Proceso de creación de Aplicaciones Híbridas y Nativas: evaluación comparativa a nivel diseño, desarrollo y rendimiento

ALUMNA: IARA NIZZA

CARRERA: INGENIERÍA EN INFORMÁTICA

MATRÍCULA: 502-11657

TUTOR: ING. ENRIQUE REINOSA

DIR. DE CARRERA: ING. SERGIO OMAR AGUILERA

CICLO: 2012

Agradecimientos

A mi familia, por su apoyo incondicional siempre, a mi mamá por todo su amor y por creer en mí incluso más que yo misma, a mi papá por enseñarme el valor del esfuerzo y compartir conmigo su conocimiento. A mis hermanas por ser mis ejemplos a seguir y demostrarme que con dedicación y resiliencia las metas se cumplen. Y a mi abuela, que estaría hoy por mi igual de orgullosa que siempre.

A mis compañeros, hoy amigos, por acompañarme y ayudarme a lo largo de toda la carrera, en los momentos difíciles y en las risas. Por enseñarme la importancia del trabajo en equipo, e impulsarme a dar lo mejor de mi.

A mis profesores, por compartir su sabiduría, formarme y prepararme para la vida profesional.

Y a mi tutor Enrique Reinosa, por su guía y por estar presente en todo este proceso.

Índice

Resumen

Teniendo en cuenta que a través de una estrategia nativa se desarrolla específicamente para una plataforma móvil, y que una estrategia híbrida permite desarrollar aplicaciones que funcionan en cualquier dispositivo y plataforma. Se buscarán los factores principales que deben considerarse para la toma de decisiones en cuanto a la implementación de una u otra opción.

Para determinar esto, se compararán y analizarán métricas específicas sobre dos prototipos funcionales de ambas tecnologías. Se obtendrán resultados y conclusiones respecto a límites y alcances de cada prototipo, tales como: tiempo de desarrollo, uso de recursos y tiempos de respuesta.

Los resultados obtenidos brindarán criterios técnicos y generales a tener en consideración para implementar una arquitectura que se ajuste a cada caso de desarrollo particular.

Introducción

Formulación del problema

En ocasiones los programadores, en su ámbito laboral, tienden a encontrarse ante la encrucijada de tener que seleccionar una tecnología a aplicar en el desarrollo de una aplicación móvil. Si se hace hincapié en que un porcentaje de ellos brinda servicios de manera independiente y de forma freelance, incluso quizá sin un propio grupo de trabajo ni asesorados por otros profesionales con roles como los de un Tech Lead o Arquitecto, puede volverse una tarea desafiante, al ser contratados por un cliente, elegir la tecnología adecuada para el desarrollo de un software de esta índole.

En aspectos generales se deberá tener en cuenta el presupuesto disponible, los tiempos pactados de entrega del producto final y la finalidad del software en sí.

Pero a un nivel más técnico es importante, a su vez, considerar los requisitos de configuración para el desarrollo de una aplicación nativa e híbrida, los conocimientos del desarrollador en cada tecnología y la curva de aprendizaje en ambos casos. También es importante analizar el rendimiento, así como el diseño visual, la experiencia de usuario, la exportación a distintas plataformas y el mantenimiento de ambas aplicaciones.

Justificación del trabajo

Del problema planteado surge la necesidad de sacar ciertas métricas que serán necesarias para analizar estos desarrollos de forma independiente y lograr realizar una comparativa de plataformas, en diversos aspectos y escenarios, sometiendo a prueba ambas aplicaciones.

Se hará foco en la curva de aprendizaje, así como en el tiempo de desarrollo de cada tecnología, seleccionando lenguajes de programación adecuados para la creación de estas aplicaciones.

La finalidad de esta tesina será aportar y orientar en la toma de decisiones a la hora de aplicar una arquitectura particular en el desarrollo de una aplicación móvil desde una perspectiva que hace foco principalmente en los programadores.

Objetivo general y específicos

Específico: Lograr determinar los criterios técnicos y generales que deberán tenerse en cuenta a la hora de implementar una arquitectura mobile que se ajuste al desarrollo de aplicaciones híbridas y aplicaciones nativas. Para poder así determinar en qué escenarios se aplica mejor la utilización de una u otra tecnología y orientar a los programadores en la toma de decisiones.

General: Evaluación de entornos de desarrollo, desarrollo y métricas específicas sobre dos prototipos funcionales de ambas tecnologías, documentación.

Elaboración de hipótesis

En esta tesina se buscará demostrar por qué la implementación de un desarrollo híbrido es más conveniente en la creación de aplicaciones móviles programadas por trabajadores independientes y freelancers, haciendo hincapié en los costos, la curva de aprendizaje, la escalabilidad, el mantenimiento y los tiempos de entrega.

Metodología

1) Investigación inicial

2) Selección de métricas a estudiar

3) Adquisición de conocimientos para desarrollo

4) Instalación de IDEs, frameworks, variables de entorno y librerías necesarias

5) Creación de prototipos (Mockups, wireframes y desarrollo)

6) Testeo de performance y pruebas a evaluar

7) Documentación

Organización del documento

Este trabajo final de carrera se encuentra organizado de la siguiente manera:

• Introducción: en esta sección se detallan el planteo y el contexto del problema; la idea directriz; las hipótesis; objetivos generales; objetivos específicos; metodología; justificación; limitaciones y alcance del trabajo;

• Marco Teórico: se describen los antecedentes y la base conceptual desde la cual da fundamento al presente trabajo final de carrera.

• Análisis del Problema: se describe el problema al cual se le busca dar solución y por el que se desprende la creación de ambos prototipos funcionales.

• Diseño del sistema: se formularán los modelos a desarrollar tanto híbrido como nativo. Su configuración y principales conceptos arquitectónicos.

• Desarrollo del prototipo: en está sección, se implementarán las métricas a las que serán sometidos ambos prototipos y se expondrán sus pruebas y resultados.

• Conclusiones: se analizan los resultados, se elaboran las conclusiones en base a las hipótesis planteadas y los resultados obtenidos.

• Bibliografía: se agrega la bibliografía consultada para el desarrollo teórico de la tesina y también la consultada para el desarrollo práctico de los prototipos funcionales.

• Anexo: se enseñaran los repositorios con el código fuente de ambas aplicaciones desarrolladas.

• Glosario: en esta sección se aclaran conceptos técnicos de algunas palabras y siglas.

Alcance del trabajo

Este trabajo consiste en el diseño y desarrollo de dos aplicaciones idénticas, una desarrollada en forma nativa y otra desarrollada en forma híbrida. Se realizará un análisis que busca determinar cuándo es recomendable desarrollar una aplicación de manera híbrida o nativa haciendo énfasis en la metodología de trabajo freelance. Con fines analíticos se implementó a su vez el uso de la cámara del celular para tomar fotografías y realizar el testeo de la interacción entre software-hardware para ambas aplicaciones.

Marco teórico

Antecedentes

Un framework es una herramienta de programación que permite desarrollar software proporcionando una estructura con componentes integrados que sirven de base para construir proyectos. Fueron creados para facilitar la tarea de desarrollo en un lenguaje de programación específico. Suelen incluir una interfaz de desarrollo de aplicaciones (API), librerías, herramientas y compiladores. [1]

Los frameworks de programación han jugado un papel fundamental en el desarrollo de aplicaciones y sistemas de software durante décadas. Y han evolucionado constantemente para adaptarse a las necesidades cambiantes de los desarrolladores y las tecnologías emergentes.

En la década de 1960 los lenguajes de programación y las herramientas de desarrollo eran limitados. No existían los frameworks tal como se conocen hoy en día. Durante la década de 1970 aparecieron los primeros lenguajes de programación estructurados, como C y Pascal. Aunque no había frameworks formales, se desarrollaron bibliotecas de funciones reutilizables que facilitaban el desarrollo de software.

Luego, en la década de 1980 con la popularización de la programación orientada a objetos, surgieron los primeros intentos de crear frameworks. Smalltalk, un lenguaje orientado a objetos, fue pionero en este campo con su entorno de desarrollo llamado "MVC" (Modelo-Vista-Controlador), que sentó las bases de muchos frameworks modernos.

Durante la década de 1990, la web comenzó a ganar popularidad y surgieron frameworks como ASP (Active Server Pages), JSP (JavaServer Pages) y PHP (Hypertext Preprocessor). Estos frameworks ayudaron a separar la lógica de presentación de la aplicación web y proporcionaron herramientas para interactuar con bases de datos y gestionar sesiones de usuario.

Para la década de 2000, con el auge de la web dinámica, surgieron algunos más avanzados y robustos. Por ejemplo, Ruby on Rails que se convirtió en un framework popular para el desarrollo rápido de aplicaciones web. Otros frameworks notables de esta época incluyen Django (Python) y Laravel (PHP). Estos introdujeron conceptos como el enrutamiento, el ORM (mapeo objeto-relacional) y la generación automática de código para agilizar el desarrollo.

Para la década de 2010 los frameworks de JavaScript ganaron popularidad a medida que el desarrollo de aplicaciones web en el lado del cliente se volvió más sofisticado. AngularJS (y su sucesor Angular), React y Vue.js se convirtieron en los principales frameworks para la creación de interfaces de usuario interactivas y reactivas. Actualmente siguieron evolucionando para adaptarse a las nuevas tendencias y necesidades. Se enfatizó la modularidad, la eficiencia y la escalabilidad. [2]

Conceptos fundamentales

A continuación se profundizará sobre las tecnologías utilizadas para el desarrollo de ambas aplicaciones, con la finalidad de contar con la información técnica implicada en esta investigación. Se dividirá este marco en tres partes, la primera se centrará en la elección del sistema de almacenamiento de datos utilizado por ambas plataformas, luego por un lado habrá una orientación hacia el aspecto del desarrollo nativo y por el otro se proveerán los conceptos utilizados para el desarrollo híbrido.

Para esta tesina la aplicación híbrida se ha desarrollado con Ionic, Angular y Capacitor en Visual Studio y la aplicación nativa con Java, en Android Studio. A su vez se utilizó Firestore como método de almacenamiento de datos para ambos casos.

Almacenamiento

Firestore es un servicio de base de datos en la nube ofrecido por Google como parte de su plataforma Firebase. Es una base de datos NoSQL (No estructurada) que se utiliza para almacenar y sincronizar datos en tiempo real entre diferentes dispositivos y plataformas.

Está diseñado para aplicaciones web y móviles, y proporciona una forma escalable y flexible de almacenar y recuperar datos. Utiliza un modelo de documentos y colecciones, donde los datos se almacenan en documentos JSON[1] y se agrupan en colecciones.

Mantiene sus datos sincronizados entre las aplicaciones de los clientes a través de escuchas en tiempo real y ofrece soporte sin conexión para dispositivos móviles y web con la finalidad de que se puedan crear aplicaciones receptivas que funcionen independientemente de la latencia de la red o la conectividad a Internet.

Es flexible a la hora de almacenar información de tal forma que admite estructuras de datos jerárquicas y los documentos pueden contener objetos anidados complejos además de subcolecciones.

Almacena en caché los datos que una aplicación usa activamente, por lo que la aplicación puede escribir, leer, escuchar y consultar datos incluso si el dispositivo está desconectado. Cuando el dispositivo vuelve a estar en línea, sincroniza todos los cambios locales con Cloud Firestore. [3]

[1] Acrónimo de JavaScript Object Notation, notación de objeto de JavaScript.

Desarrollo Híbrido

Las aplicaciones híbridas son aquellas que pueden emplearse en cualquier sistema operativo, así como equipo o marca. Esto se debe a que se componen de dos elementos, la tecnología de base web y el lenguaje nativo. El lenguaje nativo es uno de los componentes más importante en estas aplicaciones híbridas, ya que gracias a esto es posible que la app acceda ciertos sensores del sistema operativo como lo puede ser el reconocimiento por huella, la ubicación, la cámara y galería, entre otros componentes.

Las aplicaciones híbridas tienen una composición diferente de las aplicaciones nativas, más parecida a la construcción de las páginas web. La parte que corresponde al código común es la información que comparten entre los diferentes sistemas operativos. El lenguaje nativo es la parte final de estas aplicaciones, y dependen del sistema operativo donde se subirá la app. Usar el lenguaje del sistema operativo brinda una mejor experiencia de usuario y la integración correcta a las funcionalidades del hardware.

Ventajas del desarrollo Híbrido

- Facilidad de desarrollo: Las aplicaciones híbridas son más fáciles de desarrollar que las nativas porque no es necesario que el desarrollador conozca el lenguaje requerido por cada plataforma.
- Mantenimiento: Las aplicaciones híbridas ofrecen a los desarrolladores una forma de mantener su código en múltiples plataformas con el mínimo esfuerzo. Un único backend puede alimentar todas las aplicaciones, y las actualizaciones de este backend pueden enviarse rápidamente. Esto ayuda a los desarrolladores a reducir el tiempo de desarrollo, ya que no necesitan backends separados para cada plataforma que soportan.
- Rendimiento: Las aplicaciones híbridas suelen ser más ligeras que las nativas porque todas se ejecutan dentro de un mismo proceso y comparten espacio de memoria. Dependiendo de cómo se desarrolle, una aplicación híbrida también puede responder mejor que una aplicación desarrollada con tecnologías nativas.

- Compatibilidad-multiplataforma: Estas aplicaciones están diseñadas para funcionar en varias plataformas a la vez. Por lo tanto, no tienen las mismas limitaciones que las aplicaciones creadas específicamente para una plataforma.
- Consumo de tiempo: Desarrollar una aplicación híbrida puede llevar mucho menos tiempo que desarrollar una aplicación nativa. Esto es especialmente cierto si se trata de desarrollar una app para varias plataformas, donde el código puede compartirse en su mayoría.
- Rentabilidad: Las aplicaciones híbridas suelen ser más rentables que las nativas. Mediante el uso de tecnologías web híbridas, se pueden desarrollar aplicaciones híbridas por un costo menor de desarrollo que con una aplicación nativa.

Desventajas del desarrollo Híbrido

- Actualización: Las nuevas funciones tardan más en aparecer. Los marcos de desarrollo de aplicaciones híbridas son muy flexibles y útiles, pero no siempre se actualizan en cuanto se lanza una función determinada en una plataforma específica. Los desarrolladores suelen tardar un tiempo en implementar dichas funciones en el framework.
- Rendimiento y Tamaño: suelen tener un menor rendimiento y un mayor tamaño que las aplicaciones nativas. Aunque la mayoría de estos frameworks suelen ser rápidos al final del día, una aplicación nativa casi siempre será más eficiente. Los frameworks híbridos introducen inevitablemente sobrecarga, ya que utilizan bibliotecas y otros fragmentos de código para llamar al código nativo, mientras que una aplicación nativa puede hacerlo directamente. Estas bibliotecas añadidas también introducen otra desventaja en forma de aumento del tamaño de la aplicación cuando se instalan.
- Bugs: Al depender de una herramienta de desarrollo de plataforma híbrida como Capacitor o Cordova para implementar el puente a las capacidades del sistema operativo surge el inconveniente de tener que esperar a que el desarrollador externo del creador de aplicaciones híbridas lo incorpore. El desarrollo de aplicaciones híbridas requiere el uso de frameworks como Capacitor o Cordova. Cada vez que se lanza una nueva función del kit de interfaz de usuario, hay que esperar a que la herramienta se ponga al día. Cuando se crea una aplicación híbrida, no se tiene tanto control sobre la capa adicional, lo que

aumenta la posibilidad de que se produzcan errores. Los errores son un problema importante cuando se desarrollan aplicaciones híbridas con las características más recientes disponibles para un sistema operativo específico. [4]

Ionic framework

Ionic Framework es un SDK[2] de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Permite crear aplicaciones móviles de alta calidad y rendimiento con integraciones para frameworks populares como Angular, React y Vue. Ofrece prácticas recomendadas como transiciones aceleradas por hardware y gestos táctiles optimizados.

Está diseñado para funcionar y mostrarse perfectamente en todos los dispositivos móviles y plataformas actuales. Con componentes listos para usar, tipografía y un tema base (ampliable) que se adapta a cada plataforma.

Ionic emula las directrices de la interfaz de usuario de las aplicaciones nativas y utiliza los SDK nativos, uniendo los estándares de la interfaz de usuario y las características de los dispositivos de las aplicaciones nativas con toda la potencia y flexibilidad de la web abierta. Utiliza Capacitor (o Cordova) para desplegarse de forma nativa, o se ejecuta en el navegador como una PWA[3].

Permite crear e implementar aplicaciones que funcionarán en varias plataformas, como iOS nativo, Android y la web como PWA, logrando esto con una única base de código. De esta manera se puede escribir el código una sola vez y ejecutarlo en cualquier lugar. [5]

Angular framework

Angular es una plataforma de desarrollo, construida sobre TypeScript. Incluye un framework basado en componentes para construir aplicaciones web escalables de una sola página. A su vez provee una colección de librerías integradas que cubren una amplia variedad de características, incluyendo enrutamiento, gestión de formularios, comunicación cliente-servidor, entre otras cosas. [6]

[2] Software Development Kit
[3] Progressive Web App

Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Capacitor framework

Capacitor es un proyecto de código abierto que ejecuta Web Apps modernas de forma nativa en iOS, Android y Web (utilizando la tecnología Progressive Web App) al tiempo que proporciona una interfaz potente y fácil de usar para acceder a SDKs nativos y APIs nativas en cada plataforma.[7]

Puede ser útil pensar en Capacitor como un nuevo y potente navegador para aplicaciones web modernas que desbloquea toda la funcionalidad nativa de cada plataforma a través de APIs multiplataforma coherentes. Utilizando Capacitor, los desarrolladores pueden crear una aplicación y dirigirse a un conjunto de APIs independientemente de la plataforma en la que se ejecute la aplicación, en lugar de gestionar múltiples APIs para cada plataforma de destino. Esto significa que, por ejemplo, para acceder a la cámara utiliza el mismo código en iOS/Android que en la web. Facilitando así la creación de una aplicación web que se ejecuta de forma nativa en dispositivos móviles, ordenadores de sobremesa y la web como una aplicación web progresiva.

Las aplicaciones de Capacitor son aplicaciones nativas. Pueden incorporar controles de interfaz de usuario nativos y acceder a cualquier SDK nativo o API disponible en la plataforma. Pero a diferencia de las aplicaciones nativas más tradicionales, las aplicaciones de Capacitor probablemente tendrán la mayor parte de la aplicación que se ejecuta en un control WebView incrustado que desbloquea los beneficios y eficiencias multiplataforma deseados.

Visual Studio

Es un entorno de desarrollo integrado (IDE) de Microsoft. Se utiliza para desarrollar programas informáticos, incluidos sitios web, aplicaciones web, servicios web y aplicaciones móviles. Visual Studio utiliza plataformas de desarrollo de software de Microsoft.

Desarrollo Nativo

El desarrollo nativo de aplicaciones significa crear una aplicación móvil adaptada y dedicada a una plataforma específica como iOS o Android.

Dado que las aplicaciones nativas se crean específicamente para el sistema operativo, ofrecen un mayor compromiso del usuario que las aplicaciones híbridas. Las aplicaciones móviles nativas suelen tener mejor rendimiento y aspecto que sus homólogas basadas en web, que deben servir para numerosas plataformas. Además, las aplicaciones móviles nativas tienen acceso a hardware y capacidades del dispositivo, como sensores y cámaras, que no están disponibles únicamente a través de una interfaz de navegador móvil.[8]

Las aplicaciones móviles nativas, a diferencia de los sitios web y las aplicaciones web, no funcionan en el navegador. Se deben descargar de tiendas de aplicaciones específicas de cada plataforma, como App Store de Apple y Google Play. Se puede acceder a cada programa tocando su icono en la pantalla del dispositivo tras la instalación.

A su vez son más complicadas de crear que los sitios web para móviles. No hay que preocuparse por la compatibilidad o el comportamiento de los navegadores. Ya que se pueden utilizar las capacidades nativas de los sistemas operativos móviles para crear una mejor experiencia de usuario.

Estas aplicaciones suelen ofrecer una experiencia de usuario excepcional, ya que suelen ser de alto rendimiento. Como los efectos visuales se adaptan a la UX de la plataforma, la experiencia de usuario también mejora.

Ventajas del desarrollo Nativo

- Performance: La aplicación se construye y optimiza para una plataforma concreta cuando se desarrolla. Como consecuencia, la aplicación ofrece un rendimiento significativo. Las aplicaciones nativas son rápidas y receptivas, ya que están diseñadas para esa plataforma y compiladas con su lenguaje de programación y API principales. Como resultado, la aplicación es considerablemente más eficiente. El material y los aspectos visuales de una

aplicación móvil nativa ya están almacenados en los teléfonos de los usuarios, lo que se traduce en tiempos de carga rápidos.

- Seguridad: Las aplicaciones web se realizan con varios navegadores y tecnologías como JavaScript, HTML5 y CSS. Desarrollar una app móvil nativa es un método excelente para garantizar a los usuarios la seguridad constante de sus datos.
- UX: Las aplicaciones móviles nativas responden mucho mejor a las entradas y salidas del usuario. Estas apps tienen en cuenta el entorno del sistema operativo de sus dispositivos, lo que las hace parecer parte integrante del mismo. Están diseñadas específicamente para un sistema operativo concreto. Siguen normas estrictas que, en última instancia, mejoran y sincronizan la experiencia del usuario con el sistema operativo específico. Como resultado, el flujo de la app es más natural, ya que cada plataforma tiene sus criterios de interfaz de usuario. Los usuarios pueden interactuar con las apps utilizando acciones y gestos con los que ya están familiarizados si siguen unos estándares concretos.
- Accesibilidad: Los desarrolladores pueden acceder a todas las funciones de los dispositivos gracias a las aplicaciones nativas ya que logran aprovechar las capacidades del dispositivo. Estas aplicaciones pueden acceder inmediatamente al hardware del dispositivo, como el GPS, la cámara y el micrófono, lo que las hace más rápidas en su ejecución. Las notificaciones push son otra ventaja significativa de elegir el desarrollo de aplicaciones nativas. Permiten el acceso a todas las API y herramientas que ofrezca la plataforma para la que se desarrollen.
- "Bug-free": Al no depender de tecnologías híbridas como Capacitor o Cordova, el desarrollo de aplicaciones nativas tiene menos dependencias para la aparición de errores. Las aplicaciones híbridas utilizan un puente que puede ralentizar el desarrollo y dar lugar a una mala experiencia de usuario. Cuando se lanzan nuevas versiones de Android e iOS, este problema se hace más evidente. Los desarrolladores de aplicaciones nativas acceden a kits de desarrollo de software (SDK) recientes para desarrollar aplicaciones con las funciones más actualizadas. Los usuarios de aplicaciones nativas se benefician de los avances de la plataforma tras actualizar el sistema operativo gracias a este desfase temporal.

- Escalabilidad: La flexibilidad en la gestión de recursos y la variedad de herramientas accesibles también hacen que las aplicaciones diseñadas para el entorno nativo sean más escalables.

Desventajas del desarrollo Nativo

- Costoso: Puede resultar costoso lanzarlas tanto para iOS como para Android. Ya que habría que contratar a dos equipos distintos para trabajar en plataformas diferentes.
- Consumo de tiempo: Dado que el desarrollo específico para una plataforma requiere mucho tiempo, reutilizar el mismo trabajo para otra plataforma sería difícil. En su lugar, se necesitaría un equipo separado para desarrollar la versión alternativa.
- Actualizaciones: Los desarrolladores suelen introducir nuevas versiones en las aplicaciones nativas por diversos motivos. La causa más común es reparar errores y fallos de funcionamiento. En consecuencia, es necesario descargar las actualizaciones de la tienda de aplicaciones y puede que el usuario no esté al tanto de ellas o se las salte para ahorrar espacio de almacenamiento.

Java

Es un lenguaje de programación multiplataforma utilizado para crear software que lo hace compatible con muchos entornos operativos diferentes. Fue uno de los primeros lenguajes de programación orientados a objetos. Es un lenguaje de programación de dos etapas, lo que significa que es un lenguaje que genera un código objeto y luego se requiere de un intérprete para la ejecución de dicho código objeto, lo que lo diferencia de la mayoría de los demás lenguajes compilados porque no genera directamente en un archivo ejecutable. durante el proceso de compilación.

El código se escribe primero en un kit de desarrollo Java, disponible para Windows, Linux y macOS, los programadores escriben en el lenguaje de programación Java, que el kit traduce a código informático que puede ser leído por cualquier dispositivo con el software adecuado. Esto se consigue con un programa llamado compilador. Un compilador toma código informático de alto nivel como Java y lo traduce a un lenguaje que los sistemas operativos entienden, llamado bytecode.

A continuación, el bytecode es procesado por un intérprete llamado máquina virtual Java (JVM). Las JVM están disponibles para la mayoría de plataformas de software y hardware, y es lo que permite transferir el código Java de un dispositivo a otro. Para ejecutar Java, las JVM cargan el código, lo verifican y proporcionan un entorno de ejecución.

Java es un lenguaje de programación extremadamente transferible que se utiliza en diferentes plataformas y tipos de dispositivos, desde teléfonos inteligentes hasta televisores inteligentes. Se utiliza para crear aplicaciones móviles y web, software empresarial, dispositivos de Internet de las Cosas (IoT), juegos, big data, aplicaciones distribuidas y basadas en la nube. [9]

Android Studio

Es un entorno de desarrollo potente y sofisticado, diseñado con el propósito específico de desarrollar, probar y empaquetar aplicaciones Android. Es el IDE oficial para Android Studio. Es una colección de herramientas y componentes. Muchas de estas herramientas se instalan y actualizan independientemente unas de otras. Posee numerosos plugins de terceros que

proporcionan una gran variedad de funciones valiosas, no disponibles directamente a través del IDE. [10]

No es la única forma de desarrollar aplicaciones Android, existen otros IDE, como Eclipse y NetBeans.

Hay muchas maneras en las que Android Studio difiere de otros IDEs y herramientas de desarrollo. Algunas de estas diferencias son bastante sutiles, como la forma en que se instalan las bibliotecas de soporte, y otras, por ejemplo, el proceso de construcción y el diseño de la interfaz de usuario, son profundamente diferentes.

Entre sus diferencias más importantes se encuentran:

- Desarrollo de la interfaz de usuario: La diferencia más significativa entre Studio y otros IDEs es su editor de diseño, que ofrece vistas de texto, diseño y planos y, lo que es más importante, herramientas de diseño de restricciones para cada actividad o fragmento, un tema fácil de usar y editores de estilo, y una función de diseño de drag-and-drop.
- Refactorización: La refactorización también es más fácil y de mayor alcance que otros IDE. Se puede renombrar casi todo, desde variables locales hasta paquetes enteros.
- Emulación: Studio viene equipado con un flexible editor de dispositivos virtuales, que permite a los desarrolladores crear emuladores de dispositivos para modelar cualquier número de dispositivos del mundo real. Estos emuladores son altamente personalizables, tanto en términos de factor de forma como de configuraciones de hardware, y pueden descargarse dispositivos virtuales de muchos fabricantes.

Análisis del problema

Situación actual

Con el fin de facilitar el proceso de adopción de animales muchas veces abandonados, maltratados o perdidos, se desarrolló el prototipo de una aplicación que busca dar respuesta a las necesidades de sus usuarios. Wizapet fomenta una comunidad que une a los humanos en amor y servicio por los animales.

Según una encuesta nacional de mascotas de Millward Brown Argentina el 76% de los perros que llegan a un hogar fueron adoptados.

La aplicación también ofrece una sección para mascotas que se encuentran en estado de tránsito, muchas veces rescatados por personas que no tienen la capacidad de albergarlos en sus hogares. Por otro lado pueden encontrarse animales perdidos y publicar animales que se extraviaron, todo desde una misma app.

Elementos del problema

Con la aplicación Wizapet, se dispondrá de un entorno dedicado exclusivamente a las mascotas. Su principal finalidad es, a través de publicaciones, exhibir la situación del animal junto a una fotografía y otras secciones como comentarios, su raza y datos de contacto de la persona.

Esto es lo que ofrecerá el prototipo inicial, con posibilidad para sumar nuevas funcionalidades tales como geolocalización, chat y búsqueda avanzada.

Diseño del sistema

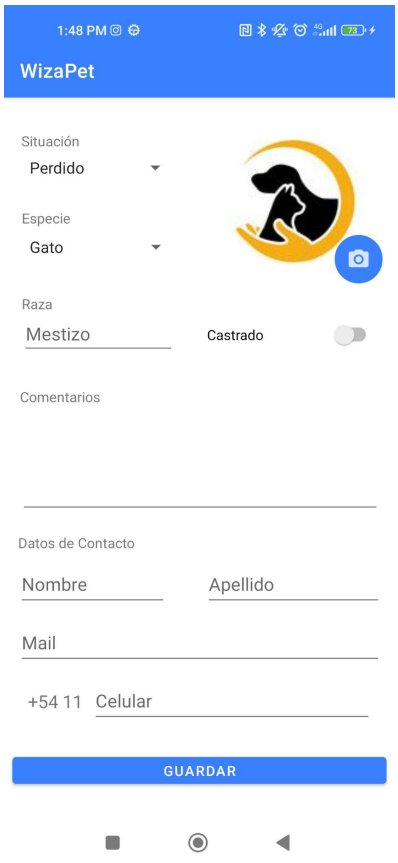
En esta sección se ahondará en los prototipos creados, analizando funcionalidades, diseño, uso de IDEs, código, librerías y base de datos.

Formulación de los modelos

El sistema a implementar en este trabajo es un prototipo de aplicación para celulares. Se desarrollaron dos versiones, una en Ionic, Angular y Capacitor (Desarrollo híbrido) y otra con Android Studio y Java (Desarrollo nativo). Ambas aplicaciones utilizan como base de datos Firebase y el diseño de la interfaz busca mantener una estética y estructura muy similar.

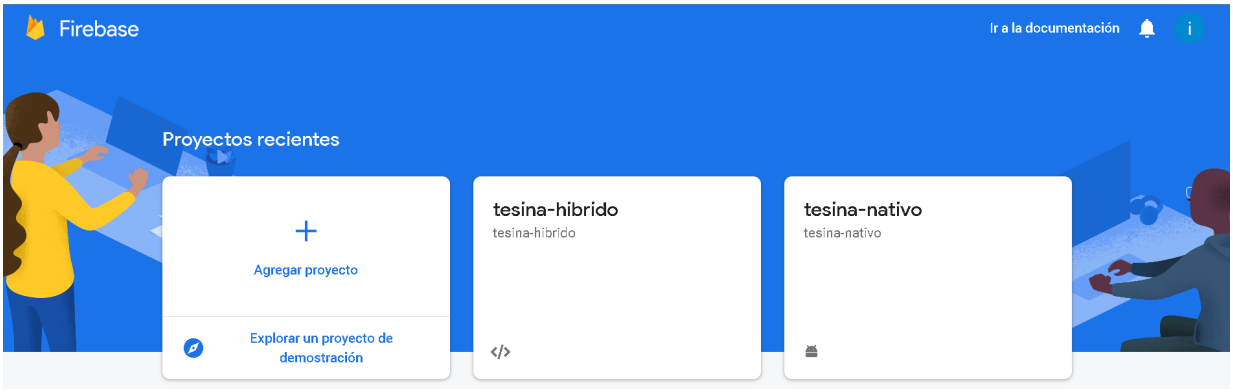
Versión Híbrida

Versión Nativa



Firebase

Es una plataforma de desarrollo que te ayuda a compilar y desarrollar aplicaciones. Desde la web https://console.firebase.google.com/ se agregaran los proyectos con sus respectivas tecnologías y se creará en cada proyecto una base de datos de Firestore que se configurarán más adelante. Un proyecto de Firebase es como un contenedor para todas sus aplicaciones y cualquier recurso y servicio aprovisionado para el proyecto.



Vista del panel de control de Firebase

La instalación de Firebase localmente se hace mediante npm:

```
npm install firebase
```

Desarrollo de prototipo híbrido

Instalación

Las aplicaciones Ionic se crean y desarrollan principalmente a través de la utilidad de línea de comandos de Ionic (la "CLI"), y utilizan Capacitor para compilar/desplegar como una aplicación nativa. La mayoría de las herramientas de la CLI se basan en Node y se gestionan a través de npm. La forma más rápida de obtener Node y NPM instalado en una máquina es a través del instalador de NodeJS (https://nodejs.org/en/).

Con Node y NPM configurados, se instalará Ionic y Capacitor CLI:

```
npm install -g @ionic/cli
npm install @capacitor/core @capacitor/cli
```

Con @ionic/angular se logra combinar la experiencia básica de Ionic con las herramientas y APIs adaptadas a los desarrolladores de Angular. Mediante el siguiente comando se instalará Capacitor en el proyecto Ionic, se asignará un nombre a la aplicación, una estructura por defecto (tabs) y se seleccionara Angular Framework para ser utilizado por la app:

```
ionic start Wizapet tabs --type=angular --capacitor
cd Wizapet
```

Luego hay que inicializar Capacitor con la información de la aplicación. Este comando pedirá que se introduzca el nombre de la aplicación [Wizapet] y su identificador (el nombre del paquete para Android [com.wizapet.app]):

```
npx cap init
```

Hay que construir el proyecto Ionic al menos una vez antes de añadir cualquier plataforma nativa:

```
ionic build
```

Luego se instalarán las plataformas nativas deseadas, Se crea la carpeta android en la raíz del proyecto. Estos son artefactos del proyecto nativo completamente separados que deben ser considerados parte de la aplicación Ionic (es decir, se edita en su =propio IDE):

```
npx cap add android
```

Para los proyectos nativos, en este caso en Android, hay que usar las IDEs para poder ejecutar y desplegar la aplicación, el siguiente comando abre Android Studio para poder buildear la app:

```
npx cap open android
```

Cada vez que se realice una compilación que cambie el directorio web (por defecto la carpeta /www), se necesitará copiar esos cambios al proyectos nativos, sincronizandolo la aplicación con Capacitor:

```
npx cap copy
```

Con las configuraciones previas efectuadas ya se podrá iniciar un servidor de desarrollo local para desarrollar y probar la aplicación:

```
ionic build
```

Base de datos

Desde el código se inicializa Firebase en la aplicación, con el fin de utilizar la API Cloud Firestore, se crea un objeto de aplicación de Firebase (un objeto similar a un contenedor que almacena una configuración común y comparte la autenticación entre los servicios de Firebase) y se agregaran los servicios necesarios. Los servicios como Cloud Firestore están disponibles para importar dentro de subpaquetes individuales.

```
// Import the functions you need from
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
import { getFirestore } from "firebase/firestore";
import { collection, addDoc, getDocs } from "firebase/firestore";

const firebaseConfig = {
  apiKey: "AIzaSyAfjTL5pun-50H3KVGIryYVqnCD2jsA3Eg",
  authDomain: "tesina-hibrido.firebaseapp.com",
  projectId: "tesina-hibrido",
  storageBucket: "tesina-hibrido.appspot.com",
  messagingSenderId: "351697777940",
  appId: "1:351697777940:web:ea9eaf190024168c1efe01",
  measurementId: "G-0797821TNE"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

// Initialize Cloud Firestore and get a reference to the service
const db = getFirestore(app);
```

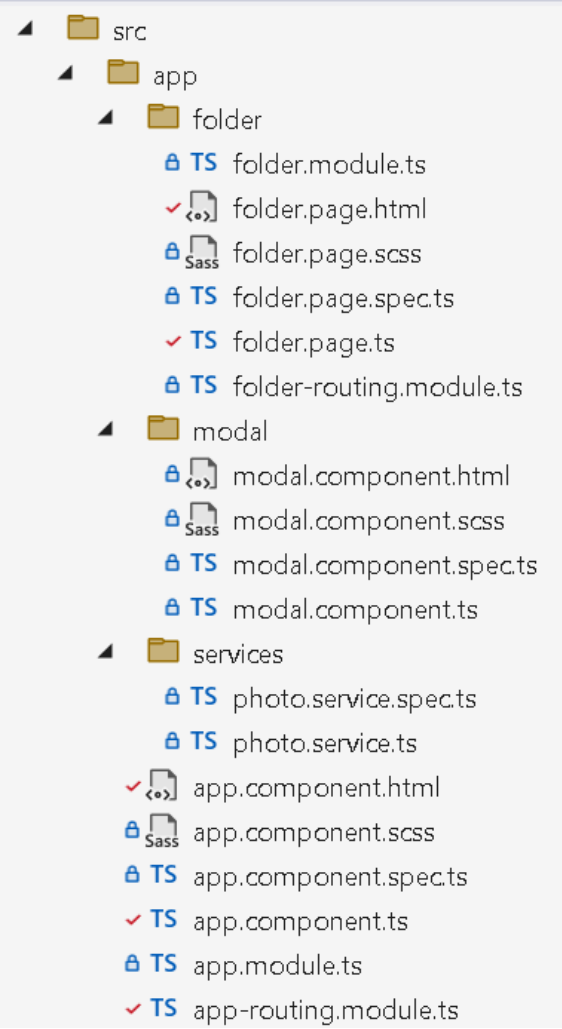
Vista del archivo src/app/folder/folder.page.ts

Estructura del proyecto

Utilizando la CLI de Angular se pueden crear automáticamente características del framework, como páginas, componentes, directivas y servicios, con Ionic Generate:

```
ionic generate [type] [name]
```

- Type: El tipo de generador (componente, directiva, página, tubería, proveedor, fichas).
- Name: El nombre del componente que se está generando



folder

Contendrá la vista principal con el formulario de carga de mascotas.

modal

Mostrará la vista de la publicación una vez creada dentro de un componente de ventana modal.

services

Toda la lógica de Capacitor (características nativas) se encapsuló en una clase de servicio donde se importan las dependencias de Capacitor y se obtienen referencias a los plugins Camera, File System y Storage.

src

La carpeta src contiene el código sin compilar.

app

Tiene todos los archivos necesarios para arrancar la aplicación y la estructura de la aplicación.

La carpeta /app Tiene 4 archivos:

- *app.component.ts* - define la estructura básica y la navegación inicial de la aplicación. Por ejemplo, si se quiere desarrollar un menú lateral este es el lugar para definir la primera página de ese menú.
- *app.component.html* - se define la navegación, es la vista raíz de la SPA (Single Page Application).
- *app.module.ts* - este es el punto de entrada de la aplicación. En esta página se definirán todos los módulos, páginas y componentes que se utilizan dentro de la app.
- *app.scss* - este es el principal punto de entrada para los archivos sass. Puede aplicarse el diseño global de la aplicación.
- *app-routing.module.ts* - este es un archivo que contendrá el ruteo de las vistas de la aplicación.

Páginas

La carpeta folder es el lugar donde se almacenan todas las páginas creadas de la aplicación. Cada página tendrá una carpeta separada y dentro de esa carpeta se incluirán los archivos .html, .module.ts, .scss y .ts.

Servicios

La carpeta Services contiene todos los servicios que se utilizarán para acceder a los datos que se presentarán en la aplicación.

Componentes

Los componentes son el principal bloque de construcción de las aplicaciones Angular/Ionic, permiten construir rápidamente una interfaz para la aplicación. Ionic viene con una serie de componentes, incluyendo modales, ventanas emergentes y tarjetas. Cada componente controla

una parte en pantalla llamada vista. Es fundamental hacerlos lo más genéricos posibles para que se puedan reutilizar.

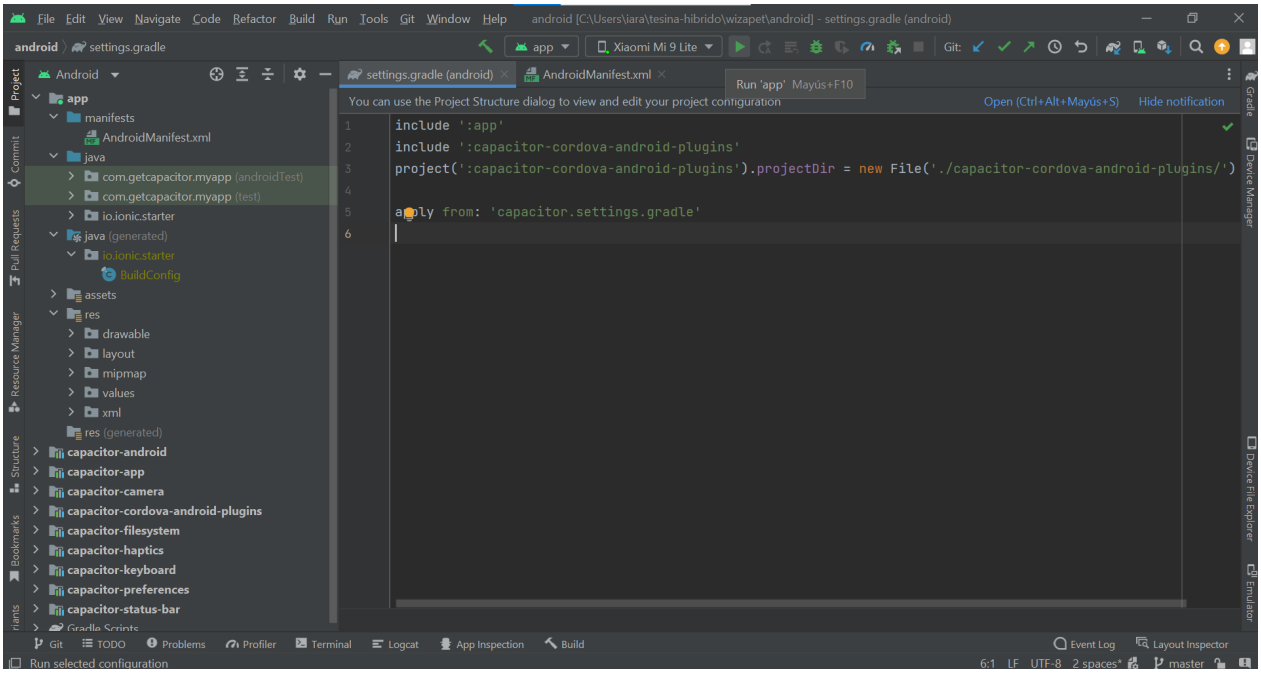
WWW

La estructura de la aplicación tendrá a la altura de la carpeta src, una carpeta www que contiene el código ya compilado, cada vez que se buildea el proyecto el contenido del www es borrado y sustituido de nuevo, así que no hay necesidad de cambiar nada en esta carpeta. Si se desea desplegar la aplicación en versión web se utilizará el código que está dentro de la carpeta www.

Ejecución de la aplicación

Desde una consola con el comando `ionic serve` se puede visualizar la versión web de la app que localmente se abrirá automáticamente desde un Browser en la url http://localhost:8100.

Para verlo en un emulador de Android o en un dispositivo físico, a través de la consola con el comando `ionic capacitor build android` se abrirá Android Studio y ya desde ahí se podrá correr la aplicación para verla en versión móvil.


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
android [C:\Users\iara\tesina-hibrido\wizapet\android] - settings.gradle (android)
android settings.gradle
app
Xiaomi Mi 9 Lite
Git:
Android
settings.gradle (android)
AndroidManifest.xml
Run 'app' Mayús+F10
You can use the Project Structure dialog to view and edit your project configuration
Open (Ctrl+Alt+Mayús+S)
Hide notification
include ':app'
include ':capacitor-cordova-android-plugins'
project(':capacitor-cordova-android-plugins').projectDir = new File('./capacitor-cordova-android-plugins/')
apply from: 'capacitor.settings.gradle'
app
manifests
AndroidManifest.xml
java
com.getcapacitor.myapp (androidTest)
com.getcapacitor.myapp (test)
io.ionic.starter
java (generated)
io.ionic.starter
BuildConfig
assets
res
drawable
layout
mipmap
values
xml
res (generated)
capacitor-android
capacitor-app
capacitor-camera
capacitor-cordova-android-plugins
capacitor-filesystem
capacitor-haptics
capacitor-keyboard
capacitor-preferences
capacitor-status-bar
Gradle Scripts
Git
TODO
Problems
Profiler
Terminal
Logcat
App Inspection
Build
Event Log
Layout Inspector
Run selected configuration
6:1 LF UTF-8 2 spaces* master

Desarrollo de prototipo nativo

Instalación

Para instalar Android Studio en Windows se siguieron estos pasos:

- Descomprimir el .zip
- Copia la carpeta android-studio en la carpeta Archivos de programa.
- Abrir la carpeta android-studio > bin.
- Iniciar studio64.exe (para máquinas de 64 bits) o studio.exe (para máquinas de 32 bits).
- Seguir los pasos del Setup Wizard en Android Studio y, luego, instalar los paquetes de SDK recomendados. Cuando haya nuevas herramientas y otras APIs disponibles, Android Studio lo informará con una ventana emergente.

Configuración de buildeo

El sistema de compilación de Android compila los recursos y el código fuente de las aplicaciones y los empaqueta en APK o paquetes de aplicaciones de Android que se pueden probar, implementar, firmar y distribuir.

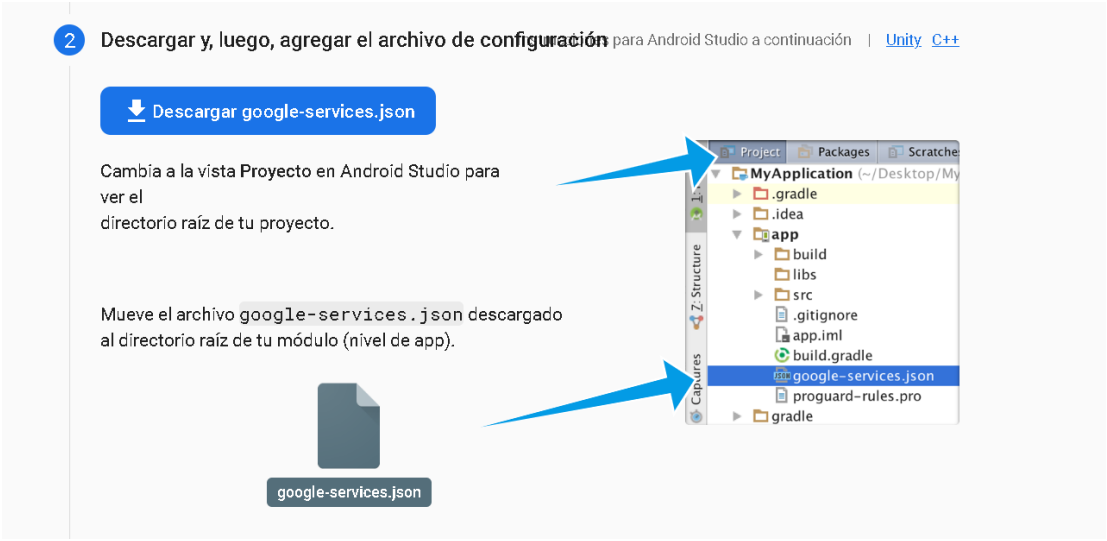
Gradle y el plugin de Gradle para Android funcionan independientemente de Android Studio. Esto significa que se pueden compilar las aplicaciones de Android desde Android Studio, desde la línea de comandos de un equipo o en equipos en los que no esté instalado Android Studio, como servidores de integración continua.

El proceso de compilación implica muchas herramientas y procesos que convierten un proyecto en un paquete de aplicaciones Android (APK). El plugin Android Gradle realiza gran parte de ese proceso de compilación.

Base de Datos

Desde Firebase al crear el proyecto se registra la aplicación con el nombre del paquete de Android:

Se importa el archivo google-services.json en el directorio app del proyecto:



Luego hay que agregar el SDK de Firebase:

Para que los SDK de Firebase puedan acceder a los valores de configuración de google-services.json, se necesita el complemento Gradle de los servicios de Google.

Agregar el complemento como una dependencia buildscript al archivo build.gradle de nivel de proyecto:

Archivo de Gradle de nivel de raíz (nivel de proyecto) (<project>/build.gradle):

```
buildscript {
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
  dependencies {
    ...
    // Add the dependency for the Google services Gradle plugin
    classpath 'com.google.gms:google-services:4.3.15'
  }
}

allprojects {
  ...
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
}
```

En el archivo build.gradle del módulo (nivel de app) se agregan los complementos google-services y cualquier SDK de Firebase que se quiera utilizar en la app:

Archivo de Gradle del módulo (nivel de app) (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:31.2.0')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Para finalizar, luego de agregar el complemento y los SDK deseados, es necesario sincronizar el proyecto de Android con archivos de Gradle.

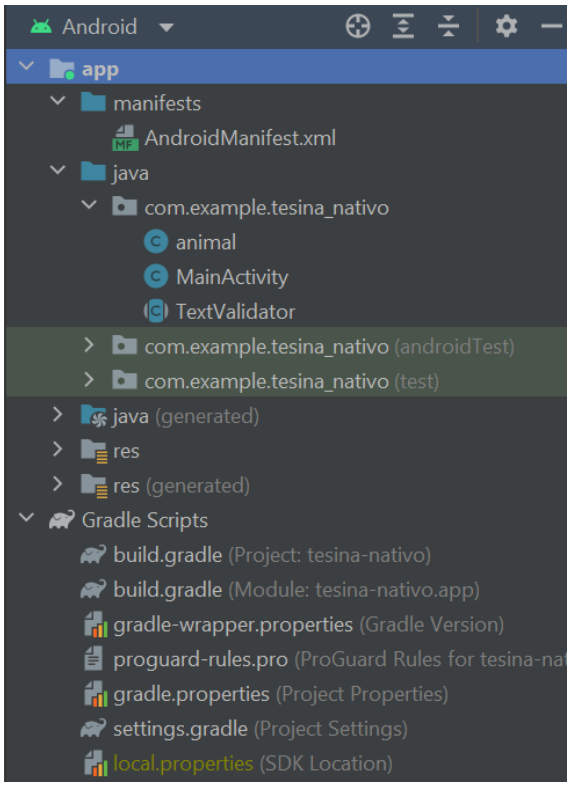
Estructura del proyecto

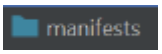
Los proyectos de Android Studio contienen todo lo que define el espacio de trabajo para una aplicación, desde código fuente y recursos hasta código de prueba y las configuraciones de compilación.

Android Studio crea la estructura necesaria para todos los archivos y los hace visibles en la ventana Project de Android Studio.

Vista Android

De forma predeterminada, Android Studio muestra los archivos del proyecto en la vista Android. Esta vista no refleja la jerarquía real del archivo, se ocultan determinados archivos o directorios que no se utilizan con frecuencia y en su lugar se organiza en módulos y tipos de archivos para simplificar la navegación entre los principales archivos fuente del proyecto.





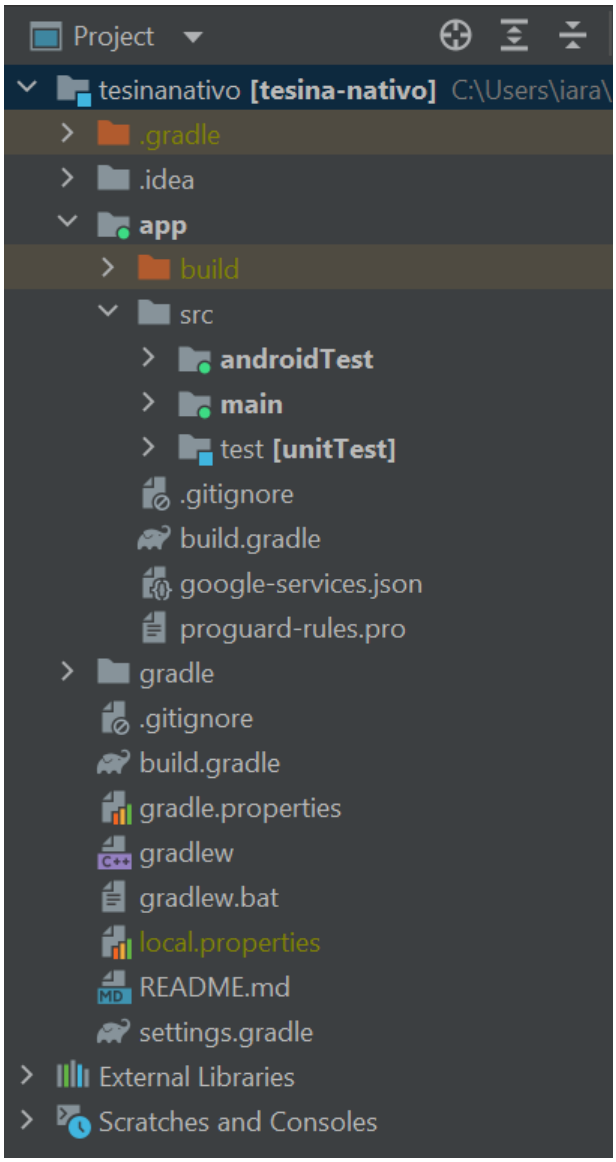
El archivo de manifiesto describe información esencial de la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play.

Contiene los archivos de código fuente de Kotlin y Java separados por nombres de paquetes, incluido el código de prueba JUnit.

Contiene todos los recursos sin código, como diseños XML, strings de IU y también imágenes de mapa de bits, divididos en subdirectorios.

Vista Proyecto

Para ver la estructura de archivos real del proyecto, incluidos todos los archivos ocultos de la vista de Android, desde la IDE hay que seleccionar la vista Project en el menú de la parte superior de la ventana.



build

Contiene resultados de compilación.

src

Contiene todos los archivos de código y recursos para el módulo en sus subdirectorios:

androidTest

Contiene código para las pruebas de instrumentación que se ejecutan en un dispositivo Android.

main

Contiene los archivos de conjunto de orígenes "principales" (el código y los recursos de Android).

test

Contiene código para pruebas locales que se ejecutan en tu JVM host.

gradle

Este archivo define tu configuración de compilación que se aplica a todos los módulos.

- Dentro de la carpeta /src el archivo build.gradle define las configuraciones de compilación específicas para el módulo.
- Dentro de la carpeta /app el archivo build.gradle define la configuración de compilación que se aplica a todos los módulos. Es esencial para el proyecto.

Ejecución de la aplicación

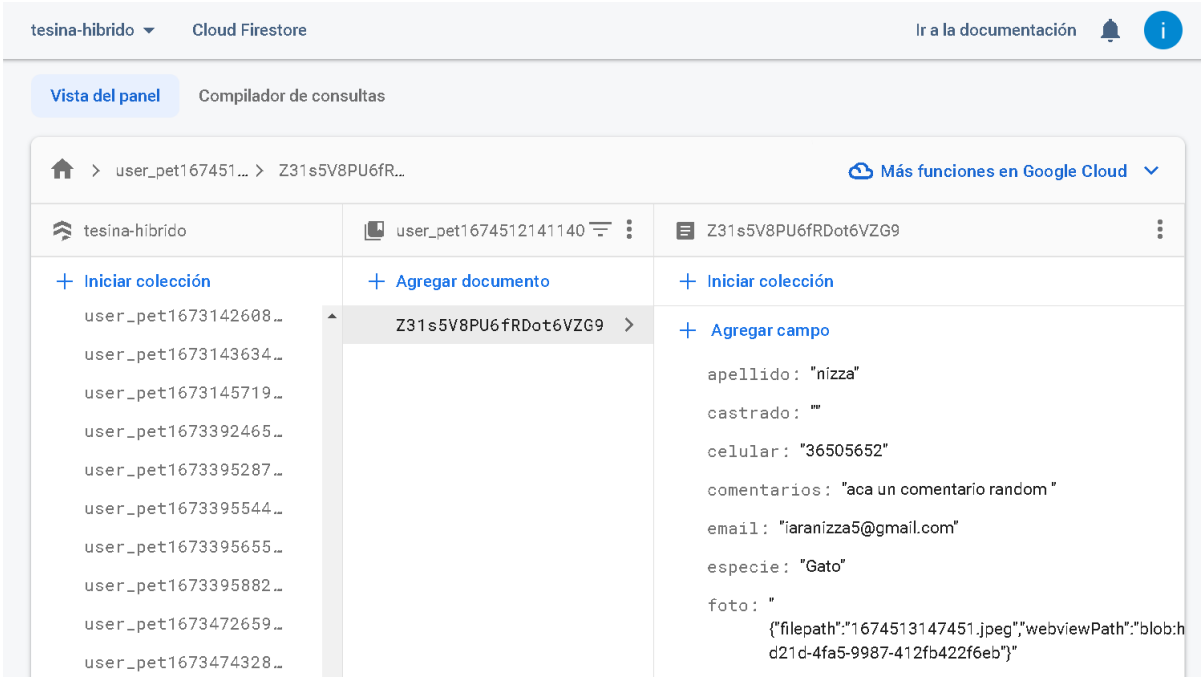
Android Studio permite utilizar el emulador Emulator para simular dispositivos Android en una computadora, tiene diversos modelos de dispositivos Android. Por otro lado también permite conexión a un dispositivo mediante USB. Para el caso de este proyecto, se utilizó la opción de ejecutar la aplicación mediante cable USB a un dispositivo Xiaomi MI 9 Lite con el Sistema Operativo Android 10 QKQ1.190828.002.

Modelo de datos

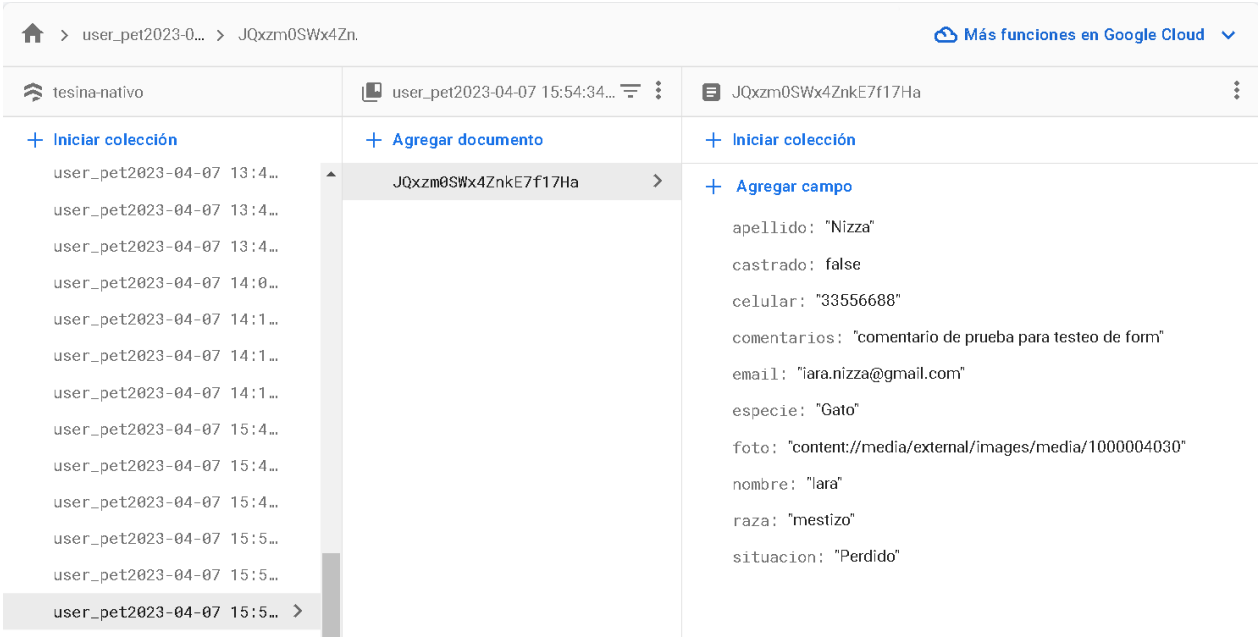
Firestore es una base de datos NoSQL orientada a documentos, se diferencia de una base de datos SQL en que no hay tablas ni filas. En su lugar, almacena los datos en documentos, que se organizan en colecciones. Cada documento contiene un conjunto de pares clave-valor.

Todos los documentos se deben almacenar en colecciones. Pueden contener subcolecciones y objetos anidados, que pueden incluir campos primitivos, como strings, o tipos de objetos complejos, como listas.

En Firestore, la unidad de almacenamiento es el documento. Un documento es un registro liviano que contiene campos con valores asignados. Cada documento se identifica con un nombre.



Vista de panel de control de Cloud Firestore de la aplicación Híbrida



Vista de panel de control de Cloud Firestore de la aplicación Nativa

Desarrollo del Prototipo

Herramientas

A continuación se compararán y analizarán métricas específicas sobre ambos prototipos funcionales.

MyFEPS es un proyecto que tiene como objetivo crear un framework integrativo que, tomando en cuenta todos los factores que influyen en el proceso de evaluación de la calidad de un producto, genera para cada producto específico una metodología de evaluación. El framework también proporciona herramientas de medición y métodos de análisis. [11]

En esta tesis se estudia en específico las aplicaciones de los puntos 3.1 y 3.7 de MyFEPS titulados Adaptabilidad y Eficiencia. Las métricas analizadas fueron:

3.1.1.Adaptabilidad a diferentes entornos

3.7.2.En los tiempos de respuesta

3.7.3.En la utilización de memoria interna

3.7.5.En la utilización del CPU

Para las siguientes métricas se utilizaron las herramientas de Android Profiler que proporcionan datos en tiempo real que ayudan a comprender la forma en que las aplicaciones utilizan los recursos de la CPU, la memoria, la red y la batería de un dispositivo celular en cuanto este esté conectado mediante un cable USB con el modo Debugger activado. [12]

Pruebas y resultados

Adaptabilidad

En el contexto de su uso, se instala y funciona en diferentes entornos y para diferentes actores con el mismo o mayor grado de calidad (que no toma en cuenta esta característica) que el pre-establecido y sin que sean necesarios recursos adicionales a los pre-establecidos.

Evaluable en términos de los recursos necesarios para la instalación y adaptación a diferentes entornos y diferentes actores y los grados de calidad del sistema por cambios de contexto.

Adaptabilidad a diferentes entornos

Este es un análisis subjetivo relacionado al tiempo que llevo la adaptación a ambos entornos, para el desarrollo de ambos prototipos se contaba con conocimientos en programación y algoritmia, en base a esto se redujo el tiempo de aprendizaje y la construcción de ambos proyectos. De todos modos, no se contaba con experiencia previa en desarrollo de aplicaciones móviles en Ionic con Angular ni en Java con Android Studio. Se calculó el tiempo de desarrollo que llevó cada prototipo mediante la siguiente métrica:

Código	Atributo	Métrica
01.1.1.U_1.2.1.A.	Esfuerzo (en horas-hombre) necesario para su adaptación a diferentes entornos en uso	En un período pre-establecido se colectan: 1. El número de entornos de uso a los que se adaptó el sistema: NE HHI ERI = 1 - (HHI - HHIP) / HHIP 2. El número TOTAL de horas hombre que llevaron las adaptaciones: NTHHA 3. Establecer un óptimo de número de horas hombre por adaptación: ONHHpA NHHpA = NTHHA / NE M = ONHHpA / NHHpA con cota superior: 1

Acrónimos: [13]

- NTHHA: Número total de horas hombre que llevaron las adaptaciones
- ONHHpA: Óptimo número de horas hombre por adaptación
- OTCpA: Horas. Adaptación
- NE: número de entornos de uso a los que se adaptó el sistema

Se considera a NTHHA como la suma del tiempo (en horas) que tomaron las configuraciones de los entornos de desarrollo, la configuración para correr las aplicaciones desde dispositivo móvil, y la suma del tiempo que llevaron los desarrollos de código para cada prototipo, incluyendo la creación y validación de los datos de un formulario, la carga de imágenes en el mismo y la configuración de una conexión a la base de datos de Firebase para el posterior almacenamiento de la información.

A su vez NE es considerado como 1 para ambas métricas dado que ambos prototipos se adaptaron a un único entorno de uso (IDE) para cada caso en particular.

El ONHHpA se estableció en 4 días enteros para el desarrollo completo de cada prototipo, considerando las funcionalidades y configuraciones a tener en cuenta para cada caso.

Aplicación Híbrida

- NHHpA = NTHHA / NE = 41hs / 1
- ONHHpA = 32hs
- M = ONHHpA / NHHpA con cota superior: 1
- M = 32/41 = 0.78

Aplicación Nativa

- NHHpA = NTHHA / NE = 52hs / 1
- ONHHpA = 32 hs
- M = ONHHpA / NHHpA con cota superior: 1
- M = 32/52 = 0.61

Eficiencia

Realiza sus funciones usando cantidades de recursos iguales o menores a los pre-establecidos. Evaluable en términos de la medición del uso de los recursos.

En los tiempos de respuesta

Para la siguiente métrica se calculó el tiempo de respuesta que le llevó a cada prototipo guardar la información en la base de datos al crear la publicación.

Código	Atributo	Métrica
07.2.3.U	Tiempo de Respuesta de la Función	1. Obtener el MOEF (Minutos de operación Estándar) para la función tipo x 2. Obtener un conjunto "N" (Tal que N >= 3) de Operadores para las Mediciones 3. Para SumDTP=0 y i=1 Hasta "N", de a Uno hacer a. Medir los minutos hombre que llevó la operación de Muestra MHO b. SumDTP+=MHOi Fin Para 4. Media de Tiempos MTO=SumDTP/N 5. Valoración = Si (MTO <= MOEF) entonces 1 sino MOEF/MTO

Acrónimos:

- MOEF: Minutos de operación Estándar
- SumDTP: Sumatoria de los MHO, para las “N” interfaces de Estudio
- MHO: minutos hombre que llevó la operación de Muestra
- MTO: Media de Tiempos

Aplicación Híbrida

- MOEF: 1.2s
- SumDTP: 1.54s + 2.3s + 2.1s + 1.6s + 1.59s = 9.13s
- MTO: 9.13/5 = 1.82s
- Valoración = (MOEF/MTO) = 1.2/1.82 = 0.65

Aplicación Nativa

- MOEF: 1s
- SumDTP: 0.3s + 0.4s + 0.4s + 0.5s + 0.42s = 2.02s
- MTO: 2.02s/5 = 0.404s
- Valoración = (MTO <= MOEF) = 1

En la utilización de memoria interna

Para la siguiente métrica se calculó el consumo más alto de memoria interna que le llevó a cada prototipo al inicializar la aplicación y al almacenar la información en la base de datos a la hora de crear una nueva publicación.

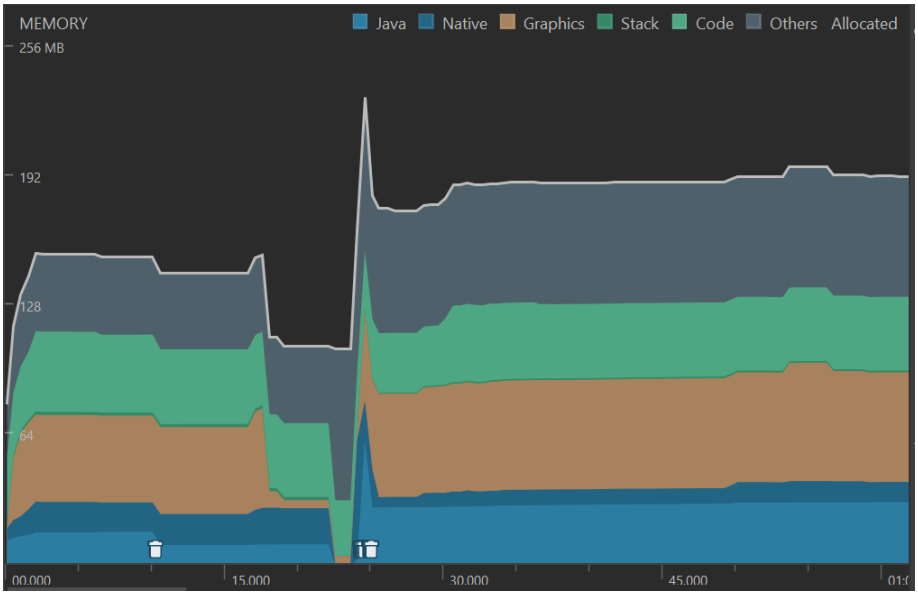
Para esto se utilizó la herramienta de Android Profiler MEMORY, integrada en Android Studio. [14]

Cómo se registra la memoria:

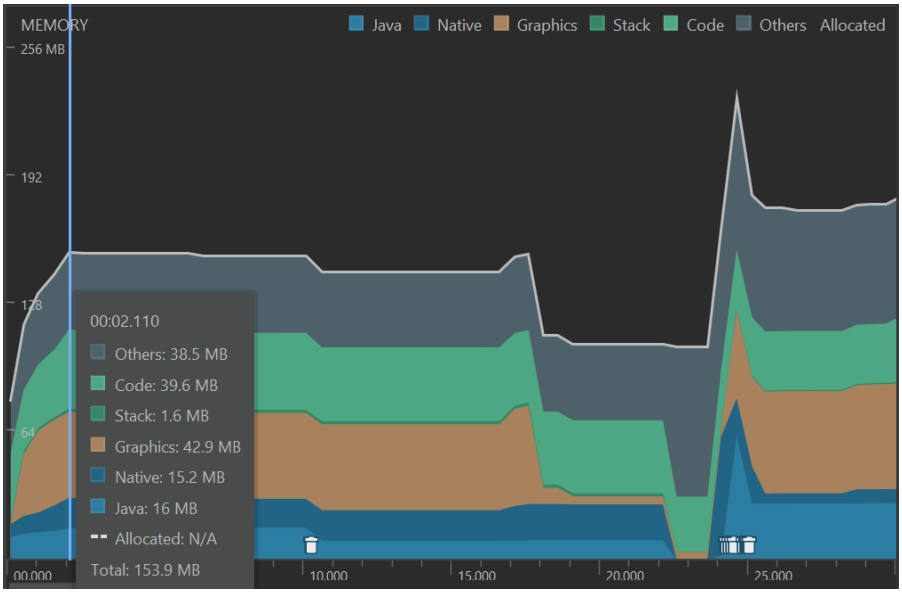
- Java: Es la memoria de objetos asignados desde código Java.
- Nativa: Es la memoria de objetos asignados desde código C o C++. Puede verse memoria nativa usada aquí porque el framework de Android utiliza ese tipo de memoria para administrar varias tareas; por ejemplo, cuando maneja elementos de imágenes y otros gráficos).
- Gráficos: Es la memoria usada para colas de búfer de gráficos con el propósito de mostrar píxeles en la pantalla (se trata de memoria compartida con la CPU, y no de memoria dedicada de la GPU).
- Pila: Es la memoria usada tanto por las pilas nativas como por las de Java en la app. Esto normalmente se relaciona con la cantidad de subprocesos en ejecución que tiene la app.
- Código: Es la memoria utilizada por la aplicación para código y recursos, como código de bytes dex, código dex optimizado o compilado, bibliotecas .so y fuentes.
- Otras: Esta categoría incluye la memoria usada por la app que el sistema no sabe cómo categorizar.
- Asignada: Es la cantidad de objetos Java asignados por la app (no se tienen en cuenta los objetos asignados en C o C++).

Código	Atributo	Métrica
07.3.1.U.	Memoria Interna Utilizada para ejecutar la Función Tipo X	Con una Aplicación Adhoc, Para todas las Funciones 1. Medir CMF(x) cantidad de Memoria usada por la Función en Carga Alta 2. Obtener una Cantidad de Memoria del sistema: CMS 3. Valoración = 1- (CMF/CMS) Valor Final = Media de Valoración Parcial

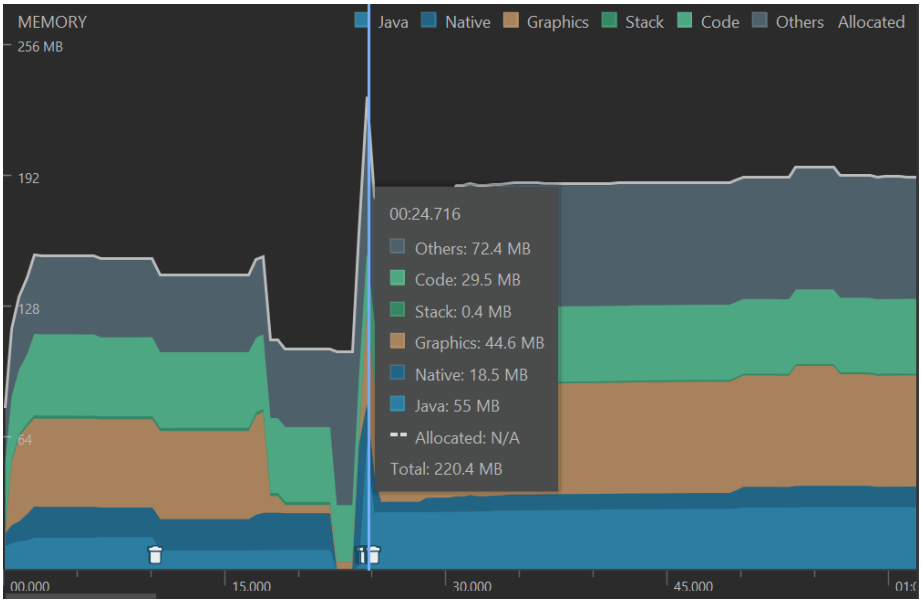
Aplicación Híbrida



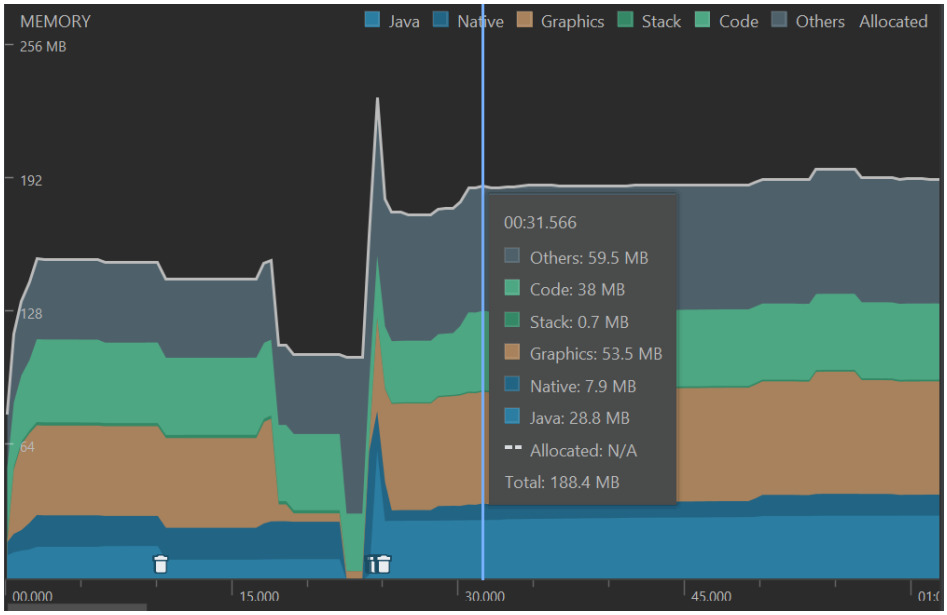
Ciclo de vida completo



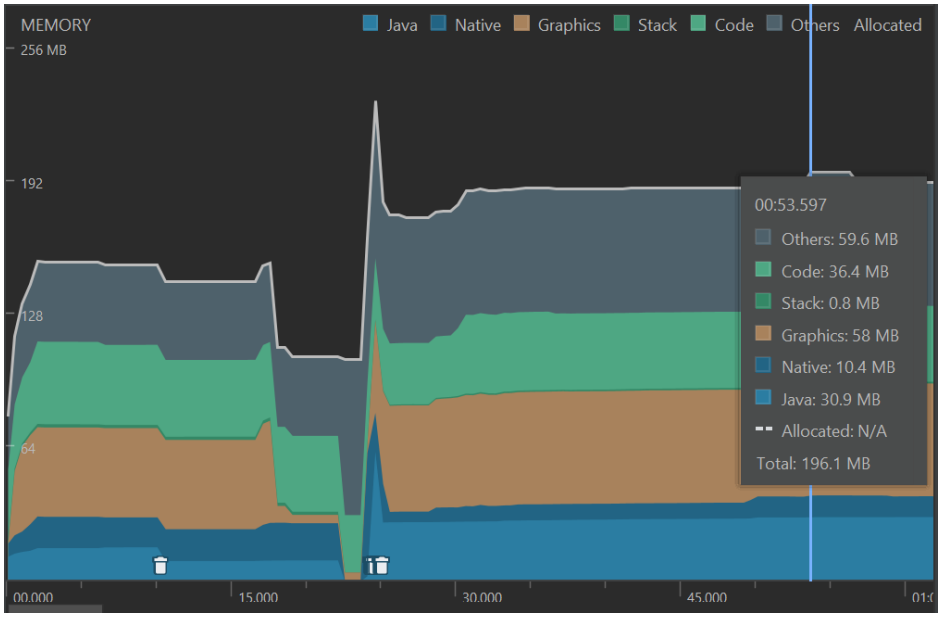
Carga inicial de la aplicación



Al capturar una imagen



En la llamada a la base de datos para almacenar la publicación



Al traer datos de la base de datos

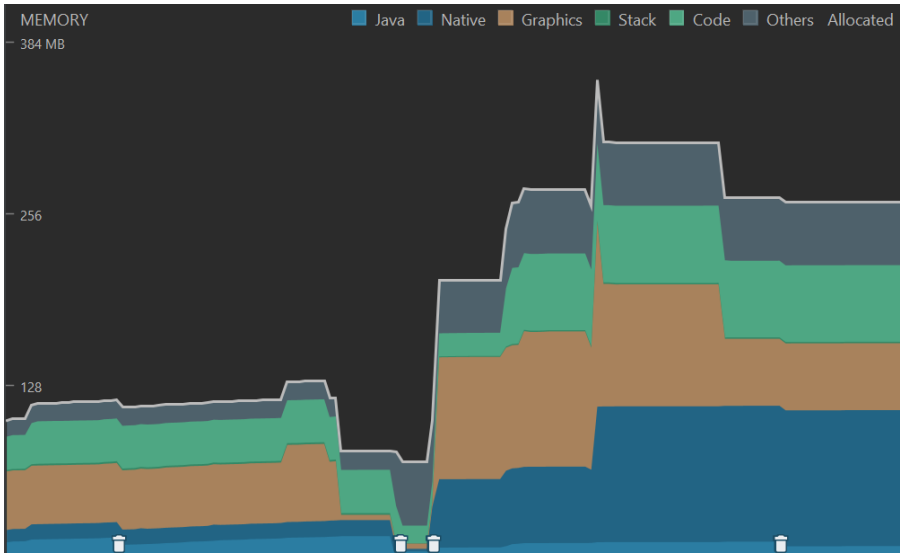
Métrica sobre consumo de memoria interna utilizada al inicializar la aplicación Híbrida:

- Se tomó la Función en Carga Alta como el punto de consumo más alto de memoria interna en la carga inicial de la aplicación, siendo la sumatoria de todas las memorias, donde CMF = 153.9 MB
- Luego para la Cantidad de Memoria del sistema se estableció un valor de CMS = 384 MB
- Valoración = 1- (CMF/CMS) = 1- (153.9/384) = 0.60

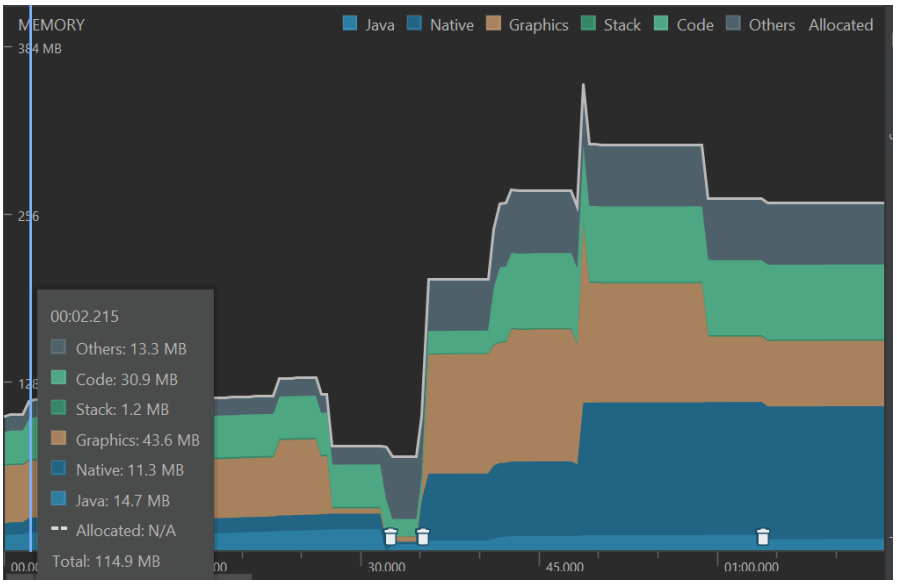
Métrica sobre consumo de memoria interna utilizada al almacenar la información en la base de datos:

- Se tomó la Función en Carga Alta como el punto más alto de consumo de memoria en la llamada a la base de datos para traer los datos almacenados de la publicación, donde CMF = 196.1 MB
- Luego para la Cantidad de Memoria del sistema se estableció un valor de CMS = 384 MB
- Valoración = 1- (CMF/CMS) = 1- (196.1/384) = 0.49

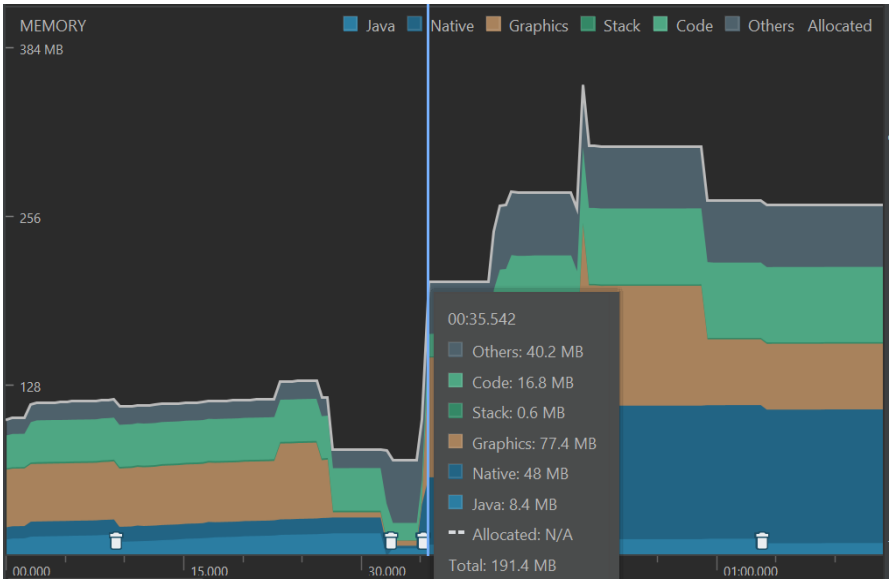
Aplicación Nativa



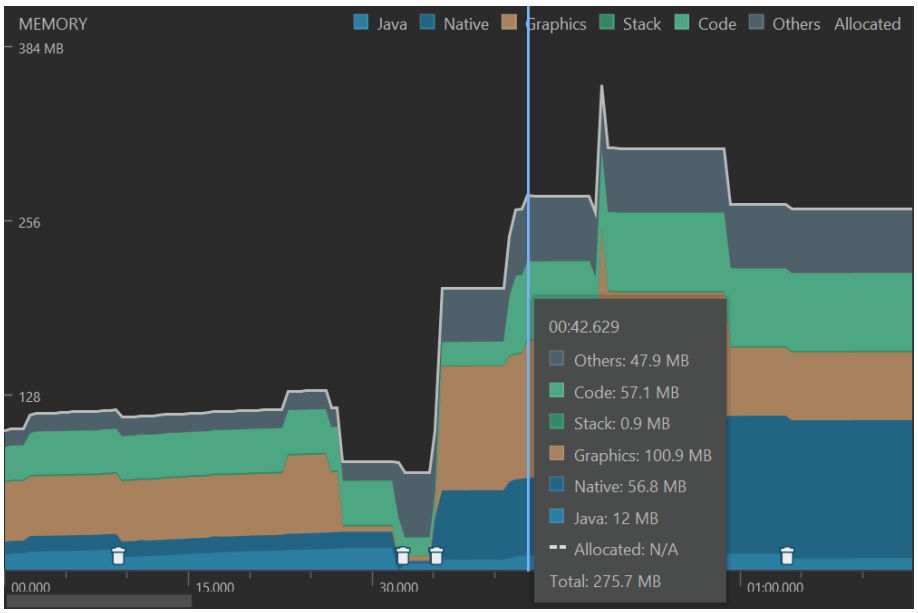
Ciclo de vida completo



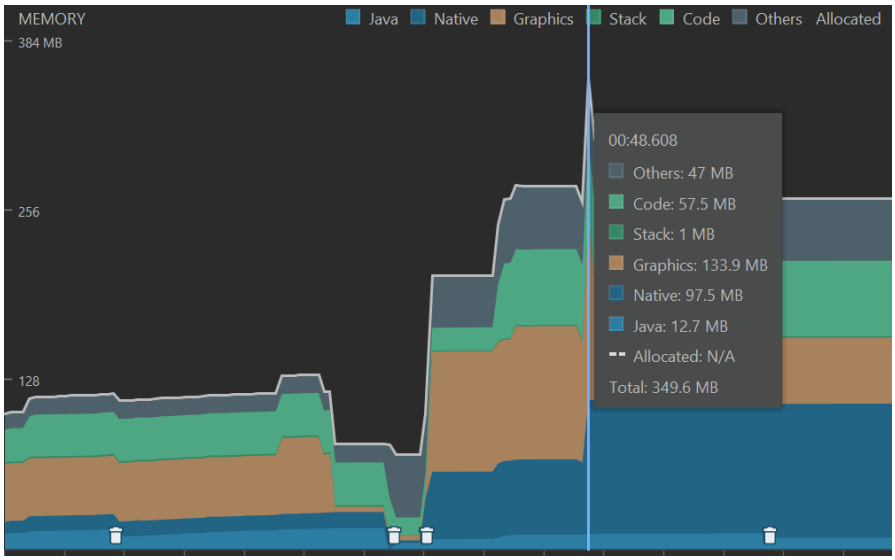
Carga inicial de la aplicación



Al capturar una imagen



En la llamada a la base de datos para almacenar la publicación



Al traer datos de la base de datos

Métrica sobre consumo de memoria interna utilizada al inicializar la aplicación Nativa:

- Se tomó la Función en Carga Alta como el punto de consumo más alto de memoria interna en la carga inicial de la aplicación, siendo la sumatoria de todas las memorias, donde CMF = 114.9 MB
- Luego para la Cantidad de Memoria del sistema se estableció un valor de CMS = 384 MB
- Valoración = 1- (CMF/CMS) = 1- (114.9/384) = 0.71

Métrica sobre consumo de memoria interna utilizada al almacenar la información en la base de datos:

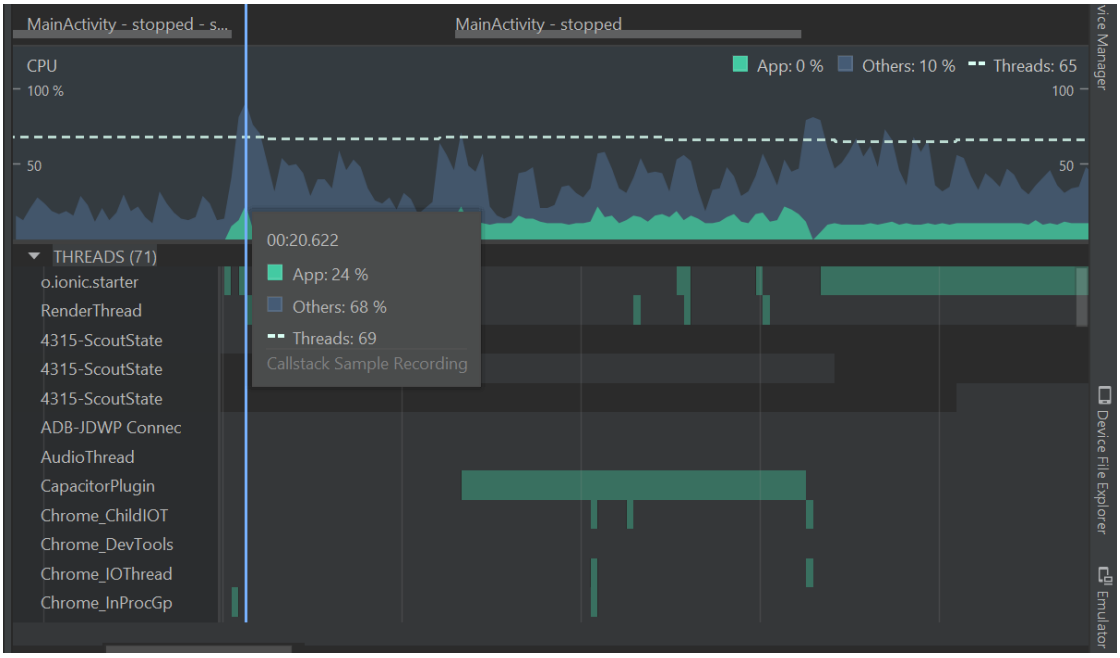
- Se tomó la Función en Carga Alta como el punto más alto de consumo de memoria en la llamada a la base de datos para traer los datos almacenados de la publicación, donde CMF = 349.6 MB
- Luego para la Cantidad de Memoria del sistema se estableció un valor de CMS = 384 MB
- Valoración = 1- (CMF/CMS) = 1- (349.6/384) = 0.09

En la utilización del CPU

Código	Atributo	Métrica
07.5.1.U.	CPU Utilizada para ejecutar la Función Tipo X	Con una Aplicación Adhoc, Para todas las Funciones 1. Con una Aplicación Adhoc, Para todas las Funciones 2. Valoración = 1- (Carga/100) Valor Final = Media de Valoración Parcial

Los siguientes gráficos muestran cada subproceso que pertenece al proceso de las apps e indican su actividad en un cronograma. Los subprocesos en verde están activos o listos para usar la CPU. Es decir, se está ejecutando o pueden ejecutarse.

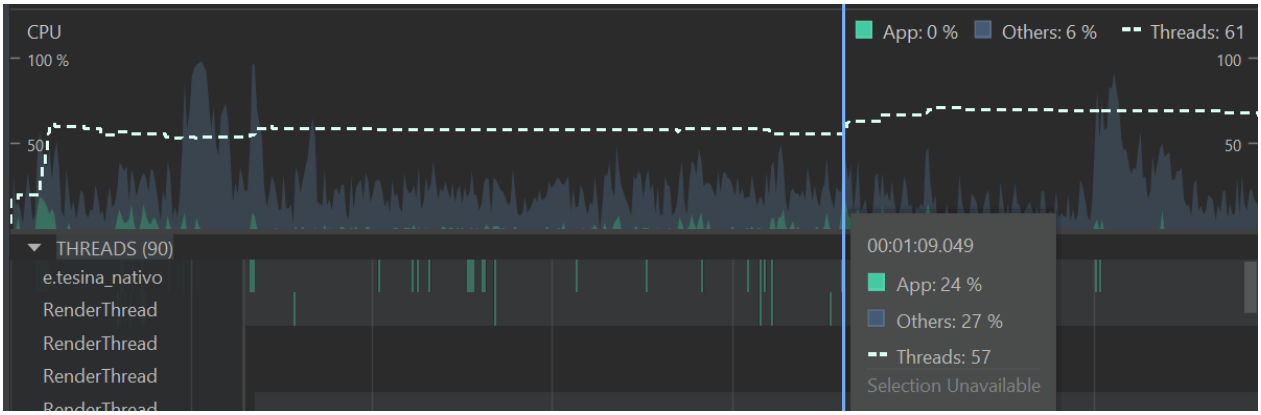
Aplicación Híbrida



Uno de los momentos de mayor consumo es al inicializar la aplicación con un consumo de CPU de 24%, mismo porcentaje de consumo utilizado para almacenar la publicación en la nube.

- Valoración = 1- (Carga/100) = 1 - (24/100) = 0.76

Aplicación Nativa



Análisis del uso de CPU para la función de almacenamiento de la publicación en la nube, valor 24%. Otro de los momentos de mayor consumo es al inicializar la aplicación con un consumo de CPU de 20%.

- Valoración = 1- (Carga/100) = 1 - (24/100) = 0.76
- Valoración = 1- (Carga/100) = 1 - (20/100) = 0.80

Conclusiones

Sobre el Estado del Arte

El objetivo ha sido la comparación de ambos prototipos funcionales desarrollados con tecnologías distintas aplicando mismas funcionalidades con la finalidad de analizar cuándo es más conveniente desarrollar una aplicación de manera híbrida y cuándo de forma nativa.

Para todos los ítems de calidad, MyFEPS define una escala continua del 0 al 1 para determinar su grado de calidad, siendo 1 el valor máximo y 0 el valor mínimo.

El modelo MyFEPS dio como resultado los siguientes grados de calidad:

Métricas MyFEPS	**Prototipo Híbrido**	**Prototipo Nativo**
Adaptabilidad a diferentes entornos	0.78	0.61
En los tiempos de respuesta	0.65	1
En la utilización de memoria interna	0.60 0.49	0.71 0.09
En la utilización del CPU	0.76 0.76	0.76 0.80

En relación al tiempo de desarrollo que llevó cada prototipo, y sabiendo que la aplicación nativa demandó 11 horas-hombre más para lograr las mismas funcionalidades que la aplicación híbrida, de esta comparativa se desprende que en relación a la adaptabilidad de la misma aplicación a diferentes entornos, el uso de un framework agiliza el tiempo de desarrollo.

Por otra parte, habiendo analizado la eficiencia en los tiempos de respuesta de ambos prototipos, luego de realizar pruebas de performance al crear múltiples publicaciones en cada caso. A la hora

de almacenar la información en las bases de datos, puede observarse que la aplicación nativa responde de forma notablemente más rápida que la híbrida, por momentos superándola con ventajas de hasta 2 segundos.

A nivel uso de recursos, en relación a la memoria interna, al inicializarse cada aplicación puede observarse que la híbrida consume más memoria que la nativa, con una diferencia de más de 40 MB.

Por otro lado, al analizar la memoria interna en la llamada a la base de datos, la aplicación nativa consumió aproximadamente 150 MB más que la híbrida.

Finalmente, a nivel uso de recursos, en relación al consumo de CPU, se registró una leve diferencia respecto a la inicialización de las aplicaciones, consumiendo un 4% menos de CPU la aplicación nativa. En referencia al almacenamiento en la base de datos, ambas aplicaciones registraron un uso del 24%.

Sobre la Investigación y Aplicaciones resultantes

Los objetivos específicos de este trabajo planteaban analizar y comparar ambos prototipos funcionales para contar con los criterios técnicos y generales a tener en consideración a la hora de tomar la decisión de implementar una tecnología híbrida o una nativa.

La hipótesis de la cual surgió esta tesina buscaba demostrar por qué la implementación de un desarrollo híbrido es más conveniente en la creación de aplicaciones móviles programadas por trabajadores independientes y freelancers, haciendo hincapié en los costos, la curva de aprendizaje, la escalabilidad, el mantenimiento y los tiempos de entrega.

Luego del desarrollo de ambos prototipos, y habiendo realizado las pruebas correspondientes, esta investigación derivó en las conclusiones a continuación mencionadas.

En cuanto a los tiempos de desarrollo, los frameworks de programación proporcionan una estructura y una base sólida para el desarrollo de aplicaciones, ya que ofrecen una serie de herramientas y bibliotecas predefinidas que permiten a los desarrolladores construir aplicaciones de manera más rápida y eficiente, acelerando el proceso de desarrollo, al aprovechar estas características evitando tener que crear todo desde cero. Los trabajadores independientes pueden tener limitaciones de tiempo para aprender múltiples lenguajes y frameworks específicos de cada plataforma. Con el desarrollo híbrido, al utilizar tecnologías web estándar como HTML, CSS y JavaScript, los desarrolladores que ya estén familiarizados con estas tecnologías pueden trasladar fácilmente sus habilidades al desarrollo móvil.

Respecto a los tiempos de respuesta, las aplicaciones nativas suelen tener una mayor capacidad de respuesta, ya que están desarrolladas utilizando lenguajes de programación específicos de la plataforma, que están optimizados y diseñados para funcionar directamente con el sistema operativo y los componentes nativos del dispositivo, logrando aprovechar al máximo las capacidades y el rendimiento de la plataforma. También tienen acceso directo a las APIs y funcionalidades de los dispositivos, lo que les permite interactuar de manera eficiente con características como la cámara y otros componentes nativos. Y además, pueden utilizar

directamente los componentes y bibliotecas nativas proporcionados por el sistema operativo y la plataforma. Todo esto logra traducirse en una mayor velocidad y capacidad de respuesta.

En contraste, las aplicaciones híbridas están construidas con tecnologías web que luego se envuelven en una capa nativa utilizando frameworks como Ionic. Aunque estos frameworks han mejorado en términos de rendimiento, las aplicaciones híbridas todavía tienen una capa adicional de abstracción entre el código y el sistema operativo, lo que tiende a introducir cierta latencia y afectar la velocidad de respuesta en comparación con las aplicaciones nativas.

Desde un punto de vista del uso de memoria interna, si bien en la carga inicial de las aplicaciones, a nivel nativo el consumo es menor, pudo observarse un mayor uso de este recurso por parte de la aplicación nativa, a lo largo de todo el ciclo de vida, llegando a picos de casi 384 MB comparado con el ciclo de vida de la aplicación híbrida, que mantuvo su consumo por debajo de los 256 MB.

Aunque no es necesariamente cierto que todas las aplicaciones nativas consuman más memoria interna que las aplicaciones híbridas, esto puede darse debido a la presencia de recursos duplicados, ya que están diseñadas específicamente para una plataforma y, a menudo, utilizan recursos específicos de esa plataforma. Esto puede llevar a la duplicación de ciertos recursos en aplicaciones nativas separadas, aumentando el consumo de memoria. Otro factor a tener en cuenta son las bibliotecas adicionales, ya que las aplicaciones nativas suelen usar bibliotecas específicas para cada plataforma, lo que puede agregar más tamaño a la aplicación en comparación con las aplicaciones híbridas, que a menudo utilizan bibliotecas y marcos multiplataforma.

Las aplicaciones híbridas al estar basadas en tecnologías webs y ser ejecutadas dentro de un contenedor nativo, pueden llegar a consumir menos memoria interna al compartir código entre plataformas y reducir la duplicación de recursos y binarios específicos del sistema operativo. Además, pueden depender menos de bibliotecas específicas del sistema operativo, ya que gran parte de la funcionalidad se implementa mediante tecnologías web reduciendo así la cantidad de bibliotecas nativas.

Luego de toda este proceso de investigación, se puede decir que, los frameworks suelen ofrecer características como la separación de responsabilidades y la modularidad, lo que facilita la escalabilidad de las aplicaciones a medida que crecen en tamaño y complejidad, también suelen incluir características y mecanismos de seguridad integrados que ayudan a proteger las aplicaciones contra amenazas comunes, como ataques de inyección de código. Esto permite a los desarrolladores construir aplicaciones más seguras desde el principio, sin tener que implementar todas las medidas de seguridad manualmente.

Al utilizar un framework, es más probable que el código sea portable entre diferentes plataformas y entornos, ya que suelen estar diseñados para ser independientes de la plataforma y compatibles con diferentes sistemas operativos, lo que permite a los desarrolladores implementar aplicaciones en una variedad de entornos sin tener que reescribir todo el código. Esto significa que los desarrolladores no necesitan crear aplicaciones completamente separadas para cada plataforma, lo que reduce significativamente los tiempos y costos de desarrollo.

Para concluir, es importante tener en cuenta que si bien el desarrollo híbrido tiene sus ventajas, también tiene algunas limitaciones, como posibles problemas de rendimiento en comparación con aplicaciones nativas complejas y restricciones en el acceso a ciertas funcionalidades específicas del dispositivo. Sin embargo, para muchos proyectos, el desarrollo híbrido puede ser una solución rentable y eficiente que permite a los desarrolladores freelancers ser más competitivos en el mercado.

En resumen, los frameworks de programación pueden acelerar el proceso de desarrollo, mejorar la productividad, facilitar el mantenimiento y escalabilidad de las aplicaciones, ofrecer soporte por parte de la comunidad, mejorar la seguridad y proporcionar portabilidad entre diferentes plataformas. Estas ventajas hacen que los frameworks sean herramientas muy valiosas para los desarrolladores en la creación de aplicaciones eficientes y robustas.

Sobre el proceso de Aprendizaje

Durante este proceso fue necesario adquirir conocimientos nuevos tales como a desarrollar en el framework de Ionic con Angular y Capacitor y también en la IDE de Android Studio. Este trabajo pudo ser llevado a cabo gracias a las habilidades adquiridas a lo largo de la carrera. A través de él se logró aprender todo el proceso de desarrollo de una aplicación híbrida y otra nativa, cómo almacenar sus datos en Firestore, cómo tomar distintas métricas a la hora de comprar rendimientos, y cómo manipular la cámara del teléfono para ambos casos. Conocimientos que han resultado ser de gran utilidad para aplicarlos en un plano laboral.

Líneas futuras de investigación

Con respecto a investigaciones futuras en base a ésta se proponen las siguientes:

- Sumar funcionalidades a la aplicación tales como búsquedas avanzadas y utilización de geolocalización, para comparar comportamientos.
- Analizar su desempeño en distintos dispositivos y tomar métricas respecto a la utilización de recursos y los tiempos de respuesta.
- Estudiar la posibilidad de ser testeada en simultáneo por múltiples usuarios.

Bibliografía

[1] *¿Qué es un Framework y qué tipos hay?*. Open Bootcamp. Visitada en Jun 25, 2022. 15:52 hs. [online]. Accesible: https://open-bootcamp.com/aprender-programar/que-es-un-framework

[2] *Historia de los lenguajes de programación*. Wikipedia. Visitada en Jun 25, 2022. 16:20 hs. [online]. Accesible: https://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n

[3] Cloud Firestore. Firebase. Visitada en Jun 25, 2022. 16:39 hs. [online]. Accesible: *https://firebase.google.com/docs/firestore?hl=es-419*

[4]*The pros and cons of hybrid app technology.* Codecontrol. Visitada en Jun 25, 2022. 17:17 hs. [online]. Accesible: *https://codecontrol.io/en/blog/the-pros-and-cons-of-hybrid-mobile-app-technology*

[5] *The mobile SDK for the Web*. Ionic Framework. Visitada en Jun 27, 2022. 18:45 hs. [online]. Accesible: *https://ionicframework.com/*

[6] *Deliver Web Apps with confidence.* Angular Framework. Visitada en Jun 27, 2022. 19:55 hs. [online]. Accesible: *https://angular.io/*

[7] *A cross-platform native runtime for web apps*. Ionic Framework. Visitada en Jun 27, 2022. 21:10 hs. [online]. Accesible: https://capacitorjs.com/

[8] *Pros and cons of native app development*. Ante Baus. Ene 3, 2022. Visitada en Jun 27, 2022. 22:42 hs. [online]. Accesible: https://decode.agency/article/native-app-development-pros-cons/

[9] *What is Java?* Amazon AWS. Visitada en Jun 27, 2022. 23:33 hs. [online]. Accesible: *https://aws.amazon.com/es/what-is/java/*

[10] *Android Studio.* Android Developers. Visitada en Jun 27, 2022. 00:42 hs. [online]. Accesible:

https://developer.android.com/studio?gclid=Cj0KCQjwy9-kBhCHARIsAHpBjHjcKZjIX79MsOmI-4es4MWydcDA1t1QsKDnpR2kjjlBO8zZfekR1JQaAueZEALw_wcB&gclsrc=aw.ds

[11] *Descripción de Atributos y Métricas MyFEPS*. MyFEPS Docs. Visitada en May 15, 2023. 17:26 hs. [online]. Accesible: https://sites.google.com/a/comunidad.ub.edu.ar/myfeps/metricas#__RefHeading___Toc329605222

[12] *Android Profiler*. Android Developers. Visitada en May 16, 2023. 19:33 hs. [online]. Accesible: https://developer.android.com/studio/profile/android-profiler?hl=es-419#:~:text=En%20Android%20Studio%203.0%20y,la%20red%20y%20la%20bater%C3%ADa.

[13] *Acrónimos - MyFEPS*. MyFEPS Docs. Visitada en May 16, 2023. 20:38 hs. [online]. Accesible: https://sites.google.com/a/comunidad.ub.edu.ar/myfeps/acronimos

[14] *Cómo inspeccionar el uso de memoria de tu app con el Generador de perfiles de memoria.* Android Developers. Visitada en May 17, 2023. 22:47 hs. [online]. Accesible: https://developer.android.com/studio/profile/memory-profiler?hl=es-419

Bibliografía auxiliar Consultada

Cómo agregar un menú desplegable en Android Studio. Arooha Arif. Ago 17, 2021. Visitada en Jul 30, 2022. 12:30 hs. [online]. Accesible: https://code.tutsplus.com/es/tutorials/how-to-add-a-dropdown-menu-in-android-studio--cms-37860

Cómo crear una IU responsiva con ConstraintLayout. Android Developers. Visitada en Jul 30, 2022. 13:42 hs. [online]. Accesible: https://developer.android.com/training/constraint-layout?hl=es-419

Create a simple File Chooser in Android. o7planning.org. Visitada en Jul 30, 2022. 20:05 hs. [online]. Accesible: https://o7planning.org/12725/create-a-simple-file-chooser-in-android

OnActivityResult method is deprecated, what is the alternative?. Stackoverflow. Ene 7, 2021. Visitada en Jul 30, 2022. 20:30 hs. [online]. Accesible: https://stackoverflow.com/questions/62671106/onactivityresult-method-is-deprecated-what-is-the-alternative

Android Studio Tutorial: Upload Image using Gallery - Get Image from Gallery. WsCube Tech. Mar 2, 2022. Visitada en Jul 30, 2022. 21:06 hs. [online]. Accesible: https://www.youtube.com/watch?v=bLi1qr6h4T4

Solución startActivityForResult Deprecated - Android Studio. Códigos de Programación - MR. Jun 22, 2022. Visitada en Jul 30, 2022. 22:01 hs. [online]. Accesible: https://www.youtube.com/watch?v=HScPl2QsQ2I

How to upload a project to GitHub. Stackoverflow. Visitada en Jul 31, 2022. 11:39 hs. [online]. Accesible: https://stackoverflow.com/questions/12799719/how-to-upload-a-project-to-github#:~:text=Follow%20these%20two%20steps%3A,your%20newly%20created%20remote%20repo.

Cómo ejecutar apps en un dispositivo de hardware. Android Developers. Visitada en Jul 31, 2022. 12:55 hs. [online]. Accesible: https://developer.android.com/studio/run/device

Cómo activar la Depuración USB en Xiaomi con MIUI 12 o MIUI 13. Héctor Romero. May 23, 2022. Visitada en Jul 31, 2022. 13:47 hs.[online]. Accesible: https://www.tuexpertomovil.com/2022/05/23/como-activar-la-depuracion-usb-en-xiaomi-en-miui-12-y-13/

Install app via usb: The device is temporarily restricted. Stackoverflow. Dec 4, 2017. Visitada en Jul 31, 2022. 14:09 hs. [online]. Accesible: https://stackoverflow.com/questions/46020237/install-app-via-usb-the-device-is-temporarily-restricted

Android ImageView Fixing Image Size. Stackoverflow. Oct 22, 2018. Visitada en Jul 31, 2022. 14:26 hs.[online]. Accesible: https://stackoverflow.com/questions/12049626/android-imageview-fixing-image-size

Diseñar y validar un formulario en Android. Programación y Más. Visitada en Jul 31, 2022. 16:24 hs. [online]. Accesible: https://programacionymas.com/series/app-android-sobre-registro-de-inventarios/disenar-y-validar-un-formulario-en-android

How to set a PNG file to an ImageView?. Stackoverflow. Dec 14, 2017. Visitada en Jul 31, 2022. 17:46 hs.[online]. Accesible: https://stackoverflow.com/questions/3818359/how-to-set-a-png-file-to-an-imageview

How to validate EditText input with a custom regex in Android?. Stackoverflow. Mar 13, 2016. Visitada en Jul 31, 2022. 18:01 hs. [online]. Accesible: https://stackoverflow.com/questions/35971163/how-to-validate-edittext-input-with-a-custom-regex-in-android

How to get EditText value and display it on screen through TextView?. Stackoverflow. Ene 4, 2017. Visitada en Jul 31, 2022. 19:30 hs. [online]. Accesible:

https://stackoverflow.com/questions/4396376/how-to-get-edittext-value-and-display-it-on-screen-through-textview

Show Error on the tip of the Edit Text Android. Stackoverflow. Feb 18, 2019. Visitada en Jul 31, 2022. 21:24 hs.[online]. Accesible: https://stackoverflow.com/questions/18225365/show-error-on-the-tip-of-the-edit-text-android

Gson User Guide. GitHub. Visitada en Ago 11, 2022. 20:33 hs. [online]. Accesible: https://github.com/google/gson/blob/master/UserGuide.md#TOC-Gson-With-Gradle

TextWatcher. Android Developers. Visitada en Ago 11, 2022.21:50 hs. [online]. Accesible: https://developer.android.com/reference/android/text/TextWatcher

Complete Form Validation in android studio - City Guide App. Coding With Tea. May 27, 2020. Visitada en Ago 11, 2022. 21:57 hs. [online]. Accesible: https://www.youtube.com/watch?v=qcDlcITNqnE

RegEx to allow all characters, length should be 1-50 characters. Stackoverflow. Jul 2, 2022. Visitada en Ago 12, 2022. 15:17 hs.[online]. Accesible: https://stackoverflow.com/questions/36196839/regex-to-allow-all-characters-length-should-be-1-50-characters

How to automatically generate getters and setters in Android Studio. Stackoverflow. Abr 10, 2019. Visitada en Ago 12, 2022. 16:13 hs. [online]. Accesible: https://stackoverflow.com/questions/23897215/how-to-automatically-generate-getters-and-setters-in-android-studio

How to get Spinner value?. Stackoverflow. Nov 2, 2014. Visitada en Ago 13, 2022. 12:41 hs. [online]. Accesible: https://stackoverflow.com/questions/1947933/how-to-get-spinner-value
https://convergeddevices.net/what-is-android-storage-emulated-0/

What is Android storage/emulated/0? (The Complete Guide). Darrel Bryant. Visitada en Ago 17, 2022. 15:36 hs. [online]. Accesible: https://convergeddevices.net/what-is-android-storage-emulated-0/

Exception 'open failed: EACCES (Permission denied)' on Android. Stackoverflow. Visitada en Ago 17, 2022. 17:04 hs. [online]. Accesible: https://stackoverflow.com/questions/8854359/exception-open-failed-eacces-permission-denied-on-android

How to store data locally in an Android app. Ryan Haines. Visitada en Ago 18, 2022. 14:28 hs. [online]. Accesible: https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/

Easily Reading and Writing Files to Internal Storage with Gson. Sean McLane. Ene 17, 2018. Visitada en Ago 18, 2022. 16:32 hs. [online]. Accesible: https://medium.com/@smjaejin/easily-reading-and-writing-files-to-internal-storage-with-gson-bdba821ca7de

Run Android Studio after installation (and other programs). Ask Ubuntu. Abr 5, 2018. Visitada en Ago 25, 2022.09:40 hs. [online]. Accesible: https://askubuntu.com/questions/639260/run-android-studio-after-installation-and-other-programs

Guardar imagen en Android Studio. Ingelist. Jul 9, 2021. Visitada en Ago 25, 2022. 14:16 hs. [online]. Accesible: https://www.youtube.com/watch?v=zP8tz8DwOYI&t=629s

Couldn't find meta-data for provider with authority. Stackoverflow. Ene 14, 2019. Visitada en Ago 25, 2022. 14:37 hs. [online]. Accesible: https://stackoverflow.com/questions/56598480/couldnt-find-meta-data-for-provider-with-authority

Cómo almacenar fotos en el dispositivo - Android Studio. Códigos de Programación - MR. Sep 15, 2020. Visitada en Ago 25, 2022. 15:57 hs. [online]. Accesible: https://www.youtube.com/watch?v=4JiYrgjwfaQ

ImageCaptureExample. GitHub. Ene 22, 2018. Visitada en Ago 29, 2022. 14:26 hs. [online]. Accesible: https://github.com/Cassendra4/ImageCaptureExample

How to run my Angular site on mobile device that is running localhost on my windows desktop [duplicate]. Stackoverflow. Mar 28, 2017. Visitada en Sep 9, 2022. 20:17 hs. [online]. Accesible: https://stackoverflow.com/questions/43071535/how-to-run-my-angular-site-on-mobile-device-that-is-running-localhost-on-my-wind

INSTALL_FAILED_USER_RESTRICTED : android studio using redmi 4 device. Stackoverflow. Ago 17, 2021. Visitada en Sep 9, 2022. 00:16 hs. [online]. Accesible: https://stackoverflow.com/questions/47239251/install-failed-user-restricted-android-studio-using-redmi-4-device

Using Capacitor with Ionic. Capacitor Docs. Visitada en Sep 14, 2022. 09:33 hs.[online]. Accesible: https://capacitorjs.com/docs/v2/getting-started/with-ionic

Ionic Capacitor Tutorial - Ionic Build Android & Ionic Build iOS. Code Swag. May 16, 2020. Visitada en Oct 7, 2022. 13:52 hs. [online]. Accesible: https://www.youtube.com/watch?v=y_UUjPkxlZ0

Project Structure. Ionic Framework. Visitada en Oct 7, 2022. 15:43 hs.[online]. Accesible: https://ionicframework.com/docs/v3/intro/tutorial/project-structure/

Adding Pages. Ionic Framework. Visitada en Oct 7, 2022. 16:08 hs.[online]. Accesible: https://ionicframework.com/docs/v3/intro/tutorial/adding-pages/

ionic git remote. Ionic Framework. Visitada en Oct 7, 2022. 16:27 hs.[online]. Accesible: https://ionicframework.com/docs/cli/commands/git-remote

Ionic Angular Overview. Ionic Framework. Visitada en Oct 7, 2022. 16:35 hs. [online]. Accesible: https://ionicframework.com/docs/angular/overview

Your First Ionic App: Angular. Ionic Framework. Visitada en Oct 7, 2022. 21:39 hs. [online]. Accesible: https://ionicframework.com/docs/angular/your-first-app

Using forms for user input. Angular Framework. Visitada en Oct 10, 2022. 14:31 hs. [online]. Accesible: https://angular.io/start/start-forms

Ionic and Forms. Ionic Framework. Visitada en Oct 11, 2022. 19:28 hs. [online]. Accesible:

https://ionicframework.com/docs/v3/developer-resources/forms/

Data Storage in Capacitor. Capacitor Docs. Visitada en Nov 20, 2022. 16:24 hs. [online]. Accesible:

https://capacitorjs.com/docs/guides/storage

FirebaseError: [code=permission-denied]: Missing or insufficient permissions. Stackoverflow. Ene 8, 2019. Visitada en Ene 7, 2023. 16:24 hs.[online]. Accesible: https://stackoverflow.com/questions/56510745/firebaseerror-code-permission-denied-missing-or-insufficient-permissions

Primeros pasos con Cloud Firestore. Firebase Docs. Visitada en Ene 7, 2023. 16:50 hs. [online]. Accesible: https://firebase.google.com/docs/firestore/quickstart

Dependency management in Gradle. Gradle Docs. Visitada en Ene 14, 2023. 16:42 hs. [online]. Accesible: https://docs.gradle.org/7.4.2/userguide/dependency_management.html#sub:centralized-repository-declaration

How to get current timestamps in Java. Mkyong. Mar 24, 2021. Visitada en Ene 14, 2023. 17:48 hs. [online]. Accesible: https://mkyong.com/java/how-to-get-current-timestamps-in-java/

Android add ID to a layout. Stackoverflow. Jul 4, 2013. Visitada en Ene 15, 2023. 14:04 hs. [online]. Accesible: https://stackoverflow.com/questions/17471808/android-add-id-to-a-layout

How do I change TextView Value inside Java Code?. Stackoverflow. Jun 5, 2015. Visitada en Ene 15, 2023. 14:09 hs. [online]. Accesible: https://stackoverflow.com/questions/4768969/how-do-i-change-textview-value-inside-java-code

How can I store a string locally in Android? [duplicate]. Stackoverflow. Nov 9, 2015. Visitada en Ene 15, 2023. 15:14 hs. [online]. Accesible: https://stackoverflow.com/questions/33616342/how-can-i-store-a-string-locally-in-android

CharSequence vs. String in Java. Baeldung. Visitada en Ene 16, 2023. 22:37 hs. [online]. Accesible: https://www.baeldung.com/java-char-sequence-string

The layout <layout> in layout has no declaration in the base layout folder [error]. Stackoverflow. Sep 28, 2018. Visitada en Ene 16, 2023. 23:32 hs.[online]. Accesible: https://stackoverflow.com/questions/52547657/the-layout-layout-in-layout-has-no-declaration-in-the-base-layout-folder-erro

Operator + cannot be applied to java.lang.charsequence. Stackoverflow. Nov 11, 2014. Visitada en Ene 17, 2023. 12:53 hs. [online]. Accesible: https://stackoverflow.com/questions/26860329/operator-cannot-be-applied-to-java-lang-charsequence

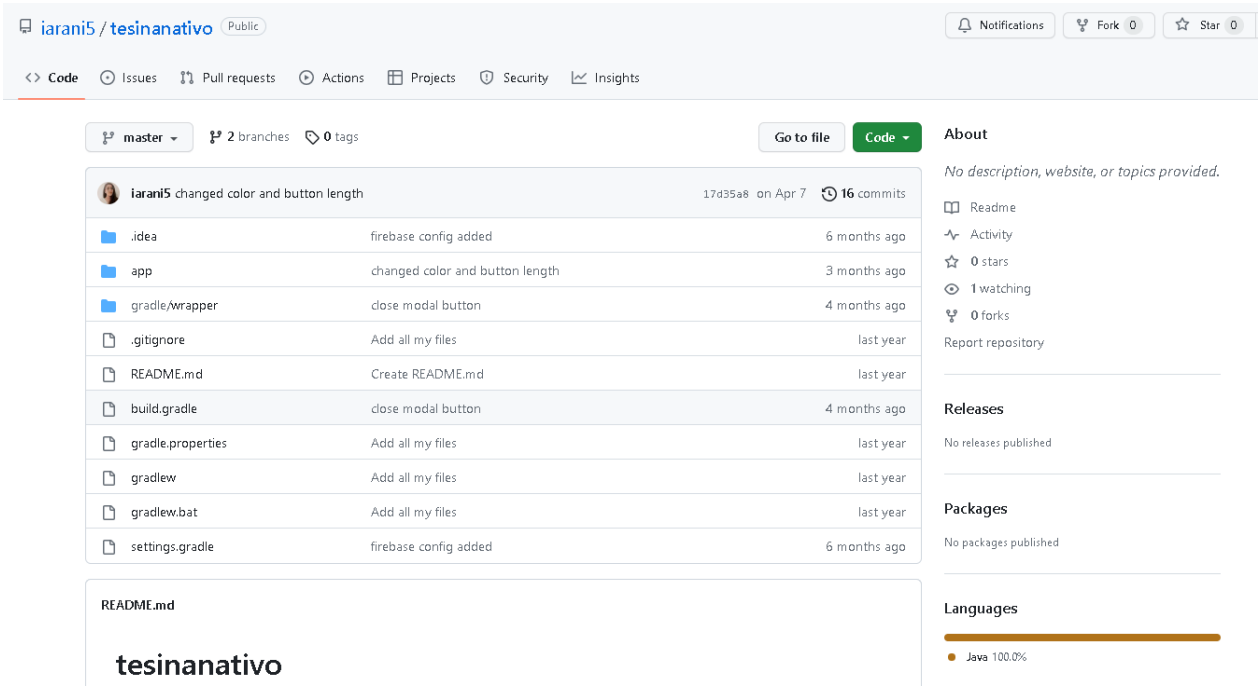
How to load an ImageView by URL in Android?. Stackoverflow. Ene 28, 2013. Visitada en Ene 17, 2023. 13:03 hs. [online]. Accesible: https://stackoverflow.com/questions/2471935/how-to-load-an-imageview-by-url-in-android

Picasso Library. Picasso. Visitada en Ene 17, 2023. 13:24 hs. [online]. Accesible: https://square.github.io/picasso/

Anexo

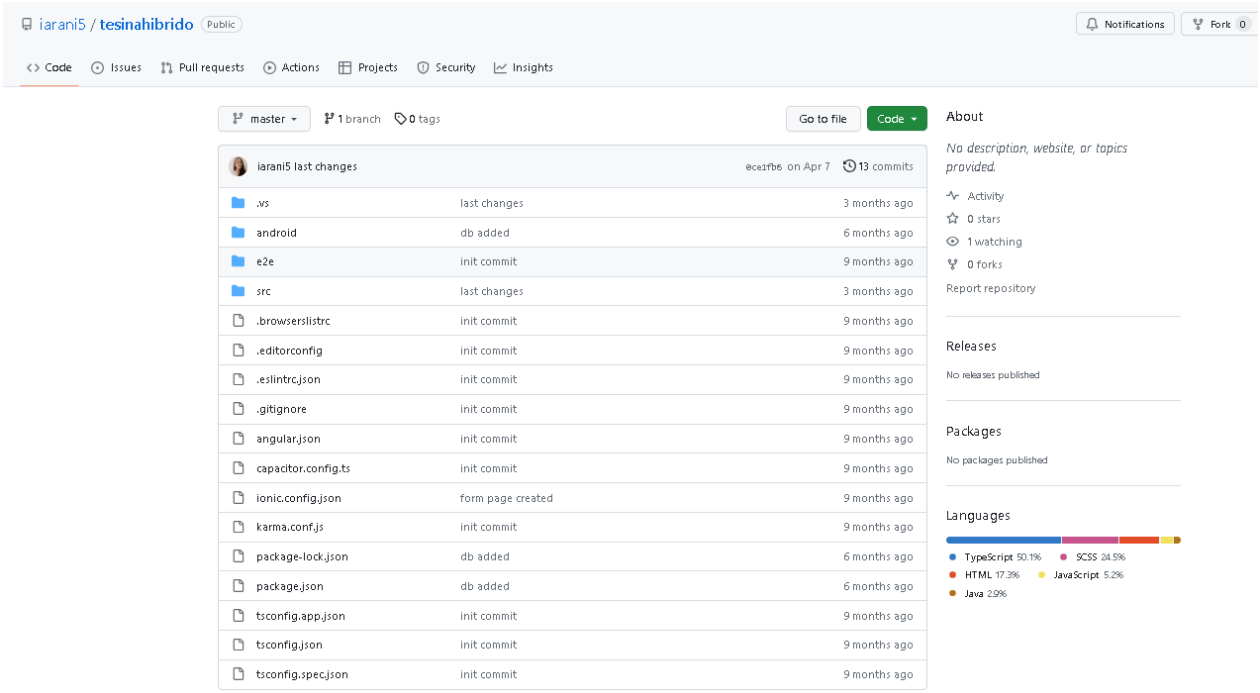
Repositorio Prototipo Nativo

https://github.com/iarani5/tesinanativo



Repositorio Prototipo Híbrido

https://github.com/iarani5/tesinahibrido



Glosario

- JSON: Es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript. Está compuesto por una combinación de pares clave-valor, donde las claves son cadenas de texto y los valores pueden ser de varios tipos de datos, como cadenas, números, booleanos, objetos, matrices o valores nulos.
- WebView: Android System WebView es un componente desarrollado por la compañía tecnológica Google, el cual añadieron a todas las versiones del sistema operativo Android con el objetivo de que los programadores incluyan contenido web a sus aplicaciones. Para ser claros, te ayudará a abrir una página web desde el interior de un aplicativo sin la necesidad de acudir a un buscador como Google Chrome, Safari, Microsoft Edge, etc.
- Progressive Web App (PWA): Es un tipo de aplicación web que combina características de las páginas web tradicionales y de las aplicaciones móviles nativas. Las PWAs están diseñadas para ofrecer una experiencia de usuario avanzada, permitiendo a los usuarios interactuar con ellas de manera similar a como lo harían con una aplicación nativa, incluso en dispositivos móviles.
- Modelo Vista Controlador (MVC): Es un patrón de arquitectura de software ampliamente utilizado en el desarrollo de aplicaciones para separar la lógica de la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Cada uno de estos componentes tiene una función específica en la aplicación y se comunica con los demás según ciertas reglas predefinidas.
- Web Apps: También conocida como aplicación web, es una aplicación que se accede y utiliza a través de un navegador web en lugar de ser descargada e instalada en un dispositivo. Estas aplicaciones se ejecutan en un servidor y los usuarios pueden interactuar con ellas a través de una interfaz de usuario accesible en su navegador web.

- SDK: Es el acrónimo de "Software Development Kit" o, en español, "Kit de Desarrollo de Software". Es un conjunto de herramientas, bibliotecas, documentación y ejemplos de

código que facilitan y agilizan el desarrollo de aplicaciones para una plataforma o sistema específico.

- IDE: Es el acrónimo de "Integrated Development Environment" o, en español, "Entorno de Desarrollo Integrado". Se trata de una aplicación informática que ofrece un conjunto de herramientas y características integradas en una sola interfaz para facilitar y mejorar el proceso de desarrollo de software.
- JVM: Son las siglas de "Java Virtual Machine". Es una parte fundamental del entorno de ejecución de Java y se encarga de interpretar y ejecutar el código bytecode generado por el compilador Java
- API: Es el acrónimo de "Application Programming Interface" o, en español, "Interfaz de Programación de Aplicaciones". Es un conjunto de reglas y protocolos que permite a distintas aplicaciones o sistemas interactuar entre sí. Las APIs actúan como intermediarios que permiten el intercambio de datos y funcionalidades entre diferentes aplicaciones, servicios o plataformas.