DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN PREVIO LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA:

"ESTUDIO DE LA EFICIENCIA DE LOS FRAMEWORKS HÍBRIDOS Y NATIVOS EN EL DESARROLLO DE APLICACIONES MÓVILES, BASADO EN BENCHMARK PARA EL CONSORCIO INFORMEGA"

AUTOR:

SALGADO ESCOBAR, STALIN SEBASTIAN

DIRECTOR:

ING. SANCHO ARIAS, JOSÉ ALBERTO

SANGOLQUÍ

2018

CERTIFICADO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERIA EN SISTEMAS E INFORMATICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***ESTUDIO DE LA EFICIENCIA DE LOS FRAMEWORKS HIBRIDOS Y NATIVOS EN EL DESARROLLO DE APLICACIONES MÓVILES, BASADO EN BENCHMARK PARA EL CONSORCIO INFORMEGA***" realizado por el señor ***SALGADO ESCOBAR STALIN SEBASTIAN***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***SALGADO ESCOBAR STALIN SEBASTIAN*** para que lo sustente públicamente.

Quito, 18 de enero del 2018

Atentamente,

ING. JOSE SANCHO
Director

AUTORIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERIA EN SISTEMAS

AUTORIA DE RESPONSABILIDAD

Yo, **STALIN SEBASTIAN SALGADO ESCOBAR**, con cedula de identidad N ° 1726056300, declaro que este trabajo de titulación "**ESTUDIO DE LA EFICIENCIA DE LOS FRAMEWORKS HÍBRIDOS Y NATIVOS EN EL DESARROLLO DE APLICACIONES MÓVILES, BASADO EN BENCHMARK PARA EL CONSORCIO INFORMEGA**" ha sido desarrollado considerando los métodos de investigaciones existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada

Quito, 12 de febrero del 2018

SALGADO ESCOBAR STALIN SEBASTIAN

CI: 1726056300

AUTORIZACIÓN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERIA EN SISTEMAS

AUTORIZACIÓN

Yo, **STALIN SEBASTIAN SALGADO ESCOBAR**, autorizo a la UNIVERSIDAD DE LA FUERZAS ARMANDAS-ESPE, la publicación, en la biblioteca virtual de la Institución del trabajo **"ESTUDIO DE LA EFICIENCIA DE LOS FRAMEWORKS HÍBRIDOS Y NATIVOS EN EL DESARROLLO DE APLICACIONES MÓVILES, BASADO EN BENCHMARK PARA EL CONSORCIO INFORMEGA"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Quito, 12 de febrero del 2018

SALGADO ESCOBAR STALIN SEBASTIAN

CI: 1726056300

DEDICATORIA

Este trabajo va dedicado a mis padres Gloria y Genaro, quienes han velado por mí, por mi bienestar y educación entregando siempre mucho amor y paciencia además de su lucha constante y perseverante que siempre han sido un gran ejemplo a seguir.

A mi hermano Christian quien ha sido siempre antes que mi hermano un amigo que siempre que he necesitado siempre ha estado ahí para mí.

Stalin Sebastián Salgado Escobar

AGRADECIMIENTO

Agradezco a Dios por darme la vida y salud y permitirme culminar un objetivo en mi carrera profesional, a mis padres y hermano por su comprensión en las labores de mi educación superior.

A mi Director, Ing. José Sancho, por brindarme la ayuda necesaria en el desarrollo y culminación de mi trabajo de investigación.

A la Carrera de Ingeniería de Sistemas e Informática y a los respectivos ingenieros por el conocimiento compartido durante estos años que he permanecido en la Universidad de las Fuerzas Armadas “ESPE”.

Stalin Sebastián Salgado Escobar

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Hoy en día, el crecimiento del uso de dispositivos móviles conjuntamente con la amplia disponibilidad de Internet, ha abierto varias aristas para brindar servicios a los usuarios, por medio de aplicaciones móviles, las mismas que desde su nacimiento ha existido un método nativo de desarrollo ya sea para Android usando Java como para IOS usando Objetive C , esta característica sumado al crecimiento de las aplicaciones móviles ha motivado el nacimiento de nuevos métodos de desarrollo de aplicaciones móviles denominadas hibridas las mismas que permiten realizar un solo desarrollo para varias plataformas. Esta investigación plantea realizar un análisis comparativo de la eficiencia del comportamiento temporal de aplicaciones móviles desarrolladas para Android estructurando un modelo de calidad IQMC comparando el método nativo versus uno de los métodos híbridos en este caso Ionic Framework, planteándose un caso de estudio el cual se detalla en el Capítulo IV, lo cual permitirá evaluar la eficiencia de los dos métodos determinado cual es el más eficiente entre los dos entregando a la comunidad de desarrollo datos que les permita decidir cuál de los dos métodos utilizar.

PALABRAS CLAVE:

- **MODELO DE CALIDAD IQMC**
- **FRAMEWORK IONIC**
- **MÉTODO HÍBRIDO**
- **MÉTODO NATIVO**

ABSTRACT

Nowadays, the growth of the use of mobile devices together with the wide availability of Internet, has opened several edges to offer services to users, through mobile applications, the same ones that since its birth there has been a native method of development and whether for Android using Java or for IOS using Objetive C, this feature, added to the growth of mobile applications, has led to the birth of new mobile application development methods called hybrid, which allow a single development for several platforms. This research proposes to perform a comparative analysis of the efficiency of the temporary behavior of mobile applications developed for Android structuring an IQMC quality model comparing the native method versus one of the hybrid methods in this case Ionic Framework, considering a case study which is detailed in Chapter IV, which will allow evaluating the efficiency of the two methods, which is the most efficient between the two, providing the development community with data that allows them to decide which of the two methods to use.

KEYWORDS:

- **IQMC QUALITY MODEL**
- **IONIC FRAMEWORK**
- **HYBRID METHOD**
- **NATIVE METHOD**

CAPITULO I
INTRODUCCIÓN

1.1 ANTECEDENTES

Hoy en día, la proliferación de dispositivos móviles, junto con la amplia disponibilidad de Internet, está abriendo nuevas oportunidades de servicio en numerosas áreas. Sin embargo, el desarrollo de aplicaciones móviles resulta ser muy difícil (Escoffier & Lalanda, 2015), debido a dos ítems principales:

1.1.1 Adaptación Dinámica

La mayoría de los recursos en una aplicación móvil son dinámicos, es decir pueden quedar indisponibles sin previo aviso, debido a fallos de hardware o software, decisiones de los usuarios, escasez de energía, operaciones de mantenimiento o disponibilidad de la red (Escoffier & Lalanda, 2015). Desde el punto de vista de la ingeniería de software genera un impacto directamente al comportamiento de la aplicación y mucho más si los usuarios son empresariales en donde sus actividades ya no se limitan al acceso a los correos electrónicos corporativos o a la lectura y creación de documentos sino requieren la capacidad de acceder a los almacenes de datos corporativos, crear y consumir informes de negocios y reducir la dependencia de su presencia física en determinada empresa (Gokhale & Singh, 2014).

1.1.2 Heterogeneidad

Las infraestructuras informáticas creadas para aplicaciones móviles son heterogéneas por naturaleza (Escoffier & Lalanda, 2015); es decir el código de la aplicación debe ser ejecutable en una variedad de plataformas de hardware y software, teniendo en cuenta que el mercado de dispositivos móviles es muy fragmentado y cambiante, en donde el 94% de aplicaciones son desarrollados en Android e iOS sin dejar de lado aplicaciones para BlackBerry y Windows Phone.

Por ejemplo, iOS promueve Objective-C, mientras que Android utiliza Java y Windows 8 C #. Estas disparidades inhabilitan el desarrollo de aplicaciones multiplataforma, lo que sugiere equipos con un amplio conjunto de conocimientos especializados y un esfuerzo de desarrollo adicional para entregar una aplicación empresarial para cada una de estas plataformas (Gokhale & Singh, 2014). Sin embargo están emergiendo herramientas hibridas como Ionic, Titanium, Xamarin que permiten implementar aplicaciones nativas multiplataforma generando el código nativo desde un lenguaje pívot HTML5, JAVASCRIPT(ANGULAR JS), CSS (Escoffier & Lalanda, 2015).

Estas herramientas hibridas están creciendo progresivamente dando paso a estudios preliminares que se ha realizado mediante la extracción de 11.917 aplicaciones gratuitas y sus metadatos relacionados desde Google Play Store, analizando desde la percepción técnica y de los usuarios finales, dando como resultado lo siguiente (Malavolta, Ruberto, Soru, & Terragniz, 2015).

- Se necesitan más esfuerzos para mejorar las aplicaciones móviles híbridas cuando se trata de características específicas de plataforma de bajo nivel.
- Los desarrolladores de aplicaciones móviles híbridas se aprovechan mucho de la reutilización de código a través de dispositivos de escritorio y las bibliotecas web específicas para móviles.
- Los usuarios finales valoran las aplicaciones híbridas y nativas móviles de manera similar.

1.2 PROBLEMÁTICA

Los estudios de analítica web a gran escala muestran que los dispositivos móviles representan el 25% de todas las visitas a páginas web en Estados Unidos, frente a 10% hace dos años (StatCounter Global, 2014). Este cambio, desde el escritorio al móvil, ha llevado a una primera revolución móvil (Hale & Hanson, 2015). Prueba de esta revolución móvil es basarnos en el mercado de aplicaciones para móviles, donde ahora cuenta con más de dos millones de aplicaciones descargadas, miles de millones de veces al año en varias tiendas de aplicaciones dedicadas (Google Play Store y Apple App Store como dominadores de mercado claros) (Lella & Lipsman, 2014). Desde la perspectiva de desarrollo de software nativo, se ha identificado un problema importante, referido a la fragmentación entre plataformas de software móviles, debido a factores de incompatibilidad multiplataforma, como son Android, iOS (Malavolta, Ruberto, Soru, & Terragniz, 2015),es decir el código Java de una aplicación para Android no puede utilizarse en otro (por ejemplo, el código Objective-C de una aplicación Apple iOS) (IBM Software, 2012), haciendo que el desarrollo y mantenimiento de aplicaciones nativas se realice

de forma específica para cada plataforma, teniendo como consecuencia los siguientes aspectos (Hale & Hanson, 2015):

- Contratación de especialistas para cada una de las plataformas
- Reutilización de código, no es factible
- Mayor tiempo de desarrollo
- Redundancia de código
- Desarrollo a la par de las dos o más tipos de plataformas (Android, IOS , otros)
- Períodos de actualización más largos como parte del ciclo de vida del mantenimiento del software
- Alto Costo en Desarrollo

Razón por la cual se hace indispensable plantear una posible solución que permitan realizar el desarrollo de aplicaciones móviles multiplataforma que disminuyan el tiempo de desarrollo, sin dejar de lado la eficiencia de las mismas en su rendimiento, tomando en cuenta que este desarrollo se realiza en lenguajes de programación amigables como son: JAVASCRIPT, que permitirán evitar la contratación de capital humano especializado para cada plataforma.

1.3 JUSTIFICACIÓN

Esta investigación permitirá ahondar un poco más en el desarrollo de aplicaciones hibridas que es la posible solución al problema identificado, como es la fragmentación que encontramos en el desarrollo de las aplicaciones móviles. Enfocado específicamente en la eficiencia que podrán tener las aplicaciones móviles usando las dos técnicas (nativa e hibrida),

cuando hablamos de sincronización de datos que se encuentran en una base de datos externa en la nube como se ilustra en la (Figura 1).

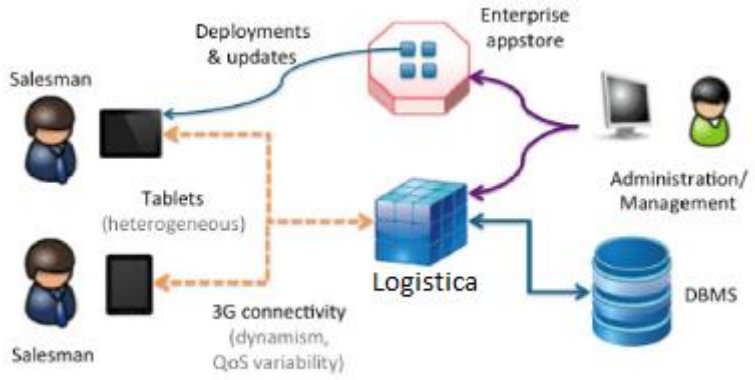


Figura 1 Diagrama de Funcionamiento

Tomando en cuenta que en el modelo nativo se deberá realizar el desarrollo de la sincronización doble por los problemas ya identificados de compatibilidad entre Android y iOS, si se diera el escenario de requerir la aplicación en las dos plataformas. Sin embargo, si nos guiamos por los antecedentes e investigaciones previas identificamos que el conocimiento de estas herramientas hibridas es muy general y empírico, con esta investigación se busca dejar un granito de arena más que permita seguir creciendo a los desarrolladores en el ámbito de las herramientas hibridas, específicamente en el uso de Ionic Framework.

1.4 OBJETIVOS

Realizar el estudio de la eficiencia de aplicaciones móviles con un framework híbrido y un nativo basado en Benchmarking.

1.4.1 Objetivos Específicos

- Analizar herramientas de software que permitan evaluar la eficiencia de las aplicaciones móviles específicamente en tiempos de respuesta, basado en el caso de estudio propuesto.
- Implementar un caso de estudio específico en las dos tecnologías de desarrollo.
- Realizar un análisis comparativo de las dos implementaciones de desarrollo móvil, enfocado en el caso de estudio usando Benchmarking

1.5 ALCANCE

Se pretende realizar un análisis comparativo entre aplicaciones móviles nativas e hibridas, es así que se plantea desarrollar un caso de estudio que permita realizar dicha comparación. Dando el paso al desarrollo una aplicación móvil nativa específicamente utilizando eclipse como ide de desarrollo basado en lenguaje java, como también utilizando un Framework Híbrido como es el caso de Ionic, que está basado en lenguaje HTML5 y Angular Js utilizando apache Cordova como su compilador, las dos aplicaciones utilizaran el SDK propio de Android.

La aplicación móvil a implementarse se basa en la siguiente arquitectura que se podrá visualizar en la (Figura 2)

Continúa

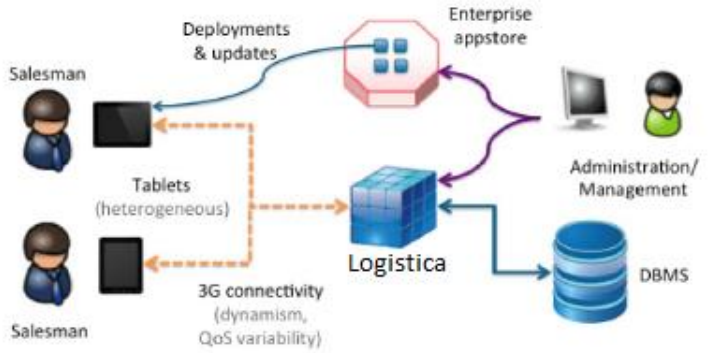


Figura 2 Arquitectura de Funcionamiento

Como se visualiza en la arquitectura se trata de una aplicación que trabaja tanto con una base de datos en la nube en este caso MYSQL como también utilizará una base de datos local que será SQLITE que está alojada en el dispositivo móvil, lo que plantea es tener una interacción directa entre las dos bases de datos en tiempo real, es decir, tanto online como en offline.

Cuando decimos online como offline se refiere, que cuando el dispositivo móvil tenga acceso a internet y esté o no, en uso de la misma, la aplicación se actualice es decir la base local tendrá los mismos datos que en la nube. Específicamente cuando en la base local (SQLITE) exista un cambio se refleje en la nube (MYSQL) y viceversa.

Como consecuencia de esta implementación se realizarán los siguientes módulos:

- Módulo de Acceso y Seguridad
- Módulo de Automatización de Logística

Con los cuales se pretenderá responder a las siguientes interrogantes:

- ¿Cuál de las dos aplicaciones es más eficiente, basándose en la norma ISO25000, con un enfoque específico al uso de recursos como también en relación al comportamiento temporal de la aplicación?

CAPITULO II

MARCO TEORICO

Identificado el problema resolver, en este caso la fragmentación en el desarrollo de aplicaciones móviles debido a incompatibilidad de los lenguajes de programación de las diferentes plataformas móviles, se plantea una solución viable como es el desarrollo de aplicaciones móviles con frameworks híbridos.

Por esta razón se realizó una búsqueda de estudios previos que permitan establecer el estado actual de desarrollo de aplicaciones móviles de forma nativa versus el uso de frameworks híbridos.

2.1 Aplicaciones Nativas

Las aplicaciones nativas ofrecen todas las funciones específicas del dispositivo y la plataforma disponibles en dispositivos móviles (como acelerómetros, lista de contactos, GPS y cámara), pero requieren múltiples bases de código diferentes para ser mantenidas para cada plataforma que un desarrollador de aplicaciones desea soportar (Heitkötter, Hanschke, & Majchrzak, 2013). Las aplicaciones nativas generalmente proporcionan la experiencia de usuario más rica y fluida, ya que utilizan dispositivos y elementos de interfaz de usuario específicos de la plataforma incluyendo su funcionalidad (Cevallos & Angulo, 2014). También son capaces de operar sin conexión y tienden a tener un mejor rendimiento, ya que interactúan directamente con el sistema operativo. A pesar de estas ventajas, las aplicaciones nativas dependen totalmente de la plataforma, tardan más tiempo en codificar y no reutilizan el código web existente. Esto significa que, para soportar aplicaciones nativas, los desarrolladores deben

aprender Objective-C (para dispositivos iOS) o Java para Android (para dispositivos Android) y crear bases de código enteramente nuevas. Estos factores se traducen en un mayor tiempo de desarrollo, redundancia de código y períodos de actualización más largos como parte del ciclo de vida del mantenimiento del software (Hale & Hanson, 2015).

Para crear una aplicación nativa, los desarrolladores deben escribir el código fuente y crear recursos adicionales, como imágenes, segmentos de audio y varios archivos de declaración específicos del sistema operativo. Utilizando herramientas proporcionadas por el proveedor de sistemas operativos (IBM Software, 2012).

Las aplicaciones nativas tienen archivos ejecutables que se descargan directamente en el dispositivo y se almacenan localmente, luego el usuario puede iniciar el proceso de instalación o, en algunos casos, por el departamento de TI de la organización. La forma más popular de descargar una aplicación nativa es visitando una tienda de aplicaciones, como la App Store de Apple, Android Marketplace (TABLA 1), pero existen otros métodos que a veces son proporcionados por el proveedor móvil (IBM Software, 2012).

En la siguiente tabla hacemos un comparativo de los diferentes sistemas operativos para dispositivos móviles

Tabla 1

Comparativa Sistemas Operativos Móviles

CARACTERISTICAS	APPLE IOS	ANDROID
LENGUAJE	OBJETIVE-C, C, C++	JAVA
HERRAMIENTAS	XCODE	ANDROID SDK
FORMATO EMPAQUETADO	.APP	.APK
TIENDAS	APPLE APP STORE	GOOGLE PLAY

2.1.1 Estructura de un proyecto Android

Al momento de crear un proyecto Android se genera una carpeta la misma que contiene todos los archivos y directorios que componen una aplicación básica como veremos a continuación en la (Figura 3).

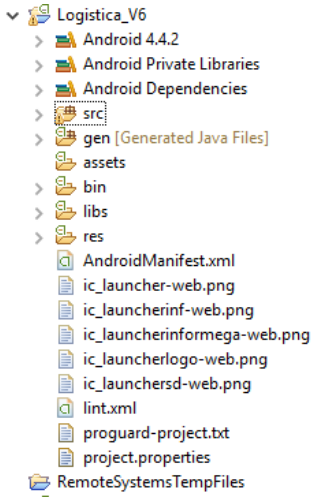


Figura 3 Estructura de una Proyecto Nativo

Fuente: (Apache Software Foundation, 2017)

A continuación, se detallará los directorios más importantes de la aplicación

- src: Este directorio contendrá todos los paquetes y clases, las mismas que son todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares. Inicialmente, Eclipse creará de manera automática el código básico de la pantalla (Activity) principal de la aplicación, en nuestro caso era MainActivity.
- gen: Este directorio contiene paquetes generados automáticamente por Eclipse, es recomendable no modificar ni mover estos archivos ya que son recursos del proyecto (Dimas, 2014).

- assets: Este directorio contiene ficheros auxiliares necesarios para la aplicación, como por ejemplo ficheros de configuración, de datos. Se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema (Dimas, 2014).
- bin: Este directorio contiene los elementos compilados de la aplicación y otros ficheros auxiliares. Cabe destacar el fichero con extensión. apk, que es el ejecutable de la aplicación que se instalará en el dispositivo móvil como se visualiza en la (Figura 4) (Dimas, 2014).

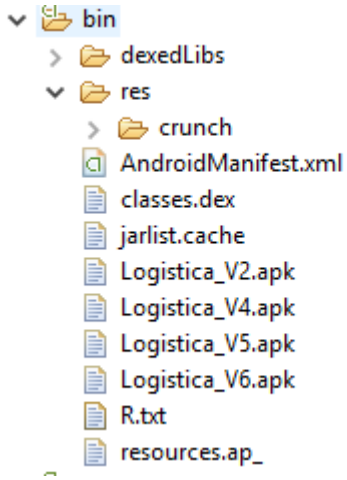


Figura 4 Estructura de la del directorio /bin

Fuente: (Android, 2017)

- libs: Este directorio contiene las librerías auxiliares, normalmente en formato ".jar" que utilicemos en nuestra aplicación Android (Dimas, 2014).
- res: El directorio /res está formado por todos los recursos de la aplicación, tomando en cuenta también subdirectorios que veremos a continuación (Dimas, 2014).
    - drawable: Este directorio incluye todas los gráficos e imágenes, que vamos a usar en la aplicación (Dimas, 2014).

  - layout: En este directorio se encuentran todos los XML, los mismos que son la parte gráfica de las activities, es decir, todos los XML son las pantallas de la aplicación (Dimas, 2014).
  - values: En este directorio contiene archivos de configuración con que usa la aplicación como títulos, nombres de variables y estilos de la misma (Dimas, 2014).
- AndroidManifest: Es un archivo de configuración muy importante para la aplicación, es la columna vertebral del proyecto, ya que en este archivo declaramos todas las actividades, los permisos, versiones del SDK del proyecto (Dimas, 2014).

2.1.2 Activity

Es un componente de la aplicación, está formada por una pantalla con la que el usuario puede interactuar, como llenar un formulario, escoger un elemento dentro de una lista. Cada actividad tiene una ventana asignada, en la que se puede generar su interfaz de usuario. Esta ventana, que es un view que generalmente abarca toda la pantalla y contiene los controles XML, los mismos que son administrados desde el activity (Android, 2017).

Entonces una aplicación está formada por múltiples activities vinculadas entre sí, tomando en cuenta que existe una actividad principal (ActivityMain) la cual se presenta al usuario cuando este inicia la aplicación por primera vez, sin embargo, cada vez que se inicia una actividad nueva, se detiene la actividad anterior (Android, 2017).

Ciclo de vida de un Activity

El ciclo de vida completo de un activity empieza entre la instancia del onCreate() y termina en onDestroy(). El activity configura el estado inicial en el onCreate(), y libera todos los recursos en onDestroy(), es decir si existe un subproceso de descarga de datos en ejecución de segundo plano, el activity puede crear ese subproceso en onCreate() y luego detenerlo en onDestroy() (Android, 2017).

El ciclo de vida visible de una actividad transcurre entre la instancia a onStart() y a onStop(). Durante ese periodo, el usuario puede ver el activity en pantalla e interactuar con ella, es decir, se instancia a onStop() cuando se empieza una nueva activity y está ya no se encuentra visible. Entre estos dos métodos, se puede conservar los recursos necesarios para mostrar la actividad al usuario. La aplicación podría llamar a onStart() y onStop() muchas veces durante el ciclo de vida completo de la actividad, ya que la actividad pasa de ser visible a estar oculta para el usuario (Android, 2017).

El ciclo de vida en primer plano de un activity empieza con la instancia a onResume() y a onPause(). En este periodo, el activity está por encima de todas las actividades en la pantalla y está listo para la interacción del usuario, frecuentemente un activity puede entrar y salir de primer plano, instanciando a onPause() ,cuando la aplicación entra en suspensión o cuando aparece un diálogo. Tomando en cuenta que este estado puede cambiar con frecuencia, el código en estos métodos debe ser bastante liviano para evitar que el usuario deba esperar como se puede visualizar en la (Figura 5) (Android, 2017).

Continúa

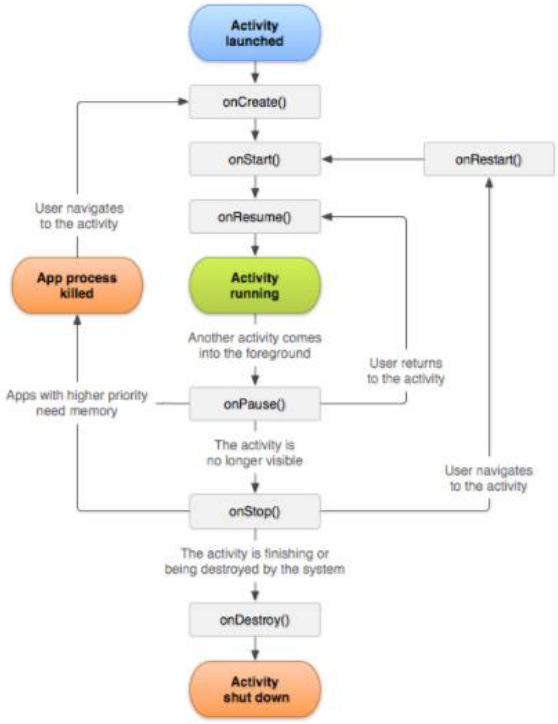


Figura 5 Ciclo de Vida de un Activity

Fuente: (Android, 2017)

2.2 Aplicaciones Hibridas

2.2.1 Definición

Una aplicación híbrida es aquella que es desarrollada para móviles en base a las tecnologías web: HTML + CSS + JavaScript (Alvarez, 2017). Son como cualquier aplicación nativa que se puede descargar a través de las tiendas de aplicaciones para cada sistema, por lo que, en principio usuarios finales no percibirán la diferencia con respecto a otros tipos de aproximaciones diferentes, como las aplicaciones nativas (Alvarez, 2017).

Al ejecutarse con tecnologías web, los desarrolladores que tienen experiencia en el desarrollo en este medio pueden aprovechar sus conocimientos para lanzarse de una manera

más rápida en el desarrollo de apps para móviles (Alvarez, 2017). Para hacer esto posible, las aplicaciones híbridas se ejecutan en lo que se denomina un “web view”, que no es más que una especie de navegador integrado en el móvil y en el que solamente se ejecuta la app híbrida (Alvarez, 2017).

2.2.2 Características

Las aplicaciones híbridas permiten a los desarrolladores crear o modificar aplicaciones web existentes y luego implementarlas en plataformas específicas utilizando contenedores nativos livianos. Las TNC son un tipo de tecnología de envoltura que utiliza motores de navegación específicos de plataforma y Apis nativas para proporcionar aplicaciones web encapsuladas con puntos finales a características de dispositivo nativas no disponibles para aplicaciones web (Foundation, 2016). Muchos desarrolladores web prefieren las aplicaciones híbridas, ya que las bases de código existentes pueden reutilizarse sin aprender y portar aplicaciones a lenguajes específicos de la plataforma como Objective-C o Java para Android. Estas ventajas se moderan sólo por un ligero impacto en el rendimiento que viene con la ejecución de JavaScript encapsulado en lugar de lenguajes de aplicación puramente nativos (Hale & Hanson, 2015).

El enfoque híbrido combina el desarrollo nativo con la tecnología web. Utilizando este enfoque, los desarrolladores escriben partes significativas de su aplicación en tecnologías web multiplataforma, al tiempo que mantienen acceso directo a las API nativas cuando es necesario y se visualiza a continuación en la (Figura 6).

Continúa

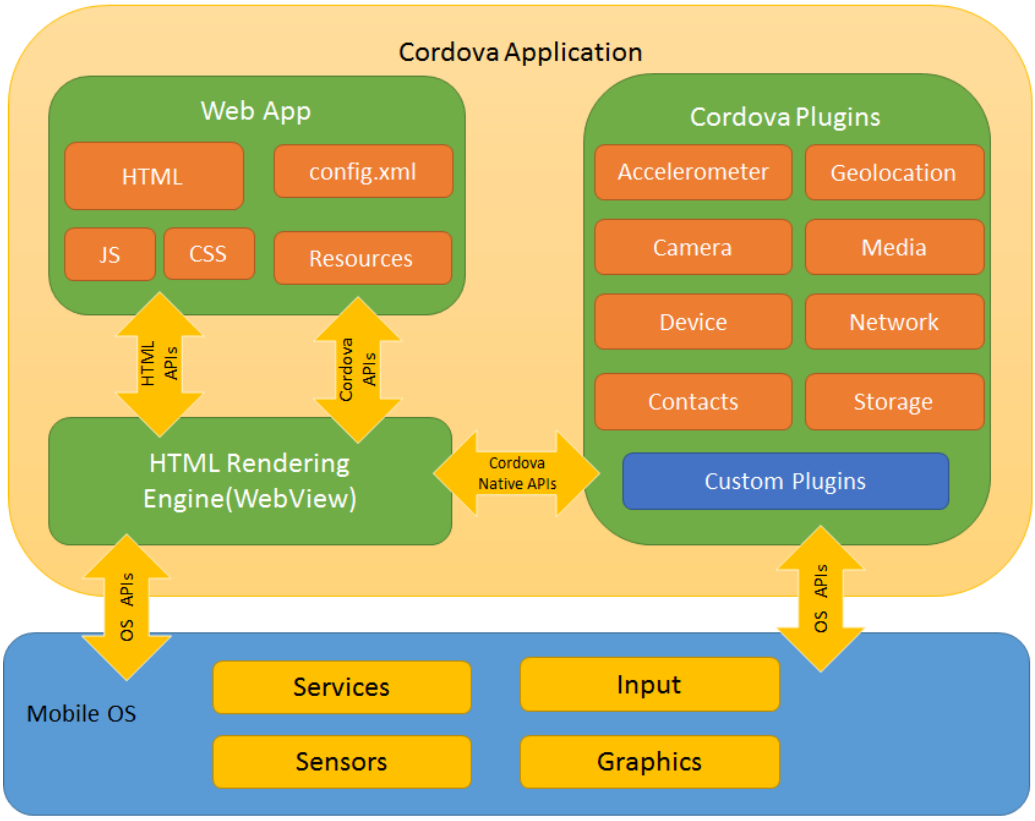


Figura 6 Arquitectura de Aplicaciones Hibridas

Fuente: (Apache Software Foundation, 2017)

2.2.3 Framework Híbrido

Ionic Framework

Figura 7 Descripción de Ionic Framework

Fuente: (López, 2017)

Ionic es un marco de desarrollo de aplicaciones móviles HTML5 orientado a la creación de aplicaciones móviles híbridas. Las aplicaciones híbridas son esencialmente sitios web pequeños que se ejecutan en una Shell de navegador en una aplicación que tiene acceso a la capa de plataforma nativa. Las aplicaciones híbridas tienen muchos beneficios sobre aplicaciones nativas puras, específicamente en términos de soporte de plataforma, velocidad de desarrollo y acceso a código de terceros (Drifty Co, 2016).

Ionic Framework está desarrollado en base al framework JavaScript de Google como es Angular Js, lo cual permite apoyarnos en todas las ventajas de desarrollo, como son, una excelente estructura de proyecto, uso de patrones de diseño de software y una buena gama de componentes y directivas (Alvarez, 2017).

Además, es un front-end que maneja todas las interacciones de apariencia y de interfaz de usuario que su aplicación necesita para ser eficiente. Algo como "Bootstrap for Native", pero con soporte para una amplia gama de componentes móviles nativos comunes, animaciones lisas y diseño hermoso (Drifty Co, 2016).

Ionic viene con elementos de interfaz de usuario de estilo nativo y diseños que obtendría con un SDK nativo en iOS o Android, pero que en realidad no existía antes en la web. Así que las aplicaciones hibridas no están destinadas a ser ejecutadas en una aplicación de navegador móvil como Chrome, sino usando la Shell de navegador de bajo nivel como UIWebView de iOS o WebView de Android, que tienen herramientas como Cordova / PhoneGap (Drifty Co, 2016)

Recursos de Ionic Framework

2.2.3.1.1 Node js

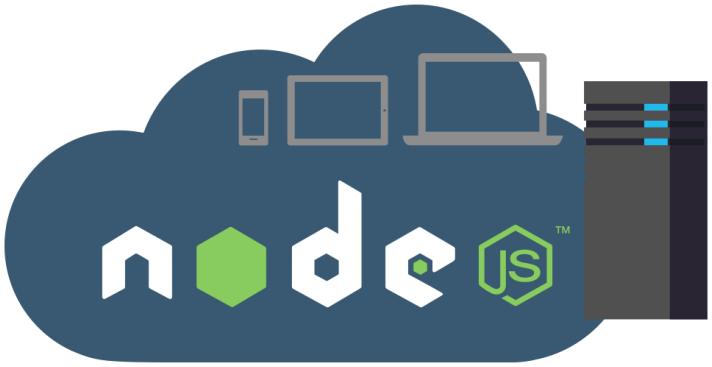


Figura 8 Descripción de Node js

Fuente: (Node.js Foundation, 2017)

Es un entorno de ejecución de JavaScript orientado a eventos asíncronos, está diseñado para construir aplicaciones escalables, esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo, tomando en cuenta que las redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar (Node.js Foundation, 2017).

2.2.3.1.2 Angular

Figura 9 Angular Js

Fuente: (Google, 2017)

Angular es una plataforma que facilita la creación de aplicaciones con la web. Angular combina plantillas declarativas, inyección de dependencia, herramientas de extremo a extremo y las mejores prácticas integradas para resolver desafíos de desarrollo. Angular permite a los desarrolladores crear aplicaciones que viven en la web, en el móvil o en el escritorio (Google, 2017).

2.2.3.1.3 Apache Cordova

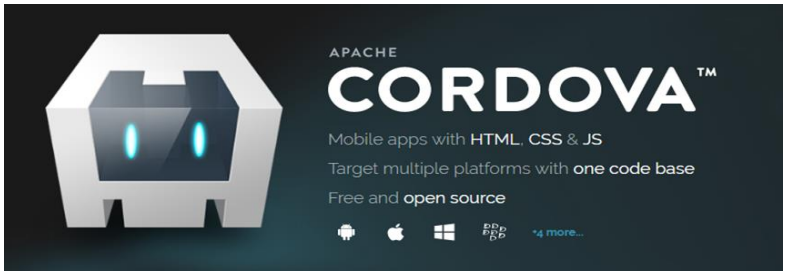


Figura 10 Descripción de Apache Cordova

Fuente: (Apache Software Foundation, 2017)

Apache Cordova es un marco de desarrollo móvil de código abierto que permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo para cada plataforma móvil (Apache Software Foundation, 2017).

Apache Cordova se creó en octubre de 2012 como un proyecto de nivel superior dentro de la Apache Software Fundation, el cual siempre permanecerá libre y de código abierto bajo la licencia Apache, versión 2.0 (Apache Software Foundation, 2017).

Instalación de Ionic Framework

En primer lugar, tenemos se debe empezar con una aclaración sobre los requisitos mínimos para la construcción de la aplicación con la versión actual de Ionic. Ionic soporta la interacción con plataformas iOS 7+ y Android 4.1 o superior. Sin embargo, dado que hay muchos dispositivos Android diferentes, es posible que algunos no funcionen. Como siempre, estamos ayuda online y mejorar la compatibilidad del dispositivo y mejorar la comunicación de la comunidad de GitHub (Drifty Co, 2016).

Basados en los recursos citados se define, 4 pasos para la instalación de Ionic

- Instalar Node Js
    - Descargar el ejecutable
- Instalar Apache Cordova
    - Comando: npm install -g cordova
- Instalar Ionic
    - Al momento de instalar Ionic se define como variable global de ambiente la palabra “ionic”, es decir podrá ser reconocida en la línea de comandos
    - Comando: npm install -g ionic
- Instalar Ionic CLI
    - Comando: npm install -g ionic@latest
- Instalar el Android SDK
    - Se recomienda instalar Android Studio

Ionic CLI

Es el intérprete por línea de comandos de Ionic, que contiene una serie de herramientas útiles para realizar, de una forma sencilla, muchas tareas habituales como:

- Start: Permite crear aplicaciones Ionic.
    - Comando: ionic start nombre-proyecto
- Start –list: Mostrara el listado de plantillas que vienen precargados en Ionic CLI, que permitirá ahorrar un poco de tiempo en el diseño inicial, desde una aplicación en blanco, una aplicación con menú lateral o una aplicación con paginación hasta proyectos con casi todos los componentes como veremos en la (Figura 11).

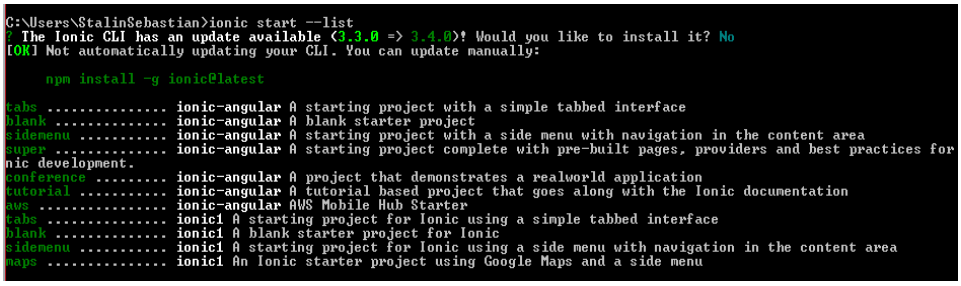


Figura 11 Listado de Plantillas Ionic CLI

- serve: Permite lanzar todo el proceso de transpilación y abre la app en el navegador.
- platform: Ayuda para añadir una nueva plataforma(Android, Ios) a una aplicación.
- build: Este comando genera la app para una plataforma especificada (ej: ionic build android), es decir genera el archivo. apk.
- emulate: Este comando lanza el emulador para visualizar la aplicación en la plataforma Ionic. Es simular a un ionic serve, solo que usa el emulador disponible en lugar del navegador.

- generate: Este comando permite generar un listado de componentes propios de Ionic como:
    - Pages (páginas)
    - Providers (Servicios),
    - Pipes.

Creación de una Aplicación Ionic

Luego de instalar Ionic se crea un directorio a elección, en el ejemplo se generará el directorio "C:\ionic\ionic3". Abrimos la línea de comandos en modo administrador (evitar problemas de permisos de lectura y escritura), seguido se ingresa al directorio creado anteriormente como se visualiza en la (Figura 12).

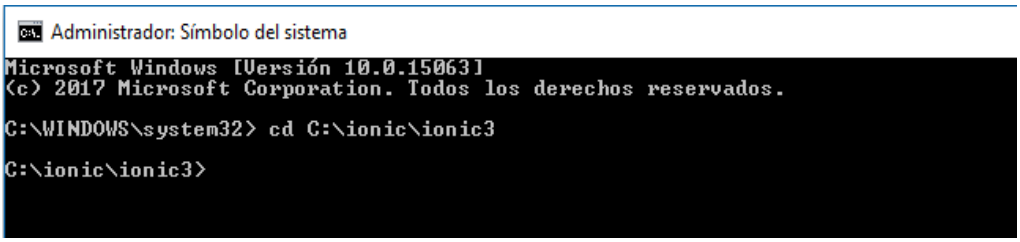


Figura 12 Directorio Ionic

A continuación, con la ayuda de Ionic CLI se ejecuta el comando "ionic start" acompañado del nombre del proyecto y la plantilla que se va a usar como se visualiza en la (Figura 13).

Figura 13 Comando Creación Proyecto Ionic

A continuación, se crea un directorio en este caso con el nombre “test” al cual se accede, para luego ejecutar el comando “ionic serve” como se visualiza en la (Figura 14).

```
C:\ionic\ionic3\test>ionic serve
```

Figura 14 Comando Visualización Proyecto Ionic

Se abre el navegador por default, usando un servidor local (Node Js) en este caso localhost usando el puerto 8100 que viene predefinido mostrando a su vez la página inicial de la aplicación como se visualiza en la (Figura 15).

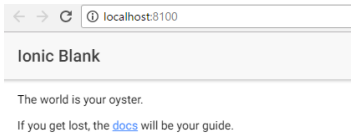


Figura 15 Proyecto Ionic Compilado en un Browser

Estructura de un Proyecto Ionic

Luego de crear el proyecto con Ionic CLI se obtiene un proyecto con una lista de carpetas y archivos que forman parte de la aplicación como se muestra en la (Figura 16).

Continúa

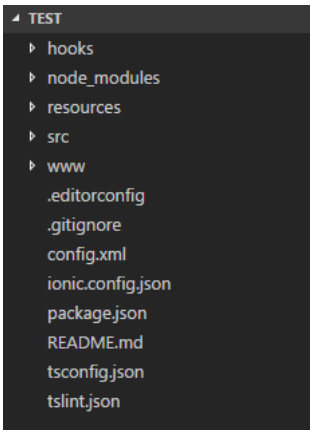


Figura 16 Listado de Archivos y Carpetas Proyecto Ionic

- Hooks: Este directorio contiene una especie de disparadores que se ejecutarán ante determinadas situaciones como quitar el código CSS que pertenece a otras de las plataformas. Esto lo hace para aligerar el proyecto, ya que a la hora de desarrollar Ionic produce el CSS de Android, iOS y Windows, pero no se necesita todo ese CSS, si lo que se está produciendo es el apk para Android (Molina, 2017).
- node_modules: Este directorio contiene todas las dependencias de npm (Node.js) que vienen definidas en archivo package.json. (Molina, 2017)
- resources: Este directorio es importante a la hora de configurar iconos de la aplicación, o el icono de la pantalla splash (que es la imagen que aparece mientras la app se está cargando) (Molina, 2017).
- src: En este directorio se encuentran los archivos fuente con el código original que se escribirá al desarrollar la aplicación. El 98% del proyecto se encuentra dentro este directorio repartido a su vez en sub carpetas como: (Molina, 2017)
    - app: En este directorio se encuentra la raíz de nuestra aplicación específicamente en dos archivos:
        - app.components

- app.modules

- assets: En este directorio contiene archivos de imágenes y otro tipo de materiales externos que se va a usar en las páginas (Molina, 2017)
- pages: En este directorio están ubicadas las páginas (pantallas), los archivos de estilos como también el controlador propio de cada página de la aplicación (Molina, 2017)
- theme: En este directorio contiene el archivo css que trabaja en todo el proyecto de forma global, y viene prestablecido un grupo de colores primarios, como fuente de la letras o palabras que se establecen en la aplicación (Molina, 2017)
- Además, se encontrarán archivos sueltos como el index.html que es el punto de arranque de la aplicación web (Molina, 2017).

- www: En este directorio están los archivos que se visualizan cuando es ejecutada la aplicación con el comando "ionic serve", se recomienda no modificar nada de este directorio (Molina, 2017).

2.3 Aplicaciones Nativas vs Hibridas

Figura 17 Aplicaciones Nativas vs Hibridas

Fuente: (IBM Software, 2012)

En una investigación análoga a la nuestra se realiza una encuesta a un grupo de 12 desarrolladores móviles expertos en donde, los sujetos P1, P8 apoya el desarrollo de aplicaciones híbridas. Los 10 entrevistados restantes están a favor de la creación de aplicaciones nativas puras y creen que el modelo híbrido actual tiende a buscar y se comportan más como páginas web que las aplicaciones móviles. P11 argumentó que "el enfoque nativo ofrece las mayores características " P4 declaró " experiencia del usuario en las aplicaciones nativas es muy superior en comparación para una aplicación web. En una serie de casos, los participantes habían completamente alejado del híbrido para el enfoque nativo (Joorabchi, Erfani, Mesbah, & Kruchten, 2013) .Por otro lado, P1 argumentó que " que realmente depende de la complejidad y el tipo de la aplicación ", por ejemplo, " el intercambio de información aplicaciones pueden adoptar fácilmente el modelo híbrido para empujar el contenido de noticias y actualizaciones a través de múltiples plataformas. " En la encuesta, el 82% respondió que tiene experiencia en el desarrollo nativo, el 11% han intentado soluciones híbridas, y el 7% ha desarrollado aplicaciones web móviles. La mayoría de los encuestados están a favor del enfoque nativos. Otros dijeron que:" HTML5 tiene mucho potencial y es probable que abordar muchos de los problemas actuales en el futuro, ya que ahorra tiempo y coste de desarrollo " (Joorabchi, Erfani, Mesbah, & Kruchten, 2013).

La mayoría de los participantes argumentaron que cuando el coste de desarrollo no es un problema, las empresas tienden a desarrollar aplicaciones nativas. Por supuesto, también depende del tipo de aplicación; que se necesitan más la experiencia del usuario de dispositivos o características específicas, nativo parece ser la opción más clara (Joorabchi, Erfani, Mesbah, & Kruchten, 2013).

Por último, cuando preguntamos a los participantes que, si el desarrollo de aplicaciones nativas será sustituido por soluciones híbridas o desarrollo web móvil, debido a sus desafíos, todos los entrevistados y el 70% de los encuestados no están de acuerdo, y el 10% indicó que siempre habrá una combinación de enfoques nativas e híbridas (Joorabchi, Erfani, Mesbah, & Kruchten, 2013).

Según (Charland & LeRoux, 2011) en su artículo "Desarrollo de Aplicaciones Móviles: Híbrido vs. Nativo" dicen que la implementación de una aplicación de software comienza con el código. En el caso de código nativo, lo más a menudo estos días, el desarrollador escribe típicamente en un dialecto C, como en el caso del iPhone. En este trabajo en Nitobi y en PhoneGap, han tenido mucha experiencia luchando con las diferentes plataformas móviles desde una perspectiva de desarrollo nativo. Por supuesto, por diversas razones de mercado o de organización, la mayoría de los desarrolladores o equipos deben ser compatibles con aplicaciones en múltiples plataformas inteligentes.

Lo que hace las cosas aún más complicadas son las diferencias entre los SDK de la plataforma reales (kits de desarrollo de software). Existen diferentes herramientas, construir sistemas, Apis y dispositivos con capacidades diferentes para cada plataforma. De hecho, la única cosa que estos sistemas operativos tienen en común es que todos ellos vienen con un navegador móvil que se puede acceder mediante programación desde el código nativo. Cada plataforma nos permite crear una instancia de navegador, sin bordes, e interactuar con su interfaz JavaScript de código nativo. Desde dentro de ese Webview podemos llamar a código nativo de JavaScript. Este es el truco que llegó a ser conocido como la técnica de PhoneGap por primera vez por Eric Oesterle, Rob Ellis, y Brock Whitten por primera SDK iPhone OS en

iPhoneDevCamp en 2008. Este enfoque fue portado más tarde para Android, BlackBerry, y luego al resto de la PhoneGap plataformas (Charland & LeRoux, 2011).

A continuación, se mostrará en la Tabla 2 un cuadro comparativo de la disponibilidad de componentes de los dos métodos de desarrollo.

Tabla 2
Comparativa Componentes Nativos e Híbridos

Generic Attributes	Native	Hybrid
Languages	Platform Specific	HTML5,CSS, Javascript
Distribution	Appstore	Appstore
Graphics	Native APIs	HTML, Canvas, SVG
Connectivity	On & offline	On & offline
Performance	Fast	Moderate
Home button	Default	Default
Platform/Device Independent	No	Yes
Dev. Time	High	Low
Device Features	**Native**	**Hybrid**
Accelerometer	Yes	Yes
Account Access (Stored Accts)	Yes	Yes
Battery State	Yes	Yes
Camera	Yes	Yes
Calendar	Yes	Yes
Contacts	Yes	Yes
Flashlight	Yes	Yes
Geolocation	Yes	Yes
Microphone	Yes	Yes
Network Sockets	Yes	Yes
Offline Storage	Yes	Yes
Push Notific.	Yes	Yes
Phone Identity	ID / IP / Phone #	ID / IP / Phone #
Phone Status	Yes	Yes
SMS	Yes	Yes
Vibration	Yes	Yes
Video	Yes	Yes

2.4 Consumo de Energía Nativo vs Híbrido

La mayor parte de las aplicaciones móviles son Android, sin embargo, está escrito completamente en Java. En una muestra de 109 proyectos se examinaron de F-Droid 4, sólo el 4% son híbridos. No está claro aún si este enfoque conduce a aplicaciones eficientes de energía (Oliveira, Torres, Castor, & Ximenes, 2016).

Se realizó un análisis de los 2 modelos, es decir aplicaciones nativas e hibridas y se midió el consumo de energía en los 2 casos. Nuestros resultados indican que es posible ahorrar energía

utilizando este enfoque híbrido. En una de las aplicaciones, TriRose, la versión híbrida de energía ahorrada hasta el 30% mediante la agrupación de las invocaciones a la parte JavaScript. Realización de la misma agrupación de operaciones utilizando únicamente Java dado sólo ganancia marginal en términos de consumo de energía (Oliveira, Torres, Castor, & Ximenes, 2016).

2.5 Normativas de Calidad con enfoque en la Eficiencia de Software

2.5.1 Norma ISO 25000

La calidad del producto, junto con la calidad del proceso, es uno de los aspectos más importantes actualmente en el desarrollo de Software (ISO25000, 2017). Relacionada con la calidad del producto, recientemente ha aparecido la familia de normas ISO/IEC 25000, que proporciona una guía para el uso de la nueva serie de estándares internacionales llamada Requisitos y Evaluación de Calidad de Productos de Software (ISO25000, 2017).

ISO/IEC 25000 constituye una serie de normas basadas en ISO/IEC 9126 y en ISO/IEC 14598 cuyo objetivo principal es guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad (ISO25000, 2017).

2.5.2 Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones (ISO25000, 2017). Esta característica se subdivide a su vez en las siguientes sub características (ISO25000, 2017):

- Comportamiento temporal: Los tiempos de respuesta y procesamiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (benchmarking) establecido (ISO25000, 2017).

Tabla 3
Ponderación de Importancia Tiempos de Respuesta

IMPORTANCIA TIEMPOS DE RESPUESTA	
0	No es importante el tiempo de respuesta en la aplicación móvil
1	Es importante el tiempo de respuesta en la aplicación móvil
2	Es primordial e importante el tiempo de respuesta en la aplicación móvil

- Utilización de recursos: Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas (ISO25000, 2017).

Tabla 4
Importancia de la Utilización de Recursos de Software y de la Aplicación

UTILIZACIÓN DE RECURSOS DE SOFTWARE Y DE LA APLICACIÓN	
0	No es importante el uso de recursos de software y de aplicación móvil
1	Es importante el uso de recursos de software y de aplicación móvil
2	Es primordial e importante el uso de recursos de software y de aplicación móvil

- Capacidad: Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos (ISO25000, 2017).

Tabla 5
Ponderación de Importancia Límites Máximos de la Aplicación Móvil

Límites máximos de la aplicación móvil	
0	No es importante el límite máximo de la aplicación móvil
1	Es importante el límite máximo de la aplicación móvil
2	Es primordial e importante el límite máximo de la aplicación móvil

2.5.3 Modelo de Calidad

Modelo IQMC

Según Coral Calero (Coral, Moraga, & Piattini, 2010), en el libro Calidad del Producto y Proceso de Software, IQMC se describe como un método que permite construir modelos de calidad fijos, a la medida y mixtos teniendo como objetivo evaluar un producto de software basados mediante un conjunto de siete pasos los mismos que se listan en la (Tabla 6).

Tabla 6
Pasos del Método IQMC

Pasos	Tarea
0	Estudio del ámbito del software
1	Determinación de características
2	Determinación de Subcaracterísticas de Calidad
3	Determinación de atributos derivados
4	Determinación de atributos básicos
5	Establecimiento de relaciones entre factores de calidad
6	Determinación de métricas para los atributos

Continúa

Fuente: (Bermeo, Campaña, & Melo, 2014)

Paso 0. Estudio del ámbito del software.

Consiste en realizar un estudio del ámbito al cual pertenecen los componentes de software y hardware con el objetivo de proporcionar la base conceptual para la identificación de las características de calidad para las cuales se quiere evaluar, en este caso la eficiencia de los mismos.

Puede ser expresado en Estándares, Modelos UML, Diagramas Arquitectónicos

Paso 1 y 2. Determinación de características y subcaracteristicas de calidad.

En estos pasos, IQMC propone la selección de características y subcaracteristicas basados en la norma ISO/IEC 25000, específicamente en este caso en la eficiencia de desempeño, como se visualiza en la (Figura 18).

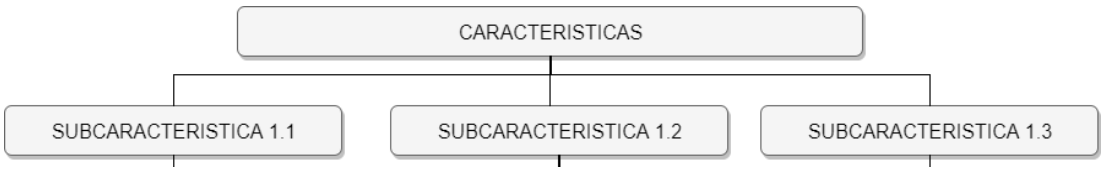


Figura 18 Características y Subcaracteristicas

Paso 3 y 4. Determinación de atributos derivados y básicos.

Estos dos pasos siguientes consisten en descomponer las subcaracteristicas en atributos derivados y estos en atributos básicos para luego poder medirlos de forma directa.

Paso 5. Establecimiento de relaciones entre factores de calidad.

Se establecen las relaciones entre factores de calidad que permiten conocer las dependencias entre los distintos factores de calidad del modelo.

La (Figura 19) se muestra de manera explícita las relaciones de profundidad que se usarán para el desarrollo del modelo de calidad

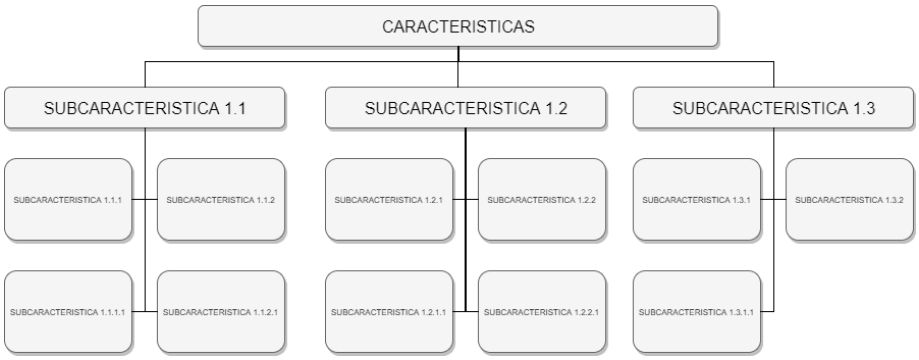


Figura 19 Relaciones de Profundidad

Paso 6. Determinación de métricas.

Se determinan las métricas para los atributos identificados.

En la (Figura 20) se muestra más claramente los pasos para la construcción de un modelo de calidad de software en este caso específico en la Eficiencia de desempeño

Continúa

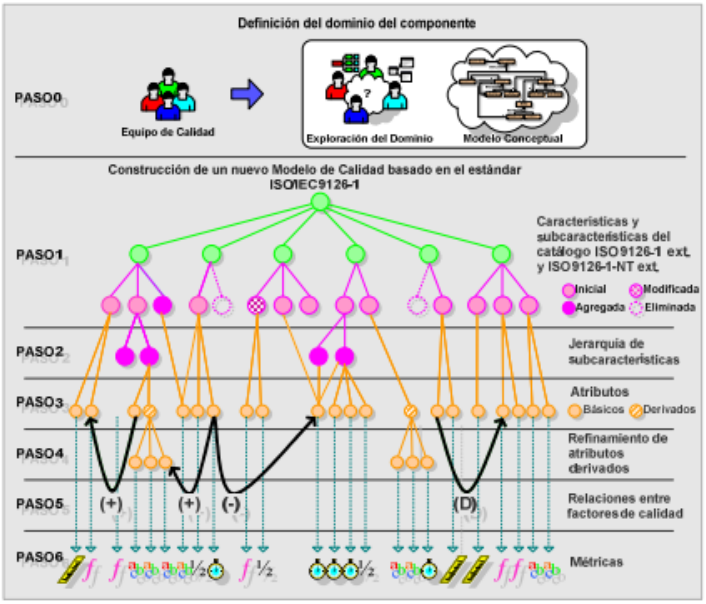


Figura 20 Pasos del Modelo IQMC

Fuente: (Coral, Moraga, & Piattini, 2010)

En estudios realizados por (Coral, Moraga, & Piattini, 2010), IQMC se basa en el catálogo de factores de calidad que proporciona la norma ISO 25000, debido a esto se decidió optar por este método para el desarrollo del modelo de evaluación (Bermeo, Campaña, & Melo, 2014).

2.5.4 Herramientas Comparativas

Benchmarking

Se denomina Benchmarking al estudio comparativo en áreas o sectores de empresas competidoras con el fin de mejorar el funcionamiento de la propia organización. Estos estudios se hicieron muy populares especialmente en EEUU en la década de los ’90, y un gran número de importantes empresas los han incorporado. Sin embargo, si no se cumplen ciertos requisitos o claves, se tornará difícil concretar los objetivos planteados. El Benchmarking no es sólo un estudio comparativo de datos. Sus alcances son más extensos: apuntan al mejoramiento de la

organización, de la estructura productiva o de las políticas internas para lograr ventajas competitivas (Boxwell, Rubiera, & MacShane, 1995).

El benchmarking puede llegar a ser importante a la hora de ver el grado de funcionalidad de los sistemas computarizados. Se puede observar que los sistemas son soluciones a problemas que se van presentando a la vez que la tecnología va creciendo, entonces verificando la funcionalidad de aplicaciones parecidas realizadas por distintos métodos, la eficiencia con la que se hizo cada proyecto, las ventajas que ofrecen dichos métodos, los mantenimientos que se realizan a dicha aplicación, pueden ser formas óptimas y eficientes de poder realizar dicho análisis.

El desempeño de los sistemas de software se mide por varias variables como: velocidad con que trabaja, tiempos de respuestas, capacidad. Pueden ser puntos de partida buenos para aplicar benchmarking, trazando objetivos y metas que mejoren estándares de trabajo y calidad profesional, para estar siempre a la vanguardia de la funcionabilidad/desempeño de los aplicativos de software.

2.6 Metodología de Desarrollo de Aplicaciones Móviles

El desarrollo de aplicaciones para proveer servicios móviles, difiere del desarrollo de software tradicional en muchos aspectos, lo que provoca que las metodologías usadas para estos entornos móviles, también difieran de las del software clásico (Ramsin & Rahimian, 2008).

Las características especiales de los entornos móviles como el canal de radio, la capacidad de los terminales, la portabilidad, el tiempo de salida al mercado “Time-to-Market”, la

movilidad del usuario, entre otras; exigen nuevas tendencias para desarrollar el software móvil en Latinoamérica (Gasca, Camargo, & Medina, 2014).

Por las razones anteriores, se propone la "Metodología para el Desarrollo de Aplicaciones Móviles, MDAM". (Gasca, Camargo, & Medina, 2014).

La metodología se encuentra enmarcada en cinco fases como se muestra en la (Figura 21), denominadas: análisis, diseño, desarrollo, pruebas de funcionamiento y entrega. A continuación, se describe cada una de las actividades que intervienen en el desarrollo de la propuesta (Gasca, Camargo, & Medina, 2014).

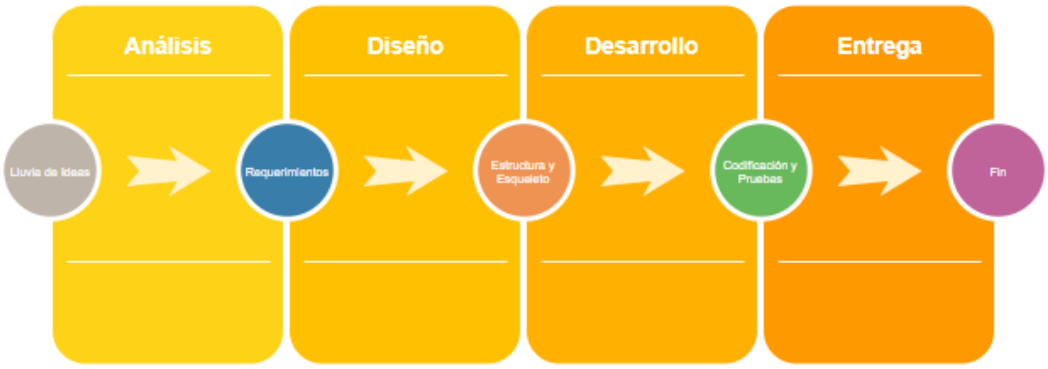


Figura 21 Fases de la Metodología MDAM

Fuente: (Gasca, Camargo, & Medina, 2014)

2.6.1 Análisis

En esta fase se analizan las peticiones o requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil "Cliente", el propósito es definir las características del mundo o entorno de la aplicación (Gasca, Camargo, & Medina, 2014).

2.6.2 Diseño

El objetivo de esta etapa es plasmar el pensamiento de la solución mediante diagramas o esquemas, considerando la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos. A esta fase se retorna si no se obtiene lo deseado en la etapa prueba de funcionamiento (Gasca, Camargo, & Medina, 2014).

2.6.3 Desarrollo

El objetivo de esta fase es implementar el diseño en un producto de software. (Gasca, Camargo, & Medina, 2014).

2.7 Herramienta para Monitorizar Componentes de la Aplicación

Simple System Monitor es una aplicación móvil que permite monitorizar todos los componentes del sistema del dispositivo Android. Se puede ver gráficas del uso de la CPU, así como el uso de la memoria RAM.

Figura 22 Simple System Monitor

Los componentes que se puede monitorizar se encuentran en una pestaña diferente, tiene una interfaz intuitiva que permite solo con pasar una pestaña para monitorizar otro componente (Rosso, 2018).

Además, Simple System Monitor permite monitorizar los componentes como el uso de memoria RAM en modo background, es decir que permite ver el uso de los componentes mientras se usan otras a aplicaciones como veremos a continuación en la (Figura 23) (Rosso, 2018).

Figura 23 Simple System Monitor Background

CAPITULO III

CONTRUCCION DEL MODELO DE CALIDAD IQMC

3.1 Introducción

Para Continuar con el análisis, se desarrolla todos los pasos para la construcción del modelo de calidad IQMC, con el objetivo de obtener la matriz de evaluación que permita realizar el análisis comparativo en el desarrollo de aplicaciones móviles nativas versus hibridas, basándonos en los factores de calidad que propone la norma ISO/IEC 25000 en este caso específico en la Eficiencia de Desempeño.

3.2 Aplicación de los Pasos del Modelo IQMC

3.2.1 Paso 0. Estudio del ámbito del software.

Para obtener estudio de componentes de software y hardware se realizó un Diagrama Arquitectónico del Caso de Estudio (Capitulo 1.5) como se visualiza en la (Figura 24).

Continúa

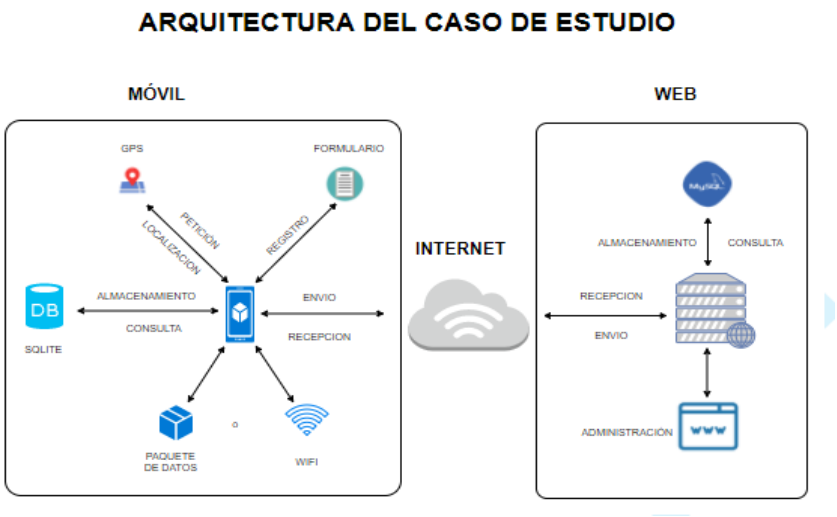


Figura 24 Arquitectura del Caso de Estudio

3.2.2 Paso 1 y 2. Determinación de características y subcaracteristicas de calidad.

Para determinar las características y subcaracteristicas de calidad me base en la norma ISO/IEC 25000, específicamente en la eficiencia de desempeño la cual se basa en tres subcaracteristicas las cuales fueron citadas en el capítulo 2.5.2, lo que me permitió establecer la siguiente jerarquía la misma que se visualiza en la (Figura 25).

Figura 25 Características y Subcaracteristicas

Como podemos visualizar la característica principal del estudio es la Eficiencia de Comportamiento y sus subcaracteristicas son el Comportamiento Temporal, Utilización de Recursos y la Capacidad.

3.2.3 Paso 3, 4 y 5. Determinación de atributos derivados y básicos

El propósito de estos tres pasos es descomponer las características antes ya identificadas en atributos derivados y estos a su vez en atributos básicos los mismos que no permitirán medirlos de forma directa como se visualiza en la (Figura 26).

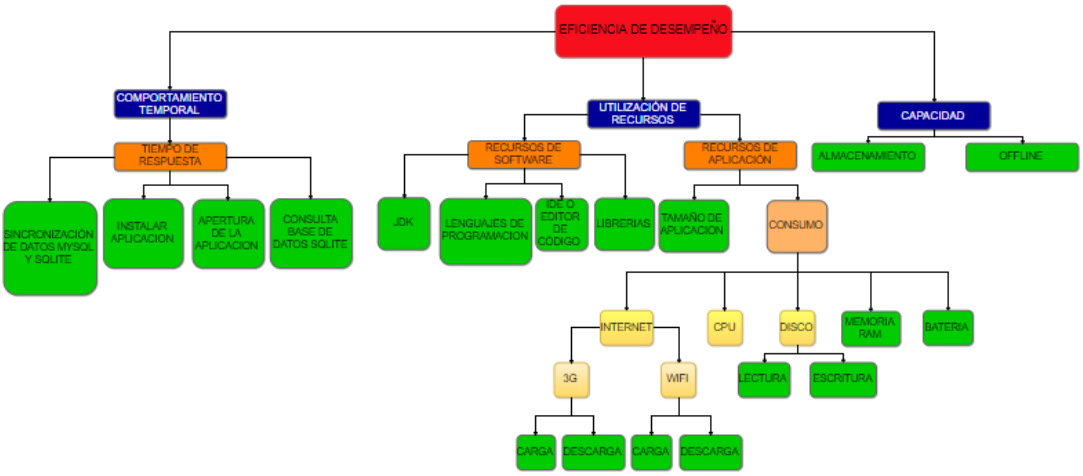


Figura 26 Descomposición de Características y Subcaracteristicas en Atributos

3.2.4 Paso 6. Determinación de métricas.

En la evaluación de las dos aplicaciones nativas e híbridas se usaron métricas objetivas o también conocidas como medidas directas que determinaron el nivel de eficiencia de cada aplicación planteados en los pasos 3,4 y 5 del método de IQMC.

Métricas de Uso de Recursos de Software

En la (Tabla 7) se describe las métricas que se usaron para evaluación de uso de Recursos de Software de la Aplicación.

Tabla 7
Métricas de Uso de Recurso de Software

#	TIEMPO DE RESPUESTA	EQUIVALENTE
1	BAJO (B)	0.33
2	MEDIO (M)	0.66
3	ALTO (A)	1

Métricas de Uso de Recursos de Hardware

En la (Tabla 8) se describe las métricas que se usaron para evaluación de uso de Recursos de Hardware de la Aplicación.

Tabla 8
Métricas de Uso de Recurso de Hardware

#	USO DE HARDWARE	EQUIVALENTE
1	BAJO(B)	0.33
2	MEDIO(M)	0.66
3	ALTO(A)	1

Métricas de Tiempos de Respuesta de Hardware y Software

En la (Tabla 9) se describe las métricas que se usaron para evaluación de tiempos de respuesta de la Aplicación

Tabla 9
Métricas de Tiempo de Respuesta

#	TIEMPO DE RESPUESTA	EQUIVALENTE
1	BAJO (B)	0.33
2	MEDIO (M)	0.66
3	ALTO (A)	1

Tabla 10
Modelo de Evaluación de la Eficiencia de Comportamiento

EFICIENCIA DE COMPORTAMIENTO						
CARACTERISTICAS SUBCARACTERISTICAS ATRIBUTO				MÉTRICA	IMPORTANCIA	DESCRIPCIÓN
1	COMPORTAMIENTO EN EL TIEMPO					
	1.1	TIEMPO DE RESPUESTA				
		1.1.1	SINCRONIZACIÓN DE DATOS MYSQL Y SQLITE	B=0,33/ M=0,66/ A=1	2	Permite medir el tiempo de respuesta

Continúa

						de sincronización de las dos bases de datos
		1.1.2	INSTALACIÓN APLICACIÓN	B=0,33/ M=0,66/ A=1	1	Permite medir el tiempo de respuesta de instalación de la aplicación
		1.1.3	INICIO DE LA APLICACIÓN	B=0,33/ M=0,66/ A=1	2	Permite medir el tiempo de respuesta de la aplicación al iniciar la misma
		1.1.4	CONSULTA BASE DE DATOS SQLITE	B=0,33/ M=0,66/ A=1	2	Permite medir el tiempo de respuesta de la aplicación a una consulta a la base de datos
2	UTILIZACIÓN DE RECURSOS					
	2.1	RECURSOS DE SOFTWARE				
		2.1.1	LENGUAJES DE PROGRAMACION	B=0,33/ M=0,66/ A=1	1	Permite determinar el número de lenguajes de programación que se utiliza para el

Continúa

						desarrollo de la aplicación
		2.1.2	IDE O EDITOR DE CÓDIGO	B=0,33/ M=0,66/ A=1	1	Permite determinar la cantidad de ides o editores de código que se utilizó en el desarrollo de la aplicación
		2.1.3	LIBRERIAS	B=0,33/ M=0,66/ A=1	1	Permite determinar la cantidad de librerías que se utilizó en el desarrollo de la aplicación
	2.2	RECURSOS DE HARDWARE				
		2.2.1	TAMAÑO DE APLICACIÓN	B=0,33/ M=0,66/ A=1	2	Permite medir la cantidad en Mb de la aplicación
		2.2.2	CONSUMO			
			2.2.2.1 MEMORIA RAM	B=0,33/ M=0,66/ A=1	2	Permite medir el consumo de la memoria RAM de la aplicación

Continúa

			2.2.2.2	BATERIA	B=0,33/ M=0,66/ A=1	2	Permite medir el consumo de batería de la aplicación
			2.2.2.3	CPU	B=0,33/ M=0,66/ A=1	2	Permite medir el consumo de CPU de la aplicación
			2.2.2.4	DISCO	B=0,33/ M=0,66/ A=1	2	Permite medir el consumo de disco de la aplicación
			2.2.2.5	INTERNET	B=0,33/ M=0,66/ A=1	2	Permite medir el consumo de internet de la aplicación
3	CAPACIDAD						
	3.1	ALMACENAMIENTO			B=0,33/ M=0,66/ A=1	2	Permite medir el límite de almacenamiento local de la aplicación.
	3.2	OFFLINE			B=0,33/ M=0,66/ A=1	3	Permite medir la capacidad de la aplicación sin tener internet

CAPITULO IV

IMPLEMENTACION DEL CASO PRÁCTICO BASADO EN EL DESARROLLO NATIVO E HÍBRIDO DE APLICACIONES MOVILES

4.1 Caso de Estudio

El caso de estudio corresponde a una aplicación móvil de logística, que permitirá al administrador del mismo tener el registro de pedidos, como también la ruta que realizaran los usuarios (mensajeros).

4.2 Modelado de Negocio

Se realizó varias reuniones con el cliente, para identificar y comprender las necesidades específicas del mismo, permitiendo conocer el dominio del problema y evaluar las posibles soluciones viables al mismo mediante el desarrollo de una aplicación móvil.

4.3 Identificación de los procesos del negocio

4.3.1 Actividad de la Empresa

Es una empresa dedicada al campo de los análisis clínicos, especialmente en análisis como diálisis, citología, tamizaje neonatal entre otros, además de poseer varios reconocimientos y certificaciones internacionales. Tiene una larga lista de clientes (Asociados) a los cuales se les brinda el servicio de Logística.

4.3.2 Proceso de Logística

ADMINISTRACIÓN

- Departamento de Calidad
- Área de Pre análisis

SOPORTE

- Área Logística

FUNCIONALIDAD

- Producción Pre análisis
- Pre ingreso
- Optimización de Rutas

La empresa necesita automatizar este proceso pues actualmente el ingreso de los pedidos se realiza cuando el mensajero llega a la empresa y se pretende realizar el ingreso del pedido cuando el mensajero arribe al cliente (asociado).

4.4 Identificación de Roles

A continuación, se detalla el listado de involucrados en el proceso de logística y la función que realiza cada uno.

- **Mensajeros**

Personal de Logística encargada de trasladarse a cada uno de los Clientes (Asociados), el mismo que es encargado de registrar en la aplicación cada uno de los pedidos.

- **Líder de Pre análisis**

Persona encargada de dar seguimiento a las rutas de los mensajeros, y gestionar el proceso de pre ingreso de los pedidos al laboratorio.

- **Líder del Departamento de Calidad**

Persona encargada de supervisar periódicamente el proceso de Logística.

- **Asociado**

Persona represéntate del Cliente (Asociado) que entrega las muestras e ingresa la contraseña al pedido.

4.5 Elicitación de Requerimientos

En esta fase procedemos a estudiar y definir el dominio del problema con la empresa Informega, involucrando al personal del departamento de calidad como el área de logística de la empresa, identificando las necesidades reales que se necesite implementar para la optimización del proceso de logística de la empresa.

4.5.1 Identificación de Stakeholders

Este proceso nos permitirá establecer todos los participantes en el desarrollo de este pequeño caso de estudio a desarrollarse.

Los Stakeholders que van a participar en el proceso de ingeniería de requerimientos son:

Por parte del cliente:

- Cliente: Gerente Departamento de Calidad. Principal contacto con la empresa y quien aceptara el desarrollo del mismo.
- Usuario 1: Jefe de Logística, persona que interactúa con la aplicación desarrollada.
- Usuario 2: Mensajero, persona que interactúa con la aplicación desarrollada.

El cliente receptara todas las necesidades que tengan todos los usuarios de la empresa

Por parte del jefe de Proyecto:

- Jefe de Proyecto: Gerente de la empresa líder del proyecto, su principal trabajo es planificar, optimizar, motivar y controlar el desarrollo de la aplicación.
- Desarrollador: Es la persona encargada de Elicitar, Diseñar, Desarrollar, y mantener la aplicación.
- Tester: Es quien se asegura que toda la aplicación funcione apropiadamente y de acuerdo con lo planificado por el cliente.

El jefe del proyecto recibe todas las observaciones y comentarios sobre la aplicación por parte de desarrolladores de software.

4.5.2 Sesiones de Elicitación

Se realizó una reunión con los stakeholders de la empresa INFORMEGA y los el jefe de proyecto y los desarrolladores que este caso actuó como ingeniero de requisitos y

desarrollador con el objetivo de tener un conocimiento completo de cuál fue la razón de ser de la aplicación.

Lluvia de Ideas

- La aplicación deberá funcionar con o sin internet.
- La aplicación deberá obtener la localización del mensajero mientras se traslada a cada uno de los puntos de la ruta definida a cada uno.
- La información de la localización de los mensajeros deberá ser enviada a una base de datos en la nube.
- La aplicación tendrá un formulario que permitirá registrar pedidos de los asociados en el dispositivo móvil.
- La aplicación deberá enviar todos los perdidos registrados en la aplicación a la base de datos en la nube.
- La aplicación deberá listar todos los registros de los pedidos y debe mostrar cuáles han sido enviados y cuáles no se han enviado a la base de datos en la nube.
- La aplicación debe contener los colores y logos corporativos de la empresa

Definición de Requisitos Generales

Basándonos en la recopilación de información mediante la lluvia de ideas, definidas en el ítem anterior, recolectado en la empresa NETLAB, explicado el proceso de logística indicado por el gerente de calidad de la empresa, a continuación, se identifica los requisitos de la aplicación de forma general.

- La aplicación deberá estar desarrollada únicamente para Android.

- La aplicación deberá obtener la localización del usuario con el objetivo de ser almacenado en la base de datos del dispositivo (SQLITE) para posteriormente cuando el dispositivo tenga internet se pueda enviar toda la información almacenada a una base de datos en la nube (MYSQL), la cual permitirá establecer la ruta del usuario en una Interface Web.
- La aplicación tendrá un formulario que permitirá registrar pedidos de los asociados en el dispositivo móvil, como muestra la Tabla 11, los mismos que deberán ser almacenados en el dispositivo para cuando el dispositivo posea internet estos registros se envíen a la base de datos en la nube.

Tabla 11
Campos del Formulario de Pedidos

Campos	Contenido
Código Asociado	Número de código de cliente asociado
Numero de Muestras	Número de muestras entregadas por parte del asociado (Campo numérico)
Actividad	Se selecciona la actividad a realizar: Retiro de Muestras, Entrega de Resultados, Entrega de Material, Banco, Entrega o Retiro de Documentos.
Guía	Se registra el número de guía que fue retirada. En este caso el número de muestras será llenado a la llegada a Recepción de Muestras
Asociado Password	Se digitan las iniciales (siglas) de la persona que realiza la entrega

- Los campos obligatorios del formulario serán:
    - Asociado
    - Numero de Muestras
    - Actividad
- La aplicación deberá listar todos los registros de los pedidos y debe mostrar cuáles han sido enviados y cuáles no se han enviado a la base de datos en la nube.
- La aplicación debe contener los colores y logos corporativos de la empresa.

4.6 Desarrollo de la Aplicación

Tomando en cuenta lo antes expuesto cabe recordar que el desarrollo de la aplicación se realizó de dos maneras, usando el método tradicional o nativo y con el uso de framework Híbrido en este caso específicamente Ionic Framework.

Basado en lo antes descrito en el capítulo 2.6, el desarrollo de la aplicación se realizó mediante la metodología **MDAM** la misma que inicia con las siguientes fases:

Dentro de la fase de análisis en el desarrollo de la aplicación móvil el primer paso es la elicitación de requerimientos la misma que se realizó en el capítulo 4.5.

4.6.1 Análisis

Características Técnicas

A continuación, se iniciará esta fase con la definición de las características técnicas necesarias para dicha implementación como se visualizará en la (Tabla 12).

Tabla 12
Características Técnicas

CARACTERISTICAS		TIPOS DE DESARROLLO MOVIL	
		NATIVO	HÍBRIDO (IONIC FRAMEWORK)
DESARROLLO	COMPUTADORA	HP PAVILION	HP PAVILION
	SO	WINDOWS 10	WINDOWS 10

Continúa

	RAM	8GB	8GB
	IDE	ECLIPSE JUNO	NA
	EDITOR DE CODIGO	NA	VISUAL STUDIO CODE
	SDK	ANDROID TARGET 24	ANDROID TARGET 25
	LENGUAJE DE PROGARAMACION	JAVA	JAVASCRIPT(ANGULAR, TYPESCRIPT), SASS
	SERVIDOR DE APLICACIONES	NA	NODE JS
	COMPILADOR EXTERNO	NA	CORDOVA
	FRAMEWORK	NA	IONIC
DISPOSITIVO	VERSION ANDROID MINIMA	>=5.0	>=4.1
	BASE DE DATOS	SQLITE	SQLITE
	GEOLOCALIZACION	GPS NATIVO	GPS NATIVO
RECURSOS EXTERNOS	SERVIDOR	CENTOS 6.9	CENTOS 6.9
	BASE DE DATOS	MYSQL 5.1	MYSQL 5.1
	WEB SERVICE	PHP	PHP

Diagrama de Casos de uso

En la (Figura 27) se muestra el diseño de los casos de uso identificados que necesitaremos en el desarrollo de la aplicación

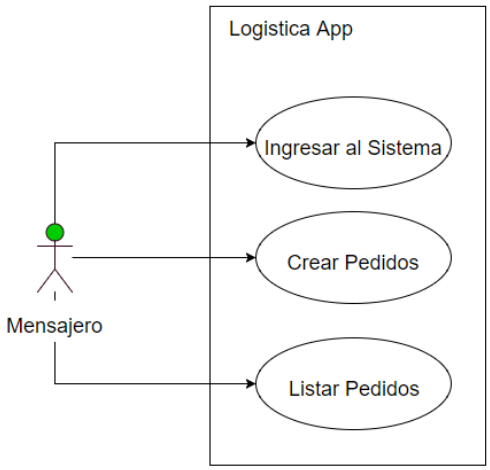


Figura 27 Diagrama de Casos de Uso

A continuación, se mostrará en la (Tabla 13), (Tabla 14), (Tabla 15) se visualiza el detalle de cada Caso de Uso antes expuesto.

Continúa

Tabla 13

Ingresar al Sistema

CASOS DE USO	Acceso al Sistema	RF1-1
ACTORES	Usuario (Mensajero)	
TIPO	ALTA/ESENCIAL	
REFERENCIAS	RF1-1	
PRE-CONDICIÓN	El usuario debe abrir la aplicación	
POST-CONDICIÓN	El usuario accedera a la pantalla principal de la aplicación	
AUTOR	Stalin Salgado	
PROPOSITO	Validar el ingreso a los usuarios registrados en la aplicación	

FLUJO NORMAL	PASO	ACCIÓN
	1	El usuario debe ingresar sus credenciales tanto usuario y contraseña
	2	El usuario debe dar clic en el boton "Ingresar"
	3	La aplicación enviara al usuario a la pantalla Principal
FLUJO ALTERNO	PASO	ACCIÓN
	3	Si el usuario y contraseña no son validos se mostrara un mensaje "Credenciales Invalidas"
FRECUENCIA	ALTA	
IMPORTANCIA	ALTA	
ESTADO	REVISADO Y TERMINADO	

Tabla 14

Crear Pedido

CASOS DE USO	Crear Pedido	RF1-2
ACTORES	Usuario (Mensajero)	
TIPO	MEDIA	
REFERENCIAS	RF1-2	
PRE-CONDICIÓN	El usuario debe haber registrado	
POST-CONDICIÓN	Almacenar el Registro	
AUTOR	Stalin Salgado	
PROPOSITO	Registrar los pedidos en la aplicación	

FLUJO NORMAL	PASO	ACCIÓN
	1	El usuario debe elegir el asociado en donde se encuentra
	2	El usuario ingresar el numero de muestras
	3	El usuario debe elegir la actividad que esta realizando
	4	El usuario debe ingresar el numero de guia si el tipo actividad es
	5	El usuario debe pedir a la personas que le entrega o recibe una
	6	El usuario debe dar clic en el boton "Guardar"
FLUJO ALTERNO	PASO	ACCIÓN
	6	Si el usuario no ingresa el asociado, el # de muestra, la actividad la aplicación no le va a dejar guardar el pedido
FRECUENCIA	ALTA	
IMPORTANCIA	ALTA	
ESTADO	REVISADO Y TERMINADO	

Tabla 15
Listar Pedidos

CASOS DE USO	Listar Pedidos	RF1-3
ACTORES	Usuario (Mensajero)	
TIPO	BAJA	
REFERENCIAS	RF1-3	
PRE-CONDICIÓN	El usuario debe haber registrado	
POST-CONDICIÓN	Vizualizar Listado de Pedidos	
AUTOR	Stalin Salgado	
PROPOSITO	Vizualizar Listado de Pedidos del dia	

FLUJO NORMAL	PASO	ACCIÓN
	1	El usuario debe dar clic en el boton "Ver Registros"
	2	La aplicación le enviara a una nueva pantalla con el Listado de
FLUJO ALTERNO	PASO	ACCIÓN
	-	Si el usuario quiere volver al formulario solo debe dar clic en el boton atrás propio de android
FRECUENCIA	BAJA	
IMPORTANCIA	BAJA	
ESTADO	REVISADO Y TERMINADO	

Cabe destacar que el análisis antes expuesto se lo puede usar con los dos tipos de desarrollo (Nativo e Híbrido).

4.6.2 Diseño

En esta fase se procedió a diseñar las interfaces preliminares de la aplicación, determinado tres pantallas principales Login, Formulario de Pedido, Listado de Registros, las mismas que a su vez se basan en los Casos de Uso detallados en el capítulo anterior como se visualiza en las (Figura 28).

Continúa

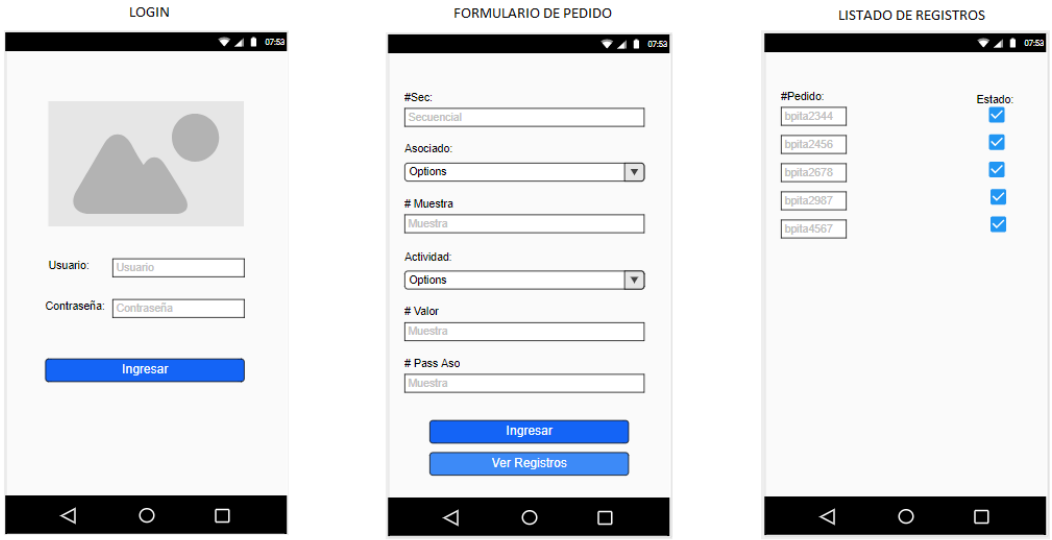


Figura 28 Diseño de Interface de la Aplicación

El diseño presentado en la Figura 28 se realizó en los tipos de desarrollo es decir en el modo nativo como en el modo Híbrido.

4.6.3 Desarrollo

En esta fase de procedió a escribir en cada una de los métodos de desarrollo establecido en esta investigación, es decir para el método nativo se usó JAVA como lenguaje de programación como en el método Híbrido se utilizó HTML5, JAVASCRIPT y CSS, y por lo antes expuesto en esta investigación se realizan dos implementaciones diferentes.

Desarrollo Nativo

Para el inicio del desarrollo en este método en específico cabe recordar que se usó como ide de desarrollo a Eclipse Juno y el SDK propio de Android y se estableció la siguiente estructura de Proyecto como se visualiza en la (Figura 29).

Continúa

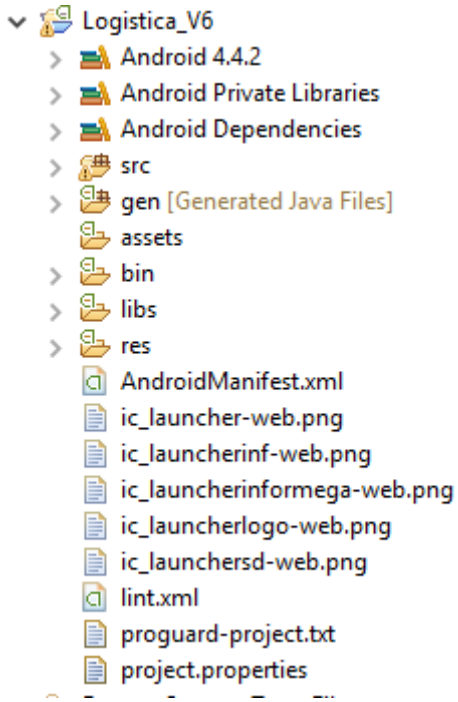


Figura 29 Estructura del Proyecto Nativo

Anteriormente en el Capítulo 2.1.1 se realizó una descripción de la estructura de un proyecto Android Nativo, razón por la cual solo nos centraremos en el directorio /src el cual contiene las clases java los cuales en este caso son los controladores de la aplicación, como también el directorio /res/layout donde está localizado las vistas de cada una de las pantallas antes descritas como son el Login, Formulario de Pedido como el Listado de Registros.

4.6.3.1.1 Librerías Utilizadas (Nativo)

- android.database.sqlite
    - Contiene las clases de administración de bases de datos SQLite que la aplicación usa para administrar su propia base de datos privada (Android Developers, 2017).

- La aplicación usa estas clases para administrar la base de datos. Se debe crear un proveedor de contenido el mismo que permita crear y administrar su propia base de datos para almacenar contenido (Android Developers, 2017).

- org.apache.http
  - Contiene las clases de administración de peticiones tan GET como POST para tener comunicación desde la base de datos privada (SQLITE), con la base de datos en la nube (MYSQL).
  - La aplicación usa estas clases para obtener datos de usuarios, asociados, actividades entre otros tipos información necesarias para la aplicación como también para él envió de geolocalización y el listado de pedidos registrados
- android.location
  - Contiene las clases de la API de que definen los servicios basados en la ubicación de Android y relacionados.
  - La aplicación usa estas clases para obtener datos de geolocalización (coordenadas X, Y) en tiempo real, utilizando el GPS del dispositivo.
- android.net
  - Contiene las clases que permiten acceder a la información de red del dispositivo.
  - La aplicación usa estas clases para realizar una validación en el momento de la petición GET o POST asía la base de datos en la nube.

4.6.3.1.2 Controladores (Nativo)

En el directorio /src tenemos las siguientes clases como se visualiza en la (Figura 30)

Continúa

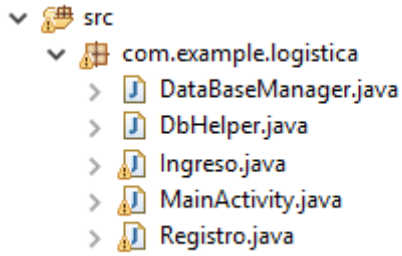


Figura 30 Directorio de Controladores

DataBaseManager.java

Esta clase contiene las operaciones de la base de datos local es decir la funcionalidad de SQLITE como es:

- Operaciones en la Base de Datos (CRUD)

DbHelper.java

Esta clase contiene la creación de la Base de Datos y sus respectivas tablas sin olvidarnos de su respectiva versión

Ingreso.java

Esta clase es controladora de la interface del Formulario de Pedido razón por la cual en esta clase se realiza el almacenamiento de los Pedidos realizados por el usuario. Además, contiene todas las librerías de geolocalización, las cuales con una subclase obtenemos la posición del usuario en tiempo real basándonos en un intervalo determinado de tiempo, dicha posición a su vez es almacenado en la base de datos local SQLITE.

MainActivity.java

Esta es la clase que es punto de inicio de la aplicación la cual tiene el siguiente flujo:

- Creación de la Base de Datos si no existiera
- Consumo del WebService
- Carga de la base de datos Local
- Visualización de la pantalla de Login
- Validación del Login con la base de datos Local

Registro.java

Esta clase es el controlador de la interface Listado de Registro, tiene como principal objetivo extraer todos los registros realizados por el usuario y mostrar su información de su estado.

4.6.3.1.3 Interfaces

En el directorio /res/layaout tenemos los siguientes archivos XML como se visualiza en la (Figura 31)

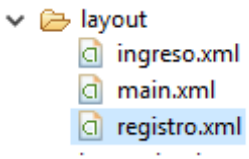


Figura 31 Directorio de Interfaces

Main.xml

Este Archivo contiene el diseño de la interface de la página de login, la misma que es la pantalla inicial de la aplicación.

Ingreso.xml

Este Archivo contiene el diseño de la interface de la página que contiene el formulario de pedido, la misma que es la pantalla principal de la aplicación.

Registro.xml

Este Archivo contiene el diseño de la interface de la página de listado de registros.

4.6.3.1.4 Resultado Final (Nativo)

A continuación, se visualizará las capturas de pantalla de las diferentes interfaces de la aplicación tomadas desde el dispositivo móvil en este caso una Tablet:

4.6.3.1.4.1 Login (Nativo)

Como se puede visualizar en la (Figura 32), tenemos la pantalla de login la misma que es el inicio de la aplicación, la misma que contiene el logo de la empresa, usuario y contraseña del mismo, estos dos últimos elementos son obligatorios para poder ingresar a la pantalla principal de la aplicación.

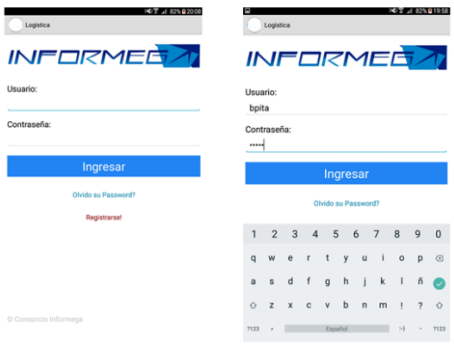


Figura 32 Login de la Aplicación Nativa

Además, en la Figura se muestra un ejemplo del ingreso de credenciales de la aplicación, en este caso es el usuario “bpita” y la contraseña que esta oculta debido a su tipo de input utilizado (password). Para terminar el proceso autentificación se debe dar clic en el botón Ingresar para poder ingresar a la pantalla principal de la aplicación.

4.6.3.1.4.2 Pantalla Principal (Nativo)

Ya realizado el proceso de autentificación de la aplicación, podremos ingresar a la pantalla principal de la aplicación con un pequeño mensaje de bienvenida y el usuario que se autentifico como se visualiza en la (Figura 33).

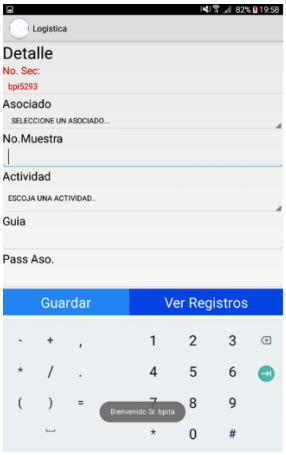


Figura 33 Pantalla Principal (Nativo)

A continuación, el usuario podrá realizar el registro de todos sus pedidos, usando el formulario que contiene la pantalla principal, el cual contiene los campos antes descritos en la fase de Elicitación de Requisitos:

- Secuencial
- Asociado

- No. Muestra
- Actividad
- Guía
- Pass Aso.

Tomando en cuenta que el usuario ya está dentro de la aplicación, inmediatamente se está ejecutando un proceso en segundo plano que está capturando la geolocalización en tiempo real del usuario mismo que es almacenado en la base de datos local (SQLITE) para posteriormente, por un proceso similar en segundo plano, dicha información está siendo enviada a la base de datos en nube (MYSQL).

En la (Figura 34) visualizaremos un ejemplo de la creación de un pedido, tomando en cuenta que los campos son obligatorios.

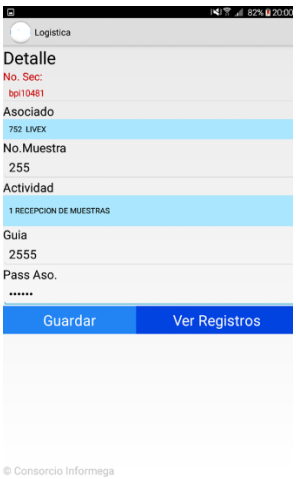


Figura 34 Ejemplo de Pedido (Nativo)

Para finalizar el proceso se debe dar clic en el botón “Guardar”, mismo que presentara un mensaje al usuario mostrando que el pedido se ha creado exitosamente.

4.6.3.1.4.3 Listado de Pedidos (Nativo)

Finalmente tenemos la pantalla de Listado de Pedidos, este módulo permite listar todos los pedidos del día actual como también el estado en el que esta es decir si ha sido enviado o no a la base de datos en la nube (MYSQL) como se visualiza en la (Figura 35).

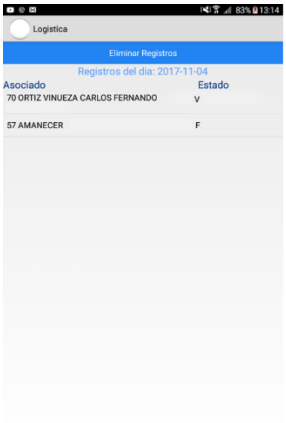


Figura 35 Listado de Pedidos (Nativo)

Desarrollo Híbrido

Para el inicio del desarrollo en este método en específico cabe recordar que se usó Ionic como Framework de desarrollo Híbrido, Visual Studio Code, Cordova y el SDK propio de Android y se estableció la siguiente estructura de Proyecto como se visualiza en la (Figura 36).

Continúa

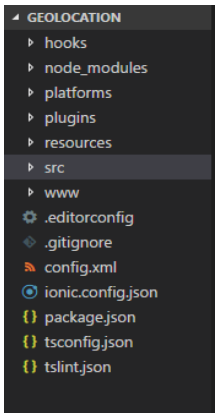


Figura 36 Estructura del Proyecto Híbrido

Anteriormente en el Capítulo 2.2.3.6 se realizó una descripción de la estructura de un proyecto Ionic (Híbrido), razón por la cual solo nos centraremos en el directorio /src el cual es el más importante en la estructura de un proyecto Ionic.

4.6.3.1.5 Librerías Utilizadas (Híbrido)

Las librerías en este tipo de Frameworks se los llama módulos, los mismos que forman parte de los plugins creados por Cordova y la cual permiten tener una interface de conexión para obtener acceso a los componentes nativos como pueden ser la cámara, base de datos interna entre otras.

Estas librerías deben ser instaladas en el proyecto ya que no vienen instaladas en la creación del proyecto, como lo es en el nativo en el cual solo con importar es suficiente, cabe rescatar que este proceso no tiene mayor dificultad, si nosotros accedemos a la documentación propia del Framework.

- SQLITE

  - Es la interface contenida en node modules que permite tener acceso a la base de datos de interna de la aplicación.

- GEOLOCATION
  - Es la interface contenida en node modules que permite tener acceso a la información de la geolocalización mediante el GPS de la aplicación.
- BACKGROUND GEOLOCATION
  - Es la interface que permite obtener la geolocalización en primer y segundo plano, es decir cuando la aplicación se está usando y cuando la aplicación es minimizada.
- HTTP MODULE
  - Es la interface que permite realizar la comunicación con la base de datos en la nube (MYSQL), es decir realizar las peticiones GET Y POST.

4.6.3.1.6 Estructura

La estructura de los proyectos mediante el uso del Ionic Framework utilizan el patrón MVC (modelo vista controlador), divido en tres subdirectorios /src/app , /src/pages y /src/providers anteriormente detallados en el Capítulo 2.2.3.6 .

El directorio app contiene a todo el proyecto y a su vez es el punto de inicio de la aplicación, es decir en este directorio se declararon todos los plugins necesarios, y se realiza todos los procesos previos a la apertura de la aplicación, en este caso específico en este directorio se realiza una petición GET al servidor para sincronizar la información a la base de datos en la nube (MYSQL) con la base de datos local (SQLITE).

El directorio pages como su nombre en inglés lo dice, contiene todas las páginas o pantallas de la aplicación con un cambio radical al desarrollo nativo. Cada página tienes su propia estructura MVC, además de tener su propio archivo de estilos, como se visualiza en la (Figura 37).

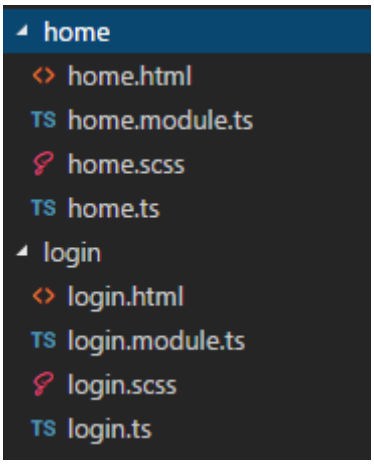


Figura 37 Estructura de Pantallas

Entonces en este escenario tenemos dos pantallas el Login, Home, pudiendo visualizar su estructura MVC.

El directorio providers contiene los controladores propios de cada uno de los plugins los mismos que son globales, es decir se pueden utilizar en cualquier pantalla anteriormente detallada en su controlador respectivo, para este caso en específico tenemos en la (Figura 38):

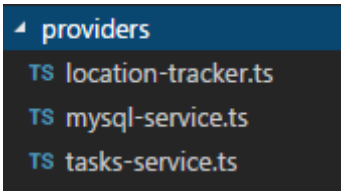


Figura 38 Providers

- Location_Tracker
    - Este controlador contiene todas las funciones necesarias para el manejo del plugin de geolocalización y su funcionamiento tanto en primero como en segundo plano.
- Mysql_service
    - Este controlador contiene todas las funciones que realizan las peticiones GET y POST según se lo requiera a la base de datos en la nube (MYSQL).
- Task_service
    - Este controlador contiene todas la funcionas para realizar operaciones en la base de datos local(SQLITE).

4.6.3.1.7 Resultado Final(Híbrido)

A continuación, se visualizará las capturas de pantalla de las diferentes interfaces de la aplicación tomadas desde el dispositivo móvil en este caso una Tablet:

4.6.3.1.7.1 Login(Híbrido)

Como se puede visualizar en la (Figura 39), tenemos la pantalla de login la misma que es el inicio de la aplicación, la misma que contiene el logo de la empresa, usuario y contraseña del mismo, estos dos últimos elementos son obligatorios para poder ingresar a la pantalla principal de la aplicación.

Continúa

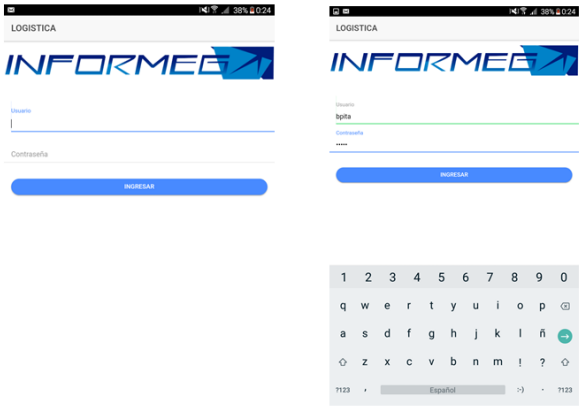


Figura 39 Login de la Aplicación Hibrida

Además, en la Figura se muestra un ejemplo del ingreso de credenciales de la aplicación idéntico a la aplicación nativa. Para terminar el proceso autentificación se debe dar clic en el botón Ingresar para poder ingresar a la pantalla principal de la aplicación.

4.6.3.1.7.2 Pantalla Principal (Híbrido)

Ya realizado el proceso de autentificación de la aplicación, podremos ingresar a la pantalla principal de la aplicación con un pequeño mensaje de bienvenida y el usuario que se autentifico como se visualiza en la (Figura 40).

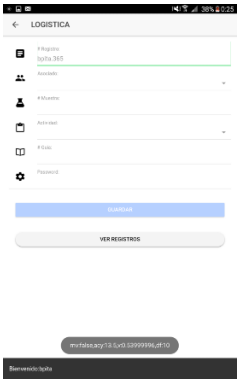


Figura 40 Login de la Aplicación (Híbrido)

A continuación, el usuario podrá realizar el registro de todos sus pedidos, usando el formulario que contiene la pantalla principal, el cual contiene los campos antes descritos en la fase de Elicitación de Requisitos:

- Secuencial
- Asociado
- No. Muestra
- Actividad
- Guía
- Password.

Tomando en cuenta que el usuario ya está dentro de la aplicación, se realiza el proceso idéntico de la aplicación nativa respecto a la captura y almacenamiento de la geolocalización.

En la (Figura 41) visualizaremos un ejemplo de la creación de un pedido, tomando en cuenta que los campos son obligatorios.

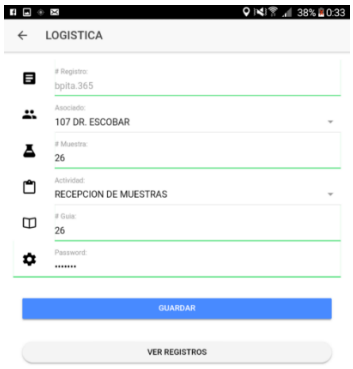


Figura 41 Ejemplo de Pedido (Híbrido)

4.6.3.1.7.3 Listado de Pedidos (Híbrido)

Finalmente tenemos la pantalla de Listado de Pedidos, este módulo permite listar todos los pedidos del día actual como también el estado en el que esta es decir si ha sido enviado o no a la base de datos en la nube (MYSQL) como se visualiza en la (Figura 42).

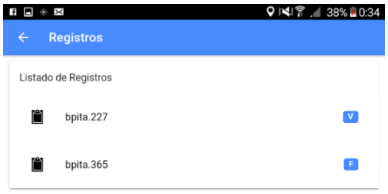


Figura 42 Listado de Pedidos (Híbrido)

CAPITULO V

BENCHMARKING APLICACIONES MOVILES NATIVAS VERSUS HÍBRIDAS

5.1 Introducción

Como se detalla en el Capítulo III, en donde implementamos el Modelo de Calidad IQMC, el cual nos permito establecer los parámetros de evaluación basados en la norma de eficiencia de comportamiento de la ISO 25000 y luego de haber realizado el desarrollo del caso de estudio planteado en el Capítulo IV, nos disponemos a evaluar la eficiencia en cada una de las aplicaciones, tanto la desarrollada en forma nativa como la realiza con frameworks híbridos en este caso con Ionic Framework.

5.2 Recopilación y Tabulación de Datos

Para la recopilación de datos se planteó tres escenarios de evaluación, basados en los indicadores obtenidos en el Capítulo III, los siguientes son:

- Tiempos de Respuestas
- Recursos de Software
- Recursos de la Aplicación

5.2.1 Tiempos de Respuestas

Sincronización de Datos

La Sincronización de Datos es la capacidad que tiene la Aplicación para comunicarse de manera bidireccional, es decir enviar datos desde el dispositivo hasta una base de datos en la nube en este caso MYSQL. Como también poder obtener datos de la base de datos para ser almacenados de forma local y poder usarla dentro de la aplicación.

En el momento de evaluar este sub escenario nos arrojó la siguiente (Figura 43)

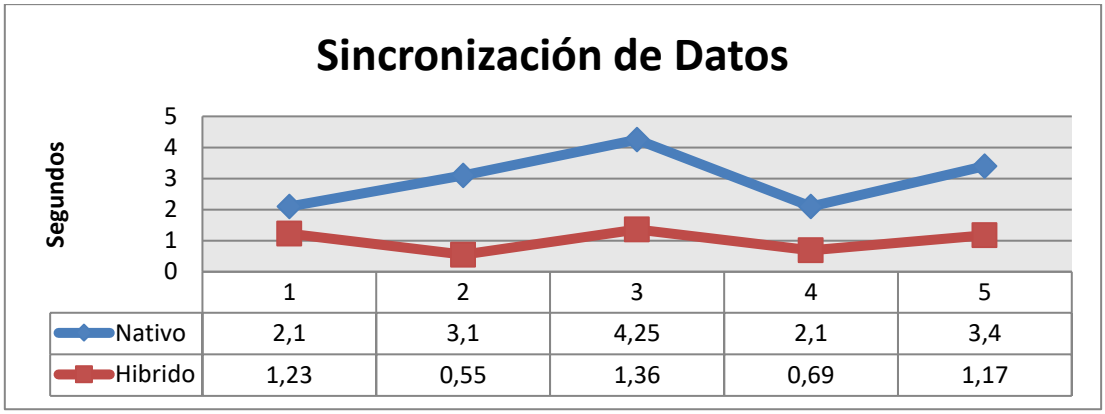


Figura 43 Sincronización de Datos (Nativo vs Híbrido)

Esta tabla nos permite visualizar los tiempos de respuesta, es decir cuánto tiempo demora cada aplicación en realizar esta funcionalidad antes detallada.

Instalación de la Aplicación

Este sub escenario como lo dice es el tiempo de respuesta que toman cada una de las aplicaciones para instalarse en un dispositivo móvil, como podemos visualizar en la (Figura 44).

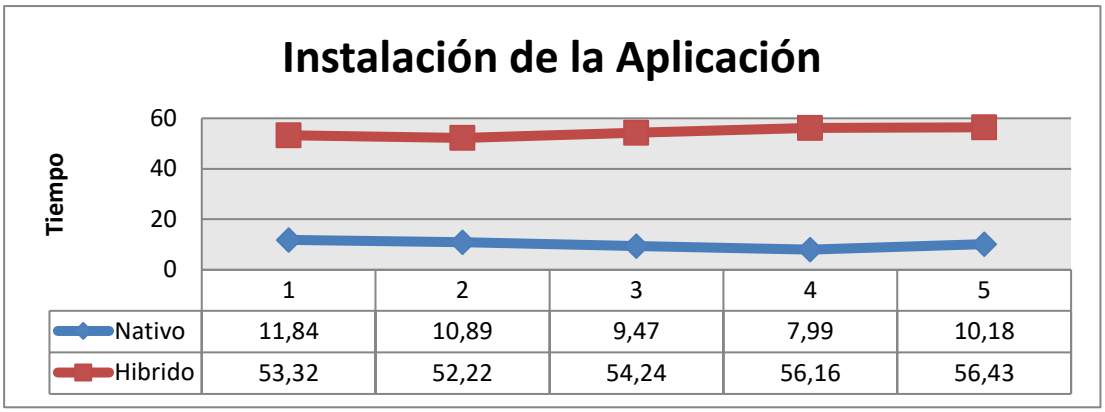


Figura 44 Instalación de la Aplicación (Nativo vs Híbrido)

Apertura e Inicio de la Aplicación

Este escenario plantea el tiempo de respuesta que se toma cada una de las aplicaciones para iniciarse, la misma que se visualiza en la (Figura 45).

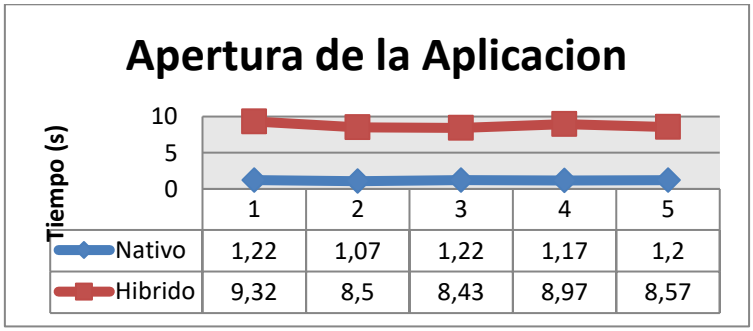


Figura 45 Apertura de la Aplicación (Nativo vs Híbrido)

Consulta a Base de Datos SQLite

Este escenario detalla los tiempos de respuesta que toma la aplicación para la consulta de datos de forma local es decir a SQLite, como se visualiza en la (Figura 46).

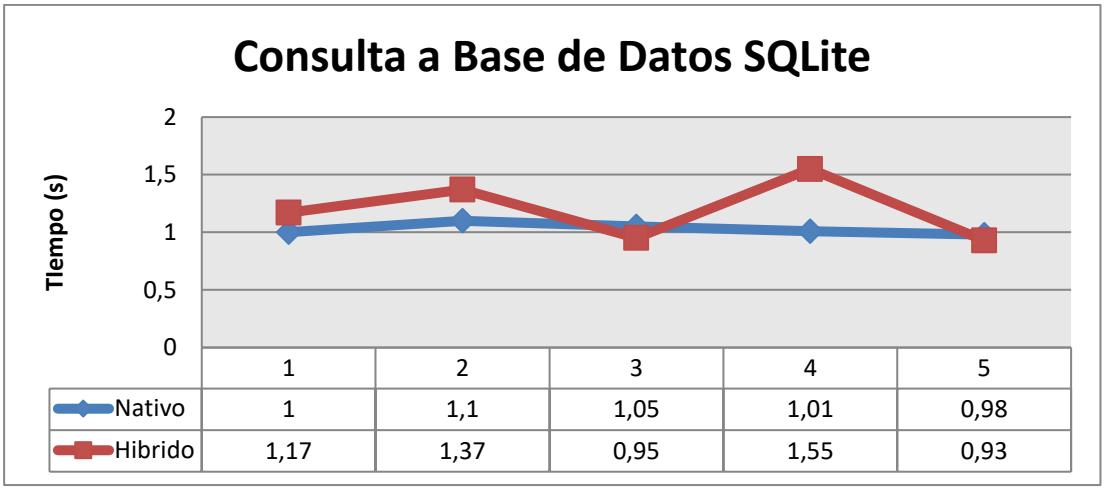


Figura 46 Consulta a la Base de Datos SQLite (Nativo vs Híbrido)

5.2.2 Recursos de Software

El escenario de Recursos de Software está formado por cuatro ítems, los cuales fueron identificados anteriormente en el Modelo IQMC en cual se identifica el número de JDK, Lenguajes de Programación, IDE o editores de Código y Librerías Utilizadas entre otras como se visualiza tanto en la (Tabla 16) como en la (Figura 47):

Continúa

Tabla 16
Recursos de Software

Tipo	JDK	Lenguajes de Programación	IDE o Editor de Código	SDK	SERVIDOR DE APLICACIONES	COMPILADOR EXTERNO	BASE DE DATOS	GPS
Nativo	1,8	JAVA	ECLIPSE LUNA	TARGET 24	NA	NA	SQLITE	GPS NATIVO
Híbrido	1,8	JAVASCRIPT	VISUAL STUDIO CODE	TARGET 26	NODE JS	CORDOVA	SQLITE	GPS NATIVO

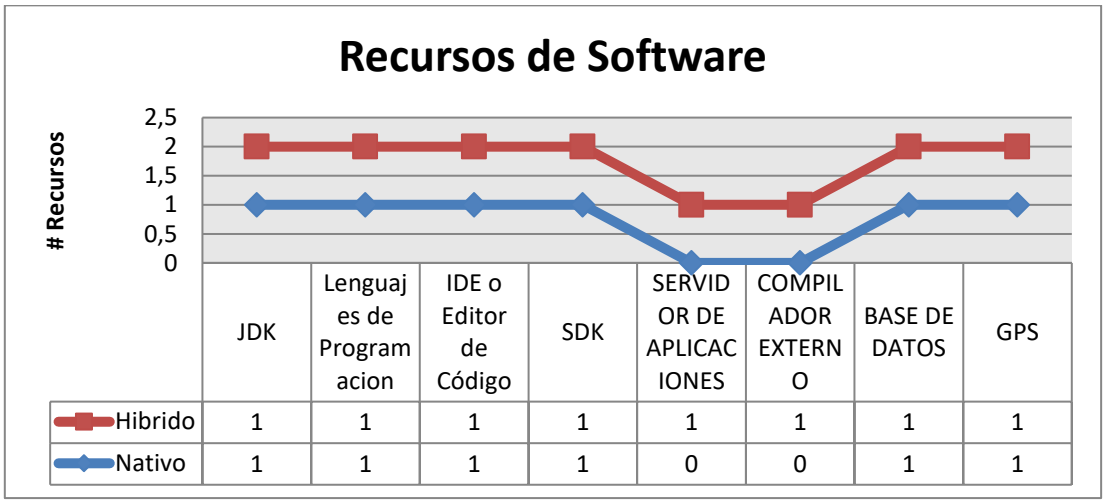


Figura 47 Recursos de Software Utilizado (Nativo vs Híbrido)

5.2.3 Recursos de la Aplicación

Como lo dice en el título de este capítulo este escenario evalúa y compara el uso de recursos de la aplicación propia, divida en el tamaño de aplicación como el consumo de varios componentes que usa la aplicación como veremos a continuación.

Tamaño de la Aplicación

En la (Figura 48) se visualiza el uso de disco y a su vez tamaño de aplicación instalada en un dispositivo móvil.

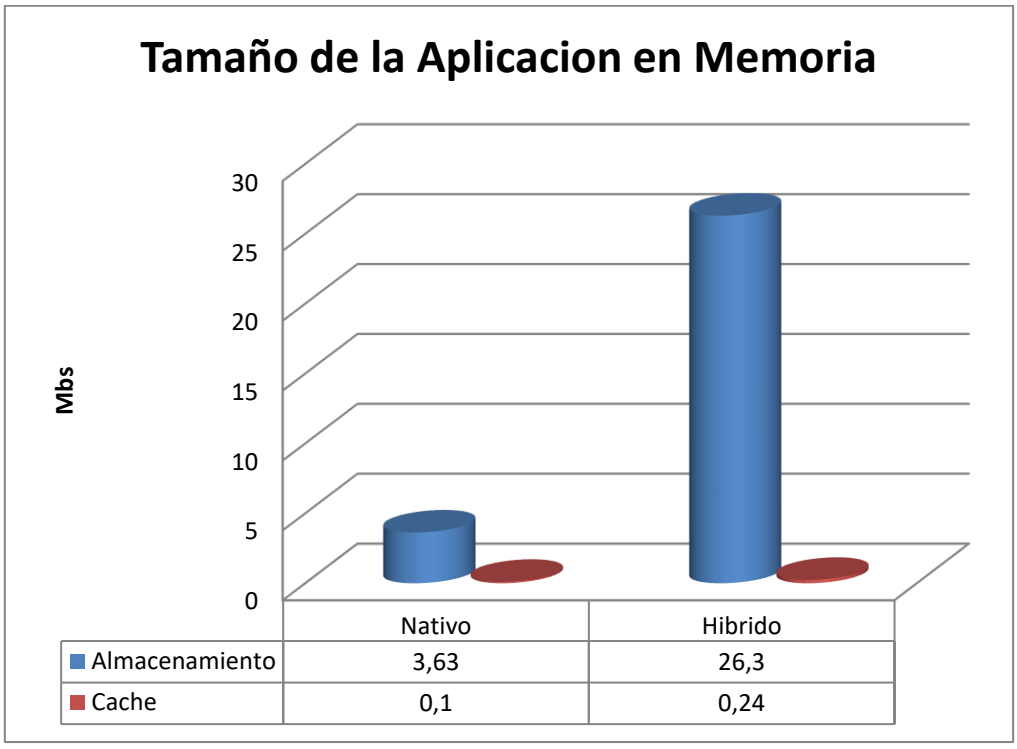
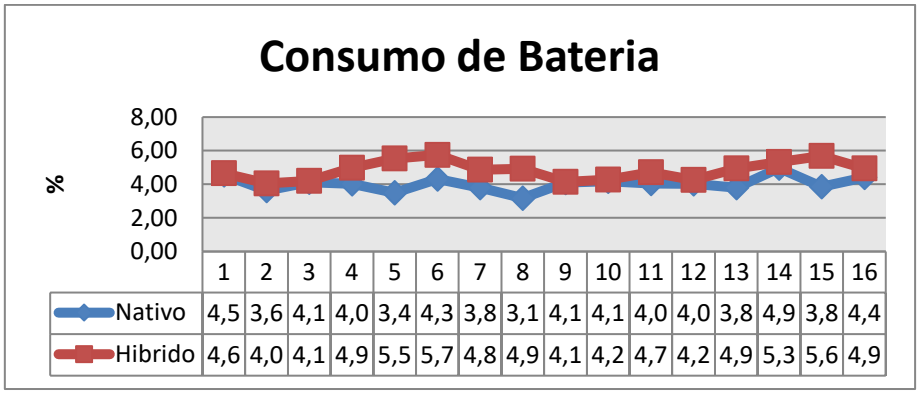


Figura 48 Tamaño de la Aplicación (Nativo vs Híbrido)

Consumo de Batería

Este escenario muestra el consumo de batería promedio respectivamente por cada técnica de desarrollo



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nativo	4,5	3,6	4,1	4,0	3,4	4,3	3,8	3,1	4,1	4,1	4,0	4,0	3,8	4,9	3,8	4,4
Hibrido	4,6	4,0	4,1	4,9	5,5	5,7	4,8	4,9	4,1	4,2	4,7	4,2	4,9	5,3	5,6	4,9

Figura 49 Consumo de Batería (Nativo vs Híbrido)

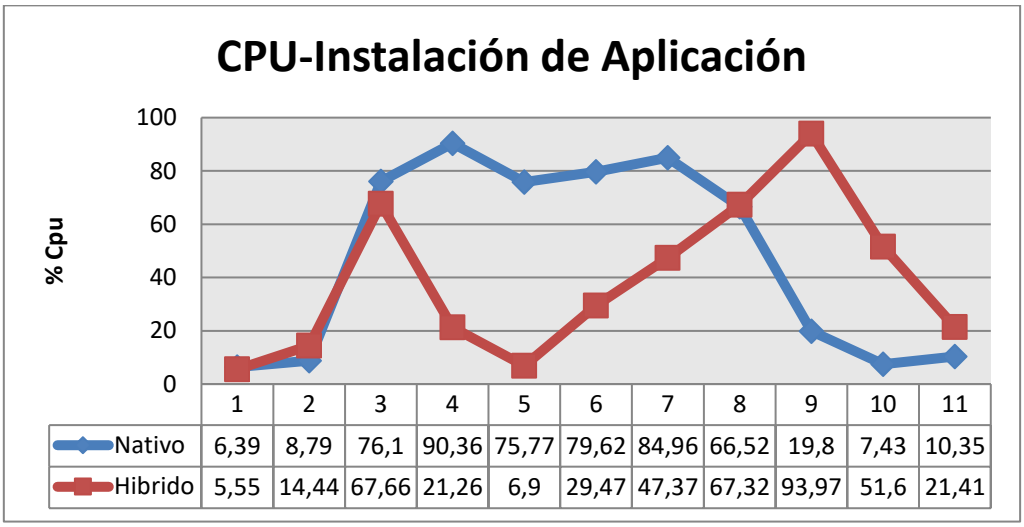
Consumo de Componentes

Este escenario es el más complejo para evaluar ya que se planteó varios sub escenarios los mismos que son los siguientes:

Continúa

5.2.3.1.1 Consumo de CPU

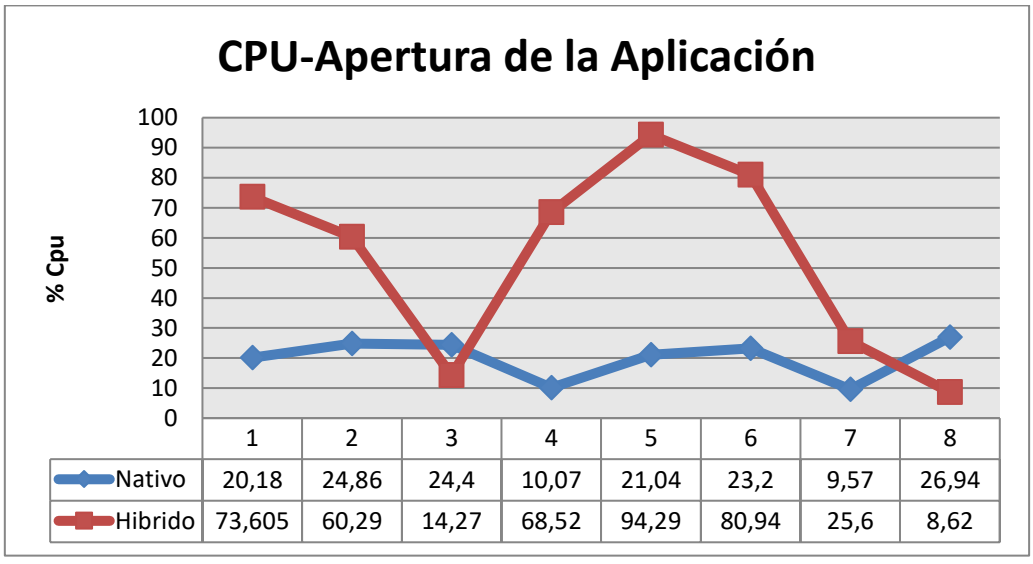
5.2.3.1.1.1 Consumo de CPU en la Instalación de la Aplicación



	1	2	3	4	5	6	7	8	9	10	11
Nativo	6,39	8,79	76,1	90,36	75,77	79,62	84,96	66,52	19,8	7,43	10,35
Hibrido	5,55	14,44	67,66	21,26	6,9	29,47	47,37	67,32	93,97	51,6	21,41

Figura 50 CPU-Instalación de Aplicación (Nativo vs Híbrido)

5.2.3.1.1.2 Consumo de CPU en la Apertura de la Aplicación



	1	2	3	4	5	6	7	8
Nativo	20,18	24,86	24,4	10,07	21,04	23,2	9,57	26,94
Hibrido	73,605	60,29	14,27	68,52	94,29	80,94	25,6	8,62

Figura 51 CPU-Apertura de la Aplicación (Nativo vs Híbrido)

5.2.3.1.1.3 Consumo de CPU en la Funcionamiento General

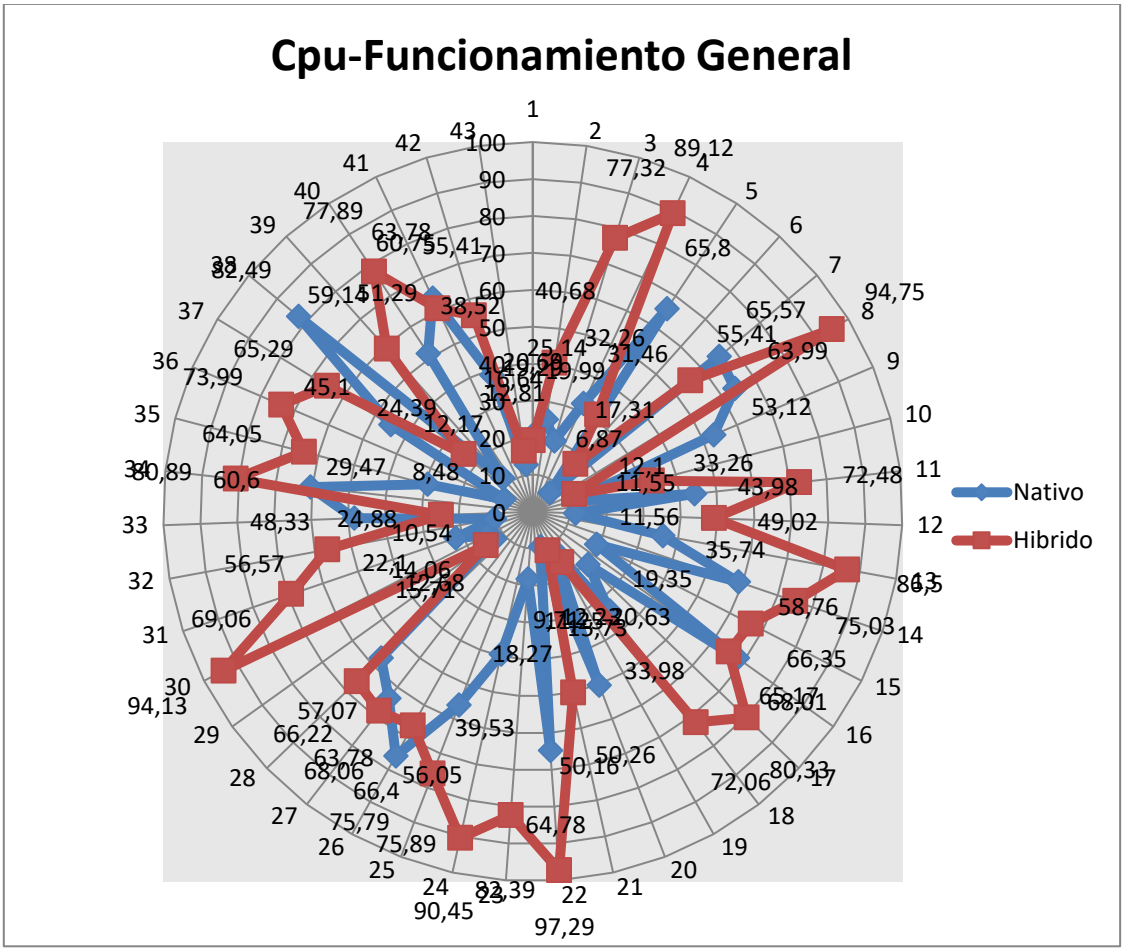


Figura 52 CPU-Apertura de la Aplicación (Nativo vs Híbrido)

5.2.3.1.1.4 Consumo de CPU-Sincronización y Consulta SQLite

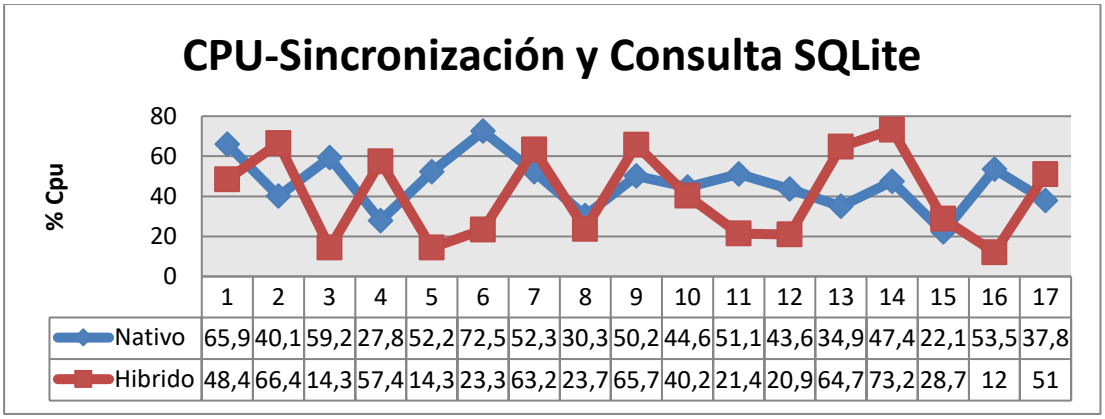


Figura 53 CPU- Sincronización y Consulta SQLite (Nativo vs Híbrido)

5.2.3.1.2 Consumo de Wifi

5.2.3.1.2.1 Consumo de Wifi -Instalación de Aplicación

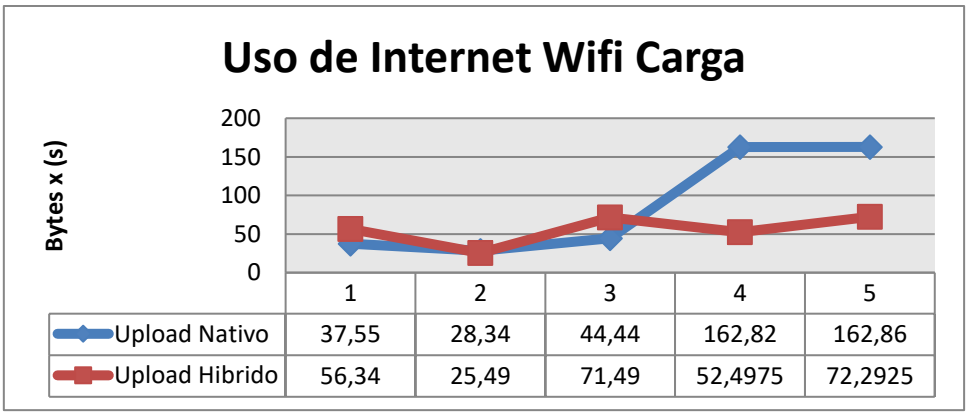


Figura 54 Wifi -Instalación de Aplicación (Carga) (Nativo vs Híbrido)

Continúa

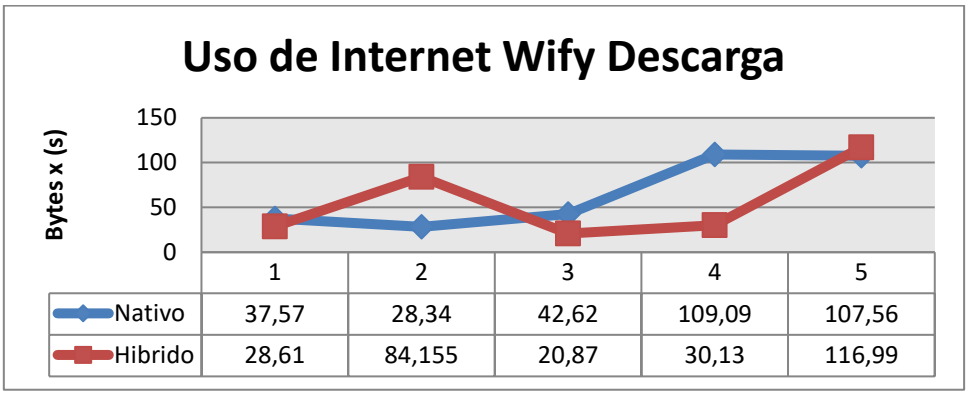


Figura 55 Wifi - Instalación de Aplicación (Descarga) (Nativo vs Híbrido)

5.2.3.1.2.2 Consumo de Wifi –Apertura de Aplicación

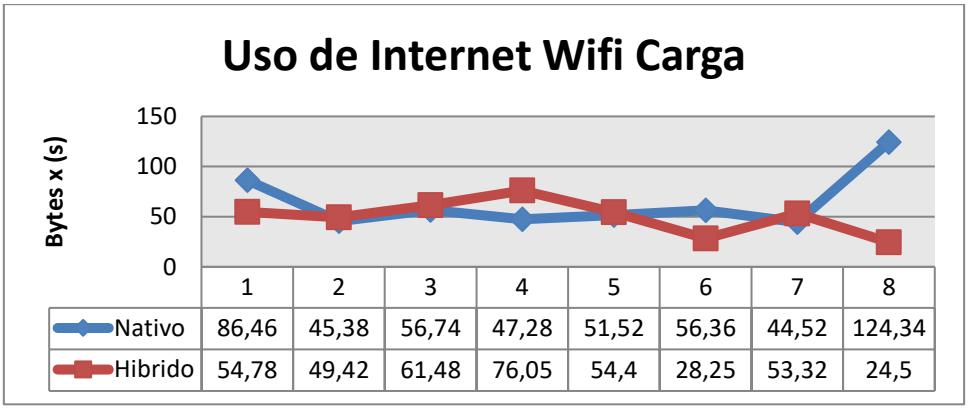


Figura 56 Wifi - Apertura de Aplicación (Carga) (Nativo vs Híbrido)

Continúa

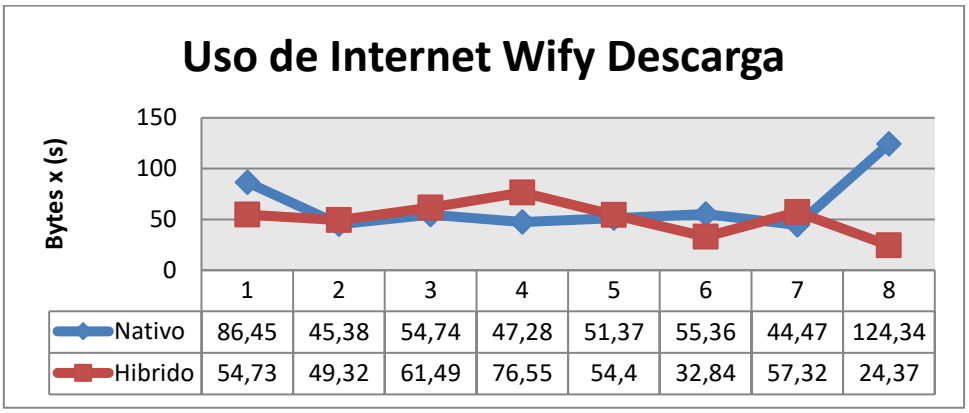


Figura 57 Wifi - Apertura de Aplicación (Descarga) (Nativo vs Híbrido)

5.2.3.1.2.3 Consumo de Wifi –Funcionamiento General

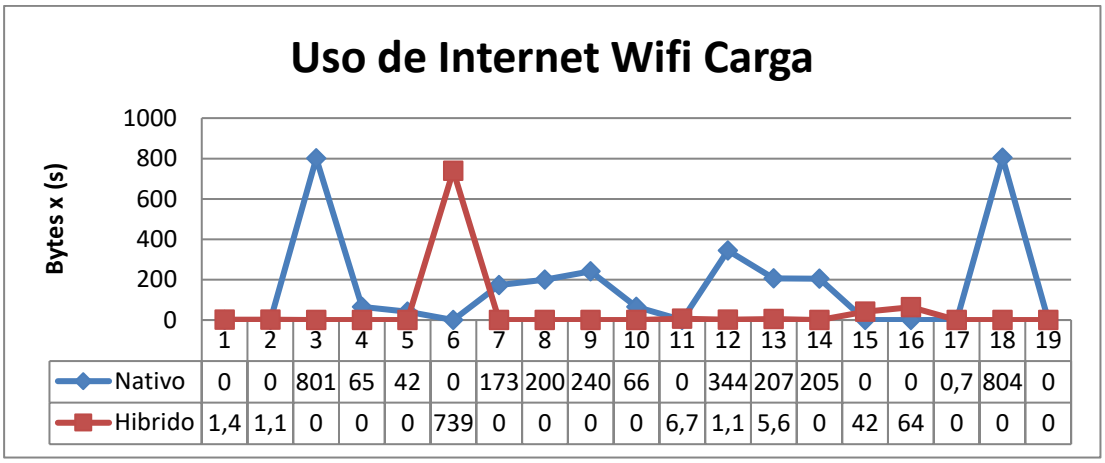
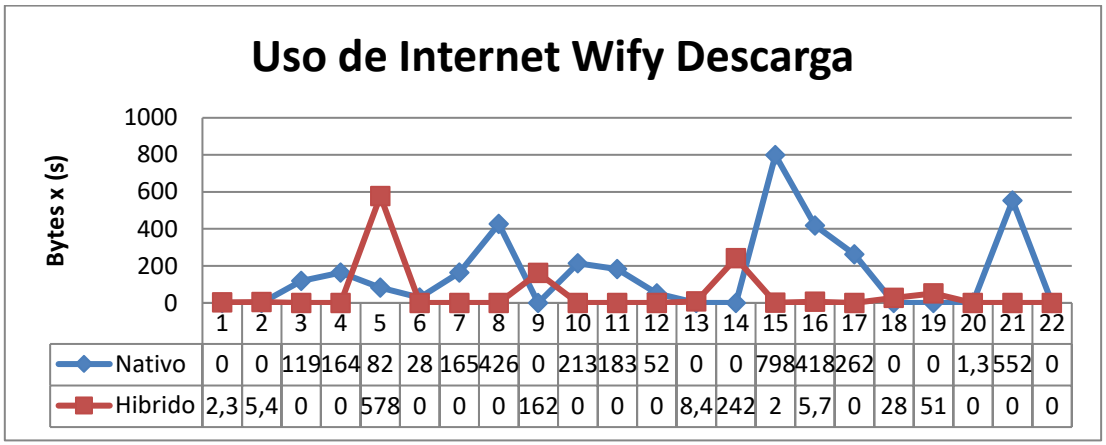


Figura 58 Wifi - Funcionamiento General (Carga) (Nativo vs Híbrido)

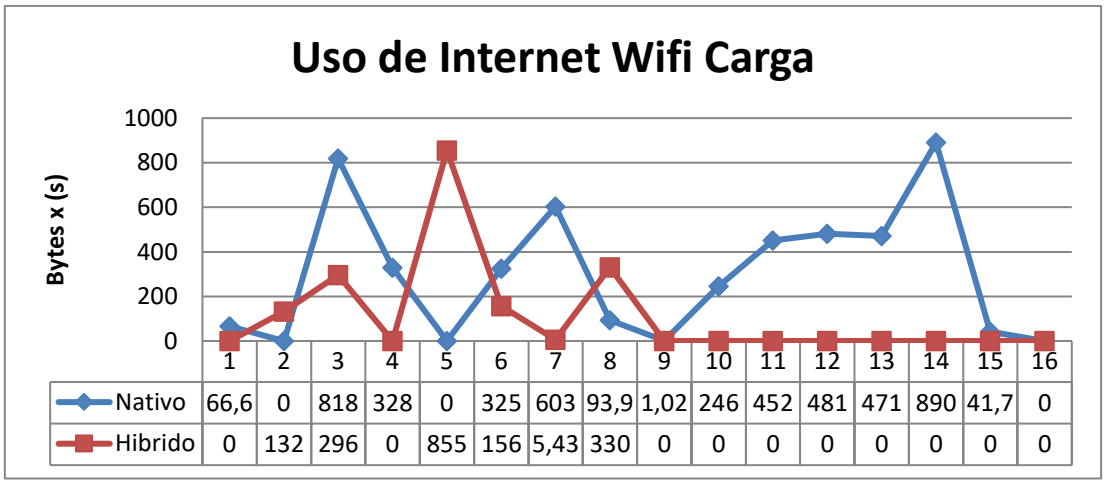
Continúa



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Nativo	0	0	119	164	82	28	165	426	0	213	183	52	0	0	798	418	262	0	0	1,3	552	0
Hibrido	2,3	5,4	0	0	578	0	0	0	162	0	0	0	8,4	242	2	5,7	0	28	51	0	0	0

Figura 59 Wifi - Funcionamiento General (Descarga) (Nativo vs Híbrido)

5.2.3.1.2.4 Consumo de Wifi – Sincronización y consulta SQLite



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nativo	66,6	0	818	328	0	325	603	93,9	1,02	246	452	481	471	890	41,7	0
Hibrido	0	132	296	0	855	156	5,43	330	0	0	0	0	0	0	0	0

Figura 60 Wifi - Sincronización y Consulta SQLite (Carga) (Nativo vs Híbrido)

Continúa

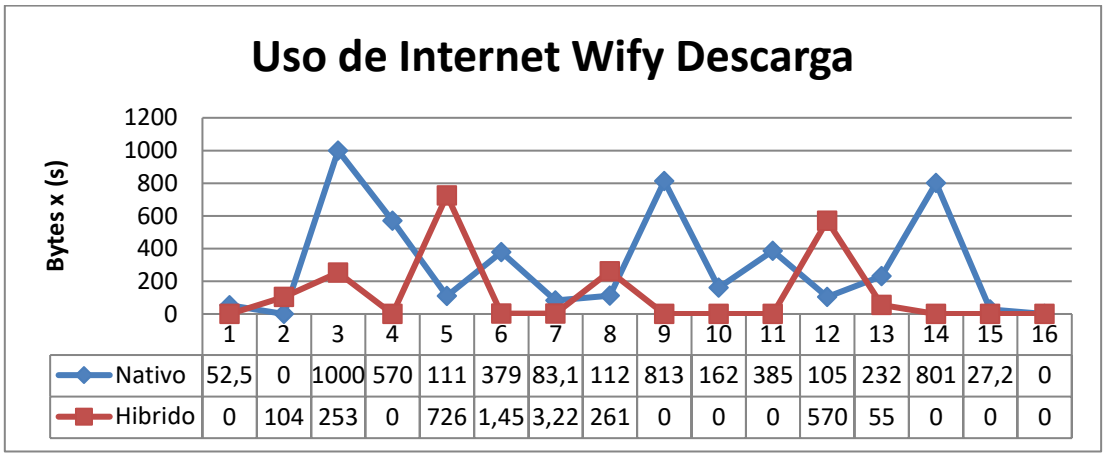


Figura 61 Wifi - Sincronización y Consulta SQLite (Descarga) (Nativo vs Híbrido)

5.2.3.1.3 Consumo de Datos (Internet Móvil)

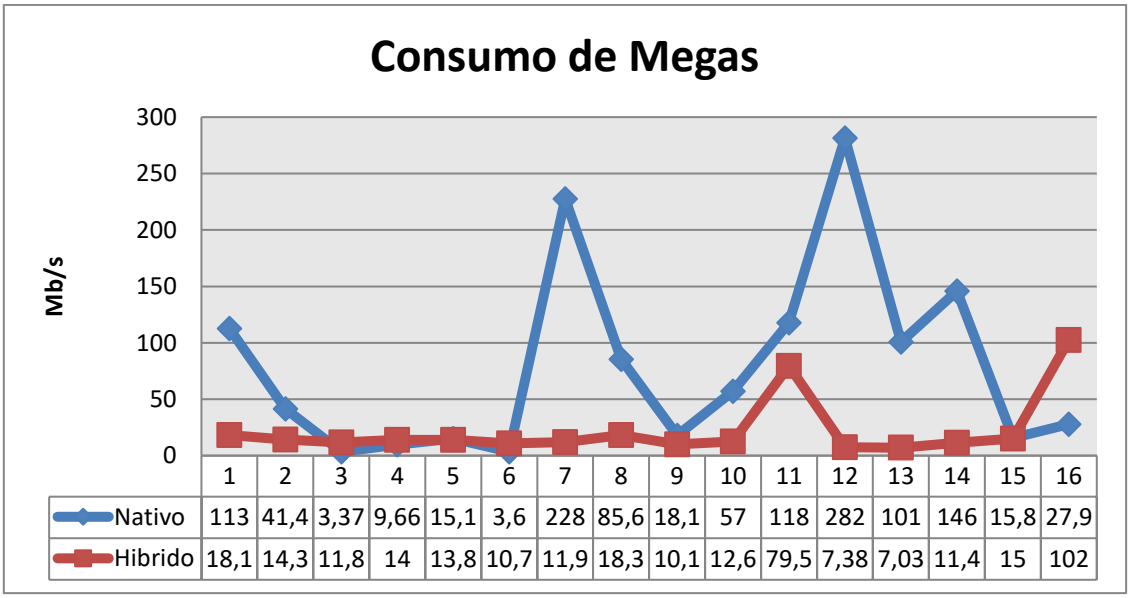


Figura 62 Consumo de Datos (Nativo vs Híbrido)

5.2.3.1.4 Consumo de RAM

5.2.3.1.4.1 Consumo de RAM - Instalación de Aplicación

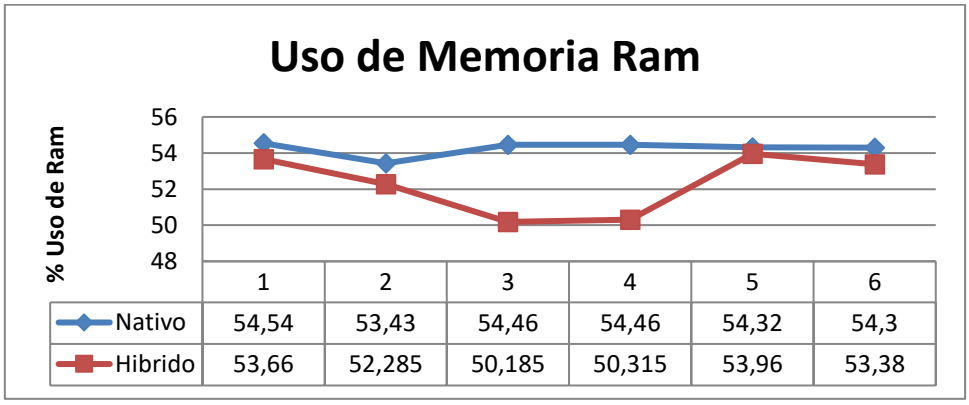


Figura 63 RAM - Instalación de Aplicación (Nativo vs Híbrido)

5.2.3.1.4.2 Consumo de RAM – Apertura de la Aplicación

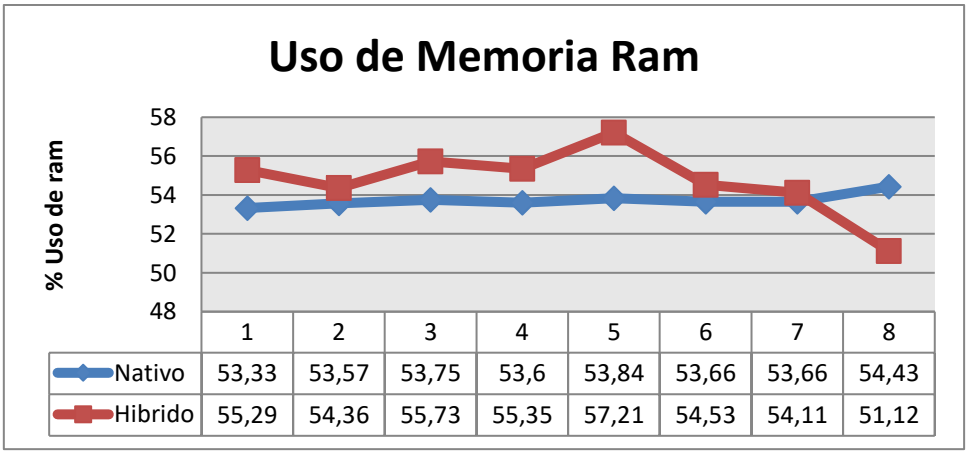


Figura 64 RAM - Apertura de la Aplicación (Nativo vs Híbrido)

5.2.3.1.4.3 Consumo de RAM – Funcionamiento General

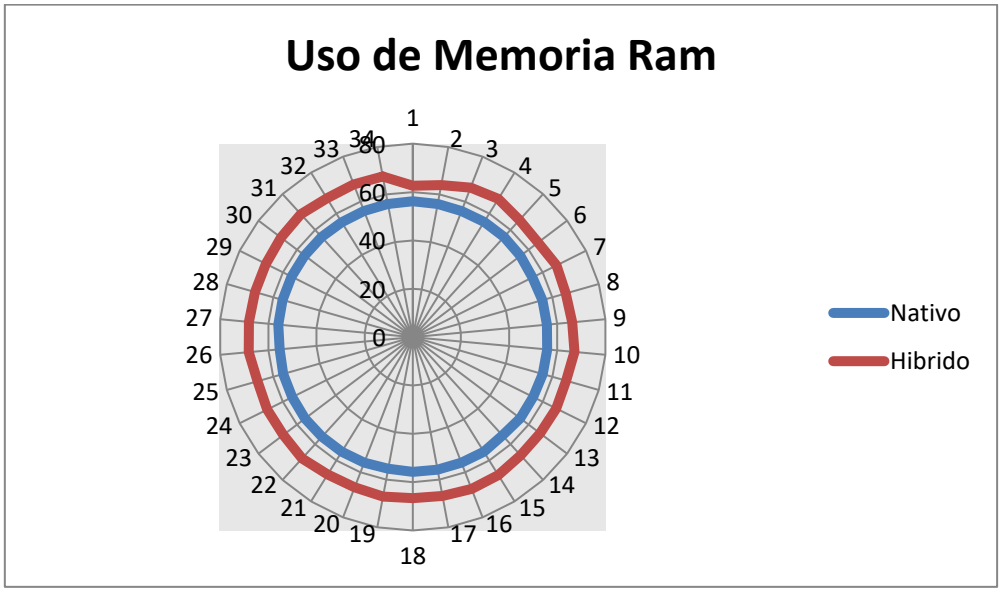
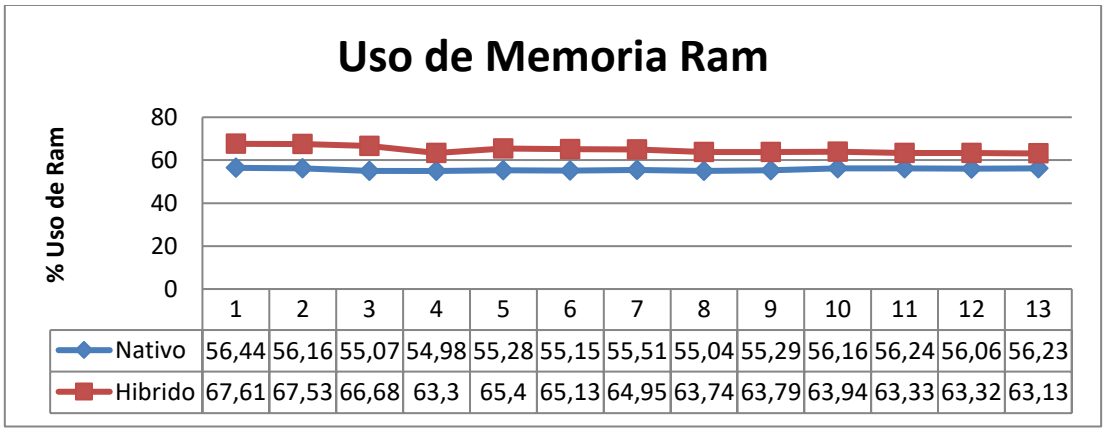


Figura 65 RAM - Funcionamiento General (Nativo vs Híbrido)

5.2.3.1.4.4 Consumo de RAM – Sincronización y Consulta Base de Datos



	1	2	3	4	5	6	7	8	9	10	11	12	13
Nativo	56,44	56,16	55,07	54,98	55,28	55,15	55,51	55,04	55,29	56,16	56,24	56,06	56,23
Hibrido	67,61	67,53	66,68	63,3	65,4	65,13	64,95	63,74	63,79	63,94	63,33	63,32	63,13

Figura 66 RAM - Sincronización y Consulta Base de Datos (Nativo vs Híbrido)

5.2.3.1.5 Consumo de Disco

5.2.3.1.5.1 Consumo de Disco -Instalación de Aplicación

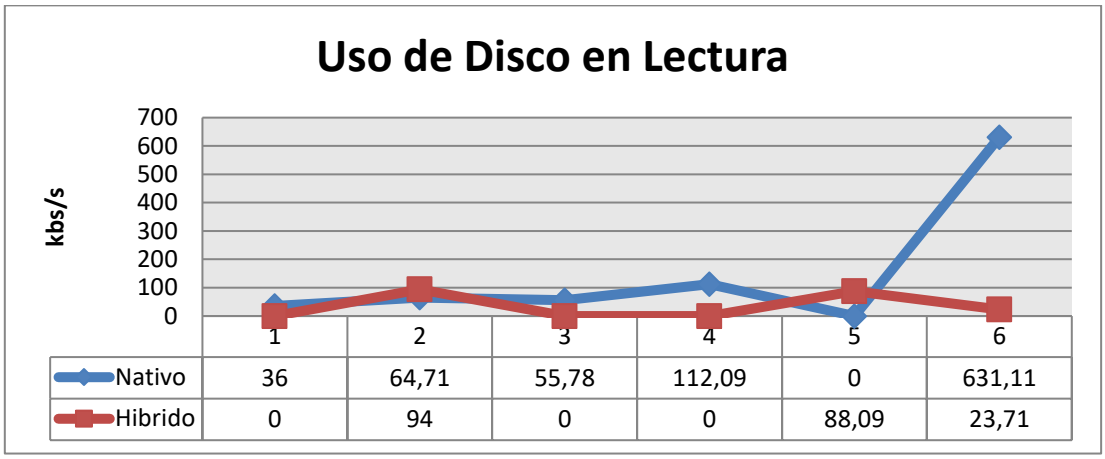


Figura 67 Consumo de Disco- Instalación de la Aplicación (Lectura) (Nativo vs Híbrido)

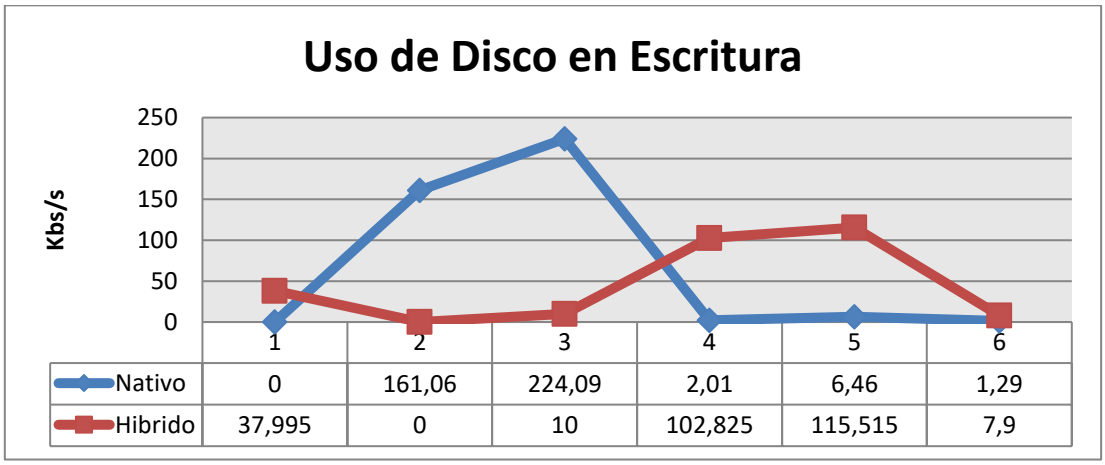


Figura 68 Consumo de Disco- Instalación de la Aplicación (Escritura) (Nativo vs Híbrido)

5.2.3.1.5.2 Consumo de Disco –Apertura de la Aplicación

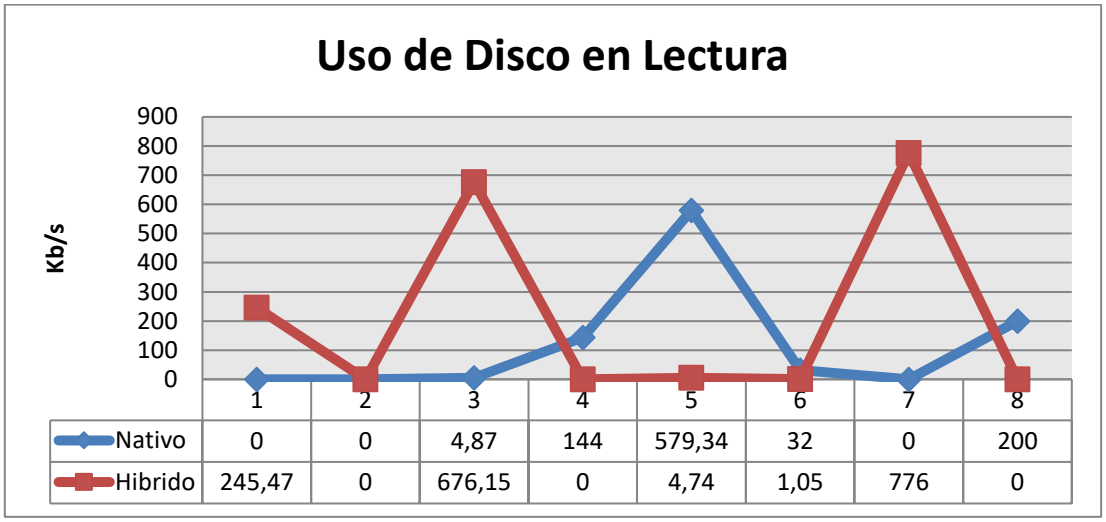


Figura 69 Consumo de Disco- Apertura de la Aplicación (Lectura) (Nativo vs Híbrido)

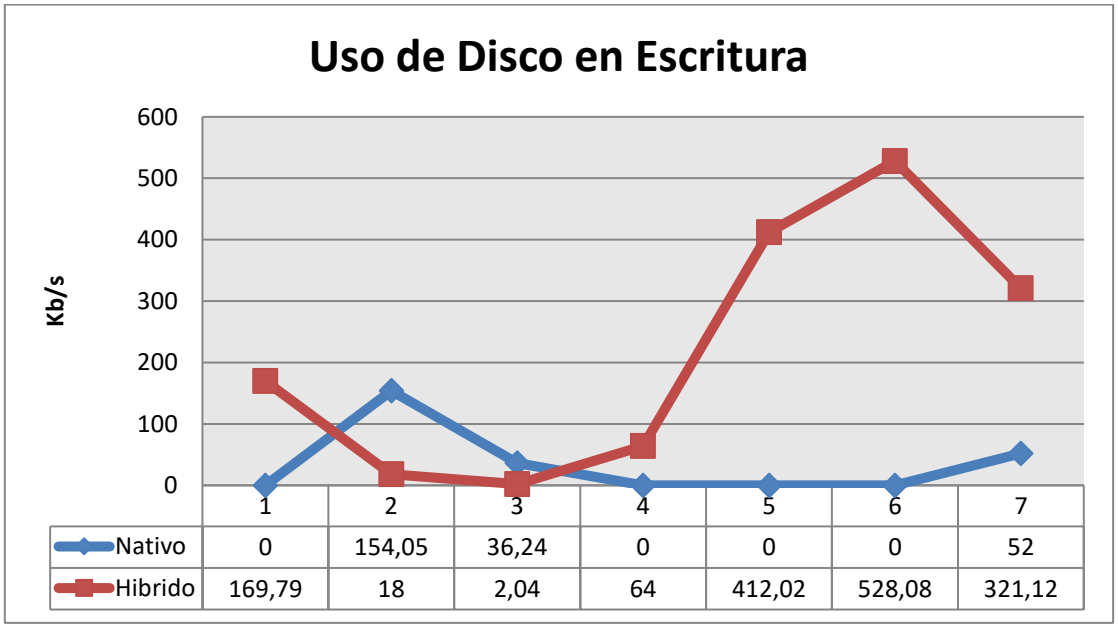


Figura 70 Consumo de Disco- Apertura de la Aplicación (Escritura) (Nativo vs Híbrido)

5.2.3.1.5.3 Consumo de Disco –Funcionamiento General

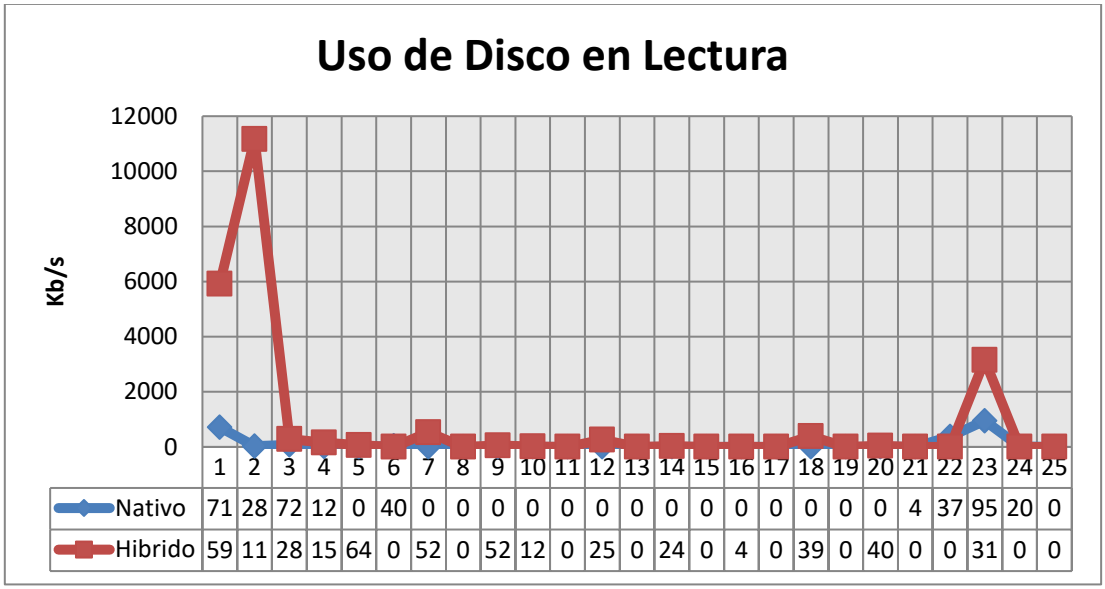


Figura 71 Consumo de Disco- Funcionamiento General (Lectura) (Nativo vs Híbrido)

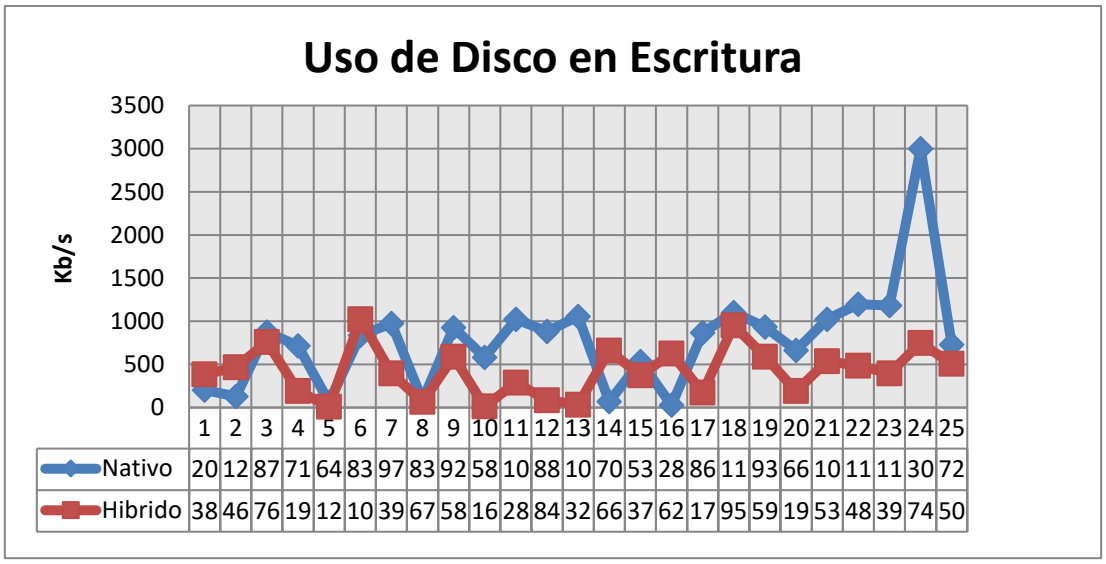


Figura 72 Consumo de Disco- Funcionamiento General (Escritura) (Nativo vs Híbrido)

5.2.4 Capacidad de la Aplicación

Este escenario se basa específicamente dos requisitos imprescindibles de la aplicación las mismas que son

- Capacidad de Almacenamiento
- Capacidad de Funcionamiento Offline

A continuación, se mostrará en la (Figura 73) la evaluación de estas dos capacidades determinadas

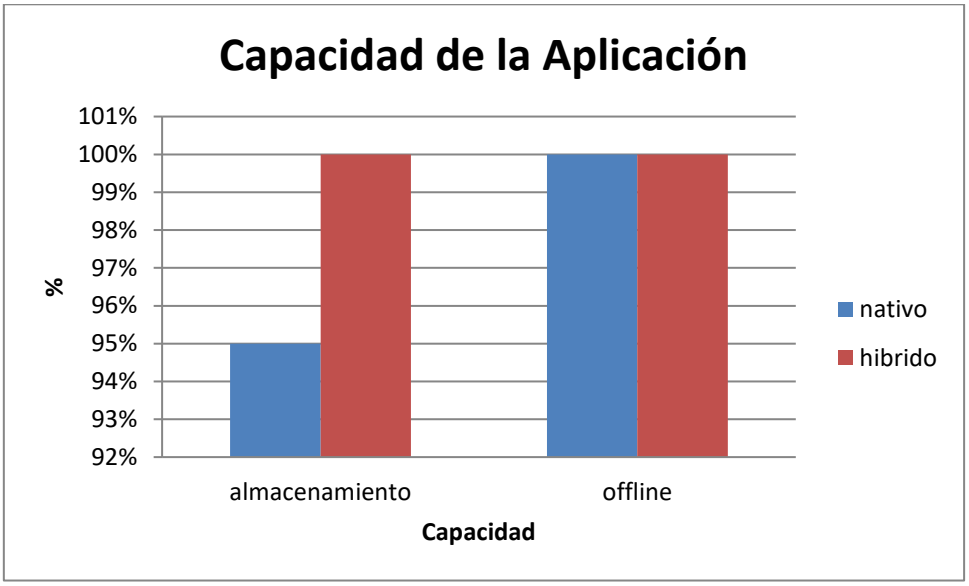


Figura 73 Capacidad de la Aplicación (Nativo vs Híbrido)

5.3 Evaluación del Comportamiento Temporal de la Aplicación

Para la evaluación del Comportamiento Temporal de la Aplicación cabe recordar que en Capitulo 3.2.4 se determinó un modelo de evaluación, el mismo que a continuación usare para asignar una métrica a cada ítem que se va a evaluar cómo se visualiza en la (Tabla 17).

Tabla 17
Evaluación de la Eficiencia de Comportamiento

EFICIENCIA DE COMPORTAMIENTO								
CARACTERISTICAS SUBCARACTERISTICAS ATRIBUTO				MÉTRICA	IMPORTANCIA	NATIVO	HÍBRIDO	JUSTIFICACIÓN
1	COMPORTAMIENTO EN EL TIEMPO							
	1.1	TIEMPO DE RESPUESTA						
		1.1.1	SINCRONIZACIÓN DE DATOS MYSQL Y SQLITE	B=1/ M=0,66/ A=0,33	2	M	B	Como se visualiza en la Figura 43 el pico más alto de nativo es 4,26 mientras que Híbrido es 1,36 segundos

Continúa

		1.1.2	INSTALACIÓN APLICACIÓN	B=1/ M=0,66/ A=0,33	1	B	A	Como se visualiza en la Figura 44 el tiempo de instalación es 55 segundos en el modo Híbrido vs 4,25 minutos en el modo nativo
		1.1.3	INICIO DE LA APLICACIÓN	B=1/ M=0,66/ A=0,33	2	B	A	Como se visualiza en la Figura 45 el tiempo de apertura es 9,32 segundos en el modo Híbrido vs 1,22 en el modo nativo
		1.1.4	CONSULTA BASE DE DATOS SQLITE	B=1/ M=0,66/ A=0,33	2	B	B	Como se visualiza en la Figura 46 los tiempos de respuesta son muy similares ya que fluctúan entre 1 y 2 segundos

Continúa

2	UTILIZACIÓN DE RECURSOS						
	2.1	RECURSOS DE SOFTWARE					
		2.1.1	LENGUAJES DE PROGRAMACION	B=1/ M=0,66/ A=0,33	1	B	B	Como se visualiza en la Tabla 16 las dos técnicas utilizan un lenguaje de programación Java para Nativo y JavaScript para Híbrido
		2.1.2	IDE O EDITOR DE CÓDIGO	B=1/ M=0,66/ A=0,33	1	B	B	Como se visualiza en la Tabla 16 las dos técnicas utilizan un ide o editor de condigo, Eclipse para Nativo y Visual Studio Code para Híbrido
		2.1.3	LIBRERIAS	B=1/ M=0,66/ A=0,33	1	M	M	Como se visualiza en la Tabla 16 las dos técnicas

Continúa

								utilizan un lenguaje de programación Java para Nativo y JavaScript para Híbrido
	2.2	RECURSOS DE HARDWARE						
		2.2.1	TAMAÑO DE APLICACIÓN	B=1/ M=0,66/ A=0,33	2	B	A	Como se visualiza en la Figura 47 el tamaño de la aplicación es de 3,63 Mb en nativo versus 26,3 Mb en Híbrido
		2.2.2	CONSUMO					
			2.2.2.1 MEMORIA RAM	B=1/ M=0,66/ A=0,33	2	M	A	Como se visualiza en el ítem 5.2.3.2.4 Consumo de RAM el consumo de RA M en varios escenarios usa desde el 50%

Continúa

									de RAM como también puede llegar a picos de 68% los dos métodos
			2.2.2.2	BATERIA	B=1/ M=0,66/ A=0,33	2	B	M	Como se visualiza en el ítem 5.2.3.2 el consumo es muy parecido en los dos escenarios tanto Híbrido como nativo, aunque se puede concluir que el Híbrido consuma mayor consumo de batería
			2.2.2.3	CPU	B=1/ M=0,66/ A=0,33	2	A	A	Como se visualiza en el ítem 5.2.3.2.1 Consumo de CPU el consumo de RAM puede

Continúa

									llegar a picos de 90% para los dos métodos debido a que se ejecutan subtareas como dentro de funcionamiento general de la aplicación
			2.2.2.4	DISCO	B=1/ M=0,66/ A=0,33	2	B	B	Como se visualiza en el ítem 5.2.3.2.5 Consumo de Disco, el consumo puede llegar a picos de hasta 2mb/s en los dos métodos
			2.2.2.5	INTERNET	B=1/ M=0,66/ A=0,33	2	M	B	Como se visualiza en el ítem 5.2.3.2.2 y 5.2.3.2. 3 consumo de

Continúa

									Internet Wifi y Datos.
3	CAPACIDAD								
	3.1	ALMACENAMIENTO			B=0,33/ M=0,66/ A=1	2	M	A	Como se visualiza en el ítem 5.2.4 Capacidad de la Aplicación, en el modo nativo existe un 95 % de capacidad de almacenamiento debido a que llega un punto donde la app se relentiza debido a la demasiada cantidad de datos almacenados, mientras que en el Híbrido no existe limitación de almacenamiento

Continúa

	3.2	OFFLINE	B=0,33/ M=0,66/ A=1	3	B	A	Como se visualiza en el ítem 5.2.4 Capacidad de la Aplicación, en el modo Híbrido como en el nativo existe un 100 % de capacidad de funcionamiento en modo offline, es decir la aplicación no deja de trabajar sino entra en un escenario sin internet

Para terminar la evaluación del Comportamiento temporal de la Aplicación se realizó la sumatoria de los puntos de Eficiencia obtenidos por cada tipo de desarrollo como se visualiza en la (Figura 74), de acuerdo a las métricas evaluadas anteriormente en la (Tabla 17).

Continúa

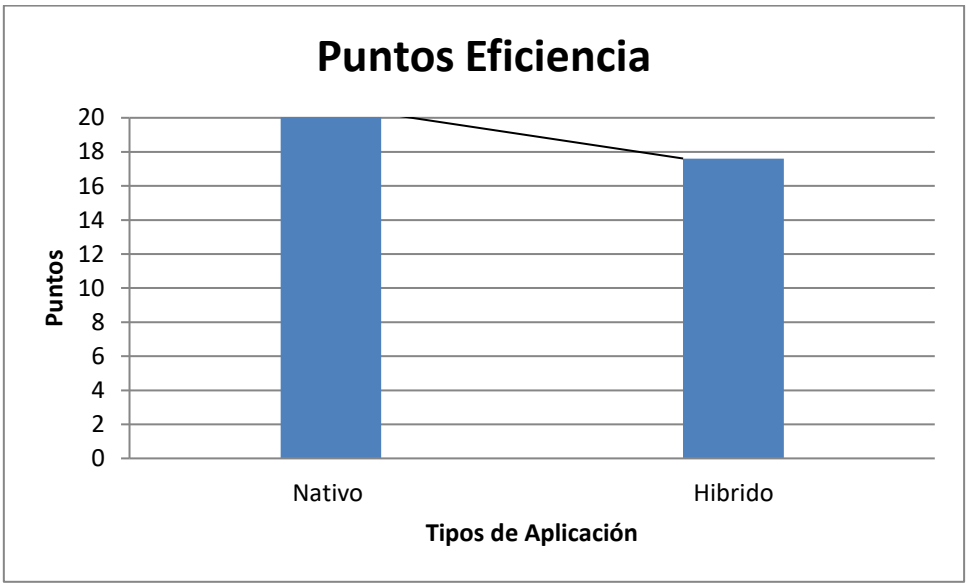


Figura 74 Puntos de Eficiencia (Nativo vs Híbrido)

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- En el análisis de herramientas que se realizó, se pudo identificar que no existe una herramienta especializada para medir indicadores de eficiencia en dispositivos móviles, más sin embargo se utilizó en este caso una aplicación que a su vez permita monitorizar el consumo de recursos (RAM,CPU, etc) y partiendo de este escenario se realizó la posterior recopilación de datos, lo que permitió a su vez se realice la evaluación de las dos aplicaciones móviles desarrolladas.
- En el Capítulo 1.5 referente al alcance de la Investigación se pidió responder a una pregunta la misma que cabe recordar es:

  ¿Cuál de las dos aplicaciones es más eficiente, basándose en la norma ISO25000, con un enfoque específico al uso de recursos como también en relación al comportamiento temporal de la aplicación?

  Para responder a la pregunta se consideró los resultados obtenidos en el Capítulo 5.3 Evaluación del Comportamiento Temporal de las dos aplicaciones, tanto con desarrollo Híbrido como nativo, dando como resultado que la aplicación nativa es más eficiente que la aplicación hibrida. Sin embargo, existen algunos ítems como:

  - Consumo de internet y la capacidad de almacenamiento la misma que en la aplicación con el método nativo llega a colgarse por la gran cantidad de

información que llega a almacenarse, característica que en el modo híbrido no sucede.

  - El algoritmo de localización en el desarrollo del caso de estudio, se implementó de diferente manera debido a que en el método Híbrido de Ionic Framework provee de plugins específicos para este requerimiento, el mismo que permite que el código y funcionalidad sea más óptimo tanto por las líneas de programación que se requiere en cada implementación como también en la facilidad de realizarlo y por consiguiente menos tiempo invertido en este requerimiento.

- Los Frameworks híbridos como Ionic Framework permiten desarrollar aplicaciones móviles, sin embargo, cabe recalcar que no permite utilizar todos los componentes de un dispositivo móvil, no obstante, cada día este tipo de Frameworks trata de ir mejorando y dando cada vez más componentes para ser usados por los desarrolladores y poder utilizar todos los componentes como en un desarrollo nativo.
- Al entregar las dos aplicaciones desarrolladas a Consorcio Informega, adjuntando a su vez los resultados obtenidos en la presente investigación realizada, y tomando en cuenta los ítems anteriormente descritos, la empresa y su Gerente, tomaron la decisión de usar en ambiente de producción la aplicación desarrollada con Ionic Framework debido a las bondades que presta este nuevo método de desarrollo como son:

  - Reutilización de código
  - Adaptabilidad dinámica a cambios y mejoras
  - Desarrollo Multiplataforma

6.2 Recomendaciones

- En futuras investigaciones se recomienda elaborar un prototipo ya sea de metodología o una herramienta que permita evaluar no solo la eficiencia de aplicaciones sino la calidad de las mismas.
- En futuros desarrollos se recomienda al desarrollador de software, que antes de escoger tecnologías hibridas para el desarrollo de aplicaciones móviles, debe realizar un análisis previo de componentes que va utilizar, verificando que dichos componentes estén habilitados en este caso en Ionic Framework, debido a que algunos aún no se los puede utilizar o están en versión Beta, lo cual evitará desarrollos fallidos.
- En futuros desarrollos se recomienda al desarrollador de software ,que antes de escoger tecnologías nativas para el desarrollo de aplicaciones móviles, se analice la complejidad que pueda existir en la implementación de determinada funcionalidad, ya que como vimos en la presente investigación se identificó que existen componentes específicos para determinadas funcionalidades en las tecnologías hibridas, que permiten implementar de manera más óptima ciertos componentes como en este caso el GPS comparado con la implementación nativa.

- En futuras investigaciones, se recomienda profundizar en el estudio de los plugins o componentes que ofrecen los frameworks híbridos, que permitan a los desarrolladores tener una base sólida de conocimiento en el momento de elegir una tecnología nativa o híbrida en el desarrollo de una aplicación móvil.

REFERENCIAS BIBLIOGRAFICAS

Alvarez, M. (02 de 03 de 2017). *desarrolloweb*. Obtenido de desarrolloweb: https://desarrolloweb.com/articulos/que-es-ionic2.html

Android. (19 de 06 de 2017). *Developer Android*. Obtenido de Developer Android: https://developer.android.com/guide/components/activities.html?hl=es-419

Android Developers. (01 de 04 de 2017). *developer.android.com*. Obtenido de developer.android.com: https://developer.android.com/reference/android/database/sqlite/package-summary.html

Apache Software Foundation. (02 de 02 de 2017). *Apache Cordova*. Obtenido de Apache Cordova: https://cordova.apache.org/docs/es/latest/guide/overview/

Bermeo, L., Campaña, M., & Melo, L. (2014). Analisis Comparativo de Frameworks Javascript: Jquery y Mootools, para la implementación de aplicaciones wen en la empresa sofya. Aplicación a un caso de estudio. *Repositorio Digital ESPE*.

Boxwell, R., Rubiera, I., & MacShane, B. (1995). *Benchmarking para competir con ventaja*. McGraw-Hill.

Cevallos, & Angulo, E. J. (2014). Case study on mobile applications UX: effect. *ETSI INFORMATICA*.

Charland, A., & LeRoux, B. (2011). Mobile Aplication Development: Web vs Native. *ACM*, 49-53.

Coral, C., Moraga, A., & Piattini, M. (2010). Calidad del Producto y Proceso de Software. Madrid. En C. Calero, *Calidad del Producto y Proceso de Software. Madrid: Ra-Ma* (pág. 665). Madrid: RA-MA.

Dimas, J. (13 de 03 de 2014). *desarrolloweb*. Obtenido de desarrolloweb: https://desarrolloweb.com/articulos/estructura-aplicacion-android-archivos-directorios.html

Drifty Co. (01 de 12 de 2016). *ionicframework*. Obtenido de ionicframework: https://ionicframework.com/docs/guide/preface.html

Drifty Co. (26 de 06 de 2016). *ionicframework*. Obtenido de ionicframework: https://ionicframework.com/docs/v1/guide/installation.html

Escoffier, C., & Lalanda, P. (2015). Managing the Heterogeneity and Dynamism. *IEEE*, 75-81.

ESPE. (2011). *UNIVERSIDAD DE LAS FUERZAS ARMADAS- ESPE*. Obtenido de http://www.espe.edu.ec/

Foundation, ©. 2. (01 de 01 de 2016). *eclipse.org*. Obtenido de eclipse.org: https://eclipse.org/modeling/emf/

Gasca, M., Camargo, L., & Medina, B. (2014). Metodología para el desarrollo. *Tecnura*, 20 - 35.

Gokhale, P., & Singh, S. (2014). Multi-platform Strategies, Approaches and. *IEEE*, 289-293.

Google. (26 de 06 de 2017). *angular.io*. Obtenido de angular.io: https://angular.io/docs

Google. (26 de 06 de 2017). *Angular.io*. Obtenido de Angular.io: https://angular.io/

Hale, M., & Hanson, S. (2015). A Testbed and Process for Analyzing Attack Vectors and Vulnerabilities in. *IEEE*, 181-188.

Heitkötter, Hanschke, & Majchrzak. (2013). Evaluating crossplatform. *Springer*, 120-138.

IBM Software. (2012). Native, web or hybrid. *WebSphere*, 2-10.

iso25000. (23 de 07 de 2017). *iso25000.* Obtenido de iso25000: http://iso25000.com/

ISO25000. (04 de 07 de 2017). *iso25000.com.* Obtenido de iso25000.com: http://iso25000.com/index.php/normas-iso-25000/iso-25010/21-eficiencia-de-desempeno

Joorabchi, Erfani, M., Mesbah, A., & Kruchten, P. (2013). Real Challenges in Mobile App Development. *IEEE*, 15-24.

Lella, A., & Lipsman, A. (2014). The U.S. Mobile App Report. *comScore Whitepaper*.

López, D. (02 de 02 de 2017). *mokware*. Obtenido de mokware: http://mokware.com.mx/blog/personalizar-icono-y-splash-en-ionic/

Malavolta, I., Ruberto, S., Soru, T., & Terragniz, V. (2015). Hybrid Mobile Apps in the Google Play Store:. *IEEE*, 36-59.

Molina, N. (26 de 06 de 2017). *desarrolloweb.* Obtenido de desarrolloweb: http://desarrolloweb.com/articulos/estructura-proyectoionic2

Node.js Foundation. (02 de 02 de 2017). *Node.js* . Obtenido de Node.js: https://nodejs.org/es/about/

Oliveira, W., Torres, W., Castor, F., & Ximenes, B. H. (2016). Native or Web? A Preliminary Study on the Energy. *IEEE*, 589-593.

Ramsin, R., & Rahimian, V. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. *IEEE*.

Rosso, R. (16 de Enero de 2018). *uptodown.com*. Obtenido de uptodown.com: https://simple-system-monitor.uptodown.com/android

StatCounter Global. (2014). Platform Comparison of Historical Web Views in the US From August 2012 to August 2014. *StatCounter Global*.