *JIRA 7 essentials : explore the great features of the all-new JIRA 7 to manage projects and effectively handle bugs and software issues* (Fourth). Recuperado de https://www.packtpub.com/mapt/book/application_development/9781786462510

Li, P. (2018). *Jira software essentials : plan, track, and release great applications with Jira software* (Second edi). Recuperado de

https://www.packtpub.com/mapt/book/application_development/9781788833516

López de Ávila, A., Lancis, E., García, S., Alcantud, A., García, B., & Muñoz, N. (2015). Informe destinos turísticos inteligentes: construyendo el futuro. En *segittur y thinktur*. Recuperado de https://goo.gl/CysEL7

Love, C. (2018). *PROGRESSIVE WEB APPLICATION DEVELOPMENT BY EXAMPLE : develop fast, reliable and engaging user... experiences with progressive web applications.* Recuperado de https://www.packtpub.com/mapt/book/application_development/9781787125421

Mainkar, P., & Giordano, S. (2019). *Google Flutter Mobile Development Quick Start Guide: Get up and running with iOS and Android mobile app development*. Recuperado de https://www.packtpub.com/application-development/google-flutter-mobile-development-quick-start-guide

Mehlhorn, N. (2017). *Modern Cross -Platform Development for Mobile Applications* (Hochschule Düsseldorf University of Applied Sciences). Recuperado de http://fhdd.opus.hbz-nrw.de/volltexte/2017/1127/pdf/Modern_Cross_Platform_Development_for_Mobile_Applications.pdf

Mohamed, L., & Abdali, A. (2017). Decision Framework for Mobile Development Methods. *IJACSA) International Journal of Advanced Computer Science and Applications*, *8*(2). Recuperado de www.ijacsa.thesai.org

Nagesh, A., & Caicedo, C. E. (2012). Cross-Platform Mobile Application Development. *Conference: ITERA*. Recuperado de https://www.researchgate.net/publication/263416908_Cross-Platform_Mobile_Application_Development

Natingga, D. (2017). *Data science algorithms in a week: data analysis, machine learning, and more.* Recuperado de https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/97817872

84586

NativeScript community. (s. f.). Native mobile apps with Angular, TypeScript, JavaScript - NativeScript. Recuperado 22 de septiembre de 2018, de Telerik website: https://www.nativescript.org/

Nitze, A., & Schmietendorf, A. (2013). Cross-Platform Mobile Application Development. *Conference: User Conference for Software Quality, Test and Innovation 2013, At Graz, AT*. Recuperado de https://www.researchgate.net/publication/259620153_Cross-Platform_Mobile_Application_Development

Olusola Olajide, A., Omotayo Abiodun, A., Adebola Okunola, O., Gabriel Omomule, T., & Michael Orimoloye, S. (2018). Performance Evaluation of Native and Hybrid Android Applications. *Communications on Applied Electronics*, *7*(16). https://doi.org/10.5120/cae2018652701

Organización Mundial del Turismo. (2016). *Alianza entre turismo y cultura en el Perú – Modelos de colaboración entre turismo, cultura y comunidad*. Recuperado de http://www.e-unwto.org/doi/10.18111/9789284417575

PromPerú. (2017a). *TURISMO EN CIFRAS Perfil del Turista Extranjero 2016*. Recuperado de http://media.peru.info/IMPP/2014/Perfil-Turista-Extranjero/Perfil-del-Turista-Extranjero-2014.pdf

PromPerú. (2017b). *TURISMO EN CIFRAS Perfil del Vacacionista Nacional 2016*. Lima.

PromPerú. (2018a). Aplicacion de Peru - Apps de Peru en tu Smartphone | Peru Travel. Recuperado 29 de agosto de 2018, de https://www.peru.travel/es-es/comunidad/peru-travel-apps.aspx

PromPerú. (2018b). *TURISMO EN CIFRAS Perfil del Turista Extranjero 2017*. Recuperado de https://www.promperu.gob.pe/TurismoIN/sitio/PerfTuristaExt

PromPerú. (2018c). *TURISMO EN CIFRAS Perfil del Vacacionista Nacional 2017*. Recuperado de https://www.promperu.gob.pe/TurismoIN/sitio/PerfVacacionistaNac

Ravulavaru, A. (2017). *Learning Ionic: Build hybrid mobile applications with HTML5, CSSS, and Angular*. Recuperado de https://www.packtpub.com/mapt/book/web_development/9781786466051

Rivero Cuesta, A. (s. f.). *Apuntes de Introducción a la Ingeniería de Software*. Recuperado de https://www.scribd.com/document/259969482/Capitulo-2-El-Ciclo-de-Vida-del-Software-pdf

Safoury, L., & Salah, A. (2013). Exploiting User Demographic Attributes for Solving Cold-Start Problem in Recommender System. *Lecture Notes on Software Engineering, Vol. 1, No. 3, August 2013(LNSE), 1*, 303-307. https://doi.org/10.7763/LNSE.2013.V1.66

SEGITTUR, & Cámara de Comercio de España. (2018). *APPs Turísticas 2018*. Recuperado de https://goo.gl/EEtvFX

SILVIA, E. C. (2013). GENERACIÓN DE RUTAS. COMPARATIVA DE SISTEMAS DE DESTINOS TURÍSTICOS. CARACTERÍSTICAS NECESARIAS (UNIVERSIDAD DE MÁLAGA). Recuperado de http://www.riuma.uma.es/xmlui/bitstream/handle/10630/6920/TFG OLGA GOMEZ JIMENEZ.pdf?sequence=1

Sp, S. (2013). GENERACIÓN DE RUTAS. COMPARATIVA DE SISTEMAS DE DESTINOS TURÍSTICOS. CARACTERÍSTICAS NECESARIAS (UNIVERSIDAD DE MÁLAGA). Recuperado de https://goo.gl/oLgBiU

Warén, J. (2016). *Cross-platform mobile software development with React Native* (Haaga-Helia University of Applied Sciences). Recuperado de https://www.theseus.fi/bitstream/handle/10024/119913/janne_waren_cross-platform_mobile_software_development_with_react_native.pdf?sequence=1

Wu, W. (2018). *React Native vs Flutter, cross-platform mobile application frameworks* (Metropolia University of Applied Sciences). Recuperado de https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1

ANEXOS

Anexo A. Historias de usuario y criterios de aceptación

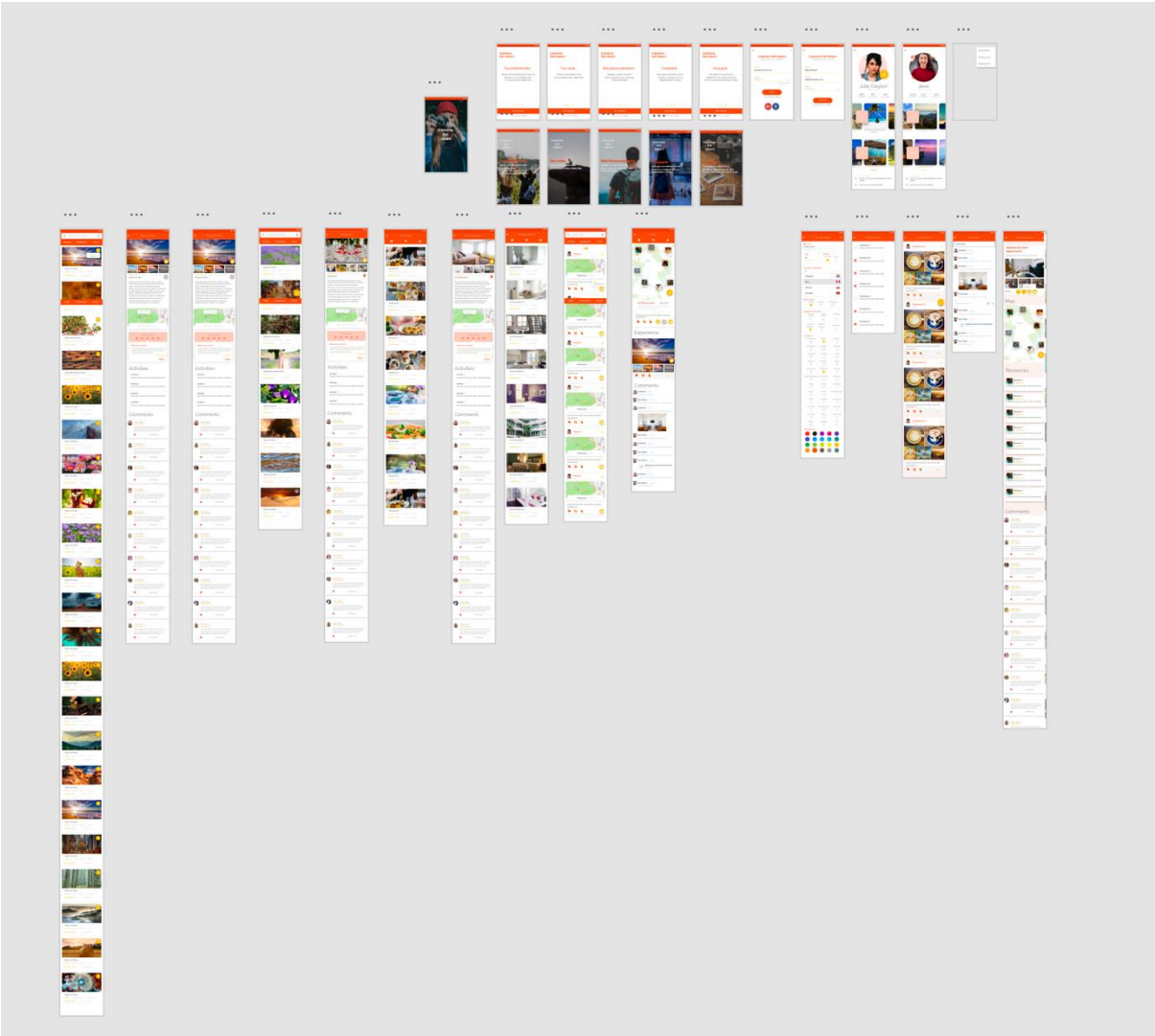
ENUNCIADO DE LA HISTORIA						CRITERIOS DE ACEPTACION				
ID	ROL	CARACTERISTICA / FUNCIONALIDAD	OBJETIVO	PRIORIDAD	COMPLEJIDAD	ESCENARIOS	TITULO	CONTEXTO	EVENTO	COMPORTAMIENTO ESPERADO
USHIS001	Turista	Puede acceder mediante redes sociales	Agilizar y facilitar el acceso	Baja	Muy alta	1	Acceso con una cuenta de Google	Si se cuenta con una cuenta de Google	Ninguno	Usuario registrado
						2	Acceso con una cuenta de Facebook	Si se cuenta con una cuenta de Facebook	Ninguno	Usuario registrado
						3	Acceso con correo electrónico previamente registrado	Si no se cuenta con una cuenta de Facebook y Google	Ninguno	Usuario registrado
USHIS002	Turista	Debe completar datos demográficos y preferencias	Brindar mejor experiencia en las recomendaciones	Alta	Media	1	Completar datos demográficos	Ninguno	Ninguno	Tener los datos demográficos en el sistema
						2	Completar lista de preferencias	Ninguno	Ninguno	Tener las preferencias en el sistema
USHIS003	Turista	Debe recibir recomendaciones de los recursos turísticos	Deducir el tiempo de búsqueda	Muy alta	Muy alta	1	Recomendaciones según datos demográficos	Cuando el usuario activo es nuevo en el sistema y además se tiene datos de otros usuarios en el sistema	Generar lista de recomendaciones	Presentar al usuario recomendaciones en base a sus datos demográficos y de otros usuarios.
						2	Recomendaciones según calificaciones y preferencias	Cuando el usuario tenga calificaciones anteriores	Generar lista de recomendaciones	Tener recomendaciones similares a los recursos calificados anteriormente

						3	Recomendaciones en base a usuario vecinos	Cuando el usuario tenga calificaciones anteriores y además exista calificaciones de otros usuarios	Generar lista de recomendaciones	Obtener recomendaciones en base a usuarios similares al activo
						4	Recomendaciones sujetas a restricciones	Cuando el usuario ingresa alguna restricción o preferencia en particular	Generar lista de recomendaciones	Tener recomendaciones sujetas a las restricciones del usuario
						5	Recomendación combinada	Cuando se cumple todas las anteriores	Generar lista de recomendaciones	Tener las mejores recomendaciones de cada enfoque
USHIS004	Turista	Debe realizar calificaciones a los recursos	Obtener recomendaciones en base a las calificaciones y ayudar a otros usuarios	Muy alta	Media	1	Calificar un recurso	Al haber visitado el recurso	Actualiza el promedio de calificación del recurso	Guardar calificaciones
USHIS005	Turista	Debe generar una ruta turística	Optimizar el viaje	Baja	Muy alta	1	Construir en base a las recomendaciones obtenidas	Cuando el usuario este satisfecho con las recomendaciones	Construir ruta	Ruta construida
						2	Descartar recursos de las recomendaciones obtenidas y construir la ruta	Cuando el usuario no quiera visitar uno o más recursos	Construir ruta	Ruta construida
						3	Agregar recursos a la lista de recomendaciones y construir la ruta	Cuando el usuario desea visitar uno o más recursos que no están en la lista de recomendaciones.	Construir ruta	Ruta construida

						4	Visitar todos los recursos sin repetir	Ninguno	Ninguno	Ruta optima
USHIS006	Turista	Debe recibir información detallada de los recursos	Mantener informado y seleccionar recursos	Muy alta	Alta	1	Información del clima	Ninguno	Presentar advertencias en situaciones negativas	Información detallada
						2	Información descrita y fotográfica	Ninguno	Ninguno	Información detallada
USHIS007	Turista	Compartir la experiencia vivida	Interactuar con otros usuarios	Baja	Muy alta	1	Comentar sobre un recurso	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
						2	Comentar sobre una ruta	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
						3	Comentar sobre un servicio	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
						4	Los comentarios incluyen material visual (Fotografías y emojis)	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
						5	Compartir o Recomendar rutas o recursos	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
						6	Compartir o Recomendar rutas o recursos por Facebook y WhatsApp	Si el usuario lo requiera	Ninguno	crear interacción con otros usuarios
USHIS008	Turista	Tener la información en varios idiomas	Mejor comprensión	Baja	Alta	1	Información en español	Si el móvil este en este idioma	Cambio de idioma	Información en el idioma según corresponda

						2	Información en ingles	Si el móvil este en este idioma, excepto en español o portugués	Cambio de idioma	Información en el idioma según corresponda
						3	Información en portugués	Si el móvil este en este idioma	Cambio de idioma	Información en el idioma según corresponda
USHIS009	Turista	Tener la información sin conexión a internet	Disponibilidad de la información	Baja	Alta	1	Funcionamiento offline	Si no te cuneta con conexión a internet	Activar modo offline	Información que el usuario necesite
USHIS010	Turista	Obtener información de servicios cercanos	Facilitar el acceso a los servicios	Baja	Alta	1	Transporte	Si el usuario lo requiera	Ninguno	Información de servicios cercanos
						2	Alimentación	Si el usuario lo requiera	Ninguno	Información de servicios cercanos
						3	Alojamiento	Si el usuario lo requiera	Ninguno	Información de servicios cercanos
USHIS011	Emprendedor o dueño del recurso	Recibir notificaciones	Mantener informado	Baja	Muy alta	1	Posibles visitas	Cuando un recurso o emprendimiento es incluido en una ruta.	Envía notificación	Usuario notificado
						2	Comentarios de los recursos o emprendimientos	Cuando un turista realiza un cometario	Envía notificación	Usuario notificado
						3	Calificaciones a los recursos o emprendimientos	Cuando un turista realiza una calificación	Envía notificación	Usuario notificado
USHIS012	Emprendedor o dueño del recurso	Actualizar información del recurso o emprendimiento	Mantener información actualizada	Baja	Alta	1	Información descrita	Ninguno	Ninguno	Información detallada
						2	Subir fotografías	Ninguno	Ninguno	Fotografías

Anexo B. Prototipo de la aplicación móvil



Para ver en su estado original y modo desarrollador en: xd.adobe.com.

Anexo C. Código fuente del sistema de recomendación del filtrado basado en contenido integrado al Backend

```
import {repository, Filter} from '@loopback/repository';
import {
  ResourceRepository,
  RatingRepository,
  UserRepository,
  PreferenceRepository,
} from '../repositories';
import {
  get,
  param,
  getFilterSchemaFor,
  post,
  requestBody,
} from '@loopback/rest';
import {Resource, Rating, Preference} from '../models';
import {authenticate} from '@loopback/authentication';
import {inject} from '@loopback/core';

// Uncomment these imports to begin using these cool features!

// import {inject} from '@loopback/context';
interface Employee {
  name: string;
}
export class ContentBasedFilteringController {
  constructor(
    @repository(ResourceRepository)
    public resourceRepository: ResourceRepository,
    @repository(RatingRepository)
    public ratingRepository: RatingRepository,
    @repository(UserRepository)
    public userRepository: UserRepository,
    @repository(PreferenceRepository)
    public preferenceRepository: PreferenceRepository,
  ) {}
  @authenticate('jwt')
  @get('/content-based-filtering', {
    responses: {
      '200': {
        description: 'Array of Resource model instances',
        content: {
          'application/json': {
            schema: {type: 'array', items: {'x-ts-type': Resource}},
          },
        },
      },
    },
  })
  async find(
    @inject('authentication.currentUser') currentUser: any,
    @param.query.object('filter', getFilterSchemaFor(Resource)) filter?: Filter,
  ): Promise<Resource[]> {
```

```
/**
    *!DATOS QUE SE NECESITAN
    **Recursos
    **User
    **Calificaciones
    *** Preferencias(actividades de cada recurso)
    *!RESTRICCIONES
    **Considerar mínimo de actividades por recurso 2
    */

   //!ALGORITMO
   /**
    * * PASOS PARA OBTENER RESULTADOS DE LA RECOMENDACIÓN BASADO EN CONTENIDO
    * - Identificar actividades no calificadas por el usuario activo.
    * - Para cada actividad no calificada, calcule su similitud con toda la
    * actividad calificada por el usuario activo utilizando la fórmula de
    * similitud de distancia euclidiana.
    * - Encuentre para cada actividad no calificada su actividad más cercana y
    *  pronostique su tasa (la tasa de la actividad calificada similar más alta).
    */

   //! PARA ESTE CASO UNA ACTIVIDAD REPRESENTA UN RECURSO

   /**
    * * PASO 1:
    **! !Identificar actividades no calificadas por el usuario activo.

    */

   /**
    * Obtener datos de los recursos solo con algunos campos
    */
   const resources: Resource[] = await this.resourceRepository.find({
     //limit: 50,
     //fields: {_id: true, department: true, name: true, activities: true},
     //include: [{relation: 'ratings', scope: {fields: {ratings: true}}}],
     //order: ['name DESC'],
   });

   /**
    * Obtengo todas las preferencias para ayudarme a construir una matriz de
    * actividades de los recursos
    */
   const preferences: Preference[] = await this.preferenceRepository.find({
     fields: {activity_code: true},
     order: ['activity_code ASC'],
   });
   /**
    * Obtengo los calificaciones del usuario activo
    */
```

```
const ratings: Rating[] = await this.userRepository
  .ratings(currentUser.user)
  .find({}, {strictObjectIDCoercion: true});

/**
 * Transforma las actividades de los recursos en una matriz para que
 * tengan una misma longitud
 */

for (const key in resources) {
  //recuperando las actividades de cada recurso
  const activities: any = resources[key].activities;

  const temp: any[] = [];
  for (const key in preferences) {
    // recorriendo cada preferencia
    const element = JSON.parse(JSON.stringify(preferences[key]));
    //Determinando si una actividad esta o no en un recurso
    const findPreference = await activities.filter(
      (x: any) => x == element.activity_code,
    );
    if (findPreference.length) {
      temp.push(1);
    } else {
      temp.push(0);
    }
  }
  resources[key].activities = temp;
}

/**
 * Separo recursos calificados y no calificados por el usuario activo
 */

const sinCalificar: Resource[] = [];
const calificadas: Resource[] = [];
for (let index = 0; index < resources.length; index++) {
  const findSincalificar: Rating[] = ratings.filter(
    x => x.resourceId === resources[index]._id,
  );

  /**
   * Elimino campos que no se necesitan
   */
```

```
delete resources[index].category;
  delete resources[index].department;
  delete resources[index].district;
  delete resources[index].province;
  delete resources[index].longitude;
  delete resources[index].latitude;
  delete resources[index].type_category;
  delete resources[index].sub_type_category;
  delete resources[index].file;
  delete resources[index].description;
  delete resources[index].particularities;
  delete resources[index].observations;
  delete resources[index].state;
  delete resources[index].gallery;

  /** TODO:
   * !Descartar calificaciones calificadas negativamente;
   */
  if (findSincalificar.length) {
    if (findSincalificar[0].rating >= 3) {
      calificadas.push(resources[index]);
    }
  } else {
    sinCalificar.push(resources[index]);
  }
}

/**
 * * PASO 2:
 ** - Para cada actividad no calificada, calcule su similitud con toda
 ** la actividad calificada por el usuario activo utilizando la fórmula de
 **  similitud de distancia euclidiana.
 */

const distance = new Distances();
//toda distancia calculado se devuelve en una sola
const allDistances: Object[] = [];
for (let sin_c = 0; sin_c < sinCalificar.length; sin_c++) {
  const sin_calificar: any = sinCalificar[sin_c];

  for (let cali = 0; cali < calificadas.length; cali++) {
    const calificada: any = calificadas[cali];
    const calculate = distance.euclidean(
      sin_calificar.activities,
      calificada.activities,
    );
    allDistances.push({
      distance: calculate,
      resourceId: sin_calificar._id,
      resourceName: sin_calificar.name,
    });
  }
}
```

```
 const algoritmos = new Algoritmos();

    //PASO 3
    //! * - Encuentre para cada actividad no calificada su actividad más
    //!cercana y pronostique su tasa(la tasa de la actividad calificada similar más alta).

    const knn: any[] = await algoritmos.knn(allDistances, 15);

    const result: Resource[] = [];
    for (const key in knn) {
      result[key] = await this.resourceRepository.findById(knn[key]);
    }

    return result;
  }
}

class Distances {
  constructor() {}

  euclidean(x: number[], y: number[]) {
    if (x.length !== y.length) {
      return 'x has a different size than y';
    }
    let accumulator: number = 0;
    for (let item = 0; item < x.length; item++) {
      accumulator = accumulator + Math.pow(x[item] - y[item], 2);
    }
    return Math.sqrt(accumulator);
  }
}

class Algoritmos {
  constructor() {}
  knn(list: any[], k: number) {
    const list1 = list.sort((x, y) => x.distance - y.distance);
    const temp: any = {};

    for (let index = 0; Object.keys(temp).length < k; index++) {
      if (!temp[list1[index].resourceId]) {
        temp[list1[index].resourceId] = [];
        temp[list1[index].resourceId].push(list1[index]);
      } else {
        temp[list1[index].resourceId].push(list1[index]);
      }
    }
    const result: string[] = [];
    for (const iterator in temp) {
      result.push(iterator);
    }

    return result;
  }
}
```

Anexo D. Codifo fuente del fitrado demografico en contenido integrado al Backend

```
import {
  ResourceRepository,
  RatingRepository,
  UserRepository,
  PreferenceRepository,
  DemographicRepository,
} from '../repositories';
import {repository} from '@loopback/repository';
import {get, param, getFilterSchemaFor, HttpErrors} from '@loopback/rest';
import {inject} from '@loopback/core';
import {User, Demographic, Rating, Resource} from '../models';
import {Filter} from 'loopback-datasource-juggler';
import {authenticate, AuthenticationBindings} from '@loopback/authentication';

// Uncomment these imports to begin using these cool features!

// import {inject} from '@loopback/context';

export class DemographicBasedFilteringController {
  constructor(
    @repository(ResourceRepository)
    public resourceRepository: ResourceRepository,
    @repository(RatingRepository)
    public ratingRepository: RatingRepository,
    @repository(UserRepository)
    public userRepository: UserRepository,
    @repository(PreferenceRepository)
    public preferenceRepository: PreferenceRepository,
    @repository(DemographicRepository)
    public demographicRepository: DemographicRepository,
  ) {}

  /**
   *!DATOS QUE SE NECESITAN
   **Recursos
   **Users: Datos Demográficos
   **Calificaciones
   */
  /**
   * !RESTRICCIONES
   * *Considerar minimo de actividades por recurso 5, imágenes 5.
   * !Los campos obligatorios son:
   * - gender
   * - age
   * - color
   * - country
   * - marital_status
   * - degree_instruction
   * !de no cumplir con alguno de ellos, no se considera para el calculo de similitud
   */
```

```
//!ALGORITMO
/**
 * * PASOS PARA OBTENER RESULTADOS DE LA RECOMENDACIÓN BASADO EN CONTENIDO
 * - Limpieza y transformación  de los datos del usuario
 * - CALCULAR LA SIMILITUD DEL USUARIO ACTIVO CON TODOS LOS OTROS USUARIOS DEL SISTEMA.
 * - Obtener k vecinos mas cercanos en cuando a la similitud obtenida.
 * - De los k vecinos obtener los recursos calificados positivamente de mayo o igual a 4.
 * - Calcular la frecuencia por recurso
 * - Obtener k recursos con frecuencias mas altas.
 */

@get('/demographic-based-filtering', {
  responses: {
    '200': {
      description: 'Array of Resource model instances',
      content: {
        'application/json': {
          schema: {type: 'array', items: {'x-ts-type': User}},
        },
      },
    },
  },
})
@authenticate('jwt')
async find(
  @inject(AuthenticationBindings.CURRENT_USER)
  currentUser: any,
  @param.query.object('filter', getFilterSchemaFor(User)) filter?: Filter,
): Promise<Resource[]> {
  //!Paso 1
  //    * - Limpieza y transformación  de los datos de todo los usuarios

  const currentUserD: any = await this.demographicRepository.findOne(
    {
      where: {userId: currentUser.user},
    },
    {strictObjectIDCoercion: true},
  );
  if (
    !currentUserD.country ||
    !currentUserD.color ||
    !currentUserD.degree_instruction ||
    !currentUserD.gender ||
    !currentUserD.marital_status ||
    (currentUserD.age < 10 || currentUserD.age > 75)
  ) {
    throw new HttpErrors.NotAcceptable(
      `No tienes información demográfica, para realizar esta
      acción asegúrese completar dicha información el tu perfil.`,
    );
  }

  //Obtener lista de usuarios
  const users: User[] = await this.userRepository.find({
    fields: {demographic: true, ratings: true, _id: true},
  });
```

```
// *Obtener datos demográficos de los usuarios
let demographics: Demographic[] = [];
for (const key in users) {
  const temp: any = await this.demographicRepository.findOne(
    {
      where: {userId: users[key]._id},
      fields: {
        color: true,
        age_range: true,
        userId: true,
        age: true,
        gender: true,
        marital_status: true,
        degree_instruction: true,
        country: true,
      },
    },
    {strictObjectIDCoercion: true},
  );
  demographics.push(temp);
}

/**
 * !Cumplir con las restricciones
 */

demographics = demographics.filter(
  x =>
    x.country &&
    x.color &&
    x.degree_instruction &&
    x.gender &&
    x.marital_status &&
    x.age,
);

/**
 * !Transformaciones
 * - Poner las edades en tres rangos
 *  - edad >=10 && edad <= 19 = teenager => adolescente
 *  - edad >=20 && edad <= 39 = young => joven
 *  - edad >=40 && edad <= 55 = adult => adulto
 *  - edad >=56 && edad <= 65 = old => Viejo
 *
 * ! Las edades fuera de rango no son consideradas
 */

let new_demographics: Demographic[] = [];
```

```
for (const key in demographics) {
  let age = '';

  const element: any = demographics[key];
  if (element.age >= 10 && element.age <= 19) {
    age = 'teenager';
  } else if (element.age >= 20 && element.age <= 39) {
    age = 'young';
  } else if (element.age >= 40 && element.age <= 55) {
    age = 'adult';
  } else if (element.age > 56 && element.age <= 65) {
    age = 'old';
  }
  demographics[key].age_range = age;

  if (element.age >= 10 && element.age <= 65 && element.userId) {
    new_demographics.push(demographics[key]);
    console.log(new_demographics);
  }
}
const userActive: Demographic[] = await new_demographics.filter(
  x => x.userId === currentUser.user,
);
new_demographics = new_demographics.filter(
  x => x.userId != currentUser.user,
);
/**
 * !PASO 2
 ** CALCULAR LA SIMILITUD DEL USUARIO ACTIVO CON TODOS LOS OTROS
 ** USUARIOS DEL SISTEMA.
 */

const distance = new Distances();

const similarity: Object[] = distance.similarity(
  userActive[0],
  new_demographics,
);

/**
 * !PASO 3
 * * Obtener k vecinos mas cercanos en cuando a la similitud obtenida.
 */
const knn: any[] = distance.knn(similarity, 3);

/**
 * !PASO 4
 ** De los k vecinos obtener los recursos calificados positivamente de
 ** mayo o igual a 4, excepto calificado.
 */
```

```
    let ratings: Rating[] = [];
     for (let index = 0; index < knn.length; index++) {
       const temp: Rating[] = await this.userRepository
         .ratings(knn[index].userId)
         .find({where: {rating: {gte: 4}}}, {strictObjectIDCoercion: true});

       for (let index1 = 0; index1 < temp.length; index1++) {
         ratings.push(temp[index1]);
       }
     }

     //Obtener calificaciones del usuario activo;

     const ratingUser: Rating[] = await this.userRepository
       .ratings(userActive[0].userId)
       .find({}, {strictObjectIDCoercion: true});

     for (const rating of ratingUser) {
       ratings = ratings.filter(x => x.resourceId != rating.resourceId);
     }

     /**
      * !PASO 5
      ** Calcular la frecuencia por recurso
      */

     const frecuencia = distance.frecuencia(ratings);
     /**
      * !PASO 6
      ** Obtener k recursos con frecuencias mas altas.
      */
     const result: Resource[] = [];
     const k_resources: any = frecuencia.slice(0, 4);
     for (let index = 0; index < k_resources.length; index++) {
       const temp: Resource = await this.resourceRepository.findById(
         k_resources[index].resourceId,
       );
       result.push(temp);
     }

     return result;
   }
}
```

```
class Distances {
  constructor() {}

  intersection = (x: any, y: any) => {
    const accumulator = [];

    for (const index of Object.keys(JSON.parse(JSON.stringify(x)))) {
      if (x[index] != null && y[index] != null) {
        if (x[index] === y[index]) {
          accumulator.push(true);
        }
      }
    }

    return accumulator.length;
  };

  // NOTE: Jaccard/Tanimoto in https://es.planetcalc.com/1664/
  tanimoto = (a: number, b: number, c: number) => c / (a + b - c);

  //NOTE: calcule similaty with tanimoto coefficient
  calSimilitud = (x: Demographic, y: Demographic) =>
    this.tanimoto(
      Object.keys(x).length,
      Object.keys(y).length,
      this.intersection(x, y),
    );

  //NOTE: CALCULATE THE SIMILITUD WITH ALL USERS
  similarity = (user: Demographic, data: Demographic[]) => {
    const simil = [];
    for (const key of data) {
      if (user != key) {
        simil.push({
          similitud: this.calSimilitud(user, key),
          userId: key.userId,
        });
      }
    }
    simil.sort((a, b) => b.similitud - a.similitud);
    return simil;
  };
```

```
  //NOTE: KNN OR K-ITEMS
  knn = (x: Object[], k: number) => {
    const knn: Object[] = [];
    for (let index = 0; index < k; index++) {
      knn.push(x[index]);
    }

    return knn;
  };

  frecuencia = (ratings: Rating[]) => {
    let result: any = {};
    const cantidad: any = [];

    for (const key of ratings) {
      if (!result[key.resourceId]) {
        result[key.resourceId] = [];
        result[key.resourceId].push(key);
      } else {
        result[key.resourceId].push(key);
      }
    }

    for (const iterator of Object.keys(result)) {
      cantidad.push({
        frecuencia: result[iterator].length,
        resourceId: iterator,
      });
    }
    return cantidad.sort((a: any, b: any) => b.frecuencia - a.frecuencia);
  };
}
```