FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

CREACIÓN DE UNA APLICACIÓN MÓVIL PARA LA REALIZACIÓN DE SOUNDWALKS CON INFORMACIÓN TURÍSTICA PARA EL CENTRO HISTÓRICO DE QUITO

AUTORES

MARCELO JOSÉ PUGA MENESES

MARCO IVÁN RIVERA GALARZA

AÑO

2018

FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

CREACIÓN DE UNA APLICACIÓN MÓVIL PARA LA REALIZACIÓN DE SOUNDWALKS CON INFORMACIÓN TURÍSTICA PARA EL CENTRO HISTÓRICO DE QUITO

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingenieros en Sonido y Acústica

Profesor Guía

MSc. Héctor Merino Navarro

Autores

Marcelo José Puga Meneses

Marco Iván Rivera Galarza

Año

2018

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, CREACIÓN DE UNA APLICACIÓN MÓVIL PARA LA REALIZACIÓN DE SOUNDWALKS CON INFORMACIÓN TURÍSTICA PARA EL CENTRO HISTÓRICO DE QUITO, a través de reuniones periódicas con los estudiante Marcelo José Puga Meneses y Marco Iván Rivera Galarza, en el semestre 2018-1, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Héctor Merino Navarro

Máster en Postproducción Digital

C.I. 1756785562

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, CREACIÓN DE UNA APLICACIÓN MÓVIL PARA LA REALIZACIÓN DE SOUNDWALKS CON INFORMACIÓN TURÍSTICA PARA EL CENTRO HISTÓRICO DE QUITO, de Marcelo José Puga Meneses y Marco Iván Rivera Galarza, en el semestre 2018-1, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Juan Francisco Jiménez Pacheco
Máster en Postproducción Digital
C.I. 1717340192

DECLARACIÓN DE AUTORÍA DE LOS ESTUDIANTES

“Declaramos que este trabajo es original, de nuestra autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

_____	_____
Marcelo José Puga Meneses	Marco Iván Rivera Galarza
C.I. 1715599716	C.I. 1722442769

AGRADECIMIENTO

Agradezco primeramente a Dios y a mi familia por estar siempre a mi lado, en los buenos y malos momentos, también quiero agradecer a todos mis amigos y compañeros con los cuales compartí durante todos estos años, he aprendido mucho de cada uno de ellos, finalmente quiero agradecer a todos los docentes, en especial a Héctor Merino por su apoyo y consejos en este último trayecto de aprendizaje.

Marco Iván Rivera Galarza

AGRADECIMIENTO

Gracias a mis padres y hermanos por su apoyo y amor incondicional. Gracias al resto de mis familiares que se han preocupado por mi futuro y me han dado consejos. Gracias a mis amigos de universidad y estudios por ser mis acompañantes durante este viaje de más de 5 años. Gracias a mis amigos eternos de colegio por ser la familia que elegí. Gracias a mis profesores que, con sus enojos y felicitaciones, me han formado como estudiante. Gracias a mi tutor Héctor Merino por ser siempre frontal con sus comentarios y desiciones. Por último, gracias a la vida por dejarme llegar hasta este punto.

Marcelo José Puga Meneses

DEDICATORIA

Dedico este trabajo a mis padres Marco Rivera y Rina Galarza, a mi abuelita Blanca Alvear y a mis hermanas Melina y Tatiana Rivera.

Marco Iván Rivera Galarza

DEDICATORIA

Para mi padre Marcelo y mi madre Catalina. Jamás podré agradecerles ni recompensar todo lo que han hecho por mi.

Marcelo José Puga Meneses

RESUMEN

El Centro Histórico es uno de los lugares de Quito más concurridos por turistas nacionales y extranjeros, no obstante, la información turística de sus atractivos no está siempre al alcance de las personas al momento de su visita. Este proyecto busca desarrollar una aplicación móvil para Android en lenguaje Java, que permita escanear códigos QR ubicados en diferentes atractivos turísticos. Cada código escaneado dará al usuario acceso inmediato a información audiovisual (fotografías, audios y texto) del lugar. La aplicación busca cambiar la forma tradicional de hacer turismo, brindando una herramienta que puede ser utilizada en cualquier lugar y el cualquier momento. Adicionalmente, se incorporaron audios informativos que permiten realizar paseos sonoros con contenido turístico. Por medio de reseñas, leyendas y personajes característicos, los audios harán que el usuario se sienta parte de la historia.

ABSTRACT

Quito’s Downtown is one of the most visited places by locals and foreigners, but the touristic information from its attractions is not always available at the moment of the visit. This project tries to develop a mobile application, for Android devices using Java language, that let users scan QR codes located in different touristic attractions. Each scanned code would give immediate access to audiovisual information (photos, audios and text) from the place. The app is looking forward to change the traditional touristic methods, providing a tool that could be used at any place and time. Additionally, informative audios with touristic content were included to help the user do soundwalks through the city. With the help of historical facts, legends and famous characters, the audio will make the users feel like they are being part of history.

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Antecedentes

El turismo es considerado una actividad fundamental para el desarrollo económico y social de un pueblo, formando parte de los quehaceres comunes en el tiempo libre de un individuo. Las personas tienden a interesarse por descubrir nuevos lugares para enriquecer su conocimiento y excitar sus sentidos. Sin importar el medio de transporte utilizado para llegar a los diferentes destinos, el caminar ha sido siempre la actividad preferente para recorrerlos y observarlos a detalle.

Las caminatas, según Butler (2006), son consideradas núcleos investigativos ya que pueden tener enfoque geográfico, filosófico, arquitectónico, ecológico-acústico e incluso artístico. En los siglos XIX y XX, gracias a poetas, escritores y filósofos, el caminar se volvió rápidamente un acto creativo, reflexivo e incluso subversivo. Es común que las personas busquen una satisfacción visual en sus caminatas, pero, en un mundo donde el urbanismo crece desmesuradamente, es cada vez más importante encontrar una sensación sonora satisfactoria.

Un *Soundwalk* (SW) es un paseo sonoro que busca mejorar la experiencia sonora de cualquier individuo. De acuerdo a Paquette y McCartney (2012), un SW es cualquier caminata que busca encontrar relaciones entre las personas y el paisaje sonoro que los rodea, por lo que requiere de una escucha consciente. Los SW buscan también mejorar la percepción de los paisajes sonoros y pretenden formar un criterio de la calidad sónica del entorno, especialmente para personas que residen en lugares urbanos (Nilsson *et al*, 2012). En ocasiones, los SW involucran también la grabación de los sonidos.

La conexión entre los paisajes visuales y sonoros es muy importante. En ciudades grandes en donde se combinan las actividades urbanas con las de recreación, existen paisajes sonoros muy ruidosos que no encajan con los aspectos visuales. Por ejemplo, en la ciudad de Xiamen, China, se realizó un estudio a manera de

SW en cinco parques diferentes. Se determinó que la mayoría de personas se sienten más influenciadas por los sonidos molestos de la ciudad, que por el ambiente pacífico y natural propio de los parques (Liu *et al*, 2014).

Los SW pueden ser utilizados también como recursos para la comunidad turística, incorporando pastillas auditivas de carácter informativo que incluyan reseñas, historias, personajes y leyendas de los distintos lugares que recorren las caminatas. En ciudades como Roma o París se han implementado dispositivos a manera de guías turísticas para museos, calles y lugares emblemáticos. Esta experiencia cuenta con elementos importantes como paisajes sonoros, efectos, diálogos y música, que involucran al oyente de una manera diferente y le permiten apreciar los elementos de mejor manera. Esta tecnología crea mayor conexión entre el usuario y las atracciones, e implica un mayor aprendizaje de lo expuesto (Elliston y FitzGerald, 2012).

Existe otro sistema de diseño propuesto por Mohseninia (2016), en el que se detalla el proceso para realizar una aplicación basada en SW con la ayuda de la tecnología digital. Mohseninia creó un dispositivo con programación Android, que interactúa con el usuario para proveer información de cualquier tipo basándose en su ubicación geográfica. Android es considerado el lenguaje más sencillo para desarrollo de aplicaciones móviles (Alshattnawi, 2014).

Otros sistemas más simples, que cuentan con emisores y receptores de señal, son también utilizados con fines turísticos e informativos. Los emisores mandan señales que cubren un área con un radio determinado y los receptores, que poseen los usuarios, reproducen los distintos audios dependiendo de la señal recibida. El principal problema de estos sistemas es la interferencia producida cuando dos áreas de cobertura se cruzan, creando conflicto en el dispositivo receptor; aún así, con un buen diseño de áreas y una óptima ubicación de dispositivos emisores, se puede resolver este inconveniente (Lyons, Gandy y Starner, 2013).

Otra ventaja de los SW es que pueden ser utilizados por personas con discapacidad visual. Éstas aprovechan la tecnología y utilizan aplicaciones basadas en su ubicación para reproducir indicaciones de futuros movimientos. Esta ayuda tecnológica genera una sensación sonora nueva para estas personas, ayudándoles a crear un entorno 'visual' basado en los elementos del paisaje sonoro reproducido (Rizopoulos, Lambrinos y Gazi, 2014). Estas aplicaciones buscan trabajar en tiempo real para ubicar a los usuarios en lugares específicos y hacer que se sientan parte del entorno, a pesar de no visualizarlo.

Hoy en día, la mayor cantidad de herramientas turísticas vienen en forma de aplicación móvil gracias al crecimiento tecnológico de los *smartphones*. Un claro ejemplo de aplicaciones para realizar SW es Soundwalkrs. Esta aplicación cuenta con SW en 8 ciudades de Latinoamérica, Brasil y España: Bogotá, Cartagena, Santa Marta, Rivera Maya, Buenos Aires, Galicia, Ciudad de México y Río de Janeiro. Cada destino cuenta con diferentes rutas que guían al usuario a recorrer por calles de diferentes partes de la ciudad, acompañándolo de historias y relatos de cada sitio. La aplicación, que se encuentra disponible para iOS y Android, ya cuenta con miles de descargas en las diferentes plataformas, recibiendo buenos comentarios.

El Centro Histórico (CH) es una de las principales atracciones de Quito y emblema del Ecuador, siendo Patrimonio Cultural de la Humanidad. Por esta razón recibe más de un millón y medio de visitantes anualmente por parte de turistas nacionales y extranjeros. El CH cuenta con más de cien atracciones turísticas entre iglesias, museos, plazas y otros lugares emblemáticos, cada uno con antecedentes históricos que remarcan a siglos pasados y que llaman la atención de los visitantes por su gran importancia cultural e histórica. Cada atracción cuenta normalmente con un servicio independiente de guía turística, pero en algunos casos, representantes de la entidad encargada del turismo de la ciudad, llamada Quito Turismo, se encargan de este servicio.

Gracias a la época en la que fue construido, el CH posee un estilo republicano europeo, formado de plazas y calles angostas que crean manzanas rectangulares y conglomeradas. El comercio, el tránsito vehicular y la cantidad de atracciones en un área relativamente pequeña, han hecho que la mayoría de visitantes recorran este espacio de la ciudad a pie. Estos factores permitirían que se realicen SW con contenido turístico en la ciudad.

1.2. Justificación

La ciudad de Quito recibe 1 560 000 visitantes anualmente, de los cuales 560 000 son personas que residen en el extranjero. El lugar más visitado es el Centro Histórico, con un 35.1% del total de visitantes (Quito Turismo, 2013). Esto abre paso a una necesidad de brindar información inmediata a turistas nacionales y extranjeros acerca de nuestra ciudad.

El principal problema recae en que esta información no está siempre, por diferentes motivos, al alcance de los visitantes, quienes muchas veces se conforman sólo con ver los lugares sin llegar a conocer la gran cantidad de riqueza histórica detrás de los atractivos.

Con la creación de una aplicación móvil, se busca brindar este servicio de manera novedosa e interactiva. La aplicación permite realizar paseos sonoros a través del CH de la ciudad, utilizando la lectura de códigos QR como recurso. Una vez leído el código el usuario puede acceder a información audiovisual del lugar. Los audios a reproducir contienen información turística, música y efectos sonoros, permitiendo al usuario escuchar mientras recorre a pie las rutas previamente establecidas por la aplicación.

Un sistema de guía turística novedoso pretende atraer tanto a viajeros que todavía no conocen Quito, como a antiguos visitantes que deseen ampliar su conocimiento y tener una nueva experiencia turística.

1.3. Alcance

Se pretende que la aplicación desarrollada llegue a manos de toda persona que requiera información turística del CH de Quito y que busque una nueva experiencia turística y auditiva. Al ser una aplicación móvil, estará al alcance de todo individuo que cuente con un smartphone Android y un par de audífonos para la información auditiva. La implementación de un nuevo sistema de guía turística atraería a cientos de visitantes nacionales y extranjeros que quieran aprender más sobre el CH de la ciudad y sus atractivos.

La variedad de idiomas son un limitante para los guías turísticos tradicionales. Por lo general hay tours en español para visitantes nacionales y en inglés para extranjeros, excluyendo al resto de idiomas. La aplicación permitiría incluir en su librería pastillas auditivas de múltiples idiomas, para incrementar la cantidad de usuarios. Otro inconveniente de los guías tradicionales es su disponibilidad de tiempo. La aplicación se puede utilizar a cualquier hora del día y los códigos QR pueden ser implementados en cualquier lugar deseado para ser leídos por los usuarios, dándoles completa independencia al momento de hacer su visita.

Otro aspecto que se podría extender en un futuro es la reproducción de audios en base a la geolocalización. A pesar de que los códigos QR son la base de la aplicación, se podría implementar un mapa con la ubicación geográfica de los destinos y la del usuario, para que sea más sencillo seguir la ruta del paseo sonoro.

De ser exitoso el sistema, se lo podría implementar no solo en el CH, sino en cualquier destino turístico del planeta. Se podría empezar por otras ciudades del Ecuador, para después expandirse por Latinoamérica y el mundo. La ventaja de la aplicación es que el sistema es fácil de duplicar y es aplicable a cualquier lugar; lo único que se tendría que elaborar para cada sitio sería el contenido (texto, imágenes y audio).

2. OBJETIVOS

2.1. Objetivo general

Desarrollar una aplicación móvil para Android que permita la lectura de códigos QR, con el fin de realizar paseos sonoros con información turística de lugares del Centro Histórico de Quito.

2.2. Objetivos específicos

- Utilizar la herramienta Android Studio y el lenguaje de programación Java para el desarrollo de la aplicación.
- Aplicar herramientas aprendidas en los diferentes cursos universitarios y complementar con fuentes externas, para abarcar la información necesaria en recursos de programación para el efectivo desarrollo de la aplicación.
- Implementar la lectura de códigos QR como fuente de información y punto de contacto entre el usuario y el contenido informativo de la aplicación.
- Crear un paseo sonoro piloto con tres atracciones turísticas del CH de Quito que compruebe el correcto funcionamiento y alcance de la aplicación.
- Recolectar información sobre hechos históricos de los sitios elegidos para crear una librería de recursos con contenido auditivo y visual.
- Producir material auditivo de calidad con el conocimiento y los recursos brindados por la universidad a lo largo de la carrera, aplicando correctamente cada una de las etapas de producción de audio.
- Evaluar la aplicación realizada por medio de una encuesta que permita analizar los resultados de diferentes aspectos de la misma.

3. MARCO TEÓRICO

3.1. *Soundwalk*

Un SW se define como un paseo que involucra una caminata en un determinado recorrido o lugar (previamente escogido), con un enfoque primordialmente auditivo brindando una experiencia sonora de calidad. La caminata pretende encontrar, de igual manera, una conexión sónica entre el individuo y el paisaje que lo rodea (Paquette y McCartney, 2012).

El término SW como tal aparece en la década de los 70 gracias a Raymond Murray Schafer, compositor, músico, escritor y ambientalista canadiense que dedicó gran parte de su trabajo a la ecología y conciencia acústica del mundo moderno. A pesar de ser un pionero, se dice que hubo ciertas actividades realizadas por grupos de personas antes que él; algunos trabajos previos que consistieron en la grabación de sonidos para analizar paisajes sonoros que incluían una caminata, ya podrían ser considerados SW (McCartney, 2014).

Actualmente los SW son utilizados para analizar paisajes sonoros de zonas urbanas donde se requiere una evaluación de dichos paisajes, pero son también considerados expresiones artísticas. Existen páginas web como soundwalk.com y aplicaciones móviles como Sounwalkrs, que recopilan y crean SW de diferentes partes del mundo elaborados por varios artistas. Éstos contienen audios de alta calidad sonora para ser escuchados en dichos lugares. En el caso de la aplicación, aparte de ser artísticos, los audios poseen información turística para que los usuarios “escuchen la historia y viajen en el tiempo” (Soundwalkrs, 2015).

Es importante evitar confusiones entre los términos *soundwalk* y *soundscape*. La palabra s*oundscape* (SC) se compone por *sound* que significa sonido y *landscape* que significa paisaje, por lo cual en español se la conoce como paisaje sonoro. Un SC incluye todos los sonidos que forman parte del ambiente de un lugar específico. Éstos pueden ser utilizados en múltiples áreas de la acústica

ambiental, desde la ecología acústica hasta el diseño urbanístico. La principal diferencia con los SW es que, a pesar de que también pueden ser reproducidos o grabados, los SC no involucran necesariamente una caminata. Para entender mejor, se podría decir que un SW puede ser utilizado para identificar el SC de un lugar, mas no a la inversa (Jean *et al*, 2013).

3.2. Códigos QR

Los códigos QR (cQR), por sus siglas '*Quick Response*', son códigos matriciales o de dos dimensiones, diferentes de los códigos de barras comunes que son lineales y unidimensionales. Fueron creados por la industria automotriz japonesa en 1994 para controlar los procesos de manufactura de vehículos. Al igual que cualquier código de barras, un cQR debe ser detectado por un lector especializado para que, una vez decodificado, se pueda obtener la información del objeto al que está vinculado. A pesar de no ser los únicos códigos con esta forma, tres cuadrados de menor tamaño en sus esquinas los diferencian del resto.

3.2.1. Diseño y funcionamiento

Los cQR fueron diseñados para ser detectados por un sensor de imagen. Una vez detectado, el código es analizado por un procesador digital previamente programado para realizar dicha tarea. El procesador ubica los cuadrados ubicados en las tres esquinas para posicionar el código. Los puntos negros de la parte interna son después convertidos a números binarios y corregidos por un algoritmo corrector propio del procesador. Una vez decodificado, el lector muestra la información vinculada al los caracteres asociados al cQR.

En la *figura 1* se puede observar la estructura de un cQR. Hay que aclarar que la densidad o cantidad de puntos de la parte interna del código, depende directamente de la cantidad de caracteres que se quiera codificar.

Para poder leer este tipo de codificación es necesario un sensor de imagen. Por lo general, se destinan los cQR a dispositivos móviles, por lo que la cámara de un

smartphone puede funcionar como lector. Lo que se necesitaría aparte es un procesador de dicha codificación, el cual es incluido en aplicaciones gratuitas que permiten la lectura de éstos. Para generarlos, en cambio, existen páginas web que cuentan con este servicio de manera gratuita e inmediata. La página requiere que se ingresen los caracteres que se quieran codificar (letras, números, o un URL) y la herramienta proporcionará un cQR con dicha información.

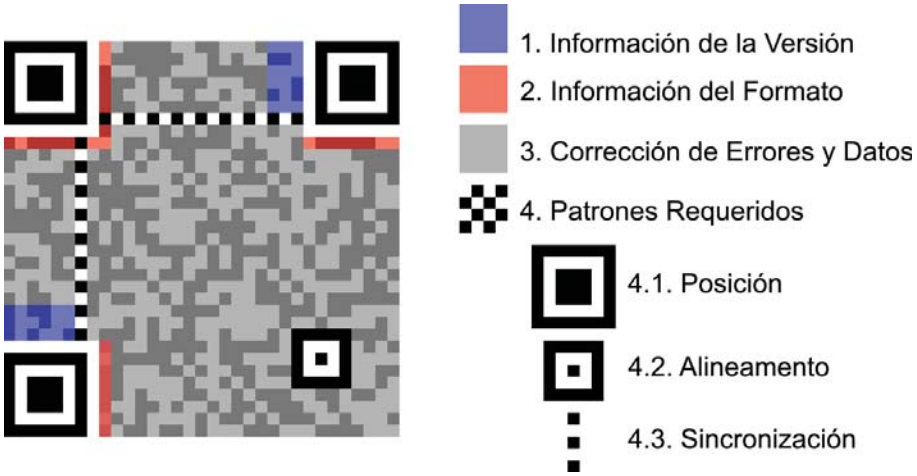

Figura 1. Estructura del código QR.
Tomado de (Atala, 2008).

3.2.2. Aplicaciones y usos de los códigos QR

A pesar de que fueron creados con el objetivo de rastrear vehículos japoneses de la marca Denso Wave durante su manufactura, con el pasar de los años la popularidad de estos códigos ha crecido y actualmente se pueden relacionar con el marketing y promoción de productos o servicios. Es común encontrarlos en pancartas o revistas con el fin de ser escaneados y enviar a los usuarios a páginas web con la información que se pretende compartir.

En el año 2010 se implementaron en Nueva York estos códigos en diferentes sitios del Central Park para que los visitantes los escaneen y encuentren información acerca de dicho lugar. Los transeúntes podían acceder a escenas de películas,

partes de canciones o historias características que habían ocurrido en los diferentes puntos del parque (IDEA, 2011).

Pero los cQR tienen, de hecho, aplicaciones ilimitadas. De acuerdo a Andre Walsh (2010), los dispositivos móviles cuentan en su mayoría con GPS integrado y esto ha facilitado a que la mayoría de aplicaciones sean basadas en la ubicación geográfica del usuario. Los cQR, sin embargo, tienen la posibilidad de brindar al usuario información de cualquier punto geográfico de manera más rápida y sencilla. Este sistema permite al usuario el acceso desde un espacio físico hacia una librería digital con recursos informativos que representa a dicho espacio.

3.3. Lenguaje Java

Java es un lenguaje de programación de propósito general utilizado en comunicación entre dispositivos, creación de aplicaciones móviles, diseño de *software*, páginas web y acceso a base de datos. Fue creado originalmente por James Gosling y comercializado en 1995 por la empresa Sun Microsystems, que fue adquirida posteriormente por Oracle. Después de ganar popularidad se creó una plataforma con el mismo nombre, la cual facilita el desarrollo de este lenguaje. Gran parte de la sintaxis de Java se deriva de los lenguajes C y C++.

El lenguaje Java fue diseñado para tener pocas dependencias de implementación. Esto quiere decir que los objetos son lo más independientes y autosuficientes posible (sebastianspeed, 2015). Adicionalmente, tiene la característica de ser WORA (*write once, run anywhere*), lo que significa que un código Java compilado puede ser ejecutado en cualquier plataforma que soporte Java, sin tener que ser re-compilado.

3.3.1. Programación orientada a objetos

La programación orientada a objetos (POO) es la que se forma de ‘objetos’ en su estructura. Un ‘objeto’ contiene datos en forma de campos o **‘**atributos’ y código en forma de procedimientos o ‘métodos’. La idea es que los métodos de los objetos

logren acceder o modificar a sus atributos. La mayoría de lenguajes de POO son basados en ‘clases’, lo que significa que los objetos son instancias o partes de una clase.

Figura 2. Diagrama que representa el funcionamiento de la POO.
Tomado de (Tah, 2007).

- **Campos o atributos**

El campo es un espacio de almacenamiento para un dato. En la POO, dentro de los campos se almacenan los datos de un objeto. En otros tipos de programación, los campos también pueden ser llamados variables.

- **Métodos**

Un método es un procedimiento o subrutina asociado a un campo o atributo. Por ejemplo, a un atributo llamado `window` se le pueden asignar métodos como `open` o `close`.

- **Mensajes**

Al ejecutar un POO, los objetos reciben, interpretan y responden a mensajes de otros objetos. Cuando un objeto recibe un mensaje, este responde ejecutando el procedimiento asociado a dicho mensaje.

- **Clases**

En POO, una clase es un *template* o plantilla predeterminada que sirve para la creación de objetos. En el caso de Java, las clases van a depender de las API (*Application Programing Interface*) que se utilicen. Cada clase está compuesta de métodos, que a su vez se componen de atributos.

3.3.2. *Application Programming Interface*

Más conocido por sus siglas API, una interfaz de programación de aplicaciones es un conjunto de subrutinas, protocolos y métodos que facilitan el desarrollo de aplicaciones. En otras palabras, es una serie de métodos comunicacionales bien definidos y estandarizados. Comúnmente, cada lenguaje de programación posee una API, por lo que Java también.

Usualmente, las API son relacionadas con librerías de *software* o sistemas operativos que incluyen especificaciones para rutinas, estructura de datos, clases de objetos, variables y llamada a procedimientos remotos. En un *software* de programación, como Android Studio por ejemplo, se pueden implementar múltiples API dependiendo de los requerimientos de la aplicación a desarrollar.

3.3.3. Usos y aplicaciones

Java es en la actualidad uno de los lenguajes más populares. La mayoría de las personas, a pesar de no saber programar, se han encontrado alguna vez instalando un componente Java en su computador. Esto se debe a que la mayoría de páginas webs o blogs, así como videojuegos y sistemas operativos, tienen como requisito tener instalado Java (Rodríguez, 2015).

Se dice que en el mundo existen más de 9 millones de desarrolladores utilizando Java. El 100% de dispositivos Blue-Ray, 125 millones de dispositivos de TV y 3000 millones de *smartphones* ejecutan Java (Oracle, 2017). Esas, entre otras cifras,

confirman que este lenguaje y plataforma es uno de los más utilizados a nivel mundial.

3.4. Android

Es un sistema operativo para dispositivos móviles, principalmente *touchscreen*, desarrollado por Google y Open Handset Alliance y con núcleo Linux. Su modelo de desarrollo es de código abierto por lo que se mantiene en constante actualización y el desarrollo de sus programas no requiere ningún tipo de licencia.

Actualmente es el sistema operativo más utilizado a nivel mundial en celulares y *tablets*, con más de 1 500 000 dispositivos activados diariamente, superando a sus competidores iOS y Windows juntos. Ciertas variantes de Android también pueden ser encontradas en cámaras fotográficas, consolas de video, computadores, televisores, entre otros dispositivos electrónicos. En la tienda Google Play existen más de 3.5 millones de aplicaciones con este sistema operativo.

3.4.1. Fundamentos de una aplicación Android

Las aplicaciones Android pueden ser escritas en lenguaje Kotlin, Java, o C++. Cualquier IDE (Entorno de Desarrollo Integrado) para Android compila el código con los datos y los recursos en un archivo *Android Package* (APK). Este archivo con sufijo `.apk` contiene todo el contenido de la aplicación y es utilizado por el dispositivo para instalar el programa.

En el sistema operativo de Android, cada aplicación funciona como un usuario diferente, lo que permite que cada aplicación tenga su propio *virtual machine* y pueda correr el código independientemente. Una ventaja de estas aplicaciones es que siempre requieren un permiso para acceder a los datos del usuario, como contactos, mensajes, cámara, *bluetooth*, o acceso a internet y solo el usuario puede garantizar esos permisos, brindándole seguridad y privacidad (Android Developers, 2014).

- **App Components**

Estos componentes son esenciales para la construcción de bloques de una aplicación Android. Cada componente es un punto de acceso por el que el sistema o el usuario pueden entrar a la aplicación. Algunos dependen de otros para su correcto funcionamiento y cada uno tiene un propósito diferente. Éstos componentes son:

- ***Activities***

Un `activity`, o actividad en español, es un componente que actúa como punto de acceso de la aplicación para interactuar con el usuario. Representa una sola pantalla que forma parte de la interfaz del usuario. Por lo general, los IDE de Android, como Android Studio, cuentan con diferentes plantillas de `activity`.

Una aplicación móvil de correo electrónico, por ejemplo, tendría un `activity` que muestra la lista de correos, otro para redactar un nuevo correo y otro en las que se abren los correos. La *figura 3* muestra la ventana de creación de un `activity`.

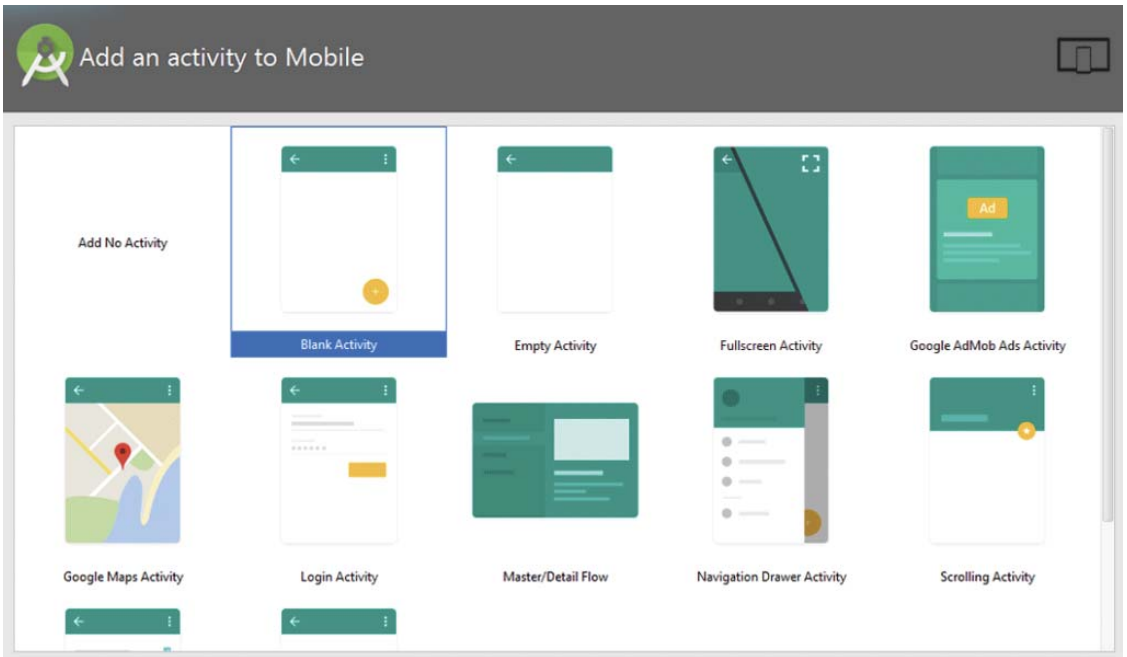

Figura 3. Ventana para agregar un nuevo `activity` en Android Studio.

- ***Services***

Un `service`, o servicio, es un punto de acceso para que la aplicación siga funcionando en segundo plano. Esto quiere decir que un `service` no provee una interfaz de usuario, a diferencia del `activity`. Por ejemplo, se puede seguir escuchando el audio de una canción mientras se está en la interfaz de la aplicación de Twitter o Facebook, todo gracias a un `service` activado en el reproductor de audio.

El `service` más común es justamente el de reproducción de música. Otro muy usado es el de la conexión a la red, el cual por lo general el usuario ni se entera que está activado.

- ***Broadcast Receivers***

El `BroadcastReceiver` es un componente que permite a la aplicación recibir eventos por parte del sistema del dispositivo móvil. Por ejemplo, una aplicación puede guardar un recordatorio para una fecha importante, pero tiene que recibir el evento proporcionado por la alarma del dispositivo para que suene el teléfono en dicha fecha.

- ***Content Providers***

Un `ContentProvider` es un componente que administra una serie de datos compartidos de la aplicación, los cuales pueden ser almacenados en los archivos del sistema, en una base de datos, o en un servidor web. Al ser un punto de acceso, un `ContentProvider` permite que otras aplicaciones modifiquen los datos, como por ejemplo la información de contacto del usuario.

- **Archivo *manifest***

Antes de que el sistema Android pueda empezar una app component, el sistema debe saber que ese componente existe leyendo el archivo `manifest`, llamado

`AndroidManifest.xml`. Si una componente está nombrada en el código principal, pero no declarada en el `manifest`, nunca se va a ejecutar.

Aparte de las componentes, en este archivo se encuentran todos los permisos que requiere la aplicación, como el uso de Internet o acceso a contactos. También se declaran las características de la aplicación, como cámara, *bluetooth*, o *touchscreen*. Por último se declaran las librerías y el nivel API del lenguaje de programación que la aplicación requiere para funcionar.

- **Recursos**

Los recursos son las imágenes, audios, videos y cualquier otro elemento relacionado con la interfaz visual del usuario. Estos recursos son indispensables ya que la aplicación no solo está hecha de código. El uso de recursos hace fácil la modificación de características de la aplicación, sin tener que alterar código de programación.

Cada recurso es asignado a un `integer` ID, el cual es tipo de dato. Estos van a ayudar a la aplicación a llamar a recursos alternativos sin tener que cambiar de `activity`. Por ejemplo, en una aplicación con dos idiomas, crear diferentes recursos de textos para cada uno permite cambiar el contenido sin necesidad de cambiar de `activity`.

3.4.2. Android Studio

Android Studio es el IDE estándar para el sistema operativo Android de Google y para el desarrollo de aplicaciones Android. Conocido en español como Entorno de Desarrollo Integrado, un IDE es un programa que facilita a los programadores el desarrollo de software. Las tres partes fundamentales de un IDE son el editor de código, las herramientas de construcción de *software* (encargadas de compilar o transformar el código a binario, empaquetar la información binaria y de correr pruebas automatizadas) y un *debugger* que encarga de ejecutar y probar el programa principal en busca de *bugs* o errores.

El lanzamiento del programa, diseñado exclusivamente para equipos Android, fue en diciembre de 2014. El *software* es gratuito y se encuentra disponible para Windows, MacOs y Linux. Su desarrollador es Google, su autor es IntelliJ y permite Java como lenguaje de programación.

Una de sus ventajas es que cuenta con un editor que permite arrastrar e insertar componentes para la interfaz gráfica del usuario y se pueden implementar *templates* de aplicaciones comunes de Android, facilitando el proceso de diseño. Adicionalmente cuenta con un dispositivo virtual propio de Android que tiene la función de probar y ejecutar aplicaciones. Esta IDE trabaja con los componentes mencionados anteriormente (TechMagazine, 2016).

Figura 4. Logo de Android Studio.
Tomado de (Olupot, 2016).

4. DESARROLLO DE LA APLICACIÓN

Durante este capítulo, se analizarán todos los procesos implicados en el correcto funcionamiento de la aplicación, así como ciertos cambios realizados en el procedimiento. Se pretende presentar el proceso de forma ordenada y clara para el buen entendimiento. Cabe recalcar que el código de programación es insertado en el informe en formato de texto, por lo tanto no son capturas de pantalla.

4.1. Detalles generales de la aplicación

La aplicación fue desarrollada en lenguaje Java, exclusivamente para dispositivos Android. Durante todo el proceso se utilizó como IDE al programa Android Studio, el cual permite incorporar fácilmente todas las partes de la aplicación gracias a sus intuitivos procesos y sus plantillas pre-programadas. También permite ejecutar el programa en cualquier momento en un *virtual machine*.

El objetivo principal de la aplicación es que el usuario pueda escanear un código QR, que idealmente estaría ubicado en distintos puntos turísticos del Centro Histórico de Quito, utilizando la cámara de su dispositivo móvil y, una vez escaneado, obtenga información turística inmediata acerca del sitio. Cada lugar cuenta con fotografías, texto y audio como fuentes de información para el usuario. Posteriormente, el individuo podrá leer otro código para obtener nueva información y así sucesivamente.

Figura 5. Flujo de información de la aplicación.

El flujo de información se puede observar en la *figura 5*. La aplicación escanea el código, que es decodificado para proporcionar la información en la misma aplicación, para que finalmente pueda ser recibida por el usuario. Por ende, la aplicación funciona como núcleo del sistema.

4.2. Carpetas y nombramiento de recursos

Antes de entrar en detalle con el código de programación hay que entender como funcionan los recursos de la aplicación.

Como se mencionó en el marco teórico, los recursos son todos los elementos que van a constituir la aplicación, aparte del código. En la aplicación desarrollada se van a implementar imágenes, texto y audios. Es importante agregar cada recurso en la carpeta `res`, mostrada en la *figura 6*. Éstos deben ser nombrados correctamente, puesto que posteriormente esos nombres serán utilizados dentro del código principal para convocarlos.

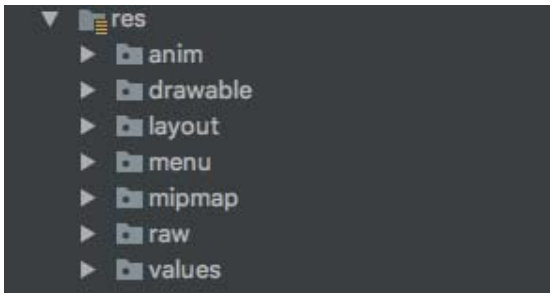

Figura 6. Ubicación de la capeta `res`.

4.2.1. Imágenes

Las imágenes son recursos fundamentales de la aplicación ya que forman parte de su interfaz gráfica. Éstas incluyen a las fotografías de los diferentes destinos turísticos y al ícono de la aplicación, las cuales deben ser insertadas en la carpeta `drawable`, indicada en la *figura 7*. Cada archivo está etiquetado de forma ordenada para facilitar su nombramiento dentro del código principal.

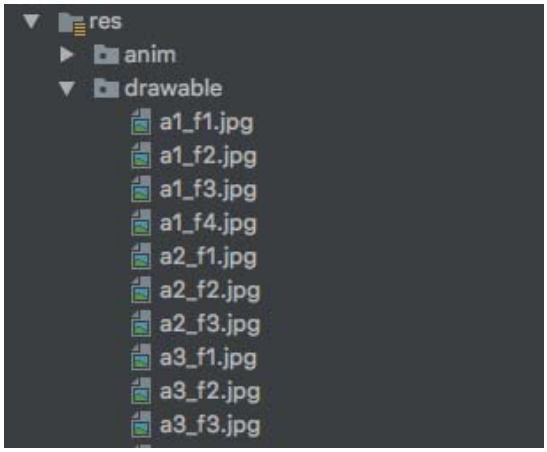

Figura 7. Ubicación de la capeta `drawable`.

4.2.2. Texto

En Java, la palabra `string` es una variable de caracteres (no numérica). En el archivo `strings.xml` se declaran los nombres a las variables para los textos respectivos, que luego se van a insertar en el código principal. Éstos se encuentran en la carpeta `strings.xml` como indica la *figura* 8.

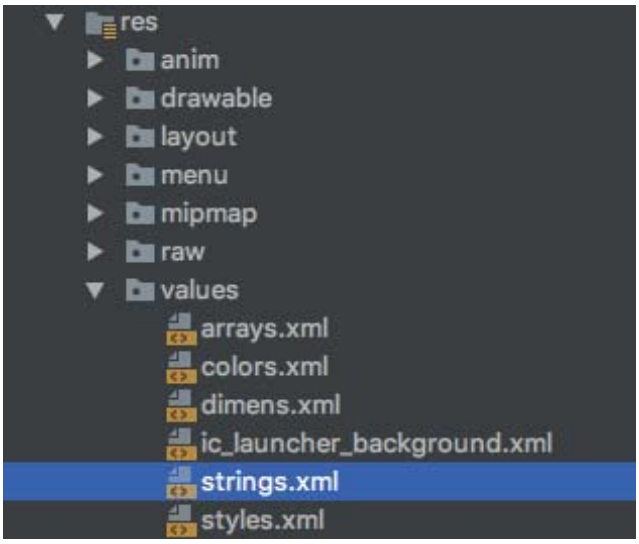

Figura 8. Ubicación del archivo `strings.xml` dentro de la capeta `res/values`.

Una variable de texto, por ejemplo, es el nombre de la aplicación. Como se puede observar, en la siguiente línea de código, se asignó al `string app_name` el texto *Soundwalk UIO*. Se escogió este nombre piloto para la aplicación en honor a los recorridos auditivos y a su lugar objetivo.

```
...
<string name="app_name">Soundwalk UIO</string>
...
```

Los textos de los títulos y descripciones de cada lugar turístico son insertados en otro archivo que es explicado más adelante, ya que éstos cambian dependiendo del cQR escaneado.

4.2.3. Audios

Por último, los audios son incluidos en la carpeta `raw`, dentro de la carpeta `res`, como indica la *figura 9*.

De igual manera, cada audio debe ser nombrado correctamente ya que esos nombres van a ser utilizados posteriormente.

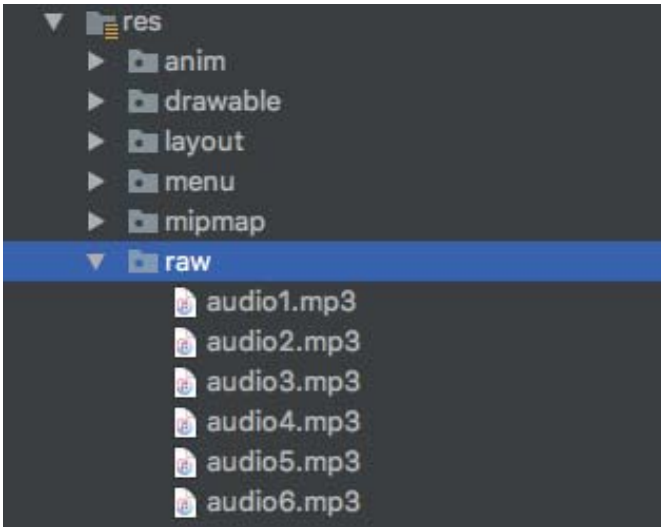

Figura 9. Ubicación de la carpeta `raw` dentro de la capeta `res`.

4.3. Archivo *manifest*

Como se describió en el marco teórico, en este archivo se encuentran los API, las componentes, las características y los permisos de acceso de toda la aplicación. Es también el primer archivo en el que se escribe código.

El archivo `AndroidManifest.xml` se encuentra en la parte izquierda en la carpeta `manifests`, como indica la *figura 10*.

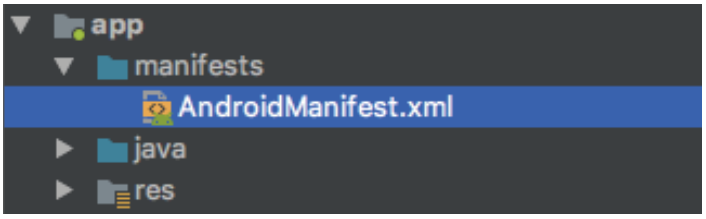

Figura 10. Ubicación del archivo `AndroidManifest.xml`.

4.3.1. Declaración de permisos de acceso

En la primera parte del archivo `manifest` se deben declarar los permisos de acceso de la aplicación.

Los primeros permisos son los de la cámara y del reproductor de música respectivamente. Se describen de la siguiente manera:

```
...
<uses-permission android:name="android.permission.CAMERA" />
<permission
android:name="android.permission.MEDIA_CONTENT_CONTROL" />
...
```

Este siguiente permiso es utilizado en múltiples procesos, pero en este caso específico, se implementó para que los audios sean interrumpidos cuando ingresa una llamada:

```
...
<uses-permission
android:name="android.permission.READ_PHONE_STATE" />
...
```

La diferencia entre el permiso `uses-permission` y el `permission`, es que el primero debe ser obligatoriamente concedido por el usuario. Con los permisos declarados se puede proceder con la siguiente parte de `AndroidManifest.xml`.

4.3.2. Datos generales de la aplicación

En la segunda parte del archivo `manifest` se van a declarar características generales de la aplicación, como el nombre, el logo o ícono y el tema o *template* escogido.

```
...
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher"
    android:supportsRtl="true"
    android:theme=“@style/AppTheme”>
...
```

Como se puede observar, en `label` se llama al recurso `string app_name`, en el cual se guardó el texto con el nombre de la aplicación que fue previamente declarado en el archivo `strings.xml`. De igual manera, en `icon` se llama al ícono con nombre `ic_laucher` guardado previamente en la carpeta `drawables`.

4.3.3. Declaración de los componentes de la aplicación

Los componentes de la aplicación también deben ser declarados en el archivo `AndroidManifest.xml`. Se implementaron tres `activities` y un `service`, como se muestra a continuación.

- **Declaración de *activities***

Se implementaron tres `activities` principales. Cabe aclarar que la apariencia de cada uno se modifica por separado en otra sección del programa. En ésta solo se declara la existencia de las actividades para que el programa las lea y las pueda ejecutar.

La primera es el `WelcomeActivity`, que es la pantalla de inicio o bienvenida de la aplicación. Como se puede observar, se declara primero el nombre del `activity`, luego la orientación de la pantalla, que en este caso es vertical o *portrait* y por último el estilo o *theme* que es el estándar de Android.

```
...
<activity
     android:name=".WelcomeActivity"
     android:screenOrientation="portrait"
     android:theme="@style/AppTheme.NoActionBar.Transparent">
...
```

La segunda actividad es `MainActivity`, el cual representa la pantalla donde se encuentra la cámara para poder leer los cQR. Se debe incluir una porción de código para activarla apenas se abra la aplicación por medio de un `intent`. Un `intent` es un 'objeto' que sirve para llamar un `activity`.

```
...
<activity
   android:name=".MainActivity"
   android:screenOrientation="portrait"
   android:theme="@style/AppTheme.NoActionBar.Transparent"/>
<intent-filter>
   <action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"/>
 </intent-filter> </activity>
...
```

Por último, se declara el `ScrollingActivity` que es la pantalla donde se encuentra la información de cada lugar. Contiene las mismas líneas de código que el `WelcomeActivity`.

```
...
<activity
    android:name=".ScrollingActivity"
    android:label="@string/title_activity_scrolling"
    android:screenOrientation="portrait"
    android:theme="@style/AppTheme.NoActionBar" />
...
```

- **Declaración de servicios**

Para esta aplicación solo se implementó un `service`, el cual permite que los audios se reproduzcan en segundo plano, sin interrupción. El nombre del servicio es `MusicService` y fue declarado de la siguiente manera como se indica en las siguientes líneas de código.

```
...
<service
     android:name=".MusicService"
     android:enabled="true"
     android:exported=“true”/>
...
```

4.4. Interfaz gráfica de usuario

En esta sección se detalla el proceso de desarrollo de la interfaz gráfica de los `activities` declarados anteriormente. Se especifican las partes u ‘objetos‘ visuales de cada pantalla de la aplicación para, posteriormente, asignarles a dichos objetos procedimientos a cumplir (el código de funcionamiento es explicado en el siguiente capítulo).

Como se explicó, la interfaz gráfica está designada por los diferentes `activities`. La ventaja que brinda Android Studio es que éstos pueden ser configurados a manera de código o visual (arrastrando e insertando objetos).

Se decidió aplicar el método visual por su facilidad. Una vez agregados los objetos, el programa se encarga de representarlos a manera de código.Para poder diseñar las interfaces gráficas hay que acceder a las carpeta `layouts` dentro de la carpeta `res` (referente a *resources*), como indica la *figura 11.*

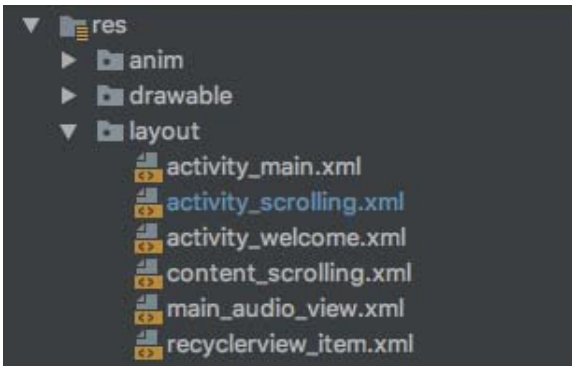

Figura 11. Ubicación de la carpeta `layout` dentro de la capeta `res`.

4.4.1. Diseño del *WelcomeActivity*

Esta `actividad` aparece cuando el usuario abre la aplicación. Es la pantalla de bienvenida que cuenta con el nombre y logotipo de la aplicación en la parte superior y un mensaje de bienvenida en la parte inferior. Adicionalmente, posee un botón el cual dirige hacia el `MainActivity` para escanear los códigos. El `WelcomeActivity` se observa en la *figura 12* ubicada en la siguiente página.

4.4.2. Diseño del *MainActivity*

Esta actividad representa la pantalla donde se encuentra insertada la cámara que funciona como lector de cQR. En la parte superior se encuentra escrito el nombre de la aplicación, seguido por la sección de la cámara, un *textbox* que dice

“Escaneando Código QR” y un botón que permite regresar a la pantalla de bienvenida. El `MainActivity` se observa en la *figura 13*.

Figura 12. Visualización del `WelcomeActivity`.

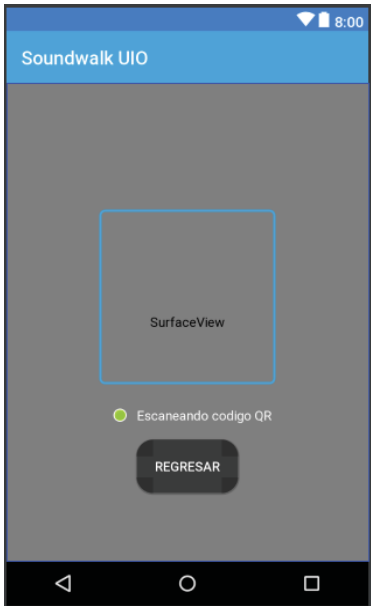

Figura 13. Visualización del `MainActivity`.

4.4.3.Diseño del *ScrollingActivity*

Esta actividad representa la pantalla en que se visualiza la información para los diferentes destinos turísticos. El `activity` es el mismo para todos pero los recursos (imágenes, texto y audio) cambian dependiendo del código leído.

Esta actividad posee el nombre, una imagen, la descripción y el `MediaPlayer` con los audios del cada lugar. Adicionalmente se insertó un botón con acceso a un *RecyclerView* donde se visualizan más fotografías. En la *figura 14* se puede observar la representación del `ScrollingActivity`.

Figura 14. Visualización del `ScrollingActivity`.

4.4.4. Diseño del *RecyclerView*

Un `RecyclerView` es una herramienta que permite la visualización de objetos de manera interactiva con el usuario. Al ser un elemento visual forma parte de la interfaz gráfica, por lo que se encuentra también dentro de la carpeta `layout`.

Para ubicar el `RecyclerView`, fue necesaria la creación de un `fragment` que es una instancia temporal en el programa, ya que no se encuentra en un `activity` como tal. Dentro del `fragment` se encuentra el `RecyclerView`.

En la *figura 15* se observa una vista preliminar del `RecyclerView` que contiene una imagen y una descripción de la misma.

Figura 15. Visualización preliminar del `RecyclerView`.

4.5. Código de funcionamiento de la aplicación

El código principal es aquel que contiene todos los procedimientos para que la aplicación cumpla con lo propuesto. Los 'objetos' de cada `activity` deben ser programados para que cumplan con sus respectivas funciones.

Los archivos donde se van a escribir los códigos de funcionamiento de la aplicación, también denominadas 'clases', se encuentran en la parte izquierda de la carpeta `java`, como indica la *figura 16*.

Cada uno de estos representa una 'clase' dentro del programa. Las clases son plantillas predeterminadas para la creación de 'objetos'. Los objetos van a incluir cada uno de los procesos que la aplicación debe realizar.

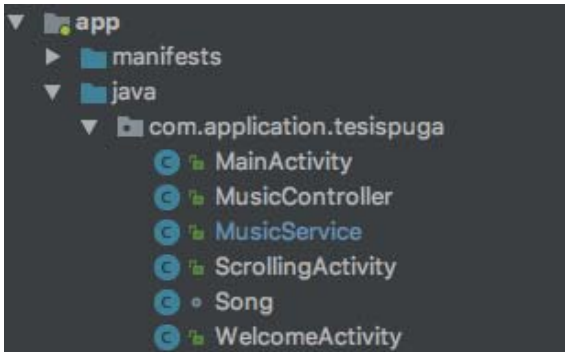

Figura 16. Ubicación de la carpeta `java` con las 'clases' del programa.

4.5.1. Procesos del *WelcomeActivity*

En el `WelcomeActivity`, al ser una pantalla de bienvenida, lo único que se debe programar es al botón para que de acceso hacia el `MainActivity`. El procedimiento es el siguiente.

Primero se inicializa la clase y se insertan dos componentes de librería: un botón y un `intent` con sus respectivos nombres. Un `intent` es un objeto para acceder a una nueva actividad.

```
...
public class WelcomeActivity extends AppCompatActivity
implements View.OnClickListener{
   Button qrButton;
   Intent intent;
...
```

Después, se inicializa el `activity` con la función `OnCreate` y se llama a la interfaz gráfica previamente creada para que aparezca en pantalla.

```
...
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_welcome);
...
```

Luego se activa el botón y se le brinda una imagen con el recurso insertado en la carpeta `drawable`. Por esta razón el botón tiene aspecto de un cQR.

```
...
qrButton = findViewById(R.id.bt_welcome_qr);
qrButton.setOnClickListener(this);
...
```

Por último se programa al botón para que, en el caso de darle click, la actividad cambie y le de paso al `MainActivity`. Esto se logra gracias a la función `onClick`.

```
...
public void onClick(View view) {
     switch (view.getId()){
        case R.id.bt_welcome_qr:
          intent = new Intent(getApplicationContext(),
MainActivity.class);
          startActivity(intent);
           break;
         default:
       break;}
...
```

4.5.2. Procesos del *MainActivity*

Esta actividad es la más compleja ya que realiza los procesos de decodificación del cQR y asignación de caracteres a variables que se asocian con los diferentes recursos a ser mostrados en el `ScrollingActivity` posteriormente.

La actividad cuenta con la cámara, un botón para regresar, un *led* y un *textbox.* Todos deben ser programados para que funcionen correctamente. Igual que en el `activity` anterior, se inicializa a la actividad creando la clase de la misma.

```
...
public class MainActivity extends AppCompatActivity
implements View.OnClickListener{
...
```

Luego se insertan componentes de librería con su respectivo nombre y se crean `strings` para que adquieran valores posteriormente. Los componentes de librería están predeterminados para realizar acciones necesarias y los `strings` ayudan a almacenar datos (caracteres en este caso).

```
...
    BarcodeDetector barcodeDetector;
    CameraSource cameraSource;
    SurfaceView cameraView;
    ImageView imageViewLed;

private String token = "";
private String tokenanterior = "";
private Button buttonBack;
public static final String EXTRA =
"com.application.tesispuga.MESSAGE";
...
```

A continuación se inicializa el `activity` con `OnCreate` y se activa el botón de retorno para que tenga una imagen de los recursos.

```
...
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

buttonBack = findViewById(R.id.button_main_back);
buttonBack.setOnClickListener(this);
...
```

La cámara funciona como lector de los códigos, por lo que hay que seguir un proceso para que funcione correctamente.

Primero se alista un comando de librería que activa el detector de código de barras, el cual viene incluido en la API del *software*.

```
...
barcodeDetector =
   new BarcodeDetector.Builder(getApplicationContext())
            .setBarcodeFormats(Barcode.QR_CODE)
            .build();
...
```

Posteriormente, se activa la cámara como entrada de información y se muestra en pantalla lo que está recibiendo.

Se debe verificar también si el usuario dio los permisos para utilizar la cámara. En el caso de que el permiso de acceso no fue concedido la aplicación muestra un mensaje de error.

```
...
cameraSource = new CameraSource
                .Builder(getApplicationContext(),
barcodeDetector)
                .setRequestedPreviewSize(1024, 768)
                .build();
cameraView = (SurfaceView) findViewById(R.id.camera_view);
cameraView.getHolder().addCallback(new
SurfaceHolder.Callback()
...
```

Luego se prepara el detector de cQR para que analice y decodifique.

```
...
barcodeDetector.setProcessor(new
Detector.Processor<Barcode>() {
public void release() {
            }
public void receiveDetections(Detector.Detections<Barcode>
detections) {
final SparseArray<Barcode> barcodes =
detections.getDetectedItems();
...
```

En la siguiente sección de código se determina si el caracter obtenido es mayor a 0, si este es el caso se lo guardará en el `string token`. Para que el lector solo reconozca códigos pertenecientes a la aplicación, se condiciona al valor obtenido para que sea menor que 3 (podrían ser los valores 0, 1 y 2), caso contrario no se ejecuta ninguna acción. Por último se asigna al objeto `intent` para acceder al `ScrollingActivity`.

```
...
if (!flag){
     if (barcodes.size() > 0) {
        token = barcodes.valueAt(0).displayValue.toString();
        tokenanterior = token;
        Log.i("token", token);

         if (Integer.parseInt(token) < 3){
            Intent intent = new
Intent(getApplicationContext(), ScrollingActivity.class);
            intent.putExtra(EXTRA, token);
            startActivity(intent);
            flag = true; }}}
...
```

El último paso es configurar al botón que permita regresar al `WelcomeActivity`.

```
...
public void onClick(View view) {
    switch (view.getId()){
        case R.id.button_main_back:

Intent intent = new Intent(getApplicationContext(),
WelcomeActivity.class);
        startActivity(intent);
            break;
        default:
       break;
        }
...
```

4.5.3. Procesos del *ScrollingActivity*

El `ScrollingActivity` es la última de las actividades y es la que presenta la información asociada a los diferentes códigos. En esta sección se configuran títulos, descripciones, fotos, audios y un botón que direcciona al `RecyclerView`.

La idea es que la aplicación muestre los recursos de cada lugar, o sea, que éstos cambien dependiendo del código leído. Para conseguir esto se realizó un arreglo o *array* que ordena todos los recursos para que sean llamados en orden dentro del `ScrollingActivity`. Los arreglos son mostrados más adelante pero en esta sección se explica su implementación.

El primer paso es generar la clase e insertar componentes y variables. Se tienen componentes y variables del `activity`, `service` y `array` respectivamente, ya que todos son utilizados en esta actividad.

```
...
Public class ScrollingActivity extends AppCompatActivity{
    TypedArray imgs;
    String[] titles;
    String[] description_array;
    AppBarLayout appBarLayout;
    TextView content;
    String extra;
    PhotoAlbumFragment photoAlbumFragment;

    private MusicService musicService;
    private Intent playIntent;
    private boolean musicBound = false;
    boolean playing = false;
    TextView infoPlayer;

    int resArrayIndex;
    public int getResArrayIndex() {
    return resArrayIndex;
    }
...
```

Posteriormente se inicializa la actividad.

```
...
protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_scrolling);
...
```

El paso que viene es fundamental, ya que se asigna a una nueva variable que va a contener al valor obtenido en el anterior activity (proveniente del cQR). Éste valor no numérico, que estaba en la variable `EXTRA`, es ahora almacenado en una nueva variable `extra`. Al ser un caracter no numérico, se lo debe convertir en un número entero para que pueda servir como referencia en los `arrays`. El número entero es almacenado en la variable `resArrayIndex`, la cual será utilizada más adelante.

```
     ...
     Intent intent = getIntent();
     extra = intent.getStringExtra(MainActivity.EXTRA);
     resArrayIndex = Integer.parseInt(extra);
     ...
```

El siguiente paso es llamar a los `arrays` *d*e títulos, imágenes y de descripción de los diferentes destinos.

```
     ...
     titles = getResources().getStringArray(R.array.backs_title);
     imgs = getResources().obtainTypedArray(R.array.backs_imgs);
     description_array =
     getResources().getStringArray(R.array.backs_description);
     ...
```

Por último, se asigna el número entero del 0 al 2, contenido en la variable `resArrayIndex`, como número de ítem de cada `array`.

```
...
final String title = titles[resArrayIndex];
final String description = description_array[resArrayIndex];
setTitle(title);
content.setText(description);
...
```

Para configurar el botón que da acceso al `RecyclerView` hay que activar el fragmento que lo contiene, de la siguiente manera.

```
...
FloatingActionButton fab = findViewById(R.id.fab);
   fab.setOnClickListener(new View.OnClickListener()
public void onClick(View view) {
   photoAlbumFragment.show(getSupportFragmentManager(),"Dialo
g");}
...
```

La última parte del código de esta clase tiene relación con el `MusicService` y el `MediaPlayer`. El `MediaPlayer` es el reproductor de audio estándar para estas aplicaciones.

Para iniciar el `MusicService` se debe correr el siguiente código. Como se puede observar, este servicio solo se inicializa si la variable `extra` existe, o sea si un código fue leído con éxito. Cabe aclarar que el contenido del `service` es declarado aparte.

```
...
  if (!musicBound) {
      if(playIntent==null){
          playIntent = new Intent(this, MusicService.class);
          playIntent.putExtra("media", extra);
          startService(playIntent);
          bindService(playIntent, serviceConnection,
Context.BIND_AUTO_CREATE);}}
else {  }
...
```

Para que el servicio funcione en segundo plano hay que activar la función `OnServiceConnected`. El `MediaPlayer` solo funcionará si este parámetro está activado.

```
...
public void onServiceConnected(ComponentName componentName,
IBinder iBinder) {
        MusicBinder binder = (MusicBinder) iBinder;
        musicService = binder.getService();
        musicBound = true;
        Log.i("SERVICE", "Get Service");}

public void onServiceDisconnected(ComponentName
componentName) {
        Log.i("SERVICE", "Disconected Service");
        musicBound = false;
         }};
...
```

4.5.4. Procesos del *MusicService*

El `MusicService` es el único servicio de la aplicación. Éste permite que el reproductor de música implementado pueda funcionar en segundo plano, para poder escuchar los audios aunque el teléfono esté bloqueado o con otra aplicación en primer plano.

En esta sección se debe programar todos los procedimientos referentes al servicio y al `MusicPlayer`, el cual funciona como reproductor de música de aplicaciones Android por defecto. Al ser de librería, gran cantidad de este código viene listo, lo único que hay que hacer es nombrar y cambiar variables para que se ejecute.

Al igual que las `activities`, se inicializa primero la clase del servicio. Se llaman también a algunos comandos de la librería y se implementan variables que serán utilizadas posteriormente. Aquí se incluyen también los `arrays` que servirán para llamar a los recursos almacenados.

```
...
public class MusicService extends Service implements
MediaPlayer.OnPreparedListener,
MediaPlayer.OnErrorListener,
MediaPlayer.OnCompletionListener{
    private MediaPlayer mp;
    private int songPosn;
    private int resumePosition;
    private final IBinder musicBind = new MusicBinder();
    TypedArray songsArray;
...
```

En esta parte se activa el servicio con la función `onCreate`. El reproductor solo funciona mientras el servicio esté encendido.

```
...
public void onCreate() {
   super.onCreate();
   songPosn = 0;
   songsArray =
getResources().obtainTypedArray(R.array.backs_songs);
   mp = new MediaPlayer();
   initMusicPlayer();
    }
...
```

Se llama a la lista de `arrays` para que elija el archivo de audio que corresponde al código escaneado. Es importante aclarar que, para que la convocatoria de un `array` funcione, se deben haber nombrado los recursos y programado de manera óptima los diferentes `arrays`, como se va a explicar en el siguiente capítulo.

```
...
public void setList(ArrayList<Song> theSongs){
    songs = theSongs;}

public class MusicBinder extends Binder{
    MusicService getService() {
       return MusicService.this;}}
...
```

Se registran los comandos que permiten que los botones de *play*, *stop*, *pause* y *resume* del `MediaPlayer` funcionen.

```
...
private void playMedia() {
   if (!mp.isPlaying()) {
       mp.start();
   }}
private void stopMedia() {
    if (mp == null) return;
    if (mp.isPlaying()) {
        mp.stop();
    }}
private void pauseMedia() {
    if (mp.isPlaying()) {
        mp.pause();
        resumePosition = mp.getCurrentPosition();
    }}
private void resumeMedia() {
    if (!mp.isPlaying()) {
        mp.seekTo(resumePosition);
        mp.start();}}
...
```

Por último, se programa para que el servicio se destruya cuando se termine de reproducir un audio y que se prepare para cuando se presione *play* otra vez.

```
...
public void onDestroy() {
    super.onDestroy();
    if (mp != null) {
        stopMedia();
        mp.release();
    }}

public MediaPlayer getMp() {
    return mp;}
...
```

4.5.5. Implementación de *arrays*

Los `arrays`, o arreglos, son herramientas que permiten a los programadores almacenar datos o recursos del mismo tipo de manera ordenada. Cada `array` tiene un nombre y una serie de ítems que lo conforman. Por lo general tienen forma matricial, por lo que se componen de filas y columnas.

En este caso los arreglos son unidimensionales (solo filas) y sus ítems están en orden ascendente, por lo que un número entero representa a cada una de las filas. Cada fila está asociado a un recurso, ya sea imagen, texto o audio. Los nombres de los recursos digitados deben coincidir con los previamente almacenados en sus respectivas carpetas. El texto se puede escribir directamente en el `array` si se desea.

Para esta aplicación se crearon doce `arrays` distribuidos de la siguiente manera.

- **Fotos de portada**

Este `array` se conforma por las fotos de portada que van a aparecer en la parte superior del `ScrollingActivity`, de la siguiente manera.

```
...
<integer-array name="backs_imgs">
        <item>@drawable/back_4</item>
        <item>@drawable/back_3</item>
        <item>@drawable/back_2</item>
</integer-array>
...
```

Por ejemplo, el ítem 0 está asociado a la imagen *back_4* que está guardada dentro de la carpeta `drawable`, el ítem 1 a la imagen *back_3* y así sucesivamente, tal como se observa en el código presentado.

- **Títulos de portada**

Este `array` se conforma por los títulos de portada que aparecen en la parte superior del `ScrollingActivity` de la siguiente manera.

```
...
 <string-array name="backs_title">
    <item>Plaza de la Independecia</item>
    <item>La Compañía</item>
    <item>San Francisco</item>
</string-array>
 ...
```

- **Reseñas cortas**

Este `array` se conforma por los textos de la reseñas cortas que aparecen en la parte inferior del `ScrollingActivity` de la siguiente manera. Los textos completos están presentados más adelante en la parte de diseño del SW.

```
 ...
 <string-array name="backs_description">
    <item>La plaza de la independencia, conocida...</item>
    <item>La iglesia de la Compañía de Jesús es...</item>
    <item>La plaza de San Francisco es la plaza...</item>
</string-array>
 ...
```

- **Audios**

Este `array` se conforma por los audios que se incluyen el `MusicService` de la siguiente manera. Cabe recalcar que hay dos audios por lugar, por lo que al caracter 0 se le asignan `audio1` y `audio2`, al 1 se le asignan `audio3` y `audio4` y al 2 se le asignan `audio5` y `audio6`.

```
...
 <string-array name="backs_songs">
    <item>@raw/audio1</item>
    <item>@raw/audio2</item>
```

```
        <item>@raw/audio3</item>
        <item>@raw/audio4</item>
        <item>@raw/audio5</item>
        <item>@raw/audio6</item>
    </string-array>
     ...
```

- **Álbumes de fotos**

Los siguientes arrays forman parte del `ReciclerView`. Al haber tres lugares se crearon tres arreglos de álbumes, cada uno con las fotos de cada destino. Adicionalmente, hay otros tres `arrays` que contienen las descripciones de cada foto de cada álbum (en el `ReciclerView` cada foto tiene su pequeña descripción).

Por ejemplo, el arreglo `album1` contiene las fotos del caracter o lugar 0, el array `album1_titles` contiene las descripciones para cada foto del `album1`. Así para los otros álbumes.

```
...
 <integer-array name="album1">
    <item>@drawable/a1_f1</item>
    <item>@drawable/a1_f2</item>
    <item>@drawable/a1_f3</item>
    <item>@drawable/a1_f4</item>
</integer-array>

<integer-array name="album2">
    <item>@drawable/a2_f1</item>
    <item>@drawable/a2_f2</item>
    <item>@drawable/a2_f3</item>
</integer-array>

<integer-array name="album3">
    <item>@drawable/a3_f1</item>
    <item>@drawable/a3_f2</item>
    <item>@drawable/a3_f3</item>
</integer-array>
```

```
<string-array name="album1_titles">
    <item>Plaza de la Independencia. </item>
    <item>Monumento a la Independencia.</item>
    <item>Gallito de la Catedral. </item>
    <item>Placa conmemorativa a Gabriel García Moreno.</item>
</string-array>

<string-array name="album2_titles">
    <item>Vista lateral de la Iglesia de la Compañía. </item>
    <item>Vista frontal de la Iglesia de la Compañía. </item>
    <item>La Compañía en la Fiesta de las Luces 2016. </item>
</string-array>

<string-array name="album3_titles">
    <item>Exterior de la Plaza San Francisco. </item>
    <item>Panorámica frontal de Plaza San Francisco. </item>
    <item>Enfoque desde abajo de la Plaza San Francisco.</
     item>
</string-array>
  ...
```

Por último se crean dos arreglos más. El primero contiene a los tres `arrays` de álbumes de fotos generados previamente y el segundo contiene los tres álbumes de descripciones. Estos son precisamente aquellos que forman parte del `ReciclerView`.

```
...
 <integer-array name="album_array">
    <item>@array/album1</item>
    <item>@array/album2</item>
    <item>@array/album3</item>
</integer-array>

<integer-array name="album_titles_array">
    <item>@array/album1_titles</item>
    <item>@array/album2_titles</item>
    <item>@array/album3_titles</item>
</integer-array>
  ...
```

4.6. Generador de códigos QR

Una vez que los códigos pueden ser leídos, hay que generarlos con la información que se necesita que decodifique. Como se explicó, la aplicación contará con tres destinos turísticos para probar su funcionamiento. Esto quiere decir que los caracteres a codificar son el 0, 1 y 2. Se eligieron números porque el código está programado para que éstos representen un ítem o fila en los diferentes `arrays`.

Existen innumerables páginas web gratuitas que permiten generar cQR de manera inmediata. Se utilizó en este caso una página llamada grcode-generator.com.

Lo único que estas páginas requieren es que se tenga establecido lo que se quiere codificar. Normalmente los cQR son utilizados para contener URL de páginas web, pero en este caso se van a codificar los caracteres mencionados.

La *figura 17* muestra un ejemplo de cómo se genera un cQR, en este caso con el caracter 0. El cQR proporcionado por la página web puede ser leído por la aplicación inmediatamente.

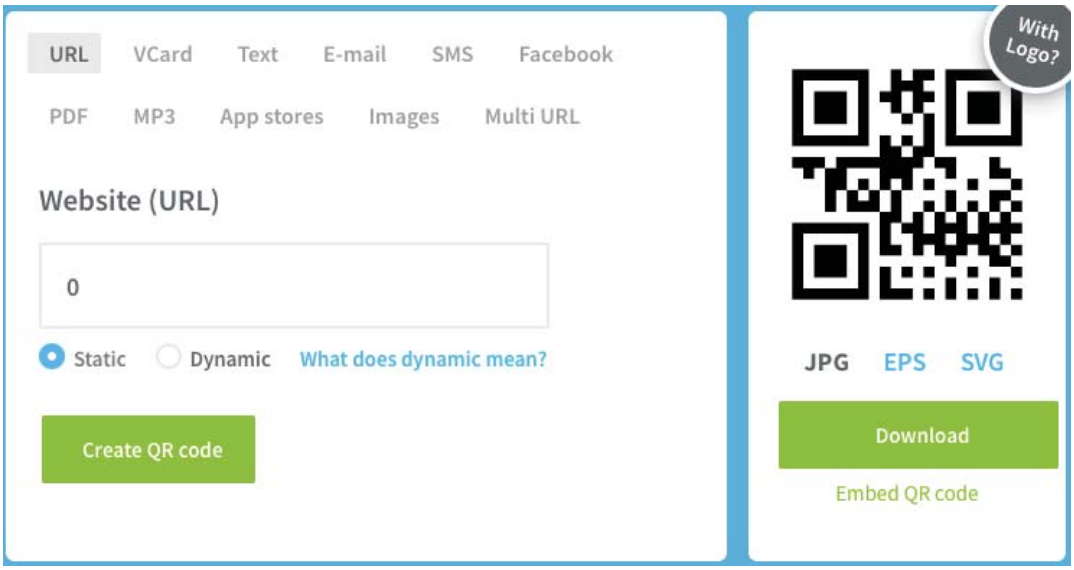

Figura 17. Ventana de grcode-generator.com, código QR para el caracter 0. Tomado de (QR Code Generator, 2017).

5. DISEÑO DEL *SOUNDWALK*

5.1. Elección de los atractivos turísticos y recursos

Para probar el funcionamiento de la aplicación Soundwalk UIO se decidió elegir tres destinos turísticos populares del CH. Para cada destino la aplicación mostrará imágenes, una pequeña descripción y audios con información turística.

Con los destinos elegidos se pretende crear una ruta, por lo que se procuró que estén uno cerca del otro. Estos lugares forman parte de este prototipo de la aplicación, por lo que pueden ser cambiados o incrementados en un futuro.

Los tres destinos elegidos para la prueba de la aplicación son los siguientes:

- Plaza de la Independencia.
- Iglesia de la Compañía de Jesús.
- Plaza San Francisco.

Cada destino se compone de tres elementos o recursos fundamentales: audio, texto e imágenes. Se pretende que todos los recursos sean de calidad para una buena demostración del funcionamiento de la aplicación.

5.2. Diseño de sonido

Los audios forman parte importante de la aplicación ya que son la principal fuente de información. Cada uno de los destinos turísticos contiene dos archivos de audio: reseña histórica y personaje o leyenda característica. Cada audio se compone de diálogo y música acompañante.

5.2.1. Producción de diálogos

Los diálogos fueron en su totalidad producidos en un *home studio* con herramientas y equipos semi-profesionales. La voz pertenece a Juan José Oleas, quien narró todos los diálogos y se encargó de dar vida a las historias.

Para las etapas de grabación, mezcla y máster se utilizaron las siguientes herramientas proporcionadas por el *home studio* mencionado:

- Micrófono Rode NT1-A.
- Filtro antipop.
- Interfaz AVID Fast Track Duo.
- Computador Mac Mini.
- DAW Pro Tools 12.6.
- Audífonos de estudio Yamaha Pro 400.
- Monitores Samson Resolv A5.

En la *figura 18* se muestra el flujo de señal en forma de diagrama de bloques perteneciente a la configuración del estudio donde se grabaron las voces.

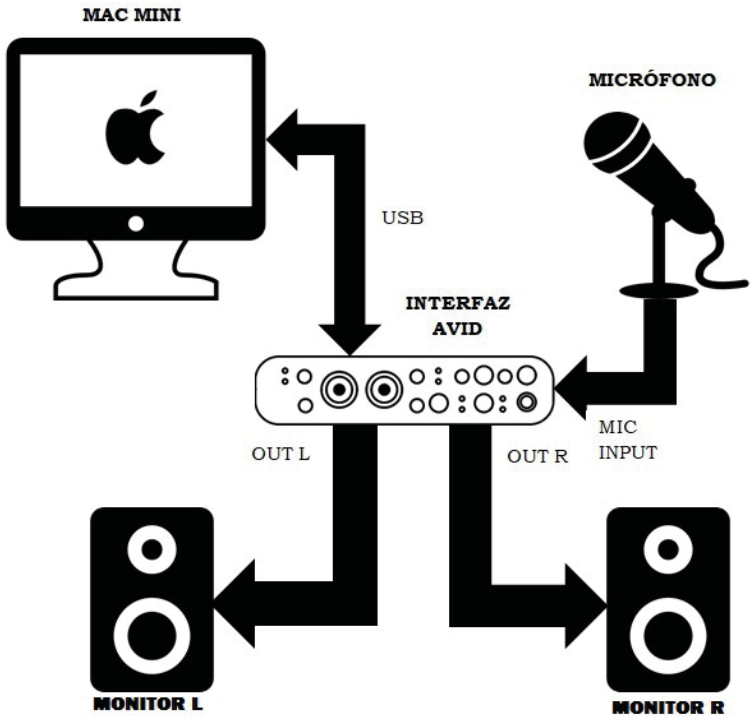

Figura 18. Diagrama de bloque y flujo de señal para la grabación de diálogos.

- **Etapa de grabación**

Para la etapa de grabación se tomó en cuenta las técnicas de grabación aprendidas durante la carrera.

Se utilizó el micrófono de condensador Rode NT1-A de patrón polar cardioide, muy popular para grabación de voces. Se colocó el micrófono aproximadamente a 30 cm de separación con respecto a la boca del locutor, con un filtro anti-pop a 7 cm del micrófono. El pedestal fue ubicado en el centro de la sala para evitar que reflexiones de paredes o ventanas cercanas sean captadas por el micrófono.

La distancia del locutor con el micrófono fue la óptima para poder captar de manera clara las narraciones y evitar coloración fuera del eje. Tomar en cuenta todos estos factores permiten que la grabación sea la adecuada y se eviten problemas de inteligibilidad. Un locutor con experiencia también facilita el proceso, ya que factores como la respiración, tono y énfasis de la voz afectan directamente con la calidad de los audios.

Para cada archivo de audio (6 en total), se creó una sesión de Pro Tools. En la misma sesión se hicieron los procesos de edición, mezcla y máster respectivamente.

- **Etapa de edición**

La etapa de edición es considerada una de las más tediosas dentro de la producción de audio. En esta etapa se deben elegir las mejores tomas de cada grabación. Se deben también hacer cortes para eliminar partes no deseadas o sonidos molestos como filtraciones, sibilancias, popeos, respiraciones, etc. Se editan todas las pistas de audio para que estén listas para ser mezcladas.

Afortunadamente, las herramientas de edición que brinda Pro Tools son muy prácticas y rápidas de ejecutar. La herramienta de *fade in* o *fade out,* por ejemplo, es necesaria cada vez que se hace un corte para evitar cambios bruscos, sonidos

molestos y especialmente clicks producidos al cortar mal una pista. En la *figura 19* se observa un ejemplo de una pista de voz con sus respectivos cortes y *fades*.

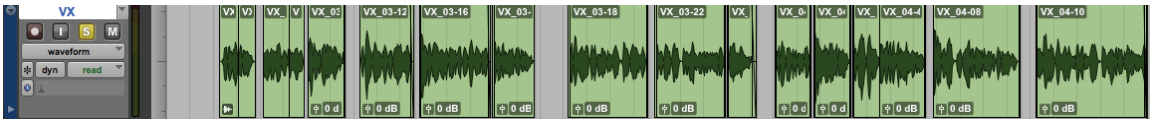

Figura 19. Screenshot de una pista de voz editada.

- **Etapa de mezcla**

Después de la grabación y edición, se procedió con la etapa de mezcla. Se busca que la pista de la voz encuentre un punto de equilibro, tanto en niveles como en frecuencias, con la de la música. Las canciones elegidas son analizadas en el siguiente capítulo, pero el proceso de mezcla se detalla a continuación.

Parte importante del proceso de mezcla es el procesamiento de las señales. Para todos los audios se contó con un mismo narrador, por lo que los procesadores para las voces de cada sesión fueron los mismos, así como sus parámetros. Esto se logró por medio de *presets* que brindan los diferentes *plug-ins.*

El primer procesador utilizado en la voz fue un reductor de ruido para eliminar ciertos *clicks* producidos por la voz y la saliva. La *figura 20* muestra la configuración del *plug-in* X-Click de la compañía Waves.

Figura 20. Screenshot de la configuración de X-Click.

Posteriormente se aplicó un ecualizador para aplicar ecualización sustractiva y eliminar frecuencias de resonancia o molestas. Esto se logra mediante barrido de frecuencias. Una vez que se encuentran las frecuencias de resonancia, se atenúan entre -7 y -8 dB. Un filtro paso alto también ayuda a eliminar frecuencias bajas innecesarias. La *figura 21* muestra la configuración del *plug-in* Q6 de la compañía Waves.

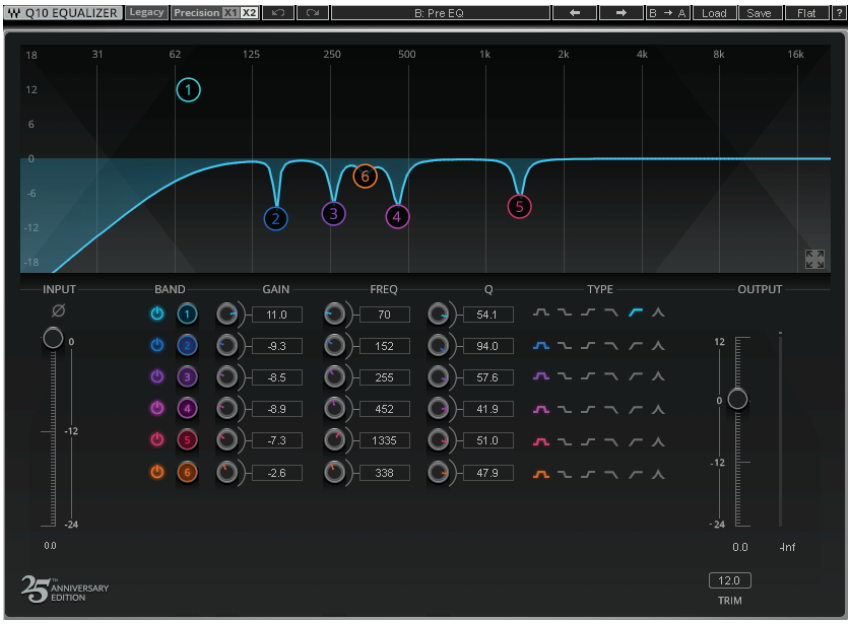

Figura 21. Screenshot de la configuración de Q6.

También se aplicó un compresor para atenuar los picos de la señal. Es importante aplicar este procesador después de la ecualización sustractiva, ya que no se desea que esas señales no deseadas afecten al comportamiento del compresor. La *figura 22* muestra la configuración del *plug-in* CLA-2A de la compañía Waves.

Posteriormente se aplicó un DeEsser que ayuda a atenuar los sonidos exagerados de las ’s’. Funciona prácticamente como un compresor por banda. En este caso se encontró molestia a partir de los 4 kHz. La *figura 23* muestra la configuración del *plug-in* Reinassance DeEsser de la compañía Waves.

Figura 22. Screenshot de la configuración de CLA-2A.

Figura 23. Screenshot de la configuración de Reinassance DeEsser.

Para la música solo se aplicó un EQ para crear espacio en el espectro de frecuencias para la voz. También se hizo un barrido de frecuencias en caso de que haya resonancias. Las canciones elegidas ya pasaron por los procesos de edición, mezcla y máster, por lo que no se le aplicó mayor procesamiento.

Para esta etapa también fue de suma importancia la implementación de un medidor de señal, que permita la lectura de valores picos y RMS de todos los *tracks* de cada sesión. Esto ayudó a tener como referencia los niveles, tanto de la voz como de la música, para aplicar a los 6 audios. El plug-in elegido para esto fue el WLM Meter de Waves, como indica la *figura 24*.

Figura 24. Screenshot de la configuración de WLM Meter.

El último paso de la mezcla fue automatizar la pista de la música para que no influya con la de la voz. Esto se logró con las herramientas de automatización de Pro Tools. Un ejemplo de pista automatizada se puede ver en la *figura 25*.

Figura 25. Screenshot de una pista de música automatizada.

- **Etapa de masterización**

En esta etapa se busca aplicar procesadores a la pista generada durante la mezcla. También fue importante la utilización de *presets* para ahorrar tiempo.

En este caso se implementó un compresor especializado en canales máster para que combine de manera sutil a las pistas de voz y música.´este procesador actúa como un 'pegamento' entre ambas pistas. En la *figura 26* se muestra el compresor Solid Bus Comp de la compañía Native Instruments.

Figura 26. Screenshot de la configuración de Solid Bus Comp.

Se pretendía también aplicar un EQ muy sutil pero se decidió no utilizarlo ya que no causaba mucha diferencia y se quería ahorrar procesamiento del computador.

Figura 27. Screenshot de la configuración de L3-LL UltraMaximizer.

No obstante, se aplicó un maximizador que permite a la pista máster alcanzar niveles estándar. Por lo general, se estima como nivel máximo para grabaciones profesionales -0.3 dBFS. Lo que hace este procesador es subir el nivel a lo que esté por debajo del umbral establecido y limitar por completo en -0.3 dBFS, como se indica en la *figura 27* de la página anterior.

5.2.2.Elección de la música

Para acompañar a los diálogos, se incluyeron pistas instrumentales en todos los audios para darles un sentido más musical. Las canciones fueron adquiridas de un proyecto llamado “Florilegio del pasillo ecuatoriano”, desarrollado por Carlos Herrera Sánchez, quien busca promover el pasillo nacional. El proyecto mencionado es sin fines de lucro y promueve este género musical tradicional, por lo que se pudo utilizarlas sin tener problemas de derechos de autor (Herrera, 2017).

Como se mencionó, al ser canciones ya producidas y previamente mezcladas y masterizadas, no se les realizó ningún tipo de procesamiento individual en la etapa de mezcla, excepto lo mencionado. La lista de canciones implementadas se las puede encontrar en la sección de anexos.

5.3. Redacción del texto

Otro recurso de la aplicación es el texto, utilizado para presentar una pequeña reseña descriptiva de cada lugar. Los textos presentados a continuación son los mostrados dentro de la aplicación.

5.3.1. Plaza de la Independencia

La plaza de la Independencia, conocida también como Plaza Grande, es la más representativa de la ciudad de Quito con un área aproximada de 8100 m². Es considerada, junto al resto del Centro Histórico, patrimonio de la humanidad por la UNESCO.

Tiene como característica principal el Monumento a la Independencia. Está rodeada por el palacio de Carondelet, la Catedral, el Palacio Arzobispal y el Municipio de Quito. Se encuentra ubicada entre las calles García Moreno, Chile, Venezuela y Espejo. Los inicios de su construcción se dieron alrededor del año 1564 y fue terminada en su totalidad en el año 1970.

La plaza cuenta con un precedente histórico muy rico, entre historias y acontecimientos y es sede de una de las leyendas más famosas que existen en la ciudad: la leyenda “El Gallito de la Catedral”.

5.3.2. Iglesia de la Compañía de Jesús

También conocida como La Compañía, es considerada uno de los lugares más hermosos para visitar en Ecuador y Latinoamérica por ser el mayor exponente de la arquitectura barroca. Se encuentra en las calles García Moreno y Sucre. La iglesia es la “Casa Madre” de los Jesuitas en Ecuador y fue construida entre los años 1597 y 1765, aproximadamente. Es considerada por la UNESCO como patrimonio de la humanidad.

La iglesia posee cuatro estilos de arquitectura: el barroco principalmente, el mudéjar o morisco, el churrigueresco y el neoclásico. La principal característica de la iglesia en cuanto a su decoración interna, es que cuenta con formas en madera de cedro tallada y bañada con pan de oro de 23 quilates, que es una característica del estilo barroco.

5.3.3. Plaza San Francisco

La plaza de San Francisco es la plaza más grande del centro histórico de Quito. En ella se encuentra la iglesia y convento de San Francisco. La plaza ha funcionado como mercado, espacio para concentraciones militares y políticas, así como lugar de recreación social. Su construcción empezó entre los años 1497 y 1533 y terminó en el siglo XVII.

La plaza alberga una de las leyendas más tradicionales e importantes de la ciudad de Quito, la famosa leyenda de Cantuña. La leyenda se recuerda normalmente en las fiestas quiteñas.

5.4. Imágenes de los lugares

Para evitar problemas de *copyright*, las fotografías a ser implementadas en la aplicación fueron tomadas por los autores. Se contó con dos cámaras diferentes: una Canon Powershot de 20 MP y la cámara frontal de un iPhone 6s de 12 MP.

Se obtuvieron múltiples tomas de los diferentes lugares y se escogieron solo las mejores y más representativas. Las imágenes forman parte de la interfaz gráfica de usuario y se las puede observar en el `RecyclerView` perteneciente a cada lugar, dentro de la aplicación.

De la Plaza de la Independencia se insertaron cuatro fotos, de la Iglesia de la Compañía tres y de la Plaza San Francisco otras tres. En el capítulo Anexos se pueden visualizar todas la imágenes mencionadas.

6. ANÁLISIS DE RESULTADOS

Para evaluar el trabajo realizado se llevó a cabo una encuesta los alrededores de los tres destinos elegidos con la ayuda de Google Forms. Adicionalmente se hizo un análisis económico tomando en cuenta aspectos como herramientas utilizadas y tiempo invertido. El fin de este análisis es de evaluar si el desarrollo comercial de una aplicación de este tipo tendría éxito o no.

6.1. Análisis de la encuesta

La encuesta está conformada por dos partes principales, una sobre características generales del turismo en Quito y otra sobre la aplicación desarrollada. Gracias a esto se pudieron analizar aspectos positivos y posibles mejoras de la aplicación.

Para poder evaluar se decidió sacar una muestra de los visitantes del CH. De la cantidad de visitantas anuales que recibe el CH, se sacó un valor aproximado diario. Este valor se redujo tomando en cuenta las edades de los visitantes que usarían la aplicación (entre 15 y 45 años que son las edades promedio para el uso del dispositivos móviles). Se llegó a la conclusión que una muestra de 30 personas sería suficiente para evaluar el comportamiento de la población.

Las imágenes presentadas a continuación forman parte de los resultados que la herramienta Google Forms brinda.

6.1.1. Resultados generales

La encuesta incluye preguntas que permiten analizar el comportamiento y los hábitos turísticos de las personas que visitan el CH de Quito. A continuación se presentan los resultados encontrados.

Como se puede observar en la *figura 28*, solo una tercera parte de los encuestados no ha utilizado nunca un servicio de guía turístico en Quito. Esto quiere decir que la gran mayoría de las personas que visitan el CH optan por este servicio para conocer más sobre la ciudad.

Figura 28. Diagrama de pastel acerca del contrato de servicio de guía turístico.

En la *figura 29* se observa que a la mayoría, el servicio de guía turístico le pareció bueno. Esto abre campo a nuevos sistemas de turismo que gusten y atraigan más a los visitantes.

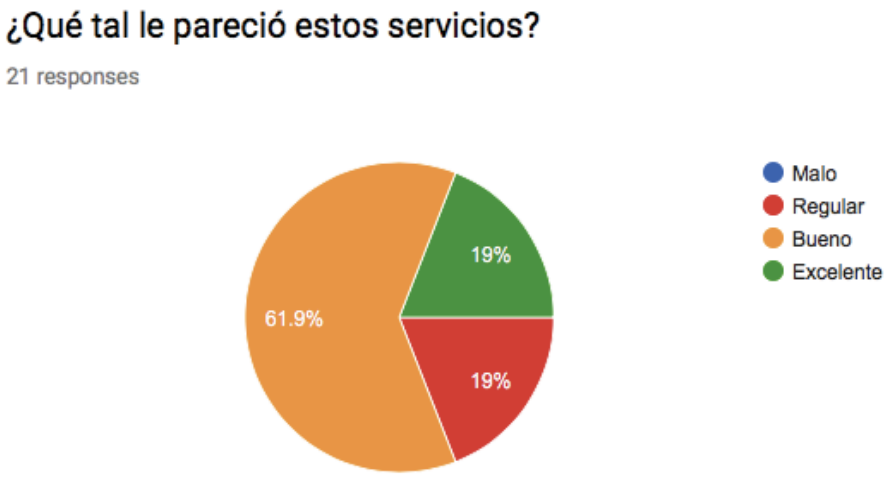

Figura 29. Diagrama de pastel acerca de la opinión del servicio de guía turístico.

En la *figura 30* se observa que casi el 90% de los visitantes han usado una aplicación móvil con fines turísticos. Con el auge de la tecnología la gran mayoría de personas hace uso de aplicaciones para sus actividades favoritas. De los encuestados la mayoría han utilizado Google Maps y Trip Avisor.

Figura 30. Diagrama de pastel acerca del uso de aplicaciones con fines turísticos.

La *figura 31* muestra que la gran mayoría de encuestados adquieren información turística del Internet. Tan solo 13% la adquieren de aplicaciones móviles, lo que puede perjudicar un poco a la aplicación desarrollada.

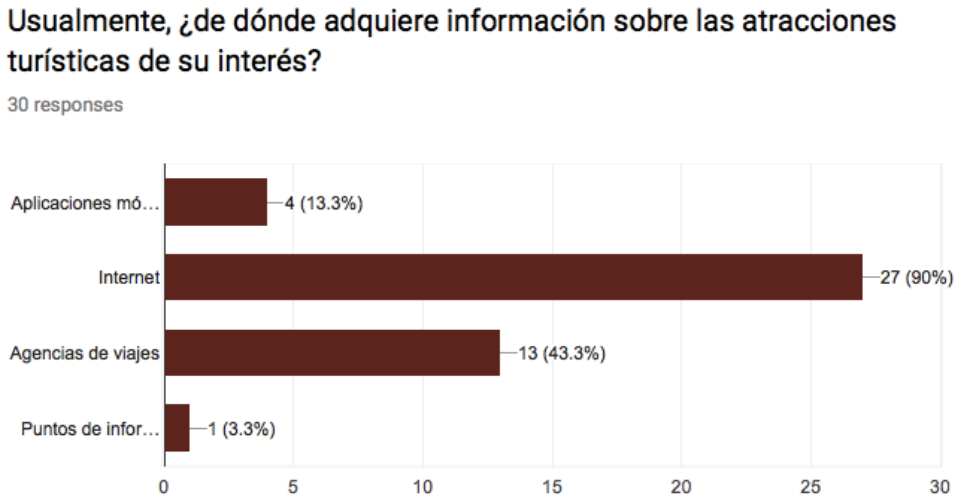

Figura 31. Diagrama de barras acerca de la fuente de información turística.

La *figura 32* muestra que solo el 30% de los encuestados conoce lo que es un SW y tan solo una persona ha utilizado una aplicación para la realización de SW. Esto puede ser visto como una ventaja, ya que las actividades nuevas llaman la atención de la gente que siempre está en busca de experiencias novedosas.

Figura 32. Diagrama de pastel acerca del conocimiento de *soundwalks*.

6.1.2. Resultados de la aplicación

En la *figura 33* se observa que casi el 100% de los encuestados piensan que la idea y la aplicación son excelentes. Esto comprueba que el trabajo realizado es totalmente aplicable en un futuro.

Figura 33. Diagrama de pastel acerca de la aplicación e idea en general.

Acerca de la interfaz gráfica, el 73% concuerda que es buena, pero no excelente. Algunos encuestados opinaron que es demasiado simple. Éste es un aspecto que debe ser considerado para mejorar en un futuro. También se recomendó agregar videos o imágenes 360. La *figura 34* muestra lo expuesto.

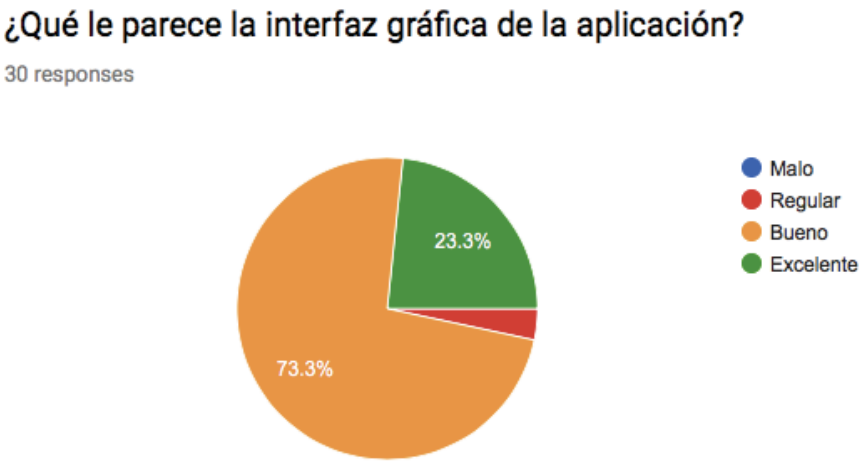

Figura 34. Diagrama de pastel acerca de la interfaz gráfica de la aplicación.

Acerca de los audios, al 80% le pareció excelente y al 20% bueno. Esto quiere decir que se puede mejorar. La implementación de efectos y una mejor redacción pueden ser claves para mejorar en este aspecto. El balance entre voces y música fue el adecuado.

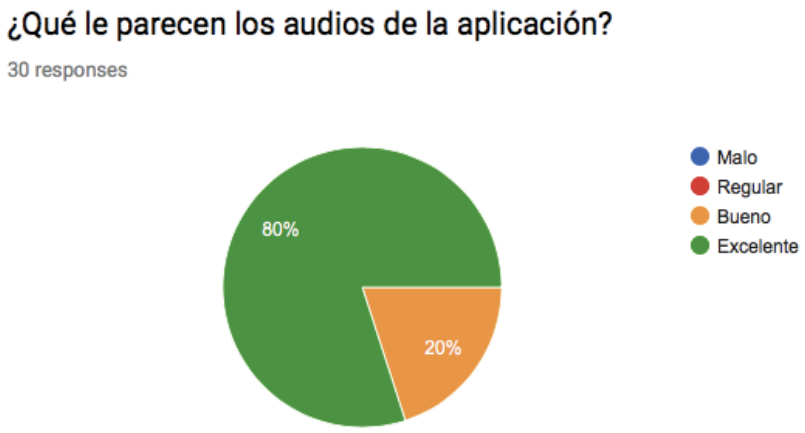

Figura 35. Diagrama de pastel acerca de los audios de la aplicación.

Del 100% de personas que respondieron que sí utilizarían este servicio, un 43% preferiría la aplicación a un guía turístico tradicional y más del 50% contestaron que depende del momento para utilizar uno u otro servicio. Así indica la *figura 36*.

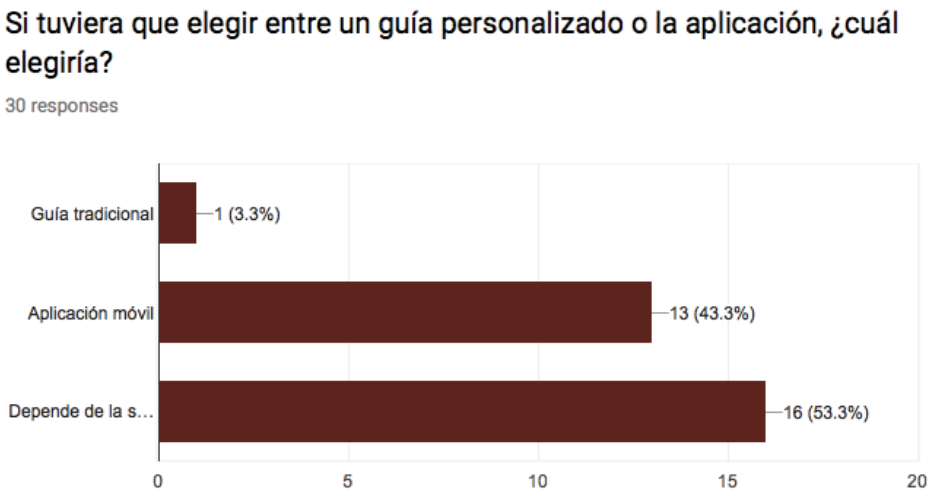

Figura 36. Diagrama de barras acerca de la preferencia de los usuarios.

Con respecto al precio, la mitad de los encuestados preferirían que la aplicación sea gratuita y contenga publicidad. Un 40% pagaría entre $1 y $5, el cual es un rango regular ya que la mayoría de las aplicaciones pagadas en el mercado cuestan eso. Esto se observa en la *figura 37.*

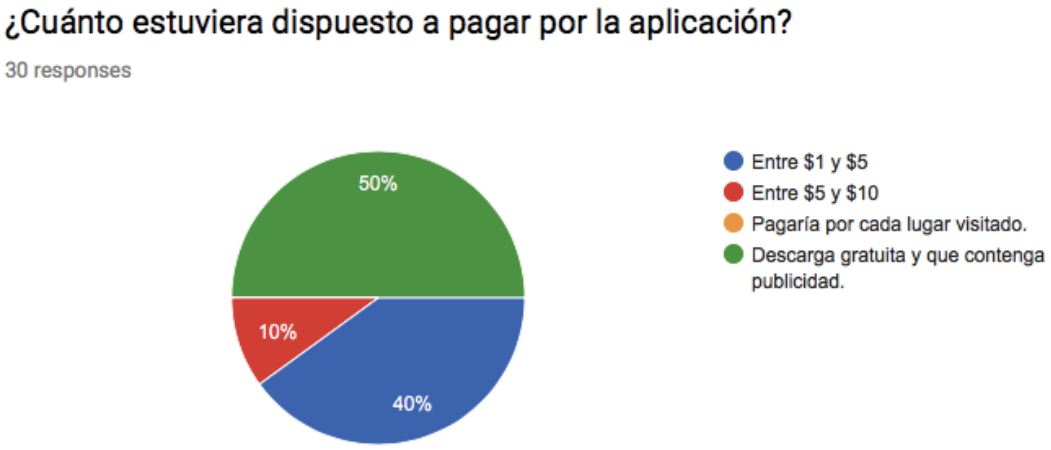

Figura 37. Diagrama de pastel acerca del precio de la aplicación.

La última pregunta pedía recomendaciones a los usuarios para agregar aspectos a la aplicación. El 100% coincidió en que debe tener múltiples idiomas, lo cual es una proyección a futuro. La mitad de concordaron que les gustaría tener algún parámetro con geolocalización y la inclusión de videos en los recursos. Lo recomendado fue analizado y forma parte de las proyecciones de la aplicación.

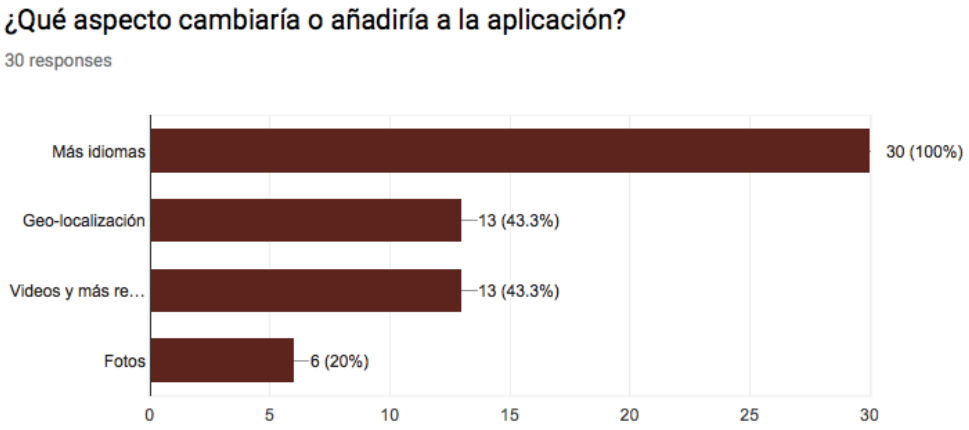

Figura 38. Diagrama de barras acerca de posibles cambios en la aplicación.

6.2. Análisis económico

El análisis económico consiste en establecer un presupuesto estimado del costo de desarrollo de la aplicación. Factores como las horas invertidas y herramientas utilizadas, son tomados en cuenta para establecer el presupuesto.

Se realizaron tablas para representar dichos datos, mostradas a continuación.

6.2.1. Horas de trabajo

Para estimar las horas invertidas se realizó una tabla que nos permita identificar las horas por cada proceso dentro del desarrollo.

De acuerdo a la tabla, se invirtieron 400 horas de trabajo en todos los procesos para la realización del proyecto.

Tabla 1.
Horas de trabajo.

Trabajo	Cantidad horas
Investigación sobre SW.	25
Programación.	200
Diseño de la ruta y recolección de información.	25
Grabación, edición y mezcla.	75
Redacción.	75
TOTAL	**400**

6.2.2. Herramientas utilizadas

Se realizó otra tabla enumerando las herramientas utilizadas y su costo tomando en cuenta una devaluación de su valor original en base al tiempo desde su adquisición.

El total del costo de las herramientas fue de $1600.

Tabla 2.
Herramientas utilizadas.

Herramienta	Costo
Micrófono	$200
Computador	$600
Interfaz	$400
Pro Tools	$400
TOTAL	**$1600**

6.2.3. Servicios varios

Por último, se elaboró una tabla con servicios requeridos para el completo desarrollo de la aplicación.

Tabla 3.
Servicios varios.

Gasto	Costo
Programador	$350
Grabación, edición y mezcla (por 6 audios)	$600
Internet y electricidad	$150
Transporte	$40
Museos	$20
TOTAL	**$1160**

De acuerdo a la tabla, se invirtió un total de $1160 dólares en servicios.

Sumando los totales de las tres tablas se obtiene total de inversión de $2760. A pesar de ser un valor sobre los 2000 dólares, la aplicación tiene tanto potencial y proyecciones a futuro que la inversión puede ser considerada baja. Si se desea ampliar las funciones agregar características a la misma, se estima que el valor podría llegar a los $5000.

7. PROYECCIONES

El potencial de expansión y crecimiento de la aplicación desarrollada es muy amplio. La proyección a corto plazo sería la implementación de este sistema para algunos lugares de la ciudad de Quito, empezando, obviamente, por el CH. Para ello, se necesitaría un permiso para la instalación de los cQR en diferentes lugares y una campaña de promoción para que los visitantes se enteren de su funcionamiento y sus ventajas. Una vez que el sistema esté funcionando en Quito, las demás ciudades optarían por implementarlo también.

Sería ideal encontrar un aliado con influencias en la ciudad. Éste ayudaría con el tema de permisos de instalación de los cQR, así como con el tema económico. Para una eficiente campaña de promoción y marketing se necesita una inversión grande de dinero. La promoción de la aplicación en medios de comunicación como radio y televisión, es esencial. Un buen prospecto para formar este tipo de alianza es Quito Turismo, que cumple con lo mencionado.

Otra proyección sería el desarrollo de la aplicación para dispositivos Apple. El sistema operativo iOS es el segundo más utilizado a nivel mundial, por lo que con el desarrollo para éste, se abarcaría prácticamente todos los dispositivos móviles de la ciudad. La utilización masiva de la aplicación incentivaría a que las personas conozcan de las principales atracciones de las ciudades, haciéndola una *must have* en los celulares de los ciudadanos turísticamente activos.

La incorporación de más idiomas es necesaria para llegar a más usuarios y expandir el uso de la aplicación. Una ventaja de ésta es que prácticamente cualquier idioma puede ser implementado. De igual manera, se podrían insertar más recursos audiovisuales, como videos e imágenes 3D, siguiendo las tendencias tecnológicas. De esta manera la aplicación sería más interactiva con el usuario. Recomendaciones obtenidas de la encuesta, como el mejoramiento de los audios y el reproductor de música, también pueden ser considerados.

La tecnología está siempre avanzando y las herramientas móviles son cada vez más populares. Con un buen asesoramiento, el crecimiento de la aplicación desarrollada podría, inclusive, llegar a otros países y continentes que, viendo el éxito local, quieran aplicar el sistema en sus ciudades. Como proyección a largo plazo se podría considerar a este proyecto como un estándar de sistemas y aplicaciones móviles con fines turísticos a nivel mundial.

8. CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

En un período de cuatro meses de desarrolló una aplicación móvil para dispositivos Android que permite la lectura de códigos QR para brindar al usuario información turística de distintos lugares del Centro Histórico de Quito. El proyecto es una alternativa de turismo que permite a los visitantes acceder a información visual y auditiva del atractivo en el que se encuentran de manera inmediata. El resultado del mismo fue un éxito, cumpliendo con todos los objetivos propuestos.

Al ser una aplicación únicamente para Android, se utilizó el *software* Android Studio y el lenguaje de programación Java para el desarrollo de la misma. El programa cuenta con herramientas visuales que facilitan el proceso de programación, así como componentes de librería que hacen más intuitiva la escritura del código. El lenguaje Java cuenta con ‘clases’, que contienen ‘objetos’ que realizan diferentes funciones para que el programa se ejecute correctamente.

Es necesario seguir cierto orden al programar para que la aplicación se ejecute correctamente. Siempre hay la posibilidad de borrar y reescribir código, no obstante, para optimizar el tiempo y evitar errores es importante conocer el funcionamiento del lenguaje. Siempre se busca que el código sea lo más compacto posible para ahorrar recursos de procesamiento.

Durante el proceso de escritura del código, es inevitable que aparezcan ciertos inconvenientes. El error más difícil de solucionar fue un *bug* que provocaba que la aplicación se cierre cada vez que leía un código QR que no pertenecía a la misma, uno asociado a un URL por ejemplo. Se realizaron varios intentos para solucionar este problema. La solución consistió en la implementación de una función `if` para condicionar al programa a que solo lea códigos que contengan caracteres entre el 0 y 2, que son los que en la programación van a estar asociados a los recursos mencionados.

En resumen, los códigos QR actúan como punto acceso a la información, ya que cada lugar turístico está asociado a un cQR, que su vez contiene codificado un único caracter que es leído e interpretado por la aplicación, para posteriormente mostrar en la pantalla los recursos (imágenes, texto y audio) de dicho lugar. El diseño de la aplicación permite que sea aplicable a cuantos lugares se desee, pero para probar su funcionamiento, se eligieron tres lugares representativos del CH de la ciudad: Plaza Grande, Iglesia de la Compañía y Plaza San Francisco.

Las aplicaciones no solo están compuestas de código, sino que necesitan recursos que se mostrarán a los usuarios. Los recursos de cada lugar fueron creados en su totalidad por los autores del proyecto. Para esto, fue necesaria una investigación exhaustiva acerca de los lugares escogidos, tanto en internet, como en libros, *flyers* y panfletos. Los textos mostrados en la aplicación son reseñas o resúmenes del lugar y las imágenes capturadas proveen diferentes tomas de los atractivos para un contacto visual con el usuario. Éstas fueron tomadas en los lugares con cámaras semi-profesionales, para posteriormente ser editadas en busca de una mejor calidad.

Los audios, por otro lado, son los recursos más importantes ya que permiten al visitante escuchar pastillas auditivas, de aproximadamente dos minutos de duración, mientras se encuentran en el lugar. La producción de los audios compete directamente a lo estudiado durante la carrera, por lo que se aplicaron técnicas aprendidas para las etapas de preproducción, grabación, mezcla y masterización. Éstas fueron realizadas en su totalidad en un *home studio* con herramientas y equipos semi-profesionales.

Cabe recalcar que todo el proceso de producción fue realizado dos veces, ya que en un primer intento no se obtuvieron los resultados esperados. El segundo intento se realizó con minuciosidad, obteniendo mayor profesionalismo y, sobre todo, una mejor calidad sonora.

La etapa de preproducción consistió en la elaboración del guión y la búsqueda de un locutor con experiencia, factor fundamental para optimizar el tiempo de grabación. Una vez preparado el locutor y configurados los equipos correctamente, se procedió con la etapa de grabación. Se grabaron los guiones de cada una de las seis pastillas auditivas (dos por cada atractivo turístico) en sesiones diferentes para facilitar los procesos de edición, mezcla y máster.

El proceso de edición presentó mayor dificultad y exigió más tiempo en su desarrollo. En esta etapa se eligieron las tomas adecuadas y se hicieron cortes correspondientes para encontrar fluidez en las narraciones. De igual manera, se eliminaron sonidos no deseados como respiraciones y ruido de fondo.

En la etapa de mezcla se buscó que los diálogos grabados suenen de manera equilibrada con las pistas de música elegidas. Para esto se realizó un ajuste de niveles y se aplicaron procesadores de señal de audio como ecualizadores y compresores. Adicionalmente, se automatizó el nivel de la pista de la música para que se ajuste de mejor manera al diálogo.

Por último, en la etapa de masterización se aplicaron: un compresor sutil para acoplar de mejor manera a las pistas de voz y música, un ecualizador para atenuar ciertas frecuencias molestas y un maximizador para alcanzar niveles estándares de audio. Es de suma importancia el dominio de las herramientas del *software* de edición y mezcla, así como del funcionamiento de los procesadores para optimizar el tiempo.

Es importante comparar los productos finales de las seis pistas para encontrar diferencias o irregularidades. Todas ellas deben tener características similares, especialmente en lo referente a niveles. Por ejemplo, la relación entre los niveles de la narración y la música debe ser la misma en todos los audios. La voz del narrador debe sonar igual, por lo que los procesadores deben ser configurados con los mismos parámetros. Para esto, se debe obtener un buen primer producto y guardar los parámetros de cada procesador en forma de *preset,* para aplicar dicho

preset en la siguiente sesión de mezcla o máster. Se puede repetir este procedimiento con cada audio.

Con todos los recursos listos, es importante evaluar la opción de implementar un servidor *web* para su almacenamiento. Actualmente, la mayoría de aplicaciones cuentan con servidores, pero este servicio tiene un costo anual. Para decidir si se emplearía o no, se determinó primero el tamaño del paquete de la aplicación terminada, el cual es de 45 MB. Este valor se consiguió utilizando formatos comprimidos, tanto de imágenes como de audio, sin comprometer la calidad de los mismos. Debido al tamaño, se concluyó que no era necesario la contratación del servidor, por lo que todos los recursos están dentro del paquete del programa. Al ser una aplicación piloto y contar únicamente con tres destinos turísticos, la cantidad de recursos no justificaban la inversión anual en un servidor. No obstante, en el caso que se quiera expandir la aplicación y aumentar destinos, su implementación sería prácticamente obligatoria.

La realización de encuestas es fundamental para evaluar cualquier producto elaborado. La opinión de terceros puede aclarar dudas y brindar críticas constructivas para posibles mejoras. La encuesta realizada proporcionó información valiosa, tanto sobre aspectos generales de los turistas, como de la aplicación en sí.

Los resultados de la aplicación en general fueron muy satisfactorios, con un 100% de personas encuestadas que usarían el servicio, a pesar de requerir ciertas mejoras. La interfaz gráfica por ejemplo, a pesar de ser amigable, resultó ser demasiado simple para algunos. De igual manera, los audios pueden ser mejorados en cuanto a la dicción del locutor y el ambiente de producción de los mismos (mejor estudio de grabación, mejores herramientas y equipos, etc.). Características adicionales como más idiomas, más recursos y la incorporación de geolocalización deben ser consideradas en el caso de que se quiera proyectar la aplicación hacia una versión comercial.

El sistema de lectura de códigos QR resultó ser estupendo. La cantidad de aplicaciones que se les pueden dar a es infinita y son más sencillos de implementar que los sistemas de geolocalización. La velocidad de lectura es muy rápida y, al estar los contenidos almacenados dentro de la aplicación, la obtención de información resulta casi inmediata. El único inconveniente que presentaron los códigos fue al momento de la lectura en el CH; en ocasiones la luminosidad del ambiente hacía que los cQR produzcan un reflejo tipo espejo, impidiendo que el lector de la aplicación trabaje con normalidad. Este inconveniente se podría solucionar cubriendo a los códigos QR con plásticos o vidrios anti-reflejos.

8.2. Recomendaciones

Planificación del proyecto

Se recomienda, antes de realizar cualquier tipo de proyecto, planificar con minuciosidad y establecer un cronograma que incluya cada parte del proceso. Cada una de estas partes debe estar bien determinada y diferenciada de las demás para ahorrar tiempo y optimizar recursos. Herramientas gráficas, como el diagrama de Gantt, ayudan a establecer una relación entre los procesos y el tiempo de duración de cada uno. Esto contribuyó en el desarrollo de este trabajo, que empezó un poco desordenado pero fue tomando forma poco a poco.

Etapa de programación

Se recomienda siempre escribir código por secciones o por actividades e ir compilando cada vez que se implemente una nueva operación. La paciencia es muy importante para no cometer errores en este tipo de proyectos. Hay que tener disposición para escribir, probar, borrar y volver a escribir, ya que las opciones al programar son prácticamente infinitas. El orden también es fundamental, ya que la cantidad de códigos puede resultar ser confusa, especialmente si se quiere realizar algún cambio.

Entrando en detalle, es muy importante entender los conceptos y componentes de los lenguajes de programación que se podrían utilizar. En este caso, fue de suma importancia hacer un estudio exhaustivo del lenguaje Java antes empezar la escritura de código. Esto influyó directamente en la eficiencia de todo el proceso de programación.

Se recomienda también usar la menor cantidad de líneas de código para llegar a un objetivo. Normalmente hay múltiples caminos para lograrlo, no obstante, se debe tratar de aplicar el más rápido. Mientras más corto sea el código, más rápido será su ejecución.

Etapa de producción de audios

La mejor recomendación para esta etapa es aplicar cuidadosamente cada herramienta o proceso aprendido. En la etapa de grabación se recomienda conocer los equipos que se van a utilizar prefiriendo siempre los de mejor calidad. De igual manera, un locutor con experiencia puede ahorrar mucho tiempo ya que necesita menos tomas para obtener una buena grabación y facilita el proceso de edición.

Para la edición se recomienda mucha paciencia y minuciosidad. Cada detalle es importante y una etapa de edición mal ejecutada casi siempre resulta en problemas al momento de la mezcla. Los *fades in* y *fades out* deben ser sutiles y los cortes no deben incluir información sonora valiosa.

Para la etapa de mezcla, como se indicó, se recomienda dominar todos los procesadores de señal antes de utilizarlos. También es importante la utilización de un medidor de nivel en el canal máster, para tener información visual de los valores pico y RMS de los diferentes *tracks*. Con una idea clara de los niveles en las pistas de voz y música, se puede tener una referencia para llegar a dichos niveles en las otras sesiones. Siempre es importante la automatización de niveles

en cada *track*, para obtener mejores resultados. Los cambios de dinámica producidos por un compresor no son los mismos que los de una automatización.

En este trabajo, todos los audios fueron grabados con un mismo locutor, por lo que los procesamientos a aplicarse en las pistas deben coincidir para obtener productos de iguales características. Para esto se procedió a terminar el primer audio con buena calidad y a guardar los parámetros de cada *plug-in* de procesamiento en un *preset*. Este *preset* permitirá obtener el mismo resultado en futuras sesiones. Por diferentes motivos, las grabaciones no son siempre idénticas. Es por esto que, al usar *presets,* se debe verificar que el resultado sea el mismo para todas las sesiones.

REFERENCIAS

Alshattnawi, S. (2014). *Building Tour Plan and Navigation System in Art Museum using Dijkstra Algorithm based on Geo-coded QR codes and Cloud Computing. International Journal of Advanced Pervasive and Ubiquitous Computing*, 6(2), 1-13. doi:10.4018/ijapuc.2014040101

Android Developers. (2014). Aspectos fundamentales de la aplicación. Recuperado el 20 de Diciembre de 2017 de https://developer.android.com/guide/components/fundamentals.html

Atala, L. (2008). Código QR Ejemplo de Estructura. *Wikipedia Commons.* Recuperado el 05 de Enero de 2018 de https://es.wikipedia.org/wiki/Archivo:C%C3%B3digo_QR_Ej%C3%A9mplo _de_Estructura.svg

Butler, T. (2006). *A walk of art: The potential of the sound walk as practice in cultural geography. Social & Cultural Geography*, 7(6), 889-908. doi: 10.1080/14649360601055821

Elliston, B. y FitzGerald, E. (2012). *Encouraging Museum Visitor Engagement Using Spontaneous Talk-In-Interaction Audio Guides. Proceedings of the 4th International Conference on Computer Supported Education*. doi: 10.5220/0003889303630372

Herrera, C. (2017). Pasillo Instrumental Ecuatoriano. Florilegio del Pasillo Ecuatoriano. Recuperado el 28 de Diciembre del 2017 de http://pasilloecuador.blogspot.com/p/blog-page_22.html

IDEA. (2011). *What are QR Codes? And how are they useful for outreach?*. IDEA. Recuperado el 13 de Octubre del 2017 de http://www.idea.org/blog/2011/09/05/what-are-qr-codes-and-how-are-they-useful-for-outreach/

Jean, Y, Young, J, Lee, P. (2013). *Soundwalk approach to identify urban soundscapes individually. The Journal of the Acoustical Society of America*, 134, 803.

Liu, J., Kang, J., Behm, H., y Luo, T. (2014). *Effects of landscape on soundscape perception: Soundwalks in city parks.* Landscape and Urban Planning, 123, 30-40. doi:10.1016/j.landurbplan.2013.12.003

Lyons, K., Gandy, M. y Starner, T. (2013). *Guided by voices: An audio augmented reality system*. HCI International 2013 - Posters' Extended Abstracts'

McCartney, A. (2014). *Soundwalking: creating moving environmental sound narratives. T*he Oxford Handbook of Mobile Music Studies, Volume 2, 212-237.

Mohseinia, P. (2016). *Designs principles for mobiles soundwalk applications*. Master of Arts in New Media Design & Production in New Media.

Nilsson, M. E., Jeon, J. Y., Rådsten-Ekman, M., Axelsson, Ö, Hong, J. Y., & Jang, H. S. (2012). *A soundwalk study on the relationship between soundscape and overall quality of urban outdoor places.* The Journal of the Acoustical Society of America, 131(4), 3474-3474. doi:10.1121/1.4709105

Olupot, N. (2016). *Guide*: *How To Create Your First Android App With Android Studio*. PC Tech Magzine. Recuperado el 30 de Diciembre de 2017 de http://pctechmag.com/2016/05/guide-how-to-create-your-first-android-app-with-android-studio/

ORACLE. (2017). Conozca más sobre la tecnología Java. *Oracle: Java* Recuperado el 27 de Diciembre del 2017, de https://www.java.com

Paquette, D., & Mccartney, A. (2012). S*oundwalking and the Bodily Exploration of Places*. Canadian Journal of Communication, 37(1). doi:10.22230/cjc.2012v37n1a2543

Quito Turismo. (2013). Quito en cifras. *Municipio de Quito*. Recuperado el 03 de Enero del 2018 de https://www.quito-turismo.gob.ec/phocadownload/EstadisticasUIO/Quitoencifras/quito%20en%20cifras%202.pdf

Rizopoulos, C., Lambrinos, L., y Gazi, A. (2014). *Design, Implementation, and Evaluation of a Location-Based System for Investigating the Parameters of Place Meaning for Visually Impaired Users.* HCI International 2014 - Posters' Extended Abstracts Communications in Computer and Information Science, 253-258. doi:10.1007/978-3-319-07854-0_45

Rodriguez, T. (2015). 20 años de Java: ¿En qué quedó el sueño de programar una vez, ejecutar en cualquier lugar?. *Xakata*. Recuperado el 27 de Diciembre del 2017 de https://www.xataka.com/aplicaciones/20-anos-de-java-celebramos-su-tremenda-influencia-en-el-mundo-del-software-y-la-programacion

Sebastianspeed. (2015). Historia de Java. *Sebastianspeed Blog*. Recuperado el 30 de Diciembre del 2017 de https://speedsebastian.wordpress.com/author/sebastianspeed/

Soundwalkrs. (2015). *About Soundwalkrs.* Recuperado el 03 de Enero del 2018 de http://soundwalkrs.com/about

Tah, J. (2008). MECANISMOS BÁSICOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS. *POO Sitio Web*. Recuperado el 30 de Diciembre del 2017 de http://www.paginasprodigy.com/lupita2007/web3_final/1.1.-MECANISMOSBSICOSDELAPROGRAMACINORIE.html

Walsh, A. (2010). QR Codes – using mobile phones to deliver library instruction and help at the point of need. Journal Of Information Literacy, 4(1), 55-65. doi:10.11645/4.1.1458

ANEXOS

ANEXO 1

PREGUNTAS DE LA ENCUESTA REALIZADA

Se puede acceder al documento de Google Forms con el siguiente enlace: https://goo.gl/forms/wxChbvEmbnQddZq03

Preguntas generales

1. ¿Ha contratado alguna vez algún servicio de guía turístico en Quito?
2. ¿Qué tal le pareció estos servicios?
3. ¿Ha utilizado alguna aplicación móvil con fines turísticos?
4. En el caso que sí, ¿cuál aplicación?
5. Usualmente, ¿de dónde adquiere información sobre las atracciones turísticas de su interés?
6. ¿Ha escuchado alguna vez sobre los soundwalks o paseos sonoros?
7. ¿Ha utilizado alguna aplicación móvil para la realización de Soundwalks?

Preguntas de la aplicación

8. ¿Qué le parece la aplicación y la idea en general?
9. ¿Qué le parece la interfaz gráfica de la aplicación?
10. ¿Qué le parecen los audios de la aplicación?
11. ¿Utilizaría este servicio?
12. Si tuviera que elegir entre un guía personalizado o la aplicación, ¿cuál elegiría?
13. ¿Cuánto estuviera dispuesto a pagar por la aplicación?
14. ¿Qué aspecto cambiaría o añadiría a la aplicación?

ANEXO 2

FOTOGRAFÍAS TOMADAS PARA LA APLICACIÓN

Figura 39. Plaza de la Independencia. Tomada con iPhone 6s.

Figura 40. Monumento a la Independencia. Tomada con Canon Powershot.

Figura 41. Gallito de la Catedral. Tomada con Canon Powershot.

Figura 42. Placa conmemorativa a Gabriel García Moreno. Tomada con Canon Powershot.

Figura 43. Vista lateral de la Iglesia de la Compañía. Tomada con Canon Powershot.

Figura 44. Vista frontal de la Iglesia de la Compañía. Tomada con Canon Powershot.

Figura 45. La Compañía en la Fiesta de las Luces 2016. Tomada con iPhone 6s.

Figura 46. Exterior de la Plaza San Francisco. Tomada con Canon Powershot.

Figura 48. Vista frontal panorámica de la Plaza San Francisco. Tomada con iPhone 6s.

Figura 47. Enfoque inferior de la Plaza San Francisco. Tomada con Canon Powershot

ANEXO 3

LISTA DE CANCIONES CANCIONES

Ángel de luz – Pasillo instrumental, Parte 2, Carlos Herrera (2017). Benigna Dávalos. 3:33

Desdichas – Benítez y Valencia, Inolvidables de Ecuador (2010). 2:49

Al morir de las tardes – Pasillo instrumental, Parte 1, Carlos Herrera (2017). José Ignacio Canelos. 3:20

Canta cuando me ausente – Pasillo instrumental, Parte 2, Carlos Herrera (2017). Hugo Noriega. 3:39

Anhelos – Pasillo instrumental, Parte 2, Carlos Herrera (2017). Francisco Paredes Herrera.3:40

Besar De Un Pétalo – Pasillo instrumental, Parte 1, Carlos Herrera (2017). Marco Tulio Hidrobo. 2:49