ISSN: 1692-7257 - Volumen 1 – Número 37 - 2021

Revista Colombiana de Tecnologías de Avanzada

Recibido: 3 de agostode 2020
Aceptado: 3 octubre de 2020

# APLICACIÓN MOVIL ANDROID PARA LA GESTION DE MANTENIMIENTO CREADA CON CLEAN ARCHITE

# ANDROID MOBILE APPLICATION FOR MAINTENANCE MANAGEMENT CREATED WITH CLEAN ARCHITECTURE

**Est. Joan Sebastian Mosquera Capera, PhD Ferley Medina** [*,**]

**Universidad Surcolombiana, Facultad de ingeniería, ingeniería en software,**

**Avenida Pastrana Borrero - Carrera 1, Neiva, Huila, Colombia**

**Resumen:** Implementación de una aplicación móvil para la gestión de mantenimiento en un taller de la ciudad de Neiva desarrollada bajo la arquitectura clean Architecture, una de las pocas aplicaciones capaces de dar recomendación de motocicletas a sus usuarios.
La aplicación cuenta con un módulo para el cliente el cual tendrá el sistema de recomendación, será capaz de brindarle a sus consumidores información acerca de las mejores motos y las mejores marcas según los demás clientes, usara un sistema de recomendación basado en filtro colaborativo, un módulo para el mecánico el cual permitirá registrar los vehículos a los cuales le apliquen mantenimiento correctivo y/o preventivas; finalmente un módulo para el administrador de la empresa con capacidad de registrar los datos personales del cliente, los datos del vehículo que recibe, enviar notificación al usuario sobre el éxito del mantenimiento de la motocicleta, o solicitar algún repuesto necesitado.

**Palabras clave:** Aplicación android, mantenimiento, sistema de recomendación, filtro colaborativo.

**Abstract:** Implementation of a mobile application for maintenance management in a workshop in the city of Neiva developed under the clean code architecture, one of the few applications capable of giving motorcycle recommendations to its users.
The application has a module for the client which will have the recommendation system, will be able to provide its consumers with information about the best bikes and the best brands according to the other customers, will use a recommendation system based on collaborative filter, a module for the mechanic which will allow to register the vehicles to which corrective and / or preventive maintenance is applied; Finally, a module for the company administrator with the ability to record the personal data of the customer, the data of the vehicle he receives, send notification to the user about the success of motorcycle maintenance, or request a spare needed.
**Keywords:** Android application, maintenance, recommender system, collaborative filter.

## 1. INTRODUCCIÓN

La tecnología avanza a pasos agigantados, día a día en el ámbito cotidiano nos vemos rodeados de numerosos artefactos que facilitan la vida cada vez más. La manera en cómo un teléfono celular nos puede dar la flexibilidad de hacer muchas cosas en muy poco tiempo es muestra de ello. desde realizar video llamadas, tomar fotografías en calidad de alta definición, hasta pedir un servicio de transporte como lo hace Uber. El Humano es un ser ingenioso, en cada momento del día se le ocurren

maravillosas ideas, pero debido a la falta de entrenamiento de nuestro cerebro, lo olvidamos con la primera distracción, si no tomamos nota de ello. La industria de las aplicaciones móviles ha creado múltiples soluciones para resolver estos problemas y muchos otros más en espacios laborales y/o personales, en las empresas, las apps se convierten en grandes ayudas, pues gestionan productos y servicios que ofrecen, controlan inventarios, almacenan facturas, funcionan como medio de pago etc.

Actualmente en el mercado se encuentran un sinfín de aplicaciones las cuales realizan una serie de eventos como almacenar datos, notas, enviar notificaciones, entre otras, pero no hay una aplicación completa la cual permita realizar un registro de datos de motocicletas, guardar en la nube, enviar notificaciones a sus usuarios y/o recomendar que motocicleta es de mejor calidad resumido en una frase, un sistema gestión de mantenimientos en un taller de motos en la ciudad de Neiva.

En Neiva hay alrededor de 350 talleres de motos, Según Alirio Gaspar presidente de la junta de mecánicos de la ciudad. Alrededor del 80% funcionan sin tener un sistema de gestión de sus clientes y vehículos que reciben, este porcentaje guarda los registros en cuadernos y/o libretas, los cuales a lo largo del tiempo llegan a perderse debido al aumento de datos o factores externos como la humedad, insectos o degradación del papel. Cuando se presenta alguna garantía los mecánicos o administradores de los talleres buscan en esos registros tardándose hasta 1 hora en encontrarlos, para luego proceder a realizar el servicio.

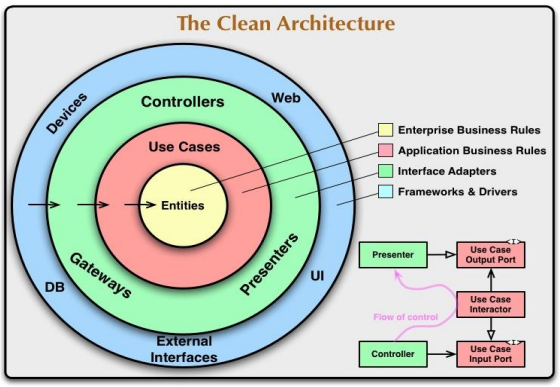



Fig. 1 CleanArchitecutre

Para resolver estos problemas se crea una aplicación que gestiona los mantenimientos que se le hacen a las motocicletas en un taller de la ciudad, esta aplicación está construida bajo la arquitectura "Clean Architecture", basada en la separación de capas, la cual considera la lógica del negocio como eje principal del sistema y define una serie de interfaces
las cuales se comunican con el resto del sistema, (Rodriguez, 2016) este modelo mejora la eficiencia, tiempos de respuesta y funcionalidad de la aplicacion.

los niveles de los cuales se compone esta arquitectura son:
UI: la cual es la interfaz de usuario tales como HTML, XML entre otros.
Presenters: Que son las clases que responden a las acciones generadas por la interfaz, y las cuales muestran la información al usuario.
Use cases: encargados de evaluar las reglas de negocio de la app. Y las Entities que son simplemente el modelo de datos y los webs services que utiliza el aplicativo (Barea, 2018).

El lenguaje de programación que se utiliza para desarrollar la aplicación es KOTLIN, después de una investigación sobre cuáles son los mejores lenguajes hoy en día para programar aplicaciones nativas android se llegó a la conclusión de:
-Kotlin es el segundo lenguaje más querido y el cuarto más demandado del mundo.
-Este es un lenguaje de programación estáticamente tipado lo que quiere decir que la maquina virtual (VM) es la que infiere el tipo de variable y por tanto no hay necesidad de especificarla algo que facilita el trabajo de los programadores.
-Está diseñado para interoperar con java sin tener problema, por ende, se puede tener módulos programados en java y otros en Kotlin.
-Es compatible con librerías ya existentes, ya que todas las que existen para java están migradas; además posee una comunidad muy grande de desarrolladores la cual cada día va creciendo más más. (Jurado, 2018)

-Recientemente, la plataforma digital TechBeacon lo incluyó en la lista de los 5 lenguajes emergentes con un futuro brillante. En 2017, en la conferencia anual Google I/O, Kotlin fue anunciado como lenguaje oficial para desarrollo Android. (Font, 2019)

como entorno de trabajo se utiliza el IDE Android studio, debido a que la aplicación es construida principalmente para sistemas operativos android y por que al observar las estadísticas mundiales de ventas, Android tiene una clara ventaja sobre iOS en términos de participación de mercado. StatCounter informa que, durante el período

comprendido entre enero de 2018 y enero de 2019, este sistema representó el 74,45 % del total del mercado, con iOS apenas un 22,85 %. (Casas, 2019)

El backend utiliza los servicios de Firebase, incluyendo base de datos en tiempo real y en nube, Cloud Messagin, para el envio de notificaciones y comunicación con el servidor-app

Este se adapta justo a las necesidades del proyecto por las siguientes razones.

- Limitar el almacenamiento de datos en el dispositivo mediante el almacenamiento de datos JSON en Firebase Realtime Database y de archivos en Firebase Storage
- Enviar notificaciones con Firebase Cloud Messaging
- Sincronización automática de datos en tiempo real en varios dispositivos
- Manejar con facilidad el caso sin conexión
- Autenticar usuarios a través de una variedad de proveedores de identidad
- Desarrollo rápido de un servicio de backend (Firebase, 2019)

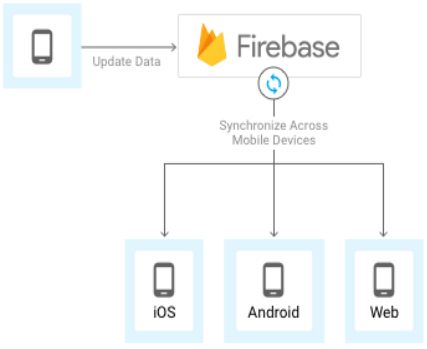



*Fig. 2 Patrón-de-diseño*

## SISTEMA DE RECOMENDACIÓN

Los sistemas de recomendaciones son herramientas que generan predicciones sobre un determinado objeto de estudio, a partir de las preferencias y opiniones dadas por los usuarios. Son muy útiles para evaluar y filtrar la gran cantidad de información disponible en la Web con objeto de asistir a los usuarios en sus procesos de búsqueda y recuperación de información. Entre las estrategias mas usadas para estos sistemas se encuentra el filtro basado en contenido el cual intenta predecir que querrá el usuario a partir de ítems visitados por este, y el filtro colaborativo utiliza la información de todas las demás personas de la base de datos de la app para poder identificar perfiles similares y aprender de los datos para recomendar productos de una manera individual: (Na8, 2019)

Se usa el framework TensorFlow para implementar el sistema de recomendación en la app, es un buen punto de partida para aplicaciones con machine learning, sus librerias son buenas, funciona bien tanto en clasificación de imágenes como en secuencias, ha sido votado como la biblioteca de Deep Learning más utilizada junto a Keras, fácil de utilizar y también cuenta con un API de alto nivel lo que lo hace rápido y eficiente. (BATHIA, 2018)

La forma en cómo se construye el sistema de recomendación es la siguiente: Primero se recopilan los datos, esto se hace mediante el modulo del usuario, el cual, a la hora de crear su perfil, selecciona la motocicleta que tiene junto a su marca, el usuario podrá darles una calificación a estas variables mencionadas. Estos datos se guardan en la base de datos para luego ser utilizadas por el sistema de recomendación. Entre mas usuarios haya en la aplicación, más preciso será.
La cantidad de datos determina qué tan buenas pueden ser las predicciones del modelo, Después de recopilar y almacenar los datos, se filtra para extraer toda la información requerida para realizar las sugerencias. Como siguiente paso se selecciona el algoritmo para recomendar, el cual es filtro colaborativo por ítem aquí se calcula la similitud entre cada par de elementos y con base a esto se puede calcular las los ítems más similares, esta predicción está dada por el siguiente modelo:
tomamos la suma ponderada de las calificaciones de "elementos-vecinos". Mediante la siguiente formula.

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,N}|)}$$

Fig. 3 EcuacionVecinos

Ahora se encuentra la similitud entre los elementos para poder dar la recomendación mediante la siguiente formula

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\| \vec{i} \|_2 * \| \vec{j} \|_2}$$

Fig. 4 similitudElementos

Una vez se aplica el modelo matemático y se obtiene las recomendaciones, el cliente visualiza los resultados. (JAYWRKR, 2019)

## METODOLOGIA

La aplicación está dividida en 3 Apps o módulos, con el fin de mejorar los tiempos de respuesta y reducir el espacio que necesita en la memoria del dispositivo. Un módulo para el usuario común y/o cliente la cual poseerá un inicio de sesión y registro, recibirá las notificaciones, solicitudes de repuestos enviadas por el administrador, y la cual contará con el sistema de recomendación, aparte de que el usuario podrá valorar la marca y la motocicleta que tiene.

La segunda aplicación será para el administrador el cual se encargará de gestionar los registros de las motocicletas que llegan al taller, enviar notificaciones a la app del usuario y solicitar permiso para incluir repuestos.

El tercer módulo, será para el mecánico, permitirá registrar los mantenimientos que realice a las motocicletas con el objetivo de brindar información al administrador sobre que trabajos hace durante la jornada laboral y así solucionara el problema a la hora de responder frente alguna garantía, sus archivos serán enviados a dos sectores de la base de datos, el primero, el más grande, es el que visualiza el admin, por ende, una vez enviado el dato, no podrá eliminarlo; solo el administrador tiene potestad de esto, y dos, un sector en el cual llega la copia de los datos previamente mencionados, en el cual el mecánico puede hacer con ellos lo que desee, son solo de uso privativo.

Para la primera versión de la aplicación se trabajará sobre el módulo del mecánico, se diseñará un inicio de sesión, un CRUD (créate, read, update, delete) el cual permite agregar, buscar, editar y visualizar los registros que se realicen durante el transcurso de la jornada, esta primera versión funcionará como MVP (mínimo producto viable) para el taller.

La segunda versión del aplicativo será el módulo del administrador, la cual será similar a la versión del mecánico, con la diferencia de que este posee acceso a los registros hechos por los mecánicos

La tercera versión es la del cliente, tendrá un inicio de sesión y un formulario para registrar nuevo usuario y la moto que tiene junto a su marca, la cual podrá valorar.

La cuarta versión implementara las notificaciones de admin a usuario.

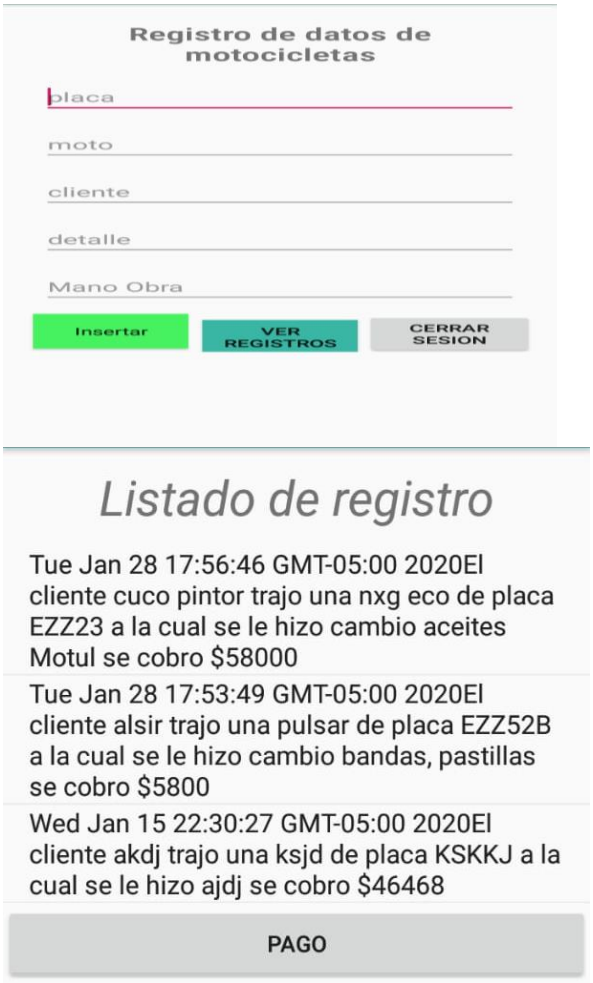
La quinta versión implementara las solicitudes para agregar un nuevo repuesto al mantenimiento. (Mensajería de emisor a receptor y viceversa)

Sexta versión, la creación e implementación del sistema de recomendación el cual se encargará de asesorar a los usuarios en cuanto a mejor marca y/o mejor moto, esta predicción es más precisa dependiendo de la cantidad de datos coleccionados de los demás usuarios registrados en la aplicación


Fig. 5 inicio Sesión

## RESULTADOS

Se crea el inicio de sesión para el administrador, el cual tiene campos que aplican validación de datos, cuando se inserta erróneamente el usuario o la contraseña arroja un mensaje de error




Se llenan todos los campos para luego ser almacenados en la nube

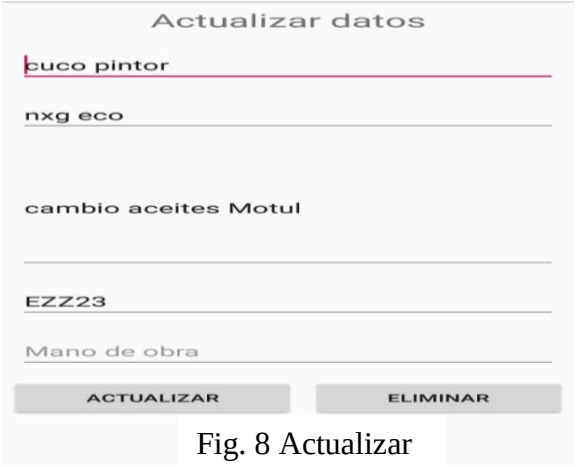



Fig. 8 Actualizar

tocando el botón Ver registros se pueden visualizar los datos almacenados en la nube seleccionando alguno de los datos que previamente se mostraron, se desplegara una interfaz la cual permite la modificación de los datos o su eliminación de la base de datos

## CONCLUSIONES

Con la implementación del aplicativo móvil en su versión MVP (mínimo producto viable) la empresa hacia la cual se orientó el proyecto tiene una mejor gestión de los mantenimientos realizados a las motocicletas, los mecánicos tienen un mejor control sobre sus registros, debido a que no tienen que escribir en cuadernos todos los trabajos realizados en la jornada laboral, ahora solo ingresan los datos en el módulo para luego ser enviado a la base de datos donde se almacenan, y cuando estos necesitan hacer alguna consulta , lo logran sin ningún problema

Se espera que con los módulos del administrador y del usuario la calidad y el prestigio en el taller mejore un 30% pues, los problemas de desordenes y demoras al consultar alguna hoja de vida disminuyen, los usuarios estarán mas pendientes de su motocicleta cuando no se encuentren con ella en el taller gracias a las notificaciones que produce la app. Por otra parte se espera un aumento de la clientela debido al plus que tendrá al ser uno de los pocos talleres de motos con aplicación móvil propia que es capaz de recomendar motocicletas y marcas

## REFERENCIAS

Barea, A. (2018). *Clean Architecture*. Obtenido de Deloitte: https://www2.deloitte.com/es/es/pages/technology/articles/clean-architecture.html#

BATHIA, R. (7 de agosto de 2018). *Tensor Flow Vs Caffe wich machine learning framework should you opt for?* Obtenido de https://analyticsindiamag.com/tensorflow-vs-caffe-which-machine-learning-framework-should-you-opt-for/

Casas, A. (25 de Febrero de 2019). *iPhone vs Android: cuota de mercado*. Obtenido de PCWorld: https://www.pcworld.es/articulos/smar

tphones/iphone-vs-android-cuota-de-mercado-3692825/

Firebase. (2019). *Firebase*. Obtenido de https://cloud.google.com/solutions/mobile/mobile-app-backend-services?hl=es-419#firebase-appengine-standard.

Font, J. (24 de Agosto de 2019). *¿Por qué usar Kotlin en Android? Beneficios, características y versiones*. Obtenido de Medium.com: https://medium.com/@javifont/por-qu%C3%A9-usar-kotlin-en-android-beneficios-caracter%C3%ADsticas-y-versiones-ea37dcfd3d7

JAYWRKR. (1 de enero de 2019). *Guía para construir un sistema de recomendación (Parte 1)*. Recuperado el 27 de 11 de 2019, de Medium: https://medium.com/@juancarlosjaramillo_88910/gu%C3%ADa-para-construir-un-sistema-de-recomendaci%C3%B3n-parte-1-2b1a65d6eac3

Jurado, F. L. (1 de Junio de 2018). *Qué es Kotlin y sus características*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-kotlin/

Na8. (27 de Agosto de 2019). *Sistemas de Recomendación*. Obtenido de https://www.aprendemachinelearning.com/sistemas-de-recomendacion/

Rodriguez, A. (7 de Diciembre de 2016). *El camino hacia una arquitectura software limpia- Hexagonal, Onion y Clean Architecture*. Recuperado el 4 de Noviembre de 2019, de http://aitorrm.github.io/t%C3%A9cnicas%20y%20metodolog%C3%ADas/arquitectura_software_limpia