



FastChat

Benkis Eduardo Sosa Mata

César Ariel Castro Linares

Diego Emilio Serrano Domingo

Yessica Taili Jiménez Castañeda

Facultad de Ingeniería en Sistemas de la Información

Programación III, Universidad Mariano Gálvez de Guatemala sede Boca del Monte

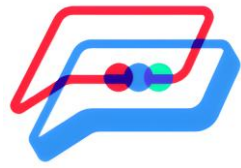
Ingeniero Abner García

Guatemala, 01 de junio de 2025



Índice

Introducción	3
Android Studio	4
¿Cómo se estructuran los proyectos en Android Studio?.....	4
Lenguaje Kotlin	7
Historia de Kotlin.....	7
Características y Ventajas de Kotlin	7
1. Interoperabilidad con código Java:.....	7
2. Curva de aprendizaje sencilla:.....	7
3. Menor tiempo de programación:.....	8
4. Orientado a objetos y programación funcional:.....	8
5. Desarrollo multiplataforma:.....	8
6. Flexibilidad:.....	8
Firebase	9
Funciones de Firebase	9
1. REALTIME DATABASE:.....	9
2. AUTENTICACIÓN DE USUARIOS:	9
3. ALMACENAMIENTO EN LA NUBE:.....	10
4. CLOUD MESSAGING:.....	10
5. HOSTING:.....	10
Ventajas del Uso de Firebase.....	10
FastChat: Diseño y Funcionalidades	12
Tour por FastChat	14
➤ Pantalla de Inicio:.....	14
➤ Pantalla principal dentro de FastChat:.....	16
➤ Opciones del perfil:	16
➤ Opción “Perfil de Usuario”:.....	17
➤ Chats en Tiempo Real:.....	19
Conclusiones	21
E-grafías	22



Introducción

El avance en la creación de aplicaciones móviles ha cambiado de manera significativa en los últimos años, no solo en los aspectos tecnológicos, sino también en las demandas de los usuarios, que buscan soluciones más fáciles de usar, seguras y que funcionen en tiempo real. Dentro de este panorama, las aplicaciones de mensajería instantánea han adquirido un rol destacado en la vida diaria, transformándose en un recurso clave para la comunicación en ámbitos personales, laborales y educativos.

En el siguiente trabajo se detalla la forma en que se crea una aplicación de mensajería instantánea, basada en el funcionamiento y las características de WhatsApp o Telegram, utilizando principalmente el lenguaje Kotlin en Android Studio, con ciertos módulos desarrollados con lenguaje Java para mejorar la comprensión del código y aprovechar bibliotecas consolidadas las cuales hacen más sencillo el desarrollo. Adicional, se comentará el uso de Firebase como gestor de base de datos, autenticación, y control de usuarios y chats e intercambio de imágenes.

En la primera parte, se describirá el entorno de desarrollo: Android Studio como IDE oficial de Google, configurado para trabajar con Kotlin como lenguaje principal, también se explicará la manera de usar la aplicación; desde el login hasta el chat entre usuarios, y nuestro sistema de base de datos. En conjunto, este trabajo no solo demuestra la capacidad de construir una aplicación de mensajería completa, sino que también resalta la documentación necesaria para los usuarios.



Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en el potente editor de código y las herramientas para desarrolladores de *IntelliJ IDEA*, Android Studio ofrece aún más funciones que mejoran tu productividad cuando compilas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Ediciones en vivo para actualizar elementos componibles en emuladores y dispositivos físicos, en tiempo real
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con C++ y NDK
- Compatibilidad integrada con *Google Cloud Platform*, que facilita la integración con *Google Cloud Messaging* y *App Engine*

¿Cómo se estructuran los proyectos en Android Studio?

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos, se incluyen los siguientes:

- Módulos de apps para Android
- Módulos de biblioteca
- Módulos de Google App Engine

De manera predeterminada, Android Studio muestra los archivos del proyecto en la vista de proyecto. Esta vista está organizada en módulos para que sea mas accesible a la información y los archivos fuentes del proyecto. Como se logra ver en la figura 1, tenemos acceso a toda la información fuente del proyecto.

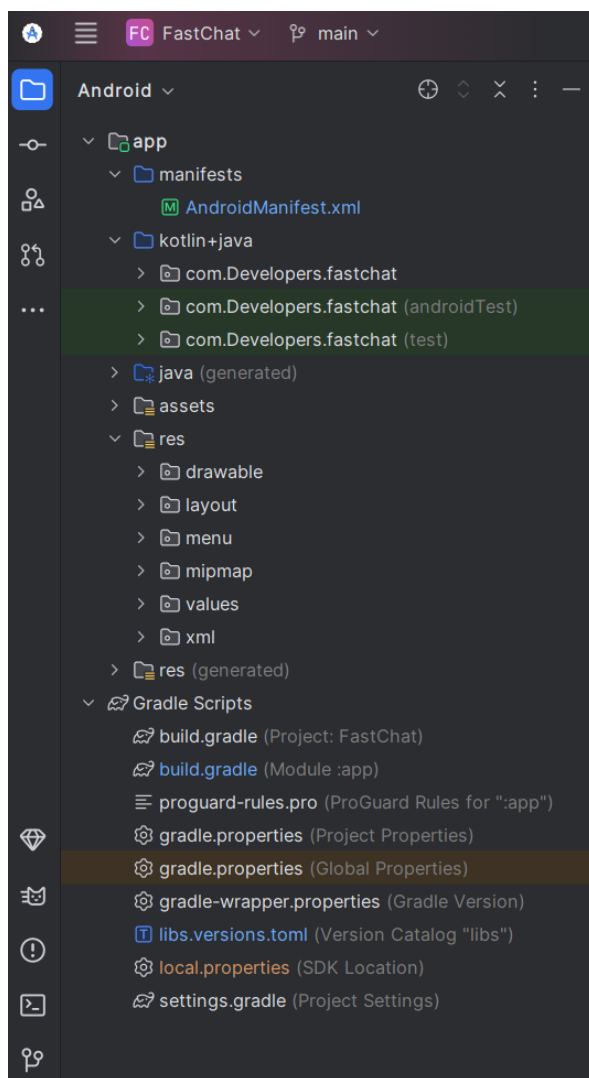


Figura 1: Archivos del proyecto en vista de proyectos.

Cada módulo de app contiene las siguientes carpetas:

1. **Manifests:** Contiene el archivo “AndroidManifest.xml”; en el se encuentra todo el mapa y reglamento de la aplicación, esto quiere decir que, gracias a esto, la aplicación sabe:
 - a. **Qué actividades tiene la app.**
 - b. **Cuál es la pantalla inicial.**
 - c. **Qué permisos necesita.**
 - d. **Qué servicios usa o usará durante la ejecución.**
 - e. **Qué versiones utilizará el proyecto.**

2. **Java + Kotlin:** Contiene los archivos de código fuente de Kotlin y Java, en otras palabras, incluye todo el código fuente de la aplicación.
3. **Res:** Contiene todos los recursos del programa que no lleva código, esto quiere decir, las cadenas de texto e imágenes o animaciones.
4. **Gradle Scripts:** Contiene toda la configuración que le dice a la app:
 - a. **Cómo compilar el proyecto.**
 - b. **Qué dependencias usará.**
 - c. **Qué versión de SDK se usa.**
 - d. **Qué módulos se están usando.**

Adicional a esto, podemos destacar que Android Studio cuenta con la posibilidad de emular distintos dispositivos para tener nuestro ambiente de pruebas para luego poder ser desplegados en dispositivos físicos. Contamos con varias versiones de sistemas operativos móviles para nuestras pruebas, y poder realizar validaciones a la hora de ser desplegado a lo físico. Como podemos ver en la figura 2, tenemos una amplia variedad de sistemas para emular.

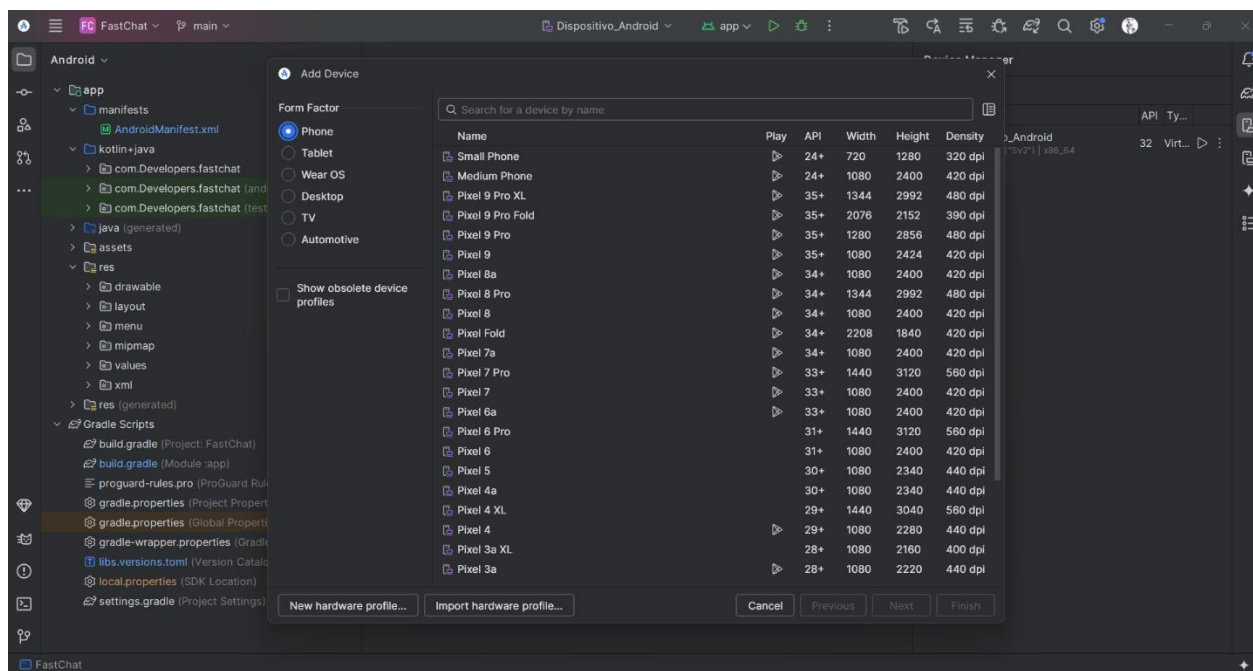
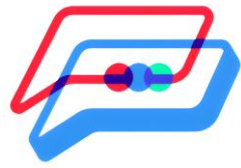


Figura 2: Se observan los distintos medios para emular directamente desde Android Studio sin necesidad de un software de terceros.



Lenguaje Kotlin

Kotlin es un lenguaje de programación moderno, pero ya consolidado, diseñado para facilitar la vida a los desarrolladores, ampliamente utilizado para los desarrolladores de Android por su facilidad de comprensión.

Historia de Kotlin

La historia de Kotlin se remonta a 2010, cuando JetBrains, la famosa empresa creadora de varios de los IDE más usados, publicó la primera versión de este lenguaje de programación. En 2012 pasó a ser de código abierto, por lo que es un sistema de programación relativamente joven, aunque se ha convertido en fundamental para comprender la evolución de las aplicaciones para dispositivos móviles. De hecho, no tuvo mucha popularidad en sus cinco primeros años de vida, y no fue hasta 2017, al anunciar Google que daría soporte a Kotlin, cuando finalmente empezó a ganar popularidad entre los desarrolladores de aplicaciones. Desde entonces, su adopción ha ido en aumento, y ha llegado a ser la opción preferida del 72 % de los desarrolladores a la hora de desarrollar para Android.

Características y Ventajas de Kotlin

Kotlin destaca por las ventajas que tiene respecto a Java a la hora de desarrollar aplicaciones móviles, además de por presentar características como simplificar la lectura del código y el propio desarrollo de este. Estas son algunas de las ventajas de usar Kotlin:

1. Interoperabilidad con código Java:

Una de las características principales de Kotlin es que está diseñado para interoperar completamente con la sintaxis del lenguaje de Java. Es decir, con una base de código existente escrita en Java, puede interactuar correctamente con Kotlin y viceversa.

2. Curva de aprendizaje sencilla:

La sencillez de la sintaxis permite una curva de aprendizaje fluida, intuitiva y fácil de usar, perfecta para los que quieran aprender su primer lenguaje de programación. Además, como es de código



abierto, hay un gran apoyo de la comunidad de Kotlin, lo que supone una gran ventaja a la hora de estar empezando a desarrollar.

3. Menor tiempo de programación:

Uno de los puntos fuertes de Kotlin es que elimina el código redundante, además de ser compacto y conciso, lo que optimiza mucho el proceso de escritura de código y evita la repetición.

4. Orientado a objetos y programación funcional:

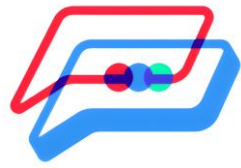
Kotlin demuestra que también se puede trabajar de la mano de la programación funcional. La posibilidad de trabajar con lambdas en este entorno simplifica las tareas más comunes y tediosas en el desarrollo.

5. Desarrollo multiplataforma:

Kotlin se puede utilizar para cualquier tipo de desarrollo, desde la web del lado del servidor y del lado del cliente, hasta Android e iOS. Como el lenguaje se ejecuta en JVM, permite compartir código entre diferentes plataformas.

6. Flexibilidad:

Kotlin da a los desarrolladores libertad de trabajar con el estilo que elijan. Por tanto, es un lenguaje altamente flexible que tiene construcciones funcionales y orientadas a objetos. Todo ello se traduce en una mejor experiencia a la hora de programar.



Firestore

Firestore de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. El uso de Firestore para el desarrollo de nuestra aplicación fue de gran ayuda, fue utilizada principalmente como base de datos en tiempo real, pero al indagar más en la plataforma, podemos aprovechar su servicio de autenticación de datos con Google, su manejo de archivos y la gran facilidad para poder gestionar la base de datos en tiempo real.

Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma. Las finalidades de las mismas se pueden dividir en cuatro grupos: desarrollo, crecimiento, monetización y análisis. Es especialmente interesante para que los desarrolladores no necesiten dedicarle tanto tiempo al *backend*, tanto en cuestiones de desarrollo como de mantenimiento.

Funciones de Firestore

Incluye los servicios necesarios para el desarrollo de un proyecto de aplicación móvil o web. Estos contribuyen a que el proceso sea más rápido, puesto que se dejan determinadas actividades a mano de Firestore, mientras que otras permiten optimizar diversos aspectos para conseguir la calidad deseada.

1. REALTIME DATABASE:

Una de las herramientas más destacadas y esenciales de Firestore son las bases de datos en tiempo real. Estas se alojan en la nube, son No SQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real, manteniéndolos actualizados, aunque el usuario no realice ninguna acción. Firestore envía automáticamente eventos a las aplicaciones cuando los datos cambian, almacenando los datos nuevos en el disco. Aunque no hubiera conexión por parte de un usuario, sus datos estarían disponibles para el resto y los cambios realizados se sincronizarían una vez restablecida la conexión.

2. AUTENTICACIÓN DE USUARIOS:

Firestore ofrece un sistema de autenticación que permite tanto el registro propiamente dicho (mediante email y contraseña) como el acceso utilizando perfiles de otras plataformas externas (por ejemplo, de



Facebook, Google o Twitter), una alternativa muy cómoda para usuarios reacios a completar el proceso.

3. ALMACENAMIENTO EN LA NUBE:

Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar los ficheros de sus aplicaciones. Al igual que la mayoría de herramientas de Firebase, es personalizable mediante determinadas reglas.

4. CLOUD MESSAGING:

Su utilidad es el envío de notificaciones y mensajes a diversos usuarios en tiempo real y a través de varias plataformas.

5. HOSTING:

Firebase también ofrece un servidor para alojar las apps de manera rápida y sencilla, esto es, un hosting estático y seguro. Proporciona certificados de seguridad SSL y HTTP2 de forma automática y gratuita para cada dominio, reafirmando la seguridad en la navegación.

Ventajas del Uso de Firebase

Como se puede concluir a partir de las funcionalidades ofrecidas, esta herramienta presenta numerosos beneficios para los desarrolladores que lo utilicen. Aunque son muchos más, algunos de ellos se recopilan brevemente.

- Muy recomendable para aplicaciones que necesiten compartir datos en tiempo real.
- Sus funcionalidades, además de ser variadas, se complementan muy bien y se pueden gestionar de forma sencilla desde un único panel. Además, no es necesario usar todas estas opciones para la aplicación, pudiendo elegir solo aquellas que más nos interesen.
- Facilita el envío de notificaciones: son muy sencillas de implementar y gestionar, además de ser extremadamente útiles para mantener la atención de los usuarios.
- Permite la monetización: desde el propio Firebase se puede agregar publicidad a la app.
- Engloba Analytics: especializado en determinadas métricas de aplicaciones móviles e integrado en el panel central de Firebase con un funcionamiento muy intuitivo. Esencial para tomar decisiones en distintas fases del proceso.

- Google ofrece numerosos documentos y tutoriales a modo introductorio e informativo (con gran profundidad) para que sumergirse en Firebase sea mucho más fácil.
- Soporte gratuito vía email, sin importar si el desarrollador utiliza la versión gratuita o de pago.
- Escalabilidad: los inicios son gratuitos, pero permite ir adaptándose a las necesidades de la aplicación con diferentes planes de pago.
- Ofrece seguridad al usuario: con los certificados SSL.
- Permite a los desarrolladores restarle atención al backend y a las infraestructuras complejas para centrarse completamente en otros aspectos.

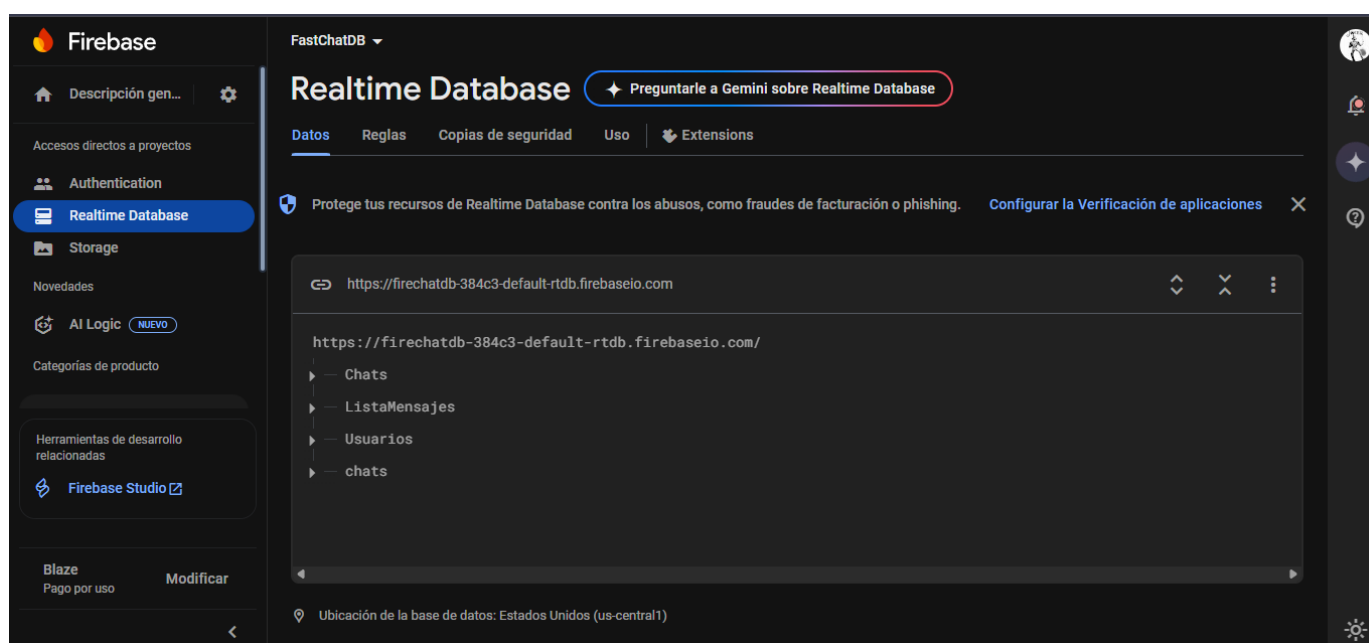
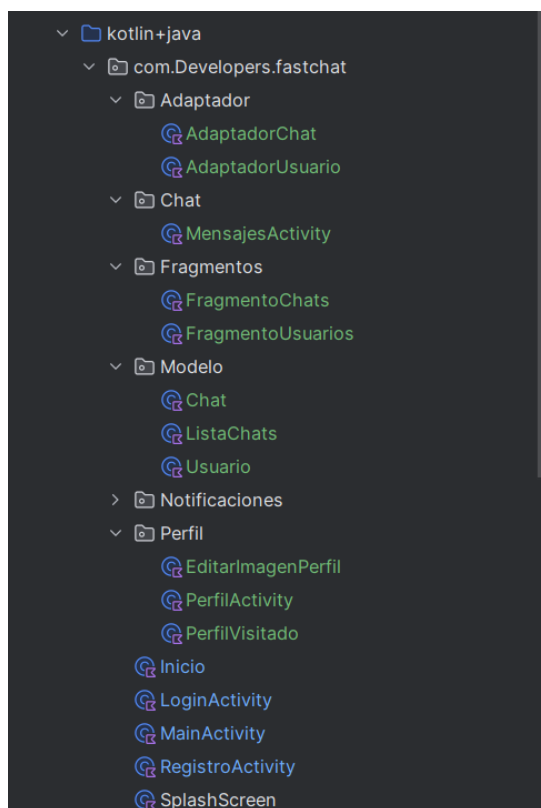


Figura 3: Se adjunta evidencia de la base de datos real de la aplicación FastChat.

FastChat: Diseño y Funcionalidades

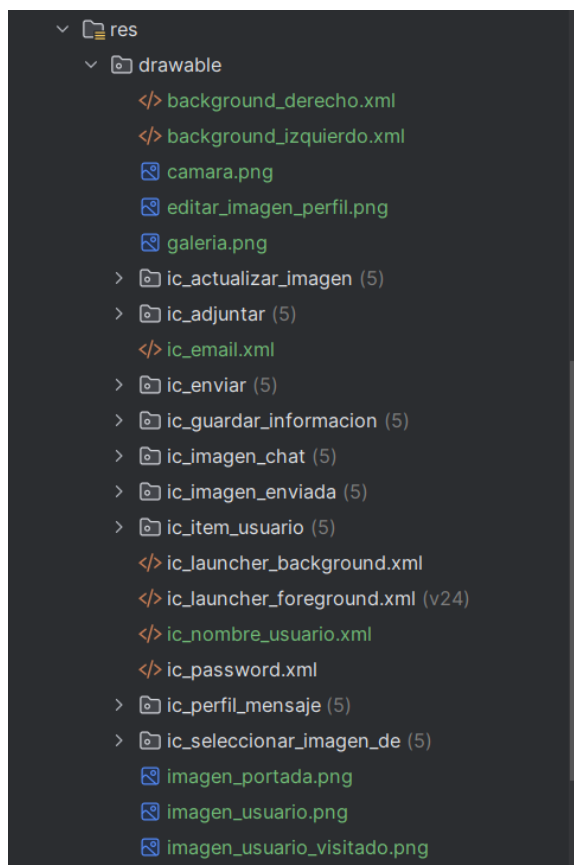
En esta sección se verá el uso de nuestra aplicación junto con el diseño utilizado y otro tipo de datos puntuales que se destacan de nuestra aplicación.

De primera mano, tenemos el apartado de toda la parte del desarrollo del código de nuestra aplicación organizado en tipo “carpetas” para un mejor entendimiento y mejor organización, esto con el fin de que, si en algún momento hay errores de compilación, podremos saber en qué parte del proyecto está originando el error:

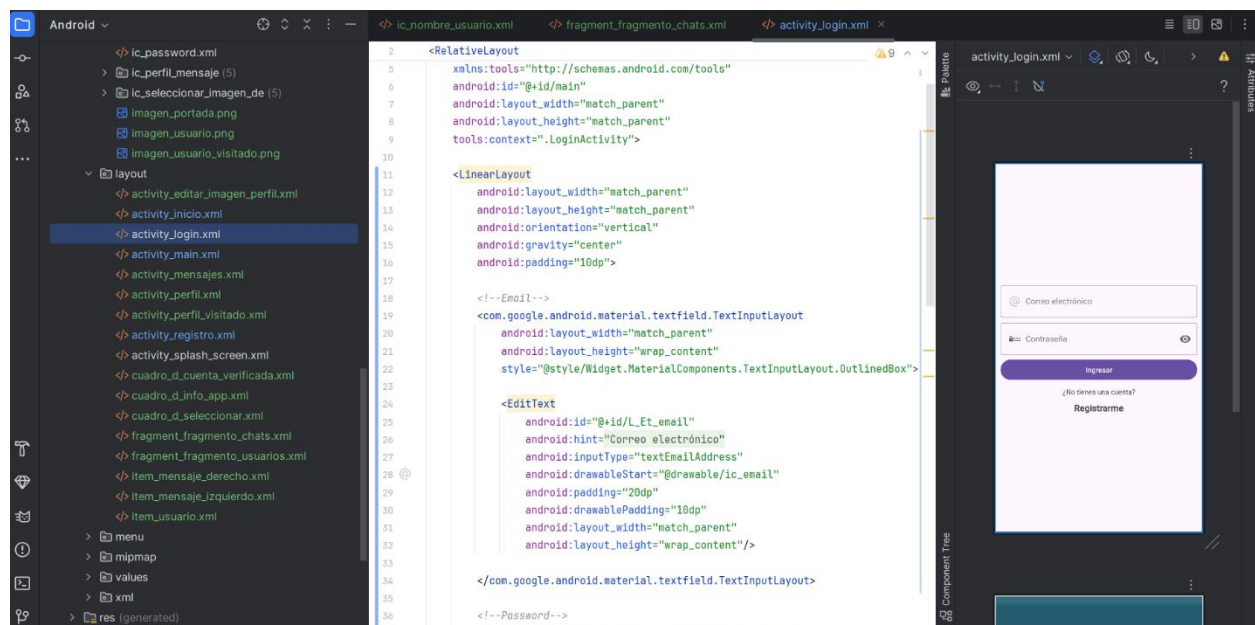


Cada carpeta está nombrada por algo específico que hará esa porción de código, o está ligado a algo puntual de la aplicación.

También contamos con la sección donde se agregan las imágenes a usar en la aplicación y los archivos xml que forman parte del diseño de la app:



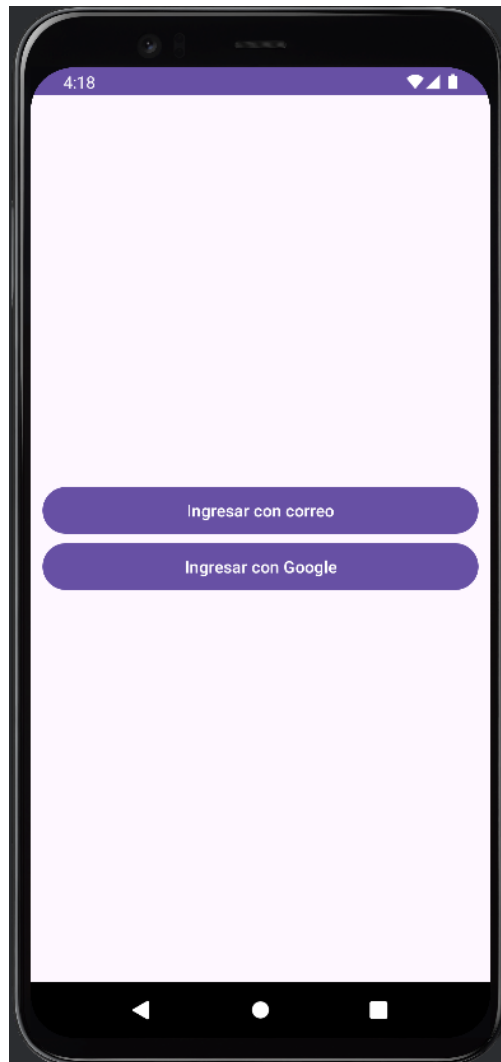
Se logran ver las imágenes usadas y los diseños de las diferentes pantallas que se verán durante la ejecución de la aplicación. Los archivos .xml son las pantallas de diseño, esto quiere decir que es como la plantilla, dentro de esa plantilla se puede modificar por medio de código para agregar o quitar información para darle forma a la pantalla que se mostrará.



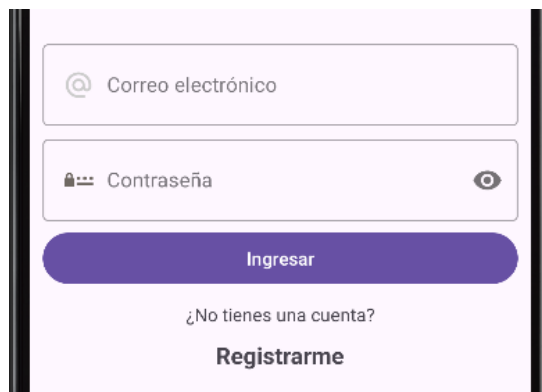
Tour por FastChat

➤ Pantalla de Inicio:

Se muestra la pantalla principal de la aplicación donde se ven las dos opciones de inicio de sesión.



Podemos ingresar por medio de un correo electrónico y desde ya podemos chatear con ese correo, si deseamos verificar la cuenta, debe de ser una cuenta con un dominio existente para que se envíe el correo de verificación y se valide la cuenta por medio de un mensaje de verificación.



Correo electrónico

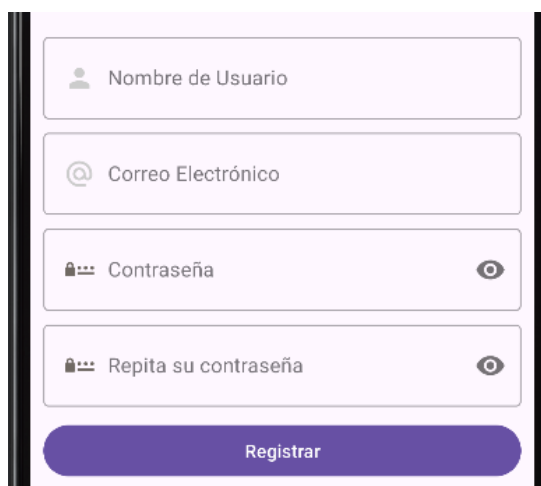
Contraseña

Ingresar

¿No tienes una cuenta?

Registrarme

Dentro de esta opción, podemos registrarnos con una cuenta de Email para poder empezar a chatear dentro de la aplicación, sin estar registrado no puedes entrar a chatear.



Nombre de Usuario

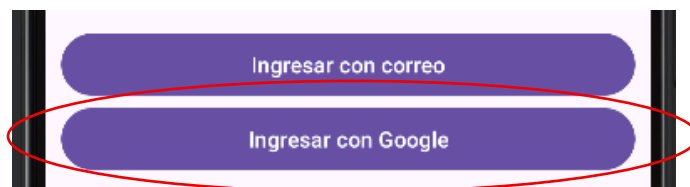
Correo Electrónico

Contraseña

Repita su contraseña

Registrar

Dentro del apartado de “Registrarme” tenemos esta pantalla de datos, estos datos serán usados para la creación de tu perfil de chat, estos datos automáticamente van a ser registrados en nuestra base de datos en Firebase para poder tener acceso a la aplicación.

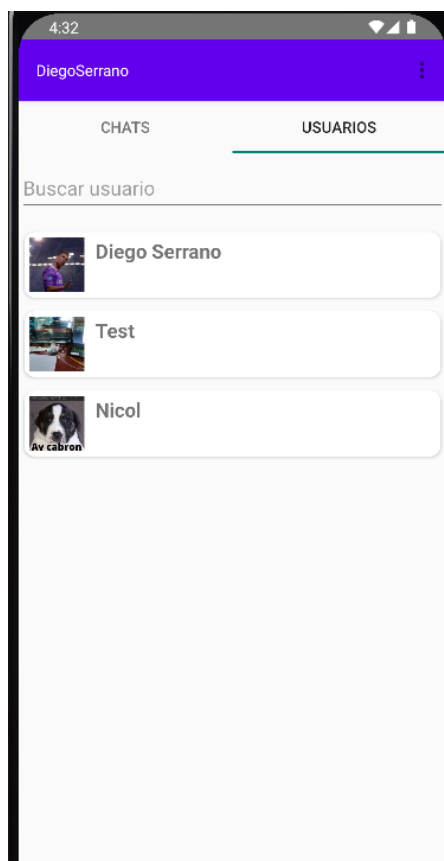


Ingresar con correo

Ingresar con Google

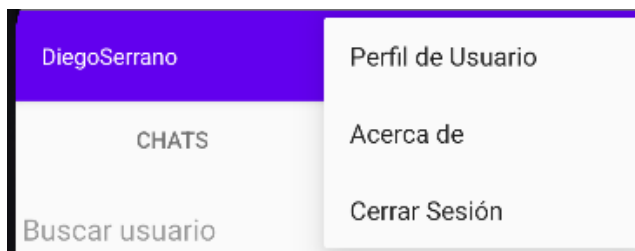
Si utilizamos la opción de ingresar con Google, la aplicación cuenta con una función de Firebase que puede autenticar directamente con las cuentas de Google que se tengan guardadas en el dispositivo, esto es mucho mas sencillo a la hora de ingresar ya que nos saltamos la parte del registro.

➤ **Pantalla principal dentro de FastChat:**



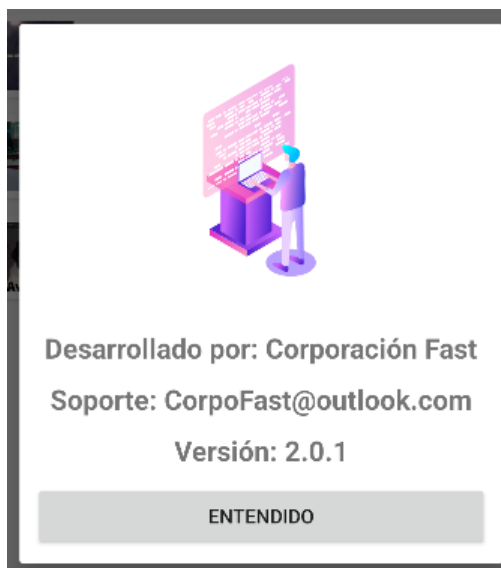
Dentro de la app, contamos con acceso a dos listas, una que se llama “Chats” en donde se alojan nuestros chats recientes con los demás contactos, y el apartado de “Usuarios” en donde aparecen todas las personas que han iniciado sesión en la app.

➤ **Opciones del perfil:**



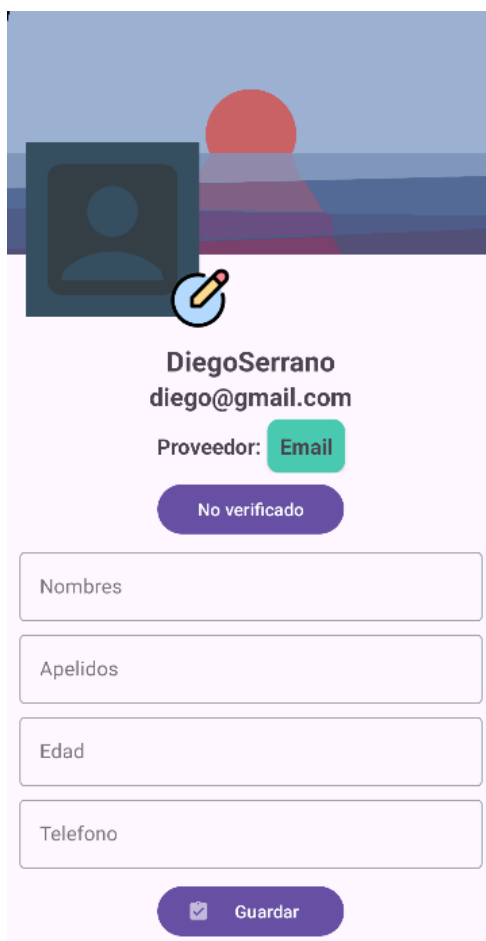
Dentro de la app, contamos con un menú de opciones, estas van puntualmente enfocadas con el usuario, la opción de “Cerrar Sesión” nos expulsa de la sesión abierta para poder ingresar con otro correo e ingresar a otro perfil, o bien, iniciar de nuevo con el mismo correo, esto con el fin de tener privacidad en nuestra app.

El apartado de “Acerca de” nos brinda una animación con datos de la app, se adjunta el ejemplo:



➤ **Opción “Perfil de Usuario”:**

En esta sección de la app podremos personalizar nuestro perfil de usuario:



En este apartado podemos editar la información del usuario, podemos agregar datos personales para que los otros usuarios puedan ver esa información. También es posible agregar una foto de perfil que será visible para todos los usuarios, contamos con los permisos en los dispositivos para poder agregar imágenes desde galería o bien, tomar una foto con el dispositivo en donde se encuentre la app. Se agrega un ejemplo de cómo se ve un perfil editado y configurado:

Perfil de Usuario



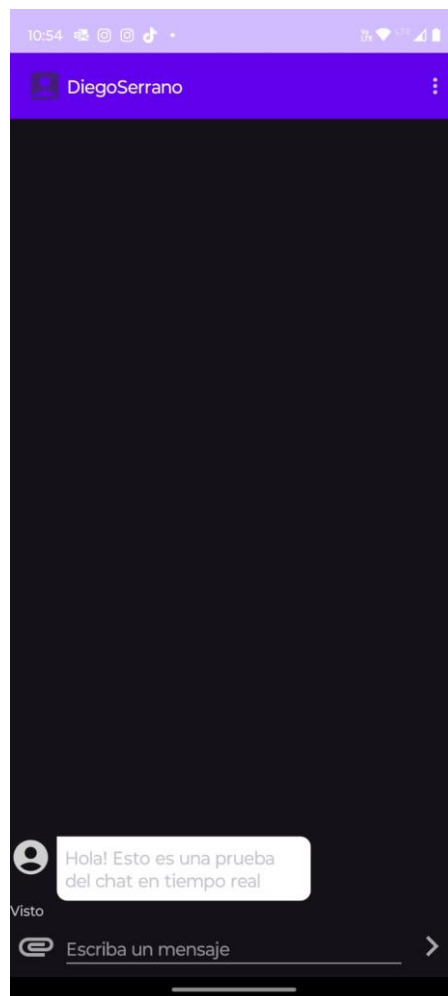
Diego Serrano
serrano.diego1607@gmail.com
Identificador:
nI8jy7NZFygiNjVyTKKgCL5xBdl1

Nombres: Diego Emilio
Apellidos: Serrano Domingo
Teléfono: 59511352
Proveedor: Google

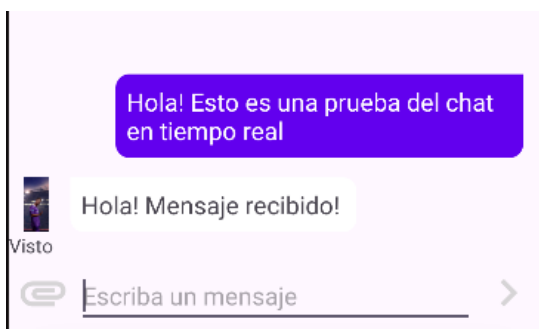
De esta manera se visualiza el perfil visto desde las otras cuentas, nos permite ver la foto de perfil, el nombre de usuario, el ID (esto sirve de referencia para nuestra base de datos), los nombres y apellidos del usuario (si los desea colocar) esto también aplica para el número de teléfono y el proveedor del correo.

➤ Chats en Tiempo Real:

Acá veremos el chat en tiempo real de la aplicación utilizando un dispositivo físico y el emulador de Android Studio:

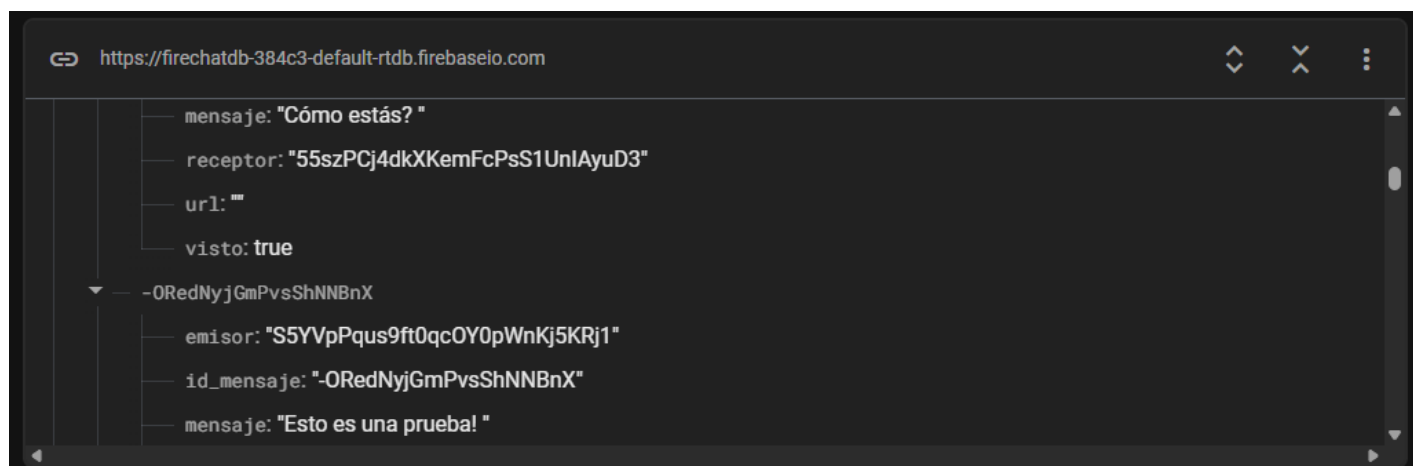


Del lado izquierdo contamos con el emulador de Android Studio enviando el mensaje, y del lado derecho, al dispositivo físico, como se puede ver, contamos con la validación del mensaje “visto” para cuando el receptor abra el mensaje del emisor.



Todos estos mensajes se verán almacenados en nuestra base de datos en tiempo real de Firebase, ya que los podemos almacenar para tener un mejor control de la información de cada chat que se realice en la app.

Firebase nos da la posibilidad de guardar las conversaciones como método de seguridad para nuestra app y para nuestros usuarios, los guarda por medio de ID de emisor y el ID del receptor y el mensaje.



Conclusiones

El desarrollo de una aplicación de mensajería instantánea similar a WhatsApp utilizando Android Studio, el lenguaje de programación Kotlin y la plataforma Firebase representa una solución sólida, moderna y eficiente para la construcción de sistemas de comunicación en tiempo real.

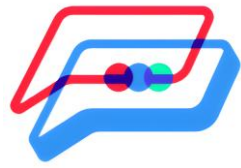
Durante el proceso de desarrollo, se pudo comprobar que Kotlin proporciona un entorno de trabajo moderno, seguro y expresivo. Su sintaxis clara y sus características orientadas a la reducción de errores, facilitan la escritura de código limpio y mantenible. Asimismo, su total compatibilidad con Android Studio y con las bibliotecas oficiales de Android lo convierten en una elección ideal para el desarrollo de aplicaciones móviles nativas.

Por otro lado, Firebase, como plataforma de backend, permitió implementar funcionalidades clave sin necesidad de gestionar un servidor propio. Su sistema de autenticación simplificó la gestión de usuarios, permitiendo el registro e inicio de sesión mediante correo electrónico, Google y otras plataformas de manera segura y confiable. Además, gracias a Firestore (o Realtime Database), se logró una comunicación fluida y sincronizada entre los usuarios, con actualizaciones instantáneas de los mensajes, simulando la experiencia de mensajería de aplicaciones comerciales ampliamente utilizadas.

El uso de Firebase Cloud Storage permitió el envío de archivos multimedia como imágenes, mientras que las reglas de seguridad definidas en Firebase ofrecieron un nivel adecuado de protección sobre los datos almacenados, asegurando que solo los usuarios autorizados puedan leer o escribir información sensible.

En términos generales, el proyecto demostró que es posible construir una aplicación funcional, robusta y escalable con herramientas y tecnologías accesibles. Esta arquitectura resulta especialmente útil para proyectos académicos.

Aun con estas consideraciones, el resultado final fue una aplicación funcional que permite enviar y recibir mensajes en tiempo real, gestionar usuarios de forma segura y mantener una experiencia fluida y moderna para el usuario final. En resumen, se concluye que la combinación de Kotlin, Android Studio y Firebase ofrece un ecosistema poderoso y práctico para el desarrollo de aplicaciones móviles de mensajería, y representa una base sólida para proyectos más avanzados o comerciales en el futuro.



E-grafías

Introducción a Android Studio. (n.d.). Android Developers. Retrieved June 9, 2025, from <https://developer.android.com/studio/intro?hl=es-419>

(N.d.). Plainconcepts.com. Retrieved June 9, 2025, from <https://www.plainconcepts.com/es/kotlin-android/>

Kotlin. (n.d.). Kotlin. Retrieved June 9, 2025, from <https://kotlinlang.org/>

Mora, S. L. (2020, May 17). Firebase: qué es, para qué sirve, funcionalidades y ventajas. DIGITAL55. <https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/>