

# SIMPLEFIT

## Sumario

SIMPLEFIT.....	1
ABSTRACT.....	2
INTRODUCCIÓN.....	3
Necesidades empresariales para el desarrollo.....	4
Recursos Humanos.....	4
Trabajadores de la empresa.....	4
Contratos de trabajo a realizar.....	4
Obligaciones en materia de Seguridad Social.....	5
Prevención de riesgos laborales.....	5
Ergonomía del puesto de trabajo:.....	5
Pausas y descansos regulares:.....	5
Iluminación adecuada:.....	5
Manejo del estrés y la carga de trabajo:.....	6
Seguridad informática:.....	6
Tramites administrativos.....	6
Requisitos funcionales de la aplicación.....	7
Análisis y Diseño.....	9
Diagrama de la arquitectura.....	9
Diagrama de casos de uso.....	9
Aplicación de Administración.....	9
Aplicación principal (Para usuarios).....	11
Diagrama de clases.....	12
Diseño de datos.....	14
Codificación.....	15
Jetpack Compose.....	16
Retrofit.....	17
JPA.....	18
Manual de usuario.....	19
Aplicación Usuarios (SimpleFit).....	19
Aplicación <i>Administradores</i> (SimpleFitAdmin).....	26
Requisitos e instalación.....	29
Conclusiones.....	29
Posibles ampliaciones y mejoras.....	30
Bibliografía.....	30

# ABSTRACT

Our SimpleFit App offers a service for users who are looking to get started in the gym, or on the other hand, do not have time to plan a routine. With our application, the user will be able to discover new workout routines that can be adapted to their needs, for this, the user will be able to visualize the different exercises per day, as well as a brief description of each routine to make the selection process easier and faster. For convenience, the user will be able to see in detail what exercises to perform that day, in turn, will be shown a series of tips and teachings for the user to acquire knowledge about training and health. We understand that users interested in our application, want to have a health and physical and mental stability, so we have dedicated a section for the user to provide some of their physiological data which can keep track, reviewing and updating them in case of change, thus, you can check if the routine that has in progress has led to any expected change in his physique. All the exercises of each routine are documented with a brief explanation and an image so that the user understands what type of exercise it is, knowing the muscle to exercise and the biomechanics of the exercise. In the search for routines, we provide a filter so that the user can sort by the difficulty of the routine or how many days of training you want to perform, since all routines are weekly, in other words, that apply from Monday to Sunday to facilitate the organization of the user. The process of selecting exercises and routines has been studied in detail so that the user is able to apply any of them without any impediment. The idea of the application is to add new routines and functionalities to expand the range of users and cover more sports areas, which would guarantee a large number of possibilities

# INTRODUCCIÓN

El proyecto SimpleFit tiene el objetivo de crear una aplicación donde los usuarios puedan desarrollarse físicamente en base a sus preferencias. Para ello, se realizará una búsqueda entre las rutinas que constan en la aplicación, todas ellas podrán ser identificadas por su nombre, la dificultad y frecuencia de entrenamiento, lo que facilita la selección de una rutina.

La aplicación no cuenta con un público específico ya que principiantes que quieran iniciarse en el gimnasio pueden usarla y aprender, y personas que tengan cierto nivel también pueden usarla si quieren probar rutinas nuevas o no tienen tiempo para planificar una rutina y días de descanso.

La idea principal es que los usuarios aprendan con la información que compartimos y puedan cumplir sus objetivos en el gimnasio de forma rápida y sin complicaciones.

# Necesidades empresariales para el desarrollo

## Recursos Humanos

### Trabajadores de la empresa

José García es un desarrollador de aplicaciones móviles con más de 5 años de experiencia en la creación de interfaces de usuario (UI) para Android. Ha trabajado en diversas startups y empresas de tecnología, destacándose por su habilidad para diseñar aplicaciones intuitivas y visualmente atractivas. José es experto en el uso de XML para el diseño de layouts y en la implementación de componentes personalizados. En los últimos 2 años, ha centrado su desarrollo en Jetpack Compose, la herramienta declarativa moderna de Android, con la cual ha creado múltiples aplicaciones desde cero, mejorando significativamente la velocidad de respuesta y la satisfacción del usuario. Además de sus conocimientos en UI, domina Kotlin y Java, integrando sus diseños con la lógica de la aplicación de manera eficiente y utilizando metodologías ágiles para el desarrollo de software.

Marta López es una ingeniera de software con más de 8 años de experiencia en el manejo de bases de datos y el desarrollo de API Rest. Es una experta en MySQL, con amplia experiencia en la creación, optimización y administración de bases de datos, diseñando esquemas complejos e implementando estrategias de indexación y consultas optimizadas. También cuenta con gran experiencia en el desarrollo de API Restful, habiendo creado numerosas API escalables, seguras y fáciles de mantener utilizando tecnologías como Node.js, Express y Laravel. Su capacidad para gestionar proyectos críticos y su enfoque en la precisión y eficiencia en el manejo de datos la convierten en una pieza clave para cualquier equipo de desarrollo.

### Contratos de trabajo a realizar

El acuerdo permanente con José García refleja un compromiso duradero tanto para él como para la compañía. Este tipo de contrato asegura una relación laboral continua, proporcionando estabilidad y seguridad mutua sin una fecha de terminación específica. José se compromete a aplicar sus conocimientos y experiencia en el desarrollo de interfaces de usuario de Android y Jetpack Compose de manera constante, lo que le permite involucrarse en una variedad de proyectos y aportar al crecimiento y éxito sostenido de la empresa.

El contrato de duración determinada de un año firmado con Marta López responde a una demanda específica de la organización para proyectos a corto plazo o aquellos que requieren habilidades especializadas en bases de datos MySQL y desarrollo de API Rest. Este tipo de contrato ofrece a la empresa la flexibilidad necesaria para abordar incrementos temporales en la carga de trabajo o proyectos específicos con una duración claramente definida.

## Obligaciones en materia de Seguridad Social

Para la incorporación de empleados, es esencial seguir varios procedimientos administrativos clave. Primero, se debe inscribir a todos los empleados en la Seguridad Social, cumpliendo con todas las normativas legales y administrativas pertinentes. Además, es necesario registrar los contratos laborales en el Servicio Público de Empleo Estatal (SEPE), asegurando la correcta formalización de la relación laboral. También se deben realizar los pagos correspondientes de las cuotas de cotización a la Seguridad Social, cumpliendo con las obligaciones fiscales y de seguridad social de la empresa. Por último, es importante tramitar la afiliación de los trabajadores al régimen de la Seguridad Social correspondiente, lo que les permitirá acceder a beneficios y protecciones como cobertura médica, prestaciones por enfermedad, maternidad y jubilación.

## Prevención de riesgos laborales

### ***Ergonomía del puesto de trabajo:***

Los programadores suelen pasar muchas horas frente al ordenador, lo que puede provocar problemas de salud si no se usan posturas correctas. Es importante tener una silla ergonómica y una disposición del escritorio que permita una buena postura y evite desequilibrios en articulaciones.

### ***Pausas y descansos regulares:***

La programación de larga duración puede causar fatiga mental y visual. Es importante tomar pausas cortas con frecuencia para descansar la vista y estirar el cuerpo.

### ***Iluminación adecuada:***

Un entorno de trabajo bien iluminado es esencial para prevenir la fatiga visual y el estrés ocular. La iluminación debe ser uniforme y ajustable para adaptarse a las necesidades personales.

### ***Manejo del estrés y la carga de trabajo:***

El desarrollo y gestión de software puede ser estresante debido a plazos ajustados y proyectos constantes. Es importante gestionar de forma efectiva el tiempo y establecer límites para evitar malestar físico y mental.

### ***Seguridad informática:***

Los programadores deben estar informados de los mejores protocolos de seguridad informática para protegerse a sí mismos y a sus herramientas de trabajo contra amenazas cibernéticas.

Se creará una Sociedad Limitada para facilitar la inclusión de socios ya que aportarán un capital acordado y resultará más sencillo la búsqueda de inversores y disfrutaremos de beneficios fiscales y administrativos.

## **Tramites administrativos**

Para registrar una empresa, es fundamental seguir diversos pasos esenciales. Primero, se debe escoger la estructura legal más adecuada, optando por una Sociedad Limitada (SL) para satisfacer los objetivos y requisitos de la empresa. Luego, se realiza el depósito del capital social mínimo requerido, que en este caso es de 8000€, en una cuenta bancaria a nombre de la entidad. Es importante seleccionar una denominación social única y disponible que refleje la identidad y actividad comercial de la empresa. Posteriormente, se redacta y firma una escritura pública de constitución de la Sociedad Limitada ante un notario, donde se establecen los estatutos y se formaliza la participación de los socios. A continuación, se solicita el Número de Identificación Fiscal (NIF) de la empresa ante la Agencia Tributaria, lo que permitirá su identificación oficial y la realización de trámites fiscales. Finalmente, se inscribe la empresa en el Registro Mercantil correspondiente a su domicilio social, formalizando así su existencia legal y otorgándole plena capacidad jurídica para operar.

## Requisitos funcionales de la aplicación

SimpleFit, está dirigida principalmente a usuarios novatos en el gimnasio con escaso conocimiento sobre ejercicios y planificación de rutinas, ofrece una solución completa. En nuestro catálogo de rutinas, los usuarios pueden elegir la opción que mejor se adapte a sus necesidades. Cada rutina proporciona detalles sobre su dificultad, objetivo general, días de entrenamiento y descanso, así como una explicación detallada de los ejercicios incluidos. Esta información facilita enormemente la selección de rutinas, permitiendo a los usuarios aprender y explorar diferentes aspectos del deporte de manera eficaz.

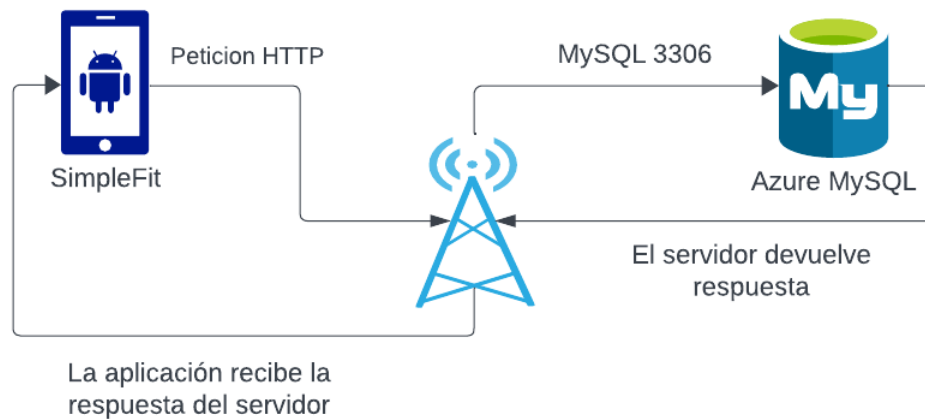
Además de los principiantes, otros públicos también pueden beneficiarse de SimpleFit. Aquellos que deseen descubrir nuevas rutinas que superen sus expectativas o que simplemente carezcan del tiempo necesario para planificar sus entrenamientos pueden encontrar en la aplicación una herramienta valiosa. La búsqueda de ejercicios, días de descanso y áreas musculares a trabajar se simplifica enormemente gracias a la interfaz intuitiva y detallada de la aplicación.

La aplicación cuenta además con una sección de seguimiento de progreso, donde los usuarios pueden registrar sus actividades y ver su evolución a lo largo del tiempo. Esto les permite mantenerse motivados y seguir avanzando hacia sus metas fitness de manera organizada y efectiva.

SimpleFit ha sido diseñada con la claridad y la usabilidad en mente, lo que la convierte en una herramienta fácil de entender para todos los usuarios. Cada función y detalle está meticulosamente explicado, garantizando que su uso sea sin complicaciones y accesible para todos los niveles de experiencia en el fitness.

# Análisis y Diseño

## Diagrama de la arquitectura



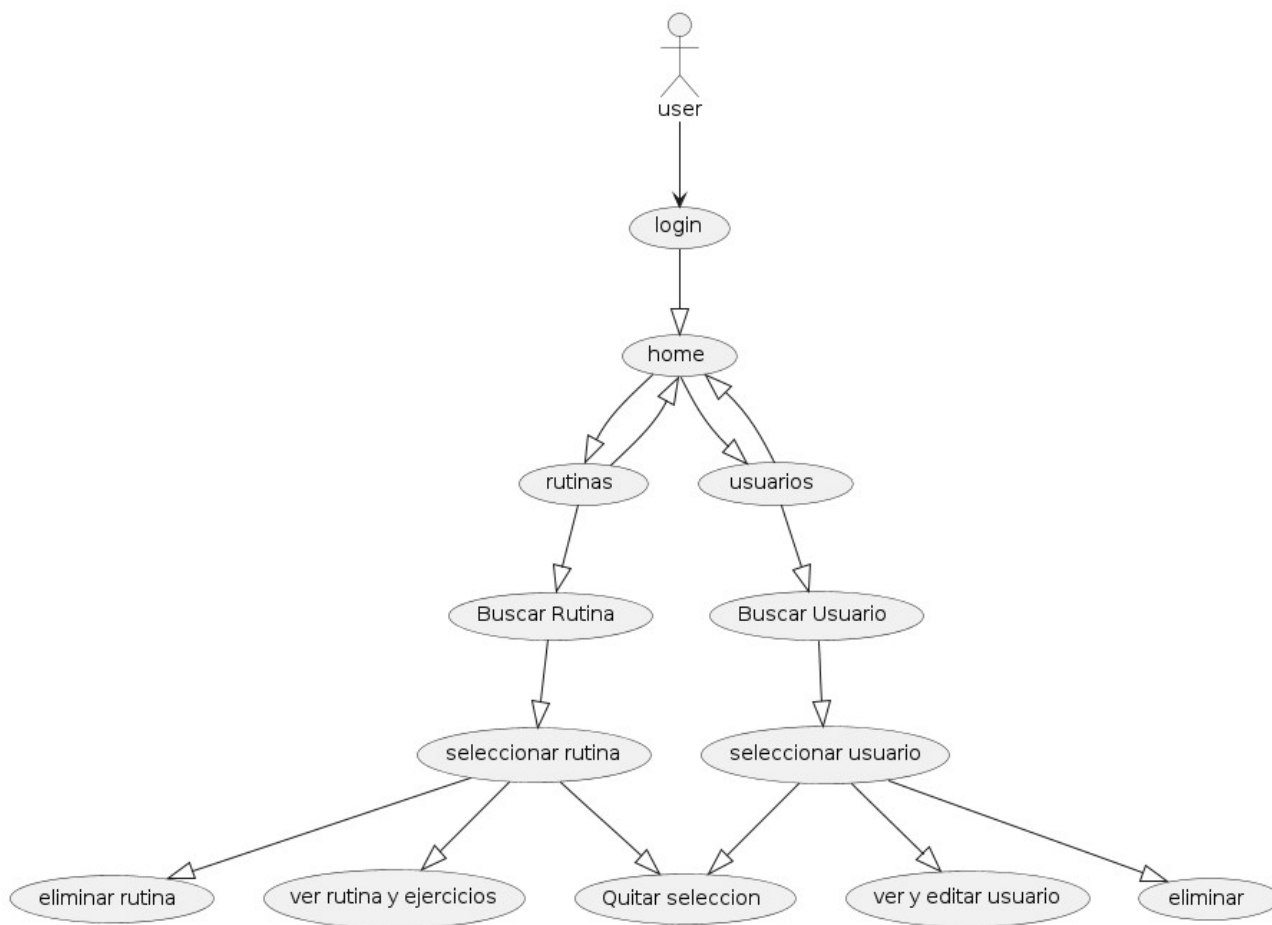
La arquitectura de nuestra aplicación móvil Android implica una interacción fluida entre varios componentes clave. La aplicación móvil actúa como la interfaz principal a través de la cual los usuarios interactúan con nuestro servicio. Utiliza tecnologías estándar de Android SDK y Retrofit para realizar solicitudes HTTP a nuestro servidor, alojado en Microsoft Azure. Este servidor, a su vez, alberga una API RESTful que gestiona la lógica de negocio de la aplicación y actúa como intermediario entre la aplicación móvil y nuestra base de datos MySQL.

Nuestro servidor en Azure se comunica con la base de datos MySQL a través del puerto estándar 3306, donde se almacenan y gestionan los datos de la aplicación. Este flujo de información es crítico para la funcionalidad de nuestra aplicación, ya que garantiza la integridad y la disponibilidad de los datos en todo momento.



# Diagrama de casos de uso

## Aplicación de Administración



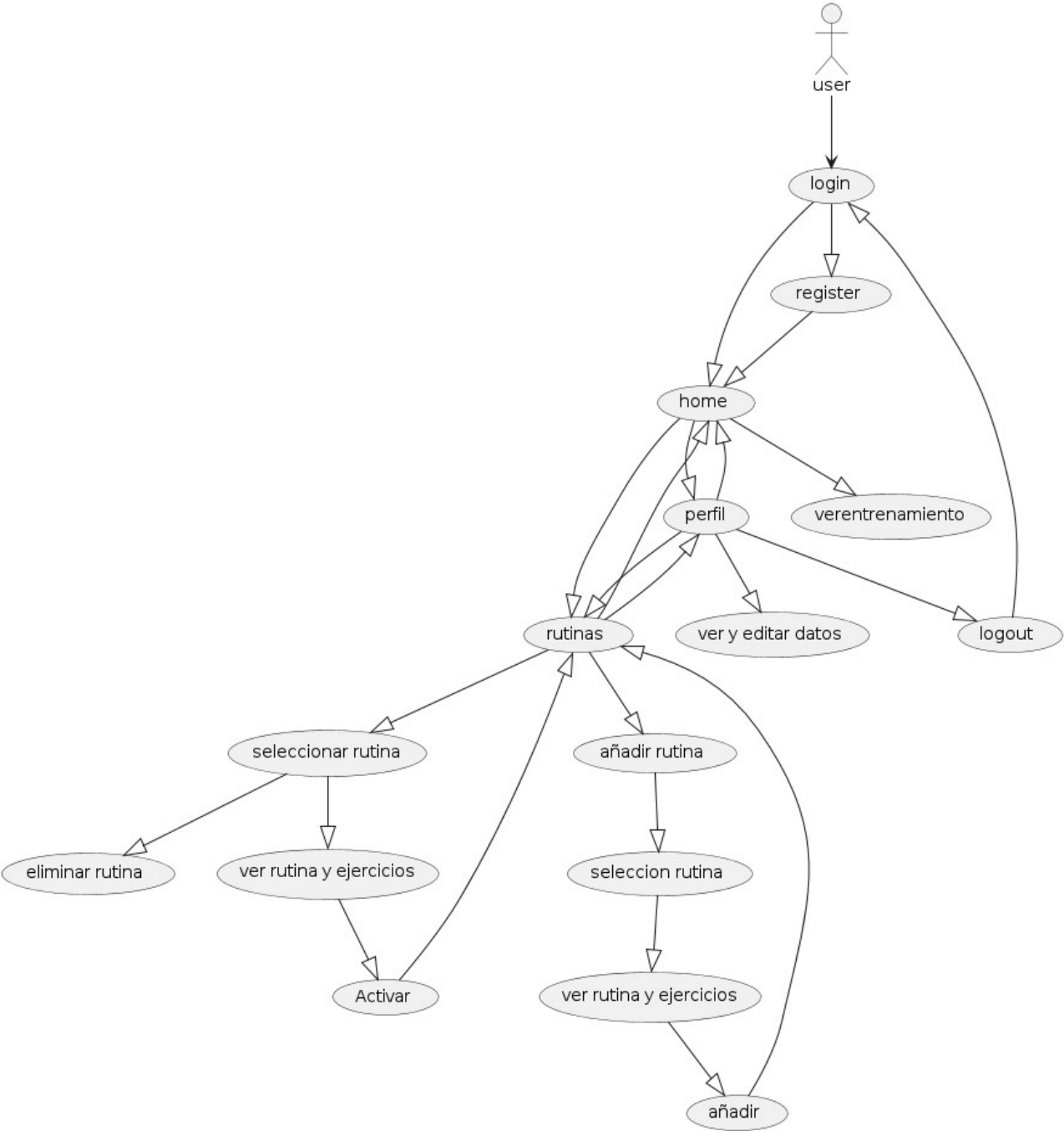
El diagrama de casos de uso de nuestra aplicación es una representación visual clara y concisa de cómo los usuarios interactúan con el sistema. Diseñado con la simplicidad y la intuitividad en mente, este diagrama refleja nuestra prioridad de ofrecer una experiencia de usuario sin complicaciones.

Una de las principales fortalezas de nuestro diagrama es su facilidad de comprensión. Cada caso de uso está cuidadosamente identificado y etiquetado, lo que permite a cualquier persona, desde usuarios finales hasta desarrolladores, entender rápidamente cómo se espera que funcione la aplicación. Esta claridad garantiza que los usuarios puedan navegar por la aplicación sin esfuerzo y realizar las acciones deseadas de manera eficiente.

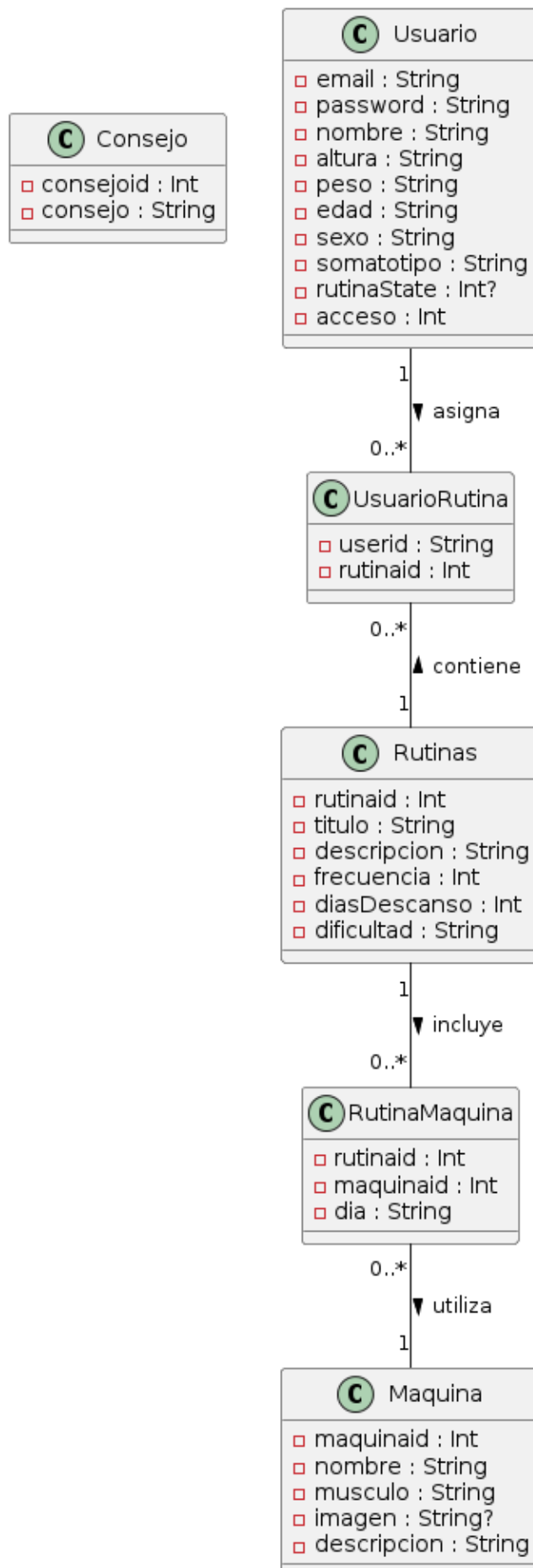
Además de su simplicidad, el diagrama de casos de uso destaca la amplitud de funciones que ofrece nuestra aplicación. Desde la gestión de rutinas de ejercicio hasta la visualización de consejos de salud, cada caso de uso representa una pieza importante del rompecabezas que conforma nuestra plataforma. Esta diversidad funcional asegura que los usuarios encuentren todo lo que necesitan dentro de la aplicación, convirtiéndola en una herramienta completa y versátil para mejorar su bienestar.

A continuación se podrá observar el diagrama de la aplicación principal, cuya implementación es similar a la de administración.

**Aplicación principal (Para usuarios)**



## Diagrama de clases



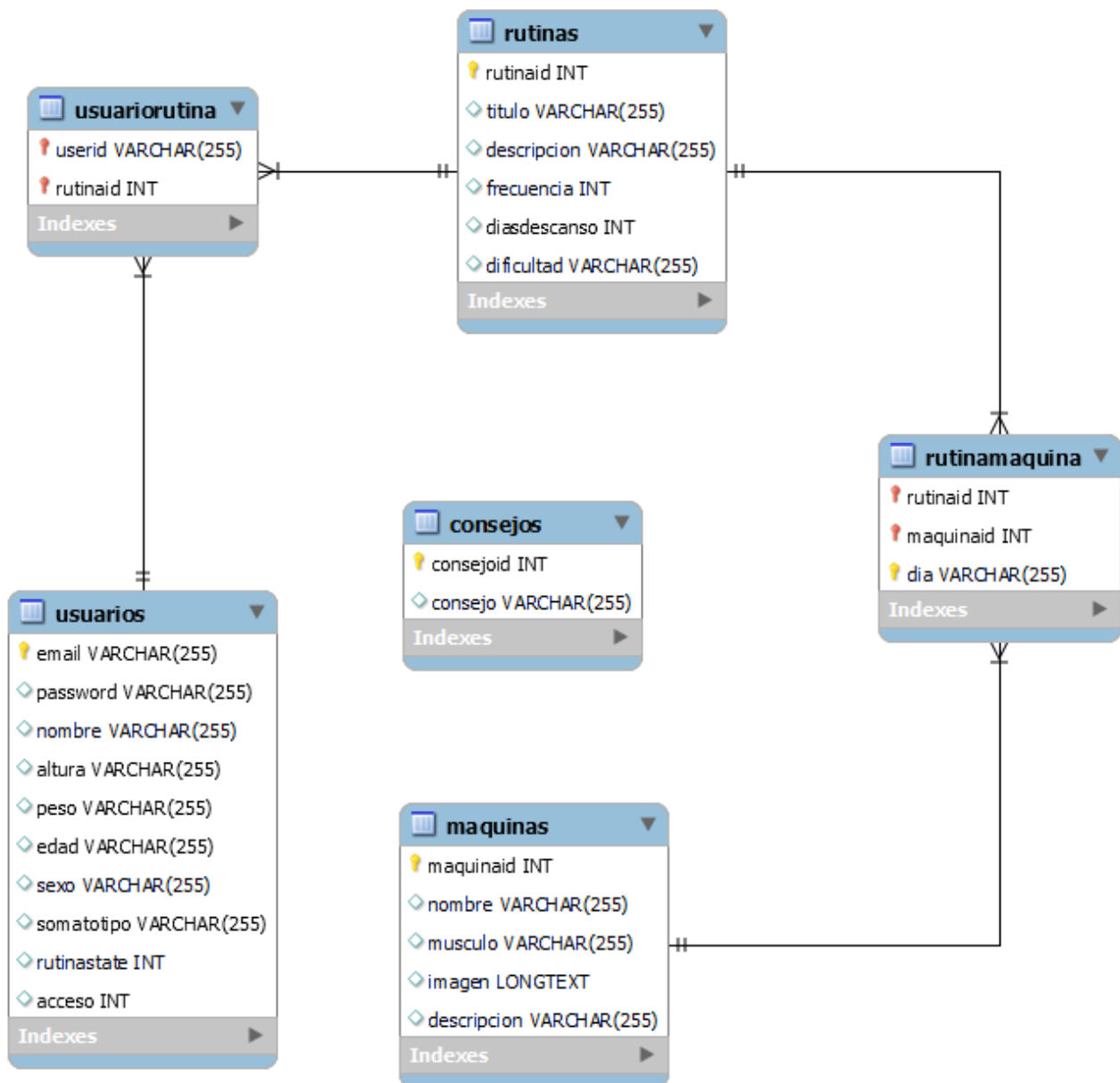
En el diagrama de clases del proyecto, las relaciones entre clases son esenciales para estructurar y organizar el sistema. Las clases incluidas (“Consejo”, “Maquina”, “RutinaMaquina”, “Rutinas”, “Usuario”, “UsuarioRutina”) y sus relaciones muestran cómo interactúan entre sí. Por ejemplo, la asociación entre “Usuario” y “UsuarioRutina” permite a un usuario tener varias rutinas asignadas. La composición implícita entre “Rutinas” y “RutinaMaquina” indica que una rutina incluye varias máquinas, y estas no tienen sentido sin la rutina específica. Además, la agregación entre “RutinaMaquina” y “Maquina” muestra que una máquina puede ser parte de varias rutinas pero también existir independientemente.

Estas relaciones aportan claridad y organización al diagrama, facilitando la comprensión de cómo los usuarios interactúan con las rutinas y cómo estas se descomponen en el uso de diferentes máquinas. Permiten la reutilización de clases como “Rutinas” y “Maquina” en diferentes partes del sistema, promoviendo la consistencia y la flexibilidad para agregar nuevas máquinas o rutinas sin afectar significativamente otras partes.

Sin embargo, a medida que se añaden más clases y relaciones, el diagrama puede volverse complejo, dificultando su comprensión. Además, la relación entre “UsuarioRutina” y “Rutinas” implica cierta dependencia, lo que puede complicar cambios en una clase sin afectar a las demás.

Resumiendo, el diagrama de clases muestra cómo las relaciones entre clases estructuran el sistema, ofreciendo beneficios en claridad y reutilización, pero también presenta desafíos en términos de complejidad y acoplamiento.

## Diseño de datos



Estas relaciones son definidas mediante claves foráneas, asegurando que los datos se mantengan coherentes y que las operaciones de inserción, actualización y eliminación se realicen de manera segura y consistente. Por ejemplo, cada vez que se asigna una rutina a un usuario o se agrega una máquina a una rutina, las claves foráneas garantizan que los identificadores de usuario, rutina y máquina existan en las tablas correspondientes.

En resumen, el diagrama de diseño de datos es esencial para comprender la estructura y las relaciones de la base de datos subyacente de nuestra aplicación. Proporciona una guía clara para el almacenamiento y la gestión de datos, asegurando la integridad y consistencia de la información, y sentando las bases para un funcionamiento eficiente y sin complicaciones de la aplicación.

# Codificación

Se han desarrollado dos aplicaciones móviles utilizando Android Studio como IDE principal. La elección del lenguaje de programación Kotlin se debe a su creciente popularidad y a sus características modernas que facilitan el desarrollo de aplicaciones Android. Kotlin ofrece una sintaxis más concisa y segura, lo que permite escribir código más legible y menos propenso a errores. Además, su interoperabilidad con Java simplifica la migración de proyectos existentes y facilita la integración de bibliotecas de Java.

Para la comunicación con el servidor y la manipulación de datos, se ha empleado Retrofit junto con Kotlin en Android Studio. Retrofit es una biblioteca de cliente HTTP para Android y Java que simplifica la comunicación con servicios web RESTful. Utilizando anotaciones en interfaces Java o Kotlin, Retrofit permite definir de manera clara y concisa las operaciones HTTP disponibles, eliminando gran parte del código repetitivo y propenso a errores asociado con la comunicación de red en aplicaciones Android. Además, Retrofit ofrece soporte para la conversión automática de datos entre JSON y objetos Java/Kotlin, lo que simplifica aún más el proceso de manipulación de datos en la aplicación móvil.

En el lado del servidor, se ha utilizado NetBeans como IDE y Java como lenguaje de programación para crear una API REST. En particular, se ha hecho uso de la biblioteca Jakarta Persistence API (JPA), que es una especificación de Java EE para el mapeo objeto-relacional en aplicaciones Java. La ventaja principal de JPA es su capacidad para simplificar el acceso y la manipulación de datos en bases de datos relacionales mediante el uso de entidades y consultas en lenguaje de objetos. Comparado con otras opciones de mapeo objeto-relacional, como JDBC (Java Database Connectivity), JPA ofrece un nivel de abstracción más alto y una sintaxis más intuitiva, lo que reduce la cantidad de código necesario y mejora la productividad del desarrollador.

En el desarrollo de la aplicación móvil, además de Kotlin y Retrofit, se ha utilizado Jetpack Compose como una herramienta para la creación de la interfaz de usuario. Jetpack Compose es una biblioteca de UI moderna y declarativa para Android que permite a los desarrolladores construir interfaces de usuario de manera más sencilla y flexible. Al utilizar un enfoque basado en funciones y composición, Jetpack Compose simplifica la creación y el mantenimiento de interfaces de usuario complejas, al tiempo que proporciona una mayor flexibilidad y rendimiento en comparación con las herramientas tradicionales de diseño de interfaces de usuario en Android. Su adopción en el mercado actual es cada vez más relevante, ya que se considera el futuro del desarrollo de UI en Android, con un crecimiento constante en su comunidad de usuarios y una sólida integración con otras bibliotecas y herramientas de Android.

En cuanto a la elección de MySQL como sistema de gestión de bases de datos en la nube, se debe destacar su rendimiento, escalabilidad y fiabilidad. MySQL es uno de los sistemas de gestión de bases de datos relacionales más populares y ampliamente utilizados en el mundo, con una amplia comunidad de usuarios y una larga historia de desarrollo y mejora continua. Ofrece una gran gama

de características y funcionalidades avanzadas, como soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), replicación y clustering, lo que lo convierte en una opción sólida para aplicaciones que requieren un almacenamiento de datos robusto y escalable. Además, MySQL es compatible con una amplia variedad de plataformas y lenguajes de programación, lo que facilita su integración con diferentes entornos de desarrollo y sistemas existentes. Comparado con otras opciones de bases de datos relacionales como PostgreSQL o SQL Server, MySQL se destaca por su facilidad de uso, rendimiento y escalabilidad, lo que lo convierte en una elección popular para una amplia variedad de aplicaciones en la nube.

En resumen, la combinación de Kotlin, Retrofit, Java, JPA, Jetpack Compose y MySQL ha permitido un desarrollo eficiente y exitoso de las aplicaciones móviles y el servicio REST, cada uno aportando su propia ventaja técnica para satisfacer las necesidades del proyecto.

A continuación se muestran capturas de algunas características destacables acerca de las herramientas mencionadas anteriormente:

## Jetpack Compose

El código configura una pantalla en Jetpack Compose con un controlador de navegación NavController. La pantalla usa Scaffold para estructurar la interfaz de usuario, incluyendo una barra de navegación inferior que se muestra condicionalmente. La función “iOpcionNavegacionSeleccionadaAPartirDeRuta” traduce las rutas de navegación en índices de navegación, lo que permite al NavBar resaltar la opción correcta basada en la ruta actual.

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SimpleFitScreen() {

    val comportamientoAnteScrollInf = BottomAppBarDefaults.exitAlwaysScrollBehavior()
    val navController = rememberNavController()
    val entradaEnPilaDeNavegacionActuasState = navController.currentBackStackEntryAsState()
    var iOpcionNavegacionSeleccionada = entradaEnPilaDeNavegacionActuasState.value?.destination?.route?.let { iOpcionNavegacionSeleccionadaAPartirDeRuta(it) } ?: 0
    var verNavegacion = derivedStateOf {
        iOpcionNavegacionSeleccionada !in 3..8
    }

    Scaffold(
        modifier = Modifier.nestedScroll(comportamientoAnteScrollInf.nestedScrollConnection),
        topBar = {},
        snackbarHost = {},
        bottomBar = {
            if (verNavegacion.value)
                NavBar(
                    navController = navController,
                    currentIndex = iOpcionNavegacionSeleccionada
                )
        },
        content = { innerPadding ->
            SimpleFitNavHost(innerPadding = innerPadding, navController = navController)
        }
    )
}

@Marcsvss
private fun iOpcionNavegacionSeleccionadaAPartirDeRuta(route: String?): Int {
    return when (route?.substringBefore(delimiter = "/")) {
        HomeRoute -> 0
        RoutinesRoute -> 1
        ProfileRoute -> 2
        LoginRoute -> 3
        RegisterAccountInfoRoute -> 4
        RegisterProfileInfoRoute -> 5
        VerRutinaRoute -> 6
        AddRutinaRoute -> 7
        VerEntrenamientoRoute -> 8
    }
}
```

## Retrofit

En la siguiente captura podemos ver la gestión de los EndPoint para conectar con la Api, especificamos las rutas para realizar la petición que necesitemos, ya sea obtener usuarios, añadir uno a la base de datos o actualizarlo,

Trata de una interfaz “usuarioService” define métodos que corresponden a operaciones CRUD sobre usuarios, utilizando Retrofit. Cada método está anotado para especificar el tipo de solicitud HTTP (GET, POST, PUT, DELETE), la URL del endpoint y los encabezados necesarios para aceptar y enviar datos en formato JSON. Los métodos son declarados como suspend para que puedan ser llamados en coroutines, facilitando así la programación asíncrona en Kotlin.

```
new *
interface UsuarioService {

    new *
    @GET("usuarios")
    @Headers("Accept: application/json", "Content-Type: application/json")
    suspend fun usuarios(): Response<List<UsuarioApi>>

    new *
    @GET("usuarios/{email}")
    @Headers("Accept: application/json", "Content-Type: application/json")
    suspend fun usuario(@Path("email") email: String): Response<UsuarioApi>

    new *
    @POST("usuarios")
    @Headers("Accept: application/json", "Content-Type: application/json")
    suspend fun insert(@Body u : UsuarioApi): Response<RespuestaApi>

    new *
    @PUT("usuarios")
    @Headers("Accept: application/json", "Content-Type: application/json")
    suspend fun update(@Body u2 : UsuarioApi): Response<RespuestaApi>

    new *
    @DELETE("usuarios/delete/{email}")
    @Headers("Accept: application/json", "Content-Type: application/json")
    suspend fun delete(@Path("email") email: String): Response<RespuestaApi>
}
```



## JPA

Este método define un endpoint GET que recupera las máquinas asociadas a una rutina y un día específicos. Utiliza JPA para interactuar con la base de datos y JAX-RS para definir el endpoint REST. El método maneja las excepciones adecuadamente y devuelve respuestas HTTP correspondientes según los resultados de la consulta o cualquier error que ocurra durante la ejecución.

```
@GET
@Path("/{rutinaid}/{dia}")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response getOne(@PathParam("rutinaid") Integer rutinaid, @PathParam("dia") String dia) { //Maquinas de una rutina un dia en concreto
    EntityManagerFactory emf = null;

    HashMap<String, String> mensaje = new HashMap<>();
    Response response;
    Response.Status statusResul;

    try {
        emf = Persistence.createEntityManagerFactory(persistenceUnitName: PERSISTENCE_UNIT);
        EntityManager em = emf.createEntityManager();
        Query query = em.createNamedQuery(string: "Rutinamaquina.findByRutinaidAndDia");
        query.setParameter(string: "rutinaid", o: rutinaid);
        query.setParameter(string: "dia", o: dia);
        List<Integer> maquinaIds = query.getResultList(); //Nos devuelve una lista de enteros que corresponderán a la id de

        if (maquinaIds == null) { //Si no devuelve nada, nos informará de que no existe ninguna rutina con ese id
            statusResul = Response.Status.NOT_FOUND;
            mensaje.put(key: "mensaje", value: "No existe rutina con ID " + rutinaid);
            response = Response
                .status(status: statusResul)
                .entity(o: mensaje)
                .build();
        } else { //En caso de que devuelva registros, recogerá los datos.
            statusResul = Response.Status.OK;
            response = Response
                .status(status: statusResul)
                .entity(o: maquinaIds)
                .build();
        }
    } catch (Exception ex) {
        statusResul = Response.Status.BAD_REQUEST;
        mensaje.put(key: "mensaje", value: ex.getMessage());
        response = Response
            .status(status: statusResul)
            .entity(o: mensaje)
            .build();
    } finally {
        if (emf != null) {
            emf.close();
        }
    }

    return response;
}
```

At  
Ve


# Manual de usuario


## Aplicación Usuarios (SimpleFit)


Como podemos observar, la primera pantalla que encontraremos será el inicio de sesión al cual podrán acceder tanto usuarios como administradores, tendrán que identificarse con email y contraseña, si el usuario no está registrado, podrá crear su cuenta presionando el boton “Crear cuenta”.

### SIMPLEFIT

#### LOGIN

 ejemplo@correo.com

 Password



Login


No tienes cuenta?

Crear cuenta


En el registro, el usuario deberá introducir tanto su nombre como el email y la contraseña, en los campos que indicados textualmente y deberán pulsar en el botón “Sign Up” para realizar el registro.

### REGISTRO


#### NOMBRE


 Nombre

#### EMAIL

 ejemplo@correo.com

#### PASSWORD

 Contraseña




Sign Up


Tras realizar el registro, se introducirán los datos para llevar un registro del progreso, rellenando los campos altura y peso con más exactitud y los siguientes campos veremos un desplegable con las distintas opciones en las cuales el usuario seleccionará la que más se adecue.

### Completa tu perfil


**ALTURA**

 Altura

**PESO**

 Peso


**EDAD**

Edad 

**SEXO**

Sexo 

**SOMATOTIPO**

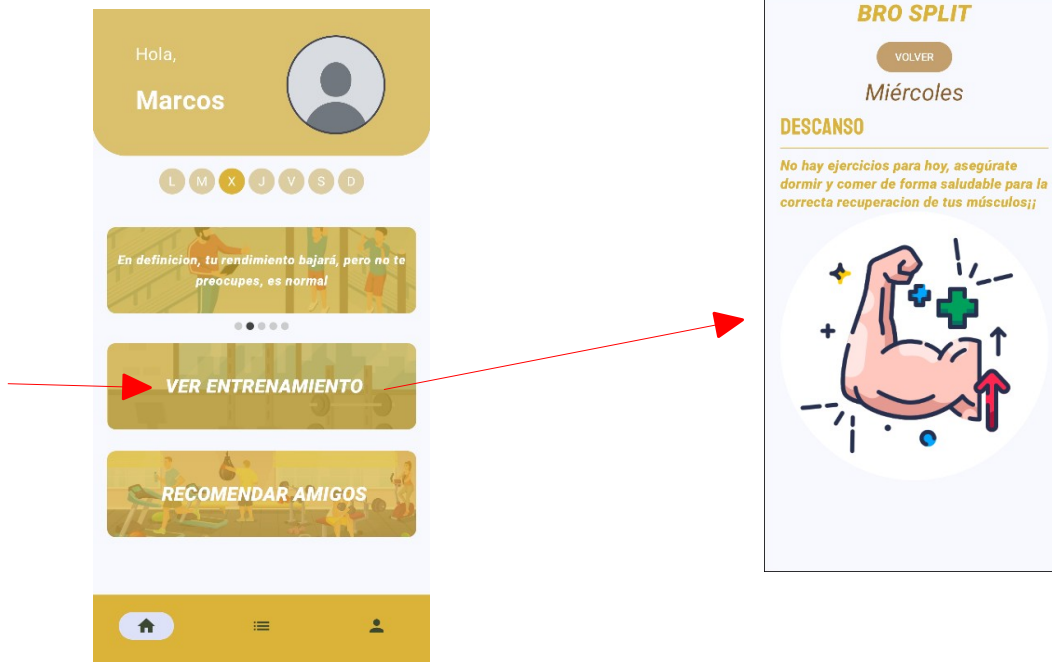
Somatotipo 

Guardar preferencias

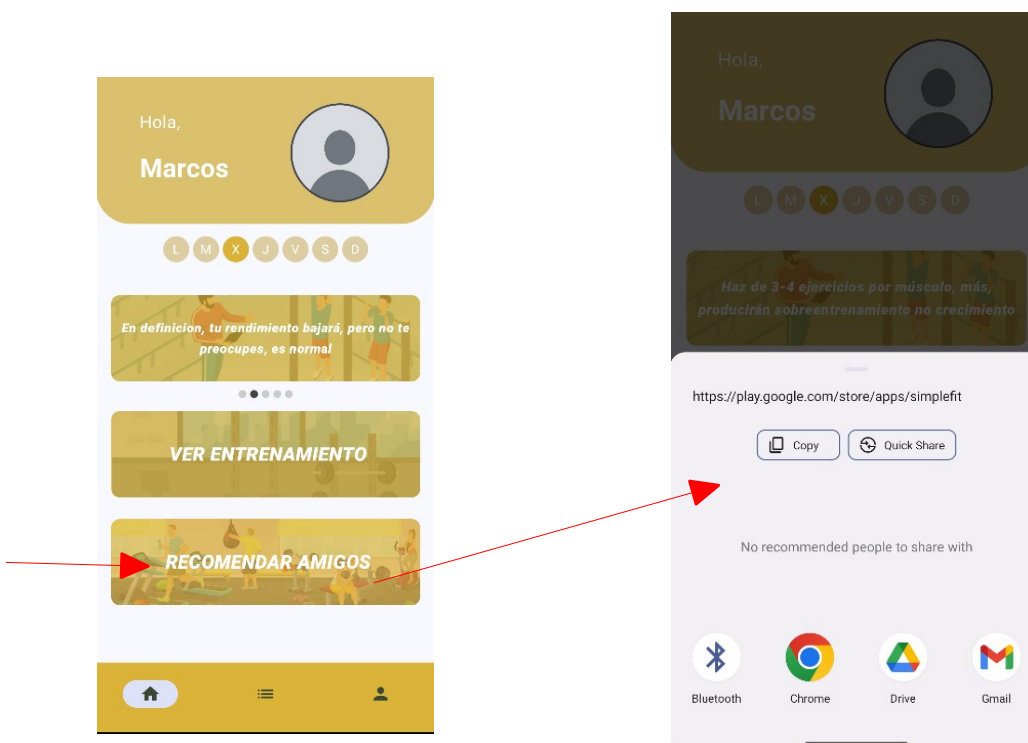
Tras iniciar sesión o registrarse y completar perfil, podremos ver la pantalla “Home” en la cual el usuario podrá cambiar su foto del perfil, ver el entrenamiento del día, una batería de consejos que podrá aprovechar en el ámbito deportivo y tendrá la opción de compartir la aplicación para que más usuarios puedan beneficiarse de su uso.



En el botón “VER ENTRENAMIENTO” podremos ver la rutina del día, en este caso, en la rutina “BRO SPLIT”, los miércoles es día de descanso.



Presionando en “RECOMENDAR AMIGOS”, nos permitirá compartir el enlace a la Play Store para descargar la aplicación, ya sea via mail, redes sociales...

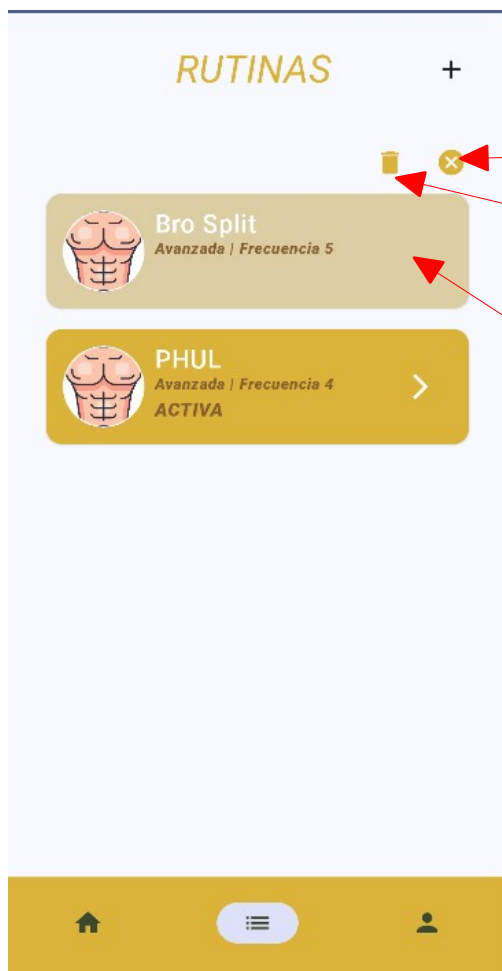




Presionando en este icono podremos seleccionar una imagen guardada en el dispositivo para utilizarla de foto de perfil

Aquí aparecerán los distintos consejos deportivos

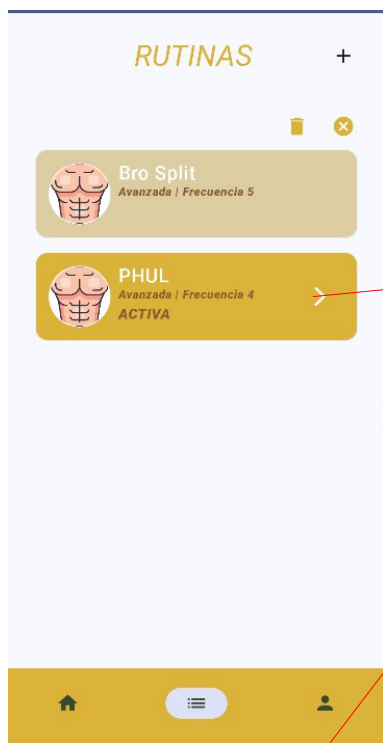
Pulsando individualmente en cada icono, podremos acceder a las distintas pantallas de la aplicación, "Home", "Rutinas" y "Perfil", respectivamente.



El icono de la X, nos permitirá anular la selección

EL cubo de basura permitirá eliminar la rutina seleccionada

Si pulsamos sobre una rutina, cambiará a un amarillo más oscuro y



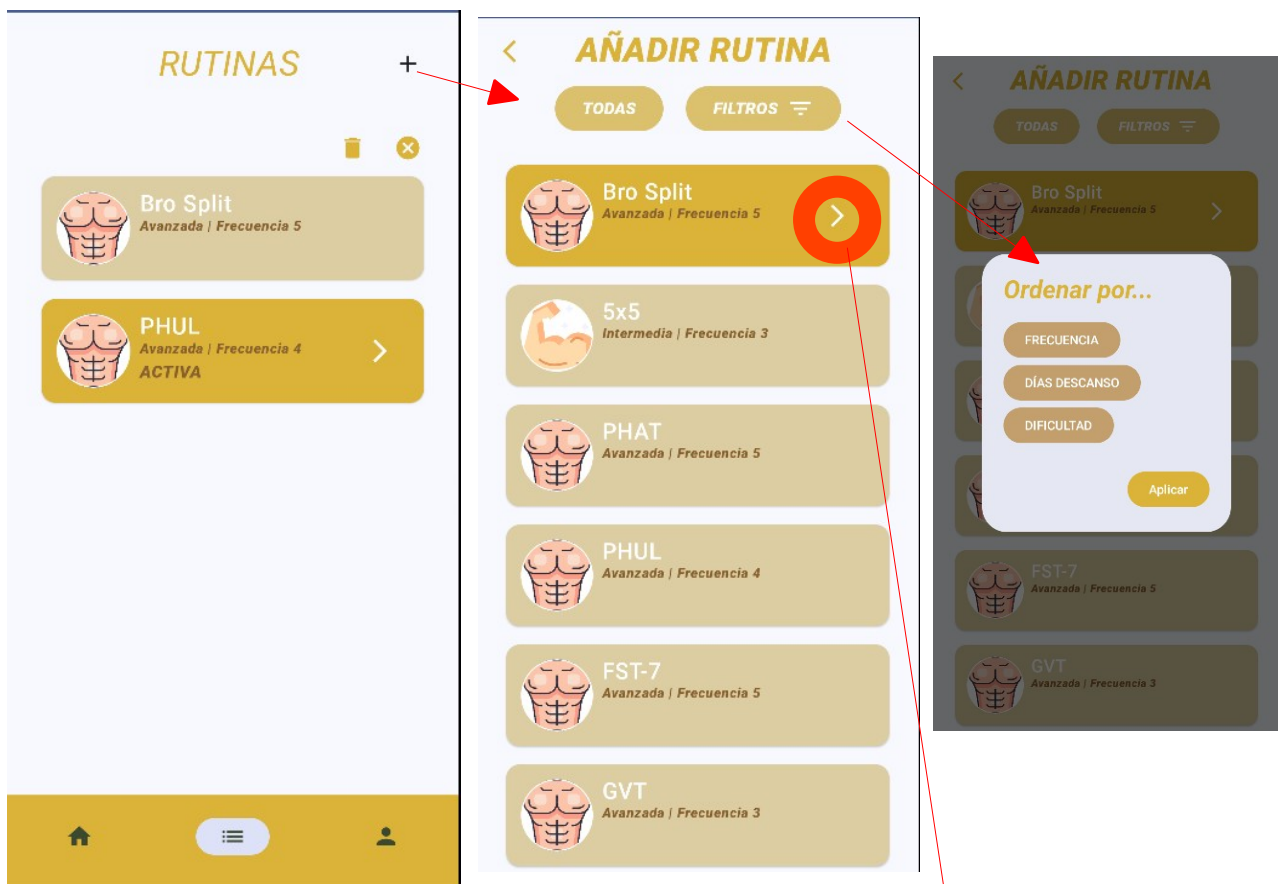
Podemos navegar entre los distintos días de la semana pulsando en los círculos L,M,X,J,V,S,D



Al pulsar sobre la flecha blanca, veremos la rutina y pulsando en desactivar/activar, la marcaremos como activa/inactiva, en el caso de Activarse, esta rutina pasará a ser nuestra rutina actual y se mostrará en la pestaña “VER ENTRENAMIENTO”, de “Home”



Si presionamos sobre cualquiera de los ejercicios, en este caso, jalón al pecho, aparecerá una ventana con una imagen y un texto explicativos



Si presionamos en el símbolo del +, podemos ver todas las rutinas disponibles en nuestro catálogo, pero esta vez para facilitar la búsqueda, hemos facilitado un filtro por dificultad o frecuencia, de la misma manera que explicamos antes, si presionamos en una rutina, cambiará de color y aparecerá una flecha blanca, la cual pulsaremos y nos llevará a la siguiente pantalla, muy similar a la mencionada previamente pero con la diferencia de que esta, tendrá un botón añadir, que añadirá la rutina a nuestra lista personal.

Aquí añadiremos la rutina a nuestra lista personal.



Por último tendremos en el apartado “Perfil”, un formulario que permite cambiar los datos introducidos anteriormente

Si pulsamos este botón, cerraremos sesión con nuestra cuenta.

**SIMPLEFIT**

*Preferecias de usuario*

"

**PESO**

84

**EDAD**

16-28

**SEXO**

Masculino

**SOMATOTIPO**

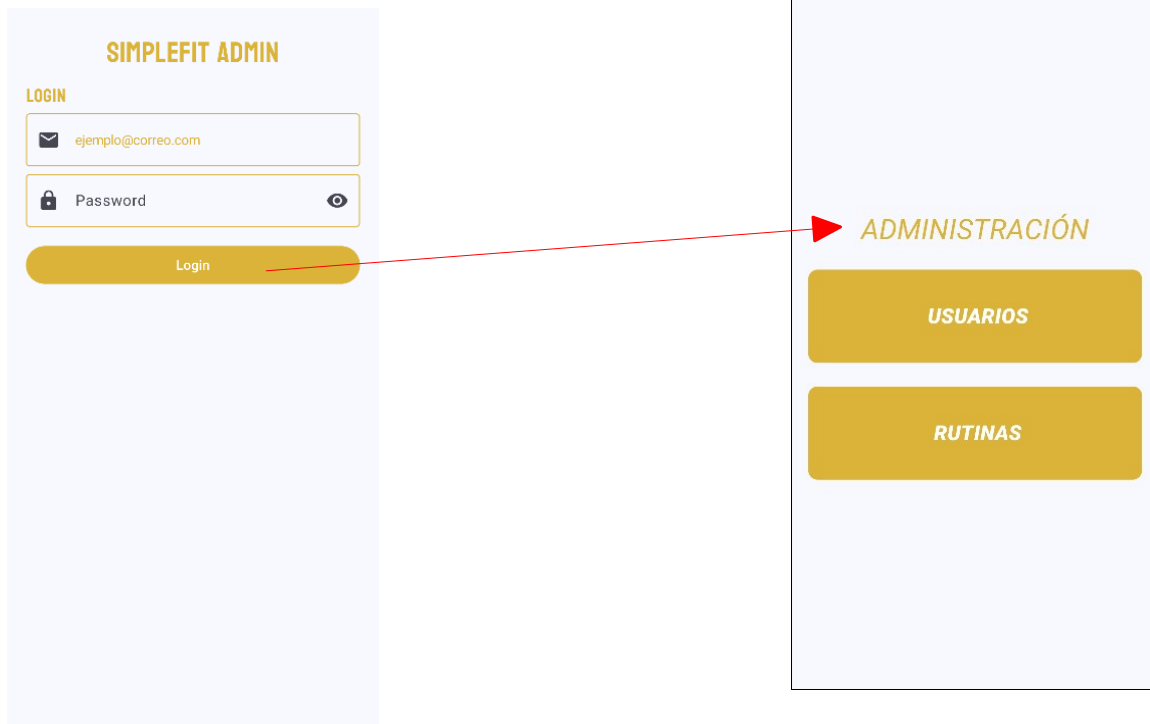
ectomorfo

Guardar preferencias

Home Menu Profile

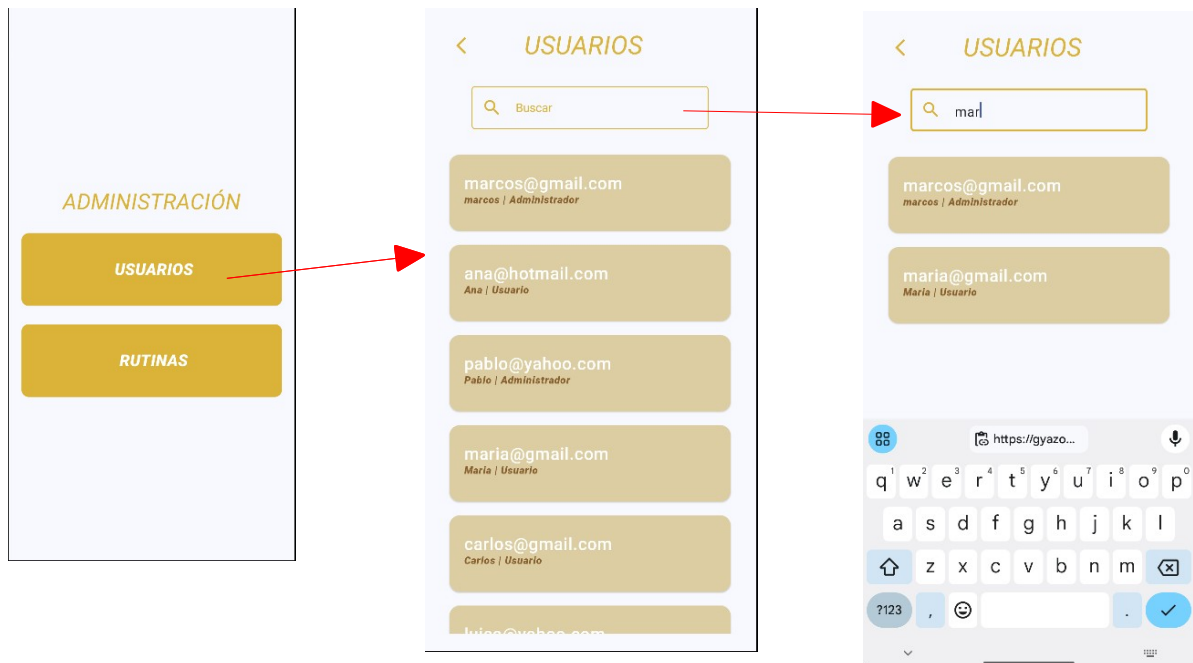


## Aplicación Administradores (SimpleFitAdmin)

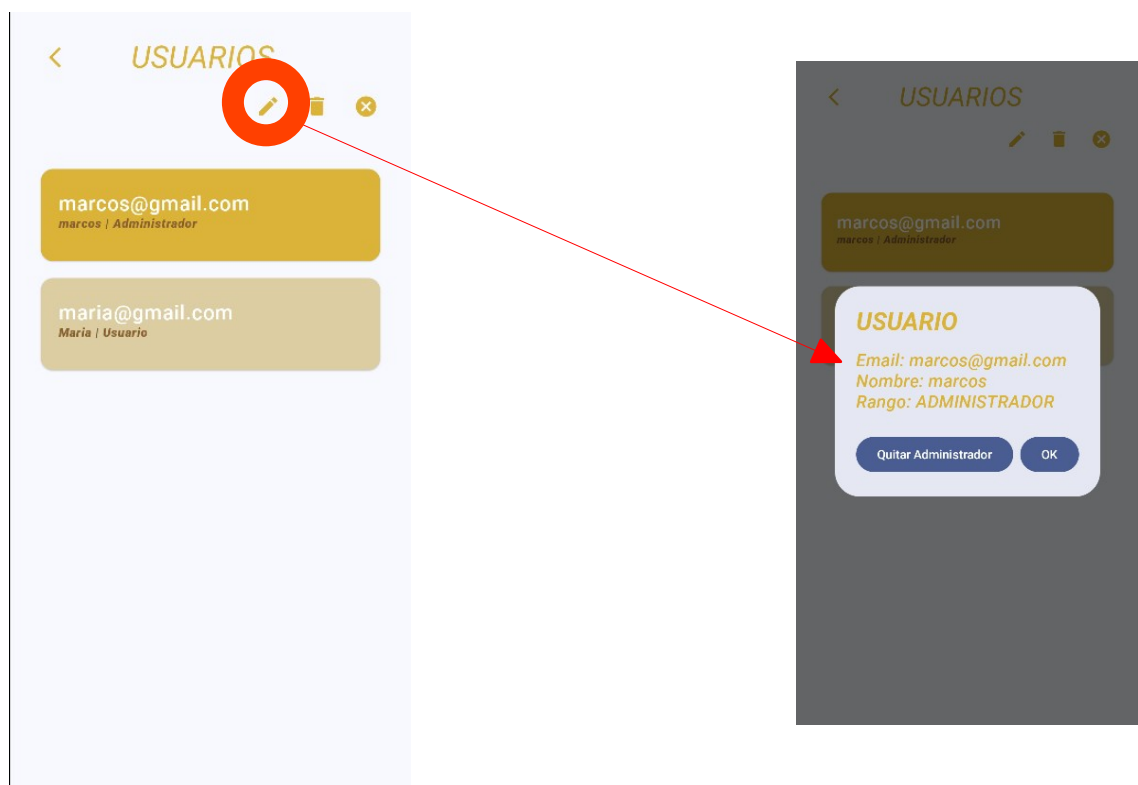


Para acceder a la aplicación introduciremos email y contraseña, de la misma manera que en la aplicación para usuarios, pero en este caso necesitaremos tener permisos de administrador.

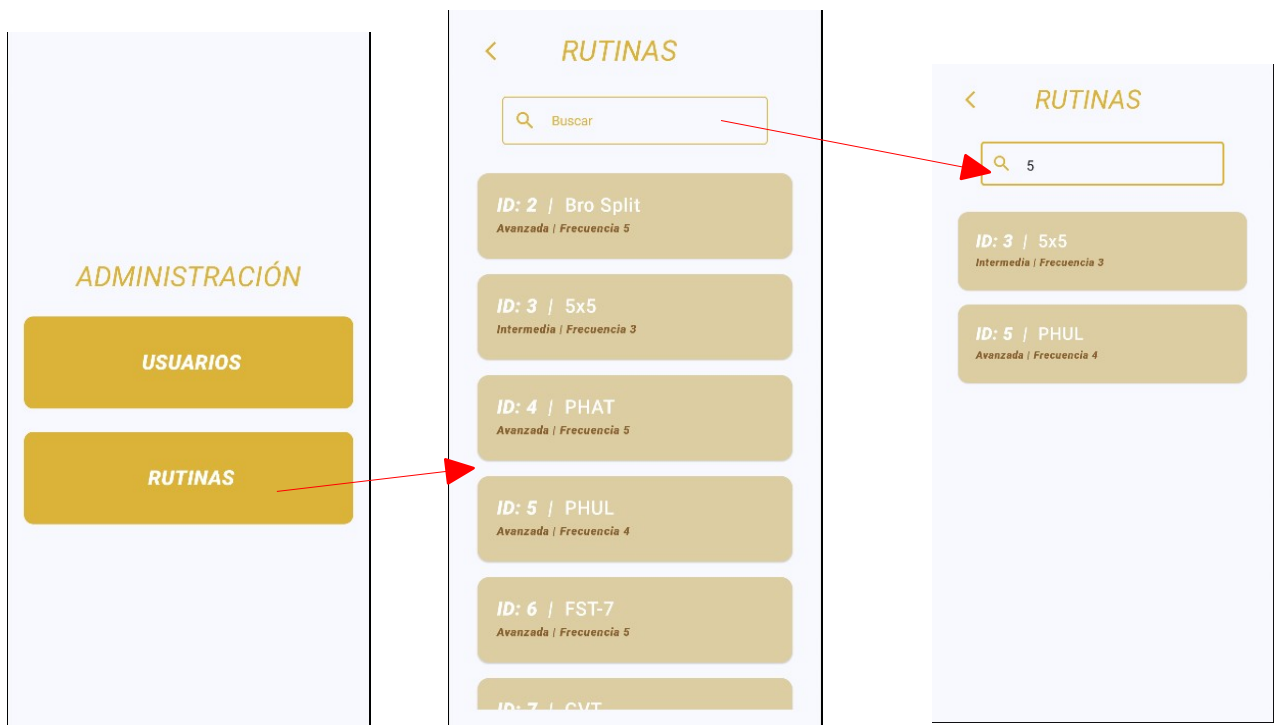
Cuando iniciamos sesión llegaremos a la ventana de “ADMINISTRACIÓN” donde podremos ver dos botones “USUARIOS” y “RUTINAS”



Si pulsamos sobre el botón usuarios veremos la lista de usuarios y podremos realizar búsquedas a través de el formulario de búsqueda situado en la parte superior de la pantalla.



Pulsando sobre el lapiz, nos aparecerá una ventana en la cual podremos dar permisos de administración, la papelera elimina al usuario de la base de datos y el icono con la X quita la selección del usuario.



Al igual que en “USUARIOS”, si pulsamos en rutinas podremos ver todas las rutinas y realizar búsquedas por nombre e ID.



## Requisitos e instalación

Para proceder a la instalación de la aplicación, simplemente deberás tener el archivo APK localizado en tu dispositivo, posteriormente, abre el archivo y continúa con los pasos indicados en el proceso, probablemente sea necesario activar la instalación de aplicaciones desconocidas para continuar, una vez dados los permisos e instalada la aplicación, aparecerá en tu album de aplicaciones para su uso.

No se trata de una aplicación pesada ni requiere alto procesamiento por lo que funcionará correctamente en la mayoría de terminales.

## Conclusiones

Desarrollar Simplefit me ha permitido darme cuenta de muchos aspectos acerca del desarrollo de proyectos, que en un futuro me servirán para optimizar el tiempo y mejorar el proceso de producción.

He aprendido a definir objetivos claros, establecer plazos realistas y distribuir tareas de manera efectiva. La organización del trabajo y la gestión del tiempo han sido cruciales para poder llevar a cabo todas las fases del proyecto, desde el diseño inicial hasta la implementación y pruebas finales.

Destacar también el desarrollo de la capacidad para enfrentar y resolver problemas de manera autónoma. Durante el proyecto, me he encontrado con diversos desafíos técnicos y he aprendido a buscar y encontrar soluciones utilizando múltiples recursos. La habilidad para investigar, evaluar diferentes opciones y aplicar la solución más adecuada es una competencia que he fortalecido considerablemente. Otra opción es acudir a expertos sobre los temas que tratas para que ellos, bajo su experiencia, puedan ayudarte a detectar errores y solucionarlos.

Todos los aprendizajes que he obtenido durante el proceso han sido gracias a los errores que he cometido los cuales tomaré como ejemplo en próximos proyectos.

En resumen, el Trabajo de fin de grado ha sido una gran oportunidad para consolidar mis conocimientos y habilidades en programación y desarrollo de software. Me ha preparado para enfrentar futuros desafíos profesionales con una mentalidad analítica y una sólida base técnica. Estoy convencido de que todo lo aprendido durante este proyecto será de gran utilidad en mi carrera y me permitirá seguir creciendo como desarrollador y profesional en el ámbito tecnológico.

## Posibles ampliaciones y mejoras

Como autocrítica me gustaría destacar la obligación de realizar una planificación previa a comenzar el proyecto tanto de los tiempos de entrega en relación a mi proyecto como de la estructuración del mismo, asignando tareas de forma específica tras el análisis ya que no seguía un plan de desarrollo y no me adelantaba a posibles mejoras o problemas, lo que me hacía perder mucho tiempo dando pasos atrás para solucionar los errores o dudar en implementar.

Para estructurar correctamente el programa, en base a este proyecto, al comenzar el desarrollo entiendo que se debe maquetar la interfaz de usuario y una vez terminada la versión final de esa UI, realizar el diseño de datos, ya que a causa de no elaborarlo de esta manera, he realizado numerosos cambios en la base de datos que ralentizaban el desarrollo de toda la aplicación y considero que habría recuperado mucho tiempo.

En un futuro a corto plazo, me gustaría añadir más rutinas y dividir las por tipos, para que usuarios que no acuden al gimnasio también puedan utilizar la aplicación, como por ejemplo la calistenia, estiramientos...

También sería destacable mejorar el seguimiento de los parámetros de usuario creando diagramas, recomendando rutinas en base a los datos introducidos, cálculo de índice de masa corporal y otros datos biométricos básicos.

## Bibliografía

La gran mayoría de la información de código lo he sacado de los apuntes del curso, ya sea pdf de aulas o proyectos realizados.

brandmark.io → Creación de una plantilla y gama de colores.

chatgpt.com → Consultas generales sobre código en todos los lenguajes.

plantuml.com → Creación de diagramas.

colormind.io → ver gamas de colores de forma avanzada.

<https://www.contadordecaracteres.info/codificador-base64-online-para-incrustar-imagenes-html-css>  
→ para pasar las imágenes a BASE64.

<https://www.uptoplay.net/filemanager.php?username=sb1g2axklv&apkonline=1> → para comprobar si el apk se ejecuta.

Lucid.app → Crear diagrama de arquitectura.

dbdiagram.io → Crear una plantilla de nuestra estructuración de tablas BBDD.

<https://es.vecteezy.com/> → Para las imágenes de los ejercicios y botones.