Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería de Sistemas

IMPLEMENTACIÓN DE FRAMEWORK PARA EL DESARROLLO DE APLICACIONES MÓVILES MULTIPLATAFORMA BASADO EN COMPONENTES JSON APLICANDO SOA

Tesis presentada por el bachiller:

López Corrales, Manuel Alejandro

para optar el Título Profesional:

Ingeniero de Sistemas con especialidad en Ingeniería de Software

Asesor:

Dr. Sulla Torres, José Alfredo

Arequipa - Perú

2022

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA DE SISTEMAS

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 11 de Octubre del 2021

Dictamen: 000231-C-EPIS-2021

Visto el borrador del expediente 000231, presentado por:

2014202221 - LOPEZ CORRALES MANUEL ALEJANDRO

Titulado:

IMPLEMENTACIÓN DE FRAMEWORK PARA EL DESARROLLO DE APLICACIONES MÓVILES MULTIPLATAFORMA BASADO EN COMPONENTES JSON APLICANDO SOA

Nuestro dictamen es:

APROBADO

1635 - SULLA TORRES JOSE ALFREDO
DICTAMINADOR

1748 - CALDERON RUIZ GUILLERMO ENRIQUE
DICTAMINADOR

2820 - ESQUICHA TEJADA JOSE DAVID
DICTAMINADOR

RESUMEN

El avance y desarrollo de la tecnología en informática es cada vez mayor y más exigente, sobre todo en cuanto a la velocidad de las comunicaciones a través de dispositivos móviles. Por esta necesidad de creación de más aplicativos, los desarrolladores buscan herramientas para agilizar la implementación de éstos. La gran mayoría de *frameworks* para el desarrollo multiplataforma del mercado utilizan tecnologías web o híbridas, lo cual hace que su rendimiento y experiencia hacia el usuario pueda ser mejorable.

Por ello, en la presente investigación se fijó como objetivo el diseño e implementación de un framework que pueda ser visualizado en cualquier dispositivo con acceso a internet, que facilite el desarrollo nativo de aplicaciones en Android y iOS utilizando los lenguajes Java y Swift.

En cuanto a la metodología, se hizo uso del Análisis Comparativo; de diferentes frameworks para evaluar el rendimiento, peso de la aplicación final o resultado y su facilidad de uso.

Para la validación del framework se desarrolló un aplicativo para Android y iOS para los docentes de la Universidad Católica de Santa María, en la cual pueden hacer la consulta de las calificaciones registradas anteriormente; su horario; acceso a datos de la biblioteca; y la firma digital de documentos vía huella digital.

En conclusión, el presente trabajo de investigación nos permitió comprobar que el framework utilizado reduce los tiempos de desarrollo y permite al usuario enfocarse en las tareas específicas; además, da la posibilidad del despliegue de actualizaciones en tiempo real.

Palabras claves: Móviles, Entorno de trabajo o *Framework*, desarrollo nativo, JSON, Web Service

ABSTRACT

The technology improves and implementation is each time stricter, especially in speed of communication through mobile devices. Due to this need to create more mobile applications, developers are looking for tools to speed up their implementations. Most frameworks on the market to multiplatform development use web or hybrid technologies, which means that their performance and user experience can be improved.

In this work the objective is the design and implementation of a framework that can be seen in any device with internet access, the native development of apps on Android and iOS using Java and Swift.

The methodology was comparative analysis of different frameworks to evaluate performance, size of application and easiness.

To validation of framework was develop an app for Android and iOS to teachers of the "Universidad Católica de Santa María", in which thet can consult the previously registered grades; their schedule; access to library; and digital sign of documents via fingerprint.

In conclusion this work probe the developed framework reduce the development time and focus on specific tasks, moreover the possibility of realtime updates.

Keywords: Mobile, Framework, native programming, JSON, Web Service

INTRODUCCIÓN

Hoy en día las empresas tienen tendencias por el predominio en los mercados digitales y de negocios, son oportunidades para acrecentar sus negocios y sus canales de venta. Es aquí donde la Ingeniería de Sistemas encuentra su oportunidad de desarrollo, estudiando y satisfaciendo las necesidades de las empresas. Gracias a la tecnología podemos innovar, y de esta forma podemos transformar un negocio, posicionarlo competitivamente automatizando o perfeccionando sus procesos, proporcionándole un factor diferenciador que le permita posicionarse con innovación en su área o sector, lo que, a su vez con seguridad, conducirá a la reducción de costos. Es decir, las empresas logran tener mayor ventaja en el mercado, con tecnología y aplicaciones, lo cual es un software. Por lo general, se requiere que las aplicaciones funcionen en múltiples dispositivos, en especial los móviles.

Al estar dividido el mercado en dos grandes plataformas, iOS y Android, los desarrolladores tienen que invertir tiempo y recursos en la tarea de desarrollo de ambas plataformas, respetando las características a las que los usuarios de esa plataforma están acostumbrados. Un *framework* para el desarrollo multiplataforma es un ambiente unificado donde se cuenta con las herramientas necesarias para el despliegue en ambas plataformas sin la tarea de la reimplementación del producto.

Este proyecto tiene el objetivo implementar un Framework que mejore los tiempos de desarrollo de aplicativos móviles, haciendo uso del formato JSON para el modelado de datos, interfaces y acciones requeridas. Además, por ser un Framework basado en componentes, se tiene la oportunidad de extender las funcionales generales y específicas requeridas para un desarrollo ágil y no restrictivo para generar componentes nuevos o extendidos de otros. Por otro lado, los componentes compilados pueden ser utilizados de forma dinámica en tiempo de

ejecución, lo que nos permitiría ofrecer actualizaciones remotas sin depender de la tienda de aplicaciones en la cual se publicó.

Así mismo se utilizó la metodología de desarrollo, Kanban, para gestionar de forma correcta el tiempo para el desarrollo, con la característica de poder medir tiempos y optimizar los cuellos de botella que se presentan en las tareas programadas.

En la primera parte del trabajo definiremos el planteamiento del problema, la justificación o razones del mismo, luego veremos los alcances y limitaciones del proyecto. Posteriormente, en el segundo capítulo planteamos la base teórica, antecedentes, las definiciones mínimas necesarias para entender la problemática y solución planteada, llegando finalmente a las especificaciones técnicas, el usuario objetivo, la sostenibilidad a lo largo del tiempo y riesgos que conlleva. El detalle del diseño del Framework, su implementación y datos relevantes se verán en el cuarto capítulo, y en el siguiente, presentaremos el estudio de un caso donde se hará uso del Framework, se demostrará sus características y la utilización de las distintas herramientas que provee. En la parte final se da a conocer la comparación con otros Framework y las conclusiones a las que hemos llegado con esta investigación.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGO

ÍNDICE DE BLOQUES

ÍNDICE DE TABLAS

CAPÍTULO 1

DESCRIPCIÓN DEL PROYECTO

1.1 Objetivos del problema

1.1.1 General

Desarrollar un *framework* para el desarrollo multiplataforma de aplicativos móviles orientados a servicios usando herramientas nativas y componentes JSON.

1.1.2 Específicos

1. Analizar otros frameworks del mercado para comparar sus beneficios y características
2. Desarrollar el framework utilizando la metodología Kanban, para las plataformas Android y iOS
3. Demostrar aplicabilidad del framework en un caso de estudio
4. Evaluar el rendimiento y optimización de los aplicativos
5. Optimizar el tiempo de desarrollo de aplicativos móviles

1.2 Alcances y limitaciones

1.2.1 Alcances

a. Definir el concepto de framework multiplataforma y explorar algunos frameworks disponibles en el mercado.
b. Documentación de la investigación realizada con respecto a los tipos de framework del mercado.
c. Implementación del framework para la plataforma Android y iOS.
d. Diseño e implementación de un caso de estudio para probar el framework desarrollado.

1.2.2 Limitaciones

- En la etapa de validación y análisis comparativo no se tomarán en cuenta los

framework web, por no pertenecer al grupo de aplicaciones nativas que se buscan en este trabajo.

- El uso del framework está pensado para aplicaciones de productividad, excluyendo a aquellas que requieran un uso exhaustivo de gráficas, como por ejemplo los juegos.

1.3 Fundamentos Teóricos

1.3.1 Antecedentes del proyecto

Adicionalmente de los antecedentes descritos en el Apéndice A, se presentan los siguientes para un mejor entendimiento del contexto del trabajo.

Los dispositivos móviles han sido ampliamente adoptados en los últimos años. Los investigadores del mercado y pronósticos dicen que las computadoras personales se volverán artefactos del pasado, gradualmente pasando a dispositivos portables de bajo costo, como tablets y teléfonos inteligentes. (International Data Corp, 2014)

Desde que apareció el exitoso formato de las aplicaciones, no deja de crecer y ser aceptado en nuestra sociedad. Vemos, por ejemplo, cuando entramos a un restaurant la carta que es una aplicación, el aforo que hay en un determinado local, comprar entradas para un concierto, hacer pedidos por delivery, comprar pasajes, etc. Muchas empresas ofrecen sus servicios a través de una app. También en el área del sector educativo es muy importante su uso que varía y se adecúa cada vez más a las necesidades del usuario, seguramente porque se puede llegar a los usuarios de una forma más directa.

Precisando sobre aplicaciones, Craven (2013) indica que las aplicaciones nativas usan de forma más efectiva el hardware del teléfono, tal como acelerómetro, giroscopio, GPS, cámara y

micrófono, y componentes de software como contactos y calendario. En la prueba de concepto de su investigación crea una aplicación haciendo uso del manejador de alarmas, notificaciones locales, acceso a ficheros, los cuales no podrían usarse en una aplicación utilizando JavaScript y HTML.

Los framework para el desarrollo de aplicaciones multiplataforma surgen de la necesidad de optimizar el proceso de desarrollo de un mismo aplicativo a varias plataformas. Una de las estrategias más utilizadas es el uso de tecnologías web para la producción del código fuente de la aplicación. Ejemplos de esta categoría serían PhoneGap, SenchaTouch o Titanium. (V. Mobile, 2012)

1.3.2 Bases teóricas del proyecto

A. Web Service

Los servicios web, nos permiten conectar dispositivos y computadoras con otras intercambiando data a través de Internet. Dieter, (2002) lo define como objetos de software que pueden ser ensamblados en Internet usando los protocolos estándar para ejecutar funciones o procesos de negocio. La clave de los servicios web es desacoplar y permitir componentes de software reutilizables.

B. Móviles

Son computadoras lo suficientemente pequeñas para mantener y ser operadas en la mano. Generalmente con una pantalla LCD u OLED, proveen una interfaz táctil con botones digitales y físicos. Muchos de estos pueden conectarse a internet y estar interconectados con otros dispositivos, tales como los sistemas de entretenimiento en automóviles vía Wi-Fi, Bluetooth, redes celulares o la tecnología NFC. Además, pueden integrar cámaras y un sistema de posicionamiento global.

C. Java – Android

Muchas aplicaciones Android están escritas en Java con las Android APIs. Android es una plataforma de código abierto de Google y esta provee las APIs necesarias para desarrollar aplicaciones con el lenguaje de programación Java. Las aplicaciones Android consisten en componentes tales como actividades, servicios, broadcaster y receivers, además los componentes se comunican mandando mensajes llamados Intents.

D. Swift

Es un lenguaje de programación compilado, de propósito general y multi paradigma desarrollado por Apple para iOS, iPadOS, macOS, watchOS, tvOS y Linux.

Fue diseñado para trabajar con Cocoa y Cocoa Touch y gran parte de código preexistente en Objective-C. Está construido con el compilador LLVM de código abierto y este ha sido incluido en XCode desde su versión 6. Otras plataformas de Apple hacen uso exclusivo de Objective-C.

E. Entorno de desarrollo (IDE)

Es un programa que provee facilidades a los desarrolladores para desarrollar software. Un IDE normalmente consiste en al menos un editor de código, herramientas de construcción automáticas y un depurador. Algunos IDEs como NetBeans y Eclipse, contienen un compilador e interprete y otros como SharpDevelop o Lazarus no. Los entornos de desarrollo son diseñados para maximizar la productividad al proporcionar herramientas a los programadores.

1.3.3 Técnicas y Herramientas

A. Apache

Es un software desarrollado de forma colaborativa, para servir como un servidor http robusto

y con características enriquecidas, con la posibilidad de extensión a través de paquetes externos. El software y código fuente son libres, pero la popularidad de apache es más a menudo atribuida al rendimiento que al costo. (Fielding R, 1997)

El proyecto es manejado por Apache Group, un grupo voluntario geográficamente distribuido quienes usan internet para comunicarse, desarrollar, distribuir y documentar. Adicionalmente miles de usuarios contribuyen con ideas, código y documentación del proyecto. (Fielding R, 1997)

B. PHP

Rasmus Lerdorf creó PHP en 1994 recolectando código y utilidades escritas en C, las cuales usó para construir páginas web para varios clientes. Originalmente fue concebido como un lenguaje de plantillas HTML. (Severance C, 2012)

La primera versión de PHP fue una herramienta de productividad que permitía acelerar el desarrollo de aplicaciones web. Rápidamente fue adoptada por otros desarrolladores web, quienes continuaron construyendo y mejorándola. (Severance C, 2012)

Dentro de la comunidad de PHP, hay pequeños grupos dedicados trabajando juntos para su desarrollo. Lerdorf prefiere dejar que el entusiasmo de los voluntarios vaya hacia adelante, incluso si ellos cometen pequeños errores que necesiten ser corregidos después a través de los revisores, que son usuarios más experimentados dentro de la comunidad. (Severance C, 2012)

C. Visual Studio Code

Es un editor de código multiplataforma ligero para escribir código, el cual funciona en los sistemas operativos OS X, Linux y Windows. Este editor ofrece a los desarrolladores soporte para múltiples lenguajes, posee además una característica de asistencia de código enriquecida y poder navegar por lenguajes como JavaScript, TypeScript, Node.js y ASP.NET con un

conjunto de herramientas. (Lardinois F, 2015)

El editor tiene todas las características estándar de un editor moderno como son resaltador de sintaxis, enlazar comandos de teclado, emparejamiento de paréntesis y sugerencias de código. (Lardinois F, 2015)

CAPÍTULO 2

DOCUMENTACIÓN TÉCNICA

Todo trabajo que se propone debe seguir un camino pragmático, es decir, deberá detallarse las tareas a realizar y detalles del proyecto. Considerando una exigencia permanente desde una perspectiva comercial en el mundo de los negocios, en esta etapa de la investigación se estudia las necesidades para la funcionalidad, vemos las herramientas para el desarrollo del framework, de igual manera, en este capítulo se presenta el modelado, configuración, los componentes y sus propiedades; detalles como la creación de nuevos componentes y la modalidad de actualizaciones automáticas.

2.1 Metodología

El proyecto se desarrolló utilizando la metodología Kanban, se programaron las diferentes actividades utilizando una representación gráfica en forma de tarjetas, que incluyen:

- Título
- Descripción
- Prioridad
- Duración estimada

Cada una de estas tarjetas fueron agregadas según la necesidad que se encontró durante el diseño de la arquitectura a baja escala. A continuación, se muestran las tarjetas que se utilizaron para este proyecto.

Se crearon dos tarjetas en la primera etapa, las cuales son el levantamiento de requerimientos y el diseño de la arquitectura como se puede apreciar en la figura 1a.

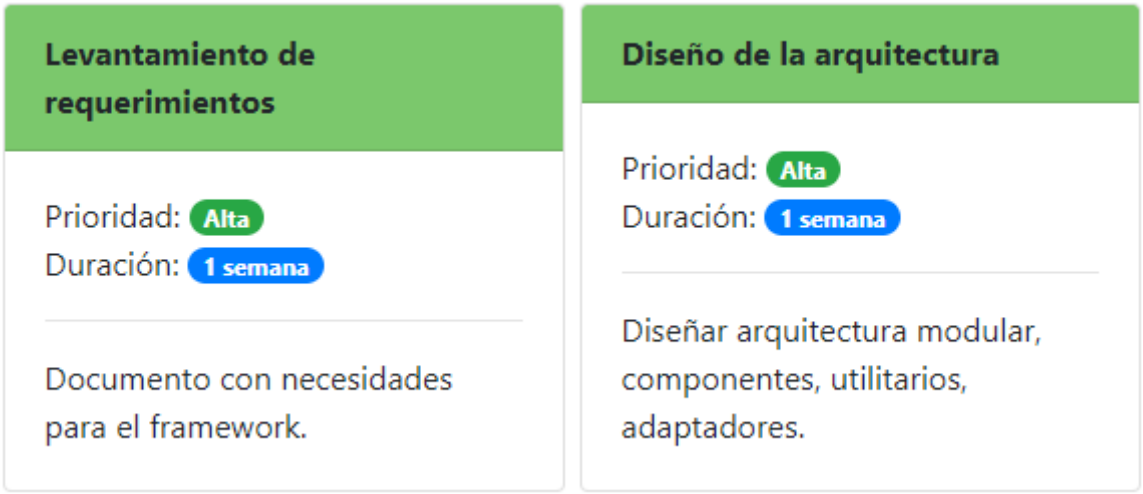


Figura 1a: Tarjetas Kanban - Fase inicial
Fuente: Propia

Uno de los elementos más importantes que se obtuvieron durante el análisis de requerimientos es el renderizado dinámico de las interfaces de usuario, esta tarea se realizó para las dos plataformas compatibles, como se observa en la figura 1b.

Figura 1b: Tarjetas Kanban - Motor de renderizado
Fuente: Propia

Un requerimiento necesario en la actualidad son las peticiones web para consumir los diferentes servicios, por este motivo se generó una tarjeta Kanban para cubrir esta necesidad como se observa en la figura 1c.

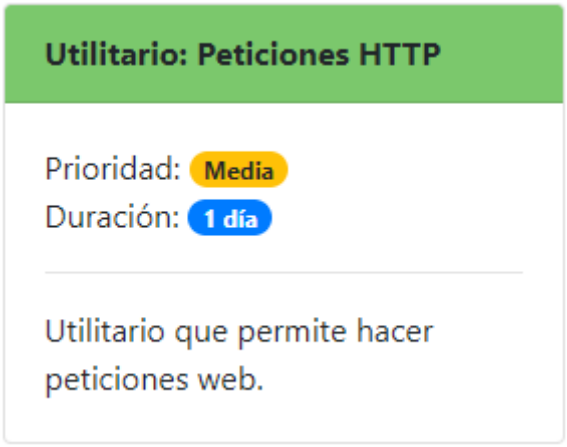


Figura 1c: Tarjetas Kanban - Utilitario para realizar peticiones HTTP
Fuente: Propia

Además, se requerían componentes gráficos para poder ingresar y mostrar la información de los servicios web, como se muestra en la figura 1d están dos componentes básicos que deben ser renderizados por el framework.

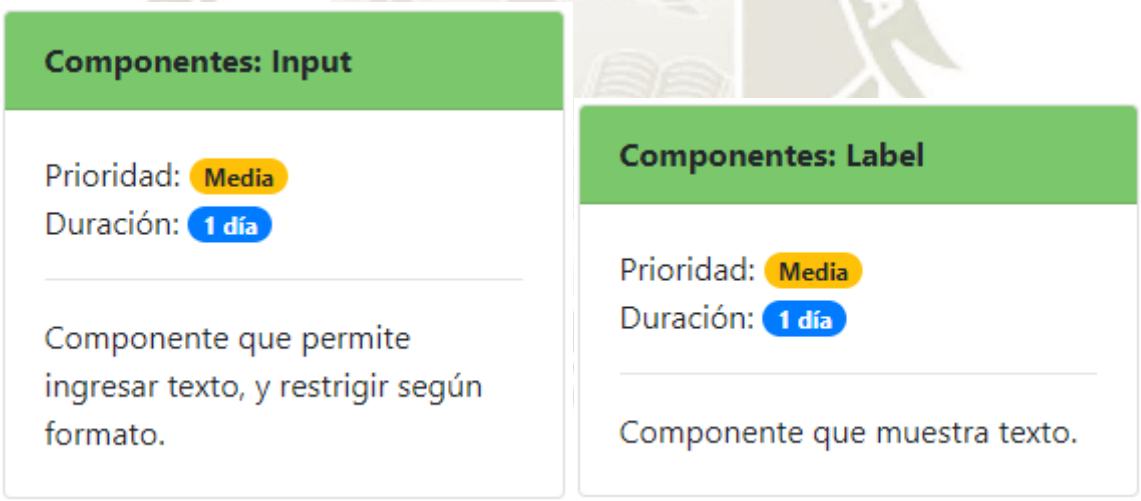


Figura 1d: Tarjetas Kanban - Componentes 1
Fuente: Propia

Otro componente muy utilizado en las aplicaciones modernas son las imágenes, se creó otra tarjeta Kanban para cumplir este requerimiento. Para interactuar con el usuario era necesario implementar un componente de tipo botón, con acciones pre-programadas y poder interactuar con el motor de renderizado y sus componentes, esta tarjeta se puede apreciar en la figura 1e.

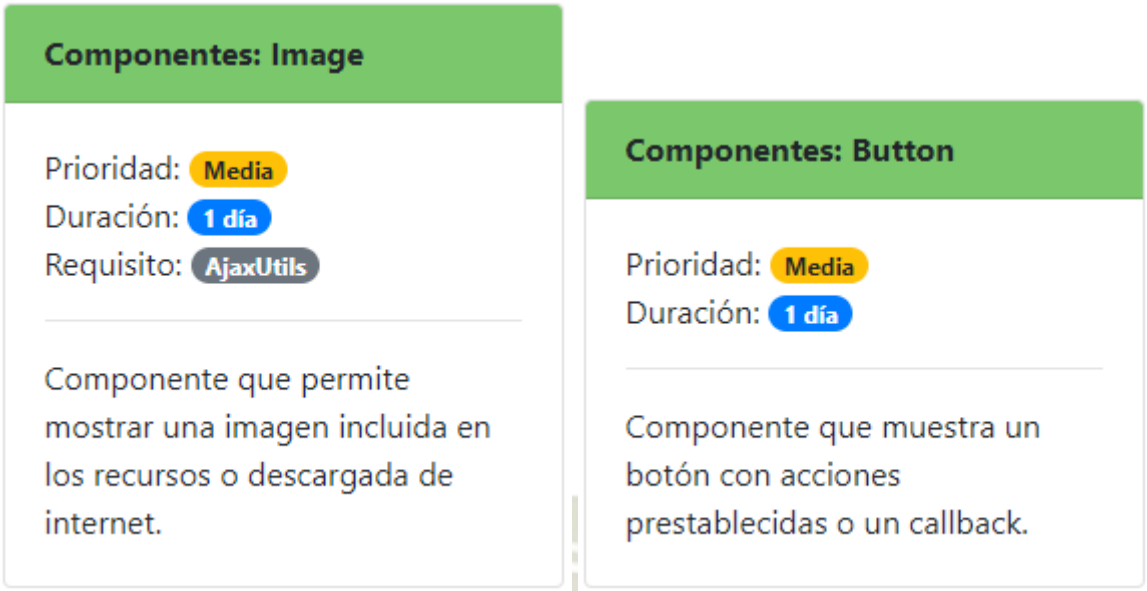


Figura 1e: Tarjetas Kanban - Componentes 2
Fuente:Propia

Para las listas desplegables se necesitaban crear contenedores y adaptadores de datos JSON para poder utilizarlos, este requerimiento se puede observar en la figura 1f.

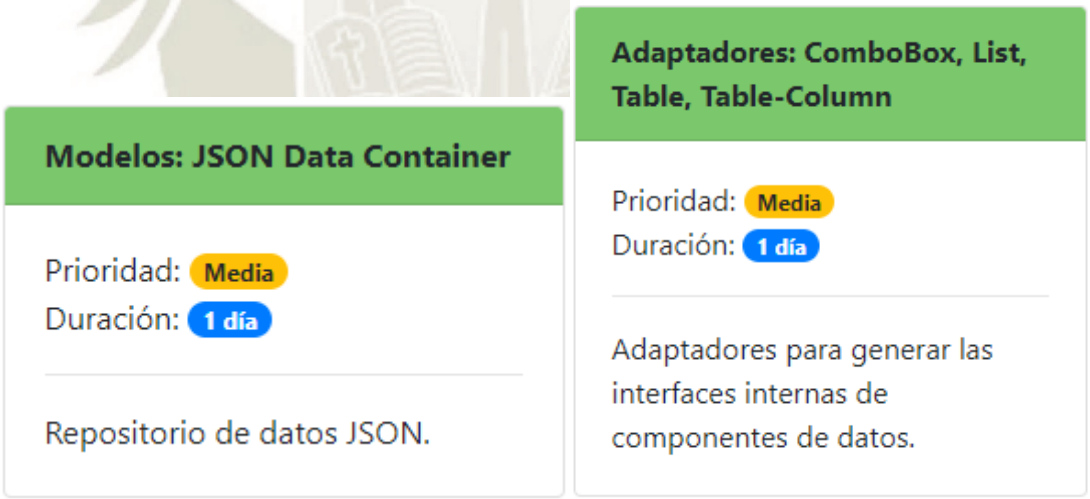


Figura 1f: Tarjetas Kanban - Adaptadores JSON
Fuente: Propia

Con los adaptadores de datos y el contenedor de datos encargado de procesar los datos almacenados localmente o recibidos por un servicio web, se necesitaban los componentes para poder visualizar los datos, por esto se crearon las tarjetas Kanban para la implementación de un ComboBox, una lista con detalle desplegable y una tabla. Estas tres tarjetas se pueden apreciar en la figura 1g.

Figura 1g: Tarjetas Kanban - ComboBox, Listas y Tablas
Fuente: Propia

Además de los componentes que pueden mostrar datos, leer los ingresos por parte del usuario se requería un elemento que pudiera mostrar elementos web estableciendo una URL y poder interactuar con esta sin salir de la aplicación.

Para cumplir con el requisito de un framework que permita las actualizaciones automáticas sin hacerlo a través de los diferentes markets y sólo utilizando la conexión a internet del usuario, se creó una tarjeta Kanban para la implementación de un componente que pudiera renderizar

una configuración cargada de un servicio web y que esta pudiera ser almacenada localmente para cuándo el usuario no contara con conexión a internet.

Ambas tarjetas se pueden apreciar en la figura 1h.

Figura 1h: Tarjetas Kanban – WebView y JSONView
Fuente propia

2.2 Costos de desarrollo

Costos	
Servicios	S/ 210.00
Costos indirectos (PC, capacitación, libros, material de trabajo)	S/ 2500.00
Internet	S/ 100.00

Otros Costos			
Recurso	Costo	Periodo	Costo 5 años
Licencia Android	$25	De por vida	$25
Licencia iOS	$100	Un año	$500

2.3 Diagrama de Casos de Uso

En esta sección se muestra el diagrama de casos de uso, y en las siguientes secciones se detallará el framework para su mayor entendimiento.

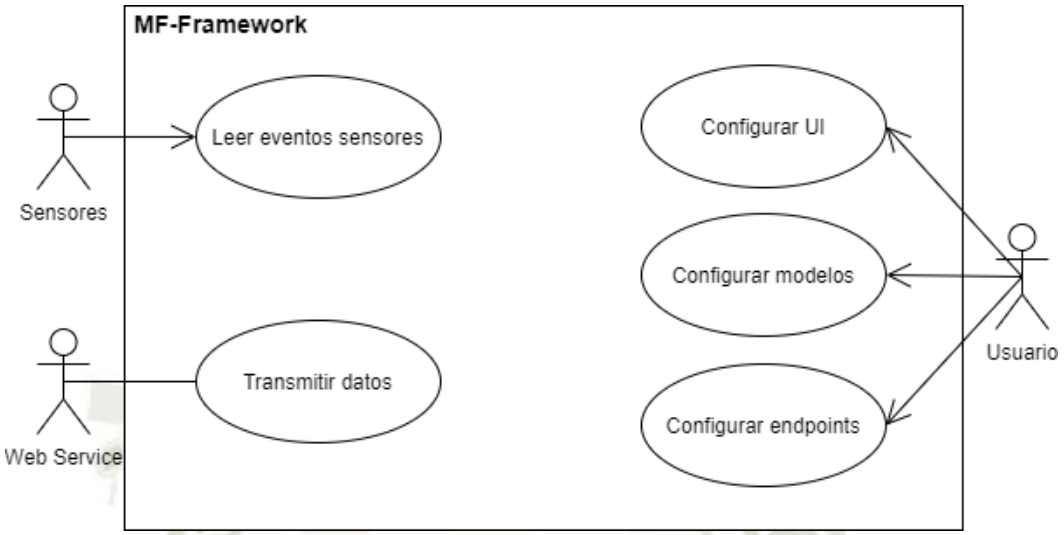


Figura 2: Diagrama de casos de uso del framework
Fuente: Propia

Tabla 1
Caso de uso – Leer configuración JSON

Caso de uso	**Configurar UI**
Código	CU01
Descripción	El framework lee el archivo de configuración JSON dónde estarán especificados los componentes visuales a ser mostrados al usuario y las interacciones entre estos.
Actores	Usuario
Precondición	Tener instalado el framework
Flujo	1. Buscar el archivo de configuración 2. Leer el archivo 3. Enviar datos leídos al render 4. Delegar tareas a los componentes correspondientes
Flujo alterno	1. Excepción: Si la configuración no es válida se mostrará el error por consola

Tabla 2
Caso de uso – Transmitir datos de solicitud HTTP

Caso de uso	**Transmitir datos**
Código	CU02
Descripción	El framework envía una solicitud HTTP y recupera la respuesta del servicio solicitado.
Actores	Web Service
Precondición	Tener configurado el framework
Flujo	1. Leer la URL del servicio web 2. Leer y transformar los parámetros a enviar 3. Ejecutar solicitud 4. Recuperar datos del servicio 5. Transformar datos utilizando el contenedor configurado 6. Enviar los datos al motor 7. Delegar datos recibidos al componente correspondiente
Flujo alterno	1. Excepción: Si la configuración no es válida se mostrará el error por consola

Tabla 3
Caso de uso – Leer eventos de sensores

Caso de uso	**Leer eventos de los sensores**
Código	CU03
Descripción	El framework lee los registros de los diferentes sensores para ser utilizados por la aplicación.
Actores	Sensores
Precondición	Tener configurado el framework Configurar la aplicación para permitir el uso de sensores
Flujo	1. Analizar si se requiere el uso de algún sensor 2. Leer eventos de los sensores 3. Enviar datos al motor 4. Delegar datos al componente correspondiente
Flujo alterno	1. Excepción: Si la configuración no es válida se mostrará el error por consola

Tabla 4
Caso de uso – Mapear componentes

Caso de uso	**Configurar modelos**
Código	CU04
Descripción	El framework lee el archivo de configuración JSON dónde estarán especificados los modelos de datos que se utilizarán para obtener la data del servicio web.
Actores	Usuario
Precondición	Tener instalado el framework
Flujo	1. Buscar el archivo de configuración 2. Leer el archivo 3. Enviar datos leídos al render 4. Delegar tareas a los componentes correspondientes
Flujo alterno	1. Excepción: Si la configuración no es válida se mostrará el error por consola

Tabla 5
Casos de uso – Renderizar UI

Caso de uso	**Configurar Endpoints**
Código	CU05
Descripción	El framework lee el archivo de configuración JSON dónde estarán especificados los endpoints de dónde se consumirá los servicios web.
Actores	Usuario
Precondición	Tener instalado el framework
Flujo	1. Buscar el archivo de configuración 2. Leer el archivo 3. Enviar datos leídos al render 4. Delegar tareas a los componentes correspondientes
Flujo alterno	1. Excepción: Si la configuración no es válida se mostrará el error por consola

2.4 Configuración inicial

En esta sección se detalla el proceso de configuración del framework y su uso.

Tabla 6
Descripción de casos de uso: Instalación del Framework

Instalación del Framework	
Descripción	El desarrollador debe copiar la carpeta principal del framework a la raíz del proyecto. ● En Android dentro de la carpeta app ● En iOS en la raíz del proyecto
Precondición	Tener la versión actualizada del framework
Flujo	1. Extraer en una carpeta los archivos del framework. 2. Buscar la carpeta del proyecto en Android Studio y XCode. 3. Copiar la carpeta mf_files en la carpeta principal del proyecto.
Flujo alterno	1. Excepción: En el paso 2 del flujo, tiene como requisito tener creado el proyecto en Android y iOS.
Postcondición	En Android Studio antes de compilar se deben añadir las librerías.

Tabla 7
Descripción de casos de uso: Añadir librerías

Añadir librerías (Solo Android)	
Descripción	En el archivo app.gradle se deben añadir las dependencias necesarias.
Precondición	Instalación del framework
Flujo	1. Abrir el proyecto en Android Studio 2. Dentro de la carpeta app, buscar el archivo app.gradle 3. En la sección dependencies añadir las librerías de material design. 4. Sincronizar
Flujo alterno	1. Excepción: En el punto 3 del flujo se deben añadir todas las librerías que utilizaremos, siendo solo material design un requisito indispensable.
Postcondición	Compilar el proyecto para validar la instalación

Tabla 8
Descripción de casos de uso: Crear actividad

Crear actividad	
Descripción	En Android se debe crear una nueva actividad que herede de MFFramework. En iOS se debe crear un nuevo controlador con su respectivo XIB relacionado.
Precondición	Instalar el framework y agregar librerías
Flujo	Android 1. Desde Android Studio agregar una clase que herede de MFFramework 2. Implementar el constructor 3. Llamar al método heredado loadView() iOS 1. Desde XCode crear nuevo un nuevo controlador 2. Cambiar la herencia del controlador a MFFramework 3. Llamar al método heredado loadView() desde ViewDidLoad()
Postcondición	Compilar y ejecutar la aplicación, si no se le pasa ningún argumento mostrará una pantalla en blanco.

Tabla 9
Descripción de casos de uso: Definir actividad en el manifiesto

Definir actividad en el manifiesto (Solo Android)	
Descripción	Se debe añadir la actividad en el manifiesto para que Android reconozca el punto de entrada.
Precondición	Añadir la actividad
Flujo	1. Abrir el archivo AndroidManifest 2. Agregar la etiqueta activity con la clase creada
Flujo alterno	1. Excepción: En el paso 2 del flujo se deben definir los tipos de actividad ya sea LAUNCHER o actividad normal.
Postcondición	Compilar y ejecutar la aplicación, si no se le pasas ningún argumento mostrará una pantalla en blanco.

Tabla 10
Descripción de casos de uso: Crear archivo JSON

Crear archivo JSON	
Descripción	Elegir la forma de mostrar la data, ya sean listas, combos, tablas o cualquier otro componente y establecer sus propiedades.
Precondición	Modelar la estructura de datos a mostrar
Flujo	1. Elegir el contenedor de datos 2. Establecer la fuente de los datos a. Local b. Ajax 3. Establecer las propiedades del contenedor de datos 4. Crear fichero formato JSON
Flujo alterno	1. La estructura del archivo JSON del paso 4 del flujo es definida en la documentación del framework.

Tabla 11
Descripción de casos de uso: Modelar estructura de datos a mostrar

Modelar estructura de datos a mostrar	
Descripción	Para mostrar la data se requiere una definición de la estructura, por lo que se deberá crear esta en formato JSON
Flujo	1. Definir datos a mostrar 2. Identificar nombre de los campos y tipos de datos 3. Plasmar la estructura en un archivo JSON
Flujo alterno	1. La estructura del archivo JSON del paso 3 del flujo es definida en la documentación del framework.

2.5 Arquitectura

Para el framework se utilizó una arquitectura cliente servidor. Para la comunicación se creó una librería RestClient para el manejo de las solicitudes en segundo plano entre el servidor y la aplicación desarrollada por el framework. El esquema general se puede observar en la figura 3, posteriormente se detallará la arquitectura a bajo nivel.

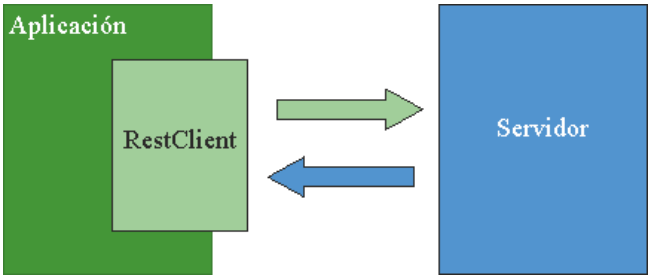


Figura 3: Arquitectura cliente-servidor
Fuente: Propia

La figura 4 muestra la arquitectura a bajo nivel del framework, incluyendo los flujos de uso de los componentes y relación con los servicios web consumidos desde un servidor. El único sensor que se ha implementado y probado es el lector de huella, y la arquitectura pensada para ser posteriormente extendida.

Los diferentes componentes mostrados en la arquitectura a bajo nivel son detallados en las siguientes secciones.

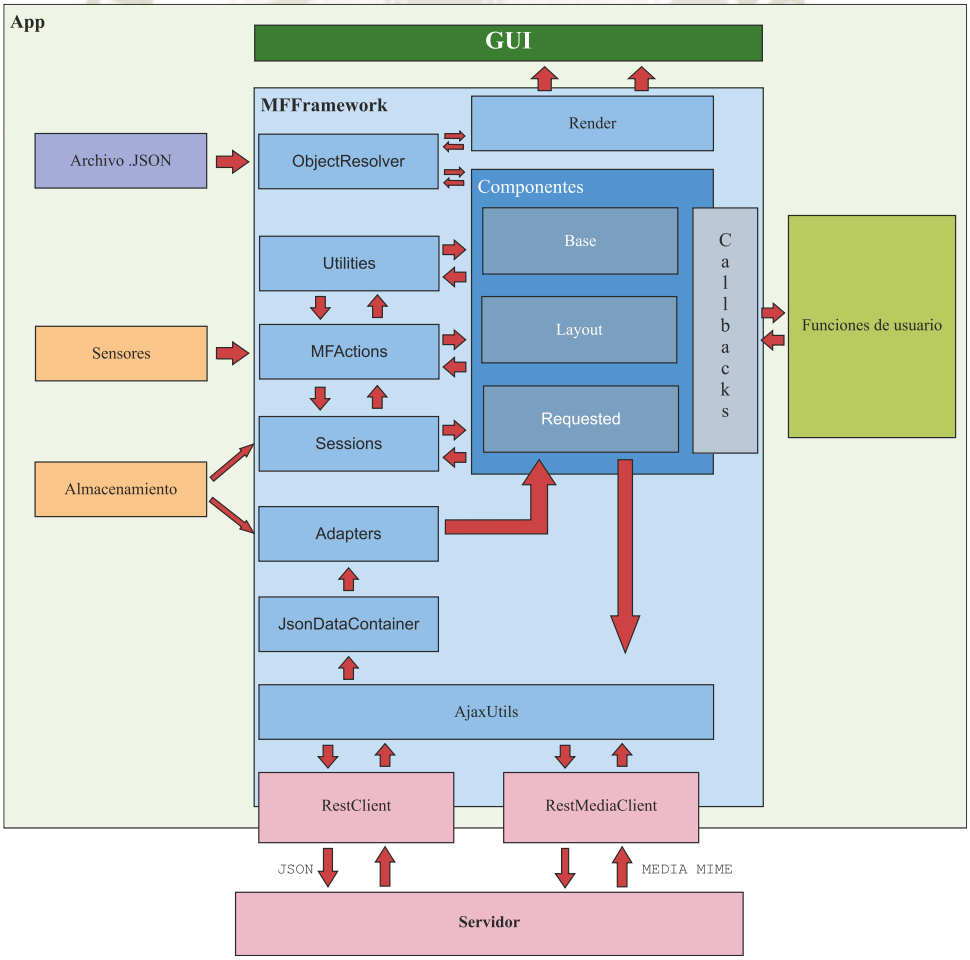


Figura 4: Arquitectura bajo nivel del framework
Fuente: Propia

2.6 Funcionamiento general

El objetivo principal del framework es reducir el tiempo de desarrollo utilizando un archivo de texto con una estructura JSON que pueda ser leído por un intérprete y ser renderizado por cada plataforma, que se puede entender mejor con la siguiente gráfica.

El funcionamiento general del framework se muestra en la figura 5.

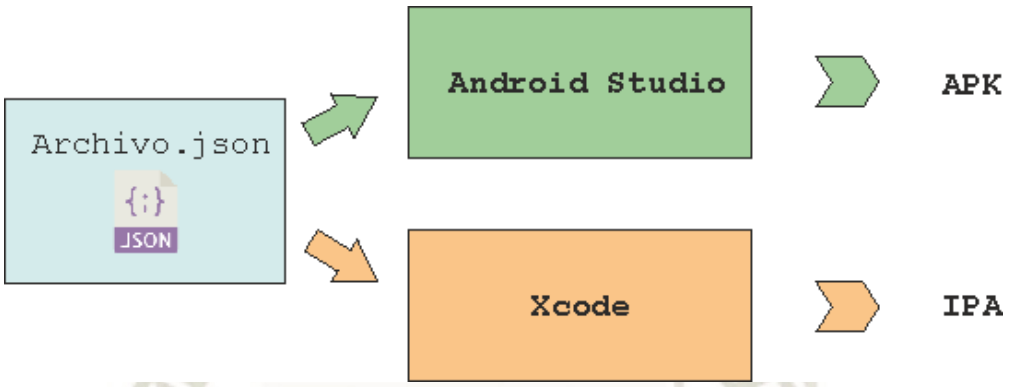


Figura 5: Funcionamiento general del framework
Fuente: Propia

La estructura de los archivos del proyecto es de la siguiente manera.

Android
mf_files [FOLDER] Adapters [FOLDER] Components [FOLDER] Layouts [FOLDER] Models [FOLDER] Util [FOLDER] CustomAdapter MFFramework OnCallbackClick

Bloque 1: Estructura de archivos en Android

En el caso de android además de las carpetas y archivos mostrados en la parte superior, se hace uso de las siguientes librerías.

com.android.support:appcompat-v7 com.android.support:cardview-v7 com.android.support:design

Bloque 2: Librerias Android

iOS
mf_files [FOLDER] Adapters [FOLDER] Components [FOLDER] Layouts [FOLDER] Models [FOLDER] Util [FOLDER] CustomAdapter MFFramework OnCallbackClick MFActions

Bloque 3: Estructura framework iOS

Una vista general de los paquetes se muestra en la siguiente imagen.

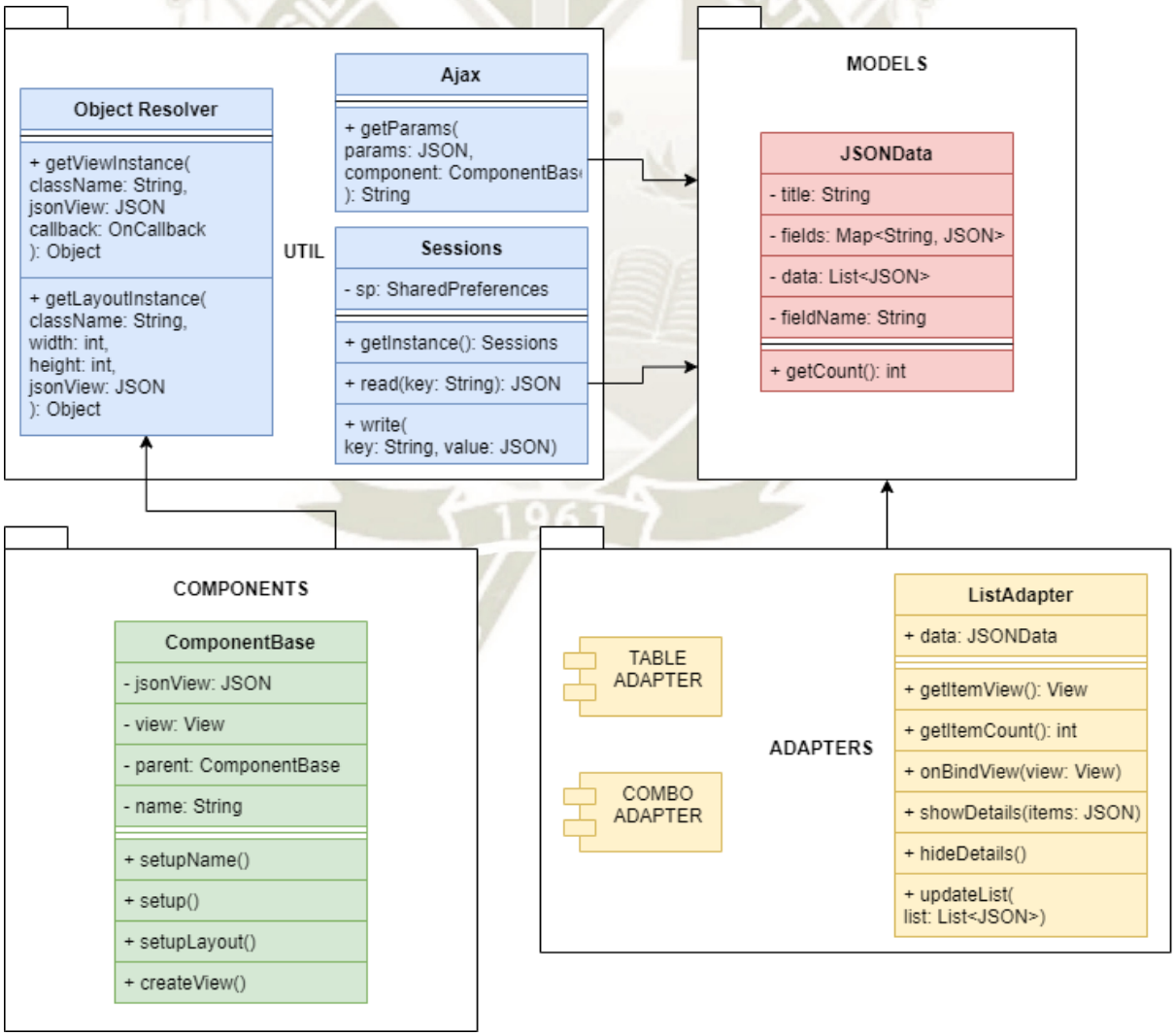


Figura 6: Vista general de paquetes del framework
Fuente: Propia

Dentro de cada lenguaje se deben especificar las actividades que deben heredar de la clase MFFramework que será la encargada de renderizado y delegar los métodos definidos por el usuario en el archivo JSON y en la clase.

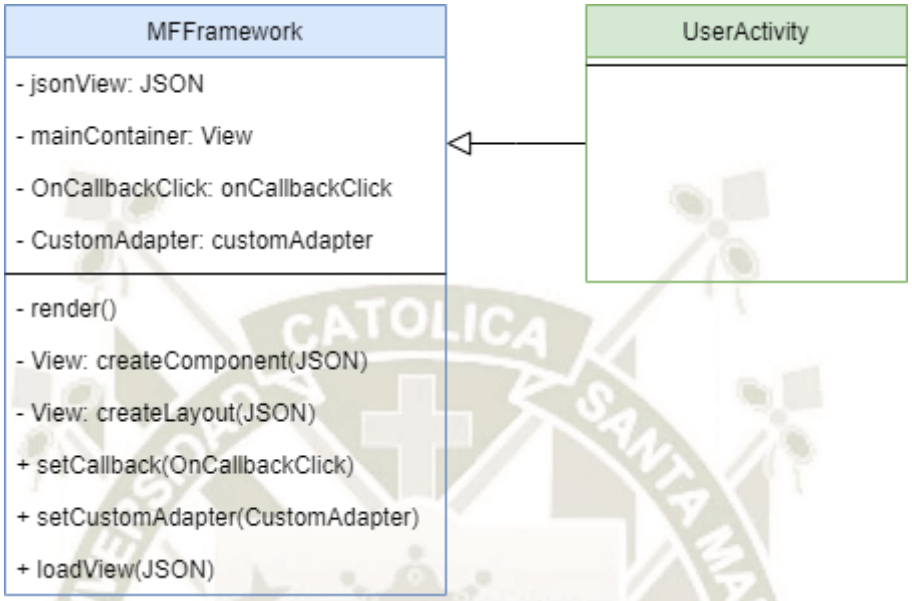


Figura 7: Clase base – MFFramework
Fuente: Propia

Como se muestra en la imagen todas las tareas de visualización son delegadas a la clase MFFramework, en esta se encuentran dos métodos que nos darán una mayor interacción para que el programador pueda hacer acciones externas a las que ya incluye el framework.

La interfaz OnCallbackClick nos permite programar acciones personalizadas después de los eventos que especifiquemos dentro del archivo json.

La interfaz CustomAdapter nos permite dar formato a los contenedores de datos, como pueden ser las tablas y las listas.

En esta sección se detallarán los métodos de la clase ya definida

- render() Nos permite crear las vistas y mostrarlas en la pantalla del usuario.
- createComponent() Crea una instancia de ComponentBase, el cual es un componente independiente.

- createLayout() Crea un contenedor de componentes, puede ser lineal o relativo.
- setCallback() Establece una interfaz para ser usada al generarse un evento.
- setCustomAdapter() Establece una interfaz para personalizar el comportamiento de un adaptador de datos hacia las vistas.
- loadView() Este método permite cargar en memoria el objeto JSON que representa las vistas y funcionalidad de la aplicación.

En la siguiente figura se muestra la interacción de los métodos definidos.

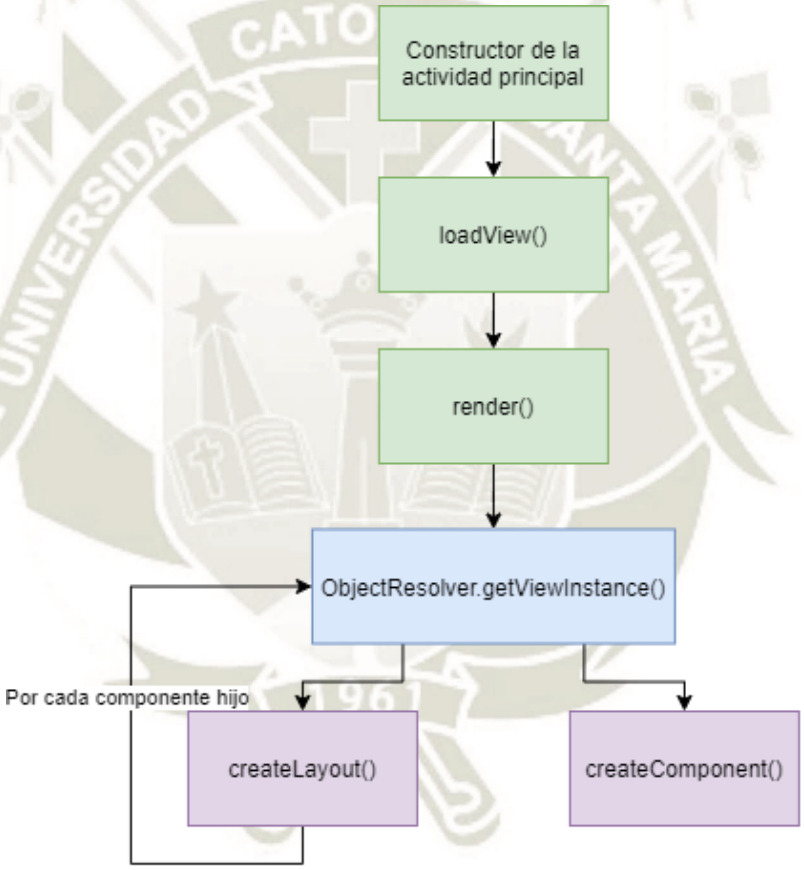


Figura 8: Diagrama de interacción entre métodos del framework
Fuente: Propia

La figura anterior nos muestra un componente llamado ObjectResolver el cual está preparado para reconocer el tipo de instancia que está recibiendo, reservarle espacio en memoria y llamar a los métodos correspondientes ya sea un Layout o un Component.

Como se puede observar en la figura 6, la renderización se hace de forma recursiva, teniendo como punto de finalización cuando ya no hay más hijos en el layout.

2.7 Renderizado

Este proceso consiste en leer la parte del objeto JSON de la vista e ir avanzando recursivamente si es una vista de tipo de layout o creando la vista del componente final.

En el caso de los ComponentBase se dividen como se puede observar en la siguiente figura.

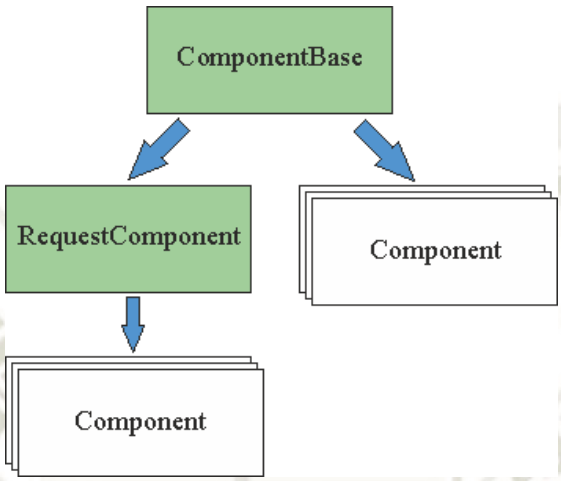


Figura 9: Subclases de ComponentBase
Fuente: Propia

Los componentes que heredan de RequestComponent tienen la capacidad de hacer consultas a un servicio web utilizando el utilitario RestClient o RestClientImage.

El otro tipo de objeto a renderizar es layout, el cual contiene una lista de hijos de tipo ComponentBase.

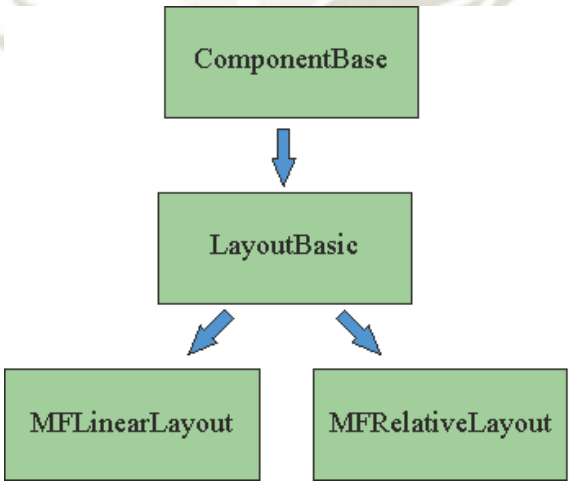


Figura 10: Estructuras Layout utilizadas por el framework
Fuente: Propia

Como se observa en la figura anterior, LayoutBasic tiene dos tipos instancias:

- MFLinearLayout: Este tipo de contenedor nos permite asignarle una orientación, y los elementos hijos se organizan según esta.

Figura 11: MFLinearLayout Horizontal
Fuente: https://droidbyme.medium.com/android-cardview-with-recyclerview-90cfeda6a4d4

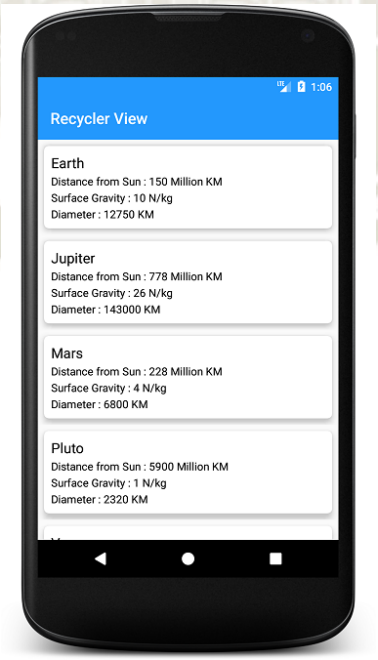


Figura 12: MFLinearLayout Vertical
Fuente: https://droidbyme.medium.com/android-cardview-with-recyclerview-90cfeda6a4d4

- MFRelativeLayout: Este tipo de contenedor nos permite posicionar cada hijo de

forma relativa al contenedor.

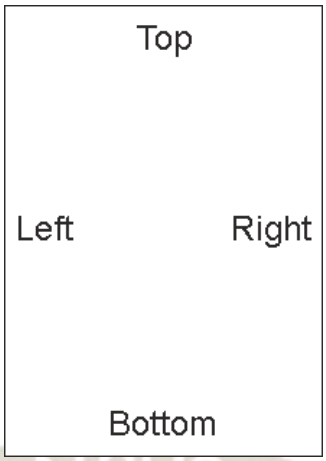


Figura 13: MFRelativeLayout
Fuente: Propia

Existe una configuración básica que poseen todos los componentes, la cual se definirá a continuación:

- width: Con este atributo se establece el ancho del componente a renderizar. Puede tomar los siguientes valores:
    - match: Se estira hasta cubrir todo el ancho del contenedor padre
    - wrap: Calcula el espacio necesario para mostrarse
    - [Integer]: Se puede proporcionar una medida en pixeles
- height: Con este atributo se establece el alto del componente a renderizar. Puede tomar los siguientes valores:
    - match: Se estira hasta cubrir todo el alto del contenedor padre
    - wrap: Calcula el espacio necesario para mostrarse
    - [Integer]: Se puede proporcionar una medida en pixeles
- padding: Establece un espacio exterior al componente. Puede tomar los siguientes valores:
    - [Integer]: Se aplica a los cuatro lados Top, Bottom, Left, Right

    - [Integer,Integer]: Se aplica el primero para el Top y Bottom, el segundo para Left y Right
    - [Integer,Integer,Integer]: Se aplica el primero para el Top, el segundo para Left y Right, y el tercero para Bottom
    - [Integer,Integer,Integer,Integer]: Se aplica el primero para el Top, el segundo para Right, el tercero para el Bottom, y el cuarto para Left
- hidden: Define si el elemento es visible. Puede tomar los siguientes valores:
    - true: Si se muestra. Puede ignorarse esta propiedad si el valor es true.
    - false: No se muestra.
- color: Establece un color de fondo al componente. Puede tomar los siguientes valores:
    - [#NNNNNN]: Color en formato hexadecimal.
- flex: Esta propiedad permite establecer un porcentaje de alto o ancho según la propiedad orientation, del contenedor MFLinearLayout que lo contenga.

Todos los elementos visuales heredan de la clase ComponentBase la cual contiene dos métodos necesarios para el renderizado:

- setup(): Este método lee los parámetros del objeto JSON correspondiente a la vista para configurar la instancia.
- createView(): Este método crea y devuelve una instancia de View que será incluida en nuestro lienzo.

2.8 Componentes

Cada componente es un elemento visual enriquecido, lo que significa que tiene funcionalidad adicional a la forma en que se muestra.

2.8.1 Componentes estándar

Los componentes estándar son elementos que funcionan de forma local, sin necesidad de

consumir algún servicio web.

Tabla 12
Descripción y propiedades del componente: MFButton

MFButton		
Descripción	Botón mejorado, permite acciones pre-programadas o definidas por el desarrollador a través de la llamada callback.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
text	String	Si
textColor	String – Color en hexadecimal	Si
style	String: [simple, bold]	Si
image	String – Nombre del recurso	Si
textSize	Integer	Si
actions	List<JSON>	Si

Tabla 13
Descripción y propiedades del componente: MFInput

MFInput		
Descripción	Componente que permite el ingreso de texto, puede personalizarse con colores, y tipo de texto.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
placeholder	String	Si
input_type	String: [text, password, number]	Si
textColor	String – Color en hexadecimal	Si
hintColor	String – Color en hexadecimal	Si

Tabla 14
Descripción y propiedades del componente: MFLabel

MFLabel		
Descripción	Componente para mostrar texto, puede cambiar su valor con los eventos y acciones definidas de otros componentes.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
value	String	Si
textColor	String – Color en hexadecimal	Si

Tabla 15
Descripción y propiedades del componente:MFModerBar

MFModernBar		
Descripción	Este componente nos permite mostrar un menú en la parte inferior de la app a modo pestañas.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
childs	List<JSON>	No
childs.title	String	Si
childs.icon	String	Si

2.8.2 Componentes compatibles con servicios web

Este tipo de componentes nos permiten hacer solicitudes a servicios a través de internet, además cuentan con contenedores de datos que pueden ser almacenados en el teléfono. El contenedor de datos almacena, procesa los datos para poder ser mostrados por la interfaz de usuario a través de los adaptadores.

A. Estructura DataContainer

La estructura de los datos debe ser modelada dentro del archivo JSON que leerá el framework, este debe incluir los parámetros que se detallan en la tabla 16.

Se incluye el nombre del parámetro el cual debe coincidir en mayúsculas, minúsculas y sin espacios; la descripción detalla el objetivo del parámetro y el tipo de dato que se espera; y por último si es un campo requerido o se puede saltar su configuración.

Tabla 16
Estructura de DataContainer

Parámetros	**Descripción**	**Opcional**
code	Nombre del campo que se utilizará como clave primaria del registro.	Si
name	Nombre del campo que se utilizará para mostrar. Si se desea mostrar varios campos, se debe utilizar el parámetro fields.	Si
fields	Es una lista de campos que se mostrarán al usuario. Si se define reemplaza la funcionalidad de name.	Si
fields.dataIndex	Nombre del campo	No
fields.type	Tipo de valor almacenado. [string, number]	No
persistant	Si se define guarda los valores en el contenedor para ser utilizados sin internet.	Si
persistant.name	Nombre del contenedor. Con este identificador el usuario puede recuperarlo o utilizarlo en otro componente.	No
ajax	Al definir este parámetro el componente utilizará la data de	Si

	un servicio web.	
ajax.url	Dirección web del servicio	No
ajax.method	Método para consumir el servicio. [GET, POST]. Si no se define se utiliza POST.	Si
ajax.params	Parámetros para enviar al servidor	Si
ajax.events	Lista de acciones a realizar durante, y después de la solicitud.	No
ajax.autoload	Este parámetro permite deshabilitar la petición automática al servicio web. [true, false]	Si
columns	Lista de columnas. Este parámetro es utilizado por el componente MFTable para mostrar la cabecera de la tabla.	No
columns.text	Nombre que aparecerá en la cabecera.	No
columns.dataIndex	Nombre del campo para mostrar en la columna.	No
data	Lista en formato JSON con los datos a mostrar, si se definió el parámetro ajax, estos datos se mostrarán hasta que finalice la solicitud.	Si

B. Estructura respuesta del servidor web

La respuesta del servidor debe indicarnos el estado de la solicitud, si fue satisfactorio o se presentó algún error.

Tabla 17
Códigos de estado de respuesta HTTP

Códigos estado de respuesta HTTP	**Descripción**
200 – OK	Solicitud realizada correctamente.
301 – Moved Permanenlty	Recurso ya no se encuentra en la dirección proveída.
404 – Not Found	Dirección no encontrada.
408 – Request Timeout	Tiempo de espera agotado.
500 – Internal Server Error	Error interno del servidor

Sólo se leerán las respuestas del servidor con código 200, las respuestas consideradas en la tabla mostrarán el error que se muestra en la descripción, cualquier otro código de respuesta

será mostrado con un mensaje definido en la clase RestClient y el código correspondiente.

Todas las respuestas correctas del servidor deben ser en formato JSON y que además contengan el atributo success de tipo booleano.

Tabla 18
Atributos para petición a Servicio Web

Atributo	**Descripción**	**Opcional**
success	Atributo de tipo booleano que nos indica si se tuvo un resultado positivo en la consulta realizada.	No
data	Si el componente posee un DataContainer y el atributo ajax definido, los datos serán extraídos de este atributo.	Si
msg	Cuando el valor de success es false, se lee este atributo para ser mostrado al cliente.	Si
…Otros	Cualquier dato que se desee agregar a la respuesta.	Si

C. Listado de componentes

Tabla 19
Descripción y propiedades del componente: MFCombo

MFCombo		
Descripción	Este componente mostrará una lista de opciones preconfiguradas o cargadas desde un servicio web. Se puede configurar un evento cuando un elemento es seleccionado, ya sea llamando a las acciones predefinidas o usando un callback.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
dataContainer	JSON	No
events	List<JSON>	Si

Tabla 20
Descripción y propiedades del componente: MFImage

MFImage		
Descripción	Este componente mostrará una imagen ya sea local o cargada desde internet.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
hiddenProgress	Boolean	Si
bitimg	String	Si
url	String	Si

Tabla 21
Descripción y propiedades del componente: MFList

MFList		
Descripción	Este componente mostrará un listado de items, cargados en el archivo JSON o desde un servicio web. Los items al ser seleccionados se desplegará información que el desarrollador configure.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
title	String	Si
dataContainer	JSON	No
events	List<JSON>	Si

Tabla 22
Descripción y propiedades del componente: MFWebView

MFWebView		
Descripción	Este componente mostrará una página web embebida en esta vista.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
url	String	No

Tabla 23
Descripción y propiedades del componente: MFJsonView

MFJsonView		
Descripción	Este componente mostrará una vista dinámica cargada desde un servicio web, este componente obtiene un fichero en formato JSON y lo renderiza en el contenedor de este componente. Si se activa la opción de cache, la aplicación mantendrá la última vista dinámica cargada.	
Propiedad	**Valores**	**Opcional**
width	wrap/match/Integer	No
height	wrap/match/Integer	No
name	String	Si
ajax	JSON	No
cache	Boolean	Si

2.9 Adaptadores

Los adaptadores nos permiten transformar los datos que tenemos en el contenedor JsonDataContainer en vistas que serán gestionadas y mostradas por los diferentes componentes. Los componentes descritos en la sección anterior hacen uso de los siguientes adaptadores.

2.9.1 ComboAdapter

Este adaptador genera una vista desplegable para ser utilizada por el componente MFCombo, utiliza el atributo fields para seleccionar la data de JsonDataContainer que debe ser mostrada al usuario.

Además, este adaptador posee una función getItem que será utilizada por el evento onSelect, y este devolverá el valor del campo code definido por el componente.

2.9.2 ListAdapter

Este adaptador genera una vista por cada elemento en el contenedor JsonDataContainer, el cual es utilizado por el componente MFList.

Los datos definidos con el atributo fields serán mostrados como un detalle de la fila seleccionada al hacerle clic.

2.9.3 TableAdapter

Este adaptador nos crea una vista con una cabecera y un cuerpo tipo tabla que es utilizada por el componente MFTable.

Se hace uso del adaptador ColumnAdapter, el cual nos permitirá manejar de forma independiente las columnas mostradas y poder desplazarnos sin perder la organización tipo tabla.

2.9.4 ColumnAdapter

Este adaptador nos crea una vista de una columna en específico de un JsonDataContainer.

Sólo se utiliza como utilitario del adaptador TableAdapter, no se utiliza por ningún componente de forma independiente.

2.10 Selectores

Los selectores son una herramienta que nos permiten conseguir los valores de los componentes, sesiones o respuestas de un servicio web, ejecutar acciones predefinidas o creadas por el desarrollador. Todo selector tiene como prefijo el carácter $.

2.10.1 Selector de sesiones

Los selectores de sesiones nos permiten recuperar los valores guardados en las preferencias de usuario o sesiones generadas con las acciones predefinidas por el framework.

Este selector tiene como prefijo MFSession$ seguido por el identificador de sesión y por último el atributo del cual se requiere el valor.

Estructura:

MFSession$[name]$[attribute]

Ejemplo:

MFSession$LOGIN$CCODUSU

2.10.2 Selector de respuestas de un servicio web

Un selector de respuestas de un servicio web nos permite conseguir los valores de la respuesta de un servicio y poder almacenarlo en una sesión o el almacenamiento del teléfono.

Este selector tiene como prefijo MFresponse$ seguido por la palabra clave getValue.

Estructura

MFresponse$getValue

2.10.3 Selector de componentes

El selector de componentes nos permite recuperar una referencia de una vista, utilizando su

identificador name y poder llamar a un método del componente recuperado.

Estructura

$[name]$[method]

Ejemplo

$txtBusqueda$getValue

2.11 Acciones predefinidas

2.11.1 Borrar sesiones y datos

Esta acción nos permite borrar todo el contenido guardado por el framework. La aplicación quedará con todos los ajustes iniciales.

2.11.2 Guardar datos

Permite guardar la respuesta de un servicio web en el dispositivo.

2.11.3 Mostrar mensaje

Muestra un mensaje por pantalla

2.11.4 Lanzar actividad

Inicia otra actividad indicada por el usuario.

2.11.5 Ejecutar método de un componente

Utiliza los selectores para ejecutar los métodos de un componente, permitiendo además cambiar los parámetros de este antes de la ejecución.

2.12 Crear nuevos componentes

El framework nos da la facilidad de extender sus capacidades dándonos una estructura base y una lista de reglas para implementar estas. Todos los componentes deben empezar con el prefijo

MF seguido por el nombre del componente. La clase de este nuevo componente debe heredar ya sea de ComponentBase o de RequestComponent en el caso que éste haga uso de un servicio web.

El constructor contendrá los siguientes parámetros:

- Context (Solo Android): El cual nos sirve para poder generar las vistas.
- JsonView: Lista de parámetros del componente a crear.
- OnCallback: Objeto para definir eventos u otras acciones que el usuario podrá personalizar.
- CustomAdapter: Si su componente utiliza un adaptador, este parámetro servirá para que el usuario pueda personalizarlo.

En la siguiente tabla se muestran los métodos heredados para crear nuestro componente:

Tabla 24
Métodos disponibles para la creación de un nuevo componente

Método	**Descripción**	**Opcional**
setup	Es el método el cuál debe leer el JsonView, proveído por el constructor, e interpretarlo para generar una vista.	No
getValue	Si el elemento tiene la opción de seleccionar un elemento, este método facilitará recuperarlo por los selctores.	Si
load	Este método es utilizado para cargar la data de un servicio web si heredó de RequestComponent.	Si
onSuccess	Controla los datos recibidos de un servicio web.	Si
onError	Controla los errores al ejecutar una consulta a un servicio web.	Si
createView	Permite controlar configuraciones de una vista después de ser renderizada.	Si

2.13 Actualizaciones automáticas

Uno de los problemas de las aplicaciones distribuidas por tiendas de aplicaciones es la actualización de estas. Los usuarios de las aplicaciones tienen la opción de actualizar de forma automática o manual, por lo que las actualizaciones pueden tardar en llegar a los usuarios.

Con el uso de MFJsonView, el framework nos provee la facilidad de ofrecer a los usuarios actualizaciones de forma ligera, solo recibiendo un fichero JSON con la actualización, y guardando está en el almacenamiento del dispositivo y estar disponible sin internet.

Además, el framework nos permite tener compilados componentes, aunque no se usen por ninguna actividad, y estos pueden ser desplegados de forma automática sin requerir una actualización a nivel binarios.

2.14 Atributos de calidad

Para la gestión de la calidad del framework se tomaron en cuenta los siguientes aspectos tomados del estándar ISO/IEC 9126.

Tabla 25
Atributos de calidad

Funcionalidad	Adaptabilidad
	Exactitud
	Seguridad
Usabilidad	Comprensibilidad
	Operatividad
Mantenibilidad	Tolerante a cambios
Fiabilidad	Tolerancia a fallos
	Uso de los recursos
Eficiencia	Comportamiento del tiempo

Detallando los atributos presentados en la tabla 25, la adaptabilidad se logró a través de una arquitectura que permite las mejoras continuas, y no restringir al desarrollador al momento de implementar un requerimiento con una característica no contemplada previamente por el framework; la exactitud fue medida a través de las pruebas unitarias y pruebas de integración por la cual pasó el framework: la seguridad implementada por el framework a través de la utilización de las mejores prácticas para el manejo de memoria y consumo de servicios web; la comprensibilidad y la operatividad fue corroborada a través de una encuesta hecha a personas con experiencia en el desarrollo de aplicaciones web y móviles; la tolerancia al cambio fue demostrada a través de la arquitectura, la cual es clara y concisa sobre su forma de uso y posterior mantenimiento; la tolerancia a fallos fue comprobada con las pruebas que se hicieron a cada uno de los componentes de forma individual, integrada y adicionalmente con un caso de estudio; el uso de recursos fue estudiado para poder hacer un uso responsable de los recursos informáticos tanto para el desarrollador como para el usuario final, utilizando las mejores prácticas; el comportamiento del tiempo fue medido con pruebas de tiempos de respuestas de los diferentes servicios y comparados con librerías terceras, adicionalmente cada uno de los procedimientos internos del framework tienen una complejidad N y lineal en el tiempo.

Los resultados obtenidos durante el proceso de pruebas, como ser observa en la tabla 26, y corregidos posteriormente llegando para todos los componentes un nivel de fiabilidad del 100%. El resto de las características de calidad mencionadas fueron aplicadas durante la creación de la arquitectura y del mismo framework.

Tabla 26
Pruebas de adaptabilidad

Componente	**Número de pruebas**	**Resultado correcto**	**Porcentaje**
MFButton	50	48	96%
MFLabel	10	10	100%
MFInput	30	29	97%
MFCombo	50	49	98%
MFList	80	76	95%
MFTable	100	95	95%

CAPÍTULO 3

CASO DE ESTUDIO

Con la finalidad de demostrar la funcionalidad del framework en estudio, en este capítulo se propone un caso de estudio donde se mostrarán las principales características que presenta el framework, incluyendo las actualizaciones automáticas y algunas situaciones prácticas, lo que nos permitirá validar el trabajo propuesto.

3.1 Problema

La aplicación desarrollada con el framework para demostrar sus características se desarrollará ante un supuesto problema o requerimiento de que los docentes de la UCSM necesitan un menú dinámico donde al abrir, se visualicen eventos de la Universidad, notas de alumnos por fase, búsqueda de libros en biblioteca, el horario. Es decir, se trata de un aplicativo denominado UCSMDocentes pensado para el uso de los docentes de la Universidad Católica de Santa María. A continuación, se presenta algunas de las opciones consideradas:

- Login
- Centro de noticias
- Visualizar notas registradas
- Búsqueda de libros de biblioteca
- Horario

3.2 Requerimientos técnicos

3.2.1 Requerimientos técnicos para el desarrollo

Para la implementación de la aplicación se necesita una MacBook lo cual es un requisito para

el despliegue en iOS.

3.2.2 Requerimientos técnicos para los servicios

Para el despliegue de los servicios web se usaron los siguientes recursos:

- Servidor Linux
- Base de datos PostgreSQL
- Python con el módulo Flask
- Apache2

3.3 Servicios web

En esta sección se presentarán los servicios que serán consumidos por la aplicación.

Tabla 27
Servicios web - UCSMDocentes

Nombre	Descripción	Estructura petición
tLoginDoc	Valida el inicio de sesión, y devuelve datos del usuario. Según la respuesta, las acciones lanzar actividad o mostrar mensaje según la respuesta	{ "user": "…", "pass": "…" }
wsApp	Recupera el menú de opciones del usuario en formato JSON y con la estructura para ser usado por un MFJsonView	{ "cnrodni": "…" }
tSecGrupDoc	Recupera la lista de secciones y grupos que enseña el docente logueado. Estos datos son llenados en un MFCombo.	{ "cnrodni": "…" }
tDocCursosNotas	Recupera las notas de un curso y grupo seleccionado. Con los datos recuperados rellena el componente MFList y despliega las notas por fase como detalle.	{ "cnrodni": "…", "id": "…" }
consultaLibro	Busca la existencia de un libro según el título, autor o contenido. Llena estos datos en el componente MFTable	{ "titulo": "…" }

3.4 Utilizando el framework

En esta sección se muestran las pantallas y la estructura del archivo JSON que utiliza el framework para renderizar la app.

3.4.1 Login

En esta pantalla se ve el uso de los diferentes contenedores, en primer lugar, el MFRelativeLayout el cual nos permite situar todos los componentes al centro de la pantalla. A continuación, un MFLinearLayout con orientación vertical, dentro de este, dos entradas de texto, uno de tipo string y otro de tipo password. En la parte inferior dos botones, uno que tiene como acciones, recuperar los valores dentro de las entradas de texto y enviarlas a un servicio web, el cual se encargará de validar el inicio de sesión, si e correcto se lanzará la siguiente actividad, sino se mostrará el mensaje de error recibido por el servidor.

Figura 14: Caso de estudio - Login

```
{
  "mf-type": "RelativeLayout",
  "width": "match",
  "height": "match",
```

```
"color": "#2E8B57",
"childs": [
 {
  "mf-type": "LinearLayout",
  "orientation": "vertical",
  "width": "match",
  "height": "wrap",
  "gravity": "center",
  "childs": [
   {
    "mf-type": "image",
    "bitimg": "logo_ucsm_docentes",
    "hiddenProgress": true,
    "width": "550",
    "height": "550",
    "margin": "60",
    "layout_gravity": "center_horizontal"
   },
   {
    "mf-type": "input",
    "name": "txtUser",
    "width": "match",
    "input_type": "number",
    "placeholder": "Usuario",
    "margin": "20 20",
    "textColor": "#eeeeee"
   },
   {
    "mf-type": "input",
    "name": "txtPassword",
    "hintColor": "#ff0000",
    "input_type": "password",
    "width": "match",
    "placeholder": "Contraseña",
    "margin": "0 20 80",
    "textColor": "#eeeeee"
   },
   {
    "mf-type": "LinearLayout",
    "layout_gravity": "center_horizontal",
    "width": "wrap",
    "height": "wrap",
    "orientation": "vertical",
    "childs": [
     {
      "mf-type": "button",
      "layout_gravity": "center_horizontal",
      "name": "btnAceptar",
      "text": "Aceptar",
      "padding": "5",
      "color": "#ffffff",
      "textColor": "#000000",
      "actions": [
       {
        "action": "ajax",
```

```
            "params": {
              "url": "http://apps.ucsm.edu.pe/WS/tLoginDoc.php",
              "params": {
                "user": "$txtUser$getValue",
                "pass": "$txtPassword$getValue"
              },
              "success": [
                {
                  "action": "alert",
                  "params": {
                    "text": "logueado"
                  }
                },
                {
                  "action": "save-session",
                  "params": {
                    "params": [
                      {
                        "id": "LOGIN",
                        "data": "$MF$response$getValue"
                      },
                      {
                        "id": "USER",
                        "data": "$txtUser$getValue"
                      }
                    ]
                  }
                },
                {
                  "action": "launch",
                  "params": {
                    "activity": "MainActivity"
                  }
                }
              ],
              "failed": [
                {
                  "action": "alert",
                  "params": {
                    "text": "DNI o contraseña incorrecta"
                  }
                }
              ]
            }
          }
        ]
      },
      {
        "mf-type": "button",
        "text": "Olvidé mi contraseña",
        "style": "simple",
        "layout_gravity": "center_horizontal",
        "textSize": "12",
        "textColor": "#EEEEEE",
        "actions": [
```

```
            {
              "action": "alert",
              "params": {
                "text": "Escribir a efb.devs@gmail.com"
              }
            }
          ]
        }
      ]
    }
  ]
}
```

Código 1: Caso de estudio - Login

3.4.2 Pantalla de inicio – Menú inferior

Para hacer el menú inferior hacemos uso del componente MFModernBar

Figura 15: Caso de estudio - Menú inferior

```
{
  "mf-type": "modern_bar",
  "width": "match",
  "height": "match",
  "childs": [...]
}
```

Código 2: Caso de estudio - Menú inferior

3.4.3 Eventos / Menú dinámico

Esta pantalla muestra el componente MFJsonView que por defecto tiene configurada una webview que mostrará la página de eventos de la universidad. Al iniciar la aplicación y renderizar este componente hace una petición al servicio web que le indica si hay una nueva configuración para que sea cargada por el componente. En el caso de tener un pendiente en el sistema de trámites se actualizará la vista con las notificaciones y otras opciones que requiera para completar. Estas actualizaciones automáticas además se utilizaron para habilitar una actividad para aprobar documentos con la huella digital del responsable.

Figura 16: Caso de estudio - Eventos/Manú dinámico

```
{
  "title": "Home",
  "icon":"icon_home",
  "mf-type":"LinearLayout",
  "orientation":"vertical",
  "width": "match",
  "childs": [
    {
      "mf-type": "LinearLayout",
      "orientation": "horizontal",
      "width": "match",
      "height": "match"
    },
    {
      "mf-type": "json_view",
      "name": "main",
      "width": "match",
      "height": "match",
      "cache": true,
      "ajax": {
        "url": "http://apps.ucsm.edu.pe/UCSMMTA/wsApp.php",
        "params": {
          "Id": "RecuperarMenu",
          "ccodusu": "$MF$Session$LOGIN$CCODDOC"
        }
      }
    }
  ]
}
```

Código 3: Caso de estudio - Eventos/Menú dinámico

3.4.4 Notas

Esta pantalla carga la lista de secciones y grupos del usuario que inició sesión, carga la lista de alumnos y muestra en el detalle de la lista sus notas por fase.

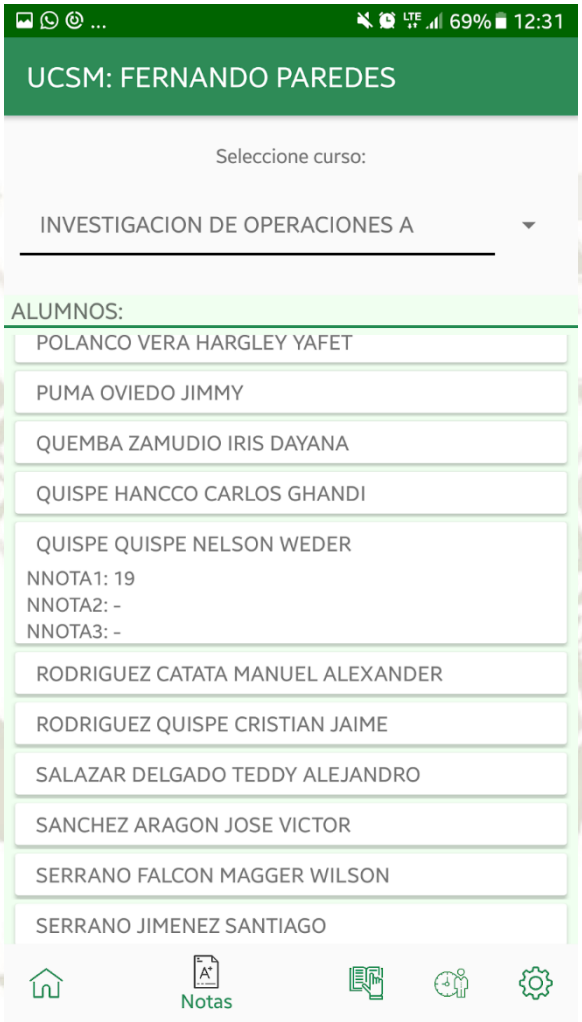


Figura 17: Caso de estudio - Notas

```
{
  "icon":"icon_grades",
  "title":"Notas",
  "mf-type": "LinearLayout",
  "orientation": "vertical",
  "width": "match",
  "childs": [
    {
      "mf-type": "label",
      "value": "Seleccione curso:",
      "width": "wrap",
      "height": "wrap",
      "layout_gravity": "center_horizontal",
      "margin": "50 20 0 20"
```

```
},
{
 "mf-type": "combo",
 "width": "match",
 "height":"wrap",
 "name":"comboCursos",
 "margin": "50 30 5",
 "events": {
  "onSelect":[
   {
    "action":"selector",
    "params":{
     "name":"listNotas",
     "method":"load",
     "params":{
      "id":"$comboCursos$getValue"
     }
    }
   }
  ]
 },
 "data_container": {
  "code": "ID",
  "name": "CDESCRI",
  "fields": [
   {
    "dataIndex": "CDESCRI",
    "type": "string"
   },
   {
    "dataIndex": "CSECGRU",
    "type": "string"
   }
  ],
  "ajax": {
   "url": "http://apps.ucsm.edu.pe/WS/tSecGrupDoc.php",
   "params": {
    "cnrodni":"$MF$Session$LOGIN$CNRODNI"
   }
  }
 }
},
{
 "mf-type":"list",
 "width":"match",
 "height":"wrap",
 "name":"listNotas",
 "color":"#F0FFF0",
 "title":"ALUMNOS:",
 "margin":"70 0 0 0",
 "data_container":{
  "code":"CCODALU",
  "name":"CNOMBRE",
  "fields":[
   {
```

```
            "text":"NOTA 1",
            "dataIndex":"NNOTA1",
            "type":"string"
          },
          {
            "text":"NOTA 2",
            "dataIndex":"NNOTA2",
            "type":"string"
          },
          {
            "text":"NOTA 3",
            "dataIndex":"NNOTA3",
            "type":"string"
          }
        ],
        "ajax":{
          "url":"http://apps.ucsm.edu.pe/WS/tDocCursosNotas.php",
          "params":{
            "id":""
          },
          "autoLoad":false
        }
      }
    }
  ]
}
```

Código 4: Caso de estudio – Notas

3.4.5 Biblioteca

Esta pantalla muestra un componente MFInput dónde se debe ingresar el título, escritor o el contenido de un libro para poder realizar su búsqueda. Se hizo uso de un servicio web proveído por el área de informática de biblioteca y se normalizó la respuesta para adaptarse al formato que solicita el framework y el cual ya fue definido en anteriores secciones.

El componente MFButton también presente, cuenta con una acción de tipo selector que recupera la referencia a la tabla donde será cargada información de los libros que coincidan con la búsqueda y usando un segundo selector se actualizará el parámetro título, extraído del componente de entrada de texto, el cual es el que debe enviarse al servicio.

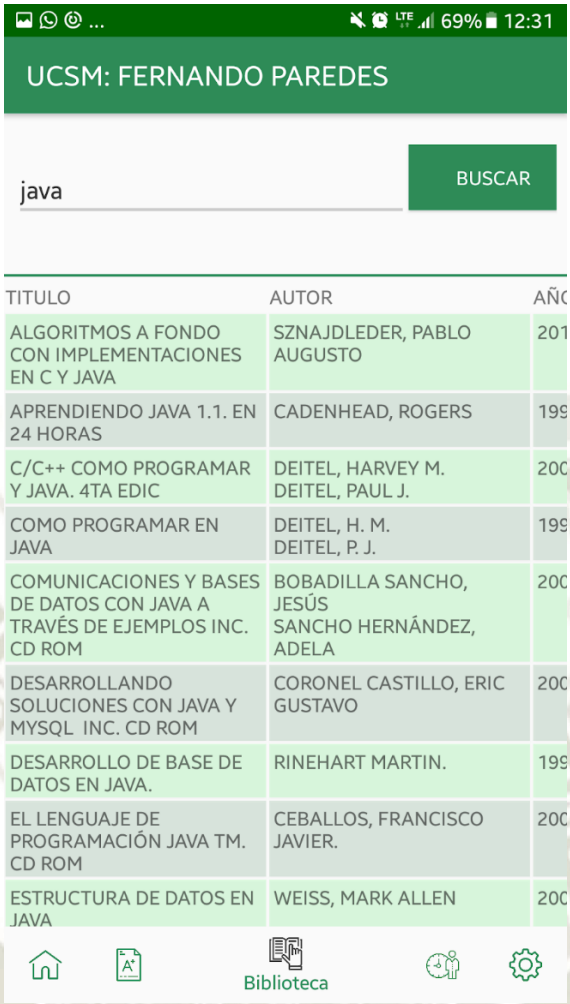


Figura 18: Caso de estudio – Biblioteca

```
{
  "title": "Biblioteca",
  "icon":"icon_library",
  "mf-type": "LinearLayout",
  "orientation": "vertical",
  "width": "match",
  "height":"match",
  "childs":[
    {
      "mf-type":"LinearLayout",
      "orientation":"horizontal",
      "width":"match",
      "height":"wrap",
      "margin":"60 20 25 20",
      "childs":[
        {
          "mf-type":"input",
          "flex":0.7,
          "name":"txtLibro",
          "textColor":"#333333"
        },
        {
```

```
        "mf-type":"button",
        "textColor" : "#FFFFFF",
        "flex":0.3,
        "actions":[
          {
            "action":"selector",
            "params":{
              "name":"tableLibros",
              "method":"load",
              "params":{
                "titulo":"$txtLibro$getValue"
              }
            }
          }
        ],
        "text":"Buscar"
      }
    ]
  },
  {
    "mf-type": "table",
    "name":"tableLibros",
    "data_container": {
      "ajax": {
        "url":"http://apps.ucsm.edu.pe/WS/consultaLibro.php",
        "params": {
          "titulo":""
        },
        "autoLoad":false
      },
      "columns": [
        {
          "text": "TITULO",
          "dataIndex": "titulo"
        },
        {
          "text": "AUTOR",
          "dataIndex": "autor"
        },
        {
          "text": "AÑO",
          "dataIndex": "anyo"
        },
        {
          "text": "EDITORIAL",
          "dataIndex": "editorial"
        }
      ]
    }
  }
 ]
}
```

Código 5: Caso de estudio – Biblioteca

3.4.6 Horario

Esta pantalla muestra el componente MFTable con persistencia, una vez cargado el horario con ayuda del servicio web, este se queda guardado en el almacenamiento del dispositivo y estará disponible cuando no haya internet, y actualizado cuando sea necesario.

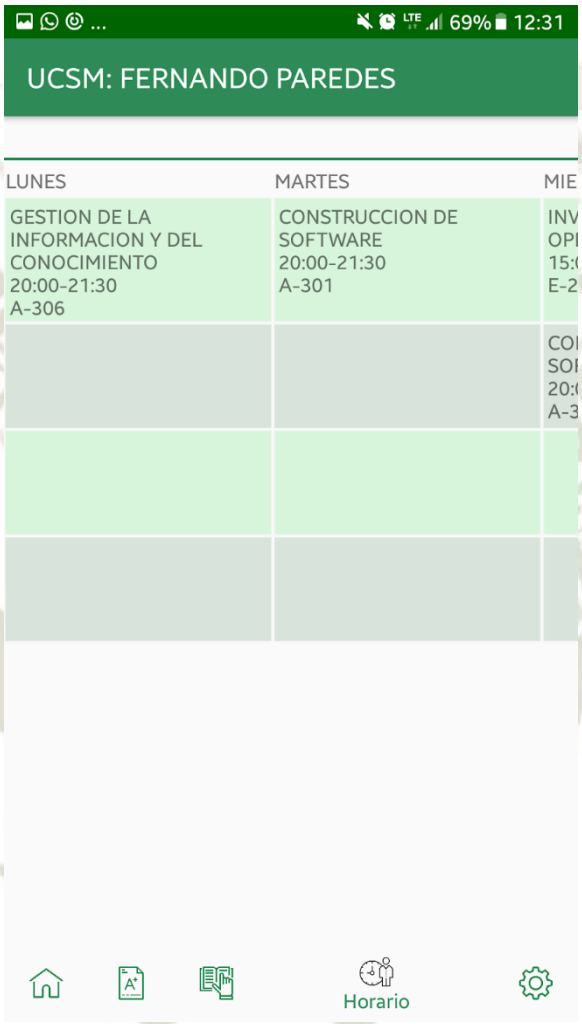


Figura 19: Caso de estudio - Horario

```
{
  "title": "Horario",
  "icon":"icon_schedule",
  "mf-type": "LinearLayout",
  "orientation": "vertical",

  "width": "match",
  "height":"match",
  "childs":[
    {
      "mf-type":"table",
      "height":"match",
```

```
    "data_container": {
      "persistant":{
        "name":"horario"
      },
      "ajax": {
        "url":"http://apps.ucsm.edu.pe/WS/horarioDocente.php",
        "params": {
          "cnrodni":"$MF$Session$LOGIN$CNRODNI"
        },
        "events":{
          "onLoad":[
            {
              "action":"callback",
              "id":1
            }
          ]
        }
      },
      "columns":[
        {
          "text":"LUNES",
          "dataIndex":"LUN"
        },
        {
          "text":"MARTES",
          "dataIndex":"MAR"
        },
        {
          "text":"MIERCOLES",
          "dataIndex":"MIE"
        },
        {
          "text":"JUEVES",
          "dataIndex":"JUE"
        },
        {
          "text":"VIERNES",
          "dataIndex":"VIE"
        }
      ],
      "data":[
      ]
    }
  }
 ]
}
```

Código 6: Caso de estudio – Horario

3.5 Pruebas

Tabla 28
Pruebas realizadas para el caso de uso

ID Caso	Requerimientos	Características	Pasos	Resultados esperados	Resultado
CP01	Validar inicio de sesión incorrecto.	Usuario: 29455317 Contraseña: 123456	1. Ingresar a la aplicación 2. Ingresar usuario y contraseña. 3. Clic en “Iniciar Sesión”	Mostrar mensaje de error.	OK
CP02	Validar inicio de sesión correcto	Usuario: 29455317 Contraseña: 29455317	1. Ingresar a la aplicación 2. Ingresar usuario y contraseña. 3. Clic en “Iniciar Sesión”	Cargar actividad principal y establecer el nombre del docente en el título.	OK
CP03	Guardar datos de sesión	Acciones configuradas para ser realizadas después del inicio de	1. Inicio de sesión	Guardar los datos devueltos por el servicio, y asignarlo en	OK

		sesión		un diccionario con el identificador establecido por el programador.	
CP04	Lanzar actividad no existente	Actividad no existente: MainActivity2	1. Configurar botón con la acción “launch”	Controlar error y lanzar error por consola para ser visto solo por el programador.	OK
CP05	Lanzar actividad existente	Actividad: MainActivity	1. Configurar botón con la acción “launch”	Abrir actividad especificada y cargar sus elementos.	OK
CP06	Cargar elementos desde un servicio	Establecer menú de la aplicación como tipo json_view	1. Crear un servicio que devuelva la estructura JSON para ser renderizada 2. Crear un elemento de tipo json_view	Una vez lanzada la aplicación, renderizar un elemento vacío hasta que el servicio nos devuelva la vista a renderizar.	OK

CAPITULO 4

RESULTADOS Y DISCUSIÓN

En este capítulo tendremos la oportunidad de comparar el framework propuesto con otras alternativas del mercado para demostrar las ventajas de su aplicación. Además, se verán las dependencias, lenguaje, precio, peso y una estimación del tiempo de carga de las aplicaciones; lo que nos llevará finalmente a la elaboración de las conclusiones.

4.1 Comparación entre frameworks

En este apartado se hará la comparación de dos frameworks que se usan actualmente en el mercado para el desarrollo multiplataforma de aplicaciones móviles. Los frameworks a comparar son Ionic, el cual fue creado por Dritfy. Co en el 2013 (Perez S., 2014); y Xamarin el cual fue creado en el año 2011 y adquirido por Microsoft en el año 2016. (Announcing Xamarin, 2011)

4.1.1 Dependencias

En la siguiente tabla se puede apreciar las librerías y programas que deben ser instalados para que los frameworks comparados funcionen.

Tabla 29
Comparativa de frameworks – Dependencias

Ionic	**Xamarin**	**MF-Framework**
Android Studio Xcode Node.js (56.2MB) Npm Ionic (23.6) Npm Cordova (51.2MB) Java JDK	Android Studio XCode Visual Studio (22GB)	Android Studio XCode

4.1.2 Frameworks dependientes

En la siguiente tabla se puede apreciar los frameworks dependientes para el uso de los framework. Ionic es el único que requiere un framework externo para su uso, entre los cuales su pueden elegir React, Angular o Vue.

Tabla 30
Comparativa de frameworks – Frameworks dependientes

Ionic	**Xamarin**	**MF-Framework**
React Angular Vue	Ninguno	Ninguno

4.1.3 Lenguajes de programación

En la tabla siguiente se muestran los lenguajes utilizados por los frameworks comparados, en Ionic se utilizan lenguajes orientados a la web, en Xamarin C# y en el caso del framework desarrollado los lenguajes nativos correspondientes a cada plataforma.

Tabla 31
Comparativa de frameworks – Lenguajes de programación

Ionic	**Xamarin**	**MF-Framework**
JavaScript TypeScript	C#	Java y Swift si se desea ampliar framework

4.1.4 Costo

En la siguiente tabla se muestran los costos de las licencias de uso de los frameworks comparados, el único que exige una licencia mensual, es el caso de Xamarin. Ionic no exige la compra de una licencia, pero se puede utilizar su dashboard con funciones mejoradas lo cual si implica una versión paga.

Tabla 32
Comparativa de frameworks - Costo

Ionic	**Xamarin**	**MF-Framework**
Gratis Dashboard Online con suscripción (opcional)	$45 al mes - versión profesional	Gratis

4.1.5 Peso del proyecto

En la siguiente tabla se muestra una comparativa en el tamaño que ocupa en disco un proyecto vacío de los frameworks comparados, siendo el desarrollado para esta tesis el más liviano.

Tabla 33
Comparativa de frameworks – Peso de un proyecto vacío

Ionic	**Xamarin**	**MF-Framework**
425MB	127MB	28.8MB

4.1.6 Peso de aplicación

En la siguiente tabla se muestra la comparativa del peso de una aplicación vacía, sin funcionalidad, en los frameworks comparados. El framework al utilizar el lenguaje nativo y componentes reutilizables es el más ligero.

Tabla 34
Comparativa de frameworks – Peso de una aplicación vacía

Ionic	**Xamarin**	**MF-Framework**
10.43MB	24.53MB	2.69MB

Como se observa en la siguiente figura, el framework que más dependencias necesita es Ionic,

necesitando 4 dependencias, dos de las cuales son para la instalación del wrapper Cordova. Además, se tienen que aprender React, Angular o Vue para empezar con el desarrollo.

La herramienta de desarrollo Xamarin necesita instalar Visual Studio, el cual en su versión profesional tiene un costo de $45 y ocupa 22GB en el disco duro.

En la figura 21 se muestran un gráfico con el número de dependencias para el desarrollo de un aplicativo sencillo que hace uso de una tabla para mostrar datos obtenidos de un servicio web.

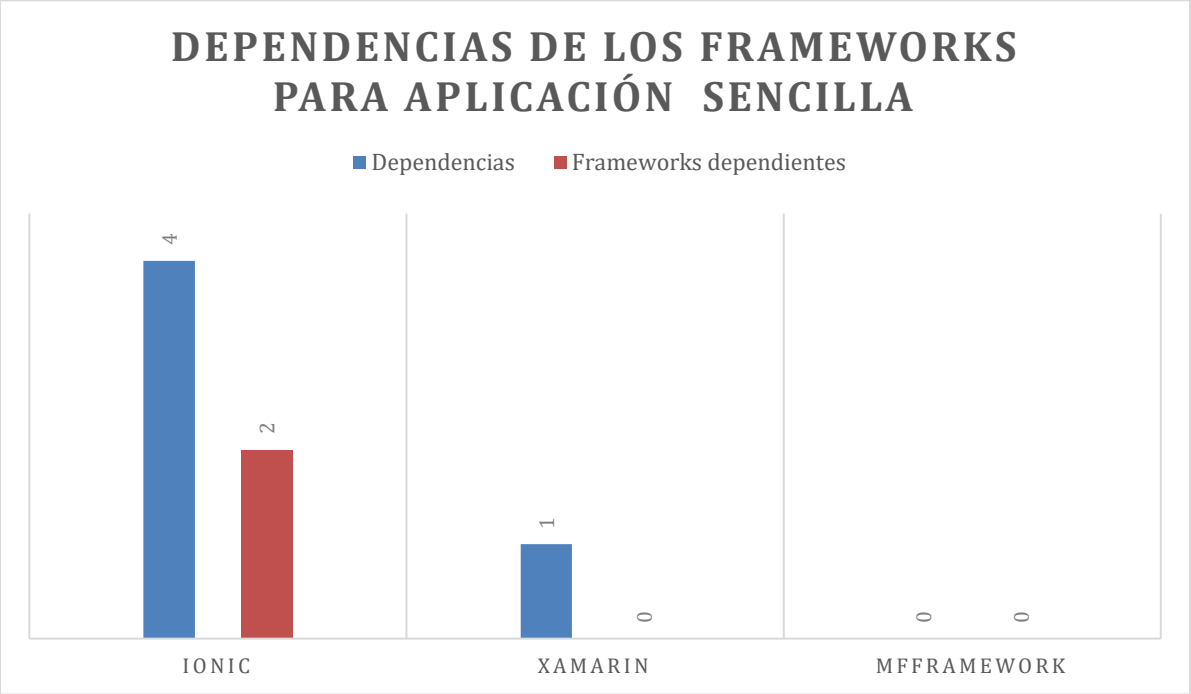


Figura 20: Comparativa de las dependencias de los framework
Fuente: Propia

El MFFramework no requiere ninguna herramienta ni complemento para la implementación de las aplicaciones. Las librerías usadas para la versión en Android son solo las distribuidas por Google y referentes a la parte visual, Material Design.

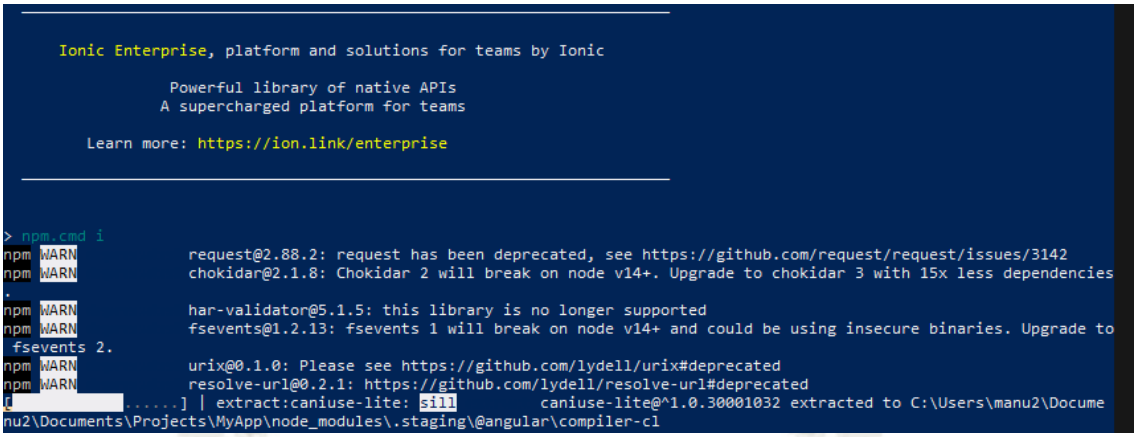


Figura 21: Advertencias en la instalación de dependencias del framework Ionic

Como se ve en la figura 20, durante la instalación de ionic a través del gestor de paquetes npm, aparecen diversas advertencias, indicándonos las librerías que ya no tienen soporte o no lo tendrán en un futuro. Además, nos advierte los paquetes que generarían problemas si se actualiza nodejs a su versión 14 o superior.

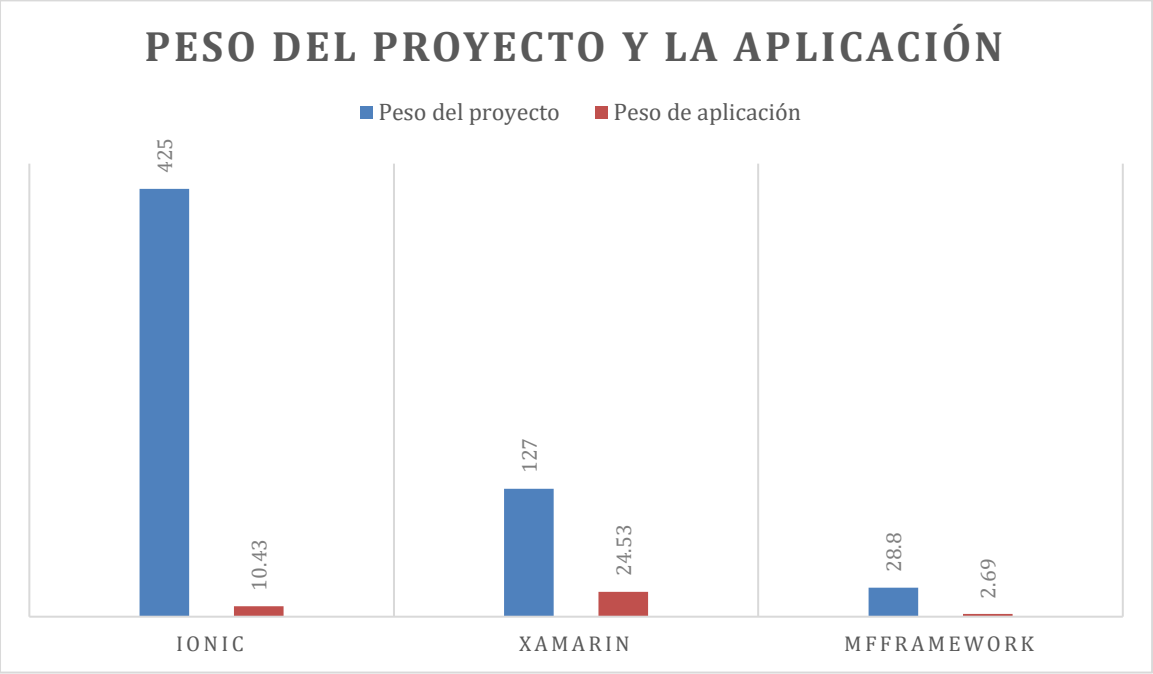


Figura 22: Cuadro comparativo - Tamaño de la aplicación
Fuente: Propia

La figura 21 nos muestra un cuadro con el peso en MB de los proyectos y las aplicaciones generadas con los diferentes frameworks que estamos analizando. El proyecto en blanco más pesado corresponde a Ionic, el cual una vez compilado queda en tan solo 10MB. En segundo

lugar, tenemos el proyecto en Xamarin el cual tiene un peso menor, pero en la aplicación ya compilada el peso se extiende hasta los 24MB.

MFFramework da solución a ambas problemáticas, el peso del proyecto en blanco es de 28MB siendo el menor comparándolos con los otros frameworks comparados. Y el peso del aplicativo compilado apenas llegando a los 2.69MB.

La siguiente ventaja con la que cuenta MFFramework es que el peso se mantiene casi constante mientras la aplicación va creciendo, esto se logra gracias a que los componentes son compilados una sola vez y se pueden utilizar múltiples veces por diferentes actividades. El peso tan solo se incrementa según el tamaño del archivo JSON que necesita el framework para realizar su trabajo de renderizado. Por lo cual el peso aproximado por cada nueva actividad es de 1KB.

Se realizó la estimación de tiempo de carga de las aplicaciones consiguiendo los siguiendo los siguientes tiempos de carga.

Tabla 35
Cuadro comparativo – Tiempos de carga

	Ionic	**Xamarin**	**MFFramework**	**% de mejora**
Primera prueba	2s 14ms	1s 10ms	25ms	88.3% - 44.3%
Segunda prueba	3s 26ms	1s 5ms	26ms	92.0% - 82.7%
Tercera prueba	1s 20ms	1s 20ms	24ms	80.0% - 80.0%
Cuarta prueba	2s 22ms	1s 13ms	25ms	88.7% - 77.9%
Promedio	2s 20ms	1s 12ms	25ms	88.6% - 77.7%

4.2 Comparación con proyectos similares

En esta sección se hará la comparación de los tiempos de desarrollo en aplicaciones móviles similares al caso de estudio.

4.2.1 Aplicación de consultas académicas para estudiantes – Tesis Universidad Nacional "Hermilio Valdizán"

La aplicación desarrollada en esta tesis tiene las siguientes funcionalidades.

- Login: Valida el ingreso a la aplicación.
- Notas: Listado de cursos actuales con sus respectivas calificaciones.
- Horario: Muestra una tabla con los horarios del estudiante.
- Record académico: Listado de cursos cursados con los promedios obtenidos.

Esta propuesta fue desarrollada en 4 meses según se indica en su documento, estando disponible sólo para el sistema operativo Android. No se especifican los tiempos detallados del desarrollo.

El backend fue desarrollado en PHP y como gestor de base de datos SQL Server.

El entorno de desarrollo utilizado fue Eclipse con el SDK de Android.

La aplicación tiene funcionalidades similares al del caso de estudio presentado, el tiempo de desarrollo de la aplicación con el framework es de alrededor de 1 semana, los servicios web 2 semanas, y el despliegue para las plataformas Android y iOS. Demostrando que el uso del framework presenta una mejora considerable en tiempos.

4.2.2 Aplicación académica para docentes – Tesis Universidad Nacional José María Arguedas

La aplicación desarrollada en esta tesis tiene las siguientes funcionalidades.

- Login: Valida el ingreso a la aplicación.
- Notas: Listado de cursos actuales con sus respectivas calificaciones.
- Marcar asistencia: Muestra el listado de estudiantes con un botón para el registro de asistencia.

Esta propuesta fue desarrollada en 70 días según se indica en su documento, estando disponible sólo para el sistema operativo Android. El tiempo que llevó la codificación fueron 30 días según el cronograma de ejecución de la tesis.

El backend fue desarrollado en PHP y como gestor de base de datos MySQL.

El entorno de desarrollo utilizado fue Android Studio.

La aplicación tiene funcionalidades similares al del caso de estudio presentado, el tiempo de desarrollo de la aplicación con el framework es de alrededor de 1 semana, los servicios web 2 semanas, y el despliegue para las plataformas Android y iOS. Demostrando que el uso del framework presenta una mejora considerable en tiempos.

4.4 Mejoras

Cómo un trabajo posterior se tiene planteado incluir nuevos paradigmas, como es la programación reactiva para un manejo de flujos de datos basado en el patrón observador y estos puedan alterar de forma directa las interfaces, las cuales pueden ser iterables y condicionales.

4.5 Validación

Para la validación, se realizó una encuesta a diferentes desarrolladores de aplicativos móviles y web para conocer su opinión acerca de la facilidad del uso del framework.

Las preguntas planteadas fueron las siguientes, acompañadas con el siguiente video https://youtu.be/43HsJY0B4dA:

1. ¿Cuál es su cargo en su puesto de trabajo?
2. ¿Cuántos años desarrolla aplicaciones (web, móvil, escritorio, etc.)?
3. ¿Encuentra sencillo el uso del framework presentado en el video?
4. Según la imagen, ¿le parece entendible la configuración para crear un ComboBox?
5. Sabiendo que el framework pesa aproximadamente 1mb y puede ser utilizado en conjunto con otras herramientas, ¿lo utilizaría para agilizar sus tiempos de desarrollo?
6. En el caso que haya marcado "No" a la pregunta anterior, indique el motivo

Con las respuestas del cuestionario se llegaron a las siguientes conclusiones:

1. El 60% de los encuestados están actualmente desarrollando aplicaciones móviles.
2. Los encuestados tienen al menos 4 años de experiencia desarrollando aplicativos.
3. Todos los encuestados encuentran sencillo y entendible el uso del framework.
4. El 78.6% por ciento lo usaría en sus proyectos. El resto no lo usuaria por su poco tiempo en el mercado y por políticas en sus centros laborales.

El cuestionario y sus respuestas se pueden observar en el Apéndice B y Apéndice C respectivamente.

4.6 Repositorio

El framework se encuentra disponible desde el repositorio en GitLab https://gitlab.com/mf_framework

CONCLUSIONES

La presente tesis se desarrolló con el objetivo de contar con una herramienta que permitiera simplificar el desarrollo de aplicaciones móviles y que éstas puedan ser desplegadas en las plataformas más populares como Android y iOS.

A partir de esta premisa se considera el propósito cumplido, se ha podido llegar a las siguientes conclusiones:

PRIMERO: Se desarrolló un framework capaz de adaptarse a los requerimientos de las aplicaciones sin tener que re-implementar componentes. Además, se diseñó una arquitectura capaz de escalar con el uso de componentes enriquecidos, contenedores de datos y adaptadores para ser utilizados en las vistas. El framework permite actualizaciones automáticas sin necesidad de ser desplegadas por los respectivos markets.

SEGUNDO: Se hizo la comparativa entre Ionic, Xamarin y el framework desarrollado para la presente tesis, llegando a los siguientes resultados. La cantidad de dependencias y mayor acoplamiento de software lo tiene el framework ionic. El peso de las aplicaciones compiladas con Xamarin es la mayor. MFFramework evita el problema de acoplamiento y reduce el peso del aplicativo resultante, usando sus propios componentes sin librerías externas, y usando su motor de renderizado para tener los componentes pre compilados y poder ser usados por varias actividades.

TERCERO: Se utilizó la metodología Kanban para el desarrollo del framework para así poder gestionar las tareas y tiempos requeridos de forma óptima. Se crearon tarjetas Kanban con título, descripción, prioridad y duración estimada de cada

tarea, a través de distintas etapas, para cubrir los requerimientos funcionales del framework.

CUARTO: Se demostró la aplicabilidad del framework a través de un caso de estudio, se desarrolló una aplicación móvil para los docentes de la Universidad Católica de Santa María para las plataformas Android y iOS, que cuenta con las opciones de consulta de horarios, visualización de eventos institucionales, consulta de calificaciones por secciones y grupos; y la opción de búsqueda de libros disponibles en la biblioteca de la universidad.

QUINTO: Se demostró una mejora en la apertura de la aplicación del 88.6% con respecto al framework Ionic y del 77.7% con respecto al framework Xamarin.

SEXTO: El uso del Framework propuesto reduce el tiempo de desarrollo de aplicativos, además de brindar la posibilidad de escalar los componentes, desplegar actualizaciones automáticas, y tener una aplicación nativa con el menor tiempo de respuesta y tamaño del compilado reducido. Además, se hizo una comparación con dos desarrollos con características similares llegando a la conclusión que con el uso del framework se obtiene una mejora considerable en el tiempo del desarrollo.

REFERENCIAS

Ahmad, M. (2018). Kanban in software engineering: A systematic mapping study

Announcing Xamarin. (2011). https://web.archive.org/web/20110518030128/http://tirania.org/blog/archive/2011/May-16.html

Benouda H. (2016). Code generation approach for mobile application using acceleo

Chahuillco, J. (2019). Desarrollo de una aplicación móvil Android en la gestión del proceso de seguimiento y tutoría para el programa de mojoramiento de la calidad de formación académica de pregrado de la Universidad Nacional José María Arguedas

Constanzo, M. (2014). Comparación de modelos de calidad, factores y métricas en el ámbito de la Ingeniería de Software

Corbalan, L., Fernandez, J., Cuitiño, A., Delia, L., Cáseres, G., Thomas, P., & Pesado, P. (2018). Development frameworks for mobile devices. A Comparative Study about Energy Consumption

Craven, M. (2013). Native Apps versus Web Apps: Which Is Best for Healthcare Applications?

Dieter, F. (2002), The Web Service Modeling Framework WSM, p. 2-4

Ferreira, C. (2018). An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development.

Fielding, R. (1997). The Apache HTTP Server Project, p. 1

Gebremariam, M. (2016), Towards end-user development of REST client applications on smartphones

Holzinger A, T. (2012), Making apps useable on multiple different mobile platforms: on interoperability for business application development on smartphones, p. 176-189

Inayatullah, M. (2019). Model-Based Scaffolding Code Generation for Cross-Platform Applications

International Data Corp. (2014). Tablet shipments forecast to top total PC shipments in the fourth quarter of 2013 and annually by 2015. http://www.idc.com/getdoc.jsp?containerId=prUS24314413

Jia X., Ebone A. (2018). A Performance Evaluation of Cross-Platform Mobile Application Development Approaches

Kwanghoon, C. (2015), A lightweight approach to component-level exception mechanism for robust android apps

Lardinois, F. (2015). Microsoft Launches Visual Studio Code. A Free Cross-Platform Code Editor For OS X, Linux and Windows

Luhunu L. (2017). Comparison of the experessiveness and performance of template-based code generation tools

Monte-Mor J. (2011). Applying MDA Approach to Create Graphical User Interfaces

Münster, L. (2018), A process-oriented modeling approach for graphical development of mobile business apps

Ombretta, M. (2016), An empirical analysis of energy consumption of crossplatform frameworks for mobile development

Perchat, J. (2013). Component Based Framework to Create Mobile Cross-platform Applications

Perez, S. (2014), Drifty, Makers of The Ionic Mobile Framework

Pinar Muyan (2017), A hands-on cross-platform mobile programming approach to teaching OOP concepts and design patterns

Que P., Guo X. y Zhu M. (2016). A comprehensive Comparison between Hybrid and Native App Paradigms

Raj, T. (2012), A study on approaches to build cross-platform mobiles applications and criteria to select appropriate approach, p. 625-629

Rathod D. (2015). Towards Compisition of Restful Web Services

Ribeiro A, da Silva (2012), Survey on cross-platform and languages for mobile apps, p. 255-260

Rieger, C. (2019). Towards the definitive evaluation framework for cross-platform app development approaches

Severance, C. (2012). Inventing PHP: Rasmus Lerdorf, p. 1-2

Smuty, P. (2012), Mobile development tools and cross-platform solutions, p. 653-656

Timmer, J. (2014), A Fast look at Swift, Apple's new programming language

V. Mobile. (2012). Cross-platform developer tools 2012. Vision Mobile Research, p. 97

Vara, H., Ponciano, Y. (2012). Diseño e implementación de una aplicación móvil de consultas académicas para estudiantes de la UNHEVAL

Wafaa, S. (2015), Taxonomy of cross-platform mobile applications development approaches, p. 3-11

APENDICES

APÉNDICE A –PROYECTO DE TESIS

1 PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1 Planteamiento del Problema

“Una década después de que Apple iniciara la tendencia hacia los teléfonos inteligentes con su primer IPhone, los dispositivos móviles y aplicaciones han sido ampliamente adoptadas. Las aplicaciones empresariales cubren tareas a pequeña escala y soporte del proceso de digitalización la cual da como beneficio” (Christoph Rieger, 2017, p. 3)

Los aplicativos móviles cada vez son más utilizados por su facilidad de uso, portabilidad y sensores que éstos poseen, el desarrollo también se ha visto afectado por este cambio, aumentando así la necesidad de herramientas que brinden la agilidad, confiabilidad y soporte para la ejecución de las labores de codificación. En la actualidad hay dos sistemas operativos que predominan el mercado, los cuáles son Android y IOS.

En el mercado existen soluciones que brindan soporte para el desarrollo para ambas plataformas, no obstante, no se posee una optimización de las aplicaciones para los usuarios finales ni facilidades de uso para los desarrolladores.

El desarrollo nativo nos permite conseguir las características que buscamos para brindar un buen producto a los usuarios finales, eficiencia, almacenamiento optimizado, uso de sensores del móvil, apariencia correspondiente al entorno en que se ejecuta.

Las tareas básicas para el desarrollo de aplicativos móviles utilizando MVC y SOA, referentes a la etapa de implementación son:

a) El modelamiento de las entidades en forma de clases.

b) Representar los casos de uso en forma de interfaces(contratos) de programación.

c) Creación de actividades implementando los contratos establecidos.

d) Diseñar las interfaces de usuario y contenedores de datos.

e) Crear adaptadores de datos para así enlazar las entidades a las vistas creadas.

f) Enlazar eventos generados por las interfaces de usuario al controlador.

g) Consumir servicios del servidor para las tareas necesarias.

En las actividades mencionadas en el anterior apartado, dos son definidas al momento de crear los servicios web, quedando como tareas pendientes pero indispensables “Enlazar los eventos”, “Crear los adaptadores de datos”, “Consumir los servicios” y “Diseñar las interfaces de usuario”.

Por lo tanto, el problema es que no existe una herramienta que nos permita la implementación de aplicaciones móviles nativas en un lenguaje unificado para las plataformas Android y iOS, el cuál sea extensible y compatible con las actualizaciones por parte del sistema operativo.

1.2 Línea y Sub-Línea de Investigación a la que corresponde el problema

Línea: Ingeniería de Software

Sub-línea: Proyecto de Software

1.3 Palabras clave

Framework, JSON, SOA

2 OBJETIVOS DEL PROYECTO

2.1 General

Desarrollar un framework para el desarrollo multiplataforma de aplicativos móviles orientados a servicios usando herramientas nativas

2.2 Específicos

a) Analizar otros frameworks del mercado para comparar sus beneficios y características

b) Reducir el tiempo de desarrollo de aplicativos móviles

c) Diseñar una arquitectura de software que permita la escalabilidad del framework

d) Evaluar el rendimiento y optimización de los aplicativos

e) Validar factibilidad de uso del framework

3 FUNDAMENTOS TEÓRICOS

3.1 Antecedentes del proyecto

Según B. Stackpole (2017), las aplicaciones móviles atraen usuarios de un amplio rango de edades, especialmente personas jóvenes, los cuales usan frecuentemente. Los jóvenes están intrínsecamente motivados por aprender tecnologías móviles emergentes las cuales usan a su día a día. Los profesionales están además interesados en ese hecho, debido al incremento de la demanda del mercado de aplicaciones móviles, trabajos libres en el *boom* del desarrollo de software portables.

En 2017 el uso de teléfonos inteligentes llegó a casi un 64% de la población mundial, y para el año 2020 se espera que sea el 85% de la población. La popularidad de los teléfonos inteligentes fue causada por la amplia variedad de aplicaciones que se podían encontrar en las tiendas virtuales dentro del equipo, las cuales cubrían necesidades en campos tales como la educación, salud y turismo. El objetivo de cualquier compañía de desarrollo es apuntar a la mayor cantidad de usuarios y brindar las mismas aplicaciones para las diferentes plataformas del mercado. Cada plataforma da a los desarrolladores un ambiente de desarrollo integrado (IDE), lenguaje de programación, APIs y una tienda virtual para distribuir las aplicaciones. De esta forma las compañías deben elegir una de las dos alternativas para desarrollar la misma App para diferentes plataformas. La primera alternativa es que los desarrolladores trabajen juntos para producir una aplicación para una plataforma específica al mismo tiempo. Esta alternativa soportará diferentes plataformas secuencialmente, pero gastando mucho tiempo de desarrollo en aprendizaje y familiarizarse con las diferentes plataformas de desarrollo. Otra alternativa es que los desarrolladores sean divididos en dos equipos separados y cada equipo trabaje en una plataforma específica. Esta alternativa soportará el desarrollo de varias plataformas en paralelo, pero requiere más desarrolladores que a primera alternativa y a un mayor costo. Por la

necesidad de lanzar una aplicación rápidamente y más económica se crearon nuevas soluciones para el desarrollo multiplataforma. El concepto principal de las soluciones multiplataforma es desarrollar una vez y correr está en cualquier lugar. Muchas de las soluciones de desarrollo móvil multiplataforma están aún bajo investigación y desarrollo. Algunas soluciones están disponibles para uso comercial pero aún no en su versión final que resuelva el problema entero.

Holzinger (2012) precisa que los dispositivos móviles tienen diferentes tamaños de pantallas y diferente relación-aspecto, tal que estos son requeridos para asegurar que las interfaces de usuario sean escalables. Se utilizan tres framework para probar su funcionalidad y uso: JQuery Mobile, PhoneGap y Titanium Mobile. Después de producir las aplicaciones en todos los framework, se procedió a realizar las actividades de la etapa de testing, lo cual tuvo que ser mediante emuladores, porque los framework no contaban con las herramientas para realizar pruebas.

Raj y Tolety (2012) clasifican los enfoques para el desarrollo multiplataforma en: Enfoque Web, Enfoque Híbrido, Enfoque interpretado y Enfoque multi-compilado. Además, detalla los estilos de aplicaciones: Dirigidos por datos en un servidor, Basados en sensores E/S, Autónomos y Cliente-Servidor. El paper se enfoca en solo un subconjunto de los enfoques de framework para el desarrollo multiplataforma, pero no explica como las soluciones existentes implementan estos enfoques.

Smutny (2012) categoriza las aplicaciones móviles web y nativas en cuatro diferentes configuraciones: Nativas, Híbridas, Aplicaciones Web Dedicadas a medida para cada plataforma, Aplicaciones Genéricas las cuales son aplicaciones Web en el navegador integrado por la plataforma. Además, precisa que los framework deben proveer las siguientes características: (1) Multiplataforma (2) El tamaño de la aplicación final debe ser ligero para su distribución (3) Optimizada para pantallas táctiles. Smunty se enfoca en el desarrollo de

aplicaciones web con diferentes frameworks: SenchaTouch, jQuery Mobile, Wink, M-Project, Jo, Titanium, PhoneGap, DHTMLX Touch

Ribeiro y da Silva (2012) describen y analizan seis herramientas: Rhodes, PhoneGap, DragonRAD, Appcelerator Titanium, mobl y Canappi mdsl. Los autores las clasifican en cuatro categorías: En tiempo de ejecución, Web dentro de una aplicación nativa, App Factory y Lenguaje específico de dominio.

El desarrollo de aplicaciones móviles es un caso especial de desarrollo de software porque el desarrollador debe considerar diferentes aspectos tal como el corto ciclo de vida y las capacidades de los dispositivos, portabilidad y especificaciones de los dispositivos tales como tamaño de las pantallas, el diseño interfaz de usuario, la navegación, seguridad y la privacidad el usuario. El ciclo de vida del desarrollo de aplicaciones móviles consiste en: (1) Analizar la idea, (2) Diseñar la interfaz de usuario, (3) Desarrollar la aplicación usando las herramientas y programas para la plataforma elegida, (4) Probar la aplicación en diferentes dispositivos y (5) Publicar la aplicación en la tienda virtual. Las nuevas funciones y actualizaciones para la aplicación son lanzadas en versiones sucesivas a través de la tienda virtual. Para desplegar la aplicación en múltiples plataformas se debe repetir el ciclo de vida visto anteriormente, excepto por la etapa de análisis de requerimientos. Las limitaciones de las aplicaciones móviles son:

A. Recursos limitados
    a. Poder computacional limitado
    b. Almacenamiento limitado
    c. Conectividad
B. Heterogeneidad de sistemas operativos
    a. Diferentes ambientes de operación de las aplicaciones (SDK)
C. Heterogeneidad de dispositivos

a. Diferentes capacidades computacionales

b. Diferentes configuraciones de hardware

c. Diferentes tamaños de pantallas

d. Métodos de entrada: Pantalla táctil, teclado, etc.

D. Experiencia de usuario

E. Mantenimiento de la aplicación

F. Desarrollar múltiples veces, una para cada plataforma

Aunque el desarrollo de aplicaciones móviles tiene muchas restricciones y desafíos, el desafío principal es dirigir como desarrollar una vez y desplegar esta en diferentes plataformas para ahorrar tiempo y esfuerzos.

3.2 Bases teóricas de la Investigación

3.2.1 Framework

Según Guerrero & Recaman, (2009) “Un Marco de Trabajo del inglés Framework, se define como un conjunto de componentes físicos y lógicos estructurados de tal forma que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información”.

Saavedra, (2009) “Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado, a partir de una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación".

Guerrero, Suárez & Gutiérrez, (2013) “Los marcos de trabajo contienen patrones y buenas prácticas que apoyan el desarrollo de un producto y un proceso con calidad”.

Un framework para el desarrollo multiplataforma móvil permite al desarrollador, construir la aplicación una sola vez y desplegarla en varias plataformas: Android, iOS. Un motor hecho en JavaScript es artefacto común que está incluido en todas las plataformas, el framework debe

incluir la integración con las características de los móviles.

a. Gráficos nativos
b. Acelerómetro
c. Camara
d. Contactos
e. Ubicación (GPS)
f. Correo
g. Orientación (vertical u horizontal)
h. SMS
i. Acceso a archivos
j. SQLite
k. Información del dispositivo

3.2.2 JSON

ECMA-404, (2017) JSON es una sintaxis de texto que facilita el intercambio de data entre todos los lenguajes de programación. Es una sintaxis de llaves, corchetes, comas y comillas que es útil en muchos contextos, perfiles, y aplicaciones. JSON proviene de JavaScript Object Notation y fue inspirado por los objetos literales de JavaScript. JSON intercambia los números flotantes en forma de texto para facilitar el transporte independiente del lenguaje que lo produzca y lea.

JSON provee una notación simple para expresar colecciones de pares clave/valor. Muchos lenguajes de programación representan esta colección cómo record, struct, dict, map, hash o un objeto.

Además, provee soporte para representar una lista ordenada de valores. Todos los lenguajes de programación tienen la característica de guardar esta representación, ya sea como list, array,

vector.

Cómo restricciones se tienen que no se puede usar para intercambiar data binaria, ni en forma de grafo acíclico.

Los valores aceptados como valor en la sintaxis son:

a) Objeto
b) Lista
c) Número
d) Texto
e) true
f) false
g) null

3.2.3 SOA

Microsoft, (2016) lo define como un estilo de diseño de software donde los servicios son proveídos a otros componentes, a través de un protocolo de comunicación en la red. Los principios básicos de la arquitectura orientada a servicios son independientes de los productos y tecnologías.

3.2.4 Aplicaciones Nativas

Son aplicaciones que son desarrolladas usando las herramientas y lenguaje de programación para cada plataforma. Estas aplicaciones solo funcionan en la plataforma para la cual fue diseñada. Poseen todas las características y funciones que provee el dispositivo y sistema operativo objetivo.

3.2.4 Clasificación de Frameworks

Los framework para desarrollo de aplicaciones multiplataforma pueden ser divididos en cuatro

clases:

1) Enfoque web: Consiste en desarrollar aplicaciones web utilizando HTML, CSS, JavaScript las cuales son accedidas por medio del navegador del móvil.

2) Enfoque híbrido: Son aplicaciones que usan un WebKit para el renderizado y proporcionan APIs para acceder a las características del móvil. Un ejemplo de este enfoque es: PhoneGap, Apache Cordova.

3) Enfoque interpretado: Da a los desarrolladores la posibilidad de escribir código usando un lenguaje diferente al nativamente soportado por la plataforma, el aplicativo incluye un intérprete lo cual puede verse reflejado en el pobre rendimiento.

4) Enfoque multi-compilado: Permite a los usuarios utilizar un solo lenguaje de programación, después de la compilación resulta en varias aplicaciones nativas.

4 PRESENTACIÓN DEL PROYECTO

4.1 Justificación

La demanda de aplicativos móviles en el mercado está creciendo con el paso del tiempo, todos los negocios están buscando expandir su alcance, lo cual solo se puede lograr dando el salto al mundo digital. La optimización de los procesos también requiere del uso integrado de plataformas para la toma de decisiones. Los teléfonos inteligentes han logrado su apogeo por su facilidad de uso y portabilidad.

Por la necesidad de crear aplicaciones móviles de manera más eficiente y cumpliendo los requisitos de calidad y confiabilidad, varias empresas y desarrolladores de software han empezado a utilizar frameworks que le ayuden a ésta tarea. Además, al no existir una sola plataforma, los frameworks se han adaptado para que el despliegue se haga sobre múltiples plataformas.

El tamaño resultado de las aplicaciones que son desplegadas por los frameworks del mercado son de tamaño excesivo por la cantidad de dependencias y la poca libertad al desarrollador para optimizar el núcleo de renderización y lógica de negocio.

Por parte de los desarrolladores, una de las tareas que conllevan más tiempo es el modelamiento de datos y su representación en la interfaz para el usuario. Por lo que el framework plantea la existencia de un repositorio en formato JSON dónde se especificarán los atributos con sus respectivos tipos. En la integración del modelo con la interfaz de usuario a través de los adaptadores, se especificará un archivo JSON que proporcionará como mostrar la data del modelo en los respectivos adaptadores a ser mostrados por la interfaz gráfica.

Con este proyecto se plantea solucionar este problema, brindando de esta forma un framework para los desarrolladores que busquen la agilización del proceso de desarrollo de aplicativos

móviles en un ambiente nativo, para dar a sus usuarios el tiempo de respuesta que esperan, una aplicación ligera y con características nativas.

4.2 Resumen del proyecto

4.2.1 Descripción del proyecto a medio y largo plazo

Este proyecto se basa en la implementación de un framework para el desarrollo multiplataforma de aplicaciones móviles que brinde a los desarrolladores una herramienta con la cual se reduzca el tiempo de desarrollo, utilizando el formato de modelado JSON para reducir el coste de almacenamiento, aumentar la comprensión del código y evitar la redundancia de uso de etiquetas XML.

Además, se busca brindar un soporte de desarrollo más intuitivo al ser las interfaces de usuario, modelado de datos e interacción con el cliente una sola labor, y dejar la tarea de lógica de negocio al servicio que le corresponde la tarea de back-end.

4.2.2 Usuario del proyecto

El proyecto está diseñado para ser utilizado por desarrolladores o personas con un conocimiento básico de codificación que deseen desarrollar aplicaciones para dispositivos móviles de una manera más simple y rápida.

4.2.3 Beneficios

- Reducción del tiempo de desarrollo de aplicaciones móviles

- Menor coste en la implementación de nuevos sistemas

- Modelado de interfaces de usuario, modelos de datos y operatividad utilizando un lenguaje más cercano al natural.

4.2.4 Impacto y sostenibilidad del proyecto

El framework en su núcleo usa el lenguaje de programación nativo para cada plataforma, por lo cual de ser necesario algún cambio o ampliación de funcionalidades se tiene acceso al código responsable de cada acción, además la arquitectura basada en componentes enriquecidos nos permite el rápido entendimiento y extensión de componentes creados y nuevos.

Al requerirse nuevos componentes el desarrollador deberá utilizar las herramientas que brinda el framework y será con esto compatible y reconocido como parte de las funcionalidades para su futuro uso dentro de la aplicación.

La sostenibilidad tecnológica radica en el uso del lenguaje de programación soportado por cada plataforma, el uso de SOA conjunto al estándar de intercambio de datos JSON, los cuales ya llegaron a un nivel de madurez en proyectos de software.

4.2.5 Riesgos que debemos afrontar

Riesgo	Probabilidad	Impacto
El desarrollador no entienda el funcionamiento del framework o tenga dudas al utilizarlo	Baja	Serio
Cada plataforma tiene sus dependencias propias, las cuales el framework no puede evitarlas, como el uso de un equipo Apple para compilar los proyectos para iPhone	Alta	Tolerable

Menos reutilización que la prevista	Baja	Tolerable
Ninguna reducción en el desempeño	Baja	Serio
Rediseño de componentes ya creados	Media	Insignificante

Entre los principales riesgos son:

- El desarrollador no entienda el funcionamiento del framework o tenga dudas al utilizarlo.
- Cada plataforma tiene sus dependencias propias, las cuales el framework no puede evitarlas, como el uso de un equipo Apple para compilar los proyectos para IPhone.

5 PLAN DE IMPLEMENTACIÓN DEL PROYECTO

5.1 Definición del proyecto

5.1.1 Aspectos técnicos

El proyecto será desarrollado con los lenguajes nativos correspondientes a las plataformas: Java (Android) y Swift (iOS)

5.1.2 Aspectos económicos

Para la realización del proyecto se usará Android Studio, XCode y los lenguajes de programación Java, Swift.

Costos	
Servicio de luz	S/.60.00
Internet	S/.100.00

Otros Costos			
Recurso	Costo	Periodo	Costo 5 años
Licencia Android	$25	De por vida	$25
Licencia iOS	$100	Un año	$500

Ingresos	
Precio por desarrollo de aplicaciones en un mes (5 proyectos)	S/.9000

Beneficio: $PD/(CS + CA + CI)$

- *PD: Precio por desarrollo*
- *CS: Costo de servicios (Luz e Internet*
- *CA: Costo de licencia Android*
- *CI: Costo de licencia iOS*

Beneficio a un mes: $9000/(160 + 84.39 + 337.55) = 15.47$

Beneficio a un año: $9000 * 12/(160 * 12) + (84.39 + 337.55) = 46.12$

Beneficio a cinco años: $9000 * 60/(160 * 60) + (84.39 + 337.55 * 5) = 47.48$

Nota: 1 dólar equivalente a 3.38 soles

5.1.3 Aspectos comerciales

- Producto que utiliza buenas prácticas

- Eficiencia en el desarrollo de aplicativos móviles

- Posibilidad de crear y mejorar componentes a utilizar

- Interfaces amigables, normalizadas y personalizables

5.1.4 Recursos del proyecto

- Macbook (necesario para el desarrollo para iOS)

- Android Studio

- XCode

6 METODOLOGÍA A EMPLEAR

La metodología de trabajo elegida es Kanban por ser una metodología altamente efectiva y eficiente.

Kanban tiene como objetivo gestionar de manera general cómo se van contemplando las tareas y sus principios son:

- Calidad garantizada: La calidad tiene una mayor importancia que el tiempo, se trata de reducir los tiempos de rediseño.
- Reducción de desperdicio: Reducción de todo lo secundario y superficial para concentrarse en lo importante.
- Mejora continua: Según los objetivos el proyecto se va desarrollando.
- Flexibilidad: Se puede dar prioridad a tareas imprevistas, las tareas siguientes deben estar consideradas en el backlog.

Esta metodología provee un flujo de trabajo constante, con la flexibilidad de cambiar las prioridades en el momento, además da una visualización del avance del proyecto.

Anderson D. (2010), nos dice que las principales prácticas de Kanban son:

1. Visualizar tu trabajo
2. Limitar trabajo en curso
3. Hacer políticas explícitas
4. Manejar el flujo
5. Implementar ciclos de retroalimentación
6. Mejorar la colaboración

7 PLAN DE TRABAJO

TAREA	DURACIÓN	FECHA
Análisis y diseño		
Requerimientos	3 días	Lunes 26 agosto 2019
Diseño de la arquitectura	10 días	Jueves 29 septiembre 2019
Implementación		
Codificación iOS	15 días	Martes 10 septiembre 2019
Codificación Android	10 días	Miércoles 25 septiembre 2019
Pruebas		

	iOS: Pruebas en entorno virtual	2 días	Lunes 30 septiembre 2019
	Android: Pruebas en entorno virtual	2 días	Jueves 5 octubre 2019
	iOS: Pruebas en entorno real	1 día	Lunes 2 octubre 2019
	Android: Pruebas en entorno real	1 día	Viernes 6 octubre 2019
	Pruebas de integración (SOA)	2 días	Lunes 9 octubre 2019

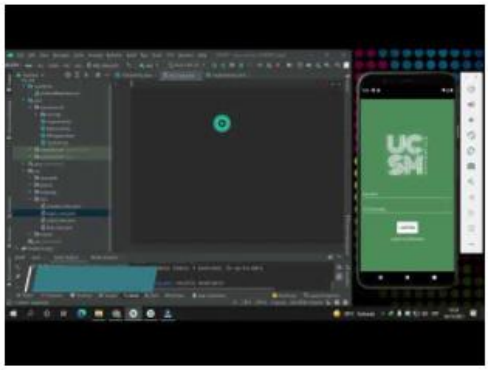
8 REFERENCIAS BIBLIOGRÁFICAS

- Holzinger, A. (2012). Making apps useable on multiple different mobile platforms: on interoperability for business application development on smartphones (p. 176-189).
- Münster, L. (2018). A process-oriented modeling approach for graphical development of mobile business apps.
- Ombretta, M. (2016). An empirical analysis of energy consumption of cross-platform frameworks for mobile development.
- Pinar Muyan (2017). A hands-on cross-platform mobile programming approach to teaching OOP concepts and design patterns
- Raj, T. (2012). A study on approaches to build cross-platform mobiles applications and criteria to select appropriate approach (p. 625-629).
- Ribeiro A, da Silva (2012). Survey on cross-platform and languages for mobile apps (p. 255-260).
- Smuty, P. (2012). Mobile development tools and cross-platform solutions (p. 653-656).
- Wafaa, S. (2015). Taxonomy of cross-platform mobile applications development approaches (p. 3-11).

APENDICE B

ENCUESTA SOBRE EL FRAMEWORK

Demo de funcionamiento



¿Cuál es su cargo en su puesto de trabajo?

Texto de respuesta corta

¿Cuántos años desarrolla aplicaciones (web, móvil, escritorio, etc.)? *

Texto de respuesta corta

¿Encuentra sencillo el uso del framework presentado en el video? *

- ○ Sí
- ○ No

Según la imagen, ¿le parece entendible la configuración para crear un ComboBox? *

- ◯ Sí
- ◯ No

Sabiendo que el framework pesa aproximadamente 1mb y puede ser utilizado en conjunto con otras herramientas, ¿lo utilizaría para agilizar sus tiempos de desarrollo? *

- ◯ Sí
- ◯ No

En el caso que haya marcado "No" a la pregunta anterior, indique el motivo

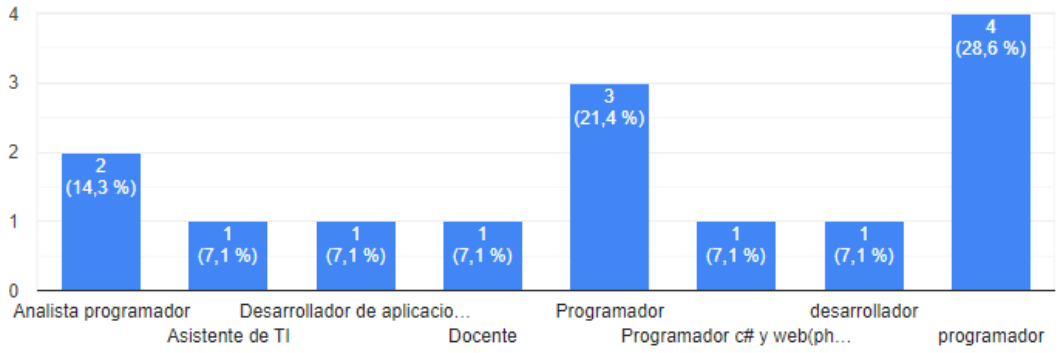
Texto de respuesta larga

APENDICE C

RESPUESTAS DE LA ENCUESTA SOBRE EL FRAMEWORK

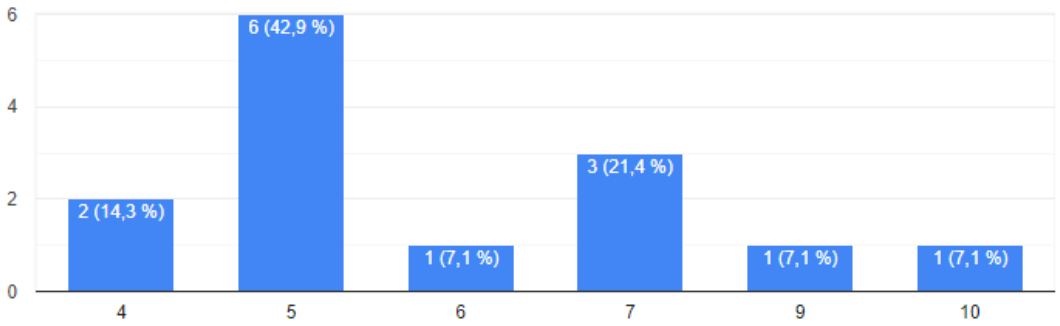
¿Cuál es su cargo en su puesto de trabajo?

14 respuestas



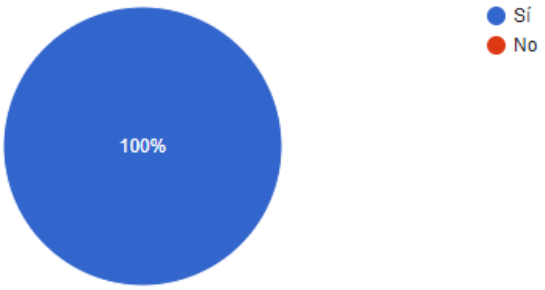
¿Cuántos años desarrolla aplicaciones (web, móvil, escritorio, etc.)?

14 respuestas



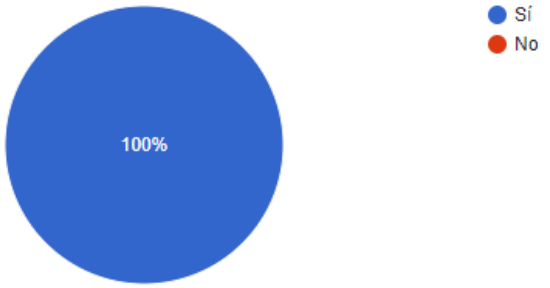
¿Encuentra sencillo el uso del framework presentado en el video?

14 respuestas



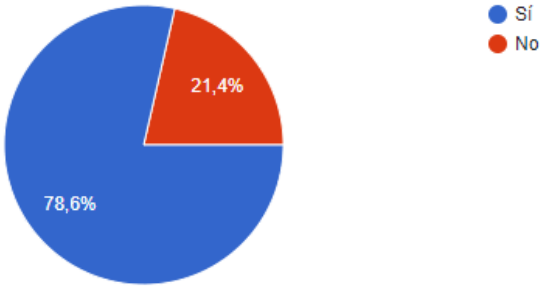
Según la imagen, ¿le parece entendible la configuración para crear un ComboBox?

14 respuestas



Sabiendo que el framework pesa aproximadamente 1mb y puede ser utilizado en conjunto con otras herramientas, ¿lo utilizaría para agilizar sus tiempos de desarrollo?

14 respuestas



En el caso que haya marcado "No" a la pregunta anterior, indique el motivo

4 respuestas

Si el framework es nuevo primero debe pasar a la fase de pruebas con varios desarrolladores, si una falla se genera y es culpa del framework estaremos limitados a esperar una solución el cual puede durar mucho tiempo en arreglarse.

En la explicación se muestra el funcionamiento y es muy buena pero después de su salida esperaría un buen tiempo para recién poder aplicarlo en un proyecto a futuro.

Lo mismo pasa con muchos frameworks no solo en Movil, tengo experiencia en web y por cambio de versión php muchas cosas cambiaron y quedaron obsoletas, muchas librerías de php 5.6, 6 y 7.2 ya no funcionan en 8.0.
Lo mismo puede pasar en Android con sus versiones (no tengo mucha experiencia en aplicaciones móviles, he utilizado kotlin para desarrollo de apps Android/iOS)

Por políticas en mi centro laboral, las librerías deben ser conocidas para todo el equipo

Ninguno

No sé cómo usarlo