Pontificia Universidad Católica del Ecuador

Facultad De Ingeniería

Escuela de Sistemas

TEMA:

“Desarrollo e implementación de un aplicativo móvil para movilidad publica ¿En qué bus vas vos ve?”

AUTOR:

Mateo Orellana

TRABAJO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

QUITO, 2023

DEDICATORIA

AGRADECIMIENTO

RESUMEN

El presente trabajo de tesis aborda la creciente necesidad de simplificar la elección de rutas de transporte público en el Distrito Metropolitano de Quito (DMQ), dada la importancia que han adquirido las aplicaciones móviles en la vida cotidiana. Es por eso por lo que se desarrolló esta aplicación la cual obtiene la mejor ruta de bus y la presenta al usuario de manera clara y concisa.

El objetivo general de este proyecto es desarrollar un aplicativo móvil llamado "¿En qué bus vas vos ve?" que proporcione información clara y detallada sobre frecuencias de transporte público, paradas y precios, para facilitar la toma de decisiones de los usuarios. Para ello, se plantean objetivos específicos, como definir los procesos y actividades de la aplicación, seleccionar las herramientas de desarrollo y diseñar e implementar el aplicativo móvil.

El alcance del trabajo se limita al desarrollo e implementación del aplicativo móvil, y se entregará un informe de resultados sobre la implementación de la aplicación "¿En qué bus vas vos ve?" que busca mejorar la experiencia de los usuarios al utilizar el transporte público en el DMQ.

En conclusión, esta tesis ofrece una propuesta innovadora y útil para simplificar la toma de transporte público en la ciudad de Quito, mediante una aplicación móvil intuitiva y actualizada, que busca satisfacer las necesidades de la comunidad y mejorar la calidad del servicio.

Palabras Clave: Aplicativo móvil, Movilidad en el DMQ, Android, Desarrollo de Software, Kotlin.

ABSTRACT

The present thesis work addresses the growing need to simplify the choice of public transportation routes in the Metropolitan District of Quito (DMQ), given the importance that mobile applications

have acquired in everyday life. That's why this application was developed, which obtains the best bus route and presents it to the user in a clear and concise manner.

The overall objective of this project is to develop a mobile application called "Which bus are you taking?" that provides clear and detailed information on public transportation frequencies, stops, and prices, to facilitate users' decision-making. To achieve this, specific objectives are proposed, such as defining the processes and activities of the application, selecting the development tools, and designing and implementing the mobile application.

The scope of the work is limited to the development and implementation of the mobile application, and a report of results will be delivered regarding the implementation of the "Which bus are you taking?" application, which aims to improve the users' experience when using public transportation in the DMQ.

In conclusion, this thesis offers an innovative and useful proposal to simplify the use of public transportation in the city of Quito through an intuitive and up-to-date mobile application that seeks to meet the community's needs and improve the quality of the service.

Keywords: Mobile application, Mobility in the DMQ, Android, Software Development, Kotlin.

ÍNDICE

CAPÍTULO I: INTRODUCCIÓN

1. MARCO DE REFERENCIA

1.1. Justificación

Es innegable que las aplicaciones móviles se han convertido en parte fundamental del día a día de todo mundo, aplicaciones de todo tipo, desde redes sociales hasta aplicaciones para hacer la vida más fácil del usuario en el día a día. Muchas veces es complicado saber cómo llegar desde un punto a otro en el distrito metropolitano de Quito (DMQ) mediante uso del transporte público. Además, las aplicaciones existentes de transporte público, muchas están desactualizadas en cuanto a rutas o simplemente fuera de servicio por completo. Según el Plan Maestro de Movilidad Sostenible del Municipio de Quito (PMMS) alrededor del 70% de ciudadanos quiteños son usuarios del sistema de Transporte Publico y en promedio los usuarios pierden en promedio unos 77 minutos diarios al movilizarse en los medios de transporte público masivo (Machado, 2022).

Estas son algunas de las motivaciones tras la realización de este proyecto el cual busca implementar una aplicación móvil para la facilidad y simplificación de la toma de transporte publico dentro del DMQ mediante una presentación clara y detallada de que frecuencia se debe tomar, en que parada se debe abordar, desembarcar y el precio de las distintas frecuencias. Al simplificar la toma de decisiones sobre que bus y que ruta en específico se debe tomar, el usuario podrá notar mejoría en el tiempo consumido en el transporte Publico sin tener que depender de aplicaciones de movilidad privadas que podrían crear un desbalance en la economía del usuario. La aplicación también tendría su impacto en el ámbito turístico, esto debido a que sería sencillo el solo descargar la aplicación y tener la información disponible de manera inmediata.

1.2. Planteamiento del problema

¿En qué bus vas vos ve?, nace de la necesidad de suplir una demanda en el mercado, ofreciendo una simplificación en la toma de decisiones sobre que bus o buses tomar para llegar de un punto a otro de la ciudad, ya que el usuario utiliza rutas más conocidas muchas veces por, costumbre o recomendaciones de amigos y familiares que termina desembocando en un tiempo más largo de viaje al utilizar rutas menos optimas y el uso de aplicaciones como Uber, Didi, entre otras para movilizarse dentro del DMQ puede llegar a resultar costoso a largo plazo.

Al usar transporte público es claro que hay rutas que son más utilizadas que otras y que varias de ellas comparten tramos en su trayecto. La aplicación sería capaz de dar todas las opciones de bus que cumplan con el trayecto deseado las cuales pueden ser menos utilizadas lo que resulta en más comodidad dentro del transporte público.

Hoy por hoy al visitar la Play Store y buscar aplicaciones de este tipo, en su mayoría se encuentran obsoletas, ya sea por falta de actualización de las rutas o por el completo abandono del proyecto y aquellas que siguen en línea presentan la información de manera confusa provocando que se tenga que realizar más búsquedas para poder determinar qué línea de buses se debe tomar.

En base al problema ya expuesto se plantea la siguiente pregunta como principal:

- ¿Cómo se puede cumplir con las necesidades de la comunidad manteniendo estándares de calidad?

También planteamos las siguientes preguntas como secundarias:

- ¿Existen procesos para respaldo constante de datos?
- ¿Se dispone de la infraestructura necesaria para aplicar lo planteado?
- ¿Qué tan grande será el impacto del proyecto en la comunidad?

1.3. Objetivo General

Desarrollar e implementar un aplicativo móvil para movilidad publica de nombre “¿En qué bus vas vos ve?”

1.4. Objetivos Específicos

- Implementar el aplicativo móvil utilizando el lenguaje de programación Kotlin y la plataforma Android, aplicando buenas prácticas de programación y asegurando que la aplicación funcione correctamente en diversos dispositivos móviles.
- Diseñar la interfaz y la estructura de navegación del aplicativo móvil "¿En qué bus vas vos ve?" para organizar y presentar claramente la información sobre opciones de transporte público, utilizando principios de usabilidad y diseño centrado en el usuario.
- Mejorar la experiencia del usuario en cuanto al uso del Transporte público masivo en el DMQ.

1.5. Alcance

Este trabajo tiene como alcance desarrollo e implementación de una aplicación móvil con el propósito de facilitar la toma de decisiones respecto a cuál de las 191 frecuencias de transporte público, específicamente de modalidad intracantonal urbano, proporcionadas por la Secretaría de Movilidad de Quito y que operan dentro del Distrito Metropolitano de Quito (DMQ), debería utilizar el usuario. Esta aplicación llevará el nombre de "¿En qué bus vas vos ve?".

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2. Marco Teórico

2.1. Antecedentes

El trabajo de titulación presentado se realizará sobre bibliografía planteada previamente por otros autores que hayan aportado a la importancia de la implementación de sistemas similares a las planteadas previamente además de evidenciar la falta de herramientas de este tipo dentro del DMQ. Según el diario colombiano Semana en el año 2012, nació Moovit (aplicación móvil de planificación de viajes y transporte público en tiempo real. Proporciona información sobre rutas y horarios de transporte público en varias ciudades alrededor del mundo) con algo de éxito en su país Natal Israel, sin embargo, el verdadero éxito de la aplicación llegó, una vez Moovit arribo a Latinoamérica, iniciando en Colombia con resultados bastante favorables mejorando en un 70% la satisfacción de los usuarios del sistema de transporte público en Bogotá, y mejorando en un 50% la percepción de complejidad de este. Para finales del 2013 contaban con 3 millones de usuarios en países como Colombia, Chile y Brasil (Semana, 2020).

En mayo del 2020 el éxito masivo de Moovit hizo que el gigante tecnológico Intel la comprase por $900 millones de dólares aproximadamente (Intel, 2020).

A pesar de que Moovit funcione en el DMQ, esta presenta la información de forma un poco confusa, dado que, las frecuencias de transporte público en Quito, además de contar con el número de frecuencia en ellas cuentan con nombres de las cooperativas que las operan, y es de esta forma es como comúnmente se las conoce en el día a día por el usuario.

El día 19 de septiembre del 2017 bajo la administración de Mauricio Rodas el Municipio de Quito lanzo el aplicativo de nombre “MOVILIZATE UIO” el cual pretendía ser:

Una herramienta tecnológica de vanguardia, con tecnología de Google Inc., que permitirá al usuario del Sistema Metropolitano de Transporte Público conocer toda la información del servicio de transporte, rutas, horarios, denuncias en línea y generar reportes en tiempo real sobre su experiencia de viaje. (Quito Informa, 2017).

El éxito de dicha aplicación fue moderado dado que para el día 15 de febrero del 2018, a 5 meses de su lanzamiento conto con 16.442 descargas, sin embargo, gracias a datos obtenidos de Google Analytics, la función de “Buscar Ruta” fue la más utilizada por los usuarios durante ese periodo (Quito Informa, 2018). Nuevamente el día 15 de diciembre del 2020 se realizó un relanzamiento del aplicativo, esta vez, bajo la administración de Jorge Yunda donde el único cambio significativo se lo vio en la incorporación de un lector de códigos QR que funcionaria en conjunto con códigos QR en las unidades de transporte público con la intención de mejorar el sistema de quejas y denuncias (Quito informa, 2020). Sin embargo, hoy en día dicha aplicación, a pesar de seguir disponible para su descarga dentro de la Play Store de Android, se encuentra fuera de servicio en lo que fue su principal función la de “Buscar Ruta” además de que se nota un claro abandono en el mantenimiento de este lo que se puede observar revisando las calificaciones y opiniones de los usuarios en la Play Store donde se puede evidenciar que cerca del último trimestre del 2021 fue cuando se dejó de dar soporte a la aplicación. Hoy en día la aplicación cuenta con una calificación de 2.3 estrellas.

2.2. Impacto tecnológico

El impacto de la tecnología en el mundo durante los últimos años es uno de los más grandes en la historia de la humanidad, gran parte de este impacto tecnológico llego de la mano del internet.

El internet ha pasado a ser parte fundamental del día a día de los individuos que forman parte de la sociedad, ha cambiado la manera en la cual se trabaja, se comunican y como se

interactúa con el resto del mundo, esto vio un, incluso mayor, incremento a lo largo del año 2020 tras la pandemia de COVID-19 (Aranzamendi, 2020).

Todas las industrias han tenido que dar el paso y hacer uso del internet para mantenerse, no solo relevantes en su campo, sino que incluso liderarlo al conseguir ventajas competitivas sobre sus competidores. Las aplicaciones móviles han sido una parte importante en el mundo de la tecnología y un mercado en el que todas las industrias tienen que estar de una u otra manera, esto ya que el número de suscripciones de smartphones a nivel mundial es de más de 6 mil millones de unidades, número que solo se estima seguirá creciendo paulatinamente año tras año (Statista, 2022).

2.3. ¿Como se puede crear una aplicación móvil en Android?

Primero debemos entender que es una aplicación móvil y porque vamos a desarrollar el sistema en ese formato, teniendo como antecedente que una aplicación móvil es un producto de software diseñado para operar sobre dispositivos móviles, en su esencia no dista mucho de lo que sería una aplicación tradicional para computadoras de escritorio, lo que las diferencia es la plataforma en la que se desarrolla y dependiendo del caso el lenguaje con el que son construidas, en este caso la aplicación se la realizara sobre la plataforma de Android.

El desarrollo de una aplicación móvil no dista mucho del desarrollo de aplicaciones convencional, la principal diferencia entre el proceso de desarrollo de una aplicación móvil y el de una aplicación web o de escritorio es la plataforma y las tecnologías para utilizar.

El primer paso del desarrollo siempre será el análisis de requerimientos tanto funcionales como no funcionales, y en base a estas se realiza la decisión sobre qué y cuales tecnologías se utilizará.

A su vez también se debe seleccionar una metodología de trabajo que se adecue al equipo y al tipo de proyecto a realizar, el tipo de metodología a implementar dictara el enfoque y la filosofía de desarrollo de la aplicación a realizar.

2.4. Tecnologías Seleccionadas

2.4.1. Lenguaje de programación

Hoy por hoy existen varias opciones a la hora de elegir que lenguaje de programación deseamos utilizar para el desarrollo de aplicaciones móviles, sin embargo, nos enfocaremos en los 2 lenguajes soportados oficialmente por Google, Java y Kotlin.

Java. _ es uno de los lenguajes de programación más conocidos y utilizados por desarrolladores alrededor del mundo ocupando el 4to puesto en el ranking de uso de Jetbrains del año 2022. Al ser un lenguaje Orientado a Objetos le permite ser multipropósito y se lo utiliza para desarrollo Back-end, desarrollo de aplicaciones de escritorio, desarrollo y manejo de Infraestructura y servidores, así como para el desarrollo de aplicaciones móviles (JetBrains, 2022).

Kotlin. _ nace en 2011 después de que Dmitry Jemerov, líder de JetBrains, declaro que no existía un lenguaje que tuviese todas las características que ellos buscaban a excepción de Scala, cuya debilidad era su lento tiempo de compilación. Así nace Kotlin, lenguaje que hoy en día ocupa un 16% del mercado y cuyo uso se basa principalmente en el desarrollo móvil (Amador, 2020).

En el año 2017 Google adopto oficialmente a Kotlin como lenguaje de desarrollo Android oficial junto a Java (The Kotlin Blog, 2017).

Tabla 1

Cuadro comparativo entre Java y Kotlin en el ámbito del desarrollo de aplicaciones para Android.

Características	Java	Kotlin	Pesos
Cantidad de Código	En Java se requiere escribir más Código para realizar la misma tarea.	Se puede realizar las mismas tareas que en Java en menos código, sin embargo, al no ser un lenguaje tan maduro, puede ser más difícil encontrar soluciones a problemas que en Java.	1
Estabilidad	Al ser un lenguaje establecido que lleva varios años en el mercado, cuenta con varias versiones que aun cuentan con soporte de los desarrolladores. En ese ámbito es seguro afirmar que Java es un lenguaje más estable que Kotlin.	Kotlin a pesar de tener el respaldo de Google sigue siendo un lenguaje relativamente nuevo que no cuenta con una larga historia de versiones estables con soporte a largo plazo, lo contrario a Java.	1
Funciones Extendidas	En Java no está disponible como tal, sino que se debe	En Kotlin esta es una de las características	1

	crear una nueva clase para poder empezar a extender la clase previa.	base permitiendo agregar métodos a clases sin modificar el código fuente.	
Corrutinas	Java trabaja con un hilo secundario para el trabajo intenso y al intentar manejar más subprocesos la probabilidad de errores en ejecución aumenta.	Kotlin adopto los conceptos y usos de Corrutinas establecidos en otros lenguajes y los aplico a sí mismo, logrando una simplificación en el uso de estas.	1
Seguridad "Null"	Uno de los problemas más comunes a la hora de desarrollar java es el error de Excepción de "*NullPointerException*"	En Kotlin no existe este tipo de error debido a que en el lenguaje no existe el tipo Null a no ser que sea definido explícitamente.	1
"Comodines"	Java cuenta con varios tipos de Comodines.	Kotlin no cuenta con comodines como tal, pero si presenta 2 tipos distintos de alternativas, las proyecciones de tipo	1

		y la variación del sitio de declaración.	
Puntuación Total	1	5	Puntuación Máxima = 6

Como podemos ver dadas las condiciones previas se determinó que el lenguaje más adecuado para el desarrollo del proyecto es **Kotlin**.

2.4.2. IDEs de Kotlin

Un IDE es el entorno de desarrollo integrado, es decir combina todas las herramientas necesarias para el desarrollo de aplicaciones bajo una misma GUI, lo cual facilita y simplifica el proceso de desarrollo.

Los IDEs de Kotlin considerados para realizar el proyecto fueron Eclipse, IntelliJ Idea y Android Studio.

Eclipse. _es un IDE orientado al desarrollo en Java, a pesar de ser reconocido como una sólida opción para el desarrollo en Kotlin, sin embargo, cabe aclarar que para poder utilizar Eclipse para Kotlin se tendrá que instalar Plugins para asegurar la compatibilidad.

IntelliJ Idea. _ es el IDE desarrollado por JetBrains, el cual es compatible con 4 lenguajes de forma nativa y en su versión gratuita mientras que en su versión comercial esta cifra aumenta a 10, sin embargo, mediante Plugins este número de lenguajes soportados aumenta a 15 y a 23 respectivamente. Según la empresa de ciberseguridad Inglesa Snyk en el año 2020 IntelliJ IDEA ocupaba el 62% del mercado de los IDEs que trabajaban sobre la Java Virtual Machine (Snyk, 2020).

Android Studio. _ es el IDE oficial de la plataforma Android, está basado en el Software de IntelliJ IDEA con un diseño enfocado específicamente para el desarrollo móvil de Android, a

partir del 2017 reconocen a Kotlin como lenguaje oficial (The Kotlin Blog, 2017), aunque cuentan con soporte para lenguajes como Java y C++.

Tabla 2

Cuadro Comparativo entre Eclipse IntelliJ IDEA y Android Studio para el desarrollo de aplicaciones en Android.

Características	**Eclipse**	**IntelliJ IDEA**	**Android Studio**	**Pesos**
Soporte para Kotlin	A través de un Plugin.	Soporte nativo y Completo.	Soporte Nativo y Completo.	1
Desarrollo para Android.	Requiere Configuraciones adicionales para ser compatible.	Tiene integración con Android.	Diseñado para desarrollo Android específicamente.	1
Herramientas visuales	No.	Si	Si	1
Emulador Android integrado.	No	Si	Si	1
Soporte Java	Si	Si	Si	1
Actualizaciones	No, descontinuado	Si.	Si.	1
Puntuación Final	1	4	6	Puntuación Máxima = 6

En resumen, **Android Studio** es la mejor opción para el desarrollo de aplicaciones Android con Kotlin debido a su integración nativa con el SDK de Android y herramientas específicas para el desarrollo móvil, adicionalmente se decidió trabajar con la API 31 de Android debido a que al ser una versión reciente contara con el soporte oficial de Google durante más tiempo. La API 31 de Android es compatible con dispositivos Android 12.0 o superiores los cuales conforman el 31% de los dispositivos Android alrededor del mundo (Composables, 2023).

CAPÍTULO III: METODOLOGÍA

3. Metodología de desarrollo del plan de tesis

El presente trabajo se realizará bajo el método de investigación aplicativa con enfoquen cualitativo.

3.1. Investigación Aplicada

La investigación aplicada busca la generación de conocimiento con aplicación en la sociedad. Los estudios realizados bajo esta modalidad de investigación presentan un gran valor agregado ya que se basan en investigación básica. Así indirectamente provoca una mejora en la calidad de vida de la población (Ortega, 2022).

Bajo la investigación aplicada se elaborará el desarrollo del marco metodológico para el posterior desarrollo de la aplicación móvil.

3.2. Enfoque Cualitativo.

Bajo el enfoque cualitativo la investigación tiene un método de recopilación de información más exploratorio la meta de los estudios cualitativos es la de explicar el porqué de las cosas proporcionando un entendimiento más profundo y humano del objeto de estudio (Muguira, 2017).

Bajo el enfoque cualitativo se utilizó el método de observación para determinar la problemática a atacar, en este caso el de suplir una demanda en el mercado, ofreciendo una simplificación en la toma de decisiones sobre que bus o buses tomar para llegar de un punto a otro de la ciudad

3.3. Metodología de desarrollo de software

Hoy por hoy al momento de iniciar un proyecto de desarrollo de Software existen principalmente dos tipos de metodologías: metodologías tradicionales y las metodologías agiles.

3.3.1. Metodologías tradicionales vs Metodologías Agiles.

Las metodologías de desarrollo de software tradicionales se caracterizan por ser más rígidas ya que no permiten realizar cambios una vez se definieron al principio del proyecto, este tipo de metodologías encuentra éxito cuando se tiene bastante claro y definido el cómo debe verse y funcionar el producto final (Ginzo Tech, 2021).

Mientras que las metodologías agiles nacen de la necesidad de tener más flexibilidad en el proceso de desarrollo de software. Por definición las metodologías agiles permiten adaptarse a las condiciones cambiantes del proyecto logrando así una mejora en la calidad del producto, una mayor satisfacción en el cliente y a largo plazo permite reducir costos (Sotomayor, 2021).

Tabla 3

Cuadro Comparativo entre Metodologías tradicionales y metodologías Agiles.

Características	Metodologías Tradicionales	Metodologías Agiles	Pesos
Flexibilidad	Se adaptan fácilmente a cambios	Son rígidas y poco flexibles ante	1

	de requisitos y prioridades durante el desarrollo del proyecto.	cambios, lo que puede generar dificultades al adaptarse a nuevas necesidades.	
Iterativo e Incremental	Utilizan ciclos de desarrollo cortos y entregas frecuentes, lo que permite obtener resultados tempranos y mejorar continuamente el producto.	Suelen seguir un enfoque lineal con etapas bien definidas, lo que puede retrasar la obtención de resultados tangibles y dificultar la retroalimentación temprana.	1
Colaboración y Comunicación.	Fomentan la colaboración activa entre los miembros del equipo y la comunicación constante con los clientes y usuarios.	La comunicación puede ser más formal y limitada, lo que puede generar malentendidos y falta de alineación con las necesidades del cliente.	1
Enfoque en el Cliente	Se centran en satisfacer las necesidades del	Pueden alejarse del enfoque en el cliente y estar más centradas	1

	cliente y entregar valor de forma temprana y continua.	en el cumplimiento de procesos y planificaciones.	
Entregas	Permiten lanzar versiones más tempranas y frecuentes del software, lo que facilita la obtención de retroalimentación temprana de los usuarios.	Las entregas suelen ser menos frecuentes, lo que puede retrasar la validación y retroalimentación del producto final.	1
Adaptabilidad	Se ajustan rápidamente a los cambios del entorno, las condiciones del mercado y las nuevas tecnologías.	Pueden tener dificultades para adaptarse a cambios externos debido a la naturaleza rígida de su estructura.	1
Transparencia	Proporcionan visibilidad clara del progreso del proyecto y permiten identificar problemas de forma temprana.	Pueden tener menos visibilidad del progreso y dificultades para identificar problemas a tiempo.	1
Puntuación Total	7	0	Puntuación Máxima = 7

Debido a que las metodologías ágiles destacan por su enfoque en la flexibilidad, la adaptabilidad, la colaboración y la entrega temprana de valor al cliente. Se decidió utilizar este tipo de metodologías para el desarrollo de la aplicación.

3.3.2. Elección de metodología dentro de las metodologías Agiles

Una vez decidido que se utilizará una metodología Ágil se debe elegir cuál de ellas será utilizada en el desarrollo de nuestra aplicación móvil. Dentro de las metodologías Agiles, las más destacadas son las siguientes:

Aspecto	**Kanban**	**Extreme Programming (XP)**	**Scrum**	**Prototipado**	**Pesos**
Flexibilidad	Altamente flexible, permite cambios en tiempo real y se adapta fácilmente a nuevas prioridades y requisitos.	Flexible, pero no tan adaptable como Kanban en términos de cambios repentinos.	Relativamente flexible, pero los cambios se realizan en las reuniones de planificación y sprint.	Puede ser flexible, pero la flexibilidad puede variar según la forma en que se enfoque el proceso de prototipado.	1
Roles y Estructura	Sin roles y estructura predefinidos,	Define roles específicos y	Define roles específicos y una estructura	Sin roles específicos, generalmente	1

	se adapta a las necesidades del equipo y la organización.	una estructura clara de equipo.	de equipo con roles predefinidos.	involucra a los interesados y al equipo de desarrollo en el proceso de prototipado.	
Entregas Continuas	Se enfoca en el flujo continuo de trabajo y la entrega constante de valor al cliente.	No se enfoca tanto en entregas continuas, sino en entregas incrementales dentro de sprints.	Entregas regulares al final de cada sprint.	Las entregas son iterativas, con el objetivo de obtener retroalimentación temprana del prototipo.	1
Planificación	Sin planificación específica de iteraciones o sprints, permite la planificación continua y flexible.	Planificación a través de iteraciones o sprints con duración fija.	Planificación a través de iteraciones o sprints con duración fija.	Se planifican iteraciones de desarrollo para la creación y mejora de prototipos.	1
Control de Flujo	Utiliza el tablero Kanban	No se centra tanto en	Utiliza el tablero Scrum para	El enfoque está en el desarrollo y	1

	para visualizar y gestionar el flujo de trabajo.	visualizar el flujo, sino en la ejecución de prácticas específicas.	visualizar el progreso del trabajo en las iteraciones.	mejora de prototipos para lograr el producto final.	
Gestión del Tiempo	No define un marco de tiempo específico para tareas, lo que permite una gestión más libre del tiempo.	Define sprints con duración fija y tiempo de desarrollo estimado para las tareas.	Define sprints con duración fija y tiempo de desarrollo estimado para las tareas.	Se centra en desarrollar prototipos en un marco de tiempo determinado para obtener retroalimentación.	1
Puntuación Total	5	2	2	4	Puntuación máxima = 6

Una vez presentadas las opciones se decidió ir por **Kanban** debido a su gran capacidad de flexibilidad ya que al ser un equipo de trabajo conformado por solo una persona no se podría aprovechar los beneficios de otras metodologías más enfocadas en la cooperación y basadas en roles asignados previamente.

4. Análisis y diseño

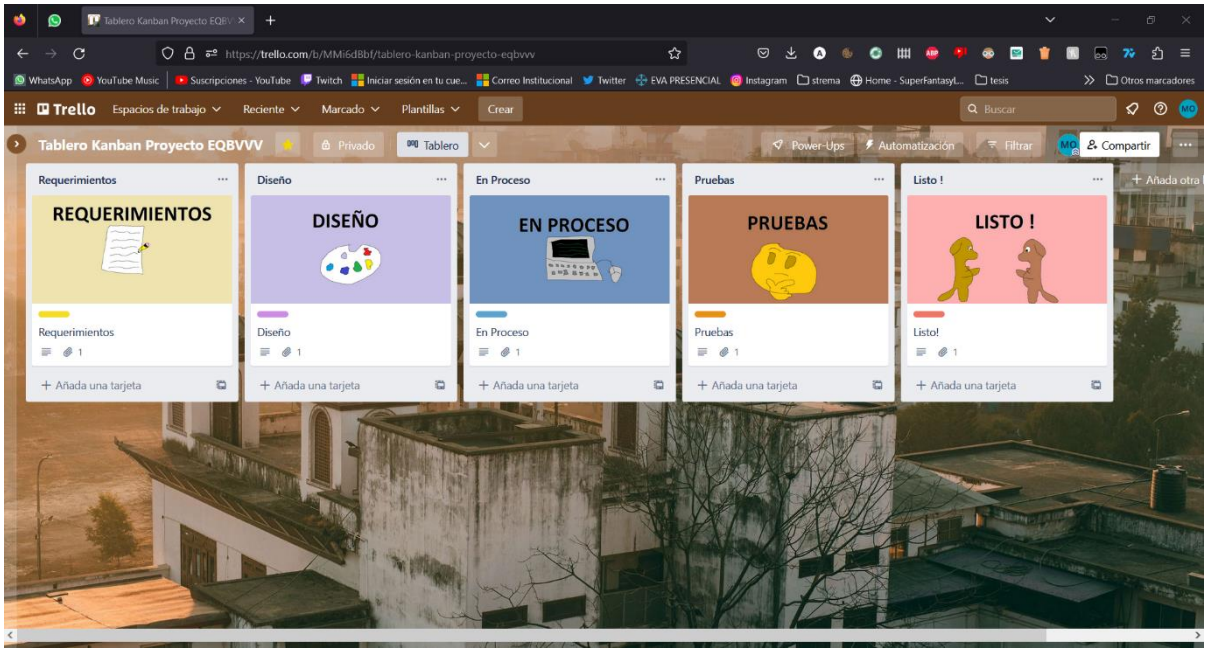
4.1. Requerimientos funcionales

Como fue definido previamente, se utilizará la metodología de Kanban, que es una metodología ágil, las cuales se basan en una pila de productos con historias de usuario para la especificación de requisitos funcionales, que son una descripción breve de una funcionalidad de software descrita mediante lenguaje común.

Para el manejo de tableros Kanban se utilizará “Trello”, un software en línea para la administración de proyectos. Para este proyecto se creó un tablero Kanban con 5 columnas detallando el proceso que deben seguir las tareas a realizarse (Fig. 1).

Figura 1

Tablero Kanban

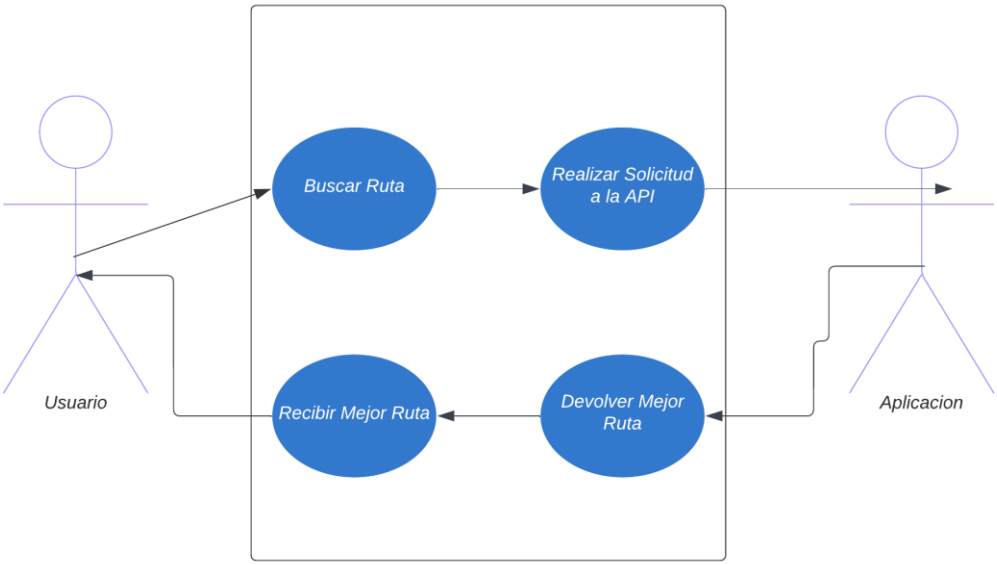


Para el desarrollo de este proyecto se definieron los siguientes requerimientos funcionales:

- **Toma de punto de partida.** _ La aplicación deberá presentar el punto actual del usuario o en su defecto, permitir ingresar un punto definido en el mapa.
- **Toma de punto destino.** _ La aplicación deberá permitir seleccionar la ubicación final a la que se desea llegar.
- **Cálculo de rutas.** _ La aplicación deberá calcular las posibles rutas desde el punto de partida hacia el punto de destino en menos de 15segundos.
- **Presentación de mejor ruta.** _ La aplicación deberá presentar la mejor ruta posible mediante instrucciones claras y concisas.

Figura 2

Diagrama de Caso de Uso de la aplicación



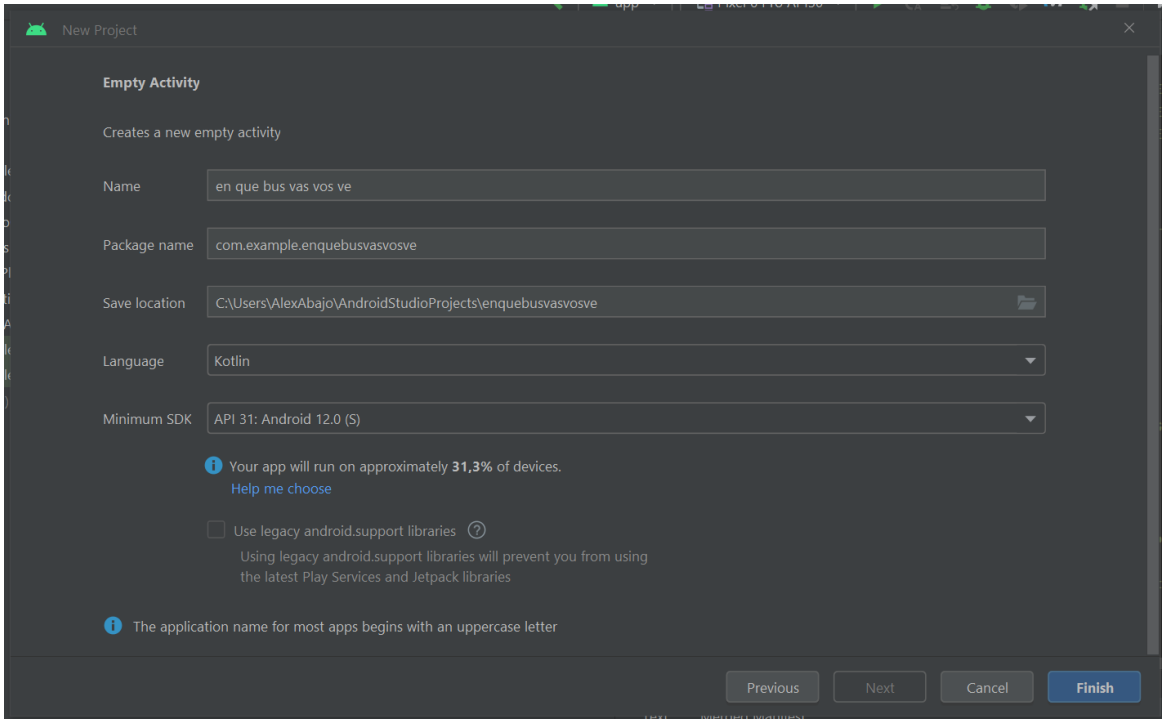
4.2. Fases de la aplicación.

De acuerdo con lo definido previamente el proyecto se lo realizar en el IDE de Android Studio por lo cual es necesario crear el proyecto dentro del mismo, donde elegiremos el tipo de

proyecto, el nombre de este y la Api para el cual desarrollaremos (Fig. 2). Una vez definidos estos parámetros, se da inicio al desarrollo del proyecto.

Figura 2

Creación de Proyecto en Android Studio.

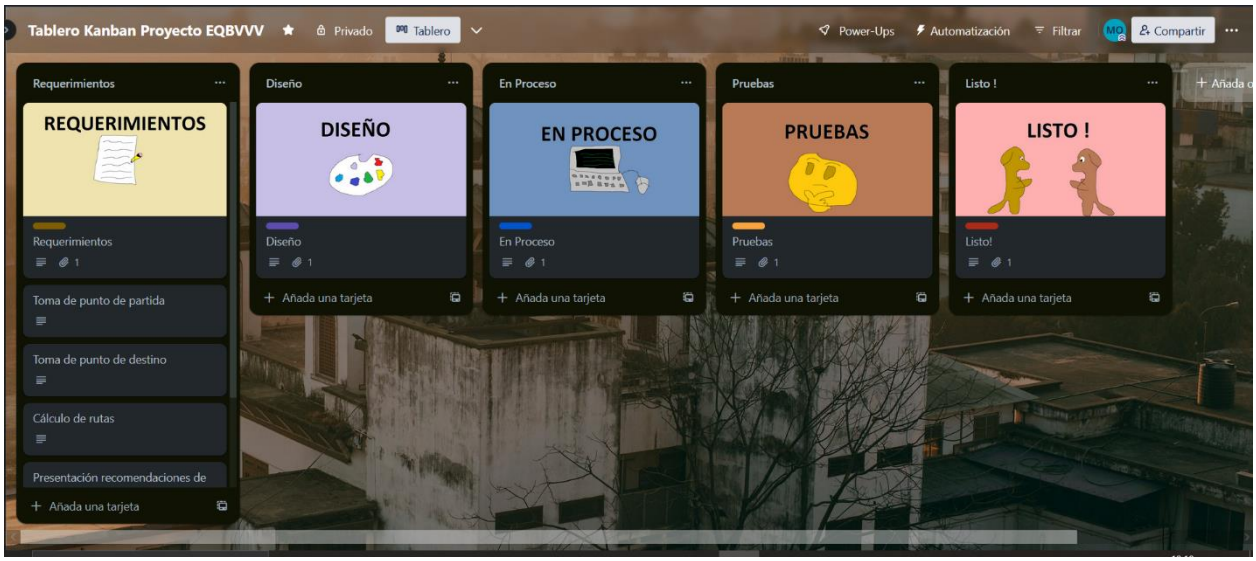


4.3. Toma de punto de partida y de destino

Debido a la similaridad entre ambos requerimientos, Toma de punto de partida y Toma de punto de destino, se los realizara a la vez. Estos son los primeros requerimientos por realizar de acuerdo con el tablero Kanban realizado (Fig. 3).

Figura 3

Tablero Kanban



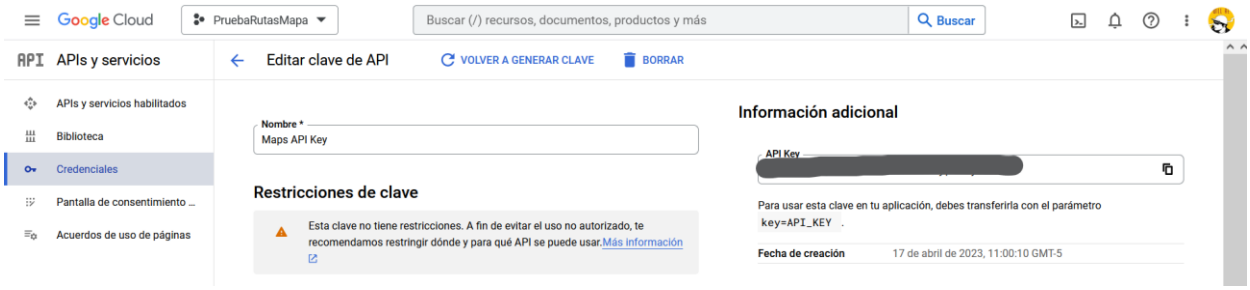
El propósito general de este requerimiento es el de poder seleccionar un punto en el mapa conocido como “punto de origen”, al iniciar la aplicación se colocará el punto en la ubicación actual del usuario, sin embargo, este mismo puede cambiar esta ubicación por alguna otra personalizada.

Desarrollo:

Al ser esta una aplicación de movilidad lo primero a realizar es incluir un mapa dinámico en el que nos podamos mover y seleccionar puntos sobre el mismo. Para esto trabajaremos con el componente SupportMapFragment de Google Maps, para lo cual tendremos que trabajar con la API de Google maps, por lo cual tendremos que registrarnos en la Cloud Console de Google para poder conseguir una API Key para poder utilizar los servicios de Google Maps (Fig. 4).

Figura 4

Google Cloud Console



Una vez tenemos el API Key, accedemos al archivo AndroidManifest.xml que se generó cuando creamos el proyecto en Android Studio, en este archivo es donde debemos agregar/editar todas las dependencias, meta-data, permisos y demás información relevante para el desarrollo de la aplicación.

Una vez añadida la información de nuestra API Key, entramos al main_activity.xml donde tendremos 2 opciones, utilizar el modo Diseño para añadir nuestro fragmento de mapa de forma visual, o mediante código en la opción de Código, de cualquier manera, este momento es donde se deben definir los constraints de los componentes, su ubicación y su tamaño, una vez realizado esto podemos dar paso a iniciar el mapa en la Activity, en este caso esto se realizará en el MainActivity.kt una vez iniciado el mapa, pedimos el acceso a los distintos permisos de usuario, para este caso debemos pedir acceso a COARSE_LOCATION y a FINE_LOCATION (Fig. 5), una vez obtenidos los permisos podemos obtener la ubicación exacta del usuario y por lo tanto podemos realizar ciertas animaciones de cámara, para ubicar el mapa sobre un área más específica, en este caso, centrada en el usuario con un zoom de 15 para tener una vista clara de los alrededores.

Figura 5

Código para Solicitar permisos.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

```

A continuación, volveremos al activity_main.xml para añadir dos TextView que cumplirán la función de etiqueta para los dos campos de origen y de destino, también añadimos los componentes de AutoCompleteTextView (Fig. 6) que nos ayudarán a auto completar direcciones para más facilidad y comodidad del usuario en el uso de la aplicación.

Figura 6

Llamado de Componente AutoCompleteTextView en el layout.

```xml

    <AutoCompleteTextView
        android:id="@+id/txtOrigen"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Origen"
        android:visibility="visible"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/lblOrigen"
        tools:visibility="visible" />

```

Una vez añadidos estos componentes en el activity_main.xml debemos, de igual manera, iniciar en el MainActivity.kt, además de escribir el código encargado del auto completado, el cual consiste en una llamada a un adaptador en una clase llamada “AutoCompletarAdaptador.java” (Fig. 7) que actúa como la lógica detrás de la llamada a la API de Google Maps que se realiza en otra clase llamada ApiPlaces.java (Fig. 8).

Figura 7

AutoCompletarAdaptador.java

```
37      @Override
38      public Filter getFilter(){
39          Filter filtro = new Filter() {
40              @Override
41              protected FilterResults performFiltering(CharSequence charSequence) {
42                  FilterResults resulFiltros = new FilterResults();
43                  if(charSequence != null){
44                      resuls = placesApi.autoCompletar(charSequence.toString());
45                      resulFiltros.values = resuls;
46                      resulFiltros.count = resuls.size();
47                  }
48                  return resulFiltros;
49              }
50
51              @Override
52              protected void publishResults(CharSequence charSequence, FilterResults filterResults) {
53                  if(filterResults != null && filterResults.count>0){
54                      notifyDataSetChanged();
55                  }
56                  else{
57                      notifyDataSetInvalidated();
58                  }
59              }
60          };
61          return filtro;
62      }
```

Figura 8

Llamada a la Api de Google Maps

```
public class ApiPlaces {
    1 usage
    public ArrayList<String> autoCompletar(String entrada){
        ArrayList<String> arrayList=new ArrayList();
        HttpURLConnection conexion=null;
        StringBuilder jsonResul= new StringBuilder();
        try{
            StringBuilder sb = new StringBuilder("https://maps.googleapis.com/maps/api/place/autocomplete/json?");
            sb.append("input="+entrada);
            sb.append("&key=AIzaSyCaHgFVdJRBWOIIm66UHOVypaGSyVnkoms");
            URL url= new URL(sb.toString());
            conexion = (HttpURLConnection) url.openConnection();
            InputStreamReader inputStreamReader = new InputStreamReader(conexion.getInputStream());

            int puntero;
            char[] buff = new char[1024];
            while((puntero=inputStreamReader.read(buff))!=-1){
                jsonResul.append(buff,  offset: 0, puntero);

            }
        }
        catch (MalformedURLException e){
            e.printStackTrace();
        }
        catch (IOException e){
            e.printStackTrace();
        }
        finally {
            if (conexion != null){
                conexion.disconnect();
            }
        }
        try {
            JSONObject objetoJson = new JSONObject(jsonResul.toString());
            JSONArray autocompletado = objetoJson.getJSONArray( name: "predictions");
            for (int i=0; i<autocompletado.length(); i++){
                arrayList.add(autocompletado.getJSONObject(i).getString( name: "description"));
            }
        }
        catch (JSONException e){
            e.printStackTrace();
        }
        return arrayList;
```

Ahora procederemos a realizar la lógica detrás de la creación de los marcadores indicando el punto de origen y destino en el mapa que se generaran una vez que se llenen los campos de los AutoCompleteTextView. Finalmente añadiremos un botón que elimine los marcadores generados en el mapa, y las direcciones escritas en los AutoCompleteTextView tanto del punto de origen como del punto de destino.

Figura 9

Aplicación ¿en qué bus vas vos ve?

Finalmente actualizamos nuestro tablero Kanban con la finalización de estas tareas (Fig. 10).

Figura 10

Tablero Kanban

4.4. Cálculo de Rutas

Para el cálculo de rutas se utilizará nuevamente el API de Google Maps. En este caso las solicitudes se realizarán después de aplastar el botón de Calcular Rutas mediante una URL (Fig. 11) que retorna una respuesta de tipo JSON, la cual debe ser tratada para obtener la información deseada.

Figura 11

Ejemplo de URL para consumir Api

```
Ejemplo de URL para solicitud JSON para conseguir Rutas:

https://maps.googleapis.com/maps/api/directions/json
?destination=-0.2094847,-78.4940086
&origin=-0.1770647,-78.4830828
&mode=transit
&key=AIzaSyCaHgFVdJRBWOIIm66UHOVypaGSyVnkoms
```

Para llevar a cabo este requerimiento debemos hacer uso de la biblioteca de Retrofit que nos facilitara el consumir la API de Google Maps. Para ello nos dirigimos al MainActivity.kt donde

construiremos la URL utilizando la información ingresada en los AutoCompleteTextView e invocando un método que nos permite obtener las coordenadas de latitud y longitud en base a dichas direcciones, ya que es mediante este tipo de coordenadas que se manejara la API de Google Maps, posteriormente realizamos la llamada a Retrofit dentro de una de las Corrutinas que se maneja dentro de un hilo de fondo que controla entradas y salidas (Fig. 12).

Figura 12

Construcción de URL y uso en Retrofit.

Esto nos lleva nuevamente a otro tipo de Adaptador que se encuentra en una clase llamada ServicioApi.kt (Fig. 13), en este caso que hará el manejo de la URL y servirá como intermediario, este adaptador se conecta directamente con la data Class RespuestaRuta.kt (Fig. 14), cuyo propósito es el de obtener solo la información relevante de la respuesta obtenida de la Api, es decir, es en este Data Class donde filtramos la información y solo nos quedamos con la información que necesitamos, que en este caso es la información de los pasos a seguir, es decir, si el usuario debe caminar a algún lado para poder tomar el bus, que bus debe tomar, y una vez se baje del bus, hacia donde debería caminar para llegar a su destino.

Figura 13

Adaptador ServicioApi.kt

```kotlin
package com.example.myapplication

import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query

interface ServicioApi {
    @GET("maps/api/directions/json")

    suspend fun getRuta(@Query("destination", encoded = true)destino:String,
                @Query("origin", encoded = true)origen:String,
                @Query("mode")modo:String,
                @Query("key")api_key:String
    ):Response<RespuestaRuta>
}
```

Figura 14

Data Class RespuestaRuta.kt

```kotlin
package com.example.myapplication

import com.google.gson.annotations.SerializedName

data class RespuestaRuta(@SerializedName("routes")val rutas:List<piernas>)
data class piernas(@SerializedName("legs")val piernas:List<pasos1>)
data class pasos1(@SerializedName("steps")val pasos1:List<instrucciones>)
data class instrucciones(@SerializedName("html_instructions")val instruccion:String)
```

Además, utilizaremos otro Adaptador de nombre ServicioApi2.kt (Fig. 15) cuyo funcionamiento es similar al de ServicioApi.kt, con la diferencia de la data Class con el que hace de intermediario, que en este caso es el RespuestaBus.kt (Fig. 16) cuyo propósito será el de obtener el número de frecuencia del bus o buses que se deberá tomar, ya que a este número podremos empatarlo con la información obtenida de la Secretaria de movilidad del municipio de Quito.

Figura 15

Adaptador ServicioApi2.kt

```kotlin
package com.example.myapplication

import ...

interface ServicioApi2 {
    @GET("maps/api/directions/json")

    suspend fun getBus(@Query("destination", encoded = true)destino:String,
                       @Query("origin", encoded = true)origen:String,
                       @Query("mode")modo:String,
                       @Query("key")api_key:String
    ):Response<RespuestaBus>
}
```

Figura 16

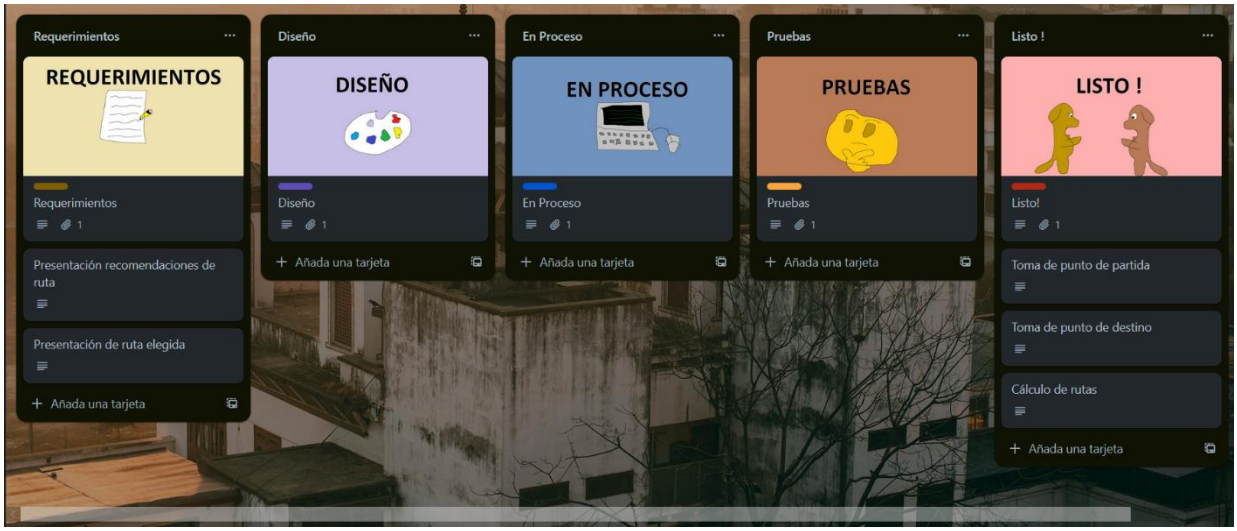
Data Class RespuestaBus.kt

```kotlin
package com.example.myapplication

import com.google.gson.annotations.SerializedName

data class RespuestaBus(@SerializedName("routes")val rutas2:List<legs>)
data class legs(@SerializedName("legs")val piernas:List<steps>)
data class steps(@SerializedName("steps")val pasos1:List<modo_transporte>)
data class modo_transporte(@SerializedName("travel_mode")val modo:String )
```

Una vez tenemos la información relevante para nuestro propósito, actualizamos nuestro tablero Kanban (Fig. 17) y damos por finalizado este requerimiento.

Figura 17

Tablero Kanban



4.5. Presentación de mejor Ruta.

En este requerimiento lo que realizaremos será presentar la mejor ruta obtenida en el requerimiento pasado, empatarlas con la información de los nombres de las frecuencias, información la cual obtuvimos de la página web de la Secretaria de movilidad del municipio de Quito, ya que así es como se las conoce popularmente, para que el usuario pueda tomar decisiones en base a ello. Para esto utilizaremos diccionarios de datos que contengan la información de las distintas frecuencias que operan en el DMQ.

Para satisfacer este requerimiento pasaremos a otro Activity en el cual se presentará las instrucciones de la Ruta, es decir, si debe caminar hacia alguna parada, que bus debe tomar, donde se debe bajar y si debe caminar nuevamente para llegar a su destino, para esto trabajaremos en el MainActivity.kt (Fig. 18), donde una vez verifiquemos que la información obtenida en el paso anterior es válida abriremos una nueva Activity y le pasaremos la información de la ruta.

Figura 18

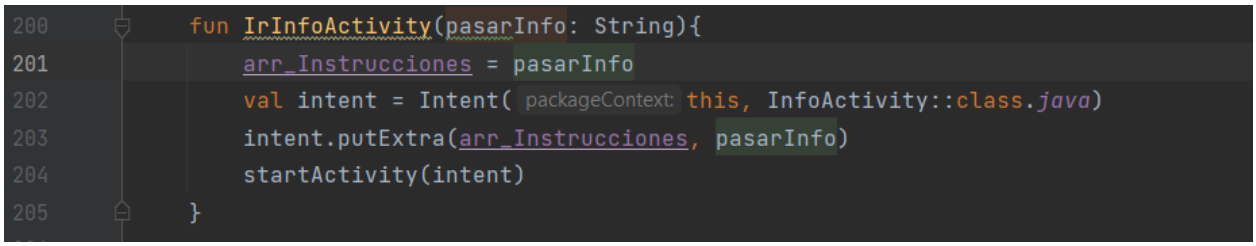
InfoActivity.kt

```
package com.example.myapplication

import ...

class InfoActivity : AppCompatActivity() {
    private lateinit var textView: TextView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_info)

        val intent = intent
        val texto = intent.getStringExtra(MainActivity.toString())
```

En este nuevo Activity de nombre InfoActivity.kt utilizaremos la información de la ruta obtenida previamente y pasada a este Activity para presentarla al usuario (Fig. 19).

Figura 19

Código para enviar información a InfoActivity.kt desde MainActivity.kt.

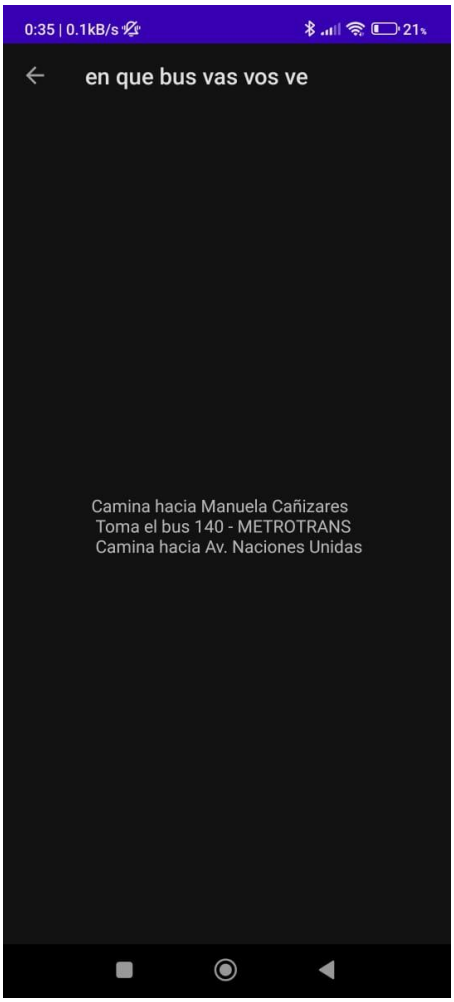


```
    fun IrInfoActivity(pasarInfo: String){
        arr_Instrucciones = pasarInfo
        val intent = Intent( packageContext: this, InfoActivity::class.java)
        intent.putExtra(arr_Instrucciones, pasarInfo)
        startActivity(intent)
    }
```

En el InfoActivity.kt tendremos que formatear el texto recibido y presentarlo al usuario (Fig.20).

Figura 20

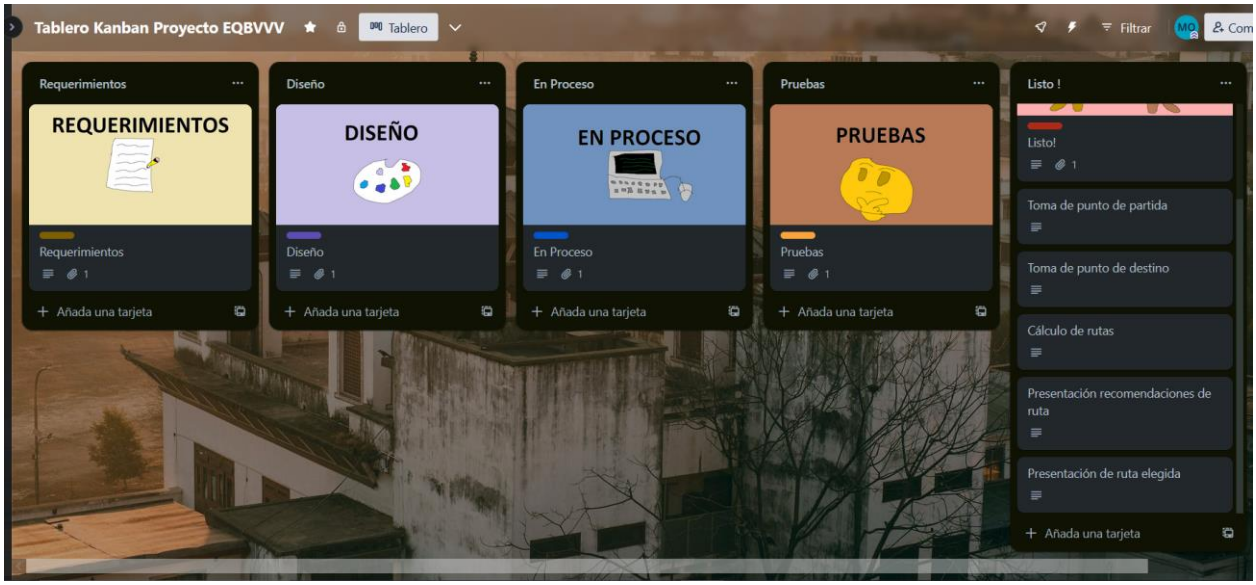
Vista de InfoActivity.kt presentando la información de manera clara y concisa para el Usuario.



Finalmente actualizamos nuestro Tablero Kanban (Fig. 21).

Figura 21

Tablero Kanban



CAPITULO V: RESULTADOS

5. Resultados

Tabla 4

Casos de Prueba Aplicación ¿en qué bus vas vos ve?

Escenarios	**Casos de Prueba**	**Estado**
Elegir punto de Origen	Se comprueba que funcione la función de Autocompletado.	Funcional
	Se comprueba que se genere el marcador en la ubicación del usuario.	Funcional

Elegir Punto de Destino	Se comprueba que funcione la función de Autocompletado.	Funcional
	Se comprueba que se genere el marcador en la ubicación elegida por el usuario.	Funcional
Botón Borrar Todo	Se comprueba que al presionar el botón “Borrar Todo” se borren todos los campos y los marcadores.	Funcional
Buscar Ruta	Se comprueba que al presionar el botón “Buscar Ruta” se envié la solicitud a la Api.	Funcional
Cambio de Activity	Se comprueba que al presionar el botón de “Buscar Ruta” se cambia de Activity.	Funcional
Presentación de Ruta	Se comprueba que al pasar de Activity se presenta la información de Ruta de manera clara y concisa.	Funcional

5.1. Pantallas Clave de la Aplicación

Pantalla final del MainActivity.kt (Fig. 22) presentando la búsqueda de ruta entre Pontificia Universidad Católica del Ecuador y Quicentro Shopping.

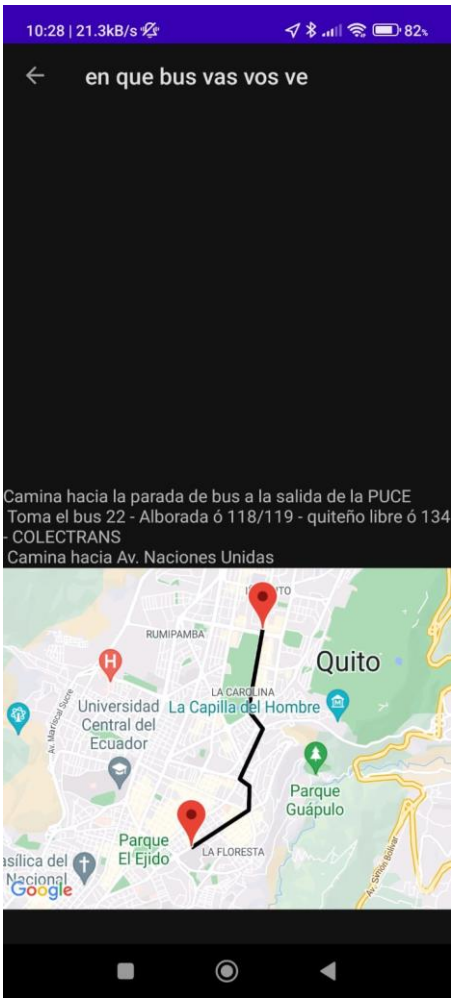
Figura 22

Pantalla Final de MainActivity.kt presentando los campos llenos para su uso.

Pantalla Final del InfoActivity.kt (Fig.23) presentando la información de la mejor Ruta al usuario.

Figura 23

Pantalla final de InfoActivity.kt presentando la información de la mejor Ruta al usuario.



CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

- Se Implemente el aplicativo móvil utilizando el lenguaje de programación Kotlin y la plataforma Android, aplicando buenas prácticas de programación y asegurando que la aplicación funcione correctamente en diversos dispositivos móviles.
- Se diseño la interfaz y la estructura de navegación del aplicativo móvil "¿En qué bus vas vos ve?" para organizar y presentar claramente la información sobre opciones de transporte público, utilizando principios de usabilidad y diseño centrado en el usuario.
- Se mejoro la experiencia del usuario en cuanto al uso del Transporte público masivo en el DMQ.

Recomendaciones:

- Se recomiendo seguir trabajando sobre el aplicativo debido al potencial de impacto que este tiene sobre la movilidad dentro del DMQ
- Se recomienda ponerse en contacto con el municipio de quito para analizar la posibilidad de ampliar el alcance de la aplicación a rutas rurales que siguen siendo parte del cantón de Quito.
- Se recomienda realizar seguimiento a las rutas de manera periódica puesto a que estas pueden cambiar y muchas veces los cambios no se ven reflejados dentro de Google Routes si no es meses mas tarde lo cual podría generar problemas en el propósito del aplicativo.

BIBLIOGRAFÍA

Amador, R. M. (2020, Diciembre 3). *Origenes de Kotlin | OpenWebinars.* Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-kotlin-y-sus-origenes/

Aranzamendi, H. A. (2020). *El Rol de la Tecnología en el nuevo contexto de COVID-19*. Obtenido de USMP Digital: https://www.administracion.usmp.edu.pe/revista-digital/numero-4/el-rol-de-la-tecnologia-en-el-nuevo-contexto-de-covid-19/

Composables. (2023, Enero 6). *Android Distribution Chart-Composables.co.* Obtenido de Composables.co: https://www.composables.co/tools/distribution-chart

Ginzo Tech. (2021, Diciembre 28). *Tipos de MEtodología Clásica en Desarrollo de Software.* Obtenido de Ginzo Tech: https://ginzo.tech/metodologia-clasica-desarrollo-software/

Intel. (2020, Mayo 4). *Intel adquiere Moovit para acelerar la oferta de movilidad como servicio de Mobileye.* Obtenido de Intel Newsroom: https://newsroom.intel.la/news/intel-adquiere-moovit-para-acelerar-la-oferta-de-movilidad-como-servicio-de-mobileye/

JetBrains. (2022). *The State of Developer Ecosystem in 2022 Infographic | JetBrains: Developer Tools for Professionals and Teams.* Obtenido de JetBrains: Developer Tools for Professionals and Teams: https://www.jetbrains.com/lp/devecosystem-2022/

Machado, J. (2022, Noviembre 8). *Los quiteños pierden hasta 77 minutos cada vez que viajan en bus.* Obtenido de Primicias - Noticias de Ecuador - Periodismo Comprometido: https://www.primicias.ec/noticias/sociedad/quito-tiempo-transporte-publico-hora-pico/

Muguira, A. (2017, Noviembre 19). *Diferencia entre Investigacion cualitativa y cuantitativa.* Obtenido de QuestionPro: https://www.questionpro.com/blog/es/diferencia-investigacion-cualitativa-y-cuantitativa/

Ortega, C. (2022, Diciembre 9). *Investigación Aplicada: Definición, tipos y ejemplos.* Obtenido de QuestionPro: https://www.questionpro.com/blog/es/investigacion-aplicada/

Quito Informa. (2017, Septiembre 19). *Municipio de Quito lanza 'MOVILIZATE UIO', la primera aplicación móvil de transporte público del Ecuador - Quito Informa*. Obtenido de Quito Informa: http://www.quitoinforma.gob.ec/2017/09/19/municipio-de-quito-lanza-movilizate-uio-la-primera-aplicacion-movil-de-transporte-publico-del-ecuador/

Quito Informa. (2018, Febrero 15). *"Movilizate Uio" supera las 16 mil descargas - Quito informa*. Obtenido de Quito Informa: http://www.quitoinforma.gob.ec/2018/02/15/movilizate-uio-supera-16-mil-descargas/

Quito informa. (2020, Diciembre 15). *Se presentó aplicación 'Movilízate UIO' - Quito Informa*. Obtenido de Quito Informa: http://www.quitoinforma.gob.ec/2020/12/15/se-presento-aplicacion-movilizate-uio/

Semana. (1 de Junio de 2020). *La historia del colombiano que llevó a Moovit a ser un referente de la movilidad.* Obtenido de Seman: https://www.semana.com/emprendimiento/articulo/quien-es-el-colombiano-que-llevo-a-moovit-al-exito/286874/

Snyk. (2020, Febrero 5). *IntelliJ IDEa dominates the IDE market with 62% adoption among JVM developers | Snyk.* Obtenido de Snyk | Developer Security | Develop fast. Stay secure. | Snyk: https://snyk.io/blog/intellij-idea-dominates-the-ide-market-with-62-adoption-among-jvm-developers/

Sotomayor, S. G. (2021, Diciembre 9). *Qué son las metodologías ágiles y cuales son sus ventajas empresariales.* Obtenido de Thinking for Innovation - El blog de Emprendimiento, Marketing, Social Media, Business & Tech y RRHH.: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

Statista. (2022). *Número de suscripciones de smartphones a nivel mundial desde 2016 hasta 2027 .* Obtenido de Statista: https://es.statista.com/estadisticas/636569/usuarios-de-telefonos-inteligentes-a-nivel-mundial/#statisticContainer

The Kotlin Blog. (2017, Mayo 16). *Kotlin on Android. Now official | The Kotlin Blog.* Obtenido de The Kotlin Blog: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/

GLOSARIO DE TÉRMINOS

ANEXOS