UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE UN PROTOTIPO FUNCIONAL PARA LA APLICACIÓN MÓVIL Q-BUS PARA LA PLATAFORMA IOS QUE BRINDE INFORMACIÓN DE LAS RUTAS DE TRANSPORTE PÚBLICO EN LA CIUDAD DE QUITO UTILIZANDO BLUETOOTH LOW ENERGY, CÓDIGOS QR Y GEO POSICIONAMIENTO

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

JUAN PABLO ROJAS HERNÁNDEZ

DIRECTOR: DR. DIEGO ORDÓÑEZ CAMACHO

Quito, abril 2016

DERECHOS DE AUTOR

DECLARACIÓN

Yo **JUAN PABLO ROJAS HERNÁNDEZ**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Juan Pablo Rojas Hernández

C.I. 1720947090

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**Desarrollo de un prototipo funcional para la aplicación móvil Q-Bus para la plataforma iOS que brinde información de las rutas de transporte público en la ciudad de Quito utilizando Bluetooth Low Energy, códigos QR y geo posicionamiento**", que, para aspirar al título de **Ingeniero en Informática y Ciencias de la Computación** fue desarrollado por **Juan Pablo Rojas Hernández**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Dr. Diego Ordóñez Camacho

DIRECTOR DEL TRABAJO

C.I. 1710449016

ÍNDICE DE CONTENIDOS

PÁGINA

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

ÍNDICE DE ANEXOS

PÁGINA

RESUMEN

El presente trabajo de titulación expone el análisis, diseño e implementación de un prototipo funcional para la aplicación móvil Q-Bus en la plataforma iOS que brinde información de las rutas de transporte público en la ciudad de Quito utilizando Bluetooth Low Energy, códigos QR y geo posicionamiento.

Para el desarrollo de la investigación se utilizó la metodología de desarrollo basada en prototipos, de esta manera se pudo obtener un producto de alta fidelidad que se asemeje en la mayor medida posible a una aplicación móvil final. Durante el proceso se utilizaron varias herramientas, entre las que destaca el entorno de desarrollo Xcode sobre el cual se codificaron todos los diseños realizados en etapas iniciales. Para dicha codificación se utilizó el lenguaje de programación ObjectiveC, además de otros protocolos como JSON para la transferencia de información entre la aplicación y el servicio web, y KML para el trazo de rutas en un mapa digital. La interfaz gráfica del prototipo fue diseñada siguiendo los estándares expuestos en el documento provisto por Apple iOS Human Interface Guidelines y se llevaron a cabo pruebas de funcionalidad, conexión y almacenamiento de información para asegurar la calidad de la aplicación.

Al finalizar todas las etapas de la metodología seleccionada, se procedió a realizar un análisis comparativo entre Q-Bus y aplicaciones semejantes existentes en AppStore para verificar que todos los objetivos propuestos se hayan logrado e identificar las fortalezas y debilidades de la aplicación desarrollada. Cabe mencionar que al ser Q-Bus un sistema prototipo, no se registraron en el mismo todas las paradas existentes en la ciudad de Quito, en su lugar, se tomó una muestra de 50 paradas ubicadas en distintas zonas de la ciudad con el fin de poder ejemplificar su funcionamiento y poner a prueba todas sus características.

ABSTRACT

This undergraduate thesis exposes the analysis, design and implementation of a functional prototype for mobile application Q-Bus on the iOS platform that provides route information for public transport in Quito city using Bluetooth Low Energy, QR codes and geo positioning.

For the research, the prototype-based methodology was used to obtain a high-fidelity product that resembles as much as possible to a final application. During the process several tools were used, the most important was Xcode IDE. It was used to codify all the designs that were made up in the early stages. The programming language selected was ObjectiveC and also were selected other protocols such as JSON for transferring information between the application and the web service, and KML for tracing routes on a digital map. GUI prototype was designed following the standards set out in the document provided by Apple, iOS Human Interface Guidelines; and the application were tested to ensure its quality in functionality, connection and storage.

Upon completion of all stages of the selected methodology, is shown a comparative analysis between Q-Bus and existing similar applications in AppStore to verify that all the objectives have been achieved and to identify the strengths and weaknesses of developed application. It is noteworthy that Q-Bus is a prototype system, were not recorded in the same all existing stops in Quito city, instead, this research took a sample of 50 stops located in different areas of the city to exemplify operation and test all of application features.

INTRODUCCIÓN

1. INTRODUCCIÓN

Actualmente los dispositivos móviles y sus aplicaciones han crecido considerablemente en términos de popularidad y utilización. La población está migrando rápidamente hacia el uso de smartphones, los cuales cada vez ofrecen mayores y mejores funcionalidades que se han convertido en necesarias para llevar a cabo muchas de las actividades diarias de todos. Es por esta razón que se analizó la opción de brindar información sobre el transporte público de la ciudad de Quito mediante una aplicación móvil para la plataforma iOS, aprovechando las características y beneficios que es posible obtener gracias a la utilización de tecnologías como Bluetooth Low Energy, códigos QR y geolocalización.

Ante la poca existencia de fuentes de información acerca de las rutas que siguen los buses en la ciudad de Quito, se vio la necesidad de realizar un proyecto que sea funcional para los usuarios de transporte público. Socialmente este proyecto apoya a los usuarios a eliminar el sentimiento de desconfianza y desconocimiento al momento de movilizarse de un lugar a otro y dando la oportunidad de una mejor planificación de sus tiempos y gatos al momento de realizar esta actividad. Se brinda también una guía que puede ser utilizada por turistas nacionales y extranjeros que visitan Quito y requieren movilizarse dentro de la ciudad gastando la menor cantidad de dinero posible. Los turistas, que posiblemente no conozcan de la ruta de ningún bus que circula por la ciudad tendrán la oportunidad de recibir sugerencias de buses por parte de la aplicación basadas en su posición actual y en la ubicación del sector al cual se quieren dirigir. El desarrollo de este proyecto apoya al crecimiento tecnológico y contribuir a la imagen innovadora y vanguardista de la ciudad ya que se utilizan tecnologías innovadoras a nivel mundial y que se usan mayoritariamente en Norteamérica y Europa, regiones que tecnológicamente se encuentran mayor desarrolladas que la nuestra. Es importante mencionar que gracias a la creciente popularidad de los dispositivos móviles, la solución que se propone mediante esta investigación

apoya considerablemente a la mantención del medio ambiente ya que se evita la impresión de folletos, trípticos o volantes reduciendo así el consumo de papel, cuya industria es una de las más contaminantes del mundo.

El objetivo general del proyecto es desarrollar un prototipo funcional para la aplicación móvil Q-Bus para la plataforma iOS que brinde información de las rutas de transporte público en la ciudad de Quito utilizando Bluetooth Low Energy, códigos QR y geo posicionamiento.

Los objetivos específicos de la investigación son: a) Determinar el funcionamiento de la tecnología Bluetooth y los dispositivos emisores de señal en la onda corta, b) Describir la definición, características y funcionamiento de los códigos QR, c) Investigar el funcionamiento de la geolocalización y mapas en los dispositivos móviles iOS, d) Implementar el prototipo funcional para la aplicación móvil Q-Bus.

MARCO TEÓRICO

2. MARCO TEÓRICO

Para el desarrollo de una aplicación móvil es necesario conocer conceptos básicos sobre la plataforma en la cual se ejecutará, así como de los aspectos que intervienen en el campo al cual esté destinada. En el caso del presente documento, se abarca como primer concepto el transporte público en la ciudad de Quito, para posteriormente hacer una revisión sobre la plataforma iOS y las funcionalidades que se incorporarán a la aplicación móvil como geolocalización, bluetooth y códigos QR.

A continuación se explica todos los conceptos que hacen referencia al proyecto, así como también algunas herramientas, métodos y metodologías de utilidad.

2.1 TRANSPORTE PÚBLICO

El transporte público, también denominado transporte de masas, hace referencia a los vehículos de transporte colectivo de pasajeros. Es un sistema ofrecido por uno o varios operadores, los cuales son los encargados de establecer rutas y horarios en los cuales el servicio podrá ser utilizado por los usuarios. Este servicio es mantenido mediante el cobro directo a los pasajeros y son regulados y subvencionados por autoridades locales o nacionales.

Sin importar el país del que se hable, el transporte público constituye una industria básica para el desarrollo de la sociedad ya que permite el traslado de un gran volumen de ciudadanos a sus actividades económicas y sociales. De ahí la importancia que tiene una correcta operación y un control adecuado por parte de las autoridades (Universidad Estatal a Distancia de Costa Rica).

2.1.1 TRANSPORTE PÚBLICO EN LA CIUDAD DE QUITO

El transporte público en la ciudad de Quito consta de los tres corredores viales principales Trolebús, Ecovía y Metrobús, los cuales son administrados por la Empresa Pública Metropolitana de Transporte de pasajeros de Quito (EPMTPQ). Además cuenta con 56 operadoras privadas que proveen aproximadamente 2100 buses que cumplen con distintas rutas dentro dela ciudad. (Secretaría de Movilidad, 2014)

2.2 APLICACIONES MÓVILES

Una aplicación móvil es un programa informático diseñado para ser ejecutado en un dispositivo portátil, como un smartphone, tablet o un reproductor de música mejorado. Generalmente son desarrolladas para ayudar en temas en particular y son distribuidas a través de una tienda de aplicaciones en línea. (Parsons, 2015)

2.2.1 APLICACIONES NATIVAS

Son todas aquellas aplicaciones que son desarrolladas para una plataforma en específico mediante el uso de un SDK, herramientas y lenguaje de programación que proporciona el proveedor de cada plataforma. (Caporarello, Di Martino, & Martinez, 2014)

Las aplicaciones nativas, en la mayoría de los casos, no requieren de internet para su funcionamiento; cuentan con un rendimiento optimizado y una interfaz gráfica adaptada al sistema operativo sobre el cual se ejecutan, además de tener acceso a funcionalidades nativas del dispositivo como GPS, Bluetooth, acelerómetro, cámara fotográfica, entre otras.

2.2.2 APLICACIONES WEB

Son aplicaciones desarrolladas utilizando HTML, CSS, JavaScript, entre otros. A este tipo de aplicaciones se pueden acceder desde cualquier dispositivo móvil que cuente con un navegador web, sin importar la plataforma. Las aplicaciones móviles web no pueden ser distribuidas a través de las tiendas de aplicaciones en línea.

2.2.3 APLICACIONES HÍBRIDAS

Son aquellas aplicaciones que se ejecutan en un navegador web embebido dentro de una aplicación nativa, es decir, usan tecnologías web para su desarrollo. Al ser en parte una aplicación nativa, tiene puede acceder a todas las funcionalidades que el dispositivo pone a disposición de las mismas, además de tener el sistema de distribución mediante las tiendas virtuales.

La Tabla 1 presenta una comparación entre los distintos tipos de aplicaciones móviles y que incluye aspectos de funcionalidad, desarrollo y distribución.

Tabla 1. Comparación entre los distintos tipos de aplicaciones móviles

Tipo de app	**Acceso a funciones del dispositivo**	**Velocidad**	**Costo de desarrollo**	**Tienda de apps**	**Proceso de aprobación**	**Interfaz gráfica optimizada**
Nativa	Total	Muy rápida	Alto	Disponible	Si	Si
Híbrida	Total	Rápida	Medio	Disponible	Si	No
Web	Parcial	Rápida	Bajo	No Disponible	No	No

2.3 IOS

iOS es un sistema operativo móvil desarrollado por la empresa Apple Inc. que se ejecuta en dispositivos iPad, iPhone y iPod; es el encargado de gestionar el hardware de los dispositivos y proporcionar las tecnologías necesarias para la implementación de aplicaciones nativas (Apple Inc., 2016).
Este sistema operativo móvil está basado en el concepto de manipulación directa mediante gestos multitáctiles por parte del usuario.

Su lanzado al mercado fue en junio de 2007, siendo la versión estable más reciente es la 9.2. El Anexo 1. Versiones de iOS muestra la evolución de las versiones del sistema operativo.

2.3.1 CARACTERÍSTICAS

En la Tabla 2 se indican las principales características del sistema operativo iOS.

Tabla 2. Principales características del sistema operativo iOS

Característica	Descripción
Tipo de núcleo	Núcleo híbrido (XNU)
Interfaz gráfica	Cocoa Touch
Plataformas soportadas	ARM
Licencia	APSL y Apple EULA
Lenguajes de programación	Objective-C, Swift

2.3.2 ARQUITECTURA

La Figura 1 muestra cómo la arquitectura de iOS está dividida mediante capas, en donde las capas inferiores contienen los servicios y tecnologías

fundamentales para el funcionamiento del sistema operativo, mientras que las capas superiores conforman un conjunto de interfaces que actúan como intermediarias en la comunicación entre el hardware y las aplicaciones que se ejecutan.

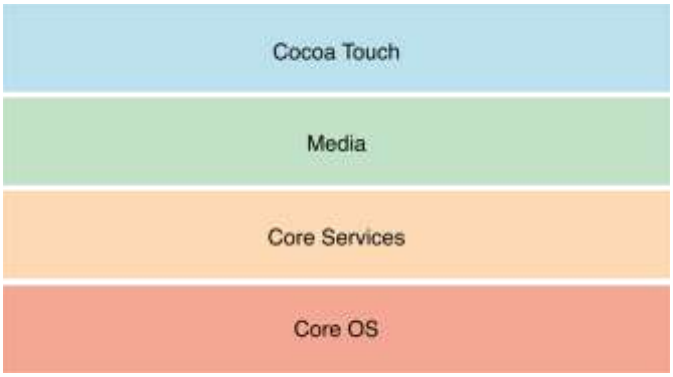

Figura 1. Arquitectura del sistema operativo iOS

(Apple Inc., 2016)

2.3.2.1 Capa Cocoa Touch

Contiene los frameworks necesarios para el desarrollo de aplicaciones nativas para iOS. Dichos frameworks permiten a las aplicaciones acceder a funcionalidades como el directorio de contactos, mapas digitales, servicio de mensajería de texto, correo, entre otros. Además esta capa define la interfaz gráfica o apariencia que tienen las aplicaciones.

2.3.2.2 Capa Multimedia

Provee tecnologías que permiten implementar contenido multimedia dentro de una aplicación incluyendo gráficos, animaciones, audio y video.

2.3.2.3 Capa Core Services

Contiene todos los servicios fundamentales que todas las aplicaciones necesitan para su funcionamiento y ejecución. Además contiene tecnologías para dar soporte a funcionalidades como la localización, iCloud y red.

2.3.2.4 Capa Core OS

Es la capa de más bajo nivel, es decir, la que tiene mayor interacción con el hardware del dispositivo. Dentro de esta capa se encuentran los frameworks para la interacción con el acelerómetro, bluetooth, accesorios externos, extensiones de red, seguridad, entre otros.

2.3.3 SDK

El Kit de Desarrollo de Software (SDK por sus siglas en inglés) de iOS provee herramientas e interfaces necesarias para el desarrollo, instalación, ejecución y pruebas de aplicaciones nativas, es decir, aplicaciones desarrolladas utilizando los frameworks de iOS y, Objective-C o Swift como lenguaje de programación.

El único IDE en el cual se pueden desarrollar aplicaciones nativas y que está certificado y soportado por Apple Inc. es Xcode. Actualmente este IDE se encuentra en la versión 7 y es posible descargarlo gratuitamente desde la Mac App Store disponible únicamente para el sistema operativo OS X.

2.4 GEOLOCALIZACIÓN

Un sistema de geolocalización es una solución tecnológica que determina la posición de un objeto en un entorno geográfico o virtual. Generalmente dicho objeto es un usuario que utiliza un sistema basado en geolocalización mientras se mantiene su privacidad (ISACA, 2011)

La geolocalización es un sistema para representar información a partir de la localización geográfica de un dispositivo ya sea móvil, computadora, GPS, entre otros. Los teléfonos móviles obtienen la posición principalmente por la triangulación de señales, es decir por la red de telefonía, el GPS o GPRS o, por la red de datos 3G o Wi-Fi. (Cacheiro González, 2014)

2.4.1 LOCALIZACIÓN PARA DISPOSITIVOS MÓVILES

Los dispositivos móviles, en especial los de la plataforma iOS utilizan un algoritmo de tres etapas que se ejecutan en un orden prioritario para determinar la posición del usuario (Warner, 2012). A continuación se detallan las etapas mencionadas anteriormente.

2.4.1.1 Sistema de posicionamiento global asistido A-GPS

Ayuda al dispositivo móvil a detectar satélites que se encuentran sobre la misma zona del planeta que el dispositivo con el fin de evitar que se consulte a todos los satélites existentes. A-GPS acelera el cálculo de la ubicación significativamente y agiliza el inicio de la funcionalidad GPS (Microsoft Corporation, 2016).

2.4.1.2 Crowdsourced Wi-Fi

El dispositivo almacena información de la red Wi-Fi a la cual está conectado y de las redes que se encuentran a su alrededor. Gracias al almacenamiento local del SSID de las redes, la dirección MAC y las coordenadas, el dispositivo puede determinar la ubicación con mayor rapidez a través de triangulación. (Warner, 2012)

2.4.1.3 Búsqueda de redes celulares

Implica un método de triangulación para poder obtener la posición del usuario utilizando la información que es almacenada por el dispositivo con respecto a las torres celulares cercanas. Se aproxima la posición del usuario mediante la medición de distancias entre el dispositivo y una serie de torres celulares que están cerca.

2.4.2 MAPAS

Un mapa es un conjunto de puntos, líneas y áreas, que están definidos tanto por su colocación en el espacio con respecto a un sistema de coordenadas, como por sus atributos no espaciales. (Dávila Martínez, 2014)

2.4.2.1 Mapas digitales

Son un conjunto de datos geográficos que se representan en una imagen cartográfica estática o dinámica en un computador o dispositivo. (Dávila Martínez, 2014)

Los mapas digitales estáticos son muy semejantes a los de papel pero se encuentran como una imagen en formato JPG, PNG, PDF, entre otros.

Por el contrario los mapas digitales dinámicos son los más comunes actualmente y ofrece imágenes de mapas desplazables de todo el planeta y otras funcionalidades como trazos de rutas o información de distintos lugares. Entre los productos más comunes que ofrecen este servicio de manera gratuita al público es Google Maps, Apple Maps y Bing Maps.

2.4.3 KML

KML es un formato de archivo utilizado para mostrar los datos geográficos en una aplicación de mapas, además de ser un estándar internacional mantenida por el Open Geospatial Consortium, Inc. (OGC). (Google Inc., 2016)

Un archive KML está basado en una estructura de etiquetas con atributos y elementos anidados de igual manera que el estándar XML, y permite representar rutas, marcas, polígonos, modelos 3D, entro otros elementos; mostrándolos en Google Earth, Google Maps o cualquier software geoespacial que implemente soporte para KML.

2.5 BLUETOOTH

Bluetooth es una especificación regulada por el grupo de trabajo IEEE 802.15.1, que permite la transmisión de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 2,4 GHz. (Prieto Blázquez, 2013)

Un dispositivo Bluetooth utiliza ondas de corto alcance (10 m), y opcionalmente medio alcance (100 m) para poder conectarse con otro dispositivo. Si dos dispositivos se van a comunicar, deben emparejarse estableciendo una red denominada piconet en la cual un dispositivo adopta el rol de maestro y el otro u otros el rol de esclavo. El dispositivo maestro será el encargado de definir la comunicación físicamente mientras que los esclavos coordinarán las transmisiones. (Bluetooth, 2016)

2.5.1 BLUETOOTH LOW ENERGY

También denominado Smart Bluetooth o Bluetooth 4.0, es una versión de Bluetooth que se caracteriza por su bajo consumo de recursos y su capacidad

de transmitir datos sin la necesidad del emparejamiento entre dispositivos. (Microsoft, 2016)

Este tipo de tecnología se convierte en ideal para su uso durante tiempos prolongados en la mayoría de dispositivos móviles mediante aplicaciones móviles que puedan explotar todos los beneficios de conectividad ofrecidos. En la actualidad, la gran mayoría de smartphones y tablets de gama media y alta incorporan esta tecnología, además de otros dispositivos como rastreadores o relojes inteligentes.

2.5.2 BEACON

Beacon es un pequeño sensor inalámbrico que puede adjuntarse a cualquier lugar u objeto. Este dispositivo transmite pequeñas señales de radio que un smartphone puede recibir e interpretar, desbloquear, obtener un micro-localización.

Las aplicaciones, utilizando un SDK, son capaces de entender su proximidad a los lugares y los objetos cercanos, el reconocimiento de su tipo, la propiedad, la ubicación aproximada, temperatura y movimiento. Los beacons pueden ser cualquier tipo de dispositivo que cuente con Bluetooth 4.0 y esté programado para ser un emisor de señales.

2.5.2.1 Funcionamiento

Un beacon emite ondas de radio a 2.4 GHz en un rango que aproximadamente puede llegar a los 70 metros, El alcance máximo exacto depende del medio ambiente en el que se encuentre (Estimote, 2016)

Cada beacon puede ser identificado ya que cada uno emite su propio identificador único (ID). Dicho ID tiene 20 bytes de longitud y está dividido en tres secciones:

- UUID (16 bytes)
- Major number (2 bytes)
- Minor number (2 bytes)

La Figura 2 muestra un ejemplo de cómo se estructuran los valores antes mencionados, lo cuales forman parte de una jerarquía (Estimote, 2016). En el caso del UUID hace referencia a una tienda, el major indica la sucursal de dicha tienda y el minor la sección de la sucursal.

Ubicación de la tienda		**San Francisco**	**Paris**	**London**
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major		1	2	3
Minor	**Ropa**	10	10	10
	Hogar	20	20	20
	Autos	30	30	30

Figura 2. Ejemplo de uso de identificadores de un beacon

(Apple Inc., 2016)

2.5.2.2 Transmisión de contenido

Generalmente los beacons son utilizados conjuntamente con una aplicación móvil que se encarga del continuo monitoreo de señales emitidas por los beacons. Los beacons no transmiten contenido como tal, lo único que transmite es su ID, el cual es recibido por la aplicación móvil que se encarga de realizar peticiones a internet o buscar contenido en la memoria interna del dispositivo para mostrar contenido o notificaciones.

2.6 CÓDIGOS QR

Código QR (abreviado de código Quick Response) es la marca comercial de un tipo de código de barras a manera de matriz (o código de barras bidimensional). El sistema de códigos QR fue una adaptación de la industria del automóvil y actualmente se lo utiliza para almacenar la información de los elementos físicos como el documento, los productos en un supermercado, los billetes de avión, etc.

Un código QR utiliza cuatro modos de codificación estandarizada (numérico, alfanumérico, byte / binarios, y kanji) para almacenar los datos de manera eficiente.

2.7 ACCESO A DATOS

2.7.1 TECNOLOGÍAS Y HERRAMIENTAS

Esta sección abarcará las herramientas y tecnologías que conforman la capa de acceso a datos. Se detallará la base de datos MySQL, una base de datos libre que cuenta con grandes características y aceptación, además de asociarse por defecto a otros proyectos open source como Moodle, Chamilo, Joomla, Wordpress o Drupal simplificando la conexión. Se detallará las principales características de PHP y, el formato JSON ideal para la transmisión de datos debido a su facilidad de lectura por casi todos los lenguajes de programación y su ligereza.

2.7.1.1 MYSQL

MySQL es la base de datos relacional de código abierto de mayor aceptación mundial y permite la oferta económica de aplicaciones de bases de datos fiables, de alto rendimiento y fácilmente ampliables basadas en la web e

integradas (Oracle, 2016). La versión más reciente de MySQL Community Server es la 5.7.10 lanzada al mercado en diciembre de 2015.

Características

Entre las características más importantes que se pueden mencionar están:

- Arquitectura cliente/servidor
  Existe un servidor de base de datos MySQL y una o varias aplicaciones/programas cliente.

- Compatibilidad SQL
  Soporta el uso de SQL, lenguaje estandarizado para la consulta y actualización de datos en una base de datos.

- Replicación
  Permita copiar el contenido de una base de datos en una o varias computadores.

- Transacciones
  La base de datos asegura que todas las operaciones que contenga una transacción se ejecuten o no se ejecute ninguna manteniendo el principio de atomicidad.

- ODBC
  Existen conectores que permiten interactuar con la base de datos desde distintos lenguajes de programación como C, C++, Java Perl, PHP, entre otros.

- Multiplataforma
  MySQL puede ser instalado en Mac OSX, Linux, Windows y en la mayoría de las variantes de Unix.

Licenciamiento

MySQL está licenciado bajo GNU GPL que obliga a que la distribución de cualquier producto derivado se haga bajo esa misma licencia. Si se desea cambiar de licencia, es posible obtener la licencia comercial de MySQL (Kofler, 2008).

2.7.1.2 PHP

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor. Es un lenguaje interpretado del lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad (Cobo, 2005). Está principalmente orientado al desarrollo web, en donde los programas son embebidos en código HTML y ejecutados a través de un intérprete previo al envío al cliente en HTML puro. Sin embargo, es posible también utilizarlo en scripts para cualquier tipo de procesamiento. La versión más reciente de PHP es la 7.0.1 liberada en diciembre de 2015.

Este lenguaje fue inicialmente concebido para ejecutarse sobre Unix en el cual puede aprovechar de mejor manera sus prestaciones, sin embargo, también se puede ejecutar en otros tipos de sistemas operativos. Además trabaja sobre la mayoría de servidores web y puede interactuar con más de veinte tipos de bases de datos (Cobo, 2005).

Algunas de las funcionalidades más importantes que PHP provee a los desarrolladores son:

- Funciones de correo electrónico
- Funciones de administración y gestión de bases de datos específicas para la mayoría de gestores comerciales.
- Funciones de gestión y transferencia de archivos
- Funciones de procesamiento de imágenes

- Funciones de generación y lectura de cookies

Licenciamiento

Está publicado bajo la licencia PHP License, la cual permite la redistribución del contenido licenciado en forma de código fuente o binaria siempre y cuando se incluya la declaración de los derechos de autor de la licencia PHP y se incluya el anuncio “This product includes PHP software, freely available from <http://www.php.net/software/>”.

2.7.1.3 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores (JSON, 2016).

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

2.8 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software permiten la correcta planificación, seguimiento y evaluación de las actividades y etapas que forman parte del proceso de desarrollo. Según la filosofía que las metodologías tengan se las puede agrupar en dos grandes grupos: tradicionales y ágiles.

La Tabla 3 muestra de manera resumida las principales diferencias entre las metodologías tradicionales y las ágiles.

Tabla 3. Comparación entre metodologías tradicionales y ágiles

Metodologías tradicionales	Metodologías ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes y posiblemente distribuidos	Grupos pequeños (hasta 10 integrantes) y trabajando en el mismo sitio
Más artefactos	Pocos artefactos
Más roles	Pocos roles
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software

(INTECO, 2009)

2.8.1 METODOLOGÍAS TRADICIONALES

Las metodologías tradicionales se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada

(Hernández, 2014). Generalmente se utiliza este tipo de metodologías cuando se conocen los requerimientos de manera completa al iniciar el proyecto, es por esto que implementar un cambio dentro del desarrollo incurre en altos costes, es decir, la falta de flexibilidad del proyecto es evidente.

Existen varias metodologías tradicionales como Microsoft Solutions Framework (MSF) o CMMI. Sin embargo en esta sección se detallará la metodología Rational Unified Process (RUP) ya que junto con el Lenguaje Unificado de Modelado UML, constituye una de las metodologías estándar más populares y usadas para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

2.8.1.1 RUP

El RUP (Rational Unified Process - Proceso Unificado Racional) es un proceso de ingeniería de software que provee un método disciplinado para asignar tareas y responsabilidades en una organización de desarrollo (Kruchten, 2004). Esta metodología está basada en seis principios clave:

1. Adaptación del proceso
2. Balancear prioridades
3. Colaboración entre equipos
4. Demostrar valor iterativamente
5. Elevar el nivel de abstracción
6. Enfocarse en la calidad

RUP se divide en cuatro fases:

Inicio

Se define el alcance inicial del proyecto con los stakeholders, se identifican los riesgos y se elabora un plan para las distintas fases e iteraciones posteriores.

Elaboración

En esta fase se realizan los procesos de definición, análisis y diseño. Se concretan los datos específicos del proyecto, se analizan los requerimientos, se identifican las funciones específicas que debe cubrir el sistema, se modela el sistema y, se diseñan datos, procesos e interfaces.

Construcción

Consiste en la implementación del software según las especificaciones que han sido desarrolladas en la etapa de elaboración.

Transición

Es la etapa de transición del producto al usuario final. Incluye actividades como distribución, soporte y mantenimiento.

La Figura 3 muestra las distintas disciplinas que intervienen en las distintas fases de RUP. Es posible observar que la gestión del proyecto está presente a lo largo de las cuatro etapas, disciplina fundamental que llevará el control de actividades y la verificación del cumplimiento de la planificación.

El modelado empresarial y la captura de requisitos juegan un papel importante durante la fase inicial. El análisis y diseño es la disciplina que más esfuerzo invierte durante la etapa de elaboración. Durante la construcción del software,

la implementación debe ser complementada con pruebas y una correcta gestión de cambios para la etapa de transición.

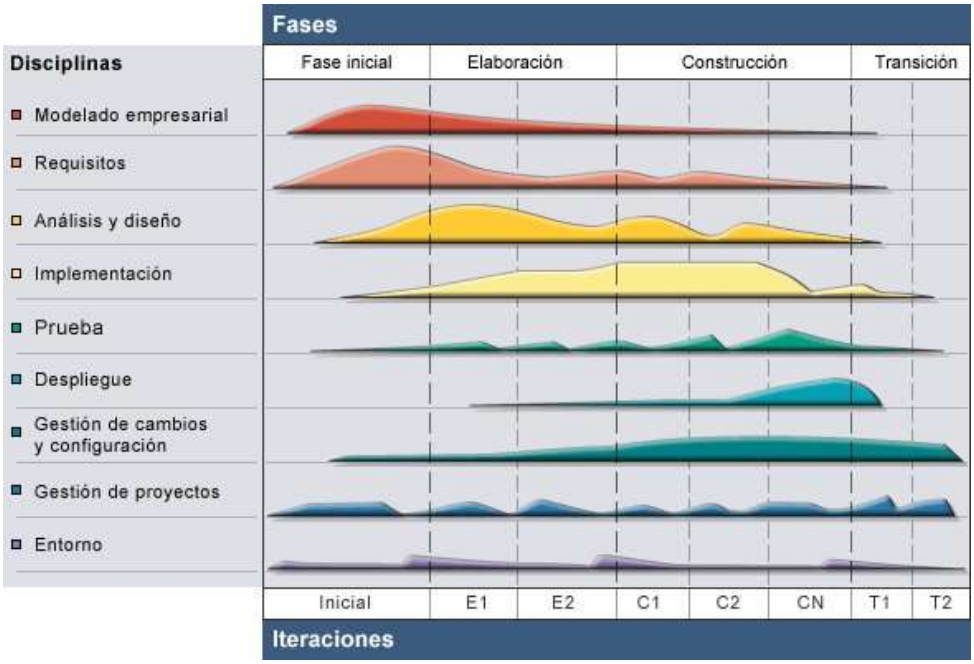

Figura 3. Disciplinas que intervienen durante las etapas de RUP

(IBM, 2016)

2.8.2 METODOLOGÍAS ÁGILES

Las metodologías ágiles nacen para contrarrestar los problemas que ocasionan las tecnologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa, es decir, no seguir estrictamente un plan para poder adaptarse a los cambios que puedan surgir durante el desarrollo (INTECO, 2009).

A partir del nacimiento del Manifiesto Ágil, documento que resume la filosofía ágil, nacen un sinnúmero de metodologías entre las cuales es posible destacar a XP, prototipado, SCRUM, Crystal Methodologies o el Desarrollo Rápido de Aplicaciones (DRA). Esta sección se centrará en las metodologías XP por su gran popularidad y uso, en el Prototipado debido a que uno de los entregables del presente trabajo de investigación es un prototipo funcional y,

en OOHDM, una metodología que se encuentra orientada principalmente al desarrollo de aplicaciones multimedia.

2.8.2.1 Metodología XP

XP se enfoca en potenciar las relaciones interpersonales, promoviendo el trabajo en equipo, la activa participación del cliente, la simplicidad en las soluciones implementadas y la rápida adaptación a los cambios. (INTECO, 2009).

En esta metodología los desarrolladores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Se implementa un desarrollo incremental mediante entregas pequeñas y frecuentes. (Sommerville, 2005)

La metodología XP consta de cuatro fases que se detallan a continuación:

Planeación

Se recopilan los requerimientos para elaborar las historias de usuario. Se realiza una planificación para las iteraciones y entregables.

Diseño

Se realizan diseños sencillos y entendibles para reducir el tiempo y el esfuerzo de su implementación. Se aplica el rediseño o refactorización, técnica para optimización del desempeño interno del sistema.

Codificación

Es la fase en la cual se desarrolla el código a partir de las historias de usuario. El código debe ser escrito mediante el uso de estándares y en pareja. En esta etapa es donde el cliente debe estar siempre disponible.

Pruebas

Se realizan todas las pruebas unitarias antes de liberar el sistema. Si es encontrado algún error se deben crear nuevas pruebas. Las pruebas de aceptación XP, se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente.

2.8.2.2 Metodología basada en prototipos

Es un enfoque basado en un proceso iterativo en que a partir de un análisis de requisitos inicial se realiza prototipos cada vez más complejos. El cliente da su opinión y se incorporan sus sugerencias para la siguiente iteración (Ganzábal García, 2014). Esta metodología permite dividir las actividades a llevar a cabo en etapas bien definidas:

Identificar requerimientos básicos

Se identificarán los requerimientos que sean necesarios para el desarrollo del prototipo funcional, el mismo que deberá contener funcionalidades que permitan atacar a los problemas y los solucionen de manera total o parcial.

Diseño del prototipo inicial

Se realizarán los diseños de la interfaz gráfica del prototipo así como del flujo de navegación que seguirá la aplicación.

Implementación del prototipo

Se codifica todos los diseños realizados en la etapa anterior.

Pruebas y mejora del prototipo

Se realizarán las pruebas correspondientes para verificar el correcto funcionamiento de la aplicación móvil y el cumplimiento de los requerimientos que hayan sido identificados. Una vez realizadas las pruebas necesarias, se hará una retroalimentación y se aplicarán los cambios que sean necesarios para garantizar la alta fidelidad del prototipo.

2.8.2.3 Metodología OOHDM

OOHDH (Object Oriented Hypermedia Design Method) es una metodología orientada al desarrollo de aplicaciones multimedia basado en el paradigma de la orientación a objetos. Propone un conjunto de tareas para reducir tiempos de desarrollo gracias a la reutilización de código y, los productos que se deben obtener en cada fase de desarrollo.

OOHDM está compuesta por cinco fases de desarrollo:

Determinación de Requerimientos

Se realizan diagramas de casos de uso para diseñar escenarios con el fin de obtener requerimientos y acciones del sistema.

Diseño Conceptual

Se construye un modelo orientado a objetos con el fin de capturar el dominio semántico de la aplicación.

Diseño de Navegación

Se define un esquema de clases de navegación específica, incluyendo clases especiales predefinidas conocidas como clases navegacionales. Estas clases son nodos, enlaces y estructuras de acceso.

Diseño de Interfaz Abstracta

Se definen los objetos de interfaz que va a percibir el usuario final y el camino en el cual aparecerán dentro de la aplicación.

Implementación

Escribir el código fuente de la aplicación y generar los recursos ejecutables.

2.9 ESTADO DEL ARTE

2.9.1 GEOLOCALIZACIÓN

La geolocalización en los dispositivos móviles ha sido tema de debate en los últimos años debidos a sus ventajas y peligros. Supone una mejora para determinadas aplicaciones, sobre todo de ocio y turísticas, pero también supone ceder privacidad de los movimientos de los usuarios de dispositivos móviles a servicios que la pueden utilizar con fines propios y dudosos.

Algunas de las aplicaciones más utilizadas por los usuarios como Instagram, Facebook, Twitter o Google Maps hacen uso de esta característica, sin embargo, en ocasiones puede resultar inquietante en qué manera nuestros datos de posición están siendo utilizados.

Pero las ventajas que brinda la geolocalización se centran en la búsqueda de servicios, lugares de interés o direcciones de referencia. Como primer ejemplo

es posible mencionar a Google Maps[1], aplicación desarrollada por Google que permite ver a los usuarios su posición actual dentro de un mapa digital, además permite trazar rutas que es posible seguir entre distintos puntos en vehículo o caminando. Otra aplicación que hace uso de esta funcionalidad es Waze[2], permite ver el estado del tráfico, presencia de policías o accidentes en las cuales se encuentra circulando el usuario en su vehículo, toda la información proporcionada por esta aplicación proviene de la comunidad.

2.9.2 BLUETOOTH LOW ENERGY

Esta tecnología ha ganado popularidad significativamente desde su introducción en dispositivos móviles Apple que cuentan con iOS 7. Apple proporciona una librería nativa para el manejo de BLE que incorpora métodos que permiten a un dispositivo móvil convertirse tanto en central como periférico, sin embargo esto resulta poco útil al momento de llevarlo a la realidad. La ventaja de esta tecnología es que es novedosa, no necesita del emparejamiento entre dispositivos, consume poca energía llegando a necesitar solo el 10% de lo que necesita Bluetooth clásico para funcionar (Pedro, 2011) y recolecta información constantemente. Sin embargo, la adquisición de beacons puede ser costosa.

La empresa Estimote[3] se dio cuenta de esto y puso en el mercado unos pequeños emisores de señal bluetooth que cumplen la función de periféricos, con un aspecto físico muy atractivo. Además, proporciona librerías de desarrollo gratuitas tanto para iOS como para Android y brinda funcionalidades como localización indoor, proximidad, búsqueda de beacons, entre otros.

[1] Sitio web oficial de Google Maps: https://www.google.com/maps/about/
[2] Sitio web oficial de Waze: https://www.google.com/maps/about/
[3] Sitio web oficial de Estimote: http://estimote.com/

Actualmente en España, la empresa iPhonedroid[4] mantiene el producto Kappta el cual es el primer sistema en España que combina Beacons y tecnología BLE para detectar la proximidad de un usuario a puntos específicos dentro o fuera de un establecimiento, permitiendo establecer una comunicación con dicho individuo en un radio de alcance determinado (iPhonedroid, 2016). El sistema está compuesto por beacons, una aplicación móvil y un sistema web que permite a los propietarios de la tienda enviar ofertas, promociones, imágenes, videos, etc.

Pero BLE no solamente es utilizado con fines publicitarios, empresas como Google, Samsung, Apple o Nike utilizan BLE para transmitir información entre smartphones y dispositivos como relojes inteligentes, zapatos o pulseras. Dicha información generalmente está asociada con perfiles de salud, deportes o notificaciones de mensajes, correo, entre otras.

2.9.3 APLICACIONES MÓVILES PARA TRANSPORTE PÚBLICO

Existen varias aplicaciones alrededor del mundo que brindan información del transporte público, sin embargo una de las más descargadas en las tiendas AppStore, Google Play y Tienda Windows es Moovit[5]. Esta aplicación cuenta con información del transporte público de más de 600 ciudades en 55 países y cuenta con más de 20 millones de usuario (Moovit, 2015). Su información proviene de la comunidad, permite ver información en tiempo real acerca del recorrido de las distintas líneas de transporte público. Además comprueba los próximos tiempos de llegada de los buses y permite controlar el trayecto seguido por el usuario. Lo más novedoso de esta app es que el usuario recibe notificaciones acerca de cuándo debe bajarse del bus en el que se encuentra viajando para llegar a su destino correctamente y que el usuario, dependiendo de la ciudad en la cual se encuentre, tiene la posibilidad de descargar mapas con rutas de los diferentes tipos de transporte público.

[4] Sitio weboficial de iPhonedroid: http://www.iphonedroid.com/es/
[5] Sitio web oficial de Moovit: http://moovitapp.com/es/

La multifuncional app Google Maps también incorpora información sobre el transporte público en muchas de las ciudades del mundo. Entre las ventajas de esta aplicación se encuentra la proporción de información de las rutas que parten desde determinada parada y el intervalo de salida, sin embargo y dependiendo la ciudad, únicamente se muestran las paradas del metro o tren.

El Gobierno de la ciudad de Buenos Aires mantiene publicada la aplicación "BA Cómo Llego"[6] disponible en AppStore, Google Play. Es una app bastante útil ya que permite obtener rutas para llegar desde un punto hasta otro tanto en auto como en transporte público (bus, metro o tren), además de brindar indicaciones para la llegada en bicicleta o caminando. Entre las principales ventajas que la aplicación brinda es la posibilidad de ver mapas temáticos (urbanismo, servicios públicos, educación, cultura entre otros), almacenar un registro de los lugares buscados y mostrar varias alternativas para llegar a un punto usando transporte público incluyendo el tiempo aproximado de viaje. Entre las desventajas que se hacen evidentes es la baja calidad de los mapas que utiliza, la difícil navegación entre las diferentes funcionalidades de la aplicación y que la búsqueda de rutas desde un lugar a otro requiere necesariamente los nombres de las calles.

Transit App[7] está disponible para Android, iOS y Windows Phone. Contiene información de transporte público en 99 ciudades repartidas en América del Norte, México y Europa. Es una app muy intuitiva que como primera opción muestra un listado de paradas ordenadas por distancia a la posición actual del usuario. Permite el trazo de rutas mediante la selección de puntos en el mapa digital para el origen y el destino, mostrando alternativas y tiempos estimados de recorridos caminando, en bus, en metro, en tren o en auto. En el caso de seleccionar el transporte público, la app traza la ruta indicando visiblemente

[6] Sitio web oficial de BA Cómo Llego: http://comollego.ba.gob.ar/
[7] Sitio web oficial de Transit App: http://transitapp.com/

que distancia se recorre caminando, que ruta sigue el bus y cuáles son las paradas.

Cabe recalcar que ninguna de estas aplicaciones funciona totalmente sin internet. Si bien es cierto las aplicaciones que se han mencionado en esta sección son bastante completas, ninguna de ellas trabaja sin una conexión a internet a menos de que el usuario haya descargado contenido adicional (Moovit).

METODOLOGÍA

3. METODOLOGÍA

Se realizó el diseño e implementación de un prototipo funcional para la aplicación Q-Bus para la plataforma iOS. A continuación se detalla el alcance, las herramientas y metodología que se utilizó para concluir con el proyecto.

3.1 ALCANCE

El presente proyecto abarcó la investigación de los componentes necesarios para implementar la aplicación móvil, es decir, se abordó temas como Bluetooth Low Energy, códigos QR, geolocalización, plataforma iOS y, herramientas y técnicas requeridas.

Cabe recalcar que la aplicación móvil desarrollada es un prototipo funcional, por lo cual no se tomaron en cuenta todas las paradas, rutas y cooperativas de buses existentes en la ciudad de Quito. Los beacons y/o códigos QR simularon estar ubicados en 5 paradas distintas de la ciudad.

La aplicación móvil desarrollada cumple con las siguientes funcionalidades:

- Identificar cuando el usuario llega a una parada de bus mediante la detección de un beacon o la lectura de un código QR.
- Mostrar la información de las rutas de buses que pasen por la parada detectada.
- Trazar la ruta que sigue el bus en un mapa digital.
- Realizar sugerencias de buses que puede tomar un usuario para llegar desde un punto hasta otro en la ciudad de Quito. Las sugerencias de buses presentan las rutas que circulen por paradas cercanas a ambos puntos. No se toman en cuenta transbordos.
- Buscar rutas de buses y poder ver su información.

3.2 EQUIPOS

La Tabla 4 presenta todos los equipos requeridos para el desarrollo del proyecto y los requerimientos que deben cumplir.

Tabla 4. Equipos requeridos para el desarrollo de Q-Bus

Equipo	Requisitos
Computadora	1. Sistema operativo Mac OS X 2. 4 GB de RAM
Impresora	1. Sin requerimientos especiales
Beacons	1. Cualquier dispositivo que pueda cumplir las funciones de emitir señales BLE
Dispositivo móvil	1. iPod touch, iPad, iPhone 2. Sistema operativo iOS 7 o superior

3.3 HERRAMIENTAS Y TÉCNICAS

La Tabla 5 presenta todas las herramientas utilizadas para el desarrollo del proyecto con su respectiva versión.

Tabla 5. Herramientas utilizadas para el desarrollo de Q-Bus

Software	Versión
Xcode	7
MySQL Workbench	6.3
Eclipse	Luna
Adobe Fireworks CS	6
OmniGraffle	6.5.1

3.4 METODOLOGÍA

Para cumplir con el principal objetivo de esta investigación se utilizó la metodología de desarrollo de software basada en prototipos. Las razones por las cuales se la eligió se detallan a continuación:

- Sólo se desarrollaron los requerimientos que se conocieron bien.
- La aplicación se desarrolló por fases.
- Se hizo una implementación parcial de la aplicación y se realizaron pruebas.
- Los usuarios pudieron probar la aplicación móvil y añadir nuevos requerimientos

3.4.1 METODOLOGÍA BASADA EN PROTOTIPOS

Esta metodología permitió dividir las actividades a llevar a cabo en etapas bien definidas:

3.4.1.1 Identificación de requerimientos básicos

Se identificaron los requerimientos que fueron necesarios para el desarrollo del prototipo funcional. De esta manera se implementaron funcionalidades que permitieron atacar a los distintos problemas y alcanzar una solución.

Se hizo uso del método de historias de usuario para obtención y presentación de requisitos del proyecto. Se elige este método ya que permite determinar requerimientos pequeños y concretos, contienen información imprescindible para detallar un requerimiento y qué es lo que el usuario necesita utilizando un lenguaje coloquial.

Para poder generar las historias de usuario se utilizó el método de observación de campo, de esta manera se pude determinar los problemas que existen en las paradas de buses y las necesidades que tienen los usuarios de transporte público.

3.4.1.2 Diseño del prototipo inicial

Se realizaron los diseños de la interfaz gráfica del prototipo así como del flujo de navegación que sigue la aplicación.

En la etapa de diseño del prototipo se utilizó el método Storyboard Prototyping, el cual consiste en una serie de dibujos o imágenes que representan la manera en que una interfaz gráfica es usada para cumplir con una tarea en particular (Snyder, 2003). Se eligió este método debido a que el diseño del prototipo, según la metodología a seguir, debió ser rápido y flexible a los cambios a lo largo de esta fase. Además permitió invertir pocos recursos tanto en tiempo como en personal y económico.

Se utilizó el lenguaje UML para diseñar los procesos e identificar el orden y el momento en que interactúan los distintos elementos del sistema. Los diagramas generados permitieron al desarrollador visualizar de mejor manera el diseño y entender el funcionamiento de la aplicación.

3.4.1.3 Implementación del prototipo

Para el desarrollo del prototipo se implementó el método en cascada ya que es uno de los más apropiados para proyectos pequeños y en los cuales se conocen bien los requerimientos al inicio del proyecto.

3.4.1.4 Pruebas y mejora del prototipo

Se realizaron las pruebas correspondientes para verificar el correcto funcionamiento de la aplicación móvil y el cumplimiento de los requerimientos que fueron identificados.

Una vez realizadas las pruebas necesarias, se hizo una retroalimentación y se aplicaron los cambios que fueron necesarios para garantizar la alta fidelidad del prototipo.

La aplicación móvil utiliza un modelo de prototipo reutilizable y de alta fidelidad ya que se lo implementó de la forma más cercana posible al diseño real en términos de interfaz gráfica, funcionamiento, interacción y tiempo.

ANÁLISIS DE RESULTADOS

4. ANÁLISIS DE RESULTADOS

El presente proyecto tuvo como objetivo el desarrollo de un prototipo funcional para la aplicación móvil Q-Bus en la plataforma iOS que brinde información de las rutas de transporte público en la ciudad de Quito utilizando Bluetooth Low Energy, códigos QR y geo posicionamiento. Para el cumplimiento de dicho objetivo, se utilizó la metodología de desarrollo basada en prototipos.

A continuación se detalle el proceso del desarrollo del proyecto con cada una de las fases descritas en la metodología antes mencionada.

4.1 IDENTIFICACIÓN DE REQUERIMIENTOS BÁSICOS

Para la identificación de requisitos se utilizó el estándar IEEE 830 y cuyos resultados se presentan en esta sección.

4.1.1 INTRODUCCIÓN

4.1.1.1 Propósito

La identificación de requerimientos se la realiza para determinar de manera clara y concreta que funcionalidades debe tener la aplicación para poder atacar a los problemas y alcanzar una solución.

4.1.1.2 Ámbito del sistema

La aplicación fue desarrollada con el fin de aportar con una alternativa al desarrollo del transporte público en la ciudad de Quito y como apoyo a los usuarios de este servicio. Q-Bus fue desarrollada para la plataforma iOS, implementando librerías de Bluetooth Low Energy, códigos QR y geo posicionamiento.

4.1.1.3 Definiciones y acrónimos

Las tablas 6, 7 y 8 muestran definiciones, acrónimos y abreviaturas que son utilizadas en la presente sección.

Tabla 6. Documento ERS: Definiciones

Término	Definición
Aplicación móvil	Aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. (Parsons, 2015)
Bluetooth	Especificación tecnológica para redes inalámbricas que permite la transmisión información que se encuentra almacenada en diversos equipos mediante una radiofrecuencia segura (2,4 GHz). (Bluetooth, 2016)
Código QR	Sistema que permite almacenar información en una especie de código de barras de última generación. (QR Channel, 2016)
Geolocalización	Localización de una persona, objeto, empresa, evento en un lugar geográfico exacto determinado por unas coordenadas. (ISACA, 2011)
Beacon	Dispositivo que emite señales de onda corta por medio de la tecnología Bluetooth. (Estimote, 2016)

Tabla 7. Documento ERS: Acrónimos

Acrónimo	Significado
ERS	Especificación de Requisitos de Software
RF	Requerimiento funcional
RNF	Requerimiento no funcional
GUI	Interfaz gráfica de usuario (Graphical User Interface)
API	Interfaz de aplicación de programación (Application Programming Interface)
SDK	Kit de desarrollo de software (Software Development Kit)
IDE	Entorno de desarrollo integrado (Integrated Development Environment)
UML	Lenguaje de modelado unificado (Unified Modeling Language)
BLE	Bluetooth Low Energy

QR	Quick Response
GPS	Sistema de posicionamiento global (Global Positioning System)

Tabla 8. Documento ERS: Abreviaturas

Abreviatura	Significado
APP	Aplicación
SO	Sistema operativo

4.1.1.4 Referencias

- Especificación de Requisitos según el estándar de IEEE 830. ANSI/IEEE Std. 830 - 1998

4.1.1.5 Visión general del ERS

El documento de ERS muestra de manera concreta y específica los puntos a considerar en el proyecto. Esta sección detalla tanto requerimientos funcionales como no funcionales involucrados en el desarrollo de la app móvil. Para cada requerimiento funcional en el proyecto, por medio de diversos diagramas, se explica su relevancia y su significado, y se muestra la interacción que conllevan en el sistema.

4.1.2 DESCRIPCIÓN GENERAL

4.1.2.1 Perspectiva del producto

- La app móvil forma parte del sistema Q-Bus.
- La app se desarrollará para la plataforma iOS versión 7 o superior.
- La app almacenará toda la información de las rutas.

- La app utilizará la geo posición del usuario para cumplir con una o más funcionalidades propuestas.
- La app hará uso del GPS y módulo Bluetooth del dispositivo móvil.

4.1.2.2 Características de los usuarios

Los usuarios a los cuales está destinada esta aplicación deberán estar familiarizados con el uso de smartphones y aplicaciones móviles para la plataforma iOS. No existe ningún tipo de restricción de género o edad.

4.1.2.3 Restricciones

1. La app estará disponible únicamente para dispositivos iPhone e iPod Touch con SO iOS 7 o superior.
2. Para el desarrollo de la aplicación se utilizará el lenguaje de programación ObjectiveC y el IDE Xcode 8.
3. La aplicación no podrá actualizar su información si no dispone de conexión a internet.
4. La aplicación no podrá detectar beacons y/o códigos QR que no se encuentren registrados en el sistema.
5. El prototipo funcional podrá reconocer un máximo de 5 paradas.
6. La precisión de la ubicación del usuario dependerá del dispositivo sobre el cual se ejecuta la aplicación.

4.1.2.4 Suposiciones y dependencias

1. Todos los beacons deberán tener un mismo UUID.
2. Los códigos QR deberán contener la información requerida por la aplicación para su correcta detección.
3. Los dispositivos en los cuales se instale la aplicación deben cumplir con los requisitos antes mencionados para garantizar una ejecución correcta de la misma.

4.1.3 REQUISITOS ESPECÍFICOS

4.1.3.1 Requisitos funcionales

Tabla 9. Requisitos funcionales app móvil Q-Bus

Identificador	Nombre	Descripción	RNF	Prioridad
RF01	Detección del estado del Bluetooth del dispositivo	La aplicación debe ser capaz de detectar si el dispositivo cuenta con tecnología BLE y si el Bluetooth está encendido o no.		Alta
RF02	Detección de beacons	La aplicación debe ser capaz de detectar beacons que se encuentren registrados en el sistema y mostrar la información que corresponda a la parada asociada al beacon.		Alta
RF03	Consulta de información y almacenamiento	La aplicación debe ser capaz de consultar al servidor toda la información concerniente a las paradas y rutas registradas en el sistema. Además debe almacenar dicha información en la memoria del dispositivo.	RNF04	Alta
RF04	Detección de códigos QR	La aplicación debe ser capaz de detectar códigos QR mediante la cámara del dispositivo y mostrar la información que corresponda a la parada asociada a dicho código.		Alta
RF05	Listado y búsqueda de cooperativas	La aplicación debe ser capaz de listar todas las cooperativas de buses registradas en el sistema y permitir hacer búsquedas por el nombre.	RNF02	Alta
RF06	Detalle de ruta	Se debe presentar el detalle de la ruta que el usuario haya seleccionado. Se debe dibujar en un mapa el recorrido que realiza el bus y las paradas por las cuales transita.	RNF02	Alta
RF07	Sugerencia de buses	La aplicación debe ser capaz de entregar sugerencias de buses que un usuario puede tomar para llegar desde su posición actual hasta un destino dentro de la ciudad de Quito.	RNF02	Alta

4.1.3.2 Requisitos no funcionales

Tabla 10. Requisitos no funcionales app móvil Q-Bus

Identificador	Nombre	Descripción	Prioridad
RNF01	Interfaz de usuario	La aplicación presentará una interfaz de usuario sencilla e intuitiva acorde a las iOS Human Interface Guidelines.	Alta
RNF02	Desempeño	Los datos almacenados por la aplicación podrán ser consultados y mostrados permanentemente sin que afecte a la fluidez normal de la aplicación.	Alta
RNF03	Rendimiento	Se debe garantizar que el uso de la aplicación no afecte al desempeño del dispositivo, ni consuma excesivos recursos como procesador, memoria o tráfico de la red.	Alta
RNF04	Disponibilidad	Se debe garantizar que el usuario pueda utilizar todas las funcionalidades de la aplicación independientemente de la conexión a internet del dispositivo.	Alta

4.1.3.3 Requisitos comunes de las interfaces

Interfaces de usuario

La interfaz de usuario estará optimizada para las pantallas de dispositivos iPhone e iPod Touch. Se seguirá las indicaciones mencionadas en el documento iOS Human Interface Guidelines para asegurar el correcto diseño para la plataforma iOS.

Interfaces de hardware

La aplicación podrá ser ejecutada con todas sus funcionalidades en dispositivos iPhone 4S o superior e, iPod Touch 5ta generación o superior debido al requisito de utilización de BLE.

Interfaces de software

Es necesario que los dispositivos sobre los cuales se va a ejecutar la aplicación cuenten con un SO iOS 7 o superior.

Interfaces de comunicación

La aplicación se conectará al servidor a través de un servicio web basado en el intercambio de mensajes JSON utilizando el protocolo HTTP.

4.2 DISEÑO DEL PROTOTIPO INICIAL

El diseño es la segunda fase de la metodología de desarrollo basada en prototipos. En esta sección se presentan diagramas de secuencia, diagramas de clase y diseño de navegación que utiliza la aplicación móvil Q-Bus.

4.2.1 DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia se los desarrolló con el fin de comprender de mejor manera el proceso que debe realizar la aplicación móvil para cumplir con las principales funcionalidades.

Las figuras 4 y 5 muestran la primera funcionalidad de la aplicación, la cual se basa en el inicio de la aplicación y actualización de la información que mostrará cuando dispone de una conexión a internet y cuando no la tiene respectivamente.

Cuando el usuario ingresa a la funcionalidad del buscador de beacons, la aplicación inicia el monitoreo para encontrar uno cercano. En el caso de que sea detectado, la aplicación recibe el UUID y en base al identificador realiza una búsqueda en los datos que tiene almacenado internamente el dispositivo para presentar la información de la ruta correspondiente. Si no existe ningún registro con dicho identificador se muestra un mensaje al usuario indicando lo sucedido. El mismo proceso sigue la funcionalidad de captura de códigos QR. Ambos procesos se detallan en las figuras 6 y 7 respectivamente.

La Figura 8 presenta la secuencia que sigue la aplicación para mostrar toda la información disponible bajo la funcionalidad “Buscador”, mientras que la Figura 9 muestra el proceso para la funcionalidad “Cómo Llego”. Ambas funcionalidades serán explicadas con mayor detalle en el siguiente apartado.

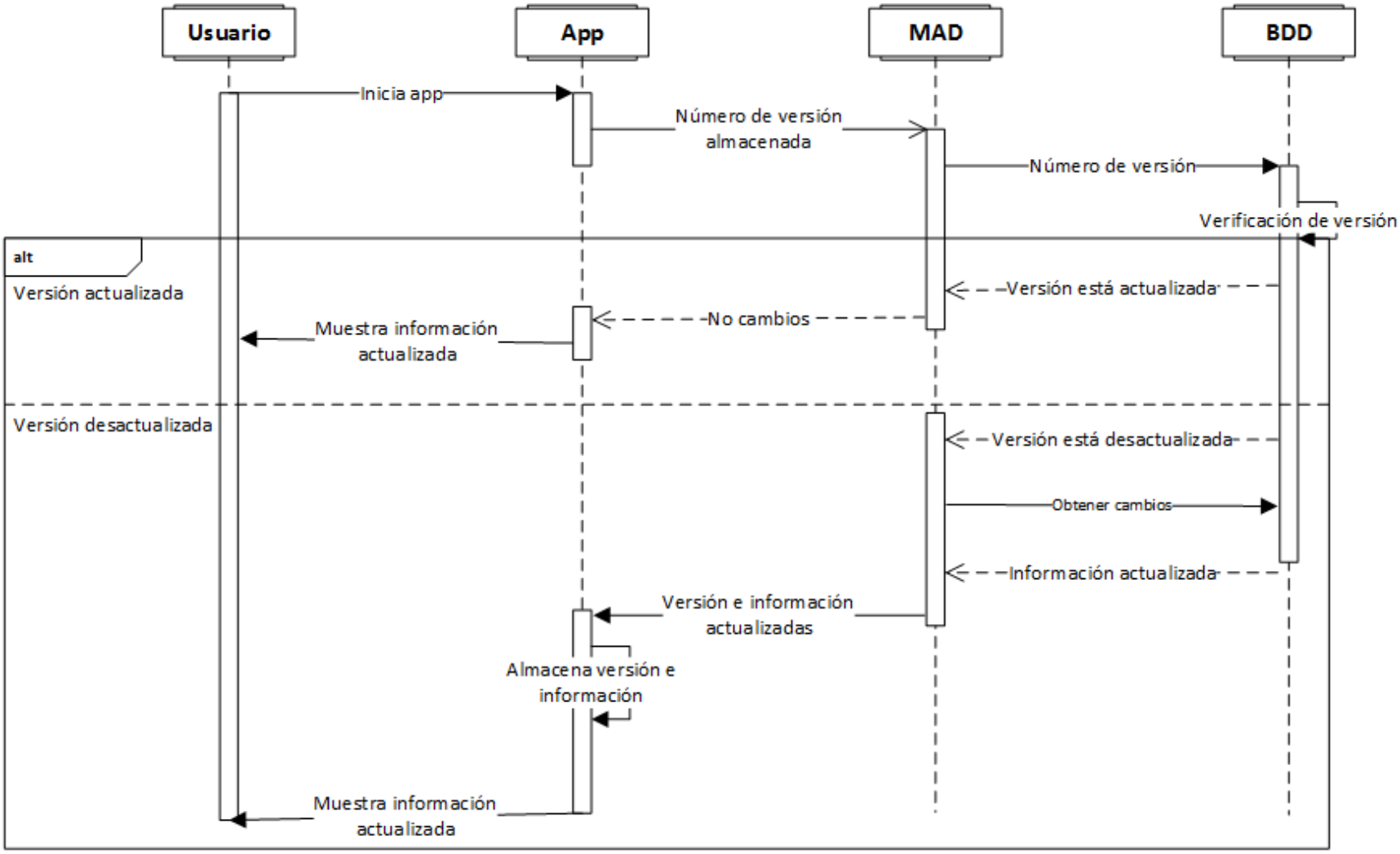

Figura 4. Diagrama de secuencia: Inicio con conexión a internet

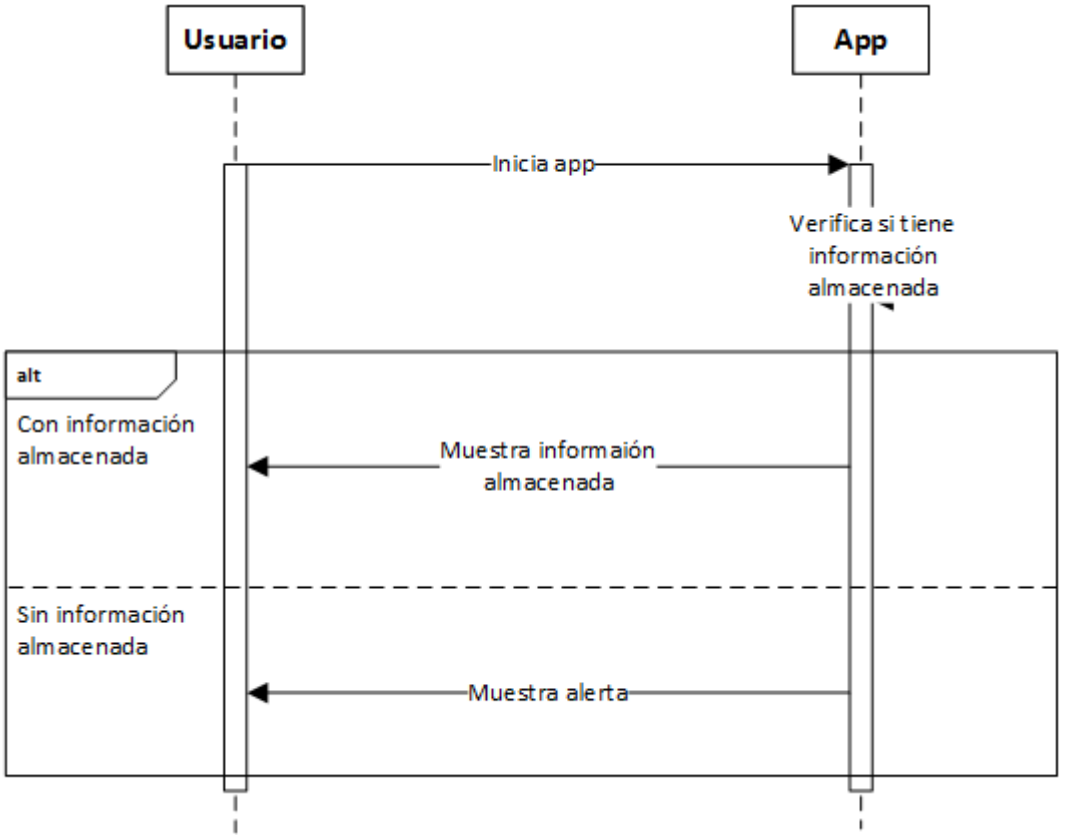

Figura 5. Diagrama de secuencia: Inicio sin conexión a internet

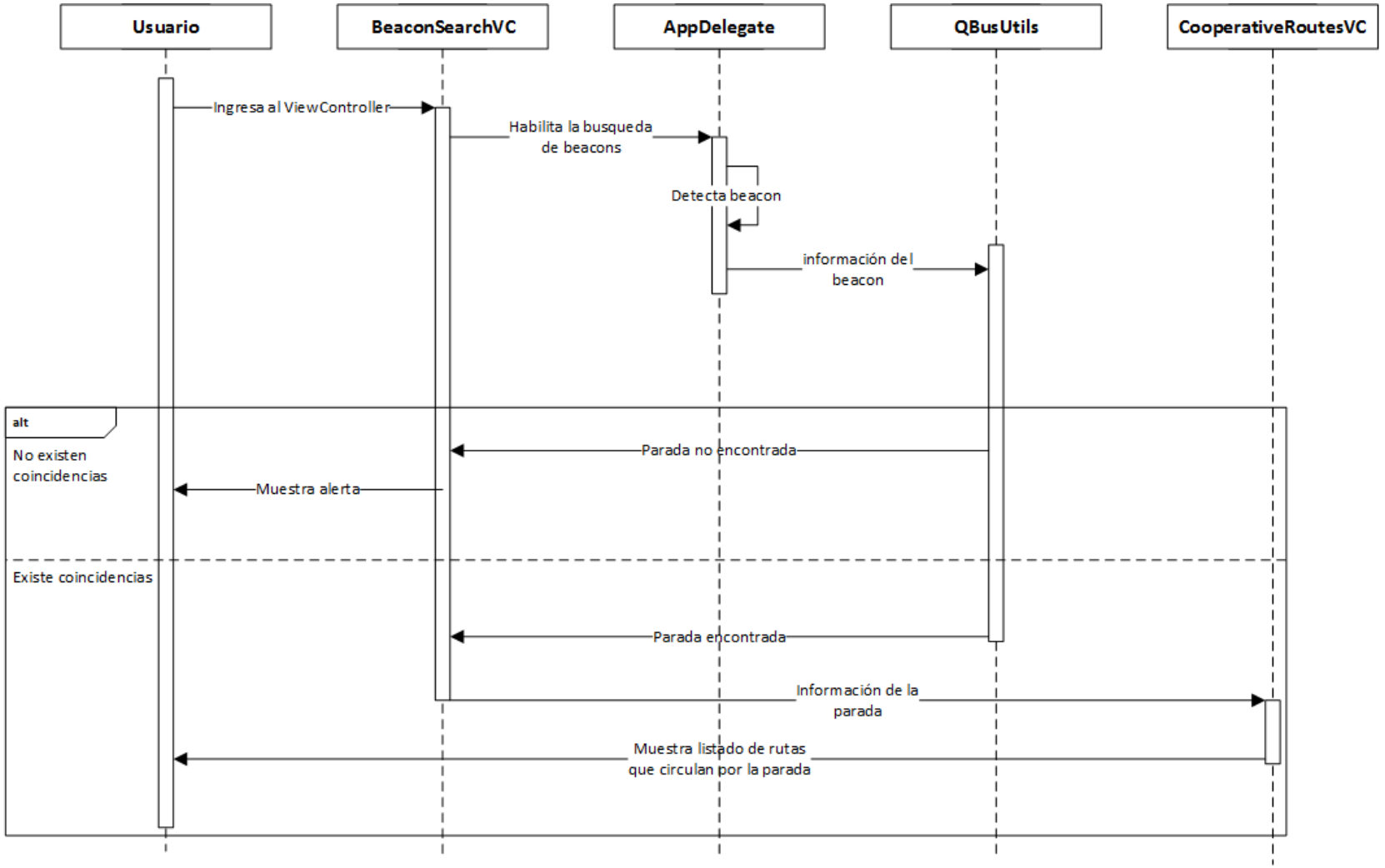

Figura 6. Diagrama de secuencia: Detección de beacon

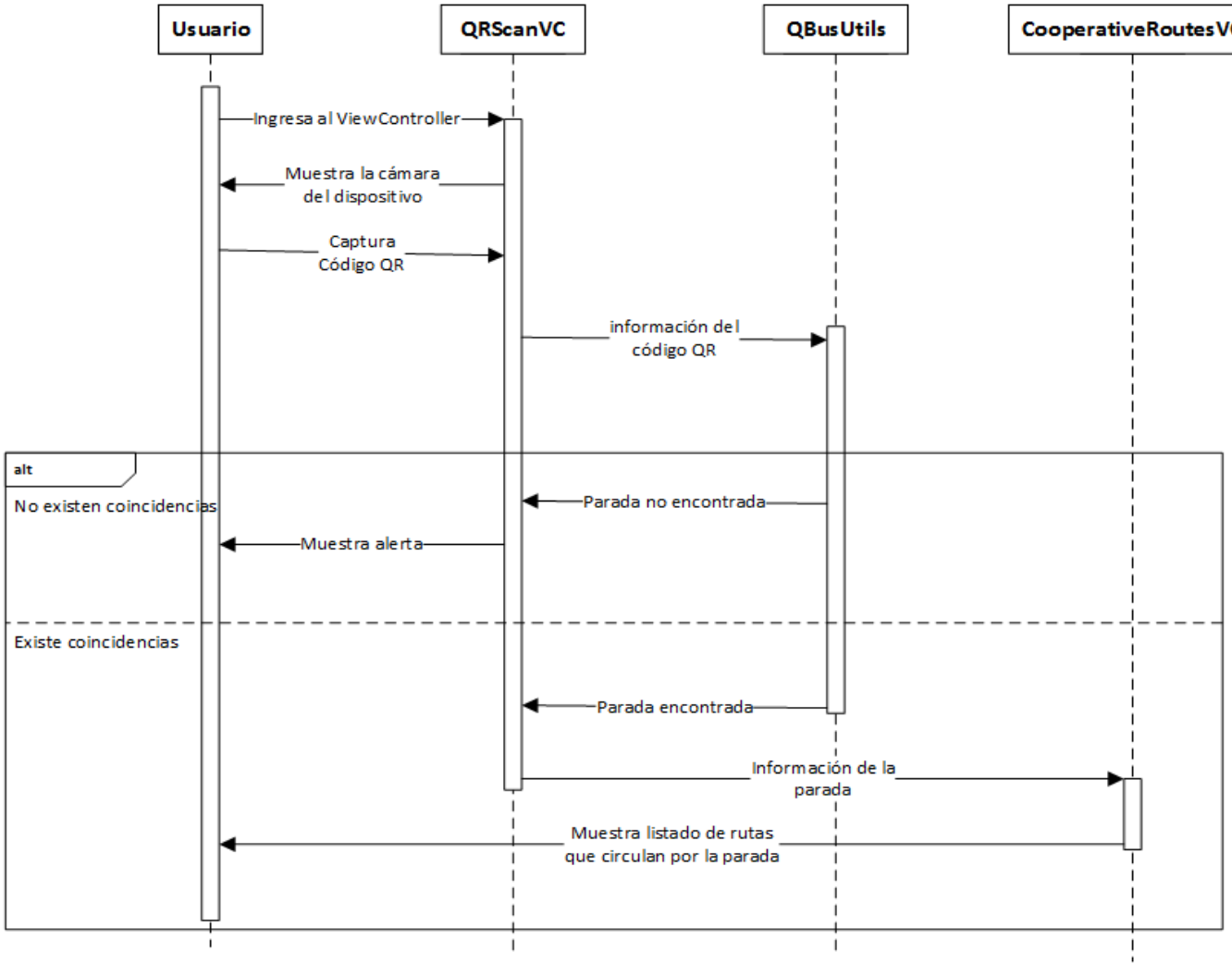

Figura 7. Diagrama de secuencia: Captura de código QR

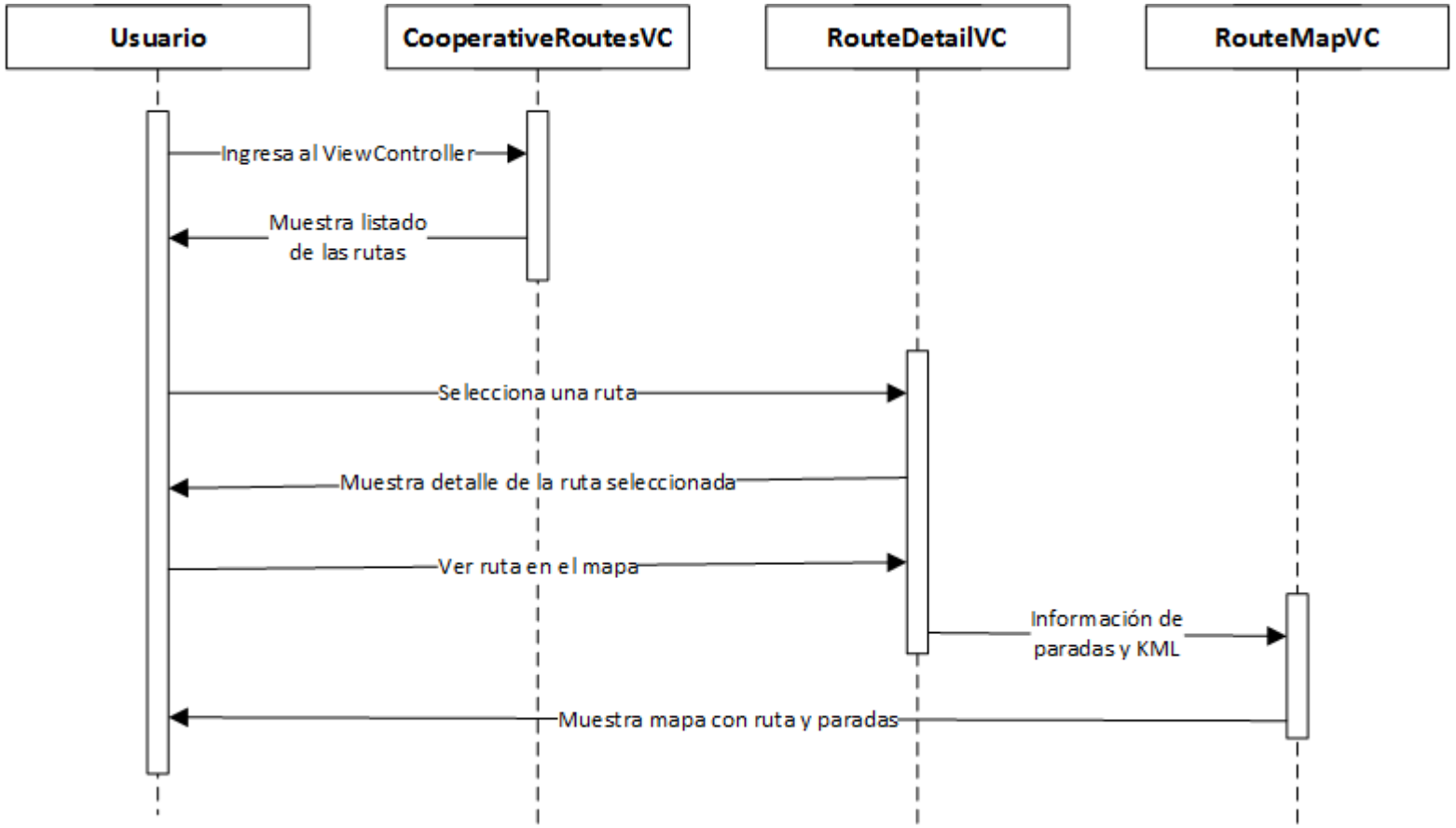

Figura 8. Diagrama de secuencia: Detalle de una ruta

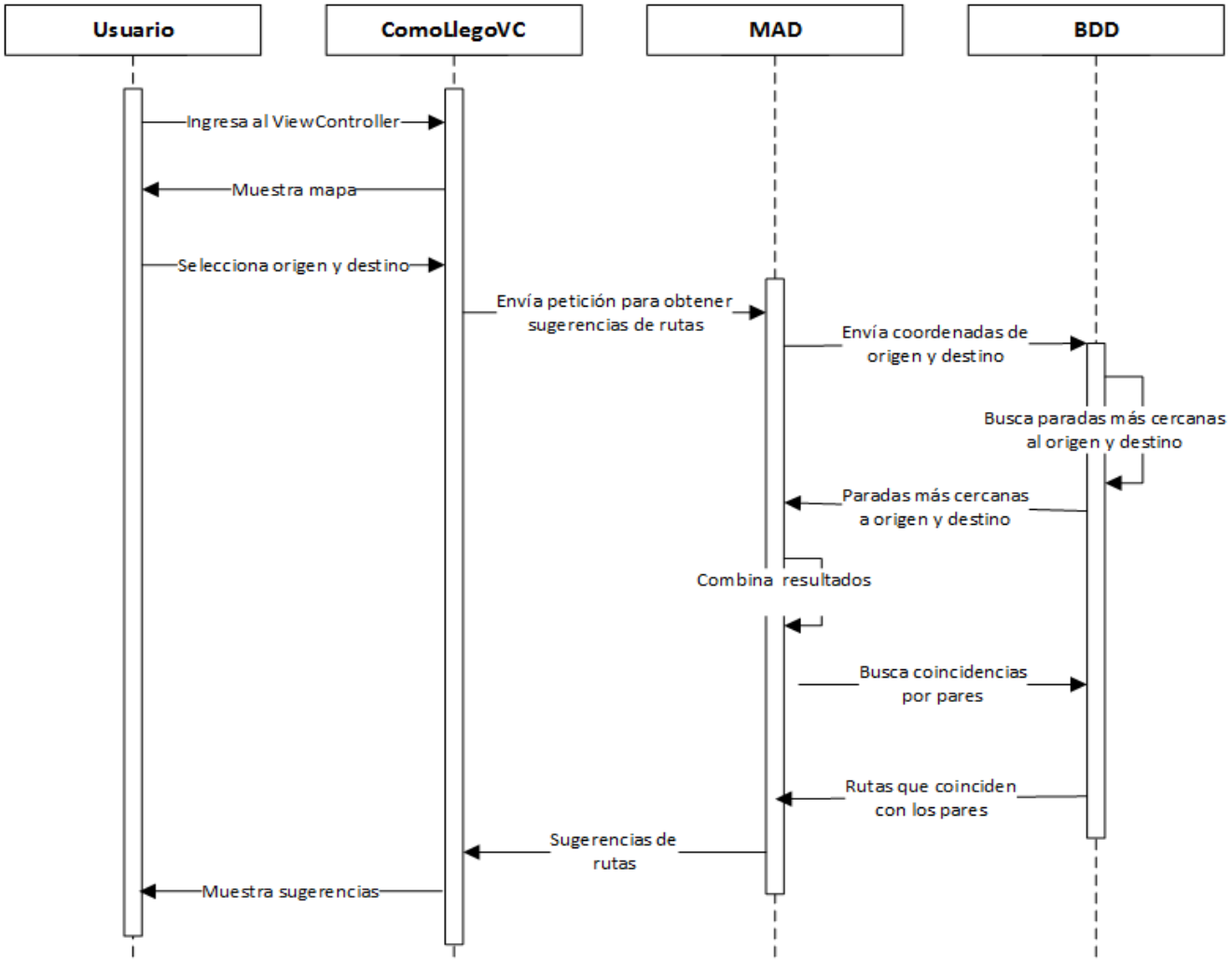

Figura 9. Diagrama de secuencia: Sugerencias de rutas

4.2.2 STORYBOARD PROTOTYPING

En la actual sección se presenta el prototipo realizado utilizando la técnica de storyboard. El fin del desarrollo de este prototipo es comprender el flujo que sigue la aplicación y cómo ésta soporta cada acción que realice el usuario de la aplicación.

La Figura 10 indica el inicio de la aplicación. La aplicación utiliza una navegación a través de pestañas y como pantalla principal se encuentra el buscador de rutas, el cual presentará un listado de todas las cooperativas de buses registradas en el sistema. Una vez que el usuario seleccione cualquier ítem de la lista, se mostrará un listado con todas las rutas que pertenecen a la cooperativa escogida. Posteriormente el usuario podrá obtener información completa de las rutas y las podrá ver trazadas en un mapa digital.

Las pestañas 2 y 3 representan el buscador de beacons y escáner de códigos QR respectivamente. Una vez que la aplicación detecte un beacon o un código QR se presentará un listado de las rutas de buses que circulen por la parada asociada a dicho beacon o código. Dicho proceso es ilustrado en la Figura 11.

En la Figura 12 muestra la disposición gráfica de la pantalla “Cómo Llego” la cual permitirá seleccionar dos puntos en el mapa que serán tomados como origen y destino para proporcionar sugerencias de rutas que podrá tomar el usuario.

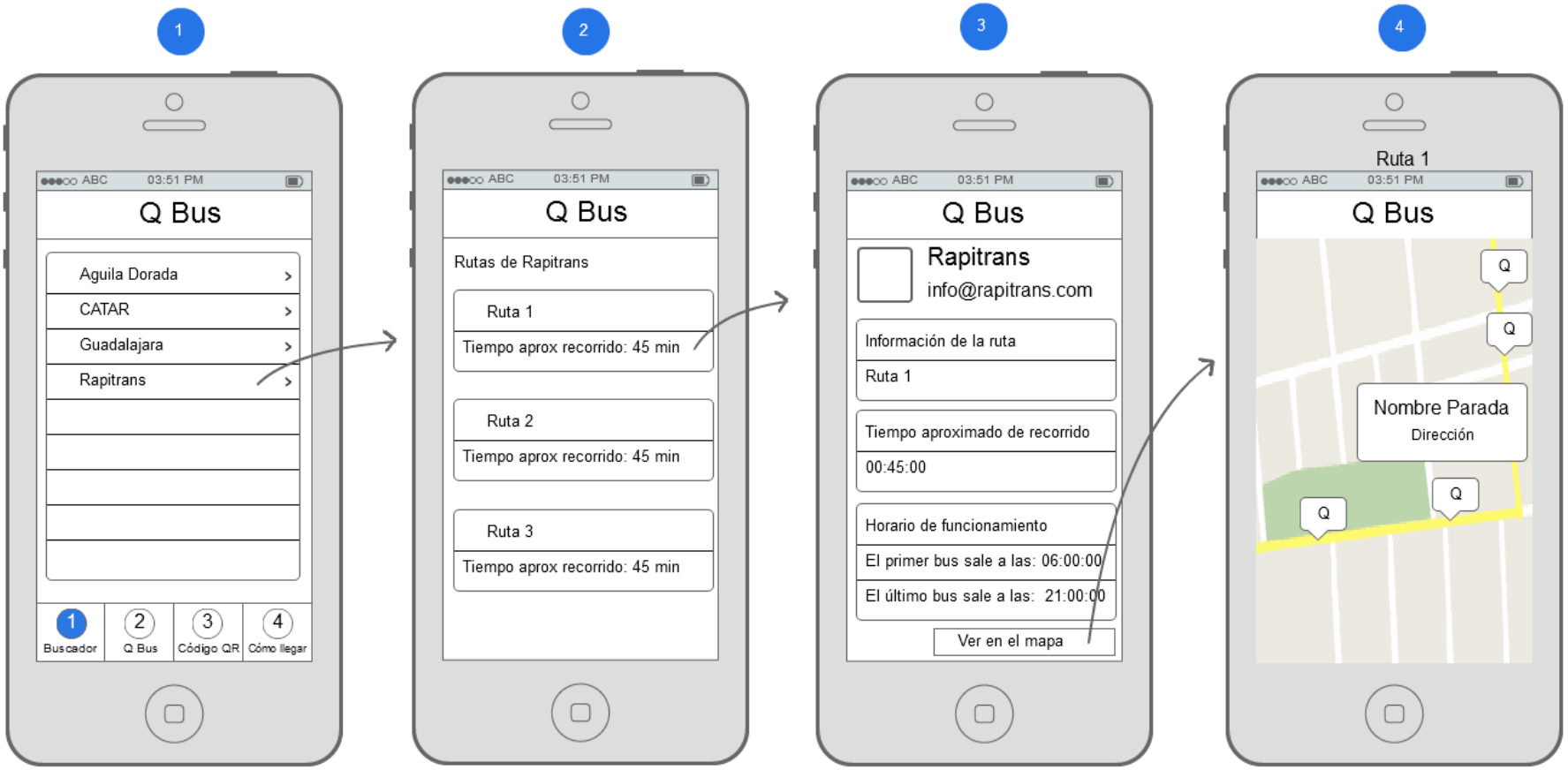

Figura 10. Storyboard: Buscador de cooperativas

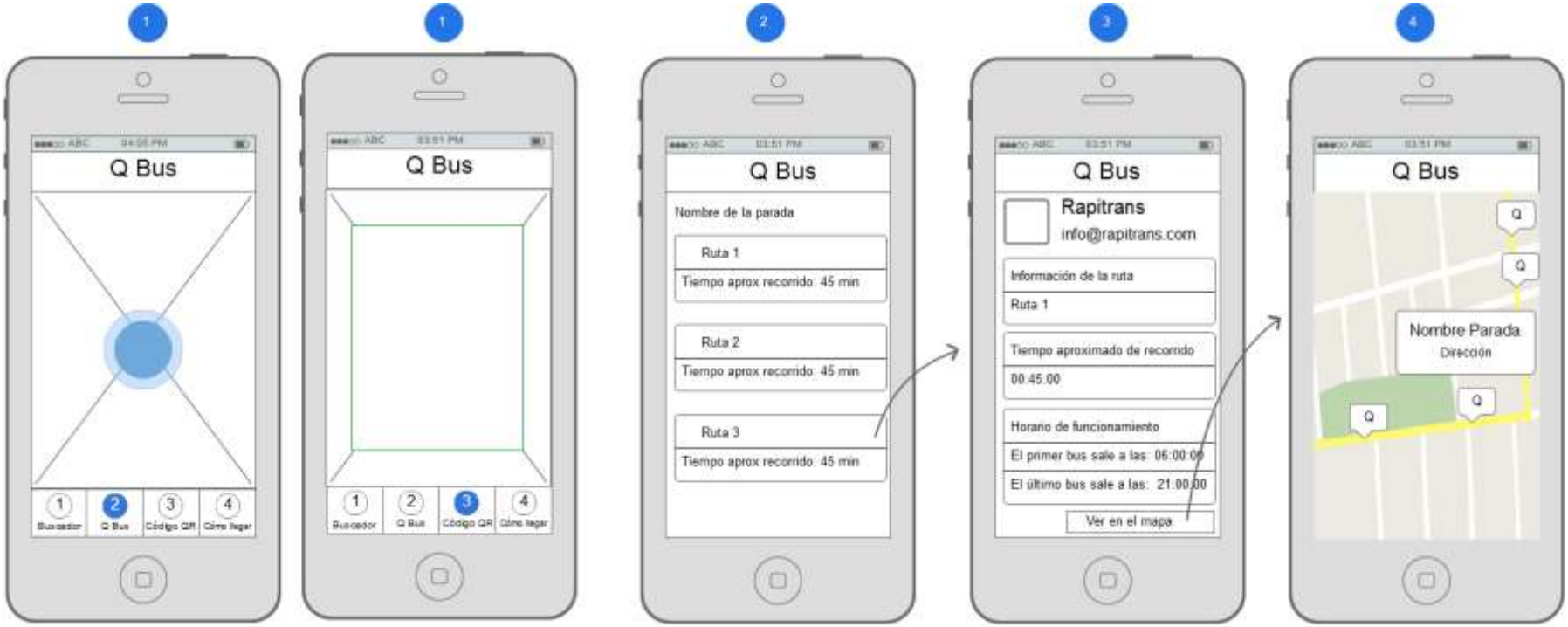

Figura 11. Storyboard: Detección de beacons y escáner de códigos QR

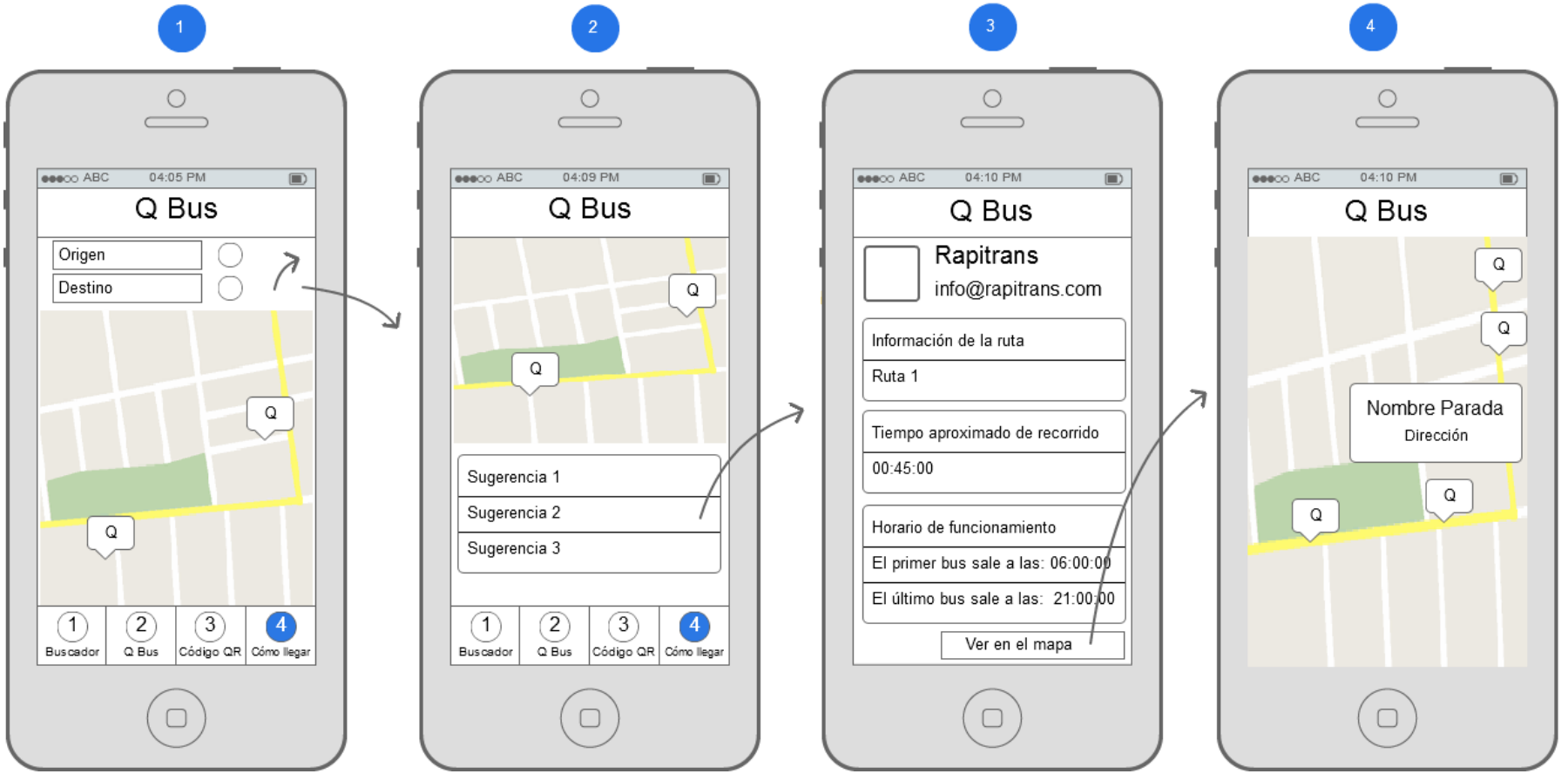

Figura 12. Storyboard: Sugerencia de rutas mediante “Cómo Llego”

4.2.3 DIAGRAMA DE CLASES

En las figuras 13, 14 y 15 se presenta el diagrama de clases generado con la herramienta OmniGraffle mediante la incorporación del proyecto desarrollado en Xcode.

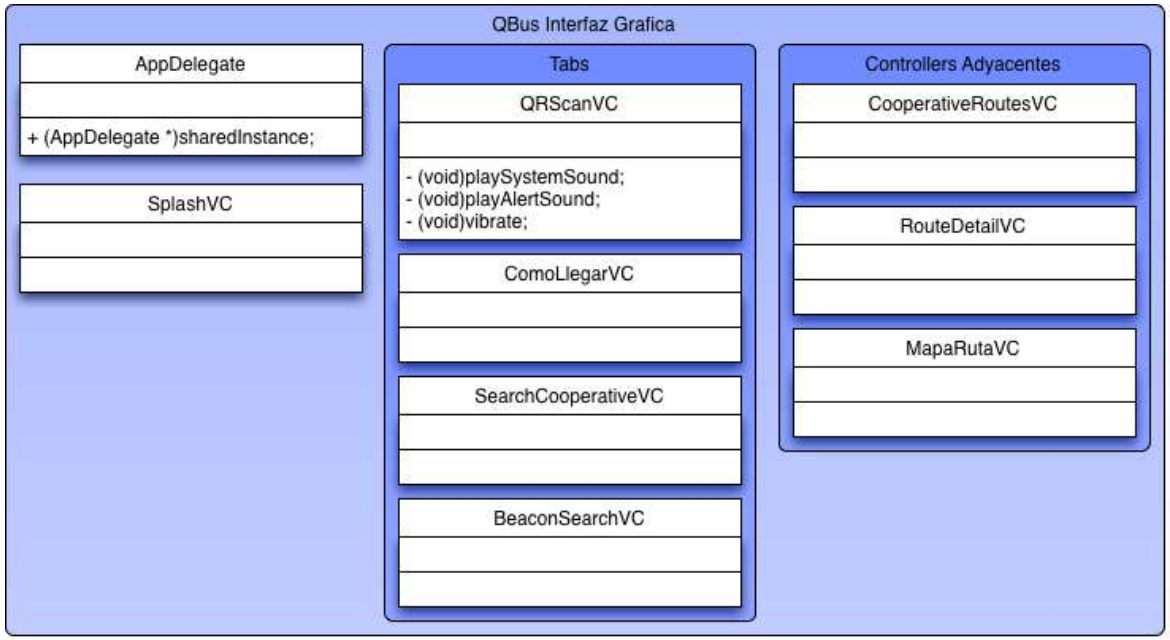

Figura 13. Diagrama de clases Q-Bus. Interfaz gráfica

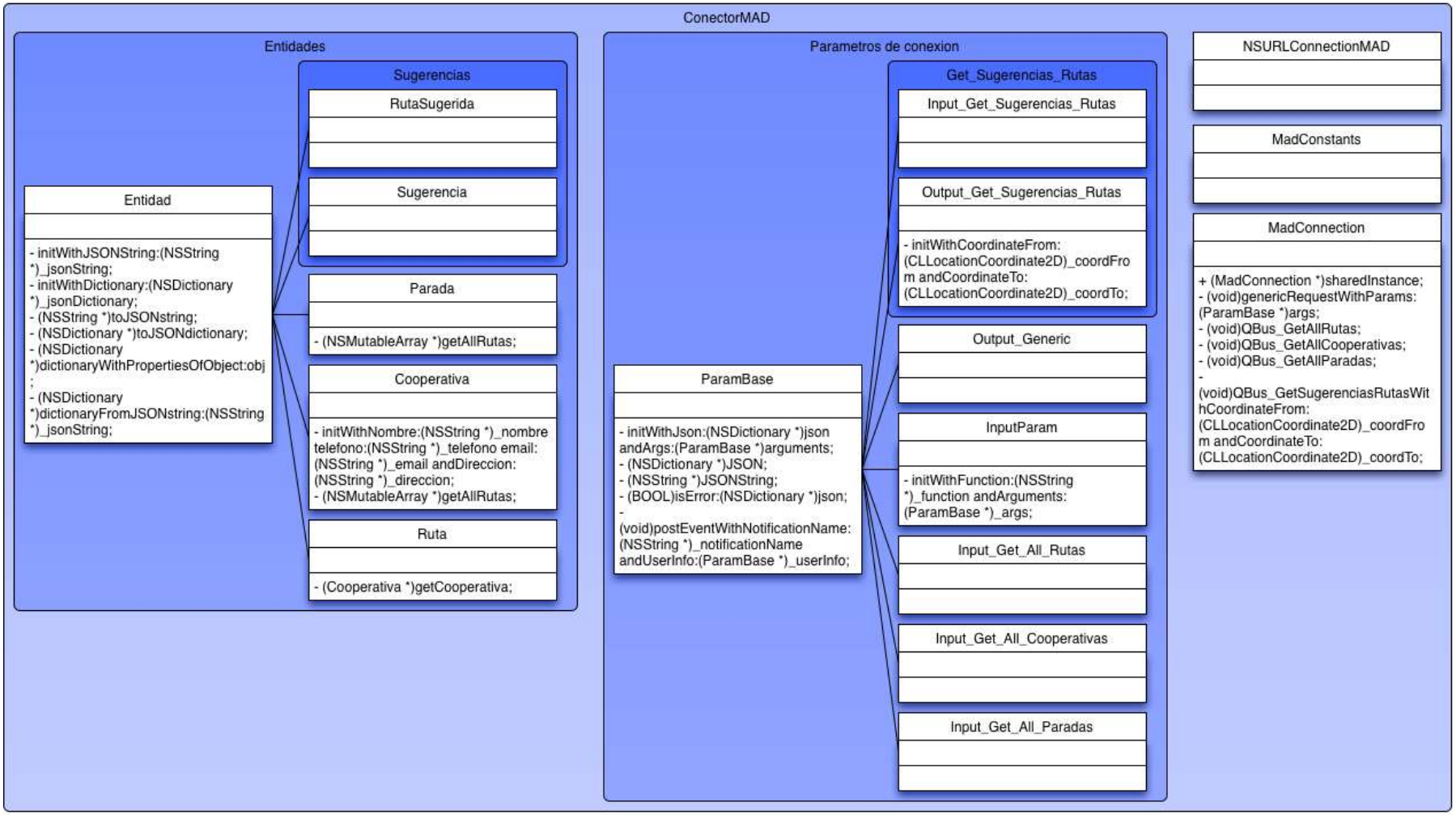

Figura 14. Diagrama de clases Q-Bus. Conector MAD

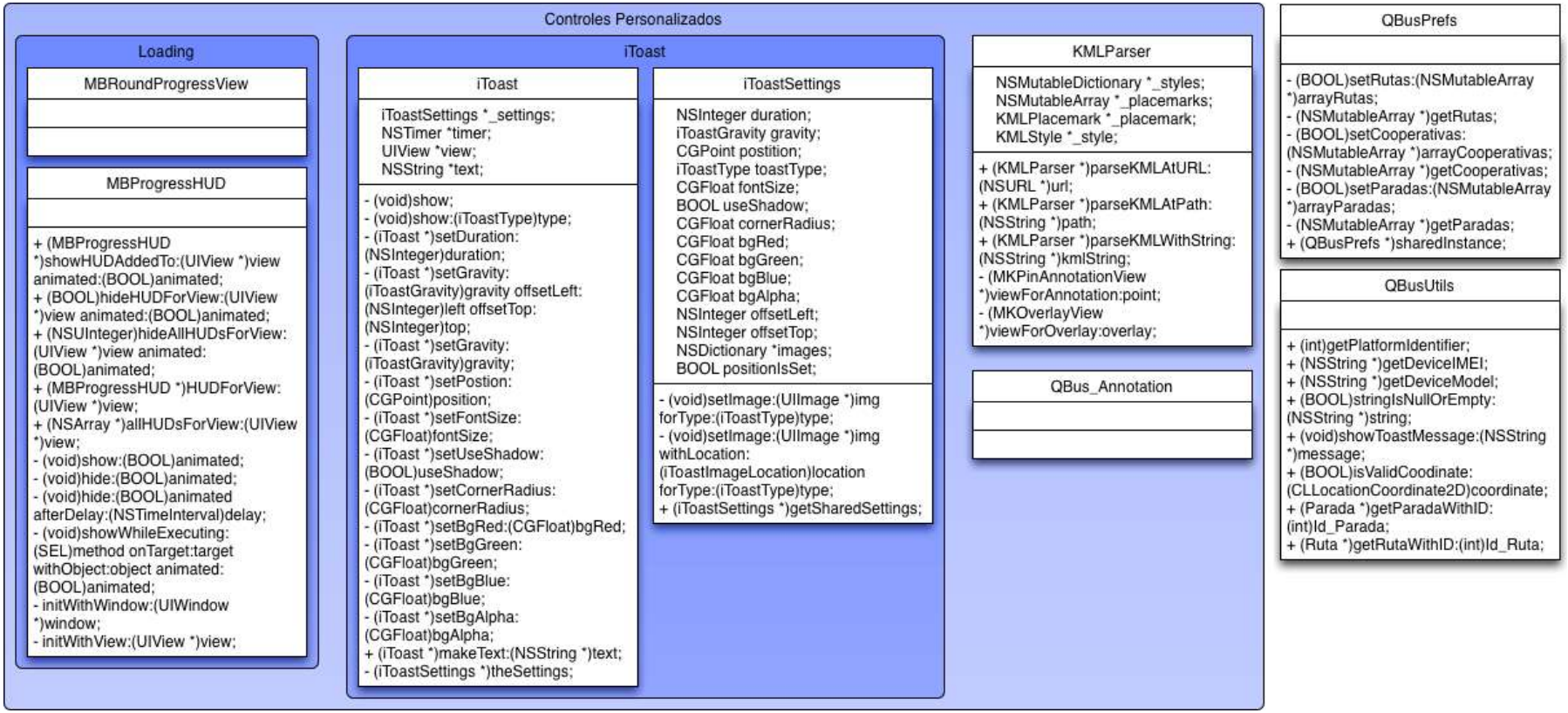

Figura 15. Diagrama de clases Q-Bus. Controles personalizados

4.3 IMPLEMENTACIÓN DEL PROTOTIPO INICIAL

4.3.1 DESCRIPCIÓN DE LA ESTRUCTURA DEL PROYECTO

La aplicación se desarrolló para el sistema operativo iOS con versión 7 o superior debido a su compatibilidad con la funcionalidad de BLE. Consta de 26 clases que se encuentran en los siguientes grupos:

- Controles Personalizados
- QBusPrefs
- QBusUtils
- Conector MAD
- QBus Interfaz Gráfica

La Figura 16 muestra los grupos y subgrupos del proyecto de Xcode.

Figura 16. Grupos en el proyecto en Xcode

4.3.2 DESCRIPCIÓN DE GRUPOS Y CLASES

Controles personalizados

- iToast

Permite presentar una alerta en la pantalla que desaparece automáticamente. Pretende emular las notificaciones Toast del sistema operativo Android.

- QBus_Annotation

Representa a una anotación, marca o pin que se mostrará en el mapa digital.

QBusPrefs

- QBusPrefs

Permite interactuar con las preferencias del sistema. Además de almacenar y recuperar información desde la memoria del sistema.

QBusUtils

- QbusUtils

Contiene métodos utilitarios que son invocados a lo largo de la aplicación.

Conector MAD

- MadConnection

Clase que permite realizar la petición al servidor y analizar la respuesta recibida.

La Figura 17 presenta el método para la creación de las peticiones HTTP a ser enviadas al módulo de acceso a datos.

```
-(void) createRequest: (InputParam *) args
{
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:
                                    [NSURL URLWithString: QBUS_URL_WEB_SERVICE]];

    //Especifico el uso del metodo POST
    request.HTTPMethod = @"POST";

    //Tipo de contenido
    [request setValue:@"application/x-www-form-urlencoded; charset=utf-8"
            forHTTPHeaderField:@"Content-Type"];

    // Armo el body de la peticion http
    NSString *stringData = [NSString stringWithFormat:@"PARAM=%@", [args JSONString]];

    NSData *requestBodyData = [stringData dataUsingEncoding:NSUTF8StringEncoding];
    request.HTTPBody = requestBodyData;

    NSURLConnectionMAD *connection = [[NSURLConnectionMAD alloc]
                                      initWithRequest:request delegate:self];
    connection.args = args;

    [connection start];

    if (arrayConnectionsPool == nil || !arrayConnectionsPool)
        arrayConnectionsPool = [NSMutableArray array];

    [arrayConnectionsPool addObject:connection];

    if (!connection)
        _responseData = nil;
}
```

Figura 17. MaddConnection. Método *createRequest*

- MadConstants

Contiene declaradas las etiquetas que serán utilizadas como protocolo en las peticiones y respuestas entre la aplicación y el módulo de acceso a datos.

- NSURLConnectionMAD

Modificación del objeto “NSURLConnection” que permite obtener la referencia a una petición HTTP. Se incluyó los argumentos que conforman cada petición.

- Entidad

Representa a una tabla de la base de datos.

- Ruta (Entidad)

Contiene toda la información correspondiente a una ruta registrada en el sistema.

- Cooperativa (Entidad)

Contiene toda la información correspondiente a una cooperativa registrada en el sistema.

- Parada (Entidad)

Contiene toda la información correspondiente a una parada registrada en el sistema.

- ParamBase

Clase que contiene métodos para la transformación del objeto en JSON y viceversa.

- InputParam (ParamBase)

Clase que representa a los argumentos de una petición HTTP realizada al módulo de acceso a datos o a la respuesta recibida del mismo.

- Input_Get_All_Paradas (InputParam)

Clase para manipular la respuesta recibida del servidor cuando se recuperan todas las paradas registradas en el sistema.

- Input_Get_All_Cooperativas (InputParam)

Clase para manipular la respuesta recibida del servidor cuando se recuperan todas las cooperativas registradas en el sistema.

- Input_Get_All_Rutas (InputParam)

Clase para manipular la respuesta recibida del servidor cuando se recuperan todas las rutas registradas en el sistema.

- Output_Generic (InputParam)

Clase que representa a los parámetros a enviar en una petición HTTP.

- Input_Get_Sugerencias_Rutas (InputParam)

Clase para manipular la respuesta recibida del servidor cuando se recuperan las sugerencias de rutas para ir desde un punto a otro.

- Output_Get_Sugerencias_Rutas (InputParam)

Clase que representa a los parámetros a enviar en la petición HTTP correspondiente a la obtención de sugerencias para llegar de un lugar a otro.

QBus Interfaz Gráfica

- AppDelegate

Clase singleton de la aplicación. Esta clase permite controlar eventos que pasan en la aplicación, como por ejemplo cuando es iniciada, finalizada, entra a background o recibe una notificación.

- SplashVC

Pantalla inicial de la aplicación. En este View Controller se realizan las peticiones HTTP para actualizar la información a presentar al usuario.

- SearchCooperativeVC

View Controller que muestra al usuario el listado de las cooperativas registradas en el sistema. Además permite realizar búsquedas por el nombre de las cooperativas para mayor facilidad de navegación dentro de la app.

- BeaconSearchVC

Pantalla en la cual se muestra una animación cuando la aplicación está en busca de nuevos beacons. Muestra además mensajes cuando el dispositivo tenga apagado el Bluetooth o la versión del mismo no sea compatible con BLE.

La Figura 18 muestra el método centralManagerDidUpdateState, el cual permite a la app identificar si el dispositivo es compatible con la tecnología BLE y el estado del Bluetooth.

```
- (void)centralManagerDidUpdateState:(CBCentralManager *)central
{
    switch ([central state])
    {
        case CBCentralManagerStateUnsupported:
            hasBLE = NO;
            NSLog(@"Este dispositivo no soporta Bluetooth Low Energy.");
            break;
        case CBCentralManagerStateUnauthorized:
            hasBLE = NO;
            NSLog(@"Esta app no esta autorizada para usar Bluetooth Low Energy.");
            break;
        case CBCentralManagerStatePoweredOff:
            hasBLE = NO;
            NSLog(@"Bluetooth esta desactivado.");
            break;
        case CBCentralManagerStateResetting:
            hasBLE = NO;
            NSLog(@"El BLE Manager esta reiniciando...");
            break;
        case CBCentralManagerStatePoweredOn:
            hasBLE = YES;
            NSLog(@"Bluetooth LE esta activado y listo.");
            break;
        case CBCentralManagerStateUnknown:
            hasBLE = NO;
            NSLog(@"El estado de BLE Manager es desconocido.");
            break;
        default:
            hasBLE = NO;
            NSLog(@"El estado de BLE Manager es desconocido.");

    }

    [self updateView];
}
```

Figura 18. AppDelegate. Método centralManagerDidUpdateState

La Figura 19 muestra el método didRangeBeaconsInRegion, el cual se ejecuta cada vez que la app detecta un beacon. Una vez que el beacon se encuentre a una distancia menor o igual a 10 cm del dispositivo, la app mostrará el detalle de la parada correspondiente al beacon.

```
- (void) beaconManager:(ESTBeaconManager *)manager didRangeBeacons:(NSArray *)beacons
              inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] > 0)
    {
        nearBeaconFound = false;

        CLBeacon *nearestExhibit = [beacons firstObject];

        if (nearestExhibit.proximity == CLProximityImmediate)
        {
            if (nearestExhibit.accuracy <= 0.1)
            {
                if([mainBeacon.major intValue] != [nearestExhibit.major intValue]  &&
                   [mainBeacon.minor intValue] != [nearestExhibit.minor intValue])
                {
                    mainBeacon = nearestExhibit;
                }
            }
        }
    }
}
```

Figura 19. AppDelegate. Método didRangeBeaconsInRegion

- QRScanVC

Pantalla que permite al usuario capturar un código QR. La Figura 20 muestra el método didReadSymbolsFromImage que se ejecuta cuando es detectado un código QR. Si la información que el código contiene coincide con alguna parada se muestra el detalle de la misma.

```
-(void) readerView:(ZBarReaderView *)readerView didReadSymbols:(ZBarSymbolSet *)symbols
         fromImage:(UIImage *)image
{
    ZBarSymbol * s = nil;

    NSString *result = nil;
    for (s in symbols)
    {
        NSLog(@"2 - %@", s.data);
        result = [NSString stringWithFormat:@"%@", s.data];
    }

    NSArray *arrayValueQR = [result componentsSeparatedByString:@";"];

    int minor = [[arrayValueQR objectAtIndex:1] intValue];

    Parada *parada = [QBusUtils getParadaWithID:minor];

    if (parada)
    {
        CooperativeRoutesVC *cooperativeRoutesVC =
                [[AppDelegate sharedInstance].mainStoryboard
                 instantiateViewControllerWithIdentifier:@"CooperativeRoutesVC"];

        cooperativeRoutesVC.parada = parada;

        [self.navigationController pushViewController:cooperativeRoutesVC animated:YES];
    }
}
```

Figura 20. QRScanVC. Método didReadSymbolsFromImage

- ComoLlegarVC

Pantalla que permite al usuario seleccionar dos puntos en un mapa digital, origen y destino, para obtener sugerencias de la aplicación sobre qué buses tomar para transportarse.

- CooperativeRoutesVC

Pantalla que muestra al usuario un listado de todas las rutas que tiene una cooperativa o las rutas que circulan por determinada parada de buses.

- RouteDetailVC

View Controller que muestra el detalle de una ruta seleccionada por el usuario.

Cuando el usuario ingrese a esta pantalla se mostrará la información correspondiente a la ruta seleccionada mediante el método que se muestra en la Figura 21.

```
- (void)viewDidLoad
{
    [super viewDidLoad];


    lblNombreCooperativa.text = cooperativa.Nombre_Cooperativa;
    lblEmailCooperativa.text = cooperativa.Email_Cooperativa;

    lblNombreRuta.text = ruta.Nombre_Ruta;
    lblTiempoRecorridoRuta.text = ruta.TiempoRecorrido_Ruta;
    lblHoraInicioRuta.text = ruta.HoraInicio_Ruta;
    lblHoraFinRuta.text = ruta.HoraFin_Ruta;
}
```

Figura 21. RouteDetailVC. Método viewDidLoad

- MapaRutaVC

Pantalla en la cual se muestra un mapa digital con el recorrido que sigue el bus y las paradas por las cuales circula. La Figura 22 muestra el método viewDidLoad en el cual se genera el contenido de tipo KML para que el mapa pueda dibujar la ruta en base a los puntos que conforman el recorrido.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;

    // Solicito autorizacion para acceder a la posicion del usuario
    if ([locationManager respondsToSelector:@selector(requestAlwaysAuthorization)])
        [locationManager requestAlwaysAuthorization];

    kml = [KMLParser parseKMLWithString: ruta.KML_Ruta];
    overly = [[NSMutableArray alloc] init];

    arregloOverlays = [kml overlays];       //guarda en el arreglo los Overlays del .kml
    [mapView addOverlays:arregloOverlays];     //agrega overlays al mapa por el delegado


    arrayParadas = [[QBusPrefs sharedInstance] getParadas];

    for (Parada *parada in arrayParadas)
    {
        CLLocationCoordinate2D coord =
        CLLocationCoordinate2DMake([parada.Latitud_Parada floatValue],
                                   [parada.Longitud_Parada floatValue]);

        QBus_Annotation *annotation = [[QBus_Annotation alloc] init];
        [annotation setCoordinate:coord];
        [annotation setTitle:parada.Nombre_Parada];
        [annotation setParada:parada];
        [mapView addAnnotation:annotation];
    }
}
```

Figura 22. MapaRutaVC. Método viewDidLoad

4.4 PRUEBAS Y MEJORA DEL PROTOTIPO

Las pruebas de Q-Bus se basaron en el enfoque de caja negra. Además, en los dispositivos móviles, lo que se busca principalmente es el correcto funcionamiento, rendimiento, y en iOS particularmente, un diseño de la interfaz gráfica amigable e intuitiva.

Las pruebas se han realizado mediante un protocolo de pruebas establecido y cuyos resultados se presentan a continuación.

Se ha decidido dividirlos en tres fases: Pruebas de requerimientos, pruebas de funcionalidad y, pruebas de resistencia y rendimiento.

4.4.1 PRUEBAS DE REQUERIMIENTOS

Para la realización de las pruebas de requerimientos, se evaluaron los requerimientos iniciales del proyecto y las funcionalidades básicas que debe cumplir una aplicación móvil de este tipo.

La Tabla 11 detalla si los requerimientos fueron cumplidos a satisfacción y los resultados obtenidos.

Tabla 11. Resultados de pruebas de requerimientos

Requerimientos	Cumplió con el requerimiento		Comentarios
	SI	NO	
Detección del estado del Bluetooth del dispositivo	X		Correcto
Detección de beacons	X		Correcto
Consulta de información y almacenamiento	X		Correcto
Detección de códigos QR	X		Correcto
Muestra listado de cooperativas y rutas por cooperativa	X		Correcto
Permite la búsqueda de cooperativas n base al nombre	X		Correcto
Muestra la información principal de las rutas	X		Correcto
Sugerencia de buses	X		Correcto

Los requerimientos funcionales del proyecto fueron cumplidos de manera correcta.

4.4.2 PRUEBAS DE FUNCIONALIDAD

En las pruebas de funcionalidad se buscó verificar si el funcionamiento de cada uno de los métodos, procedimientos y estados del sistema, era correcto y si se obtenían los resultados esperados. Los escenarios de las pruebas están basados en posibles casos de uso y en la mayor cantidad de acciones posibles que un usuario potencial pudiera realizar.

4.4.2.1 Etapa I: Conexión a Internet

Tabla 12. Resultados de pruebas funcionales: Etapa I

Acciones	Escenario	Funcionó correctamente		Comentarios
		SI	NO	
¿Inicia la aplicación sin problema?	1	X		En el caso de ser la primera vez que se inicia la app y no se ha almacenado información muestra alerta.
	2	X		
	3	X		
	4	X		
¿Muestra información actualizada?	1	X		En el escenario 4 muestra la última información descargada.
	2	X		
	3	X		
	4	X		
Muestra en un mapa digital la ruta que sigue el bus y las paradas por las cuales circula	1	X		En el escenario 4, si el mapa no ha sido cargado en memoria caché no se despliega la información en su totalidad.
	2	X		
	3	X		
	4	X		

Para esta etapa se tomaron en cuenta cuatro escenarios:

1. Dispone de conexión a internet vía WiFi y no datos móviles.
2. Dispone de conexión a internet por datos móviles y no por WiFi.
3. Dispone de conexión a internet por WiFi y por datos móviles.
4. No dispone de conexión a internet.

Se tomó en cuenta el manejo de la conexión a Internet como “Etapa I” debido a que es el medio de actualización de información de Q-Bus. Los resultados se muestran en la Tabla 12.

4.4.2.2 Etapa II: Interfaz gráfica y almacenamiento

En esta segunda etapa se ha tomado en cuenta principios básicos de diseño de una interfaz gráfica funcional. Los resultados se muestran en la Tabla 13.

Tabla 13. Resultados de pruebas funcionales: Etapa II

Actividades	Funcionó correctamente		Observaciones
	SI	NO	
Almacena correctamente la información de cooperativas y rutas	X		Correcto
Utiliza controles adecuados para cada funcionalidad.	X		Correcto
Presenta View Controllers de manera adecuada	X		Correcto
Notifica de manera adecuada cuando un error se ha producido	X		Correcto

4.4.3 CONTROL DE CALIDAD

El proceso de control de calidad se ha tomado en cuenta los siguientes dispositivos:

1. iPod Touch 5ª generación de 32 GB
2. iPhone 5S de 32 GB
3. iPhone 6 de 16 GB

4.4.3.1 Protocolo de aseguramiento de calidad

Tabla 14. Resultados del protocolo de aseguramiento de calidad

Actividades	Dispositivo iOS		
	1	2	3
Instalación en equipos	s/n	s/n	s/n
Obtener información vía WiFi	s/n	s/n	s/n
Obtener información vía EDGE, 3G, LTE	s/n	s/n	s/n
Verificar el funcionamiento del Bluetooth	s/n	s/n	s/n
Activar la localización del usuario	s/n	s/n	s/n
Detectar beacon y mostrar información asociada	s/n	s/n	s/n
Capturar código QR y mostrar información asociada	s/n	s/n	s/n
Buscar una cooperativa, ver sus rutas y detalles	s/n	s/n	s/n
Ver el recorrido de una ruta y las paradas por las cuales circula	s/n	s/n	s/n
Utilizar la funcionalidad ¿Cómo llego? Para detectar un punto en el mapa y mostrar sugerencias de buses para llegar hasta el mismo	s/n	s/n	s/n
Verificar la posición del usuario en el mapa digital.	s/n	s/n	s/n
Salir y liberar recursos del aplicativo	s/n	s/n	s/n

4.5 COMPARATIVA DE Q-BUS CON APLICACIONES SIMILARES

Para realizar la comparativa entre Q-Bus y las aplicaciones mencionadas en la sección “Estado del Arte” del presente documento, se tomaron en cuenta las funcionalidades comunes y las más sobresalientes de cada aplicación.

Las aplicaciones utilizadas para realizar la comparación fueron seleccionadas a través de una búsqueda en AppStore, siendo los parámetros de idoneidad las descargas realizadas, calificaciones obtenidas, costo y características de las apps. De las aplicaciones seleccionadas, ninguna de ellas muestran información del transporte público en una o varias ciudades de Ecuador.

Uno de los principales aspectos dentro de una aplicación móvil es la facilidad de navegación, de tal modo que el usuario no tenga que pensar mucho para poder utilizar todas las funcionalidades que provea dicha app. Para obtener sugerencias de buses para llegar desde un lugar a otro, tanto Q-Bus como Google Maps y Transit App utilizan el método de seleccionar puntos en el mapa, lo cual dependiendo de las circunstancias puede ser más sencillo para el usuario que recordar el nombre de una calle o intersecciones.

Q-Bus no presenta información en tiempo real de la ubicación de los distintos buses debido a que esta funcionalidad requeriría del uso de un GPS en cada bus, lo que supondría una inversión costosa. Sin embargo, podría ser también una tentativa para una próxima investigación ya que aportaría tanto a Q-Bus como al control de la ruta que sigue el bus y el cumplimiento de los horarios establecidos. Aplicaciones como Moovit y Transit App ofrecen esta funcionalidad, pero no hacen uso de dispositivos GPS, calculan la posición actual de un bus basado en el tiempo de recorrido, el horario de salida y la distancia que deben recorrer. El problema con este método es que no se toman en cuenta variables externas como la densidad del tráfico que puede existir, accidentes o cualquier otro factor que pueda afectar al recorrido del bus.

Una de las fortalezas de Q-Bus es que las rutas a presentarse no únicamente corresponden a buses, también pueden ser ingresadas al sistema rutas del Trolebús, Ecovía, Metrobus, Metro y sus alimentadores, lo cual permite a la aplicación brindar mayor información al usuario.

Cabe recalcar que Q-Bus, al ser una aplicación prototipo por el momento carece de algunas funcionalidades que podrían ayudarla a ser más competitiva en el mercado y atractiva para los usuarios. Tal es el caso del soporte para entregar sugerencias para llegar desde un lugar a otro teniendo en cuenta los trasbordos de los usuarios. Sin embargo, presenta también otras funcionalidades como el uso de BLE y códigos QR, para presentar

información completa de las rutas de buses que circulan por determinada parada, atractivo que ninguna otra app de las analizadas presenta.

La Tabla 15 muestra los resultados completos obtenidos durante el proceso de comparación.

Tabla 15. Comparativa entre Q-Bus y aplicaciones similares

Funcionalidad	Aplicaciones				
	Q-Bus	**Moovit**	**Google Maps**	**BA Cómo Llego**	**Transit App**
DISTRIBUCIÓN					
Distribución a través de tiendas virtuales	n/a	X	X	X	X
Aplicación gratuita	n/a	X	X	X	X
ÁMBITO					
Soporte para múltiples ciudades	-	X	X	-	X
Soporta una o más ciudades de Ecuador	X	-	-	-	-
Información proveniente de la comunidad	-	X	X	-	X
Permite visualizar información en tiempo real de la ubicación de distintos buses, metro, tranvía, etc.	-	X	-	-	X
Soporta múltiples medios de transporte público (bus, tranvía, metro, subterráneo)	X	-	X	X	X
SUGERENCIAS PARA LLEGAR DESDE UN LUGAR A OTRO					
Presenta sugerencias de rutas para llegar desde un punto hasta otro	X	X	X	X	X
Es posible seleccionar los puntos mediante la selección de puntos en el mapa	X	-	X	-	X
Permite la identificación de paradas cercanas	X	X	-	-	X
Presenta sugerencias de rutas tomando en cuenta trasbordos	-	-	X	X	X
Permite obtener información detallada de las rutas que circulan por las paradas	X	-	-	-	-
PARTICULARIDADES					
Presenta mapas temáticos e información turística	-	-	X	X	X
Implementa notificaciones	-	X	-	-	X
Funciona sin conexión a internet	X	X	-	X	X

Utiliza tecnología Bluetooth para la detección de paradas	X	-	-	-	-
Utiliza códigos QR para la detección de paradas	X	-	-	-	-

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

A continuación se presenta las conclusiones y recomendaciones que se obtuvieron en el desarrollo del proyecto

5.1 CONCLUSIONES

- El uso del formato JSON permite enviar estructuras de datos complejas mediante texto plano, lo cual debido a su ligereza facilita el intercambio de datos entre el servidor y la aplicación móvil. Además de optimizar el almacenamiento de información en la memoria de los dispositivos móviles.
- Apple a pesar de haber incorporado sus propios mapas en su SDK, mantiene el soporte para el uso del formato KML que permite a las aplicaciones móviles con mapas digitales dibujar rutas, líneas o figuras geométricas sobre los mismos.
- Para la detección de beacons se hizo uso de la librería provista por Estimote, la cual provee la funcionalidad para medir la distancia existente entre el beacon y el dispositivo móvil. Dicha funcionalidad no se encuentra disponible en la librería nativa provista por Apple.
- La adquisición de beacons supone una inversión considerable, es por ello que se presenta los códigos QR como alternativa que brinde la misma funcionalidad.
- El desarrollo e implementación del módulo de acceso a datos permite a la aplicación móvil evitar cualquier tipo de procesamiento que pueda afectar al rendimiento del dispositivo en el cual se ejecuta.
- El módulo de acceso a datos es el encargado de realizar la búsqueda de las rutas para llegar desde un lugar a otro. Al ser la presente investigación un prototipo y tener una cantidad de datos limitados, se utiliza un algoritmo de rápida implementación que permita poner a

disposición la información necesaria para cumplir con esta funcionalidad.

- Mediante las pruebas realizadas y el control de calidad se pudo asegurar el cumplimiento de los objetivos propuestos para el proyecto y de los requisitos presentados en cuanto a operatividad, interfaz gráfica y funcionamiento. Sin embargo, al ser un prototipo, no cumple con la suficiente funcionalidad como para ser lanzado al mercado.

5.2 RECOMENDACIONES

- Se recomienda configurar los beacons que se utilicen para identificar las distintas paradas de la ciudad de Quito con un mismo UUID, de tal manera que la aplicación monitoree una sola región.
- Para la entrega de sugerencias de las rutas de buses que un usuario puede tomar para dirigirse de un lugar a otro se recomienda tomar en cuenta la dirección que sigue la ruta.
- Si se desarrolla una aplicación que vaya a ser distribuida al público, se recomienda utilizar protocolos de seguridad en la transferencia de datos y encriptación de la información.
- Además de lo anterior, es importante tener en cuenta la cantidad de datos registrados en el sistema, ya que dependiendo de ello se podrá implementar un algoritmos que permita encontrar la ruta óptima de las sugerencias que actualmente se presentan.
- Para próximas investigaciones se recomienda tomar en cuenta los trasbordos que un usuario puede hacer para llegar desde un punto a otro.
- Para la puesta en producción de la aplicación móvil y su distribución a través de AppStore se recomienda seguir los lineamientos expuestos en el documento iOS…. Debido a que si alguno de los lineamientos no es cumplido, la aplicación será rechazada por el personal de Apple.

BIBLIOGRAFÍA

Apple Inc. (5 de Enero de 2016). *About the iOS Technologies*. Obtenido de iOS Developer Library: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

Bluetooth. (9 de Enero de 2016). *Bluetooth Technology Basics*. Obtenido de Bluetooth Web Site: https://www.bluetooth.com

Cacheiro González, M. L. (2014). *Educación y Tecnología: Estrategias didácticas para la integración de las TIC.* Universidad Nacional de Esucación a Distancia de Madrid.

Caporarello, L., Di Martino, B., & Martinez, M. (2014). *Smart Organizations and Smart Artifacts: Fostering Interaction Between People, Technologies and Processes.* Springer.

Cobo, A. (2005). *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web.* Ediciones Díaz de Santos.

Dávila Martínez, F. (2014). *Introducción a los Sistemas de Información Geográfica.* Servicio de Documentación Geográfica y Biblioteca - Ministerio de Fomento de España.

Estimote. (9 de Enero de 2016). *Developer Docs - Beacon Tech Overview*. Obtenido de Estimote: http://developer.estimote.com/

Estimote. (9 de Enero de 2016). *Developer Docs - iBeacon*. Obtenido de Estimote: http://developer.estimote.com/ibeacon/

Extreme Programming. (10 de Enero de 2016). *The Rules of Extreme Programming*. Obtenido de XP: http://www.extremeprogramming.org/rules.html

Ganzábal García, X. (2014). *Desarrollo y reutilización de componentes software y multimedia mediante lenguajes de guión.* Ediciones Paraninfo.

Google Inc. (6 de Enero de 2016). *What is KML?* Obtenido de Google Developers: https://developers.google.com/kml/

Guérin, B.-A. (2012). *Gestión de proyectos informáticos: Desarrollo, análisis y control.* Ediciones ENI.

Hernández, J. (2014). *Análisis y Desarrollo Web.*

IBM. (12 de Enero de 2016). Obtenido de UML, RUP, and the Zachman Framework: Better together: http://www.ibm.com/developerworks/rational/library/nov06/temnenco/

INTECO. (2009). *Ingeniería del Software: Metodologías y Ciclos de Vida.* Instituto Nacional de Tecnologías de la Comunicación de España.

iPhonedroid. (23 de Enero de 2016). *Home*. Obtenido de Kappta: https://kappta.com/es/

ISACA. (2011). *Geolocation: Risk, Issues and Strategies.*

JSON. (9 de Enero de 2016). *Introducción a JSON*. Obtenido de JSON: http://www.json.org/json-es.html

Kofler, M. (2008). *The Definitive Guide to MySQL.* Apress.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction.* Addison-Wesley Professional.

Microsoft. (9 de Enero de 2016). *Hardware - Bluetooth 4.0*. Obtenido de Microsoft: https://www.microsoft.com/hardware/es-es/support/bluetooth-4-compatibility

Microsoft Corporation. (5 de Enero de 2016). *¿Qué es el A-GPS?* Obtenido de Microsoft: https://www.microsoft.com/es-es/moviles/soporte/faq/?action=singleTopic&topic=FA114913

Moovit. (7 de Junio de 2015). Obtenido de Moovit: http://moovitapp.com/es/

Object Management Group. (21 de Mayo de 2015). *Unified Modeling Language Resource Page*. Obtenido de Unified Modeling Language: http://www.uml.org/

Oracle. (9 de Enero de 2016). *MySQL*. Obtenido de Oracle: http://www.oracle.com/es/products/mysql/overview/index.html

Parsons, J. J. (2015). *New Perspectives on Computer Concepts 2016, Comprehensive.* Cengage Learning.

Pedro, R. M. (2011). *Bluetooth 4.0 Low Energy: the future low-power consumption solution.* Universidad Politécnica de Catalunya.

PHP. (9 de Enero de 2015). Obtenido de PHP: https://secure.php.net/

Prieto Blázquez, J. (2013). *Introducción a los sistemas de comunicación inalámbricos.* Cataluña: Universitat Oberta de Catalunya.

QR Channel. (12 de Febrero de 2016). *¿Cómo funciona la tecnología QR-Code?* Obtenido de QR Channel: http://www.qrchannel.com/

Secretaría de Movilidad. (2 de Enero de 2014). *Transporte público en Quito moderniza su recaudación con la Caja Común.* Obtenido de Agencia Pública de Noticias de Quito: http://www.noticiasquito.gob.ec/Noticias/news_user_view/transporte_publico_en_quito_moderniza_su_recaudacion_con_la_caja_comun--10528

Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* San Francisco: Morgan Kaufmann.

Sommerville, I. (2005). *Ingeniería del Software.* Pearson.

The Ace Group. (2010). *QR Codes for Global Media.* The Ace Group marketing and printing solutions.

Universidad Estatal a Distancia de Costa Rica. (s.f.). *Manual de Transporte Público.* EUNED.

Warner, T. (24 de Julio de 2012). *Understanding iOS Location Services*. Obtenido de Pearson - QUE: http://www.quepublishing.com/articles/article.aspx?p=1927378

ANEXO 1

Versiones de iOS

Versión iOS	Lanzamiento	Dispositivos	Características
1.1	09/2007	iPhone 2G iPod Touch 1st Gen	• Interfaz gráfica • Gestos multitáctiles • Safari • Maps • iTunes Sync • Teclado por software • Compatibilidad con iPod touch
2.0	07/2008	iPhone 3G iPhone 2G iPod Touch 1st Gen	• App Store • Aplicaciones de terceros • Búsqueda de contactos • Soporte para cuentas Microsoft Exchange
3.0		iPhone 3GS iPhone 3G iPod Touch 2nd Gen	• Control por voz • Mensajes multimedia • Búsqueda por Spotlight • Notificaciones push • USB & Bluetooth tethering • Find my iPhone • Soporte para iPad • Soporte para teclados por Bluetooth • iBooks
4.0	06/2010	iPhone 4 iPhone 3GS iPhone 3G iPad iPod Touch 4th Gen iPod Touch 3rd Gen	• Multitarea • Carpetas en la pantalla de inicio • Facetime • Soporte para pantallas con retina • Soporte para iAd • Game Center • iTunes Ping • Fotografías HDR
5.0	10/2011	iPhone 4s iPhone 4	• Siri • Notification Center

		iPhone 3GS iPad iPad 2 iPod Touch 4th Gen, iPod Touch 3rd Gen	• PC-free • iTunes Wi-Fi Sync • iMessage • iCloud
6.0	09/2012	iPhone 5 iPhone 4s iPhone 4 iPhone 3GS iPad mini iPad 4th Gen iPad 3rd Gen iPad 2 iPod Touch 4th Gen iPod Touch 5th Gen	• Homegrown Maps and turn-by-turn navigation • Siri enhancements • Facebook integration • Passbook • iCloud Tabs • Mail enhancements • FaceTime over cellular
7.0	09/2013	iPhone 5s iPhone 5c iPhone 5 iPhone 4s iPhone 4 iPad mini iPad 4th Gen iPad 3rd Gen iPad 2 iPod Touch 5th Gen	• Visual overhaul • Control Center • AirDrop • Refreshed core apps • iTunes Radios • FaceTime Audio • BLE
8.0	10/2014	iPhone 5s iPhone 5c iPhone 5 iPhone 4S iPad mini iPad 4th Gen iPad 3rd Gen iPad 2 iPod Touch 5th Gen	• Continuity • Widgets • Extensibility • QuickType • iCloud Drive • HealthKit • HomeKit • Family Sharing

9.0	09/2015	iPhone 6 iPhone 6s iPhone 6+ iPhone 5s iPhone 5c iPhone 5 iPad mini iPad 4th Gen iPad 2 iPod Touch 5th Gen	• News, Switch to iOS, iCloud Drive • Mejoras en Siri • Soporte para iOS Public Beta program • 3D Touch

(Apple Inc., 2016)