**Índice de contenidos**

[**Objetivos** 4](#_Toc138091384)

[**Estado del arte** 4](#_Toc138091385)

[Android Studio 5](#_Toc138091386)

[Kotlin 6](#_Toc138091387)

[¿Patrones? 6](#_Toc138091388)

[Firebase 7](#_Toc138091389)

[Firebase Authentication 8](#_Toc138091390)

[Firebase Cloud Firestore 9](#_Toc138091391)

[Firebase Cloud Storage 10](#_Toc138091392)

[**Aplicación** 11](#_Toc138091393)

[Interfaz de usuario 11](#_Toc138091394)

[Login 11](#_Toc138091395)

[Registro 13](#_Toc138091396)

[Principal 15](#_Toc138091397)

[Rutinas 15](#_Toc138091398)

[verRutina 15](#_Toc138091399)

[añadirEjercicios 15](#_Toc138091400)

[Ejercicios 15](#_Toc138091401)

[verEjercicio 15](#_Toc138091402)

[Nutricion 15](#_Toc138091403)

[Alimento 15](#_Toc138091404)

[Calendario 15](#_Toc138091405)

[añadirRutinaCalendario. 15](#_Toc138091406)

[Otros detalles 15](#_Toc138091407)

[Tutorial de la aplicación 15](#_Toc138091408)

[**Conclusiones y mejoras** 15](#_Toc138091409)

[**Referencias** 15](#_Toc138091410)

[Ilustración 1. Android Studio 6](#_Toc138182754)

[Ilustración 2. Logotipo de Kotlin 7](#_Toc138182755)

[Ilustración 3. Logotipo de Firebase 8](#_Toc138182756)

[Ilustración 4. Pantalla de Login 12](#_Toc138182757)

[Ilustración 5. Pantalla de Registro 14](#_Toc138182758)

[Ilustración 6. Pantalla Principal 16](#_Toc138182759)

[Ilustración 7. Pantalla de Rutinas 18](#_Toc138182760)

# **Objetivos**

El objetivo principal de este Trabajo de Fin de Grado (TFG) es desarrollar una aplicación móvil de gimnasio utilizando Jetpack Compose y Kotlin, con el propósito de ofrecer a los usuarios una experiencia intuitiva y eficiente para llevar un seguimiento de sus entrenamientos y mejorar su rendimiento físico.

Para lograr este objetivo, se plantean los siguientes subobjetivos:

* Investigar y analizar a fondo las tecnologías Jetpack Compose y Kotlin, comprendiendo sus características, ventajas y su integración en el ecosistema de desarrollo de Android.
* Diseñar una interfaz de usuario atractiva y fácil de usar para la aplicación, centrándose en aspectos clave como la usabilidad, la estética y la funcionalidad.
* Utilizar Jetpack Compose como herramienta principal para crear interfaces de usuario dinámicas y declarativas, aprovechando sus capacidades para componer vistas de manera eficiente.
* Implementar las funcionalidades necesarias para el seguimiento de entrenamientos, como la programación de rutinas, el registro de progresos y el acceso a bibliotecas de ejercicios.

Al cumplir estos objetivos, se espera demostrar la viabilidad y las ventajas del uso de Jetpack Compose y Kotlin en el desarrollo de aplicaciones de gimnasio para dispositivos móviles. La aplicación resultante brindará a los usuarios una herramienta efectiva para mejorar su experiencia en el gimnasio, llevar un seguimiento de sus entrenamientos y alcanzar sus objetivos de fitness.

# **Estado del arte**

## Android Studio

Android Studio es el entorno de desarrollo oficial para Android, basado en IntelliJ IDEA, lo que significa que cuenta con un poderoso editor de código y herramientas para desarrolladores. Además, ofrece funciones adicionales para la compilación de aplicaciones Android, tales como:

* Sistema de compilación basado en Gradle, lo que permite personalizar y configurar el proceso de compilación, así como crear múltiples APK con diferentes funciones usando el mismo proyecto.
* Un emulador rápido.
* Un entorno de desarrollo integrado para todos los desarrollos Android.
* Integración de plantillas de código.
* Instant Run para aplicar cambios rápidamente mientras se ejecuta la aplicación sin la necesidad de generar un nuevo APK.
* Herramientas Lint para detectar problemas en cuanto a rendimiento, usabilidad, compatibilidad de versiones, etc.
* Herramientas de depuración que incluyen monitores de rendimiento y seguimiento de asignaciones de memoria.
* Compatibilidad con C++.

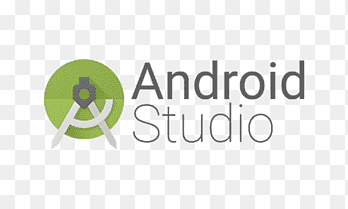


Ilustración . Android Studio

## Kotlin

Kotlin es un lenguaje de programación utilizado sobre todo para el desarrollo de aplicaciones Android que corre sobre la Máquina Virtual Java (JVM).

Fue desarrollado por JetBrains (IntelliJ IDEA y Android Studio) en 2011, aunque no se lanza su versión beta hasta Julio de 2015. En mayo de 2017, Google anuncia Kotlin como lenguaje oficial para el desarrollo de Android.

Como Kotlin y Android Studio fueron desarrollados por JetBrains, este lenguaje se encuentra perfectamente integrado en Android Studio, lo que pone a disposición de los desarrolladores las siguientes ventajas:

* **Compatibilidad**: Compatible con Java 6, lo que permite la compatibilidad con dispositivos antiguos.
* **Interoperabilidad**: Al ser interoperable con Java podemos seguir haciendo uso de las librearías de Java.
* **Performance**: Una aplicación desarrollada en Kotlin es igual de eficiente que una desarrollada en Java.
* **Curva** **de** **aprendizaje**: Es sencillo para un desarrollador Java empezar a desarrollar en Kotlin.
* **Seguro** **contra** **nulos**: Kotlin requiere que se verifique el valor no null de las variables en tiempo de compilación.
* **Programación funcional**: Kotlin incluye características de la programación funcional que se pueden combinar con la programación orientada a objetos.
* **Limpieza de código**: El código escrito en Kotlin es más legible y presenta características que permiten reducir la cantidad de código necesario.



Ilustración . Logotipo de Kotlin

## ¿Patrones?

## Firebase

Firebase es una plataforma de desarrollo de aplicaciones móviles y web desarrollada por Google. Proporciona una amplia gama de servicios y herramientas que permiten a los desarrolladores crear aplicaciones de alta calidad de manera más rápida y eficiente. Firebase se basa en la infraestructura de Google Cloud Platform y ofrece una integración perfecta con otras tecnologías de Google.

Los principales componentes de Firebase incluyen:

* Firebase Authentication: Un servicio de autenticación que permite a los usuarios registrarse e iniciar sesión en aplicaciones utilizando diferentes métodos, como correo electrónico y contraseña, proveedores de identidad externos y autenticación basada en números de teléfono.
* Firebase Cloud Firestore: Una base de datos en tiempo real y orientada a documentos que permite almacenar y sincronizar datos de manera eficiente entre clientes y servidores. Firestore ofrece una estructura de datos flexible y escalable, y proporciona potentes consultas y capacidades de sincronización en tiempo real.
* Firebase Cloud Storage: Un servicio de almacenamiento en la nube que permite a los desarrolladores almacenar y servir archivos estáticos, como imágenes, videos y documentos. Firebase Storage ofrece una alta disponibilidad y escalabilidad, y facilita la carga y descarga de archivos desde aplicaciones.

Firebase se destaca por su facilidad de uso, escalabilidad, confiabilidad y flexibilidad. Permite a los desarrolladores centrarse en la creación de características y experiencias de usuario excepcionales, ya que se encarga de la infraestructura subyacente y de la administración de servidores. Con Firebase, los desarrolladores pueden acelerar el desarrollo de aplicaciones, reducir la complejidad y ofrecer aplicaciones de alta calidad en menos tiempo.

En este proyecto nos hemos centrado en utilizar Firebase Authentication para la autenticación de usuarios y tanto Firebase Cloud Firestore como Firebase Cloud Storage para el almacenamiento de los datos en tiempo real.

Texto, Logotipo

Descripción generada automáticamente

Ilustración . Logotipo de Firebase

### Firebase Authentication

Firebase Authentication es un servicio de autenticación en la nube proporcionado por Google. Permite a los desarrolladores agregar fácilmente funcionalidades de autenticación segura a sus aplicaciones web y móviles, sin tener que desarrollar y mantener un sistema de autenticación personalizado.

Firebase Authentication ofrece una variedad de métodos de autenticación para adaptarse a las necesidades de diferentes aplicaciones y usuarios. Entre los métodos de autenticación disponibles se incluyen:

* Autenticación por correo electrónico y contraseña: Permite a los usuarios registrarse y acceder a la aplicación utilizando una combinación de correo electrónico y contraseña. Firebase maneja el almacenamiento seguro de las contraseñas y la verificación de las credenciales.
* Autenticación con proveedores de identidad externos: Firebase Authentication admite la autenticación con proveedores populares como Google, Facebook, Twitter y GitHub. Esto permite a los usuarios iniciar sesión en la aplicación utilizando sus cuentas existentes en estos servicios.
* Autenticación mediante números de teléfono: Firebase Authentication proporciona la capacidad de autenticar a los usuarios mediante la verificación de números de teléfono. Esto es especialmente útil para aplicaciones que se centran en la autenticación basada en SMS.
* Autenticación con proveedores de identidad personalizados: Firebase Authentication permite a los desarrolladores integrar proveedores de identidad personalizados utilizando OpenID Connect o SAML para autenticar a los usuarios con sistemas externos.

Una vez que los usuarios se autentican en la aplicación, Firebase Authentication proporciona una capa de seguridad para proteger los recursos y las acciones en la aplicación. Los desarrolladores pueden utilizar reglas de seguridad personalizadas para definir quién tiene acceso a qué datos y acciones, y Firebase se encargará de hacer cumplir esas reglas.

En resumen, Firebase Authentication es un servicio de autenticación en la nube que simplifica la implementación de la autenticación segura en aplicaciones web y móviles. Ofrece varios métodos de autenticación, protección de recursos y acciones con reglas de seguridad personalizadas, y funcionalidades adicionales para administrar usuarios. Con Firebase Authentication, los desarrolladores pueden garantizar la autenticación segura de los usuarios y concentrarse en la construcción de una experiencia de usuario excelente.

En este proyecto se ha utilizado para autenticar y almacenar usuarios por correo electrónico y contraseña o a través de Google.

### Firebase Cloud Firestore

Firebase Cloud Firestore es un servicio de base de datos en la nube proporcionado por Google. Se basa en un modelo de datos flexible y escalable, diseñado para almacenar y sincronizar datos en tiempo real en aplicaciones web y móviles.

Firestore utiliza un enfoque de base de datos NoSQL, lo que significa que no sigue el modelo tradicional de bases de datos relacionales. En lugar de eso, Firestore utiliza un modelo de datos basado en documentos y colecciones. Un documento es una unidad básica de almacenamiento que contiene datos en formato JSON. Estos documentos se agrupan en colecciones, lo que permite organizar los datos de manera lógica y jerárquica.

La escalabilidad y el rendimiento son características fundamentales de Firestore. El servicio está diseñado para manejar grandes volúmenes de datos y cargas de trabajo intensivas. Puede distribuir automáticamente los datos en diferentes servidores y escalar horizontalmente según sea necesario para garantizar un rendimiento rápido y constante.

Una de las ventajas clave de Firestore es su capacidad de sincronización en tiempo real. Esto significa que los cambios realizados en un dispositivo se propagan automáticamente a otros dispositivos conectados a la misma base de datos en tiempo real. Esto es especialmente útil para aplicaciones colaborativas o en tiempo real, donde múltiples usuarios pueden ver y actualizar los datos simultáneamente.

Firestore también ofrece capacidades de consulta y filtrado de datos flexibles. Puedes realizar consultas basadas en campos específicos, ordenar y filtrar resultados, y combinar múltiples condiciones de consulta para obtener los datos que necesitas de manera eficiente.

Además, Firestore se integra perfectamente con otras herramientas y servicios de Firebase, lo que amplía aún más su funcionalidad. Puedes utilizar la autenticación de usuarios, el almacenamiento de archivos, las notificaciones push y el análisis de Firebase junto con Firestore para construir aplicaciones completas y escalables.

En resumen, Firebase Cloud Firestore es un servicio de base de datos en la nube que ofrece un modelo de datos flexible, escalabilidad, sincronización en tiempo real y capacidades de consulta avanzadas. Es una opción poderosa para almacenar y sincronizar datos en aplicaciones web y móviles, y se integra estrechamente con otras herramientas de Firebase para brindar una solución completa de desarrollo de aplicaciones.

En este proyecto se ha utilizado Firebase Cloud Firestore para almacenar los datos referentes a ejercicios y rutinas.

### Firebase Cloud Storage

Firebase Cloud Storage es un servicio de almacenamiento en la nube ofrecido por Firebase, la plataforma de desarrollo de aplicaciones móviles y web de Google. Proporciona a los desarrolladores una solución escalable y confiable para almacenar y servir archivos estáticos, como imágenes, videos, documentos y otros recursos multimedia, en la infraestructura de almacenamiento en la nube de Google.

Algunas características clave de Firebase Cloud Storage incluyen:

* Almacenamiento seguro y duradero: Firebase Cloud Storage utiliza la infraestructura de almacenamiento de Google Cloud Platform, que ofrece altos niveles de seguridad y durabilidad. Los archivos se almacenan en ubicaciones geográficas redundantes para garantizar la disponibilidad y la integridad de los datos.
* Fácil integración: Firebase Cloud Storage se integra perfectamente con otros servicios de Firebase, lo que permite a los desarrolladores utilizarlo en conjunto con otros componentes, como Firebase Authentication y Firebase Firestore. Esto facilita el acceso y la administración de los archivos almacenados según las necesidades de autenticación y permisos de la aplicación.
* Carga y descarga de archivos sencilla: Los desarrolladores pueden cargar archivos en Firebase Cloud Storage desde aplicaciones móviles y web utilizando una API sencilla y fácil de usar. También pueden descargar archivos de manera eficiente para mostrarlos en la interfaz de usuario de la aplicación.
* Almacenamiento y entrega de contenido estático: Firebase Cloud Storage permite a los desarrolladores almacenar y servir contenido estático, como imágenes o archivos HTML, directamente desde la nube. Esto ayuda a mejorar el rendimiento de la aplicación al proporcionar una entrega rápida y eficiente de los recursos estáticos a los usuarios.

En resumen, Firebase Cloud Storage es una solución completa de almacenamiento en la nube que ofrece a los desarrolladores una forma sencilla y segura de almacenar y servir archivos estáticos en sus aplicaciones. Con su integración con otros servicios de Firebase y su escalabilidad inherente, Firebase Cloud Storage proporciona una base sólida para la gestión eficiente de archivos y recursos multimedia en aplicaciones móviles y web.

En esta aplicación se ha utilizado para el almacenamiento de las imágenes relacionadas a los ejercicios.

# **Aplicación**

## Interfaz de usuario

La interfaz de usuario consta de 12 pantallas: **Login, Registro, Principal, Rutinas, verRutina, añadirEjercicios, Ejercicios, verEjercicio, Nutricion, Alimento, Calendario y añadirRutinaCalendario.**

### Login

La pantalla de inicio de sesión (**LoginScreen**) implementada en esta aplicación tiene como objetivo permitir a los usuarios acceder a sus cuentas existentes para utilizar las funcionalidades de la aplicación.



Ilustración . Pantalla de Login

A continuación, se describe cómo funciona esta pantalla:

La pantalla se compone de varios elementos que facilitan la interacción del usuario con el formulario de inicio de sesión:

* Campo de correo electrónico: Este campo permite al usuario ingresar su dirección de correo electrónico.
* Campo de contraseña: Aquí, el usuario puede ingresar su contraseña para la autenticación. Además, se incluye un icono que permite alternar entre mostrar u ocultar la contraseña.
* Botón de inicio de sesión: Al hacer clic en este botón, se invoca la función **signInWithEmailAndPassword** del objeto **loginActions**. Esta función se encarga de autenticar al usuario utilizando el correo electrónico y la contraseña proporcionados. Si la autenticación es exitosa, se muestra un mensaje de "Inicio de sesión exitoso".
* Botón de inicio de sesión con Google: Al hacer clic en este botón, se invoca la función **signInWithGoogle** del objeto **loginActions**. Esta función inicia el proceso de inicio de sesión con Google utilizando la API de autenticación de Google. Si el inicio de sesión es exitoso, el usuario es redirigido a la pantalla principal de la aplicación.
* Texto y botón de registro: Se muestra un texto que indica "¿No tienes una cuenta?" y un botón "Regístrate". Al hacer clic en este botón, se invoca la función **navRegister** del objeto **loginActions**, que redirige al usuario a la pantalla de registro.

Además de estos elementos, la pantalla también utiliza el componente **Scaffold** de Jetpack Compose para estructurar el diseño general. La función **setContent** se encarga de establecer el contenido de la actividad utilizando la función **LoginScreen** como punto de entrada.

En la clase **LoginActivity**, que actúa como controlador de la pantalla, se implementan las acciones definidas en la interfaz **LoginActions**. Estas acciones se comunican con la pantalla de inicio de sesión para realizar las operaciones de inicio de sesión correspondientes, como el inicio de sesión con correo electrónico y contraseña, y el inicio de sesión con Google. Además, se manejan las respuestas de los eventos, como la autenticación exitosa o los errores de inicio de sesión.

### Registro

La pantalla de registro (**RegisterScreen**) implementada en esta aplicación tiene como objetivo permitir a los usuarios crear nuevas cuentas para acceder a las funcionalidades de la aplicación. A continuación, se describe cómo funciona esta pantalla:

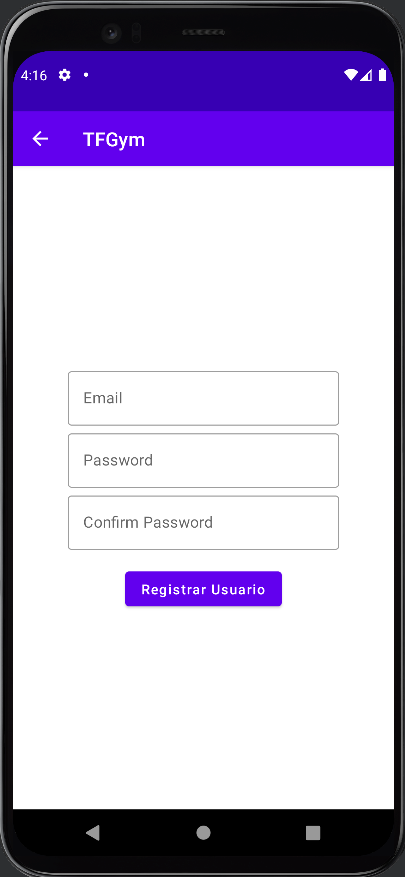


Ilustración . Pantalla de Registro

La pantalla de registro se compone de los siguientes elementos:

* Campo de correo electrónico: Este campo permite al usuario ingresar su dirección de correo electrónico.
* Campo de contraseña: Aquí, el usuario puede ingresar su contraseña para la creación de la cuenta. Al igual que en la pantalla de inicio de sesión, se incluye un ícono que permite alternar entre mostrar u ocultar la contraseña.
* Campo de confirmación de contraseña: Este campo permite al usuario ingresar nuevamente la contraseña para confirmarla y asegurarse de que coincide con la contraseña anteriormente ingresada.
* Botón de registro: Al hacer clic en este botón, se invoca la función **registerUser** del objeto **registerActions**. Esta función se encarga de procesar los datos ingresados por el usuario y realizar el registro en la base de datos. Si el registro es exitoso, se muestra un mensaje de "Registro exitoso" y se redirige al usuario a la pantalla principal de la aplicación.
* Botón de volver a la pantalla de inicio de sesión: Se muestra un botón con la opción de "Volver a inicio de sesión". Al hacer clic en este botón, se invoca la función **volverLogin** del objeto **registerActions**, que redirige al usuario a la pantalla de inicio de sesión.

Al igual que en la pantalla de inicio de sesión, la pantalla de registro utiliza el componente **Scaffold** de Jetpack Compose para estructurar el diseño general. La función **setContent** se encarga de establecer el contenido de la actividad utilizando la función **RegisterScreen** como punto de entrada.

En la clase **RegisterActivity**, que actúa como controlador de la pantalla, se implementan las acciones definidas en la interfaz **RegisterActions**. Estas acciones se comunican con la pantalla de registro para realizar las operaciones de registro correspondientes, como la creación de la cuenta en Firebase Authentication y el almacenamiento de los datos en Firebase Realtime Database. Además, se manejan las respuestas de los eventos, como el registro exitoso o los errores durante el proceso de registro.

En resumen, la pantalla de registro permite a los usuarios crear nuevas cuentas en la aplicación de gimnasio proporcionando su correo electrónico, contraseña y confirmación de contraseña. Proporciona una interfaz intuitiva y fluida para facilitar el proceso de registro y la integración con Firebase para almacenar los datos de usuario.

### Principal

La clase **PrincipalScreen** representa la pantalla principal de la aplicación de gimnasio, donde se muestran varias opciones de navegación para acceder a diferentes funcionalidades. A continuación, describiré cómo funciona esta pantalla:



Ilustración . Pantalla Principal

La pantalla se compone de los siguientes elementos:

* Barra de navegación superior: Utiliza el componente **TopAppBar** de Jetpack Compose para mostrar el título de la aplicación y un ícono de flecha hacia atrás. Al hacer clic en el ícono de flecha hacia atrás, se invoca la función **cerrarSesion** del objeto **mainNav** para cerrar la sesión actual y redirigir al usuario a la pantalla de inicio de sesión.
* Columna de opciones de navegación: Utiliza el componente **Column** para organizar las opciones de navegación en forma de lista vertical. Cada opción está envuelta en un **Box** que proporciona un área de pulsación amplia y un fondo de color correspondiente.
* Opción “Rutinas": Al hacer clic en esta opción, se invoca la función **navRutinas** del objeto **mainNav**, que redirige al usuario a la pantalla de rutinas.
* Opción "Nutrición": Al hacer clic en esta opción, se invoca la función **navNutricion** del objeto **mainNav**, que redirige al usuario a la pantalla de nutrición.
* Opción "Calendario de ejercicios": Al hacer clic en esta opción, se invoca la función **navCalendario** del objeto **mainNav**, que redirige al usuario a la pantalla de calendario de ejercicios.

En la clase **PrincipalActivity**, que actúa como controlador de la pantalla, se implementan las acciones definidas en la interfaz **MainNav**. Estas acciones se comunican con la pantalla principal para realizar las operaciones de navegación correspondientes, como redirigir al usuario a las pantallas de rutinas, nutrición y calendario de ejercicios, o cerrar la sesión actual. Además, se maneja la respuesta del evento de cierre de sesión, donde se utiliza FirebaseAuth para cerrar la sesión y se redirige al usuario a la pantalla de inicio de sesión.

En resumen, la pantalla principal (**PrincipalScreen**) muestra opciones de navegación para acceder a diferentes funcionalidades de la aplicación de gimnasio. Proporciona una interfaz de usuario intuitiva y fluida para facilitar la navegación y la interacción con las diferentes secciones de la aplicación.

Rutinas  
La clase **RutinaScreen** representa la pantalla de visualización de rutinas en la aplicación de gimnasio. A continuación, describiré cómo funciona esta pantalla:



Ilustración . Pantalla de Rutinas

La pantalla se compone de los siguientes elementos:

* Lista de rutinas: Utiliza el componente **LazyColumn** de Jetpack Compose para mostrar una lista vertical de rutinas. Cada rutina se representa mediante el componente **RutinaItem**, que muestra el nombre de la rutina y proporciona opciones para ver la rutina y eliminarla.
* Botón flotante para crear una nueva rutina: Utiliza el componente **FloatingActionButton** para mostrar un botón flotante con un ícono de "+" que representa la acción de crear una nueva rutina. Al hacer clic en este botón, se invoca la función **crearRutina()** de la interfaz **RutinasAction** para redirigir al usuario a la pantalla de selección de ejercicios.
* Barra de navegación superior: Utiliza el componente **TopAppBar** para mostrar el título de la aplicación. Además, incluye un ícono de flecha hacia atrás que permite al usuario regresar a la pantalla principal.

Al iniciar la actividad que contiene esta pantalla, se llama a la función **obtenerRutinas()** para obtener las rutinas del usuario desde Firebase Firestore. Estas rutinas se almacenan en una lista mutable (**listaRutinas**) que se utiliza para mostrar la información en la pantalla.

Cada rutina en la lista se muestra mediante el componente **RutinaItem**, que muestra el nombre de la rutina y proporciona dos opciones: "Ver rutina" y "Eliminar rutina". Al hacer clic en "Ver rutina", se invoca la función **mostrarRutina(rutina)** de la interfaz **RutinasAction**, lo que redirige al usuario a la pantalla de visualización detallada de la rutina seleccionada. Al hacer clic en "Eliminar rutina", se invoca la función **eliminarDocumento()** de la rutina correspondiente, lo que elimina la rutina de Firebase Firestore y la remueve de la lista **listaRutinas**.

En resumen, la pantalla **RutinaScreen** muestra una lista de rutinas, permite al usuario crear nuevas rutinas y proporciona opciones para ver y eliminar las rutinas existentes. Proporciona una interfaz intuitiva para administrar las rutinas de entrenamiento en la aplicación de gimnasio.

### verRutina

### añadirEjercicios

### Ejercicios

### verEjercicio

### Nutricion

### Alimento

### Calendario

### añadirRutinaCalendario.

## Otros detalles

## Tutorial de la aplicación

# **Conclusiones y mejoras**

# **Referencias**