Reemplazo de aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger. Caso de estudio: tienda virtual de artículos deportivos

Andrade Burgos, Jonathan Paul y Benavides Moreano, José Luis

Departamento de Ciencias de la Computación

Ingeniería de Sistemas e Informática

Trabajo de titulación, previo a la obtención del
título de Ingeniero en Sistemas e Informática

Ing. Coral Coral, Henry Ramiro

25 de febrero del 2020

URKUND

Urkund Analysis Result

Analysed Document:	Tesis-Andrade-Benavides.pdf (D62565225)
Submitted:	1/17/2020 5:06:00 PM
Submitted By:	${Xml.Encode(Model.Document.Submitter.Email)}
Significance:	5 %

Sources included in the report:

https://techcrunch.com/2016/05/31/nearly-1-in-4-people-abandon-mobile-apps-after-only-one-use).
https://developer.android.com/studio/intro/?hl=es-419
https://academiaandroid.com/ide-entornos-integrados-de-desarrollo-para-android/
https://marbellamiami.com/es/reviews/4312-best-ides-integrated-development-environment-for-mac-in-2019.html
https://tienda-espe-rest.herokuapp.com/subcategoria
https://tienda-espe-rest.herokuapp.com/stock
https://tienda-espe-rest.herokuapp.com/producto
https://tienda-espe-rest.herokuapp.com/producto/nombre/calentador
https://tienda-espe-rest.herokuapp.com/producto/categoria/7
https://www.open-xchange.com/fileadmin/user_upload/Resources_Pages/Mobile_Developers_Guide/Mobile_DevGuide_17thEdition_Web_Spanish.pdf

Instances where selected sources appear:

32

Firma

.....

Ing. Coral Coral, Henry Ramiro

DIRECTOR

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Reemplazo de aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger. Caso de estudio: tienda virtual de artículos deportivos**" fue realizado por los señores **Andrade Burgos, Jonathan Paul y Benavides Moreano, José Luis** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 22 de enero del 2020

Firma:

.....

Ing. Coral Coral, Henry Ramiro

C. C. 1714864830

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

RESPONSABILIDAD DE AUTORÍA

Nosotros, **Andrade Burgos, Jonathan Paul y Benavides Moreano, José Luis**, con cedulas de ciudadanía n° 1717782187 y 1724756570, declaramos que el contenido, ideas y criterios del trabajo de titulación**: "Reemplazo de aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger. Caso de estudio: tienda virtual de artículos deportivos"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 22 de enero del 2020

Firma

.....

Andrade Burgos, Jonathan Paul

C.C.: 1717782187

Firma

.....

Benavides Moreano, José Luis

C.C.: 1724756570

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

AUTORIZACIÓN DE PUBLICACIÓN

Nosotros **Andrade Burgos, Jonathan Paul y Benavides Moreano, José Luis,** con cédulas de ciudadanía n° 1717782187 y 1724756570, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Reemplazo de aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger. Caso de estudio: tienda virtual de artículos deportivos**" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 22 de enero del 2020

Firma
.....

Andrade Burgos, Jonathan Paul

C.C.: 1717782187

Firma
.....

Benavides Moreano, José Luis

C.C.: 1724756570

Dedicatoria

A mis padres José y Patricia por todo el amor, la sabiduría y el apoyo que me brindan en cada una de las decisiones que tomo. A mis hermanas, Patricia, Carolina, Paulina y Jennifer quienes siempre han estado presentes para prestarme su hombro, brindarme un consejo y ayudarme a superar cualquier obstáculo que pueda presentarse. A mis sobrinas Mikaela y Danna y a mi sobrino Benjamín porque con tan solo un abrazo o una sonrisa son capaces de alegrar cualquiera de mis días. A mi abuelita Pepita, quién con su carisma ha sido un pilar fundamental en la culminación de este objetivo. A mis tías que han sabido guiarme y acogerme como si fuera uno más de sus hijos. A mi bisabuelita Romelita que fue la más preocupada y feliz de que logrará cumplir este sueño.

José Luis Benavides

A mi Abuelita, por estar a mi lado siempre; por su paciencia y comprensión; por su sabiduría y sencillez; porque sin ella nada de esto sería posible.

A mi madre por su constancia, su apoyo incondicional; su ayuda y fortaleza.

A mi hermano, quien me ha enseñado que un niño muchas veces puede ser más sabio que un adulto.

A mi novia por su apoyo y amor; por ayudarme a crecer como profesional y como persona.

Jonathan Andrade

Agradecimiento

Agradezco a Dios por haberme entregado tantas bendiciones y poner en mi camino a todas aquellas personas que me permiten día a día alcanzar mis objetivos. A mis padres, a mis hermanas y a toda mi familia que siempre han estado ahí para guiarme, sin importar el lugar ni las circunstancias.

A mis profesores Ramiro, Diego y Rodrigo quienes han aportado inmensamente a mi formación tanto personal como profesional; a la ing. Tatiana Gualotuña, quién con sus consejos y conocimientos me ha brindado un apoyo incondicional durante mi carrera universitaria.

Al ingeniero Henry Coral, director de tesis, por apoyarme en la elaboración del proyecto, por los conocimientos y consejos que me ha brindado y por llegar a convertirse en un gran amigo al que aprecio mucho.

José Luis Benavides

Agradezco a la universidad de las fuerzas armadas ESPE por acogerme y permitirme cursar la prestigiosa carrera la cual me ha dejado una gran cantidad experiencias y recuerdos gratos. Al ingeniero Henry Coral por ser una guía, un maestro y un amigo que ha sabido encaminarnos de la mejor forma hacia el éxito. A mi familia por siempre estar presentes; a todos quienes han estado a mi lado, en esta parte de mi vida y espero sigan siendo cercanos, a mis maestros y compañeros por su paciencia y su conocimiento.

Gracias a todos por estar ahí; y sobre todo gracias por confiar en mí.

Jonathan Andrade

Tabla de contenido

Índice de tablas

Índice de figuras

Resumen

Las aplicaciones móviles facilitan el acceso a productos y servicios ofertados, también permiten a las empresas captar nuevos clientes y nichos de mercado. Es por ello que el desarrollo de aplicaciones móviles transaccionales es cada vez más común y frecuente; sin embargo, actualmente el proceso de desarrollo de dichas aplicaciones resulta demasiado costoso debido a la cantidad de plataformas para los dispositivos existentes. Facebook Messenger proporciona una plataforma de comunicación que permite a las empresas comunicarse con el usuario final a través de su aplicación de mensajería. Además, proporciona características de integración a su aplicación por medio de "Webhooks" (alteración de funcionamiento por medio de peticiones HTTP) que permiten a los desarrolladores y empresas incluir contenido externo y mostrarlo en la plataforma de Messenger. Es por esto que, el presente proyecto se enfoca en analizar la factibilidad de reemplazar las aplicaciones móviles de tipo transaccional por medio de aplicaciones integradas a Facebook Messenger. Se desarrollaron 2 prototipos de aplicaciones, la primera una aplicación móvil nativa de tipo transaccional y la segunda una aplicación integrada en Facebook Messenger orientadas a una tienda virtual de artículos deportivos. Para la obtención de resultados se entrevistó a un grupo de 20 usuarios, también se evaluó el tiempo y costo de desarrollo de ambas aplicaciones; que la aplicación integrada cumpla las mismas funcionalidades de la aplicación móvil nativa y por último la aceptación de la alternativa propuesta.

Palabras Clave:

- **ANÁLISIS COMPARATIVO**
- **APLICACIÓN MÓVIL**
- **CHAT BOT**
- **MESSENGER API**
- **FACEBOOK MESSENGER**

Abstract

Mobile applications make access to products and services offered easily, also allow companies to capture new customers and markets. For this reason, the development of transactional mobile applications is more common and frequent. However, currently the process of developing such applications is too expensive due to the number of platforms for existing devices. Facebook Messenger provides a communication platform that offer companies to communicate with the end user through their messaging application. In addition, it provides integration features to application using Webhooks (alteration of operation through HTTP requests) that allow developers and companies to include external content and display it on the Messenger platform. This project focuses on analyzing the feasibility of replacing mobile transactional applications through applications integrated to Facebook Messenger. Two prototypes of applications were developed, the first one a mobile transactional native application and the second an application integrated in Facebook Messenger aimed at a virtual sporting store. A group of 20 users was interviewed to get results and the time and cost of development of both applications were evaluated; that the integrated application complies with the same functionalities of the native mobile application and finally the acceptance of the proposed alternative.

Keywords:

- **COMPARATIVE ANALYSIS**
- **MOBILE APP**
- **CHAT BOT**
- **MESSENGER API**
- **MESSENGER FACEBOOK**

Capítulo I - Introducción

Antecedentes

Los teléfonos móviles han dejado de ser dispositivos de comunicación de un solo propósito para ser herramientas dinámicas que apoyan al usuario en una infinidad de tareas; todo esto debido a las aplicaciones móviles que ofrecen una gran variedad de herramientas para ayudar al usuario en la vida cotidiana (Böhmer, Hecht, Schöning, Krüger, & Bauer, 2011). Las empresas han visto a las aplicaciones móviles como una forma de acercarse y captar nuevos clientes, por lo que el desarrollo de aplicaciones que permite a los usuarios realizar transacciones con la empresa es cada vez más frecuente, llegando a existir cerca de 700.000 aplicaciones disponibles en el año 2013 (Fu, Lin, Li, Faloutsos, Hong, & Sadeh, 2013) y alrededor de 3.5 millones de aplicaciones en el 2018 en Google Play Store (Wang, Li, Li, Guo, & Xu, 2018).

Wei & otros indican que más de 50 millones de aplicaciones son descargadas diariamente; no obstante, el 95% de estas aplicaciones dejan de ser utilizadas por el usuario (Wei, Lee, Lu, Tzou, & Weng, 2015). Este estudio se enfoca en cómo el desarrollador debe buscar medidas para retener al usuario luego de la descarga, basándose en el costo percibido por el usuario ya sea este monetario o no. Un estudio realizado en 2012 encontró que un Smartphone posee alrededor de 22 aplicaciones instaladas (Shin, Hong, & Dey, 2012); en este contexto, el presente trabajo propone la investigación de factibilidad para implementar una alternativa embebida en una de las aplicaciones más populares en cualquier Smartphone como Facebook Messenger.

El sistema operativo de los dispositivos móviles proporciona una serie de canales de comunicación abierta entre aplicaciones. Esto promueve la colaboración entre aplicaciones y

reduce tiempo y costo de desarrollo al facilitar la reutilización de componentes (Han & Li, 2017). Según Wasserman, los dispositivos móviles pueden tener numerosas aplicaciones de diversas fuentes capaces de interactuar entre sí.

Planteamiento del problema

Según O'Connell, el estudio realizado para la empresa Localytics determinó que el 24% de los usuarios de aplicaciones móviles abandonan la aplicación después de un solo uso. También, uno de los factores más alarmantes es que el 63% de los usuarios utilizan una aplicación menos de 10 veces (O'Connell, 2017). Otro estudio realizado por la misma empresa sugiere que el tiempo crítico para determinar la fidelidad de un usuario se basa entre 30 y 90 días de uso. Actualmente las aplicaciones de Multimedia y Entretenimiento retienen a la mayoría de los usuarios con un 24% después de 90 días (Perro, 2018).

En el primer semestre del 2017, el número de aplicaciones liberadas mensualmente alcanzó su punto máximo con un promedio de 15.89 aplicaciones, en el primer semestre del 2018, el lanzamiento de aplicaciones disminuyó en 13.78% (Davenport, 2018). Lo que sugiere la inexistencia de un problema en la fase de desarrollo y despliegue de las aplicaciones, haciéndose notable el verdadero problema para las empresas surge al momento de retener a los usuarios.

Otro factor por considerar, es que las aplicaciones suelen presentar problemas de compatibilidad con el hardware y software de ciertos dispositivos, además de los problemas comunes del software como son la seguridad, el rendimiento, la confiabilidad y sobre todo la cantidad de almacenamiento utilizado (Wasserman, 2010). Es por estas razones que el usuario opta por no instalar aplicaciones que ve innecesarias o que no cumplen sus expectativas,

haciendo que muchas de las aplicaciones móviles desarrolladas por las empresas queden en desuso.

Con el fin de solventar el problema de compatibilidad del software se realizó dos prototipos: el primero, una aplicación móvil nativa de tipo transaccional, y el segundo, un prototipo de aplicación móvil transaccional integrada en Facebook Messenger. Una vez desarrollados los prototipos se realizó una comparación de las aplicaciones a nivel de uso de recursos, funcionalidad y accesibilidad a la aplicación.

Justificación

Una investigación realizada (Perez, 2016) establece que los usuarios pasan el 85% de su tiempo utilizando teléfonos inteligentes, pero la mayoría utiliza tan solo 5 aplicaciones no nativas, es decir poseen versiones en las distintas plataformas móviles. Además, las apps varían de persona en persona, pero se destacan las redes sociales, los videojuegos y las aplicaciones de mensajería instantánea.

Un factor clave de las aplicaciones móviles, es que permiten la interacción con otras aplicaciones, a través de solicitudes o simplemente compartiendo servicios para permitir el procesamiento de datos, la visualización de contenido multimedia, etc. ya sea mediante aplicaciones de fábrica, que vienen instaladas en el dispositivo, mediante el navegador web o con numerosas aplicaciones de fuentes derivadas (Wasserman, 2010). En este contexto se puede observar que la construcción de una aplicación integrada a otra es totalmente factible.

Por este motivo, se realizó dos prototipos: el primero, una aplicación móvil transaccional y el segundo una aplicación integrada en Facebook Messenger con la finalidad de comparar el

consumo de recursos, la usabilidad, el tiempo de desarrollo y verificar la factibilidad de reemplazar una aplicación nativa a través de una aplicación integrada.

Objetivos

Objetivo General

Estudiar la factibilidad de reemplazar efectivamente aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger a través de un estudio comparativo, a nivel de recursos, tiempos de desarrollo y usabilidad.

Objetivos Específicos

Desarrollar un prototipo funcional de una aplicación móvil nativa transaccional en Android, que implemente las principales funcionalidades de una tienda virtual.

Desarrollar un prototipo aplicación integrada en Facebook Messenger utilizando servicios web tipo REST y la “API” de Facebook Messenger.

Evaluar las características de la aplicación integrada con el fin de contrastar con la aplicación móvil nativa transaccional y evaluar el nivel de usabilidad, consumo de recursos y tiempo de desarrollo.

Alcance

La investigación comprende el desarrollo de una aplicación móvil transaccional que será desarrollada para dispositivos Android, así como el prototipo de la aplicación integrada a Facebook Messenger. Con el fin de verificar que el reemplazo es posible, se realizó una comparación, evaluando el nivel de usabilidad, consumo de recursos, tiempo de desarrollo y aceptación por parte del usuario.

Para el desarrollo de la investigación se ha propuesto realizar una aplicación de compra de artículos deportivos online en la cual se puede observar los productos disponibles, el costo y una pequeña descripción del artículo; también se puede añadir o quitar artículos al carrito de compras, gestionar la factura y las compras realizadas anteriormente.

Capítulo II - Marco Teórico

Aplicaciones Móviles

Las aplicaciones móviles son aplicaciones de software que se diferencian de las tradicionales debido a que presentan características específicas relacionadas con su forma de acceso y su portabilidad (IBM Software, 2018). Existen diferentes tipos de aplicaciones móviles:

Aplicaciones Web

En la actualidad los dispositivos móviles admiten navegadores web que admiten las capacidades de HTML5, CSS3 y JavaScript. Aprovechando las propiedades de HTML5 los desarrolladores pueden utilizar componentes de usuario, acceso a medios, servicios de geolocalización y funcionalidades offline. En la actualidad es común encontrar sitios web, que se han optimizado para redimensionar sus ventanas y optimizar sus servicios para dispositivos móviles, muchos de ellos incluyen complementos que permiten a la página web soportar la característica del táctil de los dispositivos. Incluso varias empresas incluyen complementos que permiten al usuario instalar una especie de acceso directo a su plataforma que aparenta la experiencia de una aplicación nativa (IBM Software, 2018).

Aplicaciones Web Progresivas

Denominadas PWApps por su nombre en inglés (Progressive Web Apps), son aplicaciones destinadas a mejorar la experiencia de un usuario web desde un dispositivo móvil (Malavolta, Procaccianti, Noorland, & Vukmirovic, 2017). Entre las características más importantes de este tipo de aplicaciones tenemos: la capacidad de utilizarse sin conexión, mostrar notificaciones “push”, y parecerse a una aplicación nativa, utilizando un trabajador de servicio (service worker) que permite a los desarrolladores utilizar un script que se ejecuta en

segundo plano que actúa como proxy de red y dispositivo (Malavolta, Procaccianti, Noorland, & Vukmirovic, 2017).

Las PWApps son aplicaciones web, que se acceden a través de una URL única utilizando un navegador instalado en el dispositivo móvil. Al ser una aplicación web brinda la misma experiencia a usuarios de todo tipo de plataformas. Además, se puede acceder a servicios de geolocalización, micrófono, cámara gracias a diferentes "APIs" desarrolladas para HTML5 (Biørn-Hansen, Majchrzak2, & Grønli1, 2017).

Aplicaciones Móviles Nativas

Las aplicaciones nativas son aplicaciones que ejecutan archivos binarios, descargados directamente en el dispositivo y almacenadas localmente. Generalmente se descargan directamente desde una tienda de aplicaciones ofrecidas por los proveedores como Play Store o el App Store de IOS. Este tipo de apps interactúan directamente con el sistema operativo del dispositivo sin intermediarios, además es capaz de interactuar con las "APIs" disponibles por defecto para el sistema operativo (IBM Software, 2018)

Para el desarrollo de una aplicación móvil nativa, los desarrolladores deben escribir el código fuente en un lenguaje específico para cada sistema operativo, además crear sus propios recursos adicionales en formatos admitidos por el sistema operativo seleccionado (IBM Software, 2018).

Existen diversas empresas que ofrecen sistemas operativos, lenguajes, herramientas y formatos en los que se desarrolla una aplicación móvil. En la tabla 1 podemos ver los principales sistemas operativos y características de las aplicaciones móviles.

Tabla 1
Resumen para desarrollo de Aplicaciones Nativas

	Apple IOS	**Android**	**Windows Phone**
Lenguaje de Programación	Objective C, C, C++	Java (C, C++)	C#, .Net
Herramientas	XCode	Android SDK	Visual Studio, Windows Phone development tools
Formato de paquete	.app	.apk	.xap
Tienda de Aplicaciones	Apple App Store	Google Play Store	Windows Phone Marketplace

Nota: Información obtenida de IBM (IBM Software, 2018)

i. Aplicaciones IOS

- **Historia**

En 1998, Apple empezó a trabajar en el proyecto Newton para desarrollar la línea de MessagePad que sería el inicio de la primera versión de un sistema operativo llamado NewtonOS; en el cual se basaría el sistema operativo de IPhone (IOS) reutilizando la forma de almacenar calendario, contactos, mensajes y tareas.

El lanzamiento del primer teléfono de Apple, el IPhone 2G en junio del 2008; dio paso a que Google, Motorola, HTC, BlackBerry y demás empresas fabricantes de teléfonos celulares saquen al mercado sus propias versiones de teléfonos inteligentes; cambiando así el concepto de teléfono celular existente.

En la versión inicial, Apple declaró que IPhone usaba una versión del sistema operativo de escritorio, OS X; que más tarde tomaría el nombre de IOS (Mullan, Riess, & Freiling, 2019).

- **Versiones**

IOS 1: fue liberado junto con el IPhone 2G en junio del 2007; este sistema operativo posee una interfaz gráfica que muestra, en la parte superior, los iconos de batería, red, Bluetooth y hora. Debajo de la parte superior se encuentran las pantallas de inicio; donde se localizan las aplicaciones que se podían descargar desde la tienda de Apple, o añadir como marcador desde Safari. Finalmente, en la parte inferior el "dock" que posee cuatro iconos que a diferencia de las pantallas de inicio no cambian al deslizar. Existieron varias mejoras a este sistema operativo, que dieron paso a nuevas versiones; llegando así hasta IOS 1.1.3 en enero del 2009 (Morrissey & Campbell, 2011).

IOS 2: Liberado con el IPhone 3G en junio del 2008, el sistema operativo ofrecía la opción de posicionamiento global (GPS) además se incluyó la "app store"; que ofrece aplicaciones para descargar y ejecutar en el IPhone. Apple da facilidades a los desarrolladores para crear sus propias aplicaciones para IPhone con la liberación del SDK de Apple en marzo del 2009; de esta manera, los desarrolladores pudieron utilizar el "API" del GPS para diferentes propósitos. Al igual que el IOS 1 esta versión también tuvo varias mejoras llegando hasta la versión 2.2.1 (Morrissey & Campbell, 2011).

IOS 3: Aparece en junio del 2009 y posee características que no poseen las versiones de IOS 1 y 2; como cortar, copiar y pegar; historial de llamadas con duración, inicio de sesión en YouTube desde el teléfono, copias de seguridad cifradas, control de voz. Además, para iPhone 3Gs se ofrece la opción de capturar y recortar vídeo, enfoque

automático de la cámara y la posibilidad de cifrado de hardware (Morrissey & Campbell, 2011).

IOS 4: El 21 de junio del 2010, Apple anuncia la versión 4 de IOS; la empresa daba un gran paso con esta versión ya que, permitía al dispositivo realizar múltiples tareas al mismo tiempo; además que agregó 1500 “APIs” para los desarrolladores, algunas de estas, permitían a la aplicación ser ejecutada en segundo plano (Morrissey & Campbell, 2011).

IOS 5: Liberado en octubre del 2011 junto con el IPhone 4s; esta versión del sistema operativo incluyó control de voz basado en lenguaje natural, o mejor conocido como Siri el asistente virtual de Apple. También tenía ciertas características como la integración con Twitter; pero la más importante fue la adición de ICloud que permite guardar toda la información del usuario en la nube (Bommisetty, Tamma, & Mahalik, 2014).

IOS 6: Liberado en junio del 2012 junto con el iPhone 5; en esta versión se eliminaron aplicaciones instaladas por defecto como YouTube y Google Maps y se añadieron el Apple Maps, Facetime, Passbook; así como mayores controles de privacidad (Bommisetty, Tamma, & Mahalik, 2014).

IOS 7: Liberado en septiembre del 2013 junto con el IPhone 5s, mejoró el sistema; cambiando la interfaz estática por una más dinámica. Se añadieron nuevas características como las actualizaciones automáticas de las aplicaciones, el bloqueo de activación y el sensor de huellas dactilares llamado Touch ID (Bommisetty, Tamma, & Mahalik, 2014).

IOS 8: El 17 de septiembre del 2014 sale al mercado el sistema operativo IOS 8, una de las principales características de este sistema operativo fue el sistema "Apple Pay" para pago seguro, mejoras incorporando ICloud drive, photos y music; además de integrar Family Sharing para que el contenido adquirido por un usuario de Apple pueda ser compartido con todos los miembros de su familia. Finalmente se libera al mercado el Home Kit que permite a los dispositivos Apple conectarse con dispositivos IOT (Tuominen & Virtaranta, 2019).

IOS 9: Liberado el 16 de septiembre del 2015, presenta características nuevas como el modo de ahorro de energía, el modo nocturno, que permite controlar la luz azul emitida por el dispositivo; y se enfoca principalmente en mejorar la velocidad, capacidad de respuesta; estabilidad y mejorar el rendimiento en dispositivos más antiguos; que servirán de base para las próximas versiones de IOS (Tuominen & Virtaranta, 2019).

IOS 10: Fecha de liberación 13 de septiembre del 2016; antes de IOS 10 las opciones para desarrolladores externos eran limitadas; con la liberación de este sistema operativo, nace la interoperabilidad y la personalización; las aplicaciones podían comunicarse entre sí, además que Siri empezó a estar disponible para terceros (Tuominen & Virtaranta, 2019).

IOS 11: Liberado el 19 de septiembre del 2017; una de las principales características añadidas, es la de soportar y facilitar el desarrollo de aplicaciones que usen realidad aumentada. Se enfocó en convertir a los IPad en sustitutos de computadores portátiles, haciendo que IOS funcione como un sistema operativo de escritorio, con nuevas opciones de arrastrar y soltar, aplicaciones de pantalla dividida,

múltiples áreas de trabajo y una aplicación que sirve para explorar los archivos del dispositivo (Tuominen & Virtaranta, 2019).

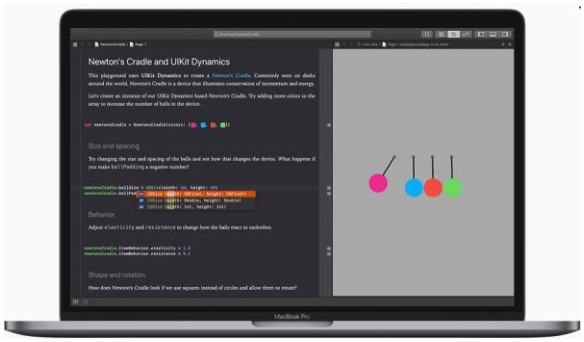
IOS 12: Esta versión de IOS fue liberada el 17 de septiembre del 2018; a diferencia de otras versiones que incluían grandes mejoras, esta se enfocó en refinar aplicaciones de uso común, añadiendo notificaciones "push" agrupadas y accesos directos desde Siri; al igual que permitió brindar, a los padres, la opción de monitorear el uso de los dispositivos mediante Screen Time (Tuominen & Virtaranta, 2019).

IOS 13: La última versión de Apple lanzada el 19 de septiembre de 2019, añade un modo oscuro para ayudar a la vista del usuario. Se aplican nuevas herramientas para la edición de fotos y video, también se realizan mejoras a nivel de seguridad, y en ciertas aplicaciones de Apple como son los mapas, Siri, mensajes; dando la oportunidad de crear una versión en forma de caricatura de personas para ser enviado al estilo de un emoticón. Además, se añade al teclado la función donde se desliza el dedo de una letra a otra para facilitar la escritura mejor conocido como "swipe". (Apple Inc, 2019)

Lenguajes de desarrollo

Originalmente, el lenguaje de desarrollo de las aplicaciones para Apple era Objective C (Goadrich, Rogers, 2011); en la actualidad este lenguaje ha sido reemplazado por Swift 5. Este presenta interoperabilidad con su predecesor, permitiendo añadir funcionalidades a una aplicación, desarrollada con Objective C, y conviviendo en conjunto con ella; esto se debe a que Swift está basado en C y Objective C; por lo que posee similares características de bajo nivel como tipos, control de flujos, operadores (Apple Inc, 2019).

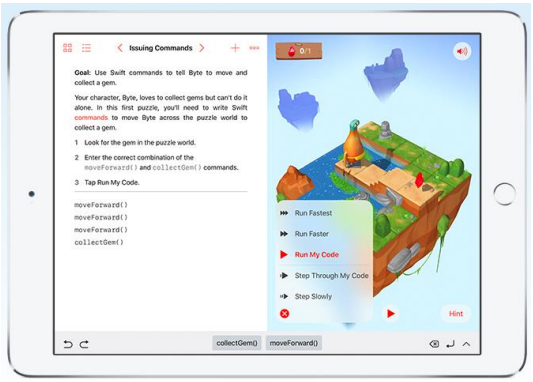
Figura 1
Desarrollo de aplicaciones en Swift



Nota: En el gráfico se observa la pantalla del entorno de desarrollo para aplicaciones IOS "Swift". Tomado de Apple (Apple Inc, 2019)

Desde su creación, este lenguaje fue diseñado para ser rápido, utilizando un compilador de alto rendimiento, transforma el código Swift en lenguaje nativo optimizado, para aprovechar el software de mejor manera. También fue diseñado con una concepción de que cualquier persona puede usarlo, evitando usar punto y coma, y tipos de datos definidos. Además, Apple añadió un plan de estudios gratuito para cualquier usuario mediante descarga de la aplicación Swift Playgrounds (Apple Inc, 2019).

Figura 2
Desarrollo de aplicaciones en Swift Playgrounds

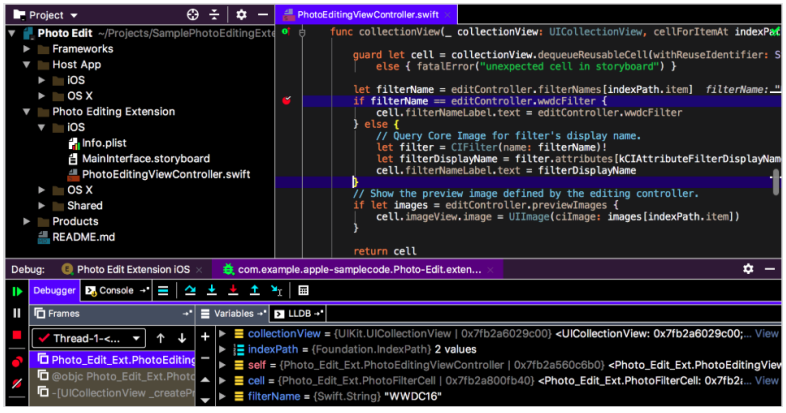


Nota: Se observa la interfaz del entorno. Tomado de Apple (Apple Inc, 2019)

- **Entorno de desarrollo integrado (IDE)**

AppCode: Desarrollado por Jetbrains, permite varios lenguajes de desarrollo como Swift, Objective-C, C y C ++, JavaScript, XML, HTML, CSS y XPath (Marbellamiami, 2019). Posee dos opciones para autocompletado y la opción de reestructurar código.

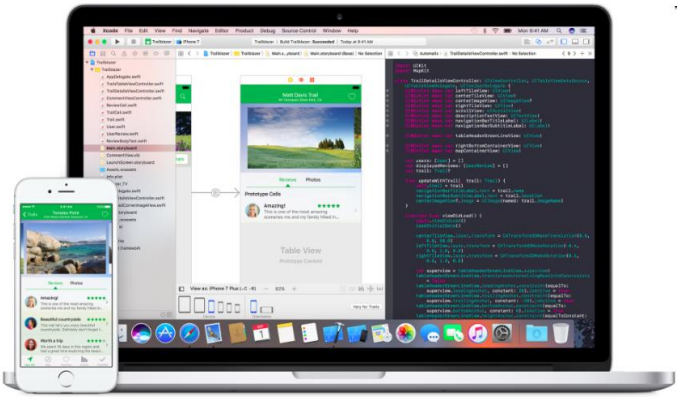
Figura 3
Pantalla de AppCode IDE



Nota: Tomado de Marbellamiami (Marbellamiami, 2019)

XCode: Es el IDE oficial de Apple (figura 4) que se integra fácilmente a Cocoa y Cocoa Touch; presenta un simulador para permitir al usuario probar sus aplicaciones, sin necesidad de un dispositivo físico; provee al usuario la opción de autocompletar, además que el compilador busca errores al momento; y ayuda a mantener los estándares como interlineado y espacios entre métodos. Permite integrarse con el sitio web del desarrollador de Apple, lo que facilita al usuario enviar directamente a la tienda de Apple una vez que la app esté lista (Apple Inc, 2019).

Figura 4
Pantalla de XCode IDE



Nota: Tomado de Apple (Apple Inc, 2019).

- **Requisitos**

Para el desarrollo en IOS, si se desea utilizar AppCode, el IDE alternativo para IOS, los requerimientos mínimos son Xcode 9.4-10.2, 2 GB RAM mínimo, aunque se recomienda 8 GB RAM; 2.5 GB de espacio de almacenamiento, mucho mejor si es un disco SSD; y al menos una resolución de 1024x768 (JetBrains, 2019).

Para instalar Xcode se necesita una computadora que utilice MacOS 10.13.6 o posterior; los requerimientos mínimos para este sistema operativo son: 250 GB 5400 RPM de almacenamiento, un procesador Intel Mobile Core 2 Duo y una tarjeta de video Nvidia GeForce 9400M (Apple Inc, 2019).

- **Distribución**

La tienda de aplicaciones de Apple es accesible a cualquier usuario de un teléfono de la marca; esta tienda ofrece diversidad de aplicaciones por categoría como cursos de cocina, juegos, libros, aplicaciones de negocios; entre otros. Los desarrolladores de dichas aplicaciones pueden ganar dinero de diferentes maneras, por ejemplo: existen aplicaciones gratis que se enfocan en generar ingresos desde otro tipo de negocio como publicidad, servicios físicos o compras dentro de la aplicación; y las pagadas que requieren de una suscripción, un solo pago al momento de la descarga o de pago en la descarga con compras en la aplicación. Todas las aplicaciones son revisadas antes de ser publicadas; al igual que las actualizaciones. Apple se enfoca en tener aplicaciones de calidad y sin material desagradable para el usuario (Apple Inc, 2019).

Actualmente existen 2 métodos oficiales para la distribución de aplicaciones en IOS:

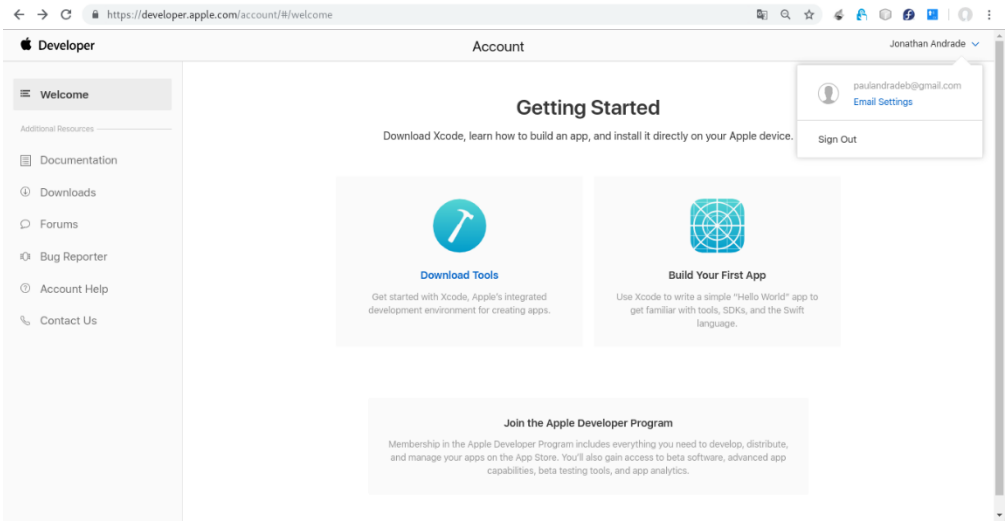
- **Apple Business Manager y Apple School Manager:** son opciones de distribución de aplicaciones para empresas y pueden ser personalizadas exclusivamente para ellas (Apple Inc., 2019).
- **Distribución Ad Hoc:** Permite distribuir la aplicación a un número limitado de usuarios para realizar pruebas (Apple Inc, 2019).

- **Licencias de desarrollo**

Para aprender cómo desarrollar aplicaciones basadas en IOS, se puede utilizar las herramientas de desarrollo, Swift y XCode de manera gratuita; sin embargo, si se desea añadir características más avanzadas y distribuir las aplicaciones desarrolladas el programa para desarrolladores de Apple tiene un costo de 99$ por año; para registrarse en este programa debe poseer un Apple ID con autenticación de dos factores activada, es decir un dispositivo Apple que permite autenticarse (Apple Inc, 2019).

Figura 5

Pantalla para registro como desarrollador Apple



Nota: En la figura 5 podemos ver la pantalla de registro para desarrolladores de Apple. Tomado de Apple (Apple Inc, 2019)

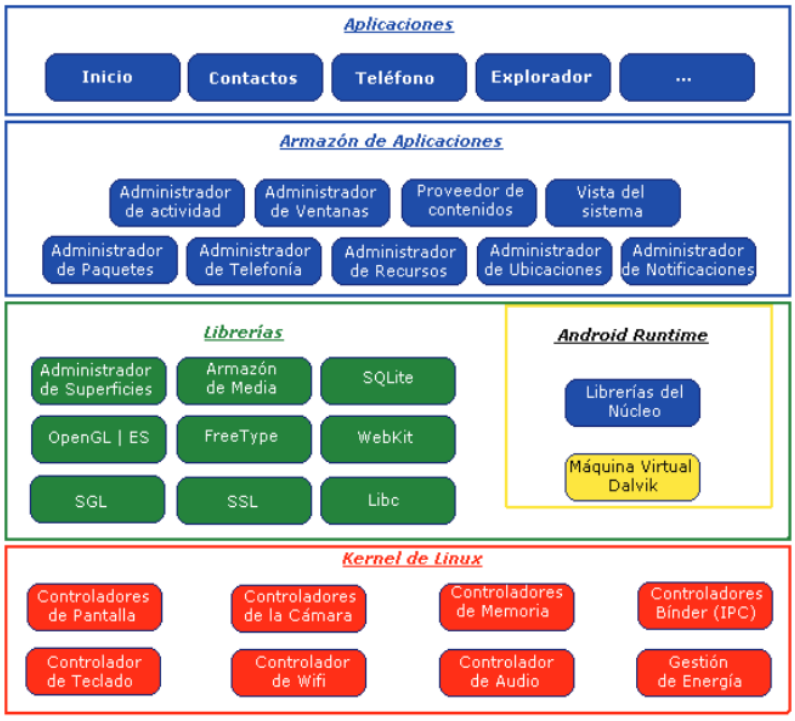
ii. Aplicaciones Android

- **Historia**

Desarrollado originalmente por la empresa Android Inc., fue adquirida por Google en 2005. El proyecto Open Handset Alliance (Alliance, 2011) conformado por cerca de 48 empresas de desarrollo de software, hardware y telecomunicaciones estableció Android como sistema

base para teléfonos inteligentes. Android es un sistema operativo basado en el núcleo de Linux (Báez, y otros, 2012), tiene acceso y gestión a los recursos debido a que se encuentra en una capa superior al núcleo (Kernel) como se observa en la figura 6.

Figura 6
Sistema de capas de Android



Nota: Tomado de Báez (Báez, y otros, 2012)

- **Versiones**

Android 1.0 Apple Pie: La primera versión de Android fue liberada en 2008, poseía una interfaz bastante básica, pero entre las características más destacadas se encontraban las actualizaciones a través del Android Marquet, un navegador web, soporte para la cámara y sincronización con los servicios de Google como Gmail,

YouTube, contactos y la agenda (Raphael, 2019). La pantalla de inicio y el navegador web de Android 1.0 se muestran en la figura 7.

Figura 7
Android 1.0 pantalla de inicio y navegador web

Nota: Tomado de Raphael (Raphael, 2019)

Android 1.1 Banana Bread: La segunda versión salió al mercado en febrero de 2009, entre sus principales características se encuentra un teclado que aparece y desaparece para la aplicación de llamadas. Además, introdujo la capacidad de guardar los adjuntos del servicio de mensajes multimedia “MMS” (Raphael, 2019).

Android 1.5 Cupcake: fue liberada a finales de abril del 2009, esta versión incluyó el primer teclado en pantalla, lo que impulsó la liberación de los nuevos teléfonos sin teclado físico. Además, incluyó el marco para widgets y la opción de grabación de video (Raphael, 2019) como se indica en la figura 8.

Figura 8
Android 1.5 pantalla con widgets

Nota: Tomado de Raphael (Raphael, 2019)

Android 1.6 Donut: liberado al mercado en septiembre de 2009, incluyó la capacidad de variar el tamaño de la pantalla y la resolución de la misma; además agregó soporte para redes CDMA (Raphael, 2019). La figura 9 muestra el navegador web con el tamaño de pantalla por defecto.

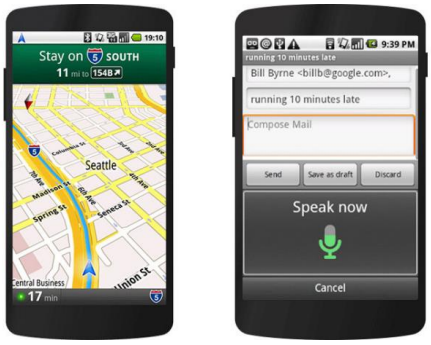
Figura 9
Teléfono con Android 1.6

Nota: Tomado de Raphael (Raphael, 2019)

Android 2.0 a 2.1 Eclair: salió al mercado en octubre de 2009, marcó historia en los teléfonos celulares ya que incluía por primera vez navegación y mapas guiados por voz; así como, la información del tráfico. Además, presentó fondos de pantalla en vivo, zum mediante gestos y la función de transformar voz a texto (Raphael, 2019).

Figura 10

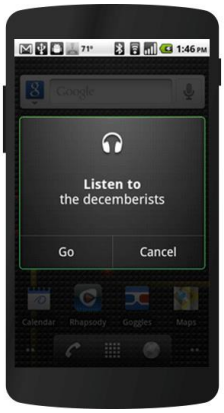
Teléfono con Android Eclair, función de voz a texto.



Nota: Tomado de Raphael (Raphael, 2019)

Android 2.2 Froyo: liberada por Google en mayo de 2010, esta versión se centró en mejorar los procesos internos del sistema operativo. Entre las características destacadas se encuentran la adición de la barra de aplicaciones rápidas conocida como "dock" en la parte inferior de la pantalla; así como, las primeras acciones por voz para obtener instrucciones o guardar notas. Froyo también brindó soporte para flash en el navegador web; lo que permitía visualizar casi cualquier página (Raphael, 2019). La figura 11 muestra la función de acciones por comando de voz de Android 2.2.

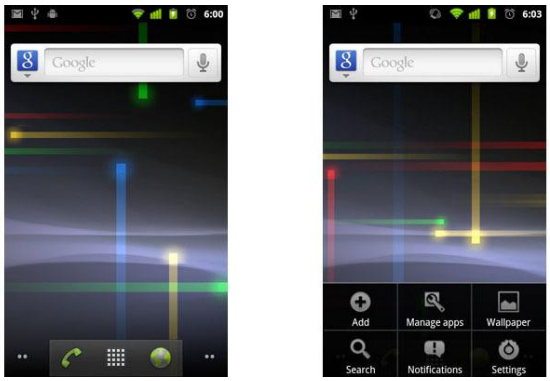
Figura 11
Teléfono con Android Froyo, dock y acciones por voz.



Nota: Tomado de Raphael (Raphael, 2019)

Android 2.3 Gingerbread: liberada en diciembre de 2010, esta versión fue el pilar para la identidad visual de Android ya que incluyó el logotipo en color verde brillante como se observa en la figura 12; además incluyó las primeras versiones de las aplicaciones y servicios de Google por defecto como Google Talk Video Vhat, Google e-books, el navegador privado y también el soporte para las aplicaciones de redes sociales (Raphael, 2019).

Figura 12
Teléfonos con Android Gingerbread.



Nota: Tomado de Raphael (Raphael, 2019)

Android 3.0 a 3.2 Honeycomb: liberado en febrero del 2011 como una versión solo para tabletas y de código cerrado. Presentó una interfaz renovada con un diseño holográfico que aprovechaba el tamaño de las pantallas de las tabletas; este concepto de una versión sólo para estos dispositivos no duró mucho. Sin embargo, varias de las ideas y aspectos incluidos en esta versión sentaron las bases del sistema operativo actual, como los botones en pantalla para los principales comandos de navegación y la lista de aplicaciones recientes (Raphael, 2019). En la figura 13 se puede observar el aspecto visual de Android 3.0 en una tableta.

Figura 13
Android Honeycomb

Nota: Tomado de Raphael (Raphael, 2019)

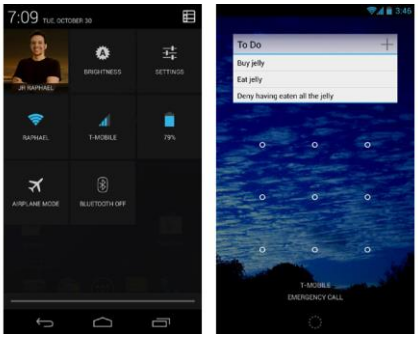
Android 4.0 Ice Cream Sandwich: liberado en octubre de 2011, esta versión sirvió como puente entre los dispositivos móviles, ya que funcionaba tanto para teléfonos como para tabletas, con una visión de interfaz de usuario única. Se mantuvieron ciertas características de su predecesor; además incluyó ciertos gestos, como deslizar para cerrar notificaciones y la funcionalidad de aplicaciones recientes; así como las primeras versiones para reconocimiento facial (Raphael, 2019). En la figura14 se observa la pantalla principal de Android 4, así como la pantalla de aplicaciones recientes.

Figura 14
Android Ice Cream Sandwich

Nota: Tomado de Raphael (Raphael, 2019)

Android 4.1 a 4.3 Jelly Bean: liberada al mercado entre julio de 2012 y julio de 2013, contenía atractivos visuales para el usuario; además se añadió la herramienta de inteligencia predictiva conocida como Google Now. Presentó el sistema de búsqueda por voz desde la pantalla y el navegador con un sistema más avanzado donde se utilizaban tarjetas que intentaban responder las preguntas directamente al momento de mostrar los resultados de la búsqueda. Añadió el soporte para multiusuario dentro del sistema operativo y una versión inicial del panel de configuración rápida en la parte superior (Raphael, 2019). En la figura 15 se puede observar las configuraciones de usuario al desplazar la barra superior de Android.

Figura 15
Android Jelly Bean



Nota: Tomado de Raphael (Raphael, 2019)

Android 4.4 Kitkat: Liberada a finales de octubre de 2013, cambió la orientación visual a colores más cálidos en la interfaz del sistema operativo. Incluyó la primera versión de "OK, Google" (asistente inteligente de Google) y añadió funcionalidades como la grabación de pantalla y la optimización del consumo de batería (Raphael, 2019). La figura 16 muestra el cambio apariencia entre Android 4.4 y su predecesor.

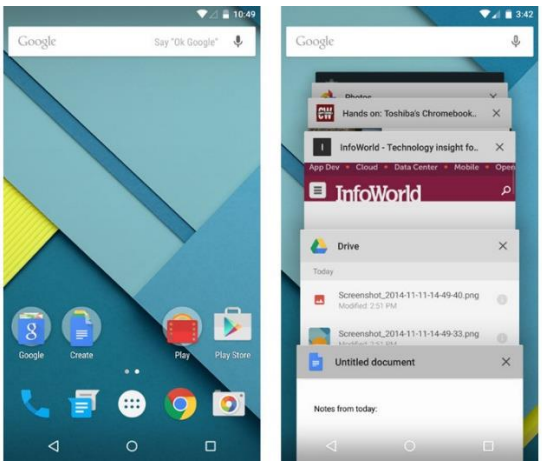
Figura 16
Android Kitkat

Nota: Tomado de Raphael (Raphael, 2019)

Android 5.0 a 5.1 Lollipop: liberada en octubre de 2014, fue crítica debido a todos los cambios que introdujo; entre los más destacados se encuentra el estándar "Material Design" que continúa presente en las actuales versiones de Android. El manejo de tarjetas tuvo gran aceptación por parte de Android convirtiéndose así en un

patrón de interfaz central para el usuario. Se introdujo la presentación de notificaciones en la pantalla de bloqueo y la lista de aplicaciones como se observa en la figura 17. Desafortunadamente debido a la cantidad de cambios realizados se produjeron varios errores (bugs), que fueron corregidos en versiones posteriores (Raphael, 2019).

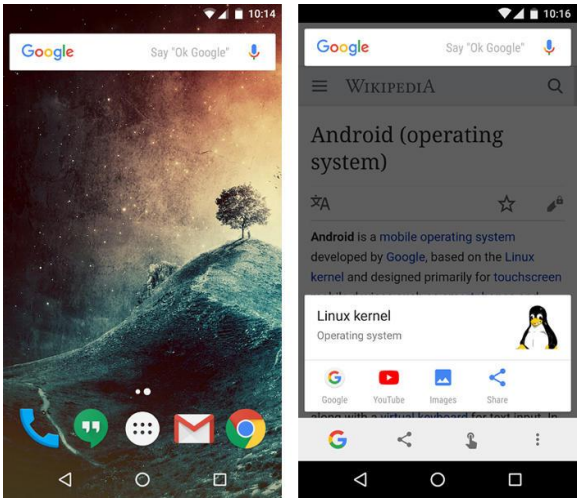
Figura 17
Android Lollipop



Nota: Tomado de Raphael (Raphael, 2019)

Android 6.0 Marshmallow: Android 6 fue liberado en octubre de 2015, esta versión parecía más una actualización a la versión anterior; tenía pocos cambios llamativos en las interfaces y colores, por lo que los usuarios casi no estuvieron conscientes de que existieran cambios. Esta versión introduce varios cambios significativos como la compatibilidad con lectores de huella y el USB-C; además factores de seguridad como la petición de permisos por parte de las aplicaciones (Raphael, 2019). La pantalla principal y el navegador web de Android 6 se muestran en la figura 18.

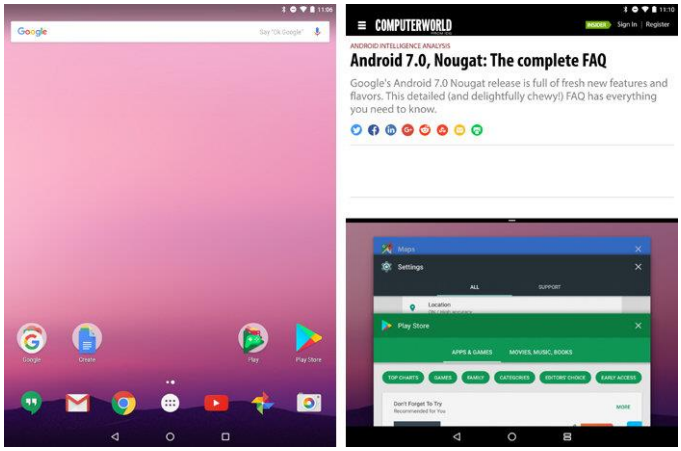
Figura 18
Android Marshmallow



Nota: Tomado de Raphael (Raphael, 2019)

Android 7.0 a 7.1 Nougat: Este sistema operativo fue liberado en agosto de 2016, entre las características más significativas introducidas en esta versión se encuentra un nuevo modo nativo que permite la pantalla dividida, un nuevo sistema para organizar las notificaciones por paquetes y la función de ahorro de datos como se muestra en la figura 19. También se implementó el Asistente de Google que se utiliza hoy en día en todas las nuevas versiones de Android y que significó el mayor esfuerzo de la compañía por proporcionar un asistente inteligente (Raphael, 2019).

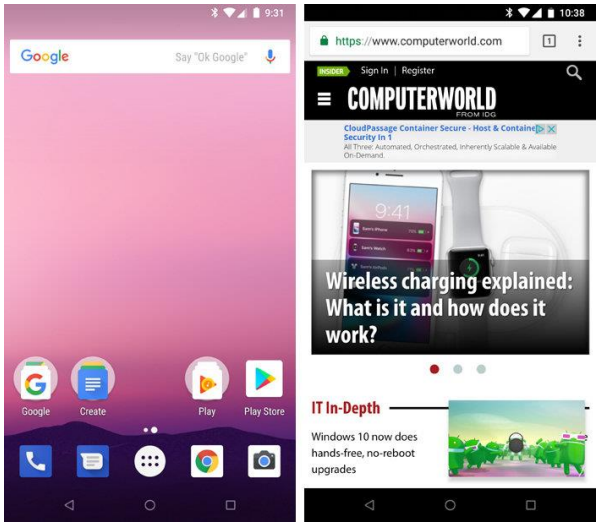
Figura 19
Android Nougat



Nota: Tomado de Raphael (Raphael, 2019)

Android 8.0 a 8.1 Oreo: Oreo salió al mercado en agosto de 2017, entre sus características se encuentran, la opción “imagen en imagen” de forma nativa, la opción de posponer las notificaciones por un período de tiempo y cambiar la manera en que las notificaciones pueden alertar al usuario. Sin embargo, el objetivo principal de Google fue alinear Android y Chrome OS para mejorar la experiencia de los usuarios en las Chromebook, debido a que se introdujo la primera versión de Project Treble; un proyecto que busca facilitar el acceso a las actualizaciones de software por parte de los fabricantes en los dispositivos (Raphael, 2019). En la figura 20 se observa la interfaz de Android 8 así como el navegador Chrome en el dispositivo móvil.

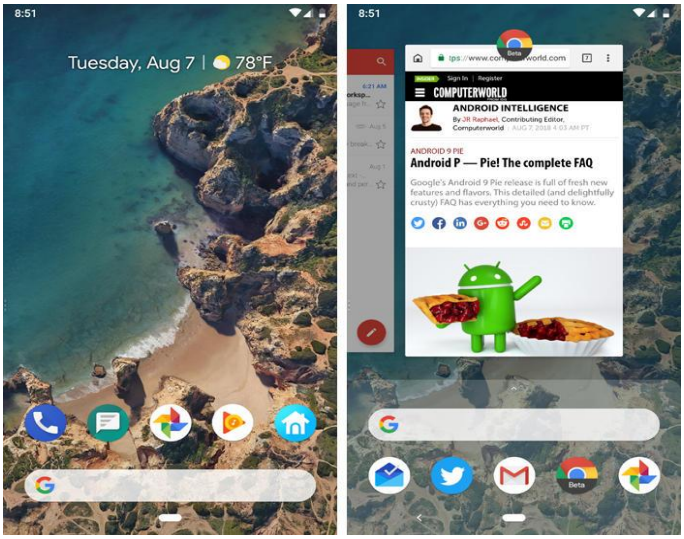
Figura 20
Android Oreo



Nota: Tomado de Raphael (Raphael, 2019)

Android 9 Pie: liberado en agosto de 2018, el cambio más significativo fue la inclusión de navegación por gestos, que permite reemplazar funciones como captura de pantallas, navegar entre aplicaciones o intercambiar las tradicionales teclas de atrás, inicio y reciente, por gestos. Además, incluyó otras funciones no tan perceptivas para el usuario, pero de sumo beneficio para el desempeño de los dispositivos como la administración de energía, el control de brillo, la administración de capturas de pantalla, actualización del manejo de los puntos de acceso Wi-Fi, mejoras en el modo de ahorro de energía y funciones nuevas para los sensores de huellas dactilares. También, hizo hincapié en las mejoras de seguridad y privacidad (Raphael, 2019). La figura 21 muestra la interfaz de Android 9 sin los botones por defecto en el “dock”.

Figura 21
Android Pie



Nota: Tomado de Raphael (Raphael, 2019)

Android 10 Q: Google lanzó la beta de la versión 10 de Android en marzo de 2019, existe poca información sobre todas las características de esta versión, pero se presentaron nuevas funciones mediante gestos, actualizaciones de seguridad, sistema de burbujas para multitareas y una base para un menú de uso compartido de Android (Raphael, 2019).

- **Lenguajes de desarrollo**

Android es una plataforma de software libre basada en Apache (servidor http) y de código abierto que permite la implementación en dispositivos móviles basados en el núcleo de Linux. Android incluye en su sistema operativo un software intermedio para capa de traducción en aplicaciones distribuidas (middleware). En 2007, Google lanzó Android como un lenguaje de desarrollo estándar para dispositivos móviles. Sin embargo, las aplicaciones Android están escritas principalmente en Java y compiladas en

formato Dalvik (DEX). Dalvik ejecuta archivos DEX que son convertidos en tiempo de compilación a clases estándar y archivos JAR. La arquitectura de Android permite la reutilización de componentes para simplificar el desarrollo y consumo de servicios y recursos. Android posee un paquete de desarrollo de software (SDK) que permite manejar varias funcionalidades y recursos del dispositivo tales como táctiles, sensores, acelerómetros, gráficos 3D y GPS; además permite la integración con aplicaciones y servicios de Google como correo electrónico, mensajería, calendarios, redes sociales, ubicación, entre otras (Gavalas & Economou, 2010).

- **Entorno de desarrollo integrado (IDE)**

Eclipse fue recomendado por Google durante varios años como el entorno de desarrollo ideal, por ser un entorno de código abierto que funciona mediante un patrón de actualización basado en “plugins". Permite el desarrollo de varios tipos de aplicaciones en diferentes lenguajes de programación; es compatible con Android mediante la instalación del kit de desarrollo de java (JDK) y las herramientas de desarrollo Android (ADT) (Oracle, 2019). La figura 22 muestra el logo del entorno de desarrollo Eclipse.

Figura 22
Logo Eclipse IDE

Nota: Tomado de Oracle (Oracle, 2019)

NetBeans: IDE de código abierto que permite el desarrollo de aplicaciones de escritorio, móviles y web a través de un conjunto de herramientas para el desarrollo de lenguajes como JAVA, HTML, JavaScript, PHP, C, C++, entre otros. NetBeans facilita el desarrollo de aplicaciones Android mediante la instalación del JDK y de los "plugins" de NBAndroid, Graddle Support, Android y NBAndroid Extensions (Oracle, 2019). En la figura 23 se observa el logo del entorno de desarrollo NetBeans.

Figura 23
Logo NetBeans IDE

Nota: Tomado de Digital Learning SL, 2019

Android Studio: actualmente es el entorno de desarrollo integrado (IDE) oficial para aplicaciones Android (Android, 2019). Posee una serie de funcionalidades que permiten analizar, fabricar, gestionar y desplegar aplicaciones Android. Dentro de estas funcionalidades ofrece un emulador, integración con plantillas de código y proyectos GitHub, herramientas, "frameworks" y soporte para integrar aplicaciones de Google como "Cloud Messaging" y "App Engine" (Android, 2019).

Figura 24
Android Studio Logo

Nota: La figura muestra el logo del entorno de desarrollo oficial para aplicaciones Android. (Android, 2019)

- **Requisitos**

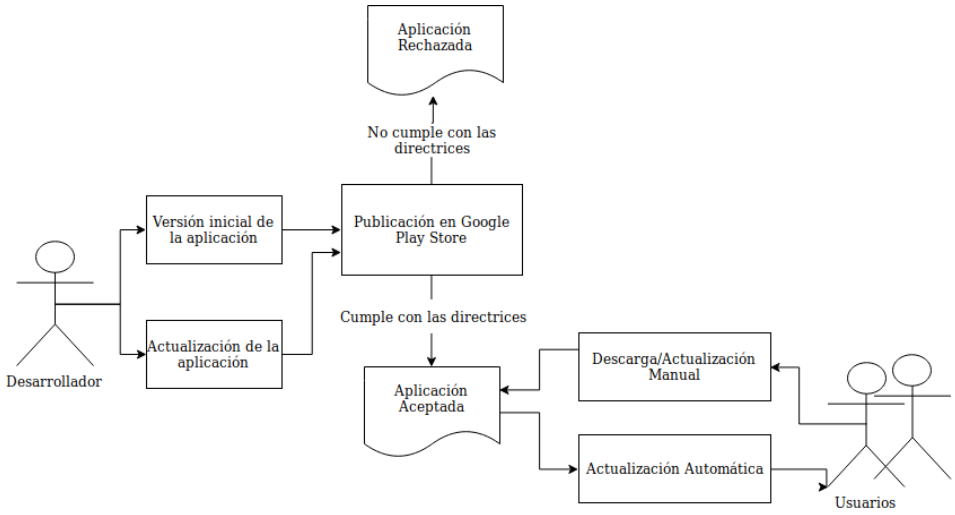
A diferencia de iOS, Android no es restrictivo en cuanto a requisitos para desarrollo. Las aplicaciones pueden ser realizadas utilizando cualquiera de los principales sistemas operativos, como Windows, Mac OS X o Linux. Debido a su flexibilidad no es restrictivo ni al software ni al hardware, por lo que puede utilizarse cualquier equipo actual (Goadrich & Rogers, 2011). Además, si el desarrollador no dispone de un dispositivo que ejecuta Android, puede probar sus aplicaciones a través de dispositivos virtuales Android (AVD), cada AVD puede ser configurado para representar un dispositivo móvil particular con diferentes tamaños de pantallas, versiones y marcas que se ejecutan en el emulador de Android (Android Developers, 2019). También permite implementar lecturas de GPS, señales de teléfono, SMS a través de una conexión Telnet y simular sensores como el acelerómetro, orientación y compás, mediante el SensorSimulator de OpenIntent (Openintents, 2019).

- **Distribución**

Existen diversas tiendas de aplicaciones que ofrecen un amplio catálogo para dispositivos Android. La tienda oficial de aplicaciones es Google Play Store la cual fue elaborada y distribuida por Google. Ofrece a los usuarios aplicaciones, libros electrónicos, películas y música. Entre su catálogo se encuentra contenido tanto gratuito como de pago y permite configurar la entrega de actualizaciones de manera tanto automática como manual. Cada aplicación contenida en la Google Play Store se encuentra en formato APK y contiene meta data, que identifica a la aplicación y permite al usuario tener una idea de los servicios y funcionalidades de la misma. Entre la información se incluye el título, el nombre del desarrollador, la puntuación que han dado los usuarios, la descripción, las aplicaciones similares, la fecha de publicación y otros datos relevantes (McIlroy, Ali, & Hassan, 2016). Podemos observar todo el proceso de publicación de una aplicación en la tienda de aplicaciones en la figura 25.

Figura 25

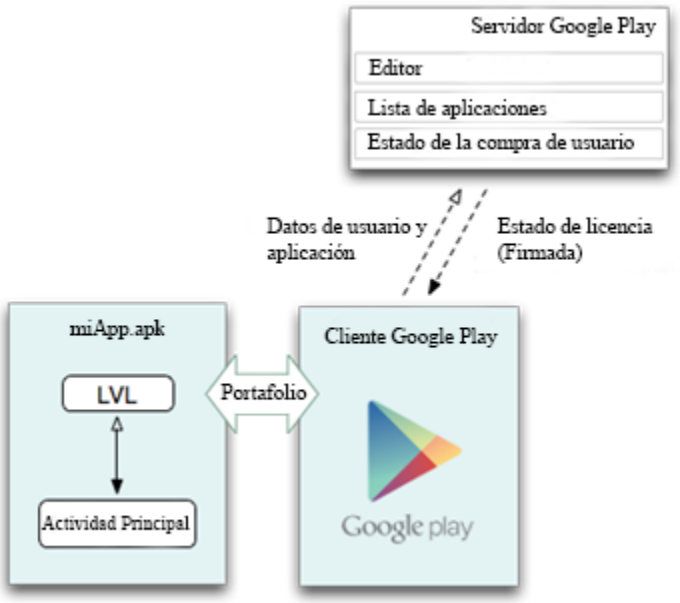
Proceso para publicar o actualizar una aplicación en Google Play Store



Nota: Tomado de McIlroy & Hassan (McIlroy, Ali, & Hassan, 2016)

Debido a que las aplicaciones pueden ser gratuitas o de pago, Google ofrece un servicio de red llamado “Google Play Licensing”, que permite comprobar si el usuario dispone de una licencia válida para el dispositivo o usuario actual. Google Play considera si la aplicación es gratis o si el usuario ha comprado la aplicación (Android Developers, 2019) como se muestra en la figura 26.

Figura 26
Proceso de verificación de licencias de Google Play



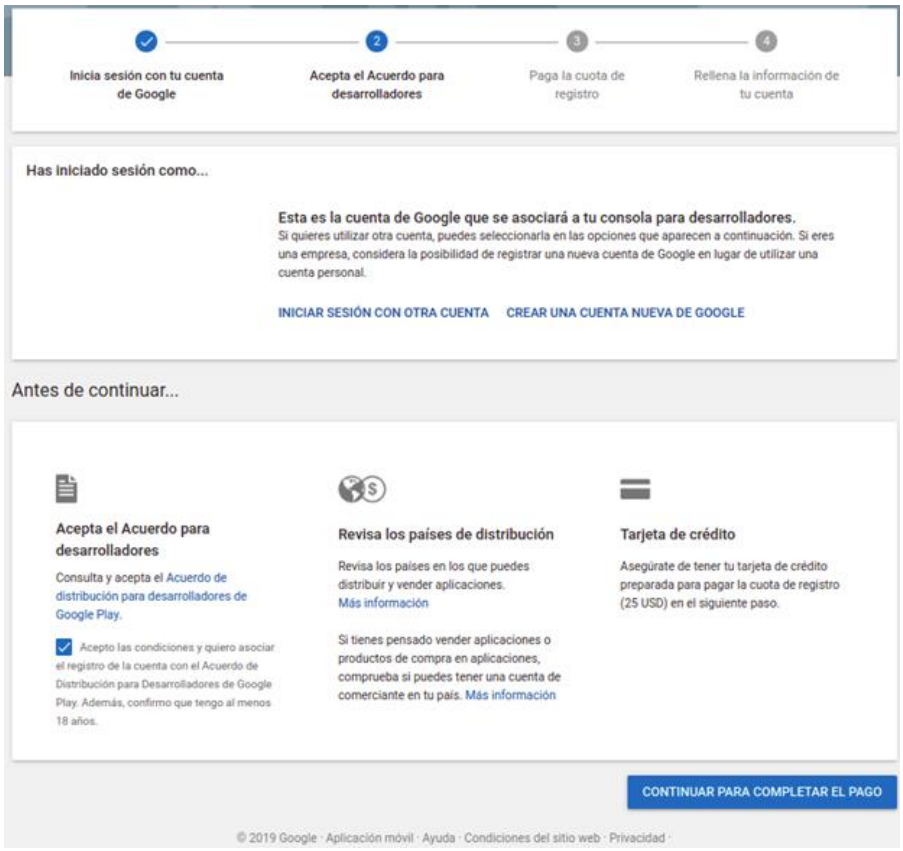
Nota: Tomado de Android Developers (Android Developers, 2019)

- **Licencias de desarrollo**

Para obtener una cuenta es necesario registrarse como desarrollador de aplicaciones de Google Play Store, se necesita utilizar una cuenta de Google que puede ser personal o empresarial dependiendo de las necesidades del usuario. El costo de la

licencia es de 25 dólares, el pago es único y asegura la licencia perpetua de desarrollador.

Figura 27
Pantalla para registro como desarrollador Android



Nota: La figura 27 muestra el proceso de registro para un desarrollador Android. Tomado de Android Developers (Android Developers, 2019)

Aplicaciones Híbridas

Las aplicaciones híbridas son un tipo de desarrollo que combina la tecnología nativa con las características web. La parte nativa de la aplicación utiliza el sistema operativo y mantiene el acceso directo en el dispositivo; mientras que mediante el consumo de servicios web o “APIs” se puede crear un motor de representación HTML incrustado que permite interactuar como puente entre el navegador y el resto de “APIs” del dispositivo. Dentro de las bibliotecas más

reconocidas para este tipo de desarrollo se encuentra PhoneGap, una biblioteca de código abierto que proporciona la interfaz de JavaScript y utilizar un contenedor nativo para crear aplicaciones avanzadas que pueden utilizar las funciones del dispositivo desde el contexto web. Además, mediante herramientas de desarrollo web se puede mantener archivos multimedia, funcionalidades y servicios en un servidor de aplicaciones o almacenadas localmente en el dispositivo (IBM Software, 2018).

Facebook

En 2003 Mark Zuckerberg lanzó un sitio web conocido como Facemash en el que reunía información sobre los estudiantes de Harvard. En febrero de 2004, Mark junto a los cofundadores Dustin Moskovitz, Chris Hughes y Eduardo Saverin crearon “The Facebook", que se expandió rápidamente desde Harvard hacia otras universidades como Stanford, Columbia y Yale. En septiembre de 2004 “The Facebook” realizó la liberación de “Facebook Wall” permitiendo a las personas compartir mensajes hacia sus amigos; lo que impulsó el crecimiento de la red social que para fines de 2004 había alcanzado un millón de usuarios activos dentro de su red. En el 2005, “The Facebook” había sido capaz de soportar más de 800 redes de instituciones educativas, en septiembre “The Facebook” cambió su nombre por Facebook y para octubre de 2005 se realizó la liberación de “Facebook Photos”, una aplicación oficial integrada con la que los usuarios podían cargar sus fotografías dentro de la red, lo que impulsó la unión de varios sitios internacionales (Facebook ©, 2019).

En el 2006, Facebook para dispositivos móviles, es liberado permitiendo añadir redes sociales de empresas y trabajos. Para agosto del 2006, Facebook introduce la primera versión de “Facebook API”; en septiembre del mismo año Facebook permite que cualquier persona pueda

registrarse llegando a tener aproximadamente 12 millones de usuarios activos en menos de cuatro meses. En mayo de 2007, Facebook hace la liberación oficial de su plataforma y es dividido en 2 partes, el núcleo que guarda la información de las personas con las características esenciales para el funcionamiento de la plataforma y las aplicaciones que ofrecen funcionalidades y características especiales a los usuarios para ser integradas tanto por los desarrolladores de Facebook como por desarrolladores externos (Facebook ©, 2019).

Facebook Messenger

Facebook Messenger se originó al utilizar los mensajes de Facebook, en dónde se permitía a los usuarios enviar mensajes instantáneos a los amigos a través de su sitio web. Estos mensajes se muestran a los usuarios en una ventana adjunta en el borde inferior derecho de la página web de Facebook. Debido a que muchos de los usuarios no querían ver todo el contenido de Facebook, pero tenían la necesidad de enviar mensajes y comunicarse con sus amigos y conocidos; Facebook creó una aplicación separada destinada a la mensajería instantánea móvil en 2011 (Zeng, 2016).

a) Plataforma de Facebook Messenger

En 2015 con la finalidad de mejorar la forma en la que tanto personas como empresas se comunican y comparten información; Facebook lanzó la plataforma de Facebook Messenger, la cual permite a los desarrolladores crear e integrar sus aplicaciones; estas pueden ser instaladas dentro de la aplicación o puede utilizar el cuerpo de los mensajes de Facebook Messenger para compartir su contenido (Facebook ©, 2019). A través de este tipo de integraciones, los usuarios pueden compartir todo tipo de contenido multimedia y las empresas

pueden ofrecer información, servicios, productos, confirmaciones de pedidos, actualizaciones de catálogos o envíos e incluso compras online (Facebook ©, 2019).

b) Aplicaciones Integradas en Facebook Messenger

Con el crecimiento exponencial de Facebook y la inclusión de Messenger dentro de la comunidad de usuarios de dispositivos inteligentes, Facebook Messenger ha continuado implementando mejoras continuas en su aplicación; convirtiéndola cada vez más en una aplicación robusta y completa. Recientemente Facebook incorporó juegos ocultos en su aplicación a los que se puede acceder utilizando comandos como "@fbchess" para el juego de ajedrez, "@dailycute" para obtener la foto de un animal o el emoji del balón de basquetbol permite acceder al juego de baloncesto (Zeng, 2016).

Es por este motivo que varias aplicaciones de terceros han visto la posibilidad de utilizar Facebook Messenger como un mecanismo de integración para ofrecer servicios y alcanzar un mayor número de usuarios.

Agente de respuesta virtual (chatbot)

Un agente de respuesta virtual o chatbot es un programa informático que puede interactuar con los usuarios utilizando el lenguaje natural. Los chatbots pueden ser utilizados para diferentes propósitos incluyendo aplicaciones de entretenimiento, educativas, aprendizaje, investigación, navegación web, entre otras. La utilidad y complejidad de estos se basa en ontologías, la administración de base de datos relacionales, el tipo de la interfaz y la facilidad de acceso a sus servicios (Al-Zubaide & Issa, 2011).

Chatbots en Facebook Messenger

Mark Zuckerberg, proclamó que los chatbots eran la solución al problema de la sobrecarga de aplicaciones. A través de la plataforma Facebook Messenger, los chatbots son capaces de responder con mensajes estructurados que incluyen texto, imágenes, enlaces y botones de acción. Es por esta razón que se pueden realizar diversas aplicaciones que permitan a los usuarios revisar contenido en tiempo real, realizar reservas, consultas de productos, noticias o incluso realizar compras al desplazarse por catálogos (Constine, 2016).

Componentes de Integración

a) ID de ámbito de página (PSID)

Cada vez que una persona comienza una conversación con un chatbot de Messenger, la plataforma asigna un código de identificación exclusivo de la página de Facebook con la que interactúa el usuario lo cual permite que los chatbots envíen la información solo a las personas que han iniciado una conversación. Además, cada vez que una persona envía un mensaje al chatbot, se incluye el código de identificación permitiendo al chatbot distinguir quién inició la acción y responder adecuadamente al usuario que mantiene la conversación (Facebook for developers, 2019).

b) APIs

La plataforma Messenger proporciona un conjunto de “API REST” que permite a los usuarios interactuar con los clientes y realizar acciones dentro del chatbot de Facebook Messenger (Facebook for developers, 2019).

- **API de envío**

La “API” de envío es el punto de integración principal con la plataforma de Facebook Messenger; esta permite enviar mensajes de texto, plantillas de mensajes estructuradas, contenido multimedia como imágenes, videos, audios y archivos. Así como incluir respuestas rápidas o acciones al remitente (Facebook for developers, 2019).

- **API de carga de archivos**

La “API” de carga de archivos permite adjuntar los activos multimedia como recursos desde una URL o archivos locales incluyendo imágenes, audio, videos y otros archivos. Además, asigna un id a los archivos adjuntos para acceder de una forma más eficiente y reutilizarlos en futuras peticiones (Facebook for developers, 2019).

- **API de perfil de Messenger**

La “API” de perfil de Messenger se utiliza para configurar, actualizar y eliminar configuraciones del chatbot. De esta manera permite que ciertas acciones como configurar una opción de pagos o crear una pantalla de bienvenida sea mucho más sencillo (Facebook for developers, 2019).

- **API de coincidencia de ID**

Cuando una persona se conecta a un sitio web mediante el inicio de sesión con Facebook, se crea un identificador específico de la aplicación; si una persona interactúa con el mismo negocio mediante Messenger se asigna otro identificador para la página,

lo que significa que un mismo usuario puede tener varios identificadores dependiendo el canal que utiliza para comunicarse con el proveedor. La “API” de coincidencia se encarga de asociar los ID de una persona con cada página proporcionando una comunicación ininterrumpida entre los canales de comunicación (Facebook for developers, 2019).

- **API de Protocolo de transferencia**

Esta “API” permite construir múltiples chatbots especializados para manejar aspectos separados de una conversación, en lugar de mantener una aplicación monolítica que podría ser demasiado complicada de administrar. Además, permite que una persona pueda tomar el mando de la conversación en cualquier momento que se considere necesario. (Facebook for developers, 2019).

- **API de información de mensajería**

La “API” de “Messaging Insights” permite recuperar de manera programada información que se encuentra en la página de Facebook, esto incluye las métricas y datos analíticos de dicha página (Facebook for developers, 2019).

c) Devolución de llamada web (Webhook)

Un “webhook” también conocido como “web callback” es un mecanismo que permite alterar el funcionamiento normal de una petición mediante notificaciones HTTP en tiempo real. Cuando sucede un evento (notificación, mensaje, archivo, imagen, video u otros) la plataforma Messenger envía una notificación al “webhook” expuesto por el

desarrollador mediante solicitudes POST, lo cual lo convierte en el único punto de integración entre el chatbot y Messenger (Facebook for developers, 2019).

Existen diferentes eventos “webhook”, en la tabla 2 se enlistan los eventos permitidos en Facebook Messenger.

Tabla 2
Eventos de Webhook en Messenger

Evento	**Descripción**
Messages	Se produce cuando se envía un mensaje a la página. Se puede recibir mensajes de texto o archivos adjuntos (plantilla, imagen, audio, video, archivo, ubicación).
messaging_account_linking	Se produce cuando se toca el botón de vincular cuenta o desvincular cuenta.
messaging_checkout_updates (beta)	Se produce cuando el usuario hace clic en el botón de comprar y se debe actualizar el precio teniendo en cuenta el costo de envío.
message_deliveries	Se produce cuando se entrega el mensaje enviado por el chatbot.
message_echoes	Se produce cuando la página ha enviado un mensaje a sí misma, por lo que el chatbot se envía un mensaje a sí mismo.
messaging_game_plays	Se produce después de que una persona haya jugado una ronda de juegos instantáneos.

Evento	Descripción
messaging_handovers	Se utiliza para notificar al usuario web cuando se realizan acciones utilizando el protocolo de transferencia de Messenger, es decir un chatbot puede pasar el control a otro sin que el usuario lo note.
messaging_options	Se produce cuando el usuario toca el botón de Enviar a Messenger, acepte una solicitud con la segmentación por lista de clientes o haya activado la opción para recibir mensajes a través de casilla de verificación.
messaging_payments (beta)	Se produce cuando el usuario realiza una solicitud para pagos.
messaging_policy_enforcement	Se produce cuando el usuario ha violado la política de Facebook en la página administrada.
messaging_postbacks	Se produce cuando alguien toca un botón de Postback, el botón inicio o un elemento del menú persistente.
messaging_pre_checkouts (beta)	Se produce cuando se requiere verificar acciones o cambiar precios antes de aceptar un pago.
message_reads	Este evento se produce cuando el usuario ha leído el mensaje enviado por el chatbot.
messaging_referrals	Este evento se produce cuando el usuario ya tiene una conversación iniciada con el chatbot y llega a ella siguiendo un enlace con un parámetro de referencia, haciendo clic en

Evento	Descripción
	un anuncio en una conversación de Messenger, iniciando la conversación desde la pestaña de sugerencias, o inicia o reanuda una conversación desde el "plugin" de chat con clientes.
Standby	Para chatbot que usan el protocolo de transferencia, este evento se produce cuando se envía un mensaje a la página, pero el chatbot no es el propietario del hilo actual, por lo que espera recuperar el control para procesar el mensaje.

Nota: Información obtenida de Facebook for developers (Facebook for developers, 2019).

d) Messenger Webview

Es una vista web estándar que permite ofrecer al cliente experiencias que requieren una interfaz más compleja y que es difícil de mostrar en el chat de la conversación; esta vista carga páginas web con herramientas y bibliotecas web de terceros (Facebook for developers, 2019).

e) Messenger Extensions JS SDK

Proporciona acceso a la información y funciones principales de Messenger, por ejemplo, obtener información como el PSID (identificación del usuario y la página), el contexto del mensaje, aceptar pagos o compartir otro tipo de información de la vista nuevamente en la conversación (Facebook for developers, 2019).

f) Extensiones de chat

Permite crear experiencias de Messenger compartidas y colaborativas en conversaciones de grupos. Por ejemplo, puede permitir que varias personas en una conversación actualicen una lista, cada vez que una persona inicia la conversación con el chatbot, se abre una nueva conversación entre el usuario y el chatbot con las opciones seleccionadas (Facebook for developers, 2019).

g) Procesamiento de lenguaje natural incorporado

Permite integrar la plataforma de lenguaje natural "Wit.ai" a los chatbots de Messenger, permitiendo detectar la intención y el significado de los mensajes enviados por el usuario. Mediante esta opción se puede procesar elementos comunes de conversación o crear entidades y entrenar a "Wit by Example" (servicio que permite entrenar un chatbot) para reconocer estructuras específicas y obtener el análisis en el "webhook" (Facebook for developers, 2019).

- **Complementos web**

Permite integrar complementos web mediante elementos estandarizados en el sitio, como botones que abren automáticamente la conversación con el chatbot, obtener datos personalizados del complemento (incluir contenido de otras páginas), información del usuario, de la ubicación, etc. (Facebook for developers, 2019).

Componentes de Conversación

Actualmente una conversación por medios electrónicos no se compone únicamente de texto, también se utilizan imágenes, audio, video y transferencia de archivos; todos estos elementos están disponibles al momento generar un agente de

respuesta en la plataforma de Facebook Messenger. La plataforma de Messenger provee plantillas de mensajes, plantillas de respuestas rápidas o acciones del usuario, plantillas para pantallas de bienvenida y plantillas para menú fijos que ayudan a estructurar la forma en la que se mostrará la conversación con el chatbot (Facebook for developers, 2019).

a) Mensajes de texto

Las conversaciones entre un chatbot de Facebook Messenger y un usuario, que solo utilizan texto pueden ser procesadas con la herramienta de procesamiento de lenguajes naturales integrada (NLP) para administrar las peticiones al chatbot (Facebook for developers, 2019).

b) Activos y archivos adjuntos

Los activos permitidos por la plataforma de Facebook Messenger son imágenes, audio, video y archivos adjuntos; estos pueden ser enviados mediante una URL o desde el gestor de archivos del dispositivo electrónico. Para subir estos activos, la plataforma ofrece la “API” de subida de archivos adjuntos en el caso que se necesite guardar el activo antes de que este sea requerido o la “API” de envío que permite guardar el archivo al momento de enviarse (Facebook for developers, 2019).

c) Plantillas de mensajes

Estas plantillas son formas en las que se logra que un mensaje tenga una estructura definida, para diferentes casos. Dichas plantillas son útiles para organizar información al momento de presentarla ya que contienen botones, imágenes y listas multimedia (listado de combinaciones entre imágenes, texto y botones) en un solo

mensaje con la finalidad de dar mayor funcionalidad al usuario. También ofrecen opciones como abrir una vista web, compartir contenido o enviar un "Postback" (petición del usuario al chatbot a través de un elemento de respuesta rápida como botón o burbuja flotante) al servicio del "webhook" (Facebook for developers, 2019).

d) Respuestas rápidas

Son opciones que se encuentran predeterminadas en el chatbot, y que el usuario puede seleccionar para responder sin la necesidad de escribir texto. Cuando se escoge una opción, esta se reemplaza en el chatbot por una sola palabra que se envía hacia el "webhook". Además, las respuestas rápidas pueden tener adjunta una imagen, que permita mejorar la experiencia del usuario (Facebook for developers, 2019).

e) Acciones del remitente

Se utiliza para darle al usuario una experiencia cercana a una conversación real; envía confirmaciones y también el indicador de lectura lo que le permite al usuario saber que el chatbot está escribiendo una respuesta (Facebook for developers, 2019).

f) Pantalla de bienvenida

Se muestra al momento de iniciar una conversación entre el usuario y la página, puede incluir información de la empresa o configurar un saludo por defecto (Facebook for developers, 2019).

g) Menú fijo

Es un menú continuo donde el usuario puede acceder de manera rápida a todas las opciones que ofrece el chatbot; este puede ser la única opción para que el usuario

pueda comunicarse; o también se puede fusionar con el editor de texto de la plataforma (Facebook for developers, 2019).

h) Vistas Web

Facebook Messenger permite abrir una vista web estándar desde la plataforma, ofreciendo así opciones al usuario que no se encuentran disponibles o pueden ser complicadas de crear utilizando los clásicos cuadros de mensaje (Facebook for developers, 2019).

Capítulo III -Estado del Arte

Planteamiento de la revisión de literatura

En esta fase se realizó una breve descripción del problema de investigación para proporcionar un contexto para la búsqueda de estudios científicos; posteriormente se procedió a definir un objetivo de búsqueda y se plantearon las preguntas de investigación para alinear la búsqueda en relación al problema de investigación y finalmente se definieron los criterios de inclusión y exclusión.

Conformación del grupo de control (GC)

Para seleccionar los estudios del grupo de control los 2 integrantes del proyecto, en conjunto con el profesor tutor, propusieron un total de 5 artículos relacionados con la temática planteada. Posteriormente se realizó la lectura de todos los artículos para determinar si eran beneficiosos para el proyecto. Finalmente, mediante una validación cruzada se seleccionó los siguientes artículos.

Tabla 3
Grupo de Control

Título	Cita	Palabras clave
"Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage"	(Böhmer, Hecht, Schöning, Krüger, & Bauer, 2011)	Mobile apps, usage, mobile devices, measuring, large-scale study, mobile application, application usage, app usage, app's installation, measuring mobile app usage.

Título	Cita	Palabras clave
"Why do people abandon mobile social games? Using Candy Crush Saga as an example"	(Wei, Lee, Lu, Tzou, & Weng, 2015)	Abandon, discontinuing, discontinue usage, mobile devices, not satisfied, embedded, lose interest, usage intention
"Why people hate your app: Making sense of user feedback in a mobile app store"	(Fu, Lin, Li, Faloutsos, Hong, & Sadeh, 2013)	mobile app market, user rating and comments, analysis, hate, dislike

Construcción y afinación de la cadena de búsqueda

Utilizando las palabras claves obtenidas en los artículos científicos del grupo de control se conformó una cadena de búsqueda inicial: ("USE" OR" ABANDON" OR "ISSUES" AND "MOBILE" AND "APP" OR "APPS" OR "APPLICATION"), esta se utilizó en la base digital ACM DIGITAL LIBRARY.

Mediante la utilización de esta cadena se obtuvo algunos resultados, sin embargo, el número de estudios era demasiado extenso, por lo que se procedió a refinar la cadena hasta obtener la cadena de búsqueda final: ("USAGE" OR "ABANDON" OR "DISENGAGEMENT" OR "UNUSED" OR "DISLIKE") AND ("MOBILE APP" OR "MOBILE APPS" OR "MOBILE APPLICATION").

Selección de estudios

Una vez establecida la cadena de búsqueda final se obtuvo un total de 627 artículos científicos relacionados con la problemática planteada.

1. **Vigencia:** Los artículos obtenidos se aplicaron fueron limitados desde el año 2010 para asegurar la actualidad de la investigación.
2. **Coherencia con la investigación:** Para verificar que los artículos encontrados se alienan con el tema de investigación se procedió a leer cada uno de los artículos obtenidos, seleccionando los de mayor relevancia y aporte dentro del contexto de la investigación.

Una vez filtrados los artículos se obtuvo un total de 6 estudios que conforman el grupo primario como se muestran en la Tabla 4.

Tabla 4
Estudios Seleccionados

Código	Título	Cita
EP1	Falling asleep with Angry Birds, Facebook and Kindle: a large-scale study on mobile application usage.	(Böhmer, Hecht, Schöning, Krüger, & Bauer, 2011)
EP2	Why do people abandon mobile social games? Using Candy Crush Saga as an example	(Wei, Lee, Lu, Tzou, & Weng, 2015)
EP3	Why people hate your app: Making sense of user feedback in a mobile app store.	(Fu, Lin, Li, Faloutsos, Hong, & Sadeh, 2013)
EP4	Understanding and prediction of mobile application usage for smart phones.	(Shin, Hong, & Dey, 2012)
EP5	Apps with Benefits: Using Benefits and Burdens to Predict Mobile App Usage.	(Cheng, Lin, Nijhawan, Westhem, & Bernstein, 2017)
EP6	A customer value, satisfaction, and loyalty perspective of mobile application recommendations	(Xu, Peak, & Prybutok, 2015)
EP7	Maintaining Social Connectedness: Hanging Out Using Facebook Messenger.	(Zeng, 2016)

Elaborar el estado del arte

Con la finalidad de evidenciar la situación actual de las aplicaciones móviles, así como de aquellas aplicaciones que se pueden integrar en otro tipo de aplicaciones se procedió a

seleccionar estudios que puedan aportar tanto a desarrollo, funcionalidad, usabilidad y diseño. Además, se observó el motivo por el cual las personas dejan de utilizar aplicaciones móviles y la factibilidad de integrar funcionalidades o aplicaciones móviles a través de servicios en la plataforma de Facebook Messenger. Los estudios seleccionados se muestran a continuación:

EP1 (Böhmer, Hecht, Schöning, Krüger, & Bauer, 2011)**: "Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage".** Böhmer & otros realizaron un análisis estadístico sobre el uso de las aplicaciones móviles. Para ello utilizaron la herramienta "AppSensor" mediante la cual se proporcionaron los datos de más de 4100 usuarios en un período superior a cuatro meses, esta aplicación fue capaz de recoger la información de las aplicaciones móviles en un dispositivo, desde que es instalada, mientras se encuentra o no en uso, cuando es actualizada y finalmente cuando es desinstalada. En este estudio se determinó que los teléfonos móviles se utilizan principalmente para la comunicación (texto y voz) y que por ello se tiene una correlación en que las aplicaciones móviles más utilizadas son las de redes sociales y música.

EP2 (Wei, Lee, Lu, Tzou, & Weng, 2015)**: "Why do people abandon mobile social games? Using Candy Crush Saga as an example".** Wei & otros se enfocan en identificar los factores por los cuales las personas de Taiwán dejan de utilizar los juegos sociales para dispositivos móviles. Además, señala que "los jugadores de juegos móviles son inconstantes, recogen y abandonan los juegos rápida y fácilmente". Los juegos sociales móviles son aplicaciones sencillas y fáciles de usar que se integran en sitios de redes sociales (SNS) o servicios de comunicación social, lo que permite a los jugadores invitar fácilmente a sus amigos a unirse a un juego para colaboración o competencia, manteniendo y fortaleciendo sus

relaciones. También indica que más de 50 millones de aplicaciones son descargadas diariamente; no obstante, el 95% de estas aplicaciones dejan de ser utilizadas por el usuario. Este estudio se enfoca en cómo el desarrollador debe buscar medidas para retener al usuario luego de la descarga, basándose en el costo percibido por el usuario ya sea este monetario (compras, micro transacciones, licencia, etc.) o no monetario (tiempo, privacidad, dificultad, etc.).

EP3 (Fu, Lin, Li, Faloutsos, Hong, & Sadeh, 2013)**: "Why people hate your app: Making sense of user feedback in a mobile app store".** Fu & otros se enfocaron en determinar la aceptación de las aplicaciones móviles mediante la herramienta "WisCom" que permite analizar las calificaciones y comentarios de los usuarios en la tienda de Google Play Store. Para la obtención de los metadatos se realizó una búsqueda de las páginas HTML que contienen los atributos propios de la aplicación como son: nombre, categoría, número de descargas, puntuación promedio de los usuarios, precio y clasificación del contenido. Posteriormente se analizó la opinión de los usuarios desde tres niveles diferentes: "análisis centrado en la palabra del comentario", el cual se refiere los comentarios emitidos por los usuarios, "análisis centrado en la aplicación", que se enfoca en la puntuación y clasificación del contenido y por último "análisis centrado en el mercado", que se refiere al número de descargas y precio. El objetivo de la herramienta es beneficiar a los usuarios finales, desarrolladores de aplicaciones, operadores de mercado y otras partes interesadas relevantes.

EP4 (Shin, Hong, & Dey, 2012)**: "Understanding and prediction of mobile application usage for smart phones".** Este artículo se enfoca en la enorme cantidad de aplicaciones disponibles para el usuario y también el gran número de aplicaciones que las personas tienen en

sus teléfonos, se plantea que para el usuario es muy difícil encontrar la aplicación deseada entre todas las que tiene almacenadas en su dispositivo; se realiza un análisis de utilización de una aplicación, basada en el tiempo para generar una pantalla de inicio donde se muestran las aplicaciones más usadas para ese intervalo de tiempo para de esta manera lograr un acceso a la aplicación mucho más rápido y sencillo.

EP5 (Cheng, Lin, Nijhawan, Westhem, & Bernstein, 2017)**: "Apps with Benefits: Using Benefits and Burdens to Predict Mobile App Usage".** Cheng & otros en su estudio proponen que los beneficios de las aplicaciones, y el nivel de carga en el usuario como una forma de predecir el uso y abandono de una app. Para ello proponen una escala de beneficios y utilizan la escala de carga del usuario propuesta por (Suh, Shahriaree, Hekler, & Kientz, 2016). La escala de beneficios consta de cuatro parámetros: si una aplicación es 1) útil e informativa, 2) agradable y permite la búsqueda de intereses, 3) permite la interacción social, y 4) tiene una buena usabilidad y diseño visual / de interacción. El estudio se realizó a 347 usuarios encontrando que los beneficios resultan ser más útiles que las cargas al momento de predecir si una aplicación se utiliza o se abandona.

EP6 (Xu, Peak, & Prybutok, 2015)**: "A customer value, satisfaction, and loyalty perspective of mobile application recommendations".** En el estudio "A customer value, satisfaction, and loyalty perspective of mobile application recommendations" los autores desarrollan un modelo de investigación basado en el valor, la satisfacción y la lealtad de un cliente (VSL). Esta investigación desglosa el valor del cliente en beneficios utilitarios que consisten en la utilidad y la calidad de la aplicación, los beneficios hedónicos que consisten en la estética y el disfrute de la aplicación y los sacrificios monetarios y no monetarios que consisten

en la dificultad, tecnicidad, riesgo de privacidad, etc. Mediante estos factores se analizaron en 347 usuarios verificando la intención de continuidad del usuario con la app y su intención de recomendarla a un tercero.

EP7 (Zeng, 2016)**: "Maintaining Social Connectedness: Hanging Out Using Facebook Messenger".** El trabajo de investigación de Zeng se direcciona en cómo mantener la conexión social mediante el uso de Facebook Messenger. El estudio explica cómo jóvenes y adultos de entre 18 y 29 años se conectan para "pasar el rato". Pasar el rato significa establecer un vínculo de familiaridad a través de una serie de conversaciones en chats, que les permitan no solo conversar, si no también realizar otro tipo de actividades. En este estudio se da una idea de las oportunidades de diseño que se presentan en Facebook Messenger, como la posibilidad de añadir y enviar gifs, pegatinas, emoticones, imágenes o videos. Además, explica sobre algunos de los recientes juegos que han sido añadidos a Messenger como por ejemplo el juego de basquetbol o el juego de ajedrez. Zeng menciona, además, la posibilidad de integrar elementos de inteligencia artificial como chatbots para garantizar una experiencia totalmente nueva e innovadora para el usuario.

Características del estado del arte

Existen varios estudios sobre la utilidad y la usabilidad de las aplicaciones móviles, otros se enfocan en los problemas que presentan dichas aplicaciones. También existen estudios que buscan identificar las causas de que las personas dejen de utilizar una aplicación de un tipo específico. Sin embargo, la mayoría de los estudios actuales se enfocan en identificar un problema y corregirlo con la finalidad mejorar las aplicaciones móviles y lograr que los usuarios se fidelicen a ellas. Pero no se logró evidenciar estudios orientados a herramientas, técnicas o

propuestas que permitan a las empresas proveer a los usuarios de una alternativa a las aplicaciones móviles, la falta de este tipo de estudios permite a los investigadores incursionar en este ámbito.

Capítulo IV - Metodología

Para comprobar si la aplicación integrada en Facebook Messenger es capaz de sustituir a la aplicación móvil era necesario comparar ambas aplicaciones de manera sistemática y precisa. El principal problema al evaluar una aplicación es que el análisis no sea objetivo, donde se dé un enfoque distinto para cada aplicación y entregar un resultado sesgado que no muestre la realidad del estudio. Para evitar este tipo de inconsistencias se seleccionó una metodología que permita: alinearse al objetivo, establecer parámetros de evaluación técnicos o de apreciación e incluir tanto a los investigadores como a usuarios externos.

Enfoque sistemático para la evaluación de desempeño

Para realizar la evaluación de las aplicaciones se tomó como referencia la metodología propuesta en el libro "Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling" (Jain, 1991), en donde se propone los siguientes pasos:

1. **Definir los objetivos del estudio y del sistema:** se definen los objetivos del estudio y se establecen los límites del sistema a estudiar.
2. **Lista de servicios y resultados:** consta de todos los servicios que el sistema provee y los resultados que se espera de cada uno de estos.
3. **Selección de métricas:** son los criterios con los que vamos a evaluar el rendimiento de cada sistema.
4. **Lista de parámetros:** se encuentran todas las variables que pueden afectar al rendimiento de un sistema.

5. **Selección de factores a estudiar:** son los parámetros, de la lista anterior, que serán modificados para evaluar los sistemas.
6. **Seleccionar la carga de trabajo (usuarios y peticiones):** se refiere al número de usuario y peticiones con las que se evaluará el rendimiento.
7. **Experimentos del diseño:** son las pruebas, o medición de las aplicaciones para obtener la información a ser evaluada.
8. **Analizar e interpretar los datos:** son el conjunto de herramientas, tablas, algoritmos o técnicas que se utilizan para analizar la información obtenida.
9. **Presentar los resultados:** consiste en mostrar el resultado de la evaluación por medio de tablas o párrafos que den respuesta a los factores de estudio que fueron planteados.

Capítulo V - implementación de la solución y análisis de resultados

Diseño de la arquitectura de la aplicación

Para el desarrollo de la investigación se optó por desarrollar 2 aplicaciones, la primera corresponde a una aplicación para dispositivos Android, basada en la nube como se observa en la figura 28. La segunda aplicación es un chatbot que se integra a la aplicación de Facebook Messenger y se evidencia en la figura 29. Ambas aplicaciones utilizan servicios REST para la comunicación con el servidor principal, el cual se encarga de la comunicación con la base de datos.

Figura 28
Arquitectura de la Aplicación Móvil

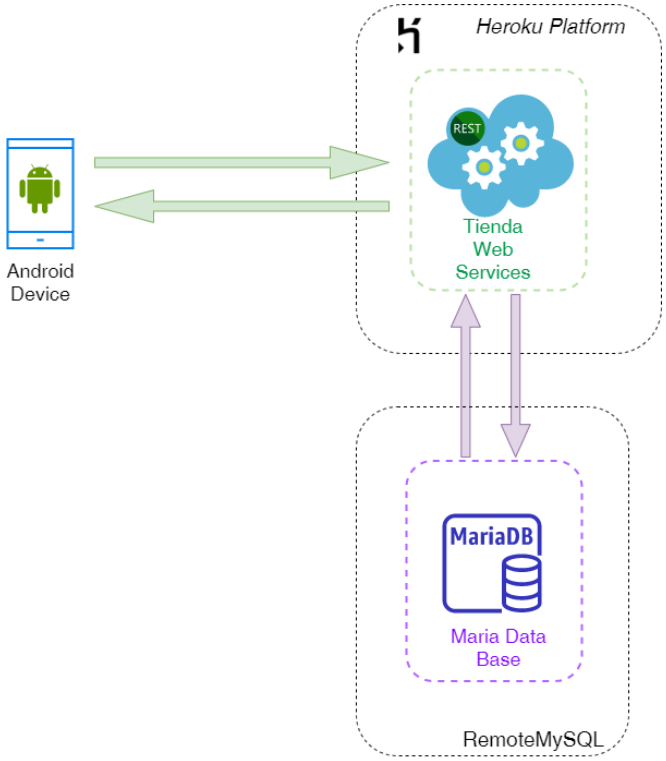
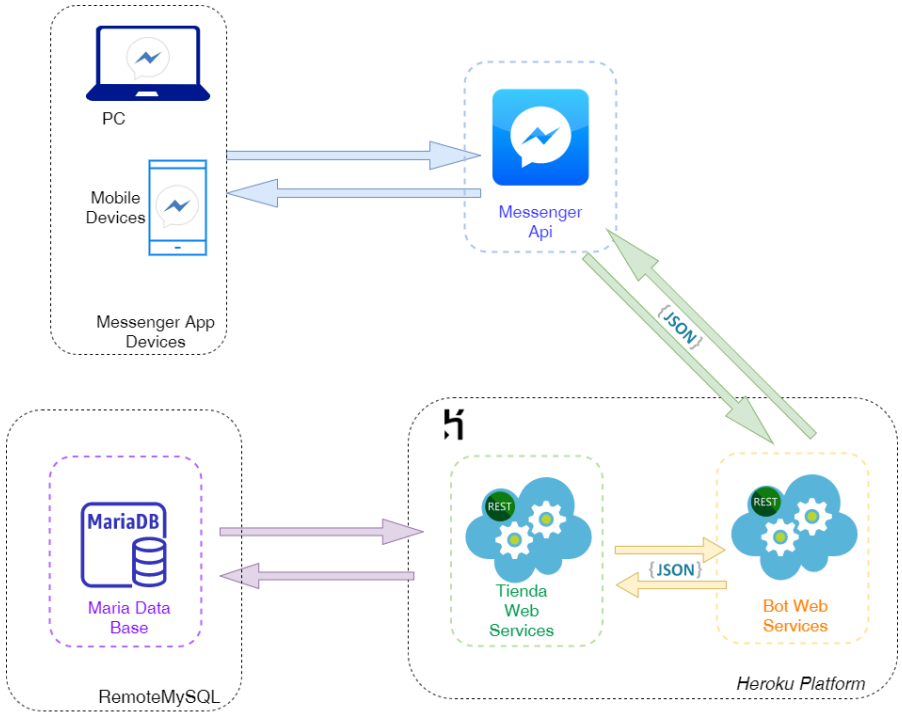


Figura 29

Arquitectura de la Aplicación Integrada en Facebook Messenger



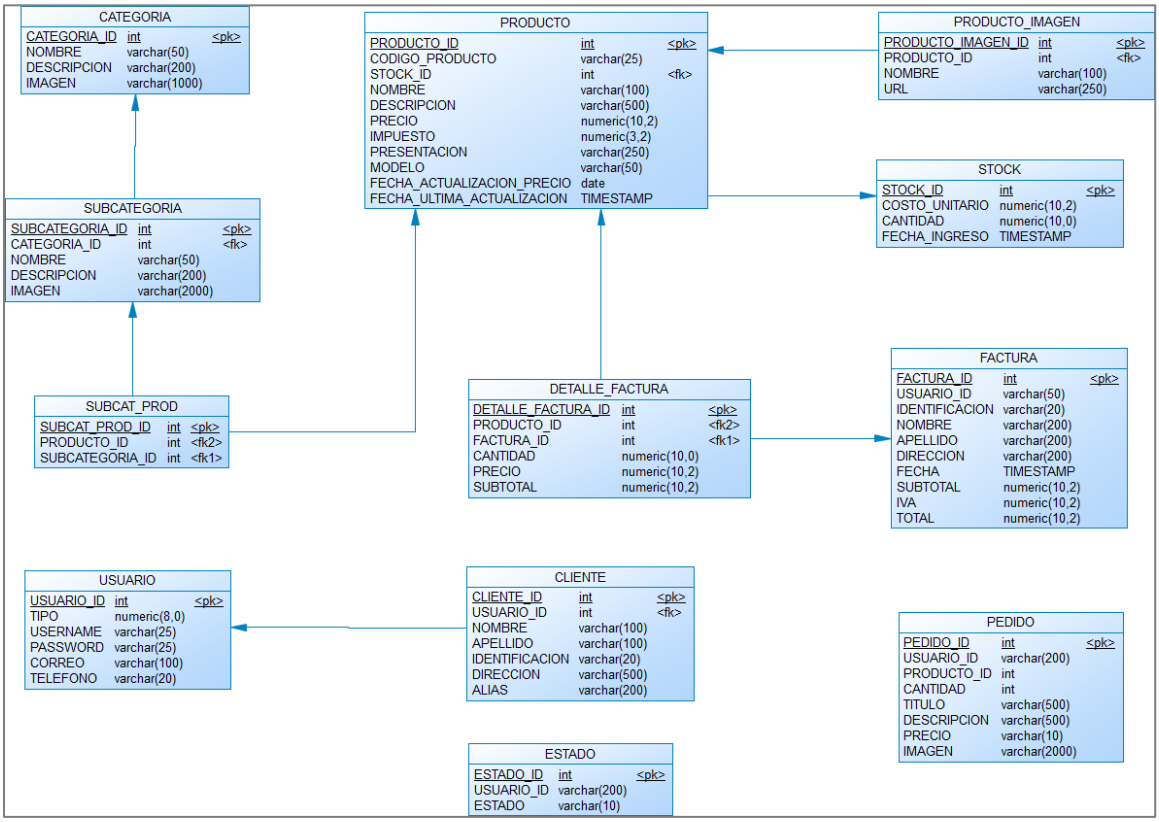
Para el despliegue de las aplicaciones se utilizó la infraestructura proporcionada por la plataforma Heroku. Además, para la gestión, mantenimiento y despliegue del servidor de base de datos se utiliza una base de datos MariaDB alojada en el servidor RemoteMySQL.

Modelado de la base de datos

Para el desarrollo de la investigación se optó por utilizar una Base de Datos relacional en MariaDB (MariaDB , 2018), la cual permite la correcta relación entre las entidades de los productos y los clientes como se puede observar en la figura 30.

Figura 30

Modelo físico de la base de datos



Las relaciones entre cada tabla señalan que: existen varias categorías y cada una de ellas puede contener una o más subcategorías. Un producto puede pertenecer a varias subcategorías y una subcategoría puede contener varios productos, por lo cual se decidió crear una entidad de relación que permita dicha asignación de una manera más fácil, eliminando la relación de varios a varios entre producto y subcategoría. Un producto puede tener una o varias imágenes, además cada producto tiene un número de stock que puede ser vendido. Un cliente puede o no ser usuario de la aplicación, sin embargo, un usuario debe estar asociado a un cliente. Para la facturación se entiende que una factura puede tener uno o varios detalles, por lo cual se almacenan los detalles de la factura, adjuntando los datos del producto, la cantidad, el precio y el subtotal de cada detalle. La tabla de pedido es una tabla temporal que permite almacenar los

productos elegidos por el cliente a manera de carrito de compras. Por último, la tabla estado, permite mantener control sobre las acciones que realiza el cliente desde la aplicación de Facebook Messenger debido a que la comunicación es asíncrona.

Diseño de servicios web

Para el desarrollo de la investigación se definió utilizar servicios Restfull, lo que permitía desarrollar la aplicación en cualquier lenguaje de desarrollo actual. Sin embargo, para este caso específico se utilizó Nodejs; que es un entorno de ejecución para JavaScript, cuyas librerías facilitan el desarrollo y despliegue de un servidor que gestione las peticiones y respuestas hacia los servicios expuestos, así como la conexión con la base de datos. Para el intercambio de paquetes entre las aplicaciones se optó por el formato de intercambio de paquetes JSON (Notación de Objetos de JavaScript) debido a que es un formato ligero de intercambio de datos y facilita tanto la lectura como escritura de datos hacia la base de datos. La aplicación se desplegó en la infraestructura web proporcionada por la librería Express, misma que proporciona un conjunto de características, middleware y métodos de utilidad HTTP, alojada en la infraestructura proporcionada por la plataforma Heroku (Heroku, 2018). Para la comunicación con la base de datos se configuró el driver de MySQL para Nodejs, mismo que ofrece un conjunto de características y funciones que minimiza la dificultad y gestionan la comunicación entre el servidor y la base de datos. El detalle de los servicios expuestos, así como el formato de las peticiones y respuestas se especifica en el **Anexo 2 (Manual de servicios Rest)**.

Diseño de la aplicación móvil

El desarrollo de la aplicación móvil se realizó usando el IDE Android Studio; se utiliza Gradle como sistema de compilación, java como lenguaje base, y XML para el diseño de las

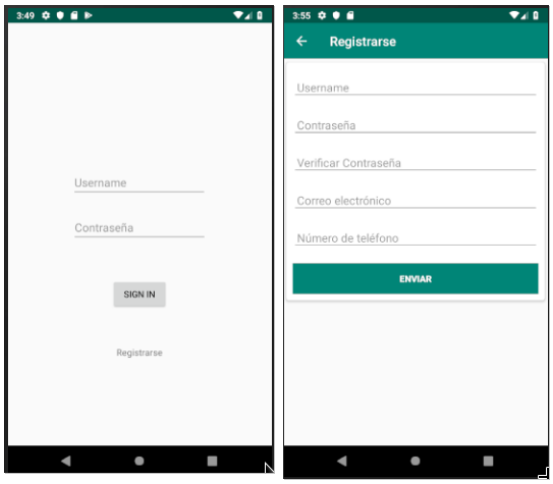
pantallas o actividades de la aplicación; se consumen los servicios Rest creados utilizando peticiones HTTP como se observa en la figura 31.

Figura 31
Consumo de servicios REST a través de peticiones HTTP desde Android.

```
// request data from menu API
HttpClient client = new DefaultHttpClient();
HttpConnectionParams.setConnectionTimeout(client.getParams(), timeout: 15000);
HttpConnectionParams.setSoTimeout(client.getParams(), timeout: 15000);
HttpUriRequest request = new HttpGet(MenuAPI);
HttpResponse response = client.execute(request);
InputStream atomInputStream = response.getEntity().getContent();
```

Se creó una aplicación con pantallas para los procesos de registro, inicio de sesión, categorías de productos, carrito de compras, registro de datos personales del usuario, verificación de compra y consultar de compras anteriores.

Figura 32
Pantalla de ingreso y registro de usuario



Nota: La figura muestra las pantallas de inicio de sesión y registro en la aplicación móvil desarrollada.

Los servicios REST generan una respuesta de tipo JSON, que contiene varios registros, este archivo es recibido por la aplicación para ser procesado; cada uno de estos registros es graficado al instante utilizando un “LayoutInflater” en la pantalla que realizo la petición; siendo

así una aplicación dinámica donde la información no necesita ser almacenada como se observa en la figura 33.

Figura 33
"LayoutInflater" para graficar dinámicamente

```
LayoutInflater inflater = (LayoutInflater) activity
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
convertView = inflater.inflate(R.layout.lsv_item_menu_list,  root: null);
holder = new ViewHolder();

convertView.setTag(holder);
```

Se utiliza SQLite en el dispositivo móvil para guardar información de la aplicación; cuando se añade un producto al carrito se inserta un registro en la base de datos para registrar el pedido; y al momento de realizar la compra se envía la información del pedido mediante un JSON a un servicio REST tipo post mediante una petición http; que será procesado en el servidor, y devolverá una respuesta cuando se haya procesado y generado la factura del pedido. El método para el almacenamiento de la información en el dispositivo se observa en la figura 34.

Figura 34
Método para guardar información en la base de datos de SQLite

```
public void addData(long producto_id, int cantidad, double precio, double impuesto, double subtotal,
                    String nombre,String descripcion,String imagen) {
    // this is a key value pair holder used by android's SQLite functions
    ContentValues values = new ContentValues();
    values.put(PRODUCTO_ID, producto_id);
    values.put(TITULO, nombre);
    values.put(CANTIDAD, cantidad);
    values.put(PRECIO, precio);
    values.put(IMPUESTO, impuesto);
    values.put(SUBTOTAL, subtotal);
    values.put(DESCRIPCION, descripcion);
    values.put(IMAGEN, imagen);


    // ask the database object to insert the new data
    try {
        db.insert(TABLE_NAME,  nullColumnHack: null, values);
    } catch (Exception e) {
        Log.e( tag: "DB ERROR", e.toString());
        e.printStackTrace();
    }
}
```

Para almacenar la información de la aplicación y del usuario se utilizan las preferencias compartidas del teléfono; con el fin de controlar las sesiones y las peticiones al servidor. El método utilizado para compartir las preferencias de usuario se puede evidenciar en la figura 35.

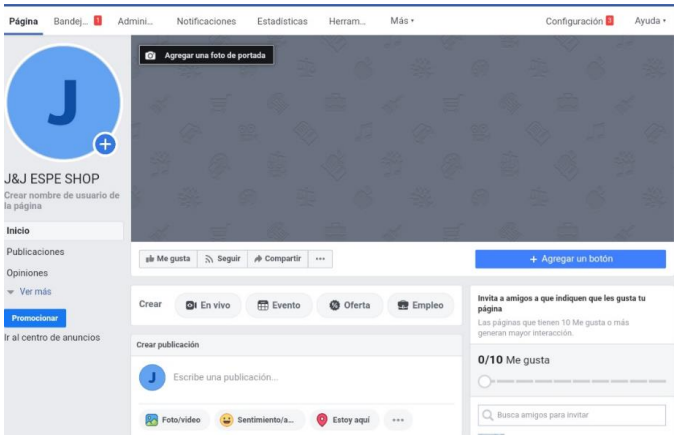
Figura 35
Uso de preferencias compartidas en la aplicación móvil

```
SharedPreferences sp = getSharedPreferences( name:                    , Context.MODE_PRIVATE);
sp.edit().putString( s:                User_ID).commit();
```

Diseño de la aplicación integrada en Facebook Messenger

Para la aplicación integrada se creó una página en Facebook que permite establecer la comunicación entre el usuario y el aplicativo, para esta página se seleccionó la categoría de negocio permitiendo a los usuarios enviar mensajes por medio del chat de Facebook Messenger. La figura 36 muestra la página creada para la aplicación en la plataforma de Facebook.

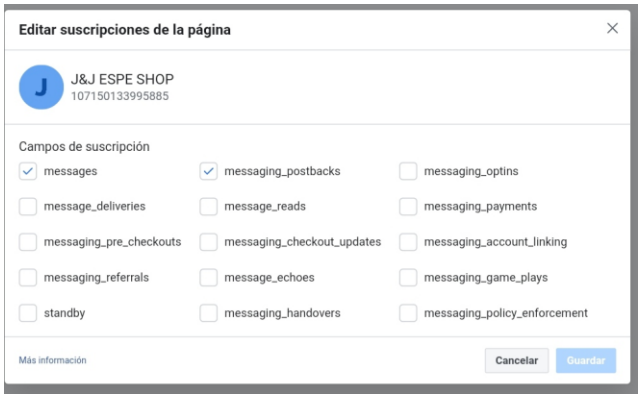
Figura 36
Página de Facebook



Para la comunicación con la “API” de Facebook Messenger se debe registrar una cuenta en la plataforma de Facebook Developers, posteriormente registrar la página y establecer el permiso para la “API” de Messenger. Posteriormente se debe seleccionar el evento “webhook” y establecer los eventos de mensajes y “Postback” como se muestra en la figura 37.

Figura 37

Registro eventos "webhook" para página de Facebook.



Un "webhook" es un método que permite alterar el comportamiento de una página o aplicación web mediante devoluciones de llamada generalmente de tipo HTTP POST. Estas devoluciones de llamada pueden ser mantenidas, modificadas y gestionadas por terceros usuarios y desarrolladores que no necesariamente deben estar afiliados al sitio web de origen de la solicitud (Krishnan, Varun, & Venkatasubramanian, 2018). Posteriormente se deben establecer 2 credenciales, la primera generada por la plataforma de Facebook Developers que permite identificar a la página (FB_PAGE_ACCESS_TOKEN), la segunda establecida por los investigadores y definida tanto en la "API" como en la aplicación web (FB_VERIFY_TOKEN). Una vez finalizada la configuración de las credenciales se crea una relación entre la página y los eventos "webhook" como se evidencia en la figura 38.

Figura 38

Configuración de credenciales en plataforma Facebook Developers.

Una vez configuradas las opciones en la plataforma de Facebook Developers se procedió a realizar la comunicación entre el servidor del chatbot y la plataforma por medio de 2 métodos, el primero **Figura 39** permite la autorización y autenticación entre la página y el chatbot y el segundo analiza el tipo de mensajes y permite procesar las respuestas **Figura 40**.

Figura 39

Método para autorización y autenticación entre página y chatbot.

```
// for Facebook verification
app.get('/webhook/', function (req, res) {
    if (req.query['hub.verify_token'] === vtoken) {
        res.send(req.query['hub.challenge'])
    }
    res.send('Sin acceso, el token de verificación no corresponde')
})
```

Figura 40

Método para análisis y generación de respuestas.

```
app.post('/webhook/', function (req, res) {
    let messaging_events = req.body.entry[0].messaging
    for (let i = 0; i < messaging_events.length; i++) {
        let event = req.body.entry[0].messaging[i]
        let sender = event.sender.id
        if (event.message) {
            if (event.message.quick_reply) {

            } else if (event.message.text) {

            }
        } else if (event.postback) {

        }
    }
    res.sendStatus(200);
});
```

Cuando el usuario envía un mensaje a la página, la “API” de Messenger envía una petición de tipo POST al evento del “webhook”. El mensaje es obtenido por el servidor del chatbot como un mensaje de tipo JSON, es analizado, procesado y finalmente se genera una respuesta de tipo JSON que es enviada hacia la “API” de Messenger, que a su vez obtiene la respuesta del servidor del chatbot, analiza el contenido y entrega una respuesta al chat del usuario final mostrando el contenido deseado. El contenido entregado corresponde a una de las categorías establecidas en las plantillas de Facebook Messenger. Las plantillas se pueden encontrar en el siguiente enlace: https://developers.facebook.com/docs/messenger-platform/send-messages/templates/?locale=es_ES. La figura 41 muestra una plantilla de tipo genérica en modo de carrusel para la selección de categorías, la figura 42 se puede observar una respuesta simple en modo de texto, la figura 43 evidencia una respuesta de tipo recibo y la figura 44 muestra el detalle de la orden realizada.

Figura 41

Respuesta Genérica tipo Carrusel.

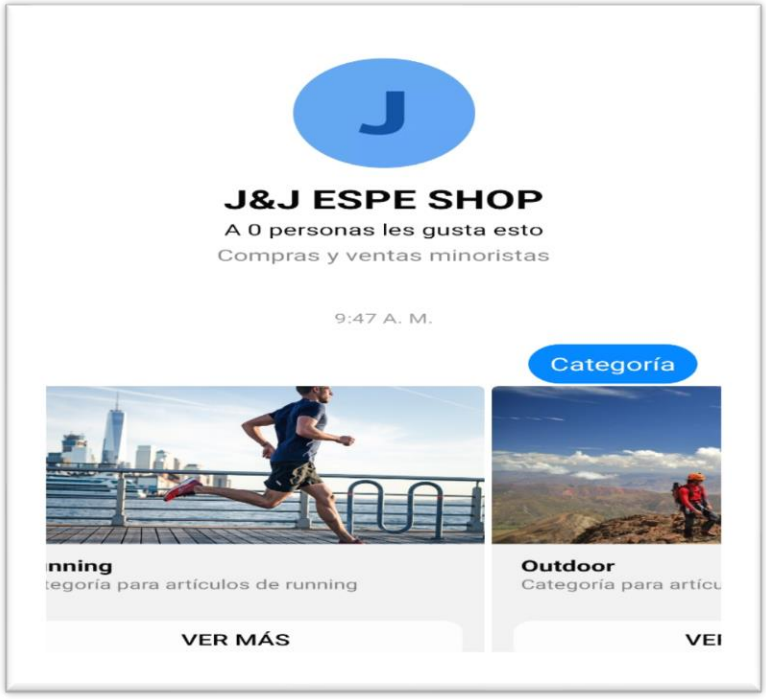


Figura 42

Respuesta tipo texto.

Figura 43
Respuesta tipo recibo.

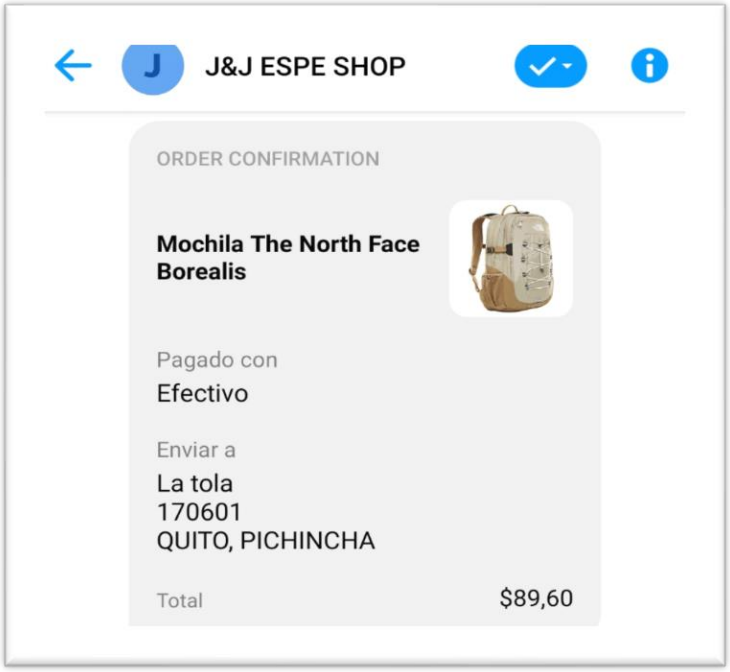
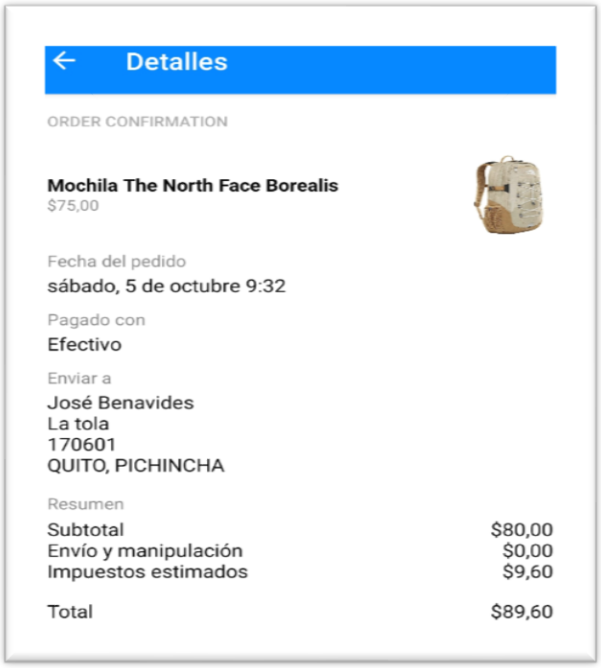


Figura 44
Detalles de respuesta tipo recibo.



Análisis de resultados

Lista de servicios y resultados

La tabla 5 muestra el detalle de los servicios que se pueden realizar en las aplicaciones desarrolladas y el resultado esperado al ejecutar cada servicio.

Tabla 5
Lista de servicios y resultados esperados

Servicio	Resultado Esperado
Iniciar Sesión	El usuario puede realizar proceso de iniciar sesión
Categoría	El usuario puede visualizar las categorías
Subcategoría	El usuario puede visualizar las subcategorías
Productos	El usuario puede visualizar los productos
Modelos	El usuario puede visualizar los modelos por producto
Selección de producto	El usuario puede elegir un producto y establecer la cantidad
Carrito	El usuario puede visualizar el carrito, añadir o eliminar más productos.
Cliente	El usuario puede ingresar los datos del cliente para la factura.
Pedido	El usuario puede generar un pedido.
Facturas	El usuario puede verificar la factura actual o un historial de sus facturas.

Selección de métricas

Con la finalidad de verificar si es posible reemplazar aplicaciones móviles transaccionales por medio de aplicaciones integradas en Facebook Messenger, se establecieron 5 métricas:

- **Usabilidad**: que se refiere a la facilidad de uso que tiene la aplicación por parte del usuario es decir que la aplicación sea “amigable con el usuario”.

- **Consumo de recursos**: esta métrica permitió observar en qué dispositivos se puede utilizar las aplicaciones.
- **Tiempo de desarrollo**: permite estimar la salida a producción de una aplicación y permite hacerse una idea de los costos de desarrollo.
- **Facilidad de mantenimiento**: permite verificar que tan fácil resulta para el desarrollador corregir bugs, actualizar, o añadir características a la aplicación.
- **Portabilidad**: define las capacidades que posee la aplicación para ejecutarse en una o varias plataformas.

Lista de parámetros

La tabla 6 presenta las métricas seleccionadas para la evaluación de las aplicaciones y la lista de parámetros a evaluar por cada una de las métricas.

Tabla 6
Listado de parámetros por métrica de análisis

Usabilidad	**Consumo de recursos**	**Tiempo de desarrollo**	**Facilidad de mantenimiento**	**Portabilidad**
Porcentaje de éxito al realizar un pedido	Almacenamiento	Desarrollo de servicios web	Tiempo de baja	Multiplataforma
Eficiencia en encontrar productos	Procesamiento	Desarrollo del “Back End” de la aplicación	Facilidad de procedimiento	Soporta diferentes versiones del SO
Satisfacción del usuario	Consumo Datos	Desarrollo del “Front End” de la aplicación	Entrega de Actualizaciones	
	Consumo memoria RAM			

Usabilidad	Consumo de recursos	Tiempo de desarrollo	Facilidad de mantenimiento	Portabilidad
	Consumo Memoria Caché			

Selección de factores a estudiar

La tabla 7 presenta el listado de los factores a evaluar en cada una de las aplicaciones, la descripción del factor y el usuario que es el encargado de probar dicho factor.

Tabla 7
Listado de factores a estudiar

Factor	Descripción del factor	Sujeto QA
Porcentaje de éxito al realizar un pedido	Permite evaluar la cantidad de personas que fueron capaces de realizar un pedido en ambas aplicaciones.	Usuario
Eficiencia en encontrar productos	Permite evaluar si los usuarios encontraron un producto al utilizar la función de búsqueda.	Usuario
Satisfacción del usuario	Permite conocer el nivel de satisfacción del usuario utilizando las aplicaciones.	Usuario
Almacenamiento	Permite al investigador conocer la cantidad de almacenamiento requerida para instalar los aplicativos.	Usuario
Procesamiento	Permite al investigador conocer la cantidad de procesamiento necesaria para utilizar el aplicativo.	Usuario
Multiplataforma	Se refiere a la capacidad del aplicativo de ejecutarse en una o varias plataformas.	Usuario
Desarrollo de servicios web	Se refiere al tiempo en horas que tomó realizar los servicios web.	Investigador
Desarrollo del “Back End” de la aplicación	Se refiere al tiempo en horas que tomó realizar el “back end” de cada uno de los aplicativos.	Investigador
Desarrollo del “Front End” de la aplicación	Se refiere al tiempo en horas que tomó realizar el “front end” de cada uno de los aplicativos.	Investigador

Factor	Descripción del factor	Sujeto QA
Tiempo de baja	Se refiere al tiempo que pasará la aplicación sin funcionar hasta desplegar una corrección.	Investigador
Facilidad de procedimiento	Se refiere a que tan complicado resulta al investigador encontrar un fallo y corregirlo.	Investigador
Entrega de Actualizaciones	Se refiere a la facilidad con la que el investigador puede entregar actualizaciones al usuario.	Investigador

Seleccionar la carga de trabajo (usuarios y peticiones)

La tabla 8 muestra el factor y la carga de trabajo asignado a dicho factor, para el desarrollo de la investigación se ejecutó de manera simultánea a 20 usuarios que evaluaron cada una de las aplicaciones para conocer sobre la usabilidad de las mismas. Además, ambos investigadores realizaron el análisis de las aplicaciones en la parte técnica para medir los factores correspondientes a desarrollo y consumo de recursos.

Tabla 8
Listado de carga de trabajo

Factor	Usuarios	Sujeto QA
Porcentaje de éxito al realizar un pedido	20	Usuario
Eficiencia en encontrar productos	20	Usuario
Satisfacción del usuario	20	Usuario
Almacenamiento adicional	20	Usuario
Procesamiento (Consumo de memoria durante ejecución)	20	Usuario
Capacidad de operar en multiplataforma	20	Usuario
Consumo de datos	10	Dispositivos de prueba
Consumo de almacenamiento	10	Dispositivos de prueba
Consumo de memoria RAM	10	Dispositivos de prueba

Factor	Usuarios	Sujeto QA
Consumo de memoria Caché	10	Dispositivos de prueba
Desarrollo del “Back End” de la aplicación	2	Investigador
Desarrollo del “Front End” de la aplicación	2	Investigador
Tiempo de baja en mantenimiento	2	Investigador
Facilidad de procedimiento de mantenimiento	2	Investigador
Entrega de Actualizaciones	2	Investigador

Experimentos del diseño

Con la finalidad de determinar si la aplicación integrada en Facebook Messenger es capaz de sustituir a la aplicación móvil se establecieron 3 parámetros de evaluación, correspondientes a la usabilidad, el consumo de recursos y el tiempo de desarrollo de las aplicaciones. Para cada uno de los parámetros se asignaron las métricas que se consideró pertinente como se muestra en la tabla 9.

Tabla 9
Listado de carga de trabajo por parámetros de evaluación

Usabilidad	Consumo de recursos	Tiempo de desarrollo
Porcentaje de éxito al realizar un pedido	Almacenamiento adicional	Desarrollo del “Back End” de la aplicación
Eficiencia en encontrar productos	Procesamiento (Consumo de memoria durante ejecución)	Desarrollo del “Front End” de la aplicación
Satisfacción del usuario	Consumo datos	Tiempo de baja en mantenimiento
Compatibilidad con versiones de sistema operativo	Consumo memoria RAM	Facilidad de procedimiento de mantenimiento
Capacidad de operar en multiplataforma	Consumo memoria Cache	

Además, para poder medir los parámetros establecidos se decidió establecer preguntas de experimentación que permitirán determinar la veracidad de los resultados como se muestra en la tabla 10.

Tabla 10
Preguntas de experimentación por parámetro de evaluación

Parámetro de evaluación	Pregunta de experimentación
Usabilidad	¿La aplicación integrada en Facebook Messenger permite realizar los mismos procedimientos incluidos en la aplicación móvil? ¿Los usuarios pueden acceder a las aplicaciones desde sus diferentes dispositivos?
Consumo de recursos	¿Fue necesario el uso de almacenamiento adicional para utilizar las aplicaciones? ¿Cuál de las aplicaciones consume más memoria para ser ejecutada?
Tiempo de desarrollo	¿Qué aplicación presenta un proceso más óptimo en su desarrollo? ¿Qué aplicación presenta un mejor procedimiento de entrega y mantenimiento?

Adicional, se realizaron pruebas técnicas en diferentes dispositivos donde se evaluó los parámetros de consumo, como se muestra en la tabla 11.

Tabla 11
Parámetros de evaluación pruebas técnicas

Parámetro de evaluación	Forma de experimentación
Consumo de datos	Se comparó la variación en el consumo de datos entre las aplicaciones antes y después de realizar la compra.
Consumo de almacenamiento	Se comparó la variación en el consumo de almacenamiento entre las aplicaciones antes y después de realizar la compra.
Consumo de memoria RAM	Se comparó la variación en el consumo de memoria RAM entre las aplicaciones antes y después de realizar la compra.
Consumo Memoria Caché	Se comparó la variación en el consumo de memoria Caché entre las aplicaciones antes y después de realizar la compra.

Analizar e interpretar los datos

Resultados basados en experiencia de los usuarios

Para el análisis de los resultados se analizó la información recolectada mediante el formato de cuestionarios Anexo 3, los cuales fueron entregados a 20 usuarios que evaluaron las aplicaciones.

Tabla 12
Resultado de usuarios que prueban la aplicación

Aplicación	SI	NO
Aplicación Móvil Nativa	20	0
Aplicación Integrada FB MSN	20	0

Figura 45

Número de usuarios que realizan las pruebas.

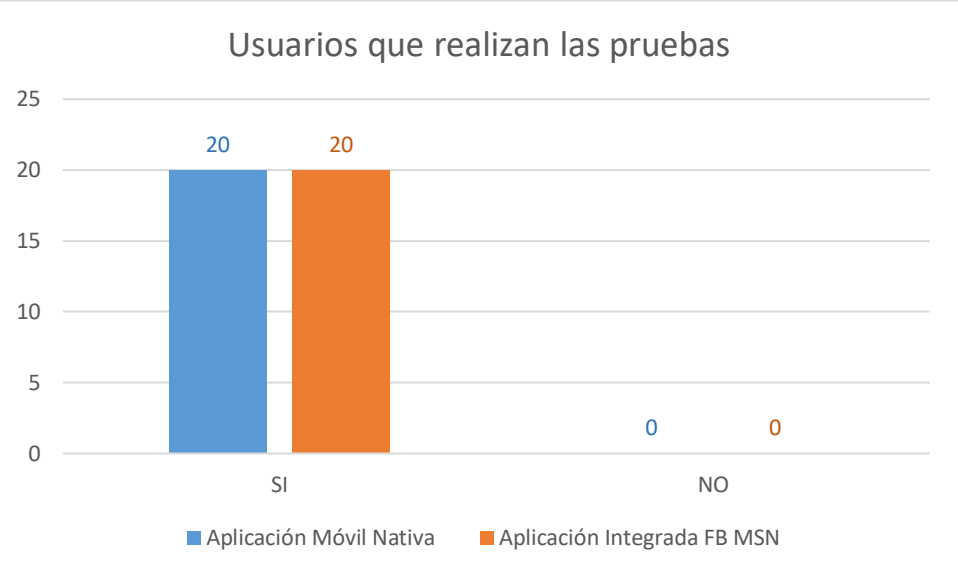


Tabla 13

Resultado cumplimiento de funcionalidades de las aplicaciones

Proceso	SI	NO
Iniciar sesión	20	0
Visualizar categorías	20	0
Visualizar subcategorías	20	0
Visualizar productos	20	0
Visualizar modelos de productos	20	0
Selección de productos	20	0
Visualizar Carrito	20	0
Establecer Cliente	20	0
Realizar Pedido	20	0
Visualizar Facturas	20	0

Figura 46

Resultado cumplimiento de funcionalidades de aplicaciones.

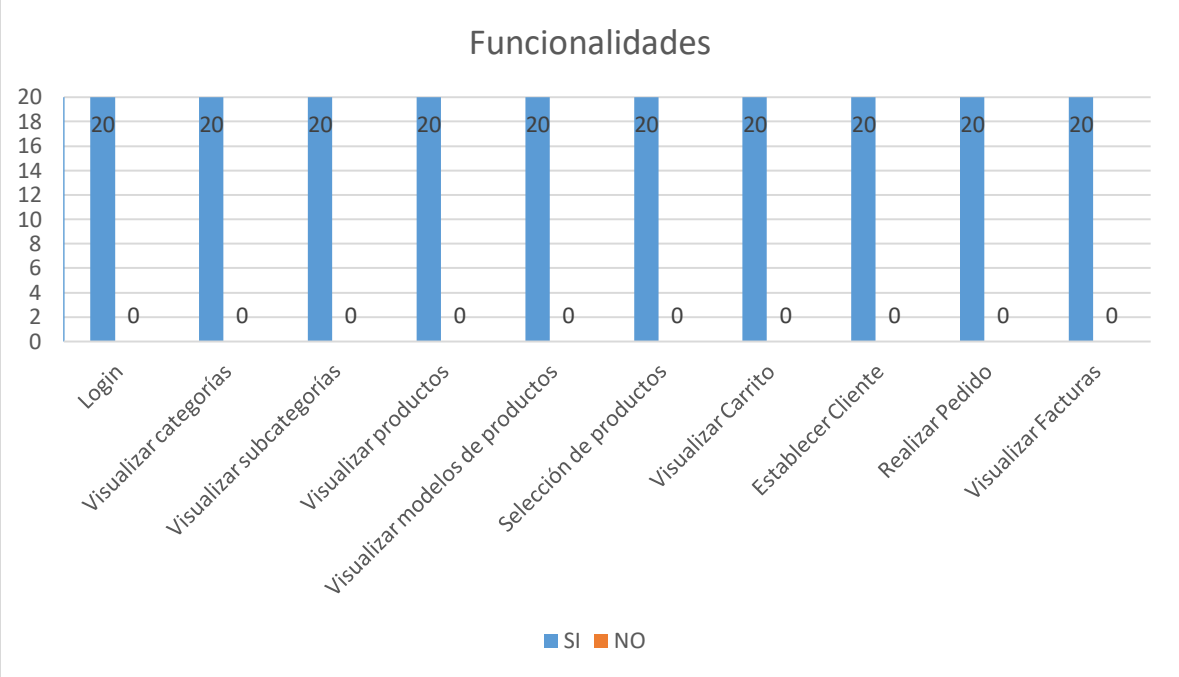


Tabla 14

Resultado aceptación de las aplicaciones

Aplicación	Calificación				
	1	2	3	4	5
Aplicación Móvil Nativa	2	4	6	8	0
Aplicación Integrada FB MSN	0	0	0	7	13

Figura 47

Resultado aceptación de las aplicaciones.

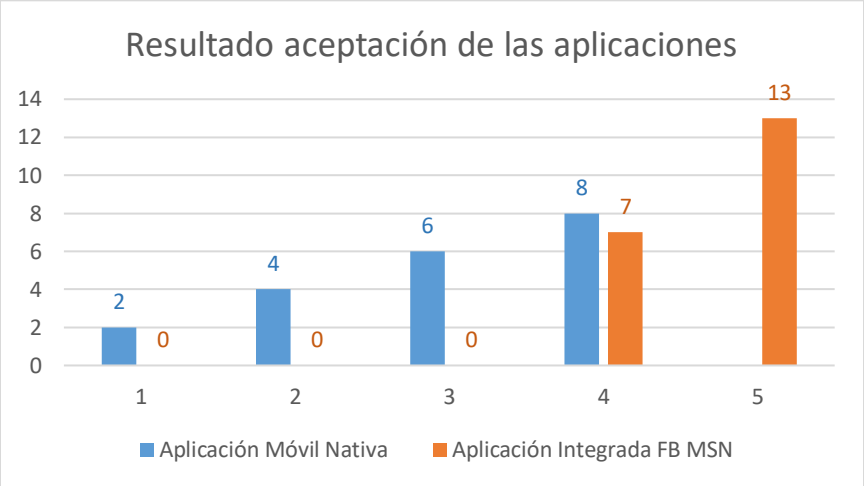


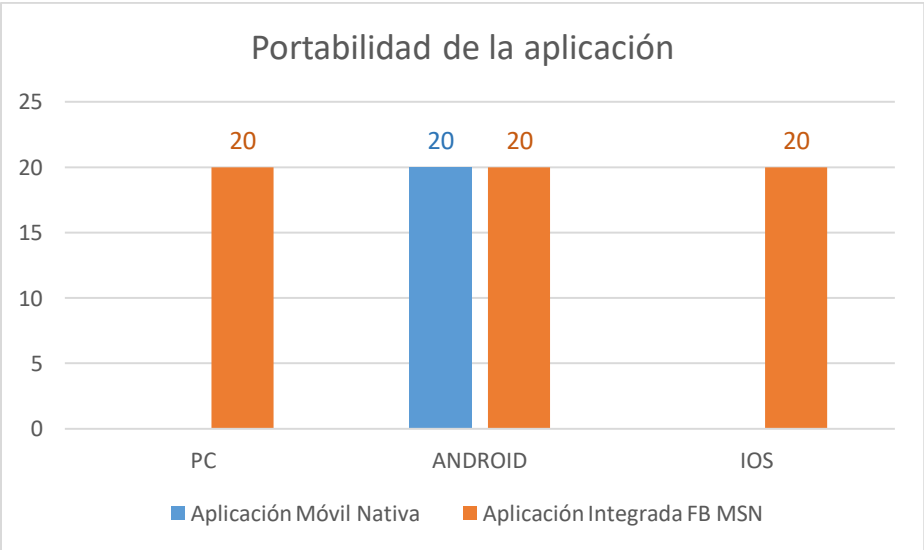
Tabla 15

Resultado de portabilidad de aplicaciones

Aplicación	PC	ANDROID	IOS
Aplicación Móvil Nativa		20	
Aplicación Integrada FB MSN	20	20	20

Figura 48

Resultado de portabilidad de aplicaciones.



CONSUMO DE RECURSOS

Tabla 16

Resultado consumo de almacenamiento adicional de las aplicaciones

Aplicación	SI	NO
Aplicación Móvil Nativa	15	5
Aplicación Integrada FB MSN	2	18

Figura 49

Resultado consumo de almacenamiento adicional de las aplicaciones.

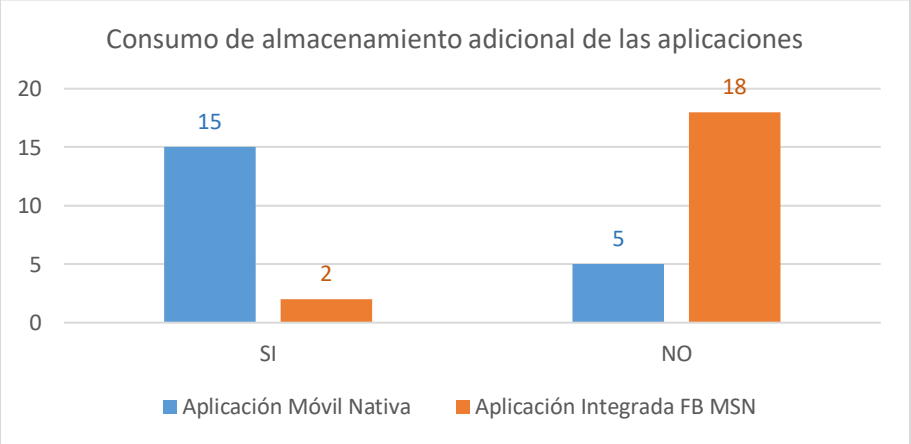
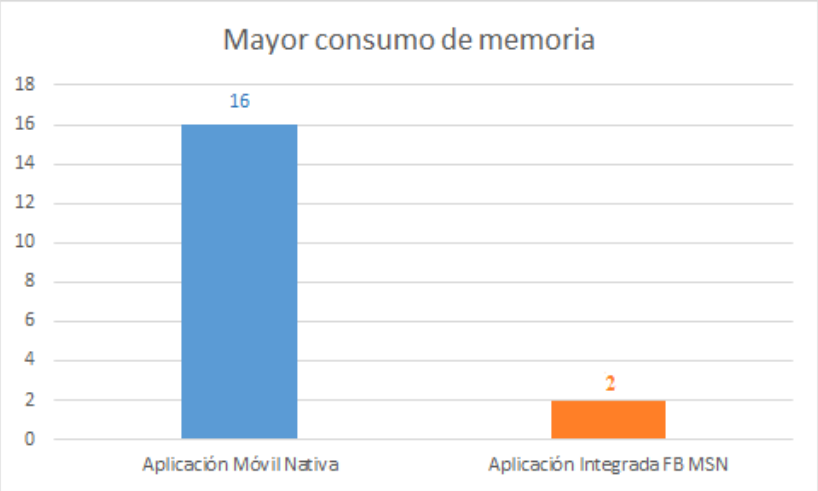


Tabla 17

Resultado mayor consumo de memoria de las aplicaciones

Aplicación	Mayor consumo de memoria
Aplicación Móvil Nativa	16
Aplicación Integrada FB MSN	2

Figura 50
Resultado mayor consumo de memoria de las aplicaciones.



Resultados de análisis técnicos

Para el análisis técnico se utilizaron 10 dispositivos Android, de gama media y baja. Se evaluó los parámetros al momento de realizar la descarga de las aplicaciones y al momento de realizar el proceso de compra en cada una de las aplicaciones.

Es importante destacar que el consumo de datos al momento de la descarga no aplica para la aplicación integrada en Facebook Messenger, debido a que la "API" como tal no requiere descarga y puede ser utilizada en un navegador o en la aplicación de Facebook Messenger tal como se muestra en la tabla 18.

Tabla 18
Consumo de datos en las aplicaciones

# Dispositivo	Tipo de Dispositivo	Momento de evaluación	App Android Nativa	Aplicación Integrada FB MSG
1	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	1.2 Mb	1.57 Mb

# Dispositivo	Tipo de Dispositivo	Momento de evaluación	App Android Nativa	Aplicación Integrada FB MSG
		Total	**4.8 Mb**	**1.57 Mb**
2	Android Gama baja	Al descargar	3.6 Mb	N/A
		Al realizar una compra	2.95 Mb	2.22 Mb
		Total	**6.55 Mb**	**2.22 Mb**
3	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	3.32 Mb	5.55 Mb
		Total	**6.92 Mb**	**5.55 Mb**
4	Android Gama baja	Al descargar	3.6 Mb	N/A
		Al realizar una compra	2.25 Mb	1.32 Mb
		Total	**5.85 Mb**	**1.32 Mb**
5	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	1.06 Mb	1.49 Mb
		Total	**4.66 Mb**	**1.49 Mb**
6	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	2.57 Mb	12.9 Mb
		Total	**6.17 Mb**	**12.9 Mb**
7	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	1.9 Mb	13.61 Mb
		Total	**5.5 Mb**	**13.61Mb**
8	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	1.8 Mb	3.35 Mb
		Total	**5.4 Mb**	**3.35 Mb**
9	Android Gama baja	Al descargar	3.6 Mb	N/A
		Al realizar una compra	2.64 Mb	2.88 Mb

# Dispositivo	Tipo de Dispositivo	Momento de evaluación	App Android Nativa	Aplicación Integrada FB MSG
		Total	**Mb**	**Mb**
10	Android Gama media	Al descargar	3.6 Mb	N/A
		Al realizar una compra	2.3 Mb	0.6 Mb
		Total	**Mb**	**Mb**

Figura 51
Resultado total consumo de datos por aplicación y dispositivo.

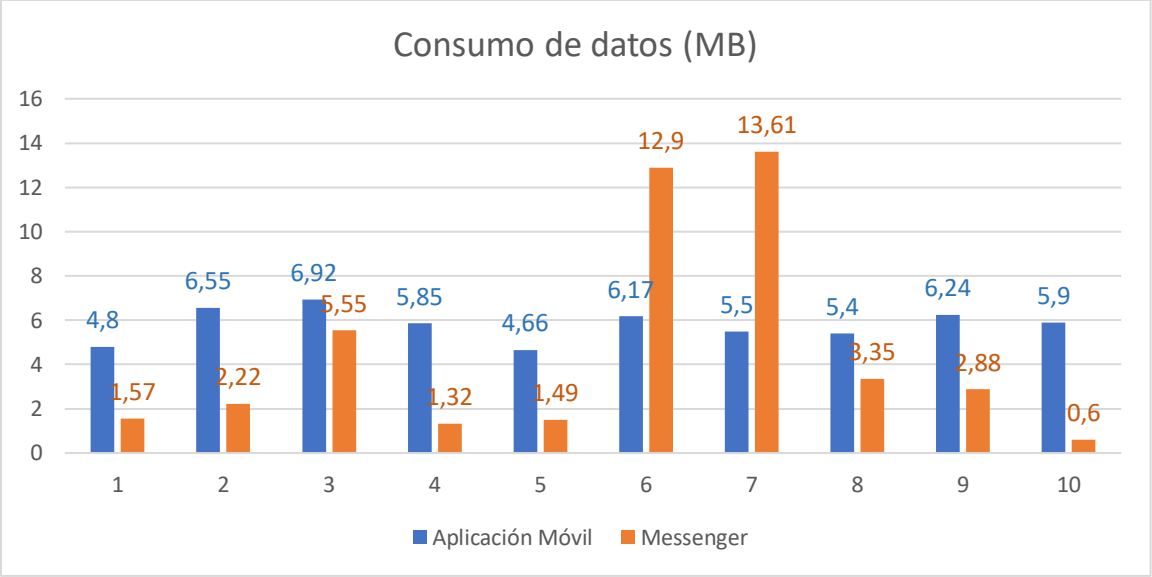


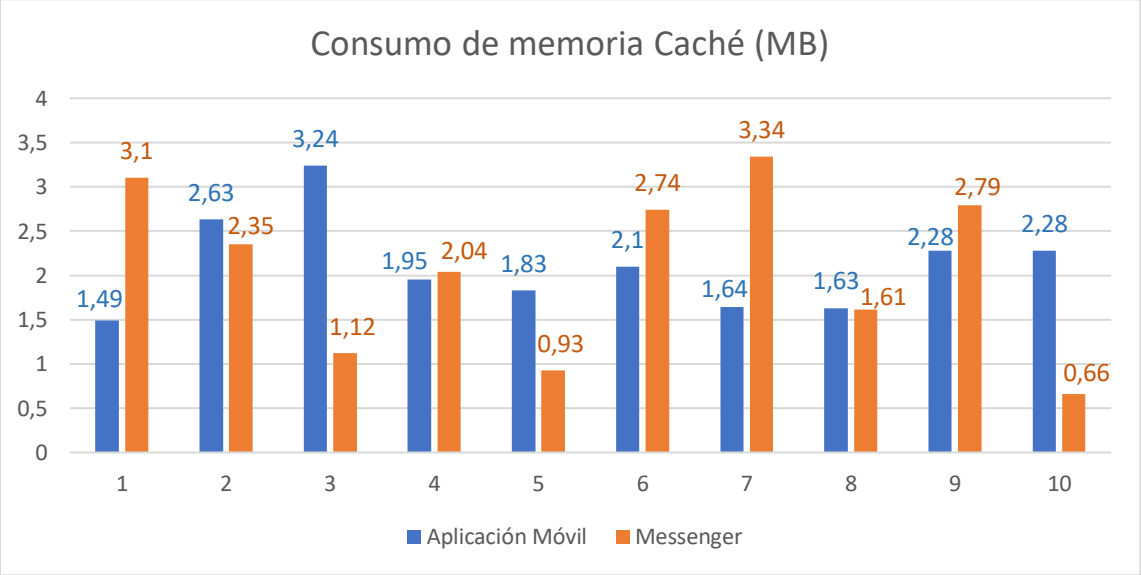
Tabla 19
Consumo de memoria caché

# Dispositivo	Tipo de Dispositivo	App Android Nativa	Aplicación Integrada FB MSG
1	Android Gama media	1.49 Mb	3.1 Mb
2	Android Gama baja	2.63 Mb	2.35 Mb
3	Android Gama media	3.24 Mb	1.12 Mb

# Dispositivo	Tipo de Dispositivo	App Android Nativa	Aplicación Integrada FB MSG
4	Android Gama baja	1.95 Mb	2.04 Mb
5	Android Gama media	1.83 Mb	0.93 Mb
6	Android Gama media	2.1 Mb	2.74 Mb
7	Android Gama media	1.64 Mb	3.34 Mb
8	Android Gama media	1.63 Mb	1.61 Mb
9	Android Gama baja	2.28 Mb	2.79 Mb
10	Android Gama media	2.28 Mb	0.66 Mb

Figura 52

Resultado consumo caché por dispositivos en aplicaciones evaluadas.



Para el cálculo del consumo de almacenamiento por aplicación que se muestra en la tabla 20. En el caso de la aplicación embebida en Facebook Messenger se comparó el consumo de almacenamiento antes de realizar la compra y luego de realizar la compra; mientras que en el caso de la aplicación móvil se evalúa con el total de almacenamiento existente únicamente

después de realizar la compra. Esto se debe a que la aplicación de Messenger o el un navegador se encuentran ya instalados y no es necesario almacenar la información para su funcionamiento.

Tabla 20
Consumo de almacenamiento interno

# Dispositivo	Tipo de Dispositivo	App Android Nativa	Aplicación Integrada FB MSG
1	Android Gama media	9.35 Mb	67.43 Mb
2	Android Gama baja	9.75 Mb	6 Mb
3	Android Gama media	10.37 Mb	14 Mb
4	Android Gama baja	13.6 Mb	11 Mb
5	Android Gama media	5.8 Mb	5 Mb
6	Android Gama media	7.29 Mb	14 Mb
7	Android Gama media	21.67 Mb	68.61 Mb
8	Android Gama media	8.81 Mb	3 Mb
9	Android Gama baja	5.95 Mb	5 Mb
10	Android Gama media	10.23 Mb	1 Mb

Figura 53
Resultado consumo almacenamiento interno.

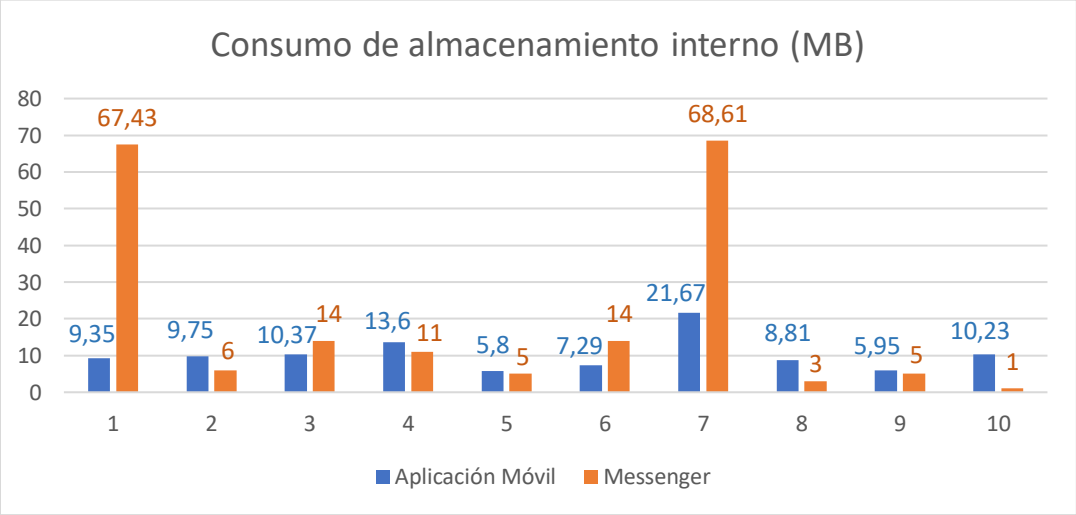
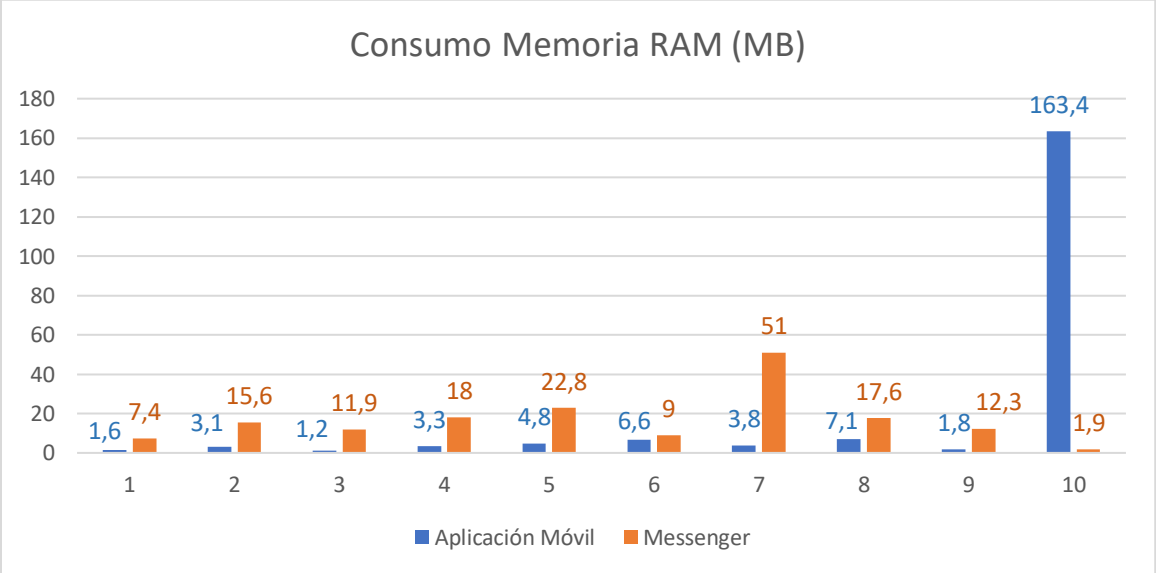


Tabla 21
Consumo de memoria RAM

# Dispositivo	Tipo de Dispositivo	App Android Nativa	Aplicación Integrada FB MSG
1	Android Gama media	1.6 Mb	7.4 Mb
2	Android Gama baja	3.1 Mb	15.6 Mb
3	Android Gama media	1.2 Mb	11.9 Mb
4	Android Gama baja	3.3 Mb	18 Mb
5	Android Gama media	4.8 Mb	22.8 Mb
6	Android Gama media	6.6 Mb	9 Mb
7	Android Gama media	3.8 Mb	51 Mb
8	Android Gama media	7.1 Mb	17.6 Mb
9	Android Gama baja	1.8 Mb	12.3 Mb
10	Android Gama media	163.4 Mb	1.9 Mb

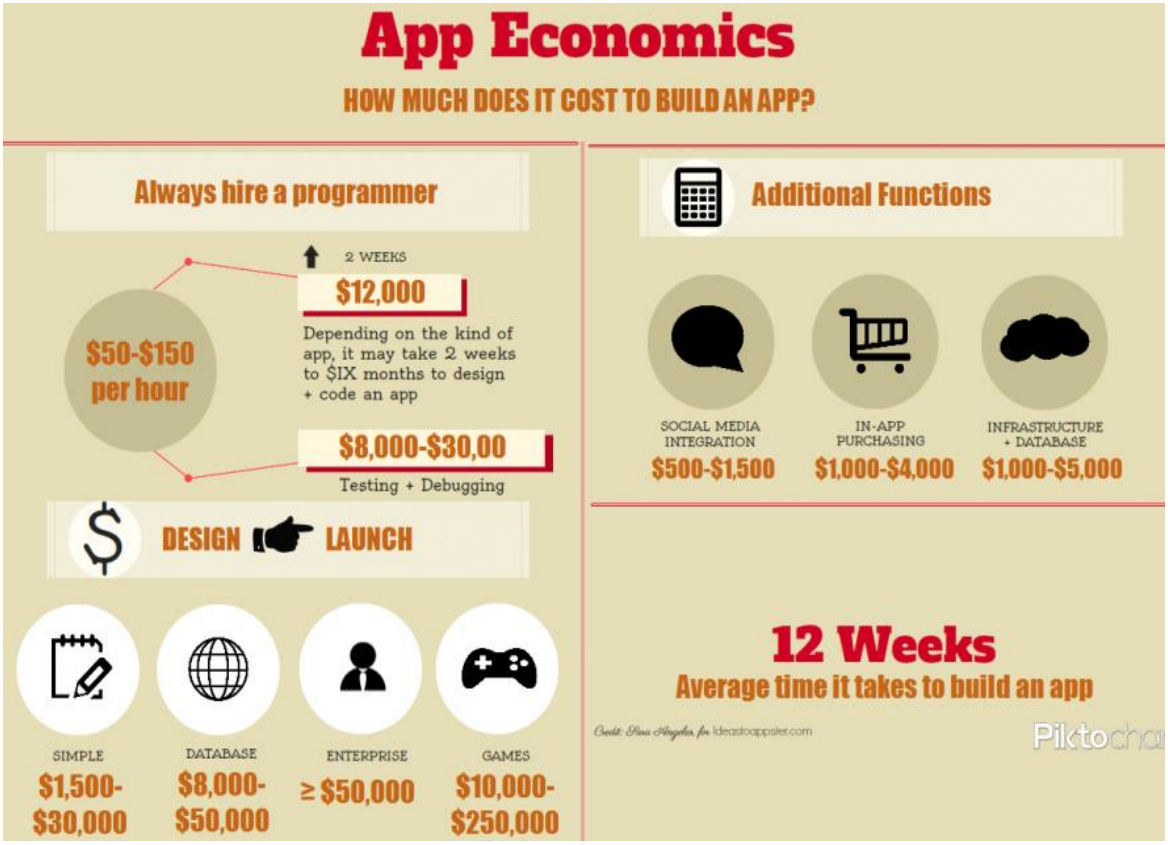
Figura 54
Resultado consumo memoria RAM.



Tiempo de desarrollo

¿Cuál fue el tiempo de desarrollo estimado en horas requerido para realizar las aplicaciones y cuál es el costo aproximado del desarrollo de cada aplicación? Estados Unidos, el precio de una aplicación básica se encuentra entre los $1500 y los $5000 dólares, para una aplicación decente el costo se encuentra por encima de los $10000 dólares y una aplicación de calidad que brinde valor a los usuarios se encuentra entre los $30000 y $150000 dólares (Angeles, 2012).

Figura 55
Resumen costo desarrollo de aplicaciones móviles.



Nota: Tomado de Ángeles (Angeles, 2012)

Sin embargo, según el portal 2000Carreras.com, dedicado a investigar el salario promedio de un empleado por carrera, se establece que el promedio del salario mensual de un programador ecuatoriano en 2019 ronda los $1300 dólares, siguiendo esta cifra podemos determinar que el salario de un programador es aproximadamente $8 por hora (2000carreras, 2019).

A continuación, se detalla el tiempo del desarrollo de los servicios web que comparten las aplicaciones, así como el tiempo de desarrollo de cada una de ellas.

Tabla 22
Resumen tareas y duración servicios web compartidos

Nombre de la tarea	Duración (horas)
Análisis y definición de requerimientos	8
Análisis y definición de métodos	8
Implementación servidor en nodejs	2
Implementación Router para peticiones REST	4
Implementación conexión con base de datos	2
Implementación servicios de categorías	8
Implementación servicios de subcategorías	8
Implementación servicios de productos	16
Implementación carrito de compras	8
Implementación métodos para facturación	8
Implementación métodos para respuestas rápidas	8
Implementación métodos de “Postback” (botones)	8
Pruebas	16
Correcciones	16
Total horas de desarrollo	**120**
Costo total	**$960**

Tabla 23
Resumen tareas y duración desarrollo app móvil nativa

Nombre de la tarea	Duración (horas)
Implementación métodos de autorización y autenticación	8
Pantalla de inicio de sesión	8
Implementación servicios de categorías	16
Pantalla menú de categorías	8
Implementación servicios de subcategorías	16
Pantalla menú de subcategorías	8
Implementación servicios de productos	16
Pantalla menú de productos	8
Implementación carrito de compras	16
Pantalla carrito de compras	8
Implementación métodos para facturación	24
Pantalla facturación	16
Implementación métodos para compras anteriores	16
Pantalla compras anteriores	16
Pruebas	16
Correcciones	32
Total horas de desarrollo	**232**
Costo total	**$ 1856**

Tabla 24
Resumen tareas y duración desarrollo app integrada en Facebook Messenger

Nombre de la tarea	Duración (horas)
Creación de página en Facebook	2
Implementación de permisos y configuraciones en plataforma Facebook Developer	2
Implementación servidor en Nodejs	2
Implementación Router para peticiones REST	8
Implementación métodos de autorización y autenticación	2
Implementación método “webhook”	2
Implementación servicios de categorías	8
Implementación servicios de subcategorías	8
Implementación servicios de productos	16

Nombre de la tarea	Duración (horas)
Implementación métodos para compras anteriores	8
Implementación carrito de compras	8
Implementación métodos para facturación	16
Implementación métodos para respuestas rápidas	8
Implementación métodos de “Postback” (botones)	8
Pruebas	16
Correcciones	16
Total horas de desarrollo	**130**
Costo total	**$1040**

¿Cuál es el proceso requerido para liberar, actualizar y dar mantenimiento a las aplicaciones?

Una vez terminado el proceso de desarrollo para liberar la aplicación móvil nativa es necesario generar el archivo .apk, el cual contiene el paquete de librerías y código necesario para instalar la aplicación móvil en el dispositivo de Android. Cuando se requiere actualizar y dar mantenimiento a la aplicación es necesario generar un nuevo archivo .apk que contiene las nuevas funcionalidades de la actualización o el mantenimiento de la aplicación, el usuario debe ejecutar el archivo y el paquete de instalación se encarga de actualizar la aplicación.

En el caso de la aplicación integrada a Facebook Messenger, para liberar la aplicación, el desarrollador debe activar los eventos de “webhook” en la plataforma de Facebook Developer, configurando la ruta del “webhook” del servidor chatbot. Una vez realizado este proceso la aplicación es liberada y puede ser utilizada por los usuarios al enviar mensajes hacia la página de Facebook de la aplicación. Cuando es necesario realizar actualizaciones, mantenimientos y nuevas funcionalidades, el desarrollador debe realizar dichos procedimientos en el servidor del chatbot y una vez terminado este proceso solo debe subir los cambios realizados al servidor. El

usuario no debe realizar ningún tipo de acción adicional lo que permite que este proceso sea casi imperceptible para él.

Resultados

Usabilidad

¿La aplicación integrada en Facebook Messenger permite realizar los mismos procedimientos incluidos en la aplicación móvil?

Una vez analizados los resultados obtenidos por los usuarios podemos observar que el 100% de los usuarios entrevistados pudieron realizar las funcionalidades implementadas en las aplicaciones por lo que podemos concluir que la aplicación integrada en Facebook Messenger permite realizar los mismos procedimientos incluidos en la aplicación móvil nativa.

¿Los usuarios pueden acceder a las aplicaciones desde sus diferentes dispositivos?

Las pruebas fueron realizadas a 20 estudiantes de la Universidad de las Fuerzas Armadas ESPE, al evaluar las aplicaciones se pudo observar que 18 usuarios pudieron probar la aplicación móvil transaccional en sus dispositivos, 2 de los entrevistados no pudieron realizar las pruebas debido a que sus dispositivos tenían una plataforma distinta a Android. Sin embargo, para realizar las pruebas se entregó dispositivos a los participantes que no los disponían para que todos pudieron probar y evaluar las aplicaciones. Cabe destacar que el problema se generó debido a la incompatibilidad de los dispositivos de los 2 usuarios con la aplicación nativa mas no hubo problemas al realizar las pruebas de la aplicación integrada en Facebook Messenger.

Consumo de recursos

¿Fue necesario el uso de almacenamiento adicional para utilizar las aplicaciones?

En su gran mayoría los usuarios entrevistados acordaron que no se necesitó utilizar almacenamiento adicional para utilizar la aplicación móvil integrada en Facebook Messenger, debido a que esta pudo ser accedida mediante las aplicaciones de Facebook Messenger que ya tenían instaladas en sus dispositivos o mediante los navegadores web. Sin embargo, para utilizar la aplicación móvil transaccional fue necesario descargar la aplicación e instalarla en sus dispositivos lo que denota tiempo para adquirir la aplicación y mayor consumo de almacenamiento interno de los dispositivos.

¿Cuál de las aplicaciones consume más memoria para ser ejecutada?

El 75% de los usuarios entrevistados (15 personas) acordaron que la aplicación móvil nativa consume más memoria al ser utilizada, debido a que ejecuta procesos internos en cada uno de los dispositivos. Si bien es cierto que tanto los navegadores web como la aplicación de Facebook Messenger en la que fue probada la aplicación integrada, poseen características y funcionalidades adicionales que son ajenas al procedimiento evaluado, no determinaron mayor carga en sus dispositivos al utilizar la aplicación integrada en Facebook Messenger.

Consumo de datos

El parámetro de consumo de datos determinó que la aplicación integrada en Facebook Messenger consume menos datos móviles, exceptuando los casos que fueron difíciles de determinar debido a que el usuario recibe mensajes ajenos al proceso de durante la prueba de la aplicación en Messenger.

Consumo de Memoria Caché

En los dispositivos de gama baja se pudo observar que la aplicación integrada en Facebook Messenger ocupa más espacio de memoria caché; debido a que la aplicación guarda imágenes, conversaciones y toda información nueva que le es provista en este tipo de memoria.

Consumo de almacenamiento.

Al evaluar la variación en el consumo de almacenamiento por aplicación se encontró que esta variación es mucho menor en Facebook Messenger que en la aplicación nativa. Excepto en el caso en que el dispositivo no cuente con la aplicación de Facebook Messenger y se debe utilizar un navegador.

Consumo de memoria RAM

Se encontró que la aplicación móvil nativa consume más memoria RAM en equipos de gama baja. En el caso que el dispositivo no cuenta con Facebook Messenger previamente la variación en el consumo de memoria RAM es elevado; y sobrepasa el consumo por parte de la aplicación móvil nativa. En los demás casos, el consumo de RAM es menor por parte de la aplicación nativa.

Tiempo de desarrollo

¿Qué aplicación presenta un proceso más óptimo en su desarrollo?

Al evaluar el tiempo y el costo de desarrollo de cada una de las aplicaciones, podemos concluir que el desarrollo de la aplicación integrada en Facebook Messenger es el más óptimo, debido a que no necesita un proceso para creación del “Front End” y que las interfaces de

comunicación y despliegue son proporcionadas directamente por Facebook lo que reduce considerablemente los costos de desarrollo y tiempo de despliegue.

¿Qué aplicación presenta un mejor procedimiento de entrega y mantenimiento?

Debido a que los cambios realizados en la aplicación móvil integrada en Facebook Messenger se realizan directamente sobre el servidor, quien se comunica al chat de Facebook Messenger por medio de su aplicación a través de servicios REST, no es necesario que el usuario actualice el aplicativo, lo que determina una ventaja competitiva en conceptos de distribución y mantenimiento de la aplicación integrada en Facebook Messenger sobre la aplicación móvil nativa.

Capítulo VI - Conclusiones, Recomendaciones y Trabajo Futuro

Conclusiones

Una vez concluida la investigación podemos notar que la aplicación integrada en Facebook Messenger demostró ser capaz de reemplazar las funcionalidades de la aplicación móvil, si bien es cierto los usuarios se han acostumbrado a utilizar aplicaciones móviles nativas, la facilidad de comunicación utilizando lenguaje natural y puestas rápidas que ofrece la "API" de Facebook Messenger permiten que el usuario pueda reconocer las acciones y utilizar la aplicación sin tener conocimientos previos de la misma. Además, la aplicación Integrada en Facebook Messenger fue capaz de ser utilizada en varios dispositivos de diferentes plataformas a diferencia de la aplicación móvil que solo pudo ser ejecutada en dispositivos Android.

No existe gran diferencia en el consumo de recursos debido a que las funcionalidades, las bases de datos, imágenes y demás contenidos multimedia se encuentran en un servidor web, existe una pequeña diferencia debido a que la app móvil debe ser instalada y la aplicación integrada puede ser ejecutada desde la aplicación de Facebook Messenger o el navegador web, mismo que suele venir preinstalado en el sistema operativo de los dispositivos. Sin embargo, debido a que la aplicación integrada se ejecuta por medio de la plataforma de Facebook, la cual incluye otro tipo de contenidos ajenos al proceso evaluado, mismos que dificulta la medición y aumentan la cantidad de recursos necesarios para realizar una transacción.

Ya que en la aplicación integrada en Facebook Messenger solo es necesario generar los procesos y servicios para el "Back End", el desarrollo de este tipo de aplicaciones disminuye el tiempo y la dificultad del desarrollo, así como el campo de conocimientos requeridos para tener que desarrollar e implementar las vistas y procedimientos para "Front End".

El desarrollo de una aplicación integrada en Facebook Messenger facilita la distribución de la aplicación entre las diferentes plataformas web o móviles, ya que al comunicarse mediante eventos “webhook” e integrarse a la “API” de Messenger, la empresa de Facebook es quien se encarga del mantenimiento, seguridad y actualizaciones de su propia aplicación. Por lo que no es necesario que los investigadores realicen desarrollos paralelos o multiplataforma para otros sistemas operativos como pc o dispositivos móviles Android e IOS y en general cualquier dispositivo que cuente al menos con un navegador web.

Recomendaciones

Con el fin de mantener una arquitectura adecuada en los servidores se utilizó los 5 principios de SOLID separando la responsabilidad de cada uno de los métodos y garantizando la integridad y veracidad de las respuestas, independientemente de la concurrencia de usuarios y solicitudes que se procesó el servidor.

Debido a que la investigación utilizó servicios REST y paquetes JSON para la comunicación, los servicios pueden ser creados en cualquier lenguaje que permita este tipo de comunicación por lo que el investigador tiene libertad de utilizar un lenguaje que sea de su experticia.

Trabajo futuro

Como trabajo futuro se pretende implementar inteligencia artificial en el servidor, debido a que Facebook Messenger permite integrar la plataforma de lenguaje natural Wit.ai (Facebook for developers, 2019), permitiendo mantener una conversación más natural y fluida entre el usuario y el chatbot, mejorando la toma de decisiones y el análisis de los mensajes, generando respuestas más adecuadas ya sea de manera semántica o visual.

Bibliografía

2000carreras. (2019). *¿Cuánto gana un Programador en Ecuador? Sueldo 2019.* Obtenido de 2000carreras: https://www.2000carreras.com/2017/10/cuanto-gana-un-programador-en-ecuador-sueldo.html

Alliance, O. H. (2011). Android overview. *Open Handset Alliance, 8*, 88-91.

Al-Zubaide, H., & Issa, A. A. (2011). Ontbot: Ontology based chatbot. In Innovation in Information & Communication Technology (ISIICT). *Fourth International Symposium on IEEE*, 7-12.

Android. (2019). *Conoce Android Studio, Developer Android .* Obtenido de Android: https://developer.android.com/studio/intro/?hl=es-419

Android Developers. (2019). *Android Developers.* Obtenido de Android: http://developer.android.com/guide/developing/tools/emulator.html

Android Developers. (2019). *Resumen de licencias.* Obtenido de Android Developers: https://developer.android.com/google/play/licensing/overview.html

Angeles, S. (2012). How much does it cost to make an app? An infographic. *Gelesen am 27*.

Apple Inc. (2019). *Developer Account.* Obtenido de https://developer.apple.com/account/#/welcome

Apple Inc. (2019). *Principles and Practices.* Obtenido de https://www.apple.com/ios/app-store/principles-practices/

Apple Inc. (2019). *Swift.* Obtenido de https://developer.apple.com/swift/

Apple Inc. (2019). *Swift Playgrounds.* Obtenido de https://developer.apple.com/swift-playgrounds/

Apple Inc. (2019). *What's Included.* Obtenido de https://developer.apple.com/programs/whats-included/

Apple Inc. (2019). *Xcode 10.* Obtenido de https://developer.apple.com/documentation/xcode_release_notes/xcode_10_release_notes

Báez, M., Borrego, A., Cordero, J., Cruz, L., González, M., Hernández, F., & Torralbo, P. (2012). Introducción a android. *EME Madrid*.

Biørn-Hansen, A., Majchrzak2, T. A., & Grønli1, T.-M. (2017). Progressive Web Apps: The Possible Web-native Unifier for Mobile. *Faculty of Technology, Westerdals Oslo ACT*.

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Bauer, G. (2011). Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage.

International conference on Human computer interaction with mobile devices and services, 47-56.

Bommisetty, S., Tamma, R., & Mahalik, H. (2014). Practical mobile forensics. *Packt Publishing Ltd*.

Cheng, K. M., Lin, V. J., Nijhawan, K., Westhem, A., & Bernstein, M. S. (2017). *Apps with Benefits: Using Benefits and Burdens to Predict Mobile App Usage.*

Constine, J. (2016). *Facebook launches Messenger platform with chatbots.* Obtenido de TechCrunch: https://techcrunch.com/2016/04/12/agents-on- messenger/

Davenport, P. (25 de Octubre de 2018). *Mobile App User Retention Continues To Soar in 2018.* Obtenido de Localytics: https://info.localytics.com/blog/mobile-app-user-retention-continues-to-soar-in-2018

Facebook ©. (2019). *Our History.* Obtenido de Facebook ©: https://newsroom.fb.com/company-info/

Facebook for developers. (2019). *Introducción a la plataforma de Messenger.* Obtenido de Facebook for developers: https://developers.facebook.com/docs/messenger-platform/introduction

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). Why people hate your app: Making sense of user feedback in a mobile app store.

Gavalas, D., & Economou, D. (2010). Development platforms for mobile applications: Status and trends. *IEEE software 28*, 77-86.

Goadrich, M. H., & Rogers, M. p. (2011). Smart smartphone development: iOS versus Android. *Proceedings of the 42nd ACM technical symposium on Computer science education*, 607-612.

Han, H., & Li, R. (2017). A practical compartmentation approach for the android app coexistence. *International Conference on Cryptography, Security and Privacy*, 49-55.

Heroku. (2018). *Heroku Platform.* Obtenido de Heroku : https://www.heroku.com/platform

IBM Software. (2018). *Native, web or hybrid mobile-app development.* Obtenido de Thought Leadership White Paper: http://www.computerworld.com.au/whitepaper/371126/native-web-or-hybrid-mobile-app-development

Jain, R. (1991). *Art of Computer Systems Performance Analysis: Techniques for Experimental Design, measurement, simulation and modeling.* Wyley Professional Computing.

JetBrains. (2019). *JetBrains.* Obtenido de https://www.jetbrains.com/objc/download/

Krishnan, S., Varun, P. A., & Venkatasubramanian, B. (2018). *Estados Unidos Patente nº Application No. 15/335,274.*

Malavolta, I., Procaccianti, G., Noorland, P., & Vukmirovic, P. (2017). Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps. *Information Management and Software Engineering*.

Marbellamiami. (2019). *Mejores IDE [entorno de desarrollo integrado] para Mac en 2019.* Obtenido de https://marbellamiami.com/es/reviews/4312-best-ides-integrated-development-environment-for-mac-in-2019.html

MariaDB . (2018). *About MariaDB.* Obtenido de MariaDB: https://mariadb.org/about/

McIlroy, S., Ali, N., & Hassan, A. E. (2016). Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store. *Empirical Software Engineering 21*, 1346-1370.

Morrissey, S., & Campbell, T. (2011). iOS Forensic Analysis: for iPhone, iPad, and iPod touch. *Apress*.

Mullan, P., Riess, C., & Freiling, F. (2019). Forensic source identification using JPEG image headers: The case of smartphones. *Digital Investigation 28*, 68-76.

O'Connell, C. (25 de Octubre de 2017). *24% of Users Abandon an App After One Use.* Obtenido de Localytics: http://info.localytics.com/blog/24-of-users-abandon-an-app-after-one-use

Openintents. (2019). *Openintents.* Obtenido de http://code.google.com/p/openintents/wiki/SensorSimulator

Oracle. (2019). *Conozca más sobre la tecnología Java, Oracle Corporation.* Obtenido de Java: https://www.java.com/es/about/

Perez, S. (2016). *Nearly 1 in 4 people abandon mobile apps after only one use.* Obtenido de Techcrunch: https://techcrunch.com/2016/05/31/nearly-1-in-4-people-abandon-mobile-apps-after-only-one-use

Perro, J. (2018). *Mobile Apps: What's A Good Retention Rate?* Obtenido de Localytics: https://info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate

Raphael, J. R. (2019). *Android versions: A living history from 1.0 to Q.* Obtenido de Computerworld: https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html

Shin, C., Hong, J. H., & Dey, A. K. (2012). g, J. H., & Dey, A. K. (2012, September). Understanding and prediction of mobile application usage for smart phones. *ACM Conference on Ubiquitous Computing*, 173-182.

Suh, H., Shahriaree, N., Hekler, E., & Kientz, J. (2016). Developing and Validating the User Burden Scale: A Tool for Assessing User Burden in Computing Systems.

Tuominen, J., & Virtaranta, J. (2019). Dynamic Branding in Mobile Applications. *Business Information Systems*.

Wang, H., Li, H., Li, L., Guo, Y., & Xu, G. (2018). Why are Android apps removed from Google Play?: a large-scale empirical study. *15th International Conference on Mining Software Repositories* , 231-242.

Wasserman, A. I. (2010). Software engineering issues for mobile application development. *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 397-400.

Wei, P., Lee, S. Y., Lu, H. P., Tzou, J. C., & Weng, C. I. (2015). Why do people abandon mobile social games? Using Candy Crush Saga as an example. *International Journal of Social, Education, Economics and Management Engineering Vol:9*.

Xu, C., Peak, D., & Prybutok, V. (2015). A customer value, satisfaction, and loyalty perspective of mobile application recommendations.

Zeng, P. (2016). Maintaining Social Connectedness: Hanging Out Using Facebook Messenger. *Doctoral dissertation, Miami University*.