



Desarrollo de Software para Móviles

Docente

Alexander Alberto Sigüenza Campos

Integrantes

Nestor Javier Artiga Larin	AL150682
Enma Sofia López Rojas	LR230079
Kevin Enrique Martínez Martínez	MM230084
José Gerardo Marroquín Vásquez	MV230090
Alba Vanessa Urrutia Cruz	UC151875

Link de repositorio: <https://github.com/KevinEnriqueMartinezMartinez/LoginDSM.git>

link de video: https://www.youtube.com/watch?v=x7_3oManGNk

DSM941 G01T

San salvador 24 noviembre de 2024

CONTENIDO	
INTRODUCCION	i
OBJETIVOS	ii
CONTENIDO	1
BENEFICIOS DE USAR KOTLIN CON FIREBASE.....	1
OPCIONES DE AUTENTICACIÓN	2
AUTENTIFICACION POR CORREO Y CONTRASEÑA	2
AUTENTICACIÓN POR CORREO	2
AUTENTICACIÓN POR CONTRASEÑA	4
PROVEEDORES EXTERNOS	6
AUTENTICACIÓN CON GOOGLE	7
AUTENTICACIÓN CON REDES SOCIALES	9
AUTENTICACIÓN TELEFÓNICA.....	10
AUTENTICACIÓN ANÓNIMA.....	12
AUTENTICA CON FIREBASE MEDIANTE UN VÍNCULO DE CORREO ELECTRÓNICO EN ANDROID	13
DESARROLLO DE PANTALLA LOGIN	14
IMPLEMENTACION EN EL PROYECTO.....	19
CONCLUSION	24
BIBLIOGRAFÍA	25

INTRODUCCION

En la creación de aplicaciones móviles modernas, uno de los elementos clave es asegurar un acceso rápido y seguro a los servicios backend. Firebase, la plataforma de desarrollo de aplicaciones móviles y web de Google, se ha consolidado como una de las soluciones más populares para incorporar funcionalidades como autenticación de usuarios, bases de datos en tiempo real, almacenamiento de datos y notificaciones push. Al integrar Firebase en una app móvil, no solo se optimiza la experiencia del usuario, sino que también se facilita la gestión de la seguridad y la capacidad de escalar la aplicación de forma eficiente.

Por otro lado, las aplicaciones móviles actuales suelen necesitar la integración con servicios externos a través de APIs (Interfaz de Programación de Aplicaciones) para ofrecer características adicionales, como la consulta de datos desde un servidor remoto, realizar pagos en línea o conectar con otras plataformas. Kotlin, el lenguaje recomendado para desarrollar aplicaciones Android, simplifica el proceso de interactuar con estas APIs mediante bibliotecas y métodos avanzados que permiten una comunicación eficiente y segura con los servidores.

Este artículo analiza cómo implementar un sistema de autenticación en una aplicación móvil utilizando Firebase, combinándolo con el consumo de APIs externas a través de Kotlin. Veremos cómo autenticar a los usuarios con Firebase, gestionar sus sesiones y realizar peticiones HTTP a servicios externos para obtener datos. Con este enfoque, se pretende ofrecer una base robusta para desarrollar aplicaciones móviles seguras, interactivas y capaces de interactuar de manera fluida con otros servicios.

OBJETIVOS

GENERAL

Investigar sobre Firebase y como aplicarla a un pequeño proyecto de ingreso a una plataforma, (implementación de un login con firebase).

Específicos:

- ❖ Conocer los tipos de autenticación con Firebase.
- ❖ Identificar las ventajas y desventajas de los tipos de autenticación con Firebase.
- ❖ Desarrollar un login con autenticación por correo electrónico y cuenta de Google con Firebase

CONTENIDO

Dentro del desarrollo de aplicaciones se encuentran una gran variedad de herramientas como lo son los gestores de bases de datos, es necesario conocer e implementar estas herramientas en nuestros desarrollos, para hacer más eficientes su funcionamiento.

Conocer de los distintos gestores de bases de datos es indispensable en el desarrollo móvil y más si estas tienen sincronización en tiempo real y una excelente escalabilidad horizontal lo que permite el manejo de grandes cantidades de datos y adicional a esto saber con qué lenguaje se hará. (Developers, s.f.)

Firebase será el gestor de bases de datos de que nos ayudará con estas exigencias y ventajas con la diversidad de tipo de autenticación con la que cuenta. Como punto de partida desarrollaremos los diferentes tipos de autenticación que Firebase nos deja utilizar

BENEFICIOS DE USAR KOTLIN CON FIREBASE

Utilizando Firebase con Kotlin ofrece varios beneficios significativos: (Obregon, 2023)

- ❖ **Eficiencia:** La expresividad de Kotlin y las características ricas de Firebase pueden reducir significativamente el código de caldera, permitiéndole centrarse más en la lógica de la aplicación.
- ❖ **Seguridad:** La función de seguridad de Kotlin-s no impide errores de programación comunes. Firebase también proporciona múltiples capas de seguridad, como la autenticación segura y las reglas de base de datos.
- ❖ **Escalabilidad:** Firestore-s poderosas consultas y capacidades en tiempo real, junto con Firebase suite de otras características, hacen que sea más fácil construir aplicaciones de Android escalables.

OPCIONES DE AUTENTICACIÓN

La mayoría de las aplicaciones necesitan conocer la identidad de un usuario. Conocer la identidad de un usuario permite que una aplicación guarde de forma segura los datos del usuario en la nube y brinde la misma experiencia personalizada en todos los dispositivos del usuario.

Firebase Authentication proporciona servicios backend, SDK fáciles de usar y bibliotecas de UI listas para usar para autenticar a los usuarios en su aplicación. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidades federados populares como Google, Facebook y Twitter, y más.



Figura 1: Firebase y proveedores de identidad federados
Tomado de sitio oficial de Firebase

Se destaca como dato importante que para poder utilizar cualquier tipo de autenticación en un proyecto con Firebase es necesario el uso de dependencias pero sobre todo una en especial: “implementación “com.google.firebase:firebase-auth-ktx”.

AUTENTIFICACION POR CORREO Y CONTRASEÑA

Dentro de este apartado englobaremos dos tipos de autenticación, los cuales serán autenticación de contraseña y enlaces de correos electrónicos, en el desarrollo de ambos es bastante similar

AUTENTICACIÓN POR CORREO

Todas y cada una de las autenticaciones de Firebase se podrán utilizar siempre que se cuente con una cuenta dentro de esta plataforma de Firebase caso contrario no se podrá utilizar esta plataforma ni sus autenticaciones, más adelante se mostrará como poder acceder a cada autenticación que se explicara a continuación.

Firebase Autenticación permitir que los usuarios autentifiquen sus direcciones de correo electrónico y contraseñas, y para administrar las cuentas basadas en contraseñas de tu aplicación. (FIREBASE , s.f.)

Para hacer uso de esta autenticación se deben de seguir una serie de pasos para agregarlo al proyecto en el que se ejecutaran:

1. Iniciar sesión en Firebase
2. Abrir la consola de Firebase
3. Dentro de la consola seleccionar Auth
 - 3.1 Abrir la sección de Auth
 - 3.2 Seleccionar en **Método de Inicio de sesión**, habilite el método de inicio de sesión **por correo electrónico/contraseña** y haga clic en **Guardar**
4. En el **archivo Gradle de su módulo (nivel de aplicación)** (generalmente <project>/<app-module>/build.gradle.kts o <project>/<app-module>/build.gradle), agregue la dependencia para la autenticación de Firebase. biblioteca para Android. Recomendamos utilizar [Firebase Android BoM](#) para controlar el control de versiones de la biblioteca, como se muestra en la siguiente imagen de ejemplo:

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation(platform("com.google.firebase:firebase-bom:32.8.0"))

    // Add the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation("com.google.firebase:firebase-auth")
}
```

Diferencia de la autenticación por correo y contraseña que se hace desde la consola de Firebase, si se quiere hacer uso de la autenticación con contraseña esta configuración se hace desde el proyecto, en el cual se agregaran un **método onCreate**, dentro del registro, donde se deberá crear la instancia del objeto **Firebase Auth**, para posteriormente inicializar la actividad, así como se muestra a continuación:

<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid #ccc; padding-bottom: 5px;"> Kotlin+KTX Android Java Android </div> <pre style="margin-top: 5px;">private lateinit var auth: FirebaseAuth // ... // Initialize Firebase Auth auth = FirebaseAuth.getInstance()</pre> </div>	<div style="border: 1px solid #ccc; padding: 5px;"> <div style="display: flex; justify-content: space-between; border-bottom: 1px solid #ccc; padding-bottom: 5px;"> Kotlin+KTX Android Java Android </div> <pre style="margin-top: 5px;">public override fun onStart() { super.onStart() // Check if user is signed in (non-null) and update UI accordingly. val currentUser = auth.currentUser if (currentUser != null) { reload() } }</pre> </div>
--	---

Con esto pasos podrá iniciar sesión una vez rellene el formulario de registro de la aplicación, se validen los valores de la contraseña.

Se podrá crear una nueva cuenta pasando la dirección de un correo y la contraseña registrada del nuevo usuario con el siguiente método: **createUserWithEmailAndPassword**, como se puede observar en el siguiente ejemplo:

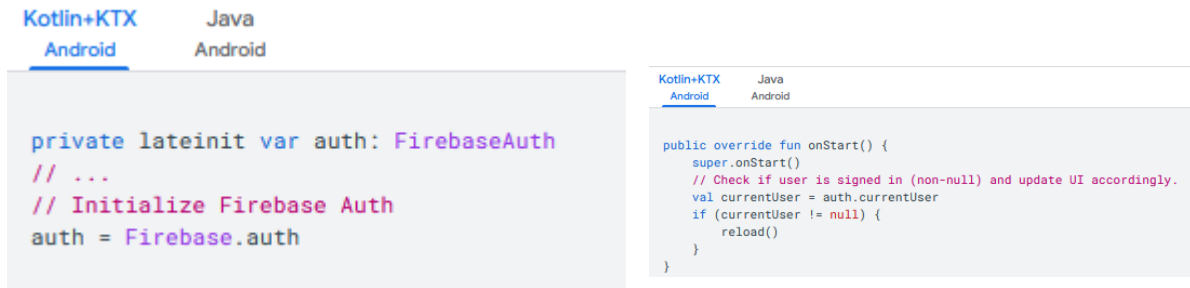
Kotlin+KTX
Android
Java
Android

```
auth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) {
            // Sign in success, update UI with the signed-in user's information
            Log.d(TAG, "createUserWithEmail:success")
            val user = auth.currentUser
            updateUI(user)
        } else {
            // If sign in fails, display a message to the user.
            Log.w(TAG, "createUserWithEmail:failure", task.exception)
            Toast.makeText(
                baseContext,
                "Authentication failed.",
                Toast.LENGTH_SHORT,
            ).show()
            updateUI(null)
        }
    }
```

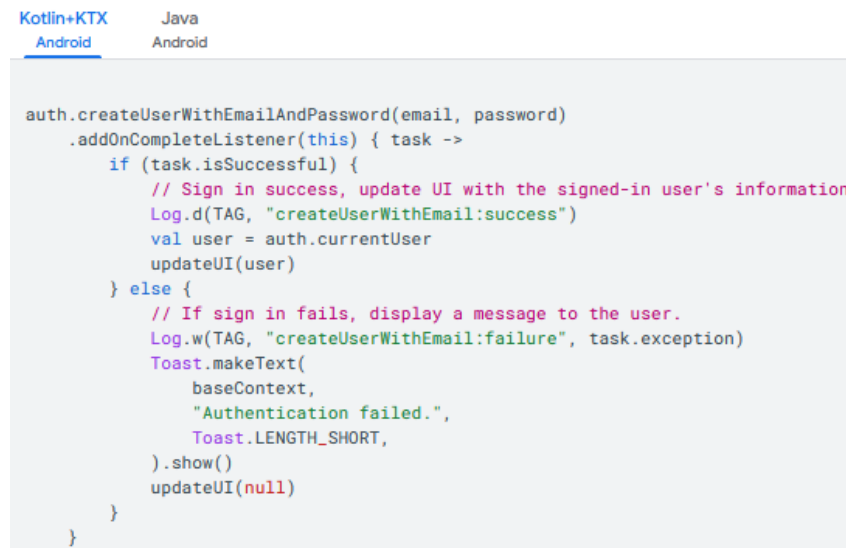
AUTENTICACIÓN POR CONTRASEÑA

Para la autenticación mediante contraseña es similar a crear una cuenta nueva, solo se le agregara unas variaciones en las líneas de código a utilizar dentro de nuestro desarrollo.

Estas variaciones se dan en el Método **onCreate**, en la clase creada para el registro donde en la instancia creada **FirebaseAuth** quedara de la siguiente manera tanto como para crearla como para inicializarla respectivamente en las imágenes que continuación se muestran :



Una vez realizado estos ajustes el usuario podrá acceder al formulario donde validara su contraseña, y se podrá crear la cuenta pasando la dirección de correo y la contraseña del nuevo usuario en **createUserWithEmailAndPassword**, el cual se hace mediante la siguientes funciones



Una vez identificado las configuraciones necesarias para la creación de usuarios mediante contraseña y correo electrónico es necesario conocer de estas autenticaciones sus ventajas y desventajas, para ello se ha creado el siguiente cuadro de detalle

Características	Ventajas	Desventajas
Seguridad	-Reduce el riesgo de reutilización de contraseñas entre aplicaciones. -Nivel básico de seguridad aceptable para proteger nuestros datos	- Las contraseñas débiles o comunes pueden ser vulnerables a ataques de fuerza bruta. - Están propensas a sufrir de Phishing.
Facilidad de uso	-Se puede autenticar y verificar el correo del usuario en un solo paso.	La experiencia puede ser menos fluida comparada con métodos como inicio de sesión social.
Recuperación de cuenta	-Los enlaces mágicos permiten iniciar sesión sin recordar contraseñas.	Requiere acceso al correo electrónico; si el usuario pierde acceso a su correo, no podrá autenticarse.
Flexibilidad	Puede actualizarse fácilmente para trabajar junto con otros métodos, como inicio de sesión social.	No siempre es intuitivo para usuarios no técnicos, quienes pueden preferir métodos biométricos.
Costo	Implementación relativamente económica, sin necesidad de dependencias de terceros.	Necesita un servidor configurado para enviar correos y gestionar datos sensibles.
Compatibilidad con móviles	Acceso sin necesidad de recordar contraseñas, práctico en dispositivos móviles.	Puede ser menos atractivo que métodos como Google o Apple Sign-In.

Tabla 1: Ventajas y desventajas de Autenticación por correo electrónico y contraseña.

Elaboración Propia

Es importante tomar en cuenta, que para este método existe la posibilidad de utilizar un vínculo, estos vínculos se usan para evitar un acceso como de un usuario no deseado o un dispositivo no deseado, Firebase Auth requiere que se proporcione la dirección de correo electrónico del usuario cuando se complete el proceso de acceso. Para que el acceso sea exitoso, esta dirección de correo electrónico debe coincidir con la dirección a la que se envió originalmente el vínculo de acceso.

PROVEEDORES EXTERNOS

Dentro de las autenticaciones que ofrece Firebase esta la de proveedores externos en la cual cuenta con los siguientes proveedores de identidades federados que se mostraran en la tabla siguiente:

Proveedores	Plataformas				
	<u>iOS</u>	<u>Android</u>	<u>Web</u>	<u>C++</u>	<u>Unidad</u>
Google	X	X	X	X	X
Iniciar sesión con Apple	X	X	X	X	X
Facebook	X	X	X	X	X
Jugos		X		X	X
Centros de Juego	X				
GitHub	X	X	X	X	X
Microsoft	X	X	X	X	X
Gorjeo	X	X	X	X	X
Yahoo!	X	X	X	X	X

Tabla 2: Proveedores Federados.
Tomado de sitio oficial de Firebase

La autenticación con estos proveedores es muy similar en cada caso por lo que se desarrollara únicamente la autenticación con Google

AUTENTICACIÓN CON GOOGLE

Dentro de las autenticaciones más comunes en diferentes plataformas se encuentra la autenticación con una cuenta de correo electrónico, en este caso la plataforma de Firebase también nos lo permite realizar mediante este método.

Es una forma fácil practica de poder acceder a diferentes plataformas a continuación mostraremos

Las configuraciones para realizar, así como algunas de las ventajas y desventajas de esta autenticación

Dentro de las configuraciones están las siguientes: En el archivo de Gradle (generalmente <project>/<app-module>/build.gradle.kts o <project>/<app-module>/build.gradle) del módulo (a nivel de

```

dependencies {
    // Import the BoM for the Firebase platform
    implementation(platform("com.google.firebase:firebase-bom:32.3.1"))

    // Add the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation("com.google.firebase:firebase-auth-ktx")

    // Also add the dependency for the Google Play services library and specify its version
    implementation("com.google.android.gms:play-services-auth:20.7.0")
}

```

app), agrega la dependencia de la biblioteca de Android para Firebase Authentication. Te recomendamos usar la [BoM de Firebase para Android](#) para controlar las versiones de las bibliotecas.

Además, como parte de la configuración de Firebase Authentication, debes agregar el SDK de Servicios de Google Play a la app.

Ventajas	Desventajas
<ul style="list-style-type: none"> - Inicio de sesión rápido. - Reduce tiempo de registro ya que facilita de implementar y administrar. La integración de Google y facebook es una opción sólida para ofrecer a los usuarios un método de inicio de sesión seguro y conveniente. - Delega la seguridad al proveedor externo. 	<ul style="list-style-type: none"> - Centralización: Si el servicio de autenticación de Google experimenta interrupciones, los usuarios podrían no poder iniciar sesión en tu aplicación. - Vinculación a Google: Los usuarios que no tengan una cuenta de Google o no quieran utilizarla quedarán excluidos. - Algunos proveedores pueden tener configuraciones un poco más complejas.
<ul style="list-style-type: none"> - Firebase es una plataforma muy diversificada para adaptarse a proveedores externos como Google - Entre sus características se encuentran: monetización, gran poder de crecimiento, desarrollo y gestión de apps multiplataforma, cloud Storage⁸⁸, cloud functions, panel central 	<ul style="list-style-type: none"> - Riesgo de cuentas comprometidas. - Punto único de falla: Si la cuenta de Google de un usuario es hackeada, todos los servicios vinculados (incluido el tuyo) estarán en riesgo. - Desconfianza del usuario: Algunos usuarios pueden preferir sistemas de autenticación separados para minimizar el impacto de un hackeo.
<p>Integración sencilla: Google ofrece bibliotecas, SDKs y documentación bien estructurada para implementar su sistema de autenticación.</p>	<ul style="list-style-type: none"> - Rechazo a unificar cuentas: Algunos usuarios pueden sentirse incómodos al usar una cuenta personal (como Google) para fines laborales o en aplicaciones desconocidas. - Falta de identidad independiente: Al usar autenticación externa, puede ser difícil para los usuarios construir una identidad única dentro de tu aplicación.

Tabla 3: Ventajas y desventajas de Autenticación con Proveedores externos.

Elaboración Propia

En resumen, la integración de Google Login con Kotlin y Firebase es una opción sólida para cualquier desarrollador de aplicaciones Android que busque una autenticación fácil de implementar y administrar. Al seguir estos pasos, tu aplicación estará lista para ofrecer a los usuarios un método de inicio de sesión seguro y conveniente, contribuyendo así al éxito general de tu proyecto.

AUTENTICACIÓN CON REDES SOCIALES

La autenticación mediante redes sociales es un flujo de autenticación de varios pasos que te permite hacer que un usuario acceda a una cuenta o vincularlo con una existente.

Las plataformas nativas y los sitios web admiten la creación de una credencial que se puede pasar a los métodos `signInWithCredential` o `linkWithCredential`. Como alternativa, en las plataformas web, puedes activar el proceso de autenticación mediante una ventana emergente o un redireccionamiento.

Ventajas	Desventajas
Acceso a datos del perfil: Con el consentimiento del usuario, puedes obtener información como nombre, correo electrónico y foto de perfil para personalizar la experiencia.	Cambios en APIs o políticas: Las plataformas pueden modificar sus APIs o políticas, lo que podría requerir ajustes técnicos en tu aplicación.
Interacción social: Las aplicaciones pueden integrar funciones sociales, como compartir contenido o conectar a los usuarios con sus amigos.	Usuarios sin redes sociales: No todos los usuarios tienen cuentas en redes sociales o quieren utilizarlas para autenticarse.
Menor carga operativa: No necesitas gestionar contraseñas ni procesos de recuperación de cuenta.	Interrupciones del servicio: Si la red social experimenta problemas, los usuarios no podrán iniciar sesión.
Fácil integración: Las APIs de autenticación de redes sociales suelen ser compatibles con estándares como OAuth 2.0, simplificando su implementación.	Restricciones geográficas: Algunas redes sociales pueden estar bloqueadas en ciertos países o regiones.
Experiencia conocida: Muchas personas están familiarizadas con el inicio de sesión a través de redes sociales, lo que reduce la fricción en el proceso.	Dependencia de una sola cuenta: Si la cuenta de red social del usuario es hackeada, su acceso a tu aplicación también estará en riesgo.
Inicio de sesión rápido: Los usuarios no necesitan crear una cuenta nueva ni recordar credenciales adicionales.	Reticencia de los usuarios: Algunos usuarios pueden desconfiar de vincular sus redes sociales a tu aplicación por preocupaciones de privacidad.
Conexión con redes sociales: Permite a los usuarios compartir contenido de tu aplicación en sus redes, aumentando tu visibilidad.	Percepción de invasión: Pedir acceso a datos adicionales (como listas de amigos o publicaciones) puede generar desconfianza.

Acceso a redes de contactos: Puedes aprovechar la conectividad social para crear experiencias más atractivas (por ejemplo, mostrando amigos que ya usan el servicio).	Integración inicial: Aunque hay documentación disponible, la implementación puede ser más compleja si deseas integrar varias redes sociales.
Acceso inmediato: Los usuarios pueden comenzar a utilizar el servicio rápidamente, mejorando la experiencia inicial.	

Tabla 4: Ventajas y desventajas de Autenticación con Redes Sociales
Elaboración Propia

AUTENTICACIÓN TELEFÓNICA

Firebase Authentication permite que un usuario acceda mediante el envío de un mensaje SMS a su teléfono. El usuario accede con un código único que se incluye en el mensaje SMS. La forma más fácil de agregar un acceso con número de teléfono a la app es usar FirebaseUI, que incluye un widget de acceso directo que implementa flujos de acceso con número de teléfono, además de acceso con redes y con contraseña.

Para usar la autenticación con número de teléfono, Firebase debe poder verificar que las solicitudes de acceso con número de teléfono provienen de la app. Firebase Authentication hace lo anterior de las siguientes maneras:

- **API de Play Integrity:** Si un usuario cuenta con un dispositivo que tiene instalados los Servicios de Google Play y Firebase Authentication puede verificarlo como legítimo con la API de Play Integrity, el acceso con número de teléfono puede continuar. Firebase Authentication habilita la API de Play Integrity en un proyecto de Google, no en el propio. Esto no contribuye a ninguna cuota de la API de Play Integrity en tu proyecto. La compatibilidad para Play Integrity está disponible a partir de la versión 21.2.0 del SDK de Authentication (versión 31.4.0 o superior de la BoM de Firebase).
- **Verificación de reCAPTCHA:** En el caso de que no sea posible usar Play Integrity (por ejemplo, cuando el dispositivo del usuario no tenga instalados los Servicios de Google Play), Firebase

Authentication utiliza una verificación de reCAPTCHA para completar el flujo del acceso con número de teléfono. El desafío de reCAPTCHA a menudo se puede completar sin que el usuario tenga que resolver nada. Es de tener en cuenta que este flujo requiere que se asocie un hash SHA1 en la aplicación. Este flujo también requiere que tu clave de API no tenga restricciones o que se incluya en la lista de entidades permitidas de `PROJECT_ID.firebaseio.com`.

Estas son algunas situaciones en las que se activa reCAPTCHA:

- Si el dispositivo del usuario final no tiene instalados los Servicios de Google Play.
- Si la app no se distribuye en Google Play Store (en el SDK de Authentication v21.2.0 o una versión posterior).
- Si el token de SafetyNet no es válido (en versiones del SDK de Authentication anteriores a la v21.2.0).

Ventajas	Desventajas
Realizan pruebas con facilidad en entornos de desarrollo sin ningún esfuerzo adicional. Por ejemplo, puedes tener la capacidad de desarrollar en un simulador de iOS o un emulador de Android sin los Servicios de Google Play.	Es menos segura que los demás métodos disponibles.
Permite configurar números de teléfono ficticios para el desarrollo con Firebase console.	Posee altas probabilidades de suplantación de identidad.

Tabla 5 : Ventajas y desventajas de Autentificación Telefónica
Elaboración Propia

Es importante tomar en cuenta la seguridad de este tipo de método, si bien la autenticación con solo un número de teléfono es conveniente, es menos segura que otros métodos disponibles, ya que la posesión de un número de teléfono o un dispositivo móvil en sí, se puede transferir con facilidad entre usuarios. Además, en los dispositivos con varios perfiles de usuario, cualquier usuario que reciba mensajes SMS puede acceder a una cuenta con el número de teléfono del dispositivo.

Si se utiliza el acceso con número de teléfono en una app, lo más sugerible es ofrecerla junto con métodos de acceso más seguros, además de informar a los usuarios acerca de las desventajas de usar el acceso con número de teléfono.

AUTENTICACIÓN ANÓNIMA

Con Firebase Authentication podemos crear y usar cuentas anónimas temporales, estas cuentas se pueden usar para permitir que los usuarios que aún no se hayan registrado en una aplicación trabajen con datos protegidos por reglas de seguridad. Firebase genera un usuario único para cada sesión anónima y le asigna un ID de usuario (UID). Esto permite almacenar datos relacionados con ese usuario, como preferencias o progresos, sin necesidad de que se registre una cuenta.

Ventajas	Desventajas
- Persistencia de datos: A pesar de ser una cuenta anónima, puedes almacenar datos en Firebase relacionados con el usuario, y más tarde vincularlos a una cuenta real sin perder la información.	- Información limitada: Sin datos del usuario (nombre, correo electrónico, etc.), es difícil ofrecer una experiencia personalizada. (Falta de personalización).
- Fluidez en la experiencia de usuario: Permite a los usuarios interactuar con la app sin fricción ni interrupciones, eliminando la necesidad de un registro inicial.	- Sin historial persistente : Las aplicaciones no pueden guardar el progreso o las preferencias del usuario si no se vincula a una identidad.
- Flexibilidad: La autenticación anónima se puede combinar con otros métodos de autenticación para ofrecer una experiencia flexible y adaptable al usuario.	- Facilita comportamientos malintencionados : Los usuarios maliciosos pueden aprovechar el anonimato para realizar actividades inapropiadas o ilegales sin temor a ser identificados.
Los usuarios pueden no sentirse motivados a interactuar de forma consistente si no tienen un perfil asociado. (Menor retención de usuarios)	

Tabla 6: Ventajas y desventajas de Autenticación Anónima

Elaboración Propia

utilizar autenticación anónima en una aplicación permite ofrecer una experiencia inicial sin fricción, y sin comprometer las funcionalidades clave. Con Firebase, la simplicidad y el poder van de la mano, y esta funcionalidad es un ejemplo perfecto de ello.

AUTENTICA CON FIREBASE MEDIANTE UN VÍNCULO DE CORREO ELECTRÓNICO EN ANDROID

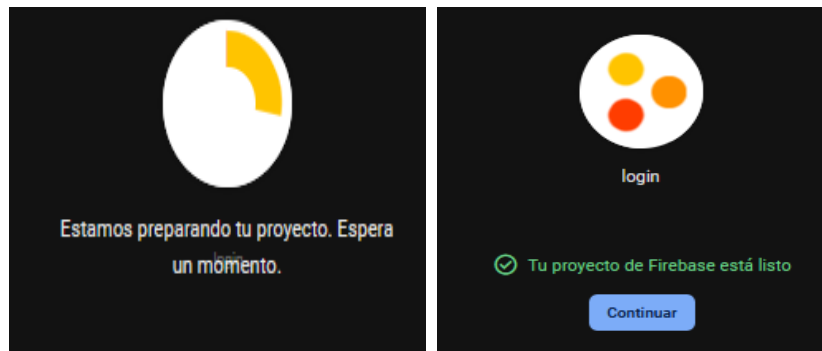
La autentificación de Firebase permite a los usuarios iniciar sesión mediante un correo electrónico con un vínculo tan fácil como dar un clic, sobre el verificando también la dirección. Se configuran dependencias y ajustes en Firebase Console y se implementa mediante `sendSignInLinkToEmail`. El flujo incluye generar un `ActionCodeSettings`, enviar el vínculo y completar el acceso verificando el correo electrónico.

Se utiliza Firebase Dynamic Links para redireccionar a la app móvil. Es posible vincular o reautenticar usuarios y diferenciar entre métodos de acceso como contraseña o vínculo. La configuración debe garantizar seguridad y compatibilidad en Android.

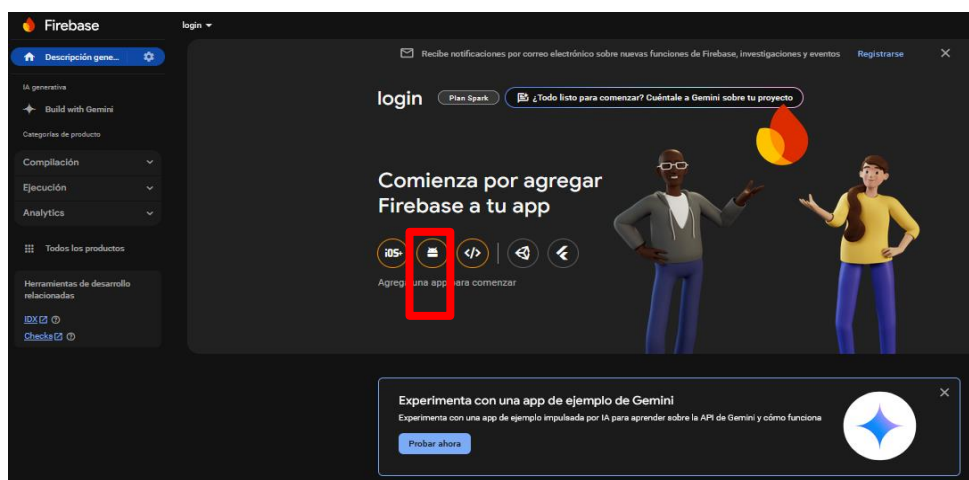
Aspecto	Ventajas	Desventajas
Seguridad	<ul style="list-style-type: none"> - Reduce riesgos asociados con contraseñas. - Verifica la propiedad del correo electrónico. 	<ul style="list-style-type: none"> - Vulnerable si el correo es interceptado. - Requiere HTTPS para máxima seguridad.
Usabilidad	<ul style="list-style-type: none"> - Simplifica el acceso al eliminar contraseñas. - Ideal para dispositivos móviles. 	<ul style="list-style-type: none"> - Puede resultar confuso si el usuario no abre el vínculo en el dispositivo correcto.
Recuperación	<ul style="list-style-type: none"> - Permite acceso sin restablecer contraseñas olvidadas. 	<ul style="list-style-type: none"> - El usuario necesita acceso continuo al correo electrónico.
Flexibilidad	<ul style="list-style-type: none"> - Compatible con actualizaciones de métodos de acceso (por ejemplo, cambiar de contraseña a vínculo). 	<ul style="list-style-type: none"> - Dependencia de Firebase Dynamic Links, lo que puede requerir configuraciones adicionales.
Costo	<ul style="list-style-type: none"> - Uso gratuito dentro de los límites de Firebase. 	<ul style="list-style-type: none"> - Puede requerir infraestructura adicional (como correos transaccionales si el proyecto escala).
Compatibilidad	<ul style="list-style-type: none"> - Funciona bien en Android y otros dispositivos móviles con Dynamic Links. 	<ul style="list-style-type: none"> - Requiere configuración específica para cada plataforma.
Otros aspectos	<ul style="list-style-type: none"> - Compatible con usuarios que ya tienen cuentas en otros métodos. - Mejora la experiencia del usuario al evitar pasos adicionales como crear contraseñas. 	<ul style="list-style-type: none"> - Depende de la correcta configuración de dominios y paquetes en Firebase Console.

DESARROLLO DE PANTALLA LOGIN

Una vez terminado esta acción se procederá a establecer una pantalla donde se nos muestra que se está cargando nuestro inicio de nuestra aplicación

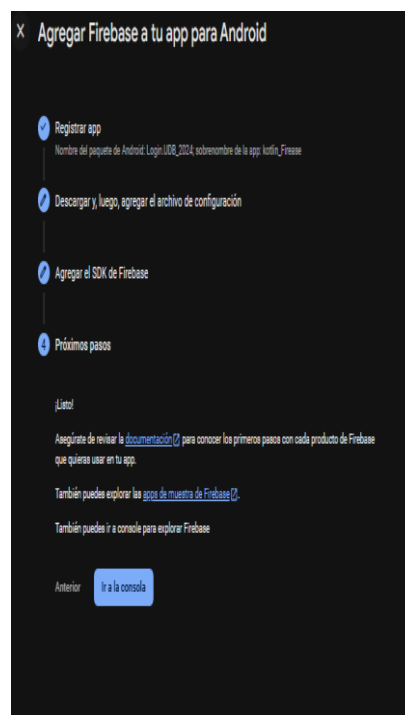
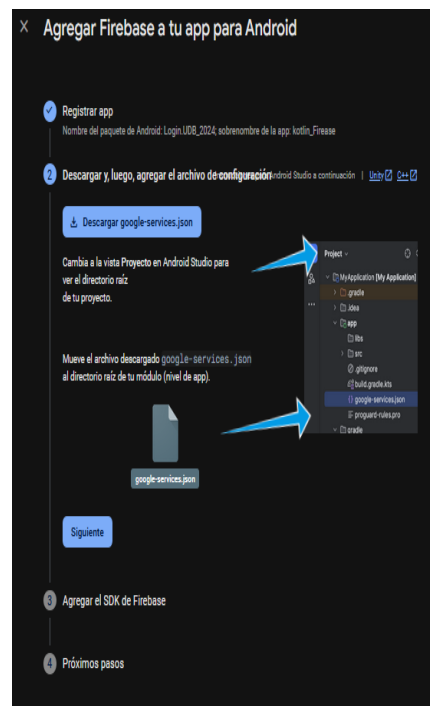


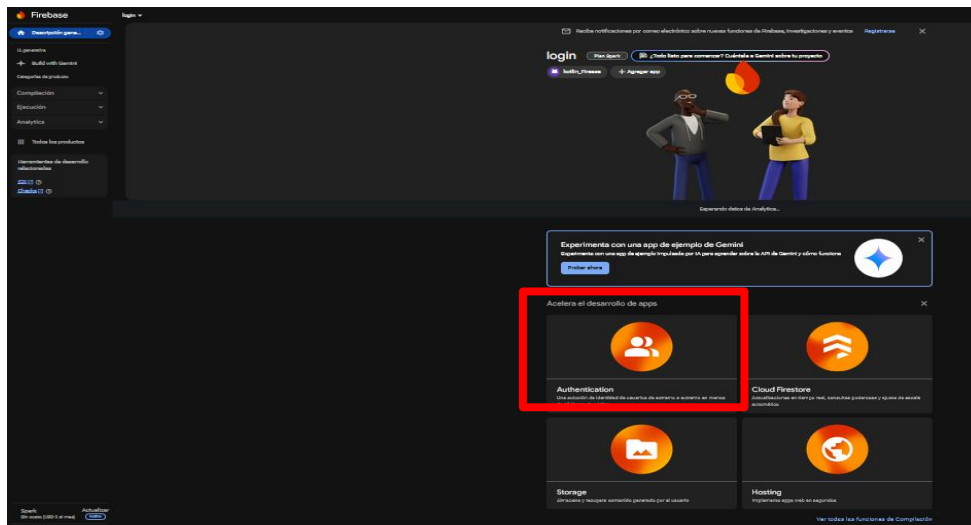
Una vez adentro procederemos a seleccionar la plataforma para la cual se creará un login en este caso será para Android por lo que seleccionaremos este icono.



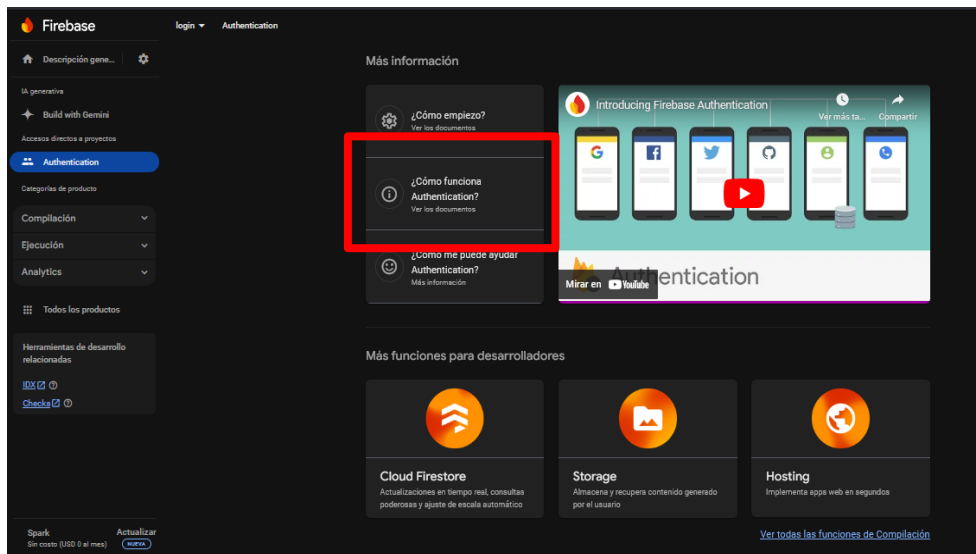
Una vez se seleccionado esta opción se presentará la siguiente pantalla para configurar el proyecto que se está creando

En la nueva ventana complementamos



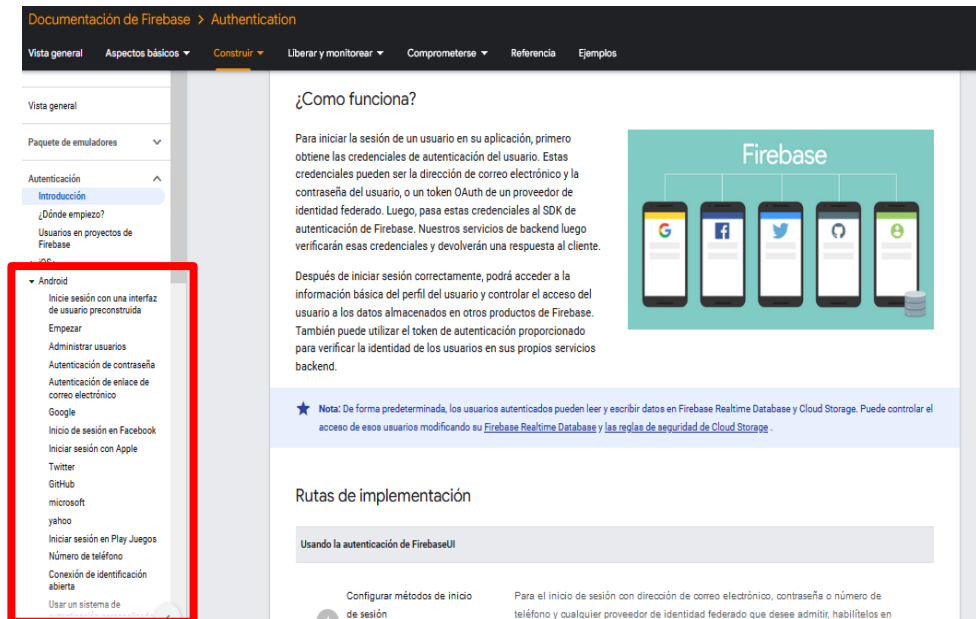


En la página siguiente podemos documentar mas sobre los tipos de autenticación



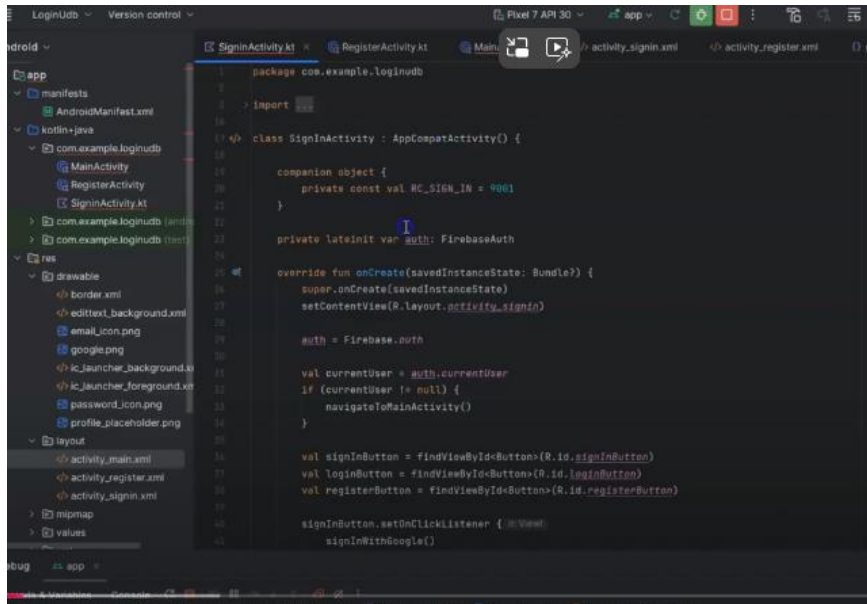
Nos documentamos sobre la autenticación que realizaremos y configuramos como se nos detallo anteriormente

Se nos abre la ventana con diversidad de información pero la que nos compete e interesa esa al lado izquierdo de la ventana que se nos abre posteriormente de seleccionar el botón marcada en la pantalla anterior



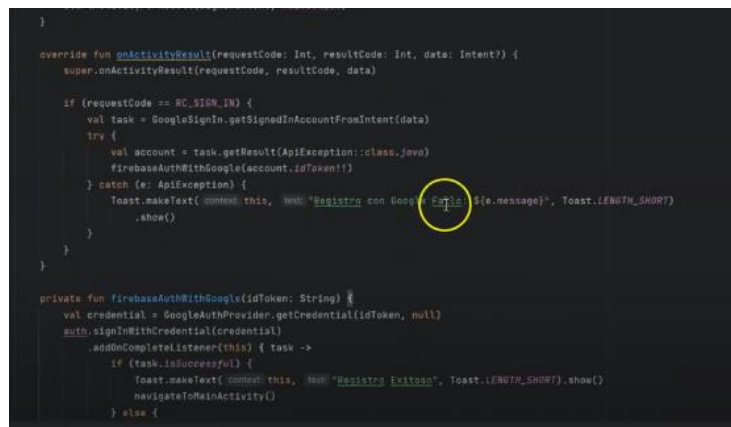
IMPLEMENTACION EN EL PROYECTO

Dentro de nuestro Android studio configuráramos según la documentación previamente examinada, y nos da como resultado la creación de la CLASE SIGNINACTIVITY

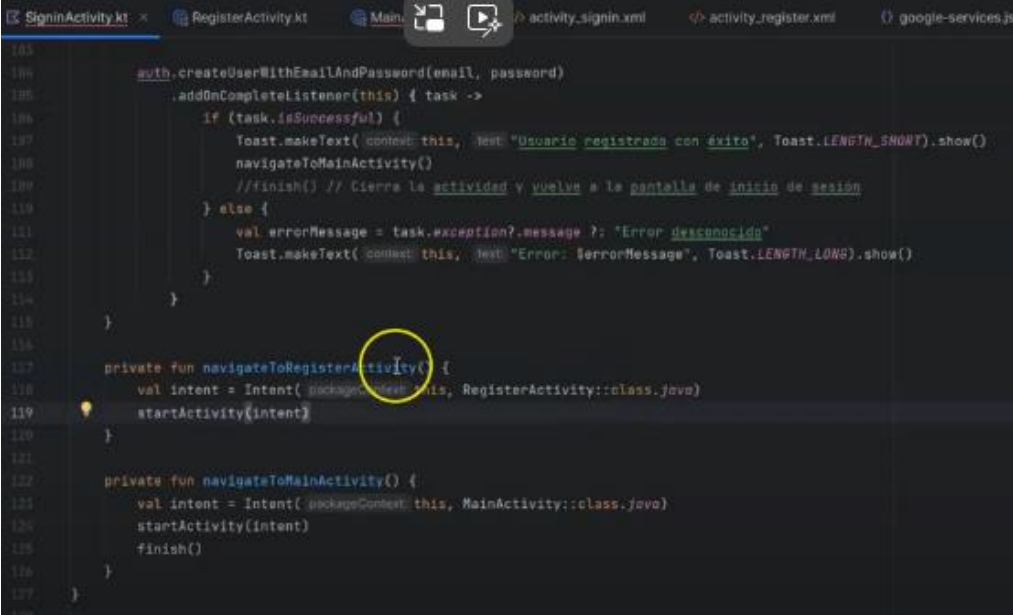


Gestiona los métodos de autenticación de Google y del correo y contraseña, en ambos métodos verificamos los datos de los usuarios mediante las funciones dentro de esta clase, para posteriormente redirigirlos a la clase de MainActivity, que estudiaremos más adelante.

Además de los métodos de autenticación en la clase `SignInActivity` también establecemos las acciones a realizar en caso de que ocurran fallos al momento de hacer el debdo registro, cuyos fallos se notificaran al usuario mediante mensajes.



Además de las acciones a realizar en caso de fallo dentro de esta clase también se establecen como redirigirse a la clase de registro que veremos mas adelante

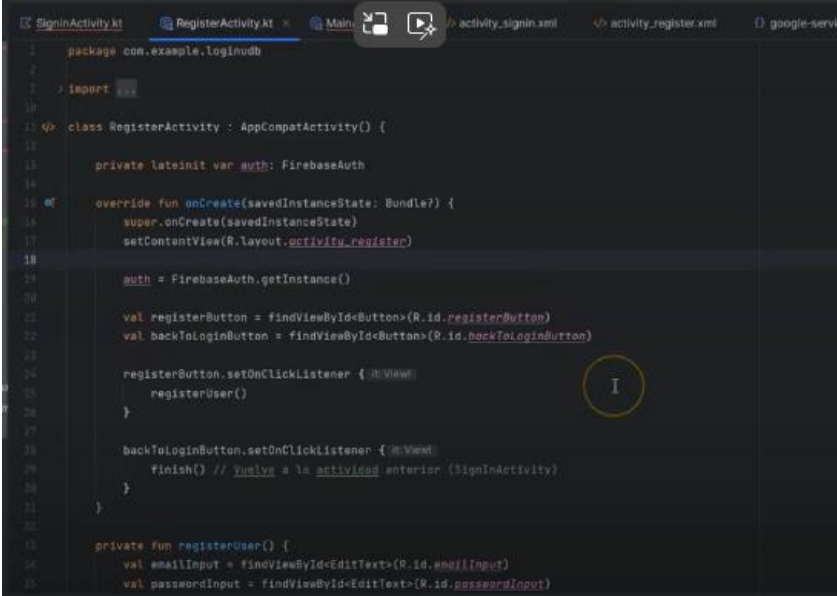


```

183
184     auth.createUserWithEmailAndPassword(email, password)
185     .addOnCompleteListener(this) { task ->
186         if (task.isSuccessful) {
187             Toast.makeText(context, this, text "Usuario registrado con éxito", Toast.LENGTH_SHORT).show()
188             navigateToMainActivity()
189             //finish() // Cierra la actividad y vuelve a la pantalla de inicio de sesión
190         } else {
191             val errorMessage = task.exception?.message ?: "Error desconocido"
192             Toast.makeText(context, this, text "Error: $errorMessage", Toast.LENGTH_LONG).show()
193         }
194     }
195 }
196
197 private fun navigateToRegisterActivity() {
198     val intent = Intent(packageContext, RegisterActivity::class.java)
199     startActivity(intent)
200 }
201
202 private fun navigateToMainActivity() {
203     val intent = Intent(packageContext, MainActivity::class.java)
204     startActivity(intent)
205     finish()
206 }
207 }
208
  
```

CLASE DE REGISTERACTIVITY

Dentro de esta clase se guardan las funciones que permiten a los usuarios realizar un registro exitoso dentro de la plataforma de Firebase



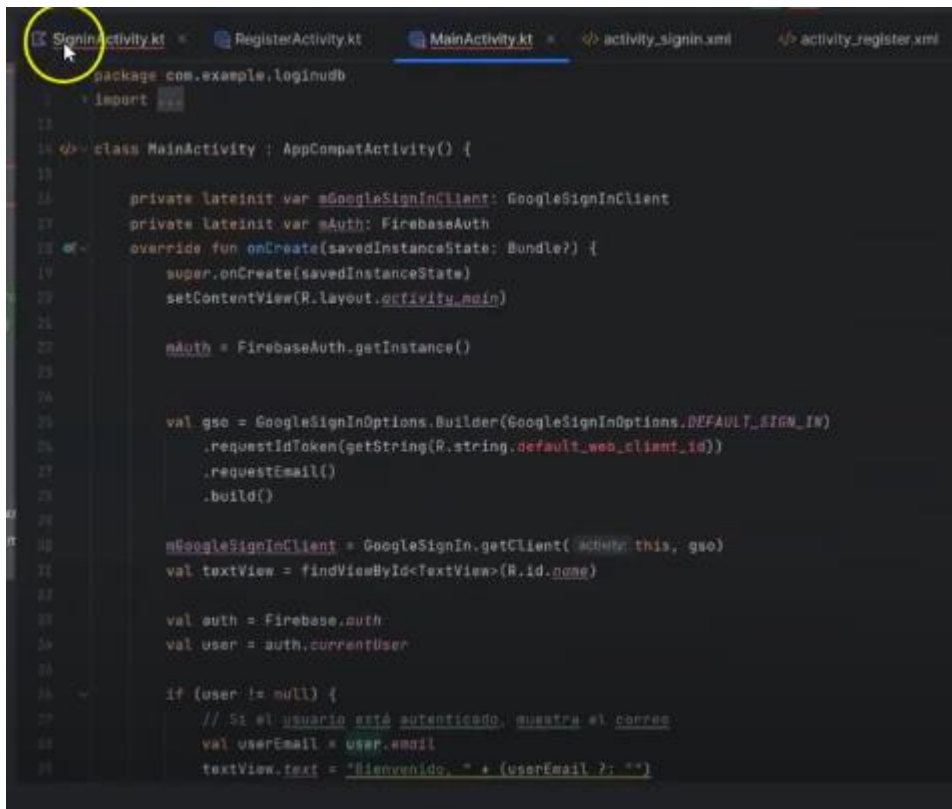
```

1 package com.example.logindb
2
3 import androidx.appcompat.app.AppCompatActivity
4
5 class RegisterActivity : AppCompatActivity() {
6
7     private lateinit var auth: FirebaseAuth
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_register)
12
13         auth = FirebaseAuth.getInstance()
14
15         val registerButton = findViewById<Button>(R.id.registerButton)
16         val backToLoginButton = findViewById<Button>(R.id.backToLoginButton)
17
18         registerButton.setOnClickListener {
19             registerUser()
20         }
21
22         backToLoginButton.setOnClickListener {
23             finish() // Vuelve a la actividad anterior (SignInActivity)
24         }
25     }
26
27     private fun registerUser() {
28         val emailInput = findViewById<EditText>(R.id.emailInput)
29         val passwordInput = findViewById<EditText>(R.id.passwordInput)
30
31     }
32 }
  
```


Dentro de esta clase se establecen las validaciones para establecer un correcto registro de ingreso a la plataforma creada, estableciendo parámetros de seguridad para la contraseña y los mensajes de éxito o no dependiendo de los valores ingresados a la app creada

CLASE MAIN ACTIVITY

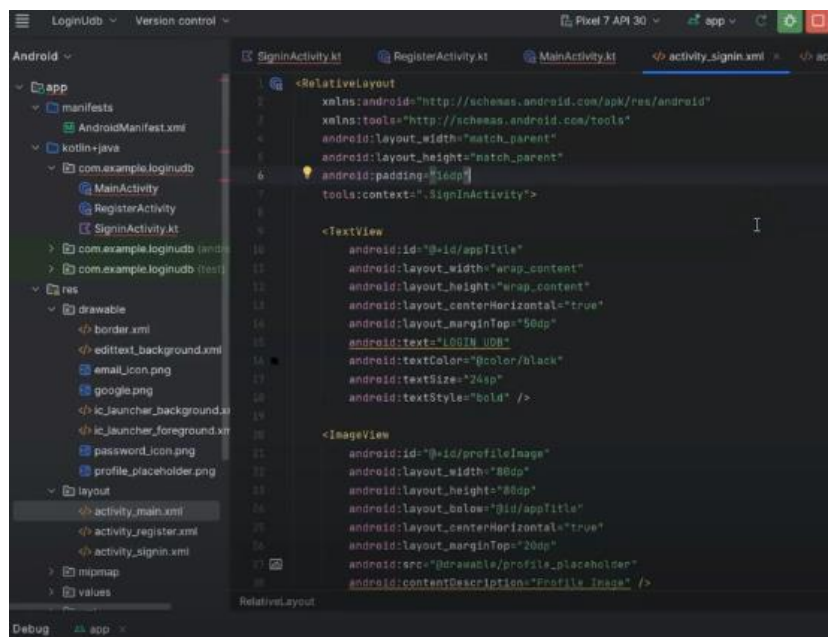
Esta es la clase principal de nuestra aplicación donde validamos los valores de las otras clases mencionadas previamente



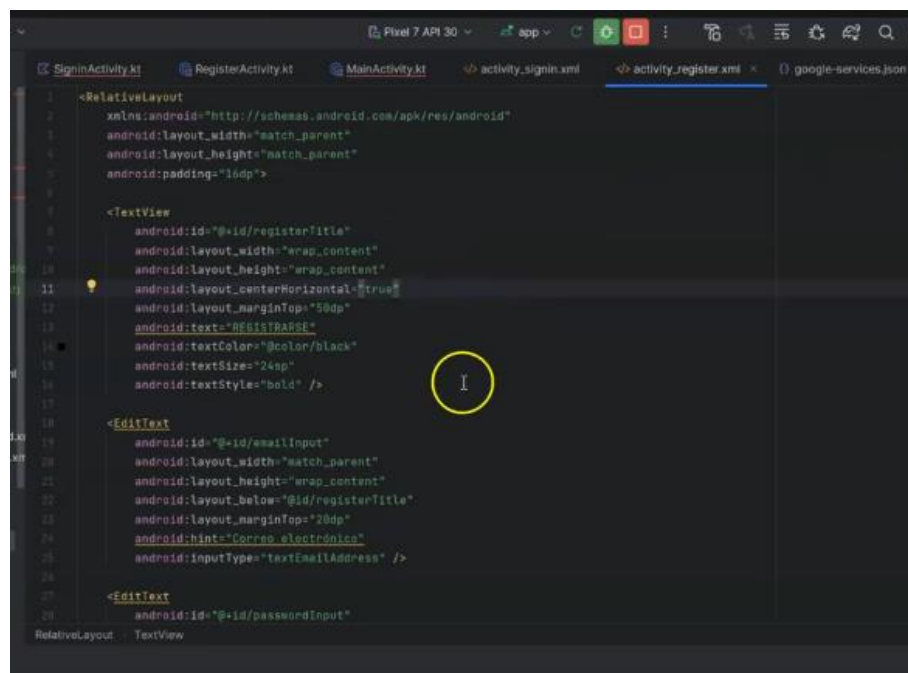
Así como las autenticaciones necesarias para poder hacer uso de las autenticaciones de Firebase, para observar el correcto funcionamiento de la app y Firebase, ver video de aplicación del funcionamiento de la app.

VISTAS DE APP

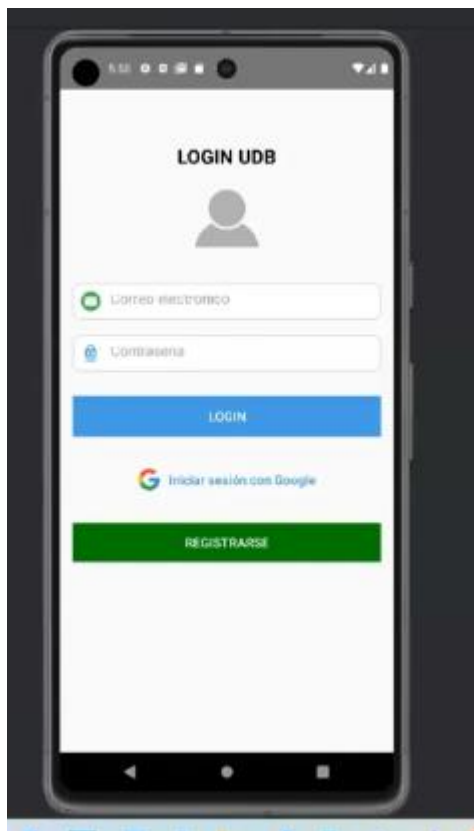
Tenemos las vistas, inicialmente el diseño de la pagina de inicio, esta hecho con un archivo XML



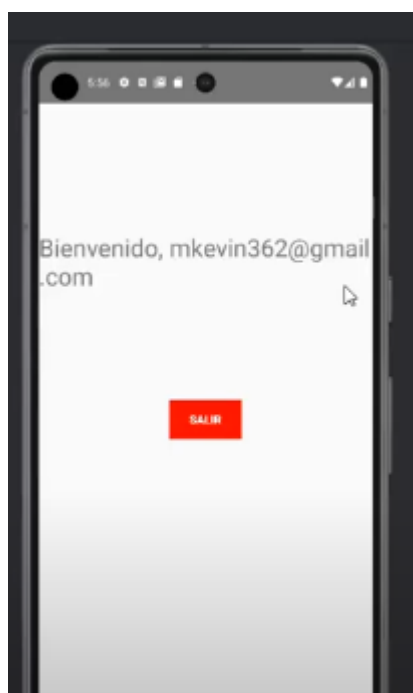
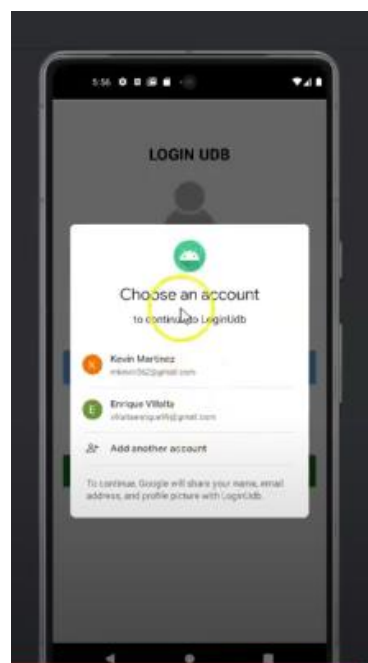
Adicional a la página anterior se tiene otra otra vista para el registro de nuevos usuarios



Las vistas de la nuestra aplicación quedaron de la siguiente forma



Las pantallas para registrar un nuevo usuario de ve de la siguiente manera



En esa última vista efectivamente se pudo acceder mediante la app y firebase a la app creada.

CONCLUSION

La combinación de Kotlin y Firebase es una estrategia óptima para el desarrollo de aplicaciones Android modernas. Proporciona un equilibrio entre facilidad de uso y robustez técnica, lo que permite a los equipos de desarrollo enfocarse en la experiencia del usuario y las funcionalidades principales de la aplicación, en lugar de invertir tiempo en construir sistemas de autenticación desde cero.

La autenticación, como componente crítico en la mayoría de las aplicaciones, se convierte en una funcionalidad segura, confiable y fácil de mantener al usar Firebase. Esto no solo reduce los riesgos asociados con la gestión manual de credenciales, sino que también mejora la experiencia del usuario final mediante procesos de inicio de sesión rápidos y familiares.

La implementación de la autenticación en aplicaciones Android utilizando Kotlin y Firebase es una solución efectiva que combina simplicidad, escalabilidad y seguridad. Firebase provee un conjunto robusto de herramientas que facilitan la gestión de usuarios, mientras que Kotlin, como lenguaje oficial de Android, ofrece un entorno moderno y eficiente para el desarrollo.

En conclusión, Firebase, combinado con Kotlin, es una herramienta poderosa para el desarrollo de autenticación en Android, ofreciendo un enfoque que es a la vez eficiente y escalable, ideal para proyectos de todos los tamaños.

BIBLIOGRAFÍA

Developers. (s.f.). *Desarrolla apps para Android con Kotlin*. Obtenido de <https://developer.android.com/kotlin?hl=es-419>

FIREBASE . (s.f.). *Autenticarse con Firebase usando cuentas basadas en contraseña en Android*. Obtenido de https://firebase.google.com/docs/auth/android/password-auth?hl=es&authuser=0#create_a_password-based_account

Obregon, A. (23 de JUNIO de 2023). *Usando Kotlin con Firebase - Una guía para desarrolladores de aplicaciones para Android*. Obtenido de <https://medium.com/@AlexanderObregon/using-kotlin-with-firebase-a-guide-for-android-app-developers-28f3fd2b57bc>