

# **“DAILYGYM: APP DE ENTRENAMIENTO”**

**PROYECTO DE DESARROLLO. ELABORACIÓN DE UNA  
APLICACIÓN PARA DISPOSITIVO MÓVIL**

**CICLO FORMATIVO DE GRADO SUPERIOR DE DESARROLLO DE  
APLICACIONES MULTIPLATAFORMA**

**PAULA OLIVA ARELLANO**

**ALFONSO GARCÍA**

**Curso 2023-2024**

## ÍNDICE

1. Resumen .....	3
2. Abstract .....	4
3. Introducción .....	5
4. Descripción y Objetivos .....	6
5. Contenidos .....	8
6. Metodología y planificación .....	10
7. Desarrollo .....	17
8. Conclusiones y proyección a futuro .....	24
9. Bibliografía.....	26
10. Anexos.....	28

## 1. Resumen

Desde este proyecto, hemos desarrollado una aplicación móvil orientada al sector fitness para crear rutinas personalizadas, donde el usuario podrá definir aspectos clave de su entrenamiento como el objetivo de la rutina, los días que la conforman y la estructuración de ejercicios por día. Asimismo, se han incluido las funcionalidades necesarias para registrar los detalles de cada entrenamiento, con el propósito de que el usuario pueda consultar dichos registros y ajustar sus objetivos diarios en consecuencia. El contenido de la aplicación se ha estructurado en tres secciones principales (Perfil, Rutinas y Registros), que presentan los datos correspondientes de forma clara y organizada. Esto permite al usuario identificar fácilmente el cometido de cada apartado, proporcionando una experiencia fluida al evitar la sobrecarga de información. La persistencia se ha realizado con almacenamiento local mediante el empleo de una base de datos SQLite y SharedPreferences. Hemos optado por una navegación basada en el reemplazo de fragmentos, ofreciendo información más específica a medida que avanzamos: empezamos visualizando la lista de rutinas creadas, seleccionamos aquella que queremos ver en detalle, elegimos un día de entreno, vemos los ejercicios asociados a ese día y, por último, clicamos en un día para ver los registros realizados. Nos situamos desde un marco metodológico alineado con prácticas ágiles de desarrollo; concretamente, nos hemos basado en la planificación a través de sprints que propone la metodología SCRUM. La implementación de clases se ha ajustado a la arquitectura propuesta por el patrón de diseño Modelo-Vista-Controlador (MVC), que nos permite comprender las responsabilidades asumidas por cada una de ellas. Como entorno de desarrollo recurrimos a Android Studio y como lenguaje de programación hemos empleado Java. La aplicación se destina, por tanto, a dispositivos Android que cuenten, como mínimo, con una API 24 para poder ser compilada y ejecutada de forma satisfactoria. Como proyección futura de la aplicación, nos proponemos permitir su escalabilidad y mejora de accesibilidad mediante el uso de almacenamiento en la nube, incorporar funcionalidades avanzadas gracias a algoritmos de inteligencia artificial y buscar la optimización de diseño de la interfaz, logrando una aplicación completa que otorgue beneficios significativos a los usuarios.

*Palabras clave:* aplicación, fitness, entrenamiento, rutinas, ejercicios, registros.

## 2. Abstract

In this project, we have developed a mobile application oriented towards the fitness sector to create personalized routines. Users can define key aspects of their training, such as the goal of the routine, the training days and the structure of exercises for each day. Additionally, we have included the necessary functionalities to record the details of each workout, allowing users to consult these logs and adjust their daily objectives accordingly. The content of the application is structured into three main sections (Profile, Routines, and Logs), showing the data clearly and organized. This allows users to easily identify the purpose of each section, providing a smooth experience by avoiding information overload. Data persistence is handled through local storage using an SQLite database and SharedPreferences. We have opted for navigation based on fragment replacement, providing more specific information as users progress: starting with a view of the list of created routines, selecting one for detailed viewing, choosing a training day, viewing the exercises associated and, finally, clicking on a day to see the recorded logs. We adopted a methodological framework aligned with agile development practices, specifically using sprint planning as proposed by the SCRUM methodology. The implementation of classes adheres to the architecture proposed by the Model-View-Controller design pattern (MVC), allowing us to clearly understand the responsibilities assumed by each class. As our development environment, we used Android Studio, and Java was employed as the programming language. Therefore, the application is intended for Android devices with at least API 24 to ensure proper compilation and execution. Looking to the future, our projection for the application includes enabling scalability and improving accessibility through the use of cloud storage, incorporating advanced functionalities thanks to artificial intelligence algorithms, and striving for optimization in interface design, creating a comprehensive application that provides significant benefits to users.

*Keywords:* application, fitness, training, routines, exercises, logs.

### 3. Introducción

La presente aplicación tiene como propósito ofrecer un software que permita llevar a cabo un registro diario de entrenamiento, caracterizándose por la facilidad de uso y tratamiento por parte del usuario. Esto significa poder plasmar con detalle aspectos importantes en el desarrollo de un buen plan de entrenamiento como la estructuración de los días semanales destinados al trabajo o la distribución de ejercicios.

El sector del fitness está en una tendencia creciente; cada vez más personas tratan de incluir en su día a día rutinas de entrenamiento orientadas a diversos fines, como pueden ser aumentar la masa muscular, potenciar la fuerza máxima, mejorar la resistencia cardiovascular u optimizar la pérdida de grasa. En cada caso, elaborar un registro de los entrenamientos constituye una base para plantear objetivos de progreso a medio y largo plazo.

Actualmente, existen multitud de aplicaciones relacionadas con la actividad física y el deporte, la mayoría de ellas centradas en propuestas de ejercicios y rutinas ya estructuradas. En el caso de aquellas que funcionan como un diario de entrenamiento, encontramos aplicaciones que convierten esta tarea en un acto tedioso y acaban por no ser utilizadas con la recurrencia necesaria. Según nuestra propia experiencia, no cumplen con las labores de anotar el entrenamiento realizado de forma simple, poder consultar el entrenamiento de días anteriores y realizar una comparativa para ajustar en ese momento el tipo de entreno. Pongamos un ejemplo.

Imaginemos que un usuario realiza en un día de entrenamiento, un ejercicio de sentadilla con barra con 50 kg de peso, 3 series de 8 repeticiones. Disponer de este diario de entrenamiento y poder consultarlo en el momento de repetir esa rutina, permitirá establecer nuevos objetivos y ver qué tan capaz es de conseguirlos.

En síntesis, la propuesta pretende desarrollar una aplicación que permita registrar y llevar a cabo un seguimiento de los propios entrenamientos realizados, obteniendo un diario de sesiones de forma rápida y sencilla para que el usuario pueda revisar y consultar, sin invertir demasiado tiempo, las últimas sesiones de forma simultánea a la ejecución de su actual entrenamiento.

## 4. Descripción y Objetivos

Pretendemos desarrollar una aplicación orientada al registro de los entrenamientos realizados para su posterior consulta y seguimiento en la planificación de la mejora de la condición física. Su funcionalidad esencial será, por tanto, ofrecer un medio de anotaciones que permita la estructuración personalizada de rutinas, pudiendo seleccionar los días de los cuales dicha rutina se compone, qué tipo de entrenamiento se llevará a cabo cada día y en qué consistirá, detallando cada ejercicio realizado.

Para ejemplificarlo, el usuario podrá elaborar una rutina individual (o varias) donde reflejar los detalles de la misma para poder identificarla, como el nombre y una pequeña descripción a modo de información adicional, y qué días conformarán dicha rutina. Al plasmarlo en la aplicación, obtendría una vista similar a la de la Figura 1.

Figura 1

*Wireframing interfaz de usuario*



*Nota.* Elaboración propia

Habiendo estructurado la rutina, al seleccionar cada día podremos incluir los ejercicios que realizamos en cada sesión, con registros de cada uno de ellos donde especificaremos número de series, repeticiones y el peso cargado.

Como mencionábamos, la aplicación se dirige a todo tipo de usuarios que incluyan el entrenamiento físico como parte de su día a día, principalmente a aquellos enfocados en una rutina constante y regular con la meta de progresar hacia un objetivo de los ya mencionados. Aquellas personas que realicen actividad física, pero entre sus objetivos no esté registrar sus marcas o evaluar su rendimiento, no sacarán el máximo provecho al uso de la aplicación.

Siguiendo con su distribución, la aplicación podrá encontrarse disponible para descargar e instalar a través de Play Store, la plataforma desarrollada por Google para la distribución de aplicaciones de diversa índole que se encuentra disponible para cualquier dispositivo que cuente con un sistema operativo Android (Tito Durand et al., 2023).

En referencia a los objetivos del desarrollo de este proyecto, definimos como objetivo principal proporcionar un medio útil de registro y consulta de los entrenamientos realizados que el usuario pueda incluir en su rutina diaria de forma rápida, sencilla e intuitiva. Partiendo de esta idea general, establecemos una serie de objetivos específicos que se detallan en la Tabla 1.

Tabla 1

*Objetivos específicos*

1. Desarrollar una aplicación funcional de registro de rutinas y ejercicios.
2. Diseñar una interfaz atractiva y dinámica que facilite la experiencia del usuario.
3. Permitir la consulta inmediata de las propias rutinas realizadas.
4. Facilitar el seguimiento del progreso conseguido en cada ejercicio.
5. Minimizar el tiempo dedicado al registro de los ejercicios ejecutados.
6. Incluir el uso de la aplicación como herramienta complementaria durante el desarrollo de los entrenamientos.

## 5. Contenidos

El conjunto de vistas que constituyen la aplicación y que mencionamos en este apartado se puede consultar en el Anexo 1.

La aplicación diseñada cuenta con tres secciones principales: Perfil, Rutinas y Registros. Desde el perfil (Figura 2), el usuario puede introducir datos referentes a sus características individuales: nombre, edad, peso, altura, sexo y objetivo. En la versión actual de la aplicación, el perfil almacena la información que proporciona el usuario, pudiéndola modificar en cualquier momento. En versiones más avanzadas, los datos recogidos en esta sección podrán utilizarse para ofrecer una experiencia de usuario más personalizada, con sugerencias y orientaciones que sirvan de ayuda en la consecución de los objetivos personales.

Por otro lado, la sección de rutinas cuenta con una serie de funcionalidades que describimos a continuación e integra el propósito fundamental de la aplicación. En la vista (Figura 3), encontramos tres partes diferenciadas: en primer lugar, nos muestra la rutina seleccionada como predeterminada, es decir, aquella que estemos siguiendo actualmente; después, podemos visualizar la lista de rutinas que hemos creado; y, por último, encontramos el botón “+ Crear Rutina”. Existen dos elementos interactivos en esta interfaz: el texto “Tus Rutinas” y el botón, los cuales nos dirigen a la sección Tus Rutinas (Figura 4) y Crear Rutina (Figura 5), respectivamente.

Para crear una rutina introducimos un nombre, descripción, los días que formarán la rutina y para quién será, ya que es posible que queramos diseñar una rutina para otra persona que no seamos nosotros mismos. Las rutinas que creemos desde esta sección serán las que conformen la lista de rutinas.

Desde Tus Rutinas podemos visualizar las rutinas que hemos creado con la información mencionada anteriormente (nombre, descripción y días de entreno). Además, disponemos de un botón situado a la derecha para borrar dicha rutina si lo deseamos. Podemos también clicar sobre cada una de las rutinas, avanzando hacia otra vista (Figura 6) donde encontramos el detalle de la rutina. Desde esta interfaz, podemos seleccionar cada día de entreno para configurarlo añadiendo ejercicios y establecer la rutina como predeterminada con el botón “Seguir rutina”, siendo esta la



que nos aparezca en la sección principal Rutinas tras el texto de “Tu rutina actual”. Al clicar sobre un día de entreno, avanzamos hacia la siguiente vista (Figura 7), donde tenemos la opción de agregar o eliminar ejercicios y visualizamos el nombre de los ejercicios que hayamos agregado.

Para agregar un ejercicio, clicamos en el botón y navegamos hacia el siguiente apartado (Figura 8). Introducimos nombre, descripción y músculo o grupo muscular al que enfocamos el trabajo y clicamos en el botón Guardar. Para eliminar ejercicios, marcamos el checkbox del ejercicio que queremos eliminar y confirmamos (Figura 9).

Finalmente, al clicar sobre un ejercicio concreto, avanzamos hacia la vista en detalle del ejercicio (Figura 10), donde podemos consultar las anotaciones registradas con detalles de fecha, peso, repeticiones y series, de más reciente a más antigua, realizar nuevos registros (Figura 11) o eliminar los existentes (Figura 12).

El último de los apartados principales es el de Registros (Figura 13), destinado a visualizar de forma rápida la lista de todos los ejercicios que hayamos agregado y los registros realizados en cada uno de ellos. Esta sección nos permite consultar directamente cada ejercicio a través de un desplegable, sin tener que navegar por todos los apartados mencionados en la sección Rutinas.

Para manejar y dar persistencia a toda esta información referente a las rutinas creadas, ejercicios agregados y registros realizados, trabajamos con una base de datos SQLite, estructurando los datos en tres tablas: rutinas, ejercicios y registros (ver Anexo 2). Por su parte, el almacenamiento de los datos personales en la sección Perfil se gestiona con SharedPreferences.

## 6. Metodología y planificación

Para determinar la metodología empleada, indagamos entre la variedad de alternativas actuales. La elección de esta, tal como indican Estrada-Velasco et al. (2021), depende de factores como el tipo de proyecto, el entorno de desarrollo o las demandas del cliente. Según Rivas et al. (2015), las metodologías se agrupan en clásicas, orientadas a objetos, ágiles, formales, para la web y otras metodologías.

Teniendo en cuenta lo anterior, y ante la necesidad de que el proceso de desarrollo sea flexible, nos inclinamos hacia la propuesta de las metodologías ágiles, las cuales se caracterizan por la “capacidad de adaptación a los requerimientos que surgen en el desarrollo” (Velásquez Restrepo et al., 2019, p.16).

Según respaldan autores como Szalvay (2004), Herrera Uribe y Valencia Ayala (2007) o Mendes Calo et al. (2010), las metodologías ágiles se fundamentan en los principios declarados en el Manifiesto Ágil, un documento creado por expertos de desarrollo de software en el año 2001. Estas metodologías están indicadas para aquellos proyectos donde el producto final es difícil de detallar pormenorizadamente al inicio del proceso, siendo más revelador recurrir a una continua retroalimentación durante el desarrollo para reconocer las necesidades reales del software.

En nuestro caso, durante el proceso, realizamos revisiones del estado del proyecto al completar iteraciones que dan como resultado un producto entregable, sobre el que incorporar mejoras y nuevos componentes funcionales. Obtenemos así un desarrollo progresivo de las partes constitutivas del software.

Dadas las circunstancias, no contamos con un equipo de desarrollo. Esto nos lleva a no poder situarnos desde un marco metodológico estricto; en su lugar, realizamos una combinación de prácticas ágiles. Teniendo en cuenta esta salvedad, orientamos el proyecto a través de la metodología SCRUM que, según indica López Menéndez de Jiménez (2015, p.8), “emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo”. Estos ciclos iterativos e incrementales se conocen como *sprint* y permiten el desarrollo progresivo de la aplicación (Amaya Balaguera, 2013).

Molina Ríos et al. (2021) sitúan a SCRUM como la metodología más adecuada para el desarrollo de aplicaciones móviles. Estos autores identifican las siguientes fases en el desarrollo del software: planificación de sprint, etapa de desarrollo, revisión de sprint y retroalimentación.

Volviendo la vista a nuestro proyecto, podemos seguir la línea de evolución de la aplicación a través de los sprints recogidos en el Anexo 3. Hablaremos entonces de las tareas llevadas a cabo tomando como base esta estructura de sprints.

### **Secciones principales**

El primer objetivo del diseño es estructurar las tres secciones principales: Perfil, Rutinas y Registros. Se representan a través de fragmentos que permiten mantener como elementos comunes el menú de navegación y la barra superior. Para manejar la navegación entre estos tres fragmentos, configuramos un componente de tipo `BottomNavigationView` en `MainActivity`. De esta manera, pretendemos hacer siempre accesible para el usuario el movimiento entre estas secciones.

Dado que la funcionalidad más importante de la aplicación la vamos a integrar en la sección Rutinas, definimos la visualización de esta por defecto al iniciar la aplicación.

Comprobamos que la navegación está correctamente configurada, pero observamos un comportamiento indeseado: al reemplazar un fragmento, este se añade a la pila `backstack` y, si clicamos en el botón atrás de nuestro dispositivo, en lugar de salir de la aplicación, recorremos toda la pila. Es por esto que incluimos un nuevo parámetro booleano en el método `replaceFragment` de `MainActivity` para poder decidir si queremos que el fragmento se añada o no a la pila.

Como hemos mencionado anteriormente, la sección Rutinas concentra la mayor labor de desarrollo del proyecto, por lo que el siguiente sprint se enfoca en su estructuración.

### **Sección Rutinas**

El diseño visual de este fragmento queremos que conste de tres partes: la rutina actual, la lista de rutinas creadas y un botón para crear nuevas rutinas. Decidimos utilizar un componente `CardView` para agrupar los detalles de la rutina seleccionada

como predeterminada, un ListView para la lista de rutinas y un elemento Button para manejar la creación de rutinas.

Queremos comprobar si este diseño se ajusta a las expectativas o si, por el contrario, necesitamos modificar o añadir algún elemento. Decidimos entonces implementar rutinas de forma estática para poder visualizar en la lista de rutinas. Creamos las clases java Rutinas y DiasEntreno y el adaptador correspondiente.

Observamos que las rutinas se visualizan correctamente, pero el desplazamiento en la pantalla no es como nos gustaría, ya que el movimiento de scroll solo se realiza en el elemento ListView y buscamos que este desplazamiento involucre a toda la vista. Sustituimos entonces el ListView por un RecyclerView y conseguimos este comportamiento.

Ya teniendo creada nuestra clase Rutinas (idRutina, nombreRutina, descripcionRutina, diasEntreno), pasamos a diseñar la actividad que permita al usuario crear sus propias rutinas

### **Crear Rutina**

En este caso, decidimos integrar la funcionalidad en una actividad independiente, al tratarse de una tarea específica que se desarrolla exclusivamente en esta vista. Incluimos elementos de tipo EditText para el nombre y la descripción y una serie de checkboxes para seleccionar los días de la rutina.

Una vez tenemos el diseño para crear rutinas, avanzamos hacia uno de nuestros objetivos principales: almacenar y recuperar las rutinas creadas.

### **Persistencia de rutinas**

Para iniciar el trabajo de persistencia, creamos una clase BaseDatos que extiende de SQLiteOpenHelper. Definimos las columnas de la tabla rutinas, creamos la tabla y un método para insertar las rutinas, al cual pasamos como parámetro un objeto Rutinas.

En la actividad Crear Rutina, obtenemos los datos introducidos en los elementos EditText y los checkboxes de los días de entreno, cuya selección manejamos con el adaptador MyAdapter. Para evitar que se guarden registros con campos vacíos (por

ejemplo, si no se introduce nada en el nombre), se realiza una comprobación antes de llamar al método para guardar la rutina y se muestra un mensaje Toast. Si, por el contrario, se realiza correctamente la inserción en la tabla rutinas de la base de datos, aparece un diálogo de confirmación informando de que el proceso de guardado ha sido exitoso.

## **Configurar listado de rutinas**

Para ello, utilizamos el listado de rutinas de la sección Rutinas donde hasta ahora obteníamos la información de manera estática.

Definimos en BaseDatos un método para recuperar las rutinas. Modificamos el adaptador y el RecyclerView que teníamos definidos y comprobamos que la información se almacena y recupera de forma satisfactoria.

Conseguido el objetivo del sprint, reparamos en que el diseño visual es un tanto escaso, puesto que hasta ahora solo estamos trabajando con cadenas de texto. Es por esto que tomamos la decisión de incluir una imagen asociada a la rutina. En Crear Rutina, añadimos un RadioGroup para determinar quién está creando la rutina, y en la clase Rutinas definimos un campo nuevo: autorRutina (asociaremos una imagen si el autor es Hombre y otra si el autor es Mujer). Hacemos las modificaciones pertinentes en BaseDatos para almacenar esta selección.

Sin embargo, el proceso de guardar la rutina nos da error. Descubrimos que, al realizar una modificación en la estructura de la base de datos, tal como incluir una columna en la tabla rutinas, debemos actualizar la versión de la misma, si no, no detecta la nueva columna. Por último, añadimos un ImageView en la lista rutinas para visualizar la imagen asociada.

El siguiente paso es configurar un apartado donde mostrar la información completa de las rutinas (nombre, descripción y días), además de permitir su eliminación.

## **Sección Tus Rutinas**

Creamos un nuevo fragmento al cual accedemos clicando en el texto “Tus Rutinas” de la sección Rutinas. En él, incluimos un RecyclerView que manejamos con el adaptador RutinasAdapter. Cada rutina se representa con un CardView que contiene

el nombre de la rutina, la descripción y los días de entrenamiento, además de un botón para eliminar la rutina. Incluimos, por tanto, un nuevo método en BaseDatos que borre registros de la tabla rutinas pasando como parámetro el id de la rutina.

El sentido de esta nueva sección es poder acceder a la lista de rutinas en detalle, además de, como hemos mencionado, manejar su eliminación, ya que desde la sección principal solo obtenemos su visualización, sin contar con elementos interactivos.

Obtenida la lista de rutinas y habiendo comprobado que su borrado se realiza correctamente, configuramos la navegación hacia la vista de una rutina concreta.

### **Fragmento Detalle Rutina**

Creamos un nuevo fragmento donde incluimos un CardView para mostrar nombre y descripción de la rutina y un LinearLayout para cargar de forma dinámica los días de entreno. En este caso, necesitamos transmitir información del fragmento anterior al actual, para saber sobre qué rutina se ha hecho clic.

Asignamos entonces el evento onClick a cada elemento rutina mostrado en Tus Rutinas y configuramos el método newInstance del nuevo fragmento para recibir como parámetro un objeto rutina, que el adaptador identifica por su posición en la lista.

Conseguido esto, añadimos un Button con el texto “Seguir Rutina” cuya función es guardar esta rutina como predetermina. Se mostrará en la sección principal Rutinas. Esto se consigue gracias a SharedPreferences, que guarda el id de la rutina.

### **Detalle Día Entreno**

El procedimiento es muy similar al descrito en el anterior sprint: creamos el nuevo fragmento (Detalle Día), diseñamos su vista, asignamos el evento onClick a cada día de entreno mostrado y transmitimos la información que necesitamos reutilizar en este nuevo fragmento.

Comprobamos que la información se transmite correctamente añadiendo algunos elementos TextView para mostrar datos (por ejemplo, el nombre de la rutina) que luego decidimos suprimir por considerar redundantes.

## **Agregar y Eliminar Ejercicios**

Creamos una nueva clase, Ejercicios, con los campos idEjercicio, nombreEjercicio, descripcionEjercicio y musculoPrincipal. En el fragmento Día Entreno, añadimos un botón para agregar y otro para eliminar ejercicios. Creamos un nuevo fragmento al cual navegamos clicando en “Agregar”, con elementos EditText para nombre, descripción y músculo principal, y un botón para guardar el ejercicio. Transmitimos idRutina e idDiaEntreno al crear una instancia de este fragmento, para asociar cada ejercicio a una rutina y un día concretos.

Incluimos la nueva tabla ejercicios en BaseDatos y añadimos un método para insertar y eliminar ejercicios. Al clicar en el botón de guardar, se comprueba que no hay ningún campo vacío y se procede a insertar el nuevo registro en la tabla. Configuramos un mensaje Toast para indicar que el guardado se ha realizado con éxito.

Una vez podemos agregar ejercicios, necesitamos visualizarlos. En el fragmento Día Entreno, añadimos un RecyclerView para cargar los ejercicios asociados. Creamos un método en BaseDatos para obtener los ejercicios y configuramos el adaptador. Para eliminar ejercicios, mostramos un AlertDialog con una lista de checkboxes para seleccionar los ejercicios que queremos eliminar.

## **Detalle Ejercicio**

La navegación hasta este nuevo fragmento se realiza de forma idéntica a la descrita en puntos anteriores. En esta vista, mostramos el nombre del ejercicio, su descripción y músculo principal, información que obtenemos al pasar como parámetro el objeto ejercicio al crear la instancia de este fragmento. Incluimos un botón para crear los nuevos registros y un RecyclerView donde cargar los registros almacenados.

## **Agregar Registros**

Creamos una nueva clase Registro con los campos idRegistro, peso, repeticiones, series y fecha. También necesitamos incluir una nueva tabla en BaseDatos, así como los métodos para insertar y eliminar registros y obtener los registros por ejercicio.

Configuramos el evento onClick del botón “Nuevo Registro” para mostrar un DialogFragment donde introducir peso, repeticiones y series. El botón de guardar de

este diálogo realiza una nueva inserción en la tabla registros, asegurando que todos los campos estén completos. La fecha se obtiene recurriendo a la clase SimpleDateFormat. Se muestra un Toast si el registro se realiza correctamente.

## **Recuperar Registros**

Definimos un método en BaseDatos para obtener los últimos 10 registros almacenados, en orden descendente de inserción, pasando como parámetro el id del ejercicio. Creamos y configuramos el adaptador RegistrosAdapter para poder visualizar la lista de registros recuperada en el RecyclerView del fragmento Detalle Ejercicio. Para eliminar registros, incluimos un evento onClick sobre cada elemento de la lista, mostrando un DialogFragment de confirmación.

Además de visualizar los registros en este apartado, contamos con la sección principal de Registros, la cual nos disponemos a configurar. Para este cometido, creamos dos nuevos adaptadores: AllEjerciciosAdapter y AllRegistrosAdapter.

En este caso, incluimos un RecyclerView para mostrar la lista de todos los ejercicios almacenados en la tabla ejercicios de la base de datos, con independencia del día de entrenamiento o rutina a la que pertenezcan, por lo que añadimos un nuevo método en BaseDatos. Cada ejercicio aparece representado en un elemento CardView que despliega u oculta sus registros asociados. Esto se consigue con un evento onClick sobre cada ítem que alterna la visibilidad de los registros. Igualmente, incluimos un TextView que se mostrará en caso de que no hay ningún registro.

## **Sección Perfil**

Para finalizar con el diseño de la aplicación, tenemos que estructurar la sección del Perfil, cuyo propósito es guardar los datos personales del usuario.

Creamos una nueva clase UserProfile con los campos nombre, edad, peso, altura, sexo y objetivo. En la vista del fragmento, incluimos elementos EditText para introducir estos datos, excepto para sexo y objetivo, para los cuales empleamos un Spinner. También añadimos un botón de guardar para dar persistencia a esta información. Al clicar en él, se crea una nueva instancia de UserProfile la cual se almacena y recupera mediante la clase PreferenceManager que recurre a SharedPreferences.



## 7. Desarrollo

En este apartado, abordamos el desarrollo del proyecto, enfocándonos en la arquitectura que integra las diferentes partes involucradas en la aplicación. Reparamos en cómo se comunican e interaccionan los elementos que constituyen y aportan la funcionalidad deseada a nuestro programa.

Nos basamos en el patrón MVC, el cual, según autores como Bascón Pantoja (2004), Sokolova et al. (2013) y Rodríguez Vega (2020), permite agrupar los componentes de la aplicación en tres conjuntos claramente definidos, atendiendo a su responsabilidad, propósito y función: modelo, vista y controlador. Así, la capa modelo va a contener todas las clases que representan y encapsulan la información; la vista es la capa encargada de presentar al usuario los datos recogidos en el modelo; y, por último, el controlador maneja el flujo de la aplicación en respuesta a las acciones del usuario, pudiendo modificar el modelo y comunicarse con la vista, es decir, coordina la interacción entre el modelo y la vista. En el Anexo 4, incluimos la distribución de las clases de nuestra aplicación en los grupos mencionados.

Cabe mencionar que, en el contexto de una aplicación Android, requerimos cierta flexibilidad de la arquitectura MVC, dado que existen clases que asumen responsabilidades tanto de la vista como del controlador.

Centrándonos en la capa modelo, las clases que la forman son las siguientes:

**Rutinas:** representa las rutinas de entrenamiento y permite gestionar los objetos rutina que se crean en la aplicación.

**DiasEntreno:** representa cada día de entrenamiento. Como campo de la clase Rutinas, tenemos una colección ‘List’ de días de entreno.

**Ejercicios:** define las características particulares de cada ejercicio.

**Registro:** representa los registros que se pueden realizar para cada ejercicio.

**UserProfile:** representa las características de un perfil de usuario, incluyendo información personal y objetivos de entrenamiento.

**BaseDatos:** configura la estructura adecuada de los datos a la hora de ser almacenados. Esta clase maneja la persistencia de las rutinas y sus respectivos días de entreno, los ejercicios y los registros. Igualmente, define los métodos para el guardado y recuperación de la información, asegurando que los datos sean gestionados de manera eficiente.

**PreferenceManager:** representa una capa de abstracción sobre cómo se almacenan y recuperan los datos del perfil de usuario.

Estas clases encapsulan la información y proporcionan métodos de acceso (getters) y de modificación (setters) que nos permiten manejar y presentar los datos de forma eficiente.

Para el resto de clases, especificamos si actúan como vista (V), como controlador (C), o ambas (VC).

### **MainActivity (C)**

Actúa como controlador principal de la aplicación. Coordina la navegación entre los fragmentos y actualiza la UI según las acciones del usuario, como la respuesta a las selecciones del menú de navegación inferior. Aporta el método `replaceFragment` y actualiza el texto de la barra superior.

### **PerfilFragment (VC)**

Presenta los datos personales del perfil de usuario. Los elementos de la interfaz de usuario (en adelante, UI) son de tipo `EditText`, `Spinner` y `Button`. Permite configurar adaptadores `ArrayAdapter` para cada `Spinner`. Gestiona la interacción del usuario y se comunica con el modelo; maneja los eventos de clic en el `Button` “Guardar Perfil”, interactúa con `PreferenceManager` para guardar y recuperar los datos del perfil y actualiza los elementos de la UI con esta información almacenada.

### **RutinasFragment (VC)**

Se encarga de presentar información relacionada con las rutinas del usuario. Como elementos de la UI, cuenta con un `CardView` para la rutina principal y un `RecyclerView`

para la lista de rutinas, cuya disposición de elementos se configura a través de la clase Adaptador. También encontramos elementos TextView y un Button.

Igualmente, maneja los eventos de clic del botón “Crear Rutina” y el TextView “Tus Rutinas”. Emplea métodos de la clase BaseDatos para obtener información y actualizar los elementos de la UI. Desde SharedPreferences, obtiene la rutina principal.

### **RegistrosFragment (VC)**

Presenta el listado de ejercicios almacenados a través de un RecyclerView, configurando el adaptador AllEjerciciosAdapter. Además, maneja la lógica de obtención de información relacionada con los ejercicios desde BaseDatos.

### **CrearRutina (VC)**

Presenta la UI necesaria para crear una nueva rutina. Incluye campos de entrada (EditText), RecyclerView para mostrar las opciones de selección de días de entreno, RadioGroup para elegir autor de la rutina, ImageButton para volver atrás y el Button “Guardar Rutina”.

A su vez, maneja la lógica de creación de nuevas rutinas, incluye validaciones de entrada de datos y gestiona los eventos de clic. Interacciona con el modelo a través de BaseDatos para guardar una nueva rutina.

### **TusRutinasFragment (VC)**

Muestra la lista de rutinas a través de un RecyclerView, utilizando el adaptador RutinasAdapter. Asimismo, maneja la obtención y eliminación de rutinas, interactuando con BaseDatos y actualiza la vista cuando una rutina es eliminada. Maneja la navegación hacia el fragmento DetallesRutinaFragment.

### **DetallesRutinaFragment (VC)**

Presenta los detalles de la rutina (nombre, descripción) mediante CardView y TextView y los días de entrenamiento de forma dinámica a través de un RecyclerView. Incluye un elemento Button para establecer dicha rutina como predeterminada. Gestiona el evento del botón “Seguir Rutina” para establecer dicha rutina como la

principal, actualizando SharedPreferences. Gestiona la navegación hacia el fragmento DetallesDiaEntrenoFragment.

### **DetallesDiaEntrenoFragment (VC)**

Presenta los detalles de un día de entreno. Muestra los ejercicios asociados a ese día mediante un RecyclerView y utilizando EjerciciosAdapter. Proporciona dos elementos Button para agregar y eliminar ejercicios.

Como controlador, obtiene la lista de ejercicios a través de BaseDatos, responde a los eventos de clic del botón “Agregar”, navegando hacia el fragmento AgregarEjerciciosFragment, y del botón “Eliminar”, mostrando un AlertDialog que permite al usuario seleccionar los ejercicios que desea eliminar. Actualiza la UI en consecuencia de los ejercicios eliminados.

### **AgregarEjerciciosFragment (VC)**

Proporciona elementos EditText para la entrada de datos y un Button para guardar el ejercicio. Además, recoge la información introducida por el usuario y gestiona los eventos de clic del botón de guardar, validando los campos de entrada y comunicándose con BaseDatos para insertar el nuevo ejercicio.

### **DetallesEjercicioFragment (VC)**

Presenta los detalles del ejercicio específico mediante elementos TextView. Carga los registros asociados utilizando un RecyclerView y la clase RegistrosAdapter. Proporciona un Button para nuevos registros.

También obtiene la información de ejercicios y registros de BaseDatos, responde a los eventos de clic del botón “Nuevo Registro”, abriendo un cuadro de diálogo a través de la clase NuevoRegistroDialogFragment. Actualiza la lista de registros en respuesta al guardado o eliminación de un registro.

### **NuevoRegistroDialogFragment (VC)**

Muestra la UI con elementos EditText para ingresar los datos del nuevo registro, así como un Button para guardar el registro. Como controlador, gestiona el evento clic del

botón “Guardar Registro”, validando los datos y comunicándose con BaseDatos para insertar el nuevo registro.

### **ConfirmDeleteDialogFragment (V)**

Es llamado por RegistrosAdapter para mostrar un diálogo de confirmación para borrar el registro. Proporciona dos botones para confirmar o cancelar la eliminación.

Finalmente, hablamos de los adaptadores empleados, cuya función principal es vincular datos concretos a una vista determinada.

### **Adaptador (V)**

Proporciona la vista de cada elemento de la lista rutinas integrada en rutinasFragment.

### **MyAdapter (VC)**

Proporciona la vista para cada elemento de la lista de días de entreno, integrada en CrearRutina. Asimismo, maneja la lógica de selección del checkbox de cada día de entrenamiento al crear una nueva rutina.

### **RutinasAdapter (VC)**

Proporciona la vista para cada elemento de la lista de rutinas integrada en TusRutinasFragment. Como controlador, gestiona la lógica para eliminar una rutina al clicar en el botón de eliminar, mostrando un diálogo de confirmación.

### **EjerciciosAdapter (VC)**

Proporciona la vista para cada elemento de la lista de ejercicios integrada en DetallesDiaEntrenoFragment. Gestiona el clic sobre cada ejercicio, navegando hacia DetallesEjercicioFragment.

### **RegistrosAdapter (VC)**

Proporciona la UI para cada elemento de datos de la lista de registros, vinculando los datos correspondientes. También responde a los eventos de clic sobre cada elemento de registro, mostrando una instancia de ConfirmDeleteDialogFragment. Gestiona el borrado de registros.

### **AllEjerciciosAdapter (VC)**

Proporciona la vista para cada elemento en la lista de ejercicios de registrosFragment. Gestiona la lógica para alternar la visibilidad de los registros de cada ejercicio al interactuar con los elementos de la lista.

### **AllRegistrosAdapter (V)**

Proporciona la vista para cada elemento de la lista de registros mostrada en registrosFragment.

Además de sustentarnos en el patrón MVC, es relevante señalar que la arquitectura de la aplicación responde a un diseño modular basado en fragmentos. De esta manera, podemos organizar con claridad el código y navegar entre las diferentes secciones, lo que aporta fluidez a la experiencia de usuario. En el Anexo 5, se incluye un diagrama representativo de esta estructura, con los fragmentos que componen la aplicación y el flujo de navegación entre ellos.

Como entorno de desarrollo (en adelante, IDE) empleamos Android Studio, dado que hemos aprendido con esta herramienta y se trata de la plataforma oficial para el desarrollo de aplicaciones Android. En relación con esta elección, escogemos Java como lenguaje de programación, puesto que Android Studio permite utilizar este o Kotlin y, en nuestro caso, todo lo aprendido en relación al desarrollo de aplicaciones ha sido mediante el uso de Java.

Este IDE ofrece una estructura de repositorios y carpetas que hemos mantenido, ya que nos permite organizar de forma óptima el conjunto de archivos necesarios para el correcto funcionamiento de la aplicación. Esta organización queda reflejada en el Anexo 6. Atendiendo a esta estructura, vamos a destacar aquellos aspectos que consideramos relevantes en términos de configuración, por lo que nos centramos en el archivo build.gradle.kts (Module:app), perteneciente a la carpeta Gradle Scripts.

Encontramos en este archivo la configuración esencial para compilar y construir la aplicación. Se utiliza un SDK de compilación 34, lo que nos permite aprovechar las últimas características y optimizaciones disponibles en la plataforma Android y asegurar que la aplicación sea compatible con las versiones más recientes del sistema

operativo. El mínimo de SDK se establece en API 24, correspondiente a Android 7.0 (Nougat). Esta elección nos permite alcanzar el 97,4% de dispositivos Android, lo que se traduce en una amplia compatibilidad sin sacrificar el acceso a las características modernas introducidas en versiones posteriores.

Por otro lado, habilitamos la vinculación de vistas “viewBinding”, lo que nos permite manejar de forma más eficiente la interacción con las vistas de nuestro código. Este hecho es especialmente útil para gestionar la navegación entre los fragmentos principales a través del BottomNavigationView incluido en MainActivity.

También cabe señalar la incorporación de las dependencias necesarias para el proyecto, entre las que se encuentran bibliotecas que proporcionan funcionalidades adicionales como “recyclerview”, que permite la implementación de listas personalizables de forma eficiente, algo que resultado esencial tal como estructuramos nuestra aplicación; “navigation”, que simplifica la navegación entre los componentes de la aplicación; o Material Design, esencial para disponer de componentes modernos que mejoran la apariencia y usabilidad de la aplicación.

## 8. Conclusiones y proyección a futuro

En este apartado, reflexionamos acerca del grado de consecución de los objetivos planteados al inicio del proyecto, las limitaciones encontradas en el desarrollo del mismo y los avances más interesantes que la aplicación podría incorporar en un futuro.

Planteamos un proyecto cuya finalidad principal era crear una aplicación móvil que integrara las funcionalidades necesarias para permitir al usuario diseñar sus propias rutinas de entrenamiento, estructurar los ejercicios por días y poder realizar registros de cada ejercicio. Igualmente, el usuario debía poder visualizar los registros anotados para seguir el progreso.

El resultado ha sido positivo, puesto que se ha logrado este propósito principal. Además, ofrecemos una UI que, desde nuestro punto de vista, cumple con los objetivos de facilidad de manejo y navegación intuitiva, ya que el contenido y las funcionalidades se encuentran claramente diferenciadas gracias a la estructuración en fragmentos.

Por tanto, podemos evaluar como satisfactorio el resultado del proyecto. No obstante, la actual aplicación cuenta con una serie de limitaciones que hemos de considerar para mejorar la funcionalidad ofrecida en futuras versiones.

En primer lugar, reparamos en el modo de almacenar los datos. En este caso, hemos optado por almacenamiento interno a través de base de datos SQLite y SharedPreferences. Aunque esta gestión de la información nos ha permitido cumplir con los objetivos, también limita la escalabilidad de la aplicación. Además, impide la disponibilidad de los datos en un dispositivo distinto de donde son almacenados. Por ello, una de las primeras mejoras se enfocaría en trabajar con una base de datos en la nube que permitiera la accesibilidad desde múltiples dispositivos. Asimismo, buscaremos incorporar un control de acceso de usuarios mediante login para asegurar la privacidad de los datos.

Otra limitación relevante se refiere a la carencia de funcionalidades avanzadas que aporten una experiencia de usuario más completa y personalizada. En este caso, se buscaría incorporar algoritmos de inteligencia artificial que ofrezcan al usuario



sugerencias y recomendaciones de rutinas, ejercicios, estructuración de entrenamientos y descansos, etc., que lo guíen en la consecución de sus objetivos.

Asimismo, sería beneficioso incorporar herramientas visuales, como gráficos o estadísticas, que representasen el progreso del usuario. Esto se podría desarrollar desde la sección perfil, añadiendo campos relevantes en la evaluación del progreso, como porcentaje de grasa o de masa muscular.

Por último, cabe señalar que el tiempo estipulado para desarrollar el proyecto, junto con las habilidades del equipo de desarrollo, han supuesto una limitación para el diseño de una aplicación más completa y atractiva.

En definitiva, este proyecto ha dado como resultado una aplicación móvil funcional para registrar rutinas de entrenamiento, que constituye una base sólida con amplísimo potencial de mejora para transformar este software en una herramienta mucho más completa y beneficiosa para los usuarios.

## 9. Bibliografía

- Amaya Balaguera, Y. D. (2013). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual. *Revista de Tecnología*, 12(2), 111-124. <https://doi.org/10.18270/rt.v12i2.1291>
- Bascón Pantoja, E. (2004). El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing. *Acta Nova*, 2(4), 493-507. [http://www.scielo.org.bo/pdf/ran/v2n4/v2n4\\_a05.pdf](http://www.scielo.org.bo/pdf/ran/v2n4/v2n4_a05.pdf)
- Estrada-Velasco, M. V., Núñez-Villacis, J. A., Saltos-Chávez, P. R., Cunuhay-Cuchipe, W. C. (2021). Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software. *Dominio de las Ciencias*, 7(4), 434-447. <http://dx.doi.org/10.23857/dc.v7i4.2429>
- Herrera Uribe, E. y Valencia Ayala, L. E. (2007). Del manifiesto ágil sus valores y principios. *Scientia et Technica*, XIII(34), 381-386. <https://www.redalyc.org/articulo.oa?id=84934064>
- López Menéndez de Jiménez, R. E. (2015). Metodologías Ágiles de Desarrollo de Software Aplicadas a la Gestión de Proyectos Empresariales. *Revista Tecnológica*, 8, 6-11. [http://fcaenlinea.unam.mx/anexos/1728/Unidad\\_1/u1\\_act2\\_2.pdf](http://fcaenlinea.unam.mx/anexos/1728/Unidad_1/u1_act2_2.pdf)
- Mendes Calo, K., Estévez, E. C. y Fillottrani, P. R. (2010). *Evaluación de metodologías ágiles para desarrollo de software*, XII Workshop de Investigadores en Ciencias de la Computación, Red de Universidades con Carreras en Informática, 455-459. <https://sedici.unlp.edu.ar/handle/10915/19546>
- Molina Ríos, J. R., Honores Tapia, J. A., Pedreira-Souto, N. y Pardo León, H. P. (2021). Estado del arte: metodologías de desarrollo de aplicaciones móviles. *3C Tecnología. Glosas de innovación aplicadas a la pyme*, 10(2), 17-45. <https://doi.org/10.17993/3ctecno/2021.v10n2e38.17-45>
- Rivas, C. I., Corona, V. P., Gutiérrez, J. F. y Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista Tecnología e Innovación*, 2(5), 980-

986.

[https://www.ecorfan.org/bolivia/researchjournals/Tecnologia\\_e\\_innovacion/vol2num5/Tecnologia\\_e\\_Innovacion\\_Vol2\\_Num5\\_6.pdf](https://www.ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf)

Rodríguez Vega, S. M. (2020). *Desarrollo de aplicaciones android: TodoMusica* [Trabajo de Fin de Grado]. Universidad de La Laguna. <http://riull.ull.es/xmlui/handle/915/20599>

Sokolova, K., Lemercier, M. y García, L. (2013). Android Passive MVC: a Novel Architecture Model for the Android Application Development. *PATTERNS 2013: The Fifth International Conferences on Pervasive Patterns and Applications* (pp.7-12). IARIA. <https://www.thinkmind.org/index.php?view=instance&instance=PATTERNS+2013>

Szalvay, V. (2004). An Introduction to Agile Software Development. *Danube Technologies*, 3, 1-11. [https://profs.info.uaic.ro/adrian.iftene/Licenta/Documentatie/Intro\\_to\\_Agile.pdf](https://profs.info.uaic.ro/adrian.iftene/Licenta/Documentatie/Intro_to_Agile.pdf)

Tito Durand, R. R., Alvan Ventura, E. Y., Moran Fuño, J. A. y Guevara Gutiérrez, M. A. (2023). Uso de las redes neuronales para determinar la calificación de una aplicación publicada en Google Play Store. *Innovación y software*, 4(1), 161-197. <https://www.redalyc.org/articulo.oa?id=673874721012>

Velásquez Restrepo, S. M., Vahos-Montoya, J. D., Gómez-Adasme, M. E., Pino-Martínez, A. A., Restrepo-Zapata, E. J. y Londoño-Marín, S. (2019). Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. *Revista CINTEX*, 24(2), 13-23. <https://doi.org/10.33131/24222208.334>

## 10. Anexos

### Anexo I: Conjunto de vistas

Figura 2

#### Sección Perfil

5:56

DailyGym

**Perfil**

Nombre  
Paula

Edad  
27

Peso (kg)  
52.0

Altura (cm)  
157.0

Sexo  
Mujer

Objetivo  
Ganar masa muscular

GUARDAR

Perfil Rutinas Registros

Figura 3

*Sección Rutinas*

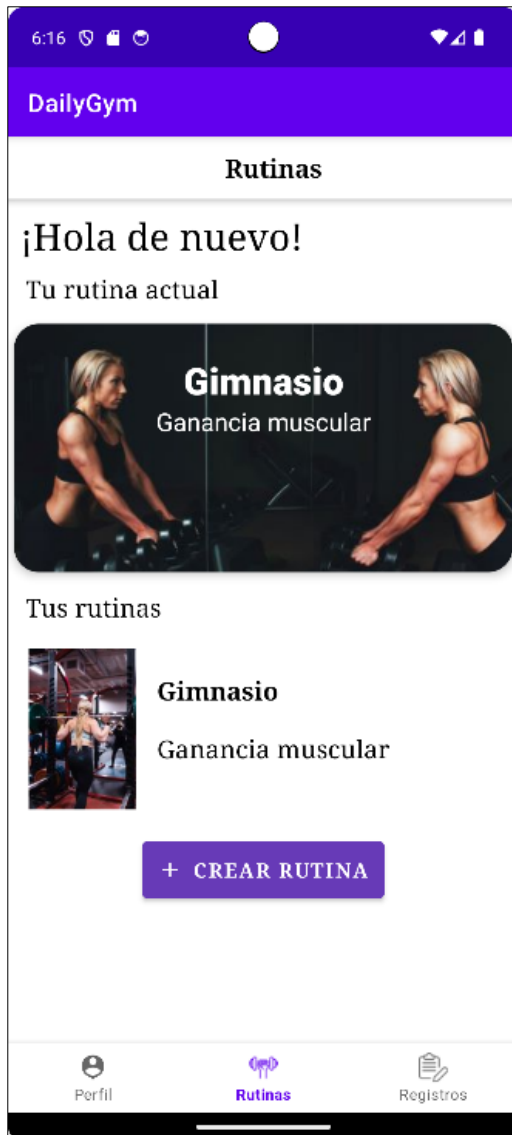


Figura 4

Sección *Tus Rutinas*

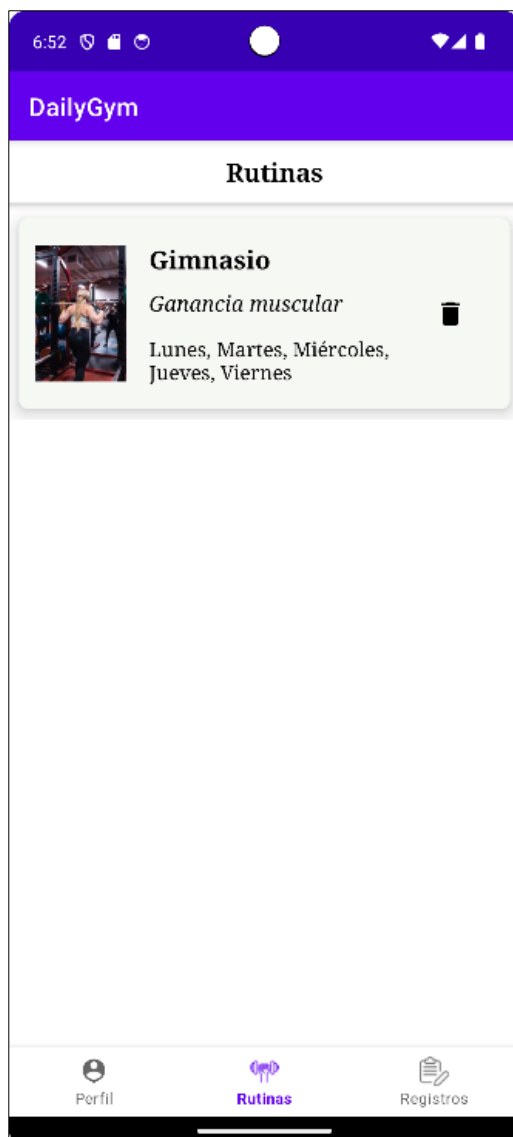


Figura 5

*Crear Rutina*

10:40

DailyGym

← Crear Rutina

Elige un nombre

Nombre

Añade una descripción

Descripción

Selecciona los días

☐ Lunes ☐ Martes

☐ Miércoles ☐ Jueves

☐ Viernes ☐ Sábado

☐ Domingo

Rutina para...

☐ Hombre ☐ Mujer


 GUARDAR RUTINA

Figura 6

*Detalle Rutina*

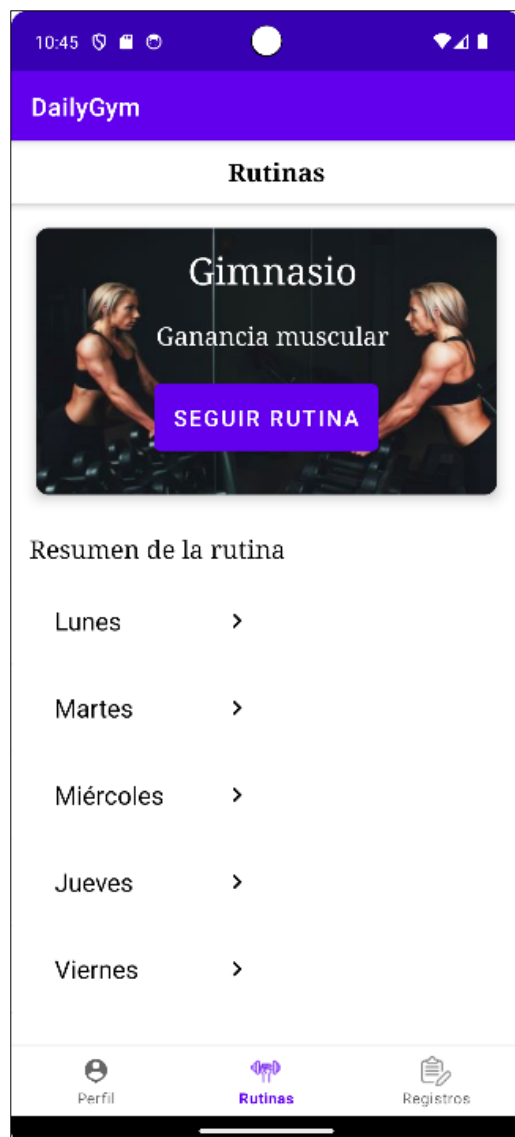




Figura 7

*Detalle Día*

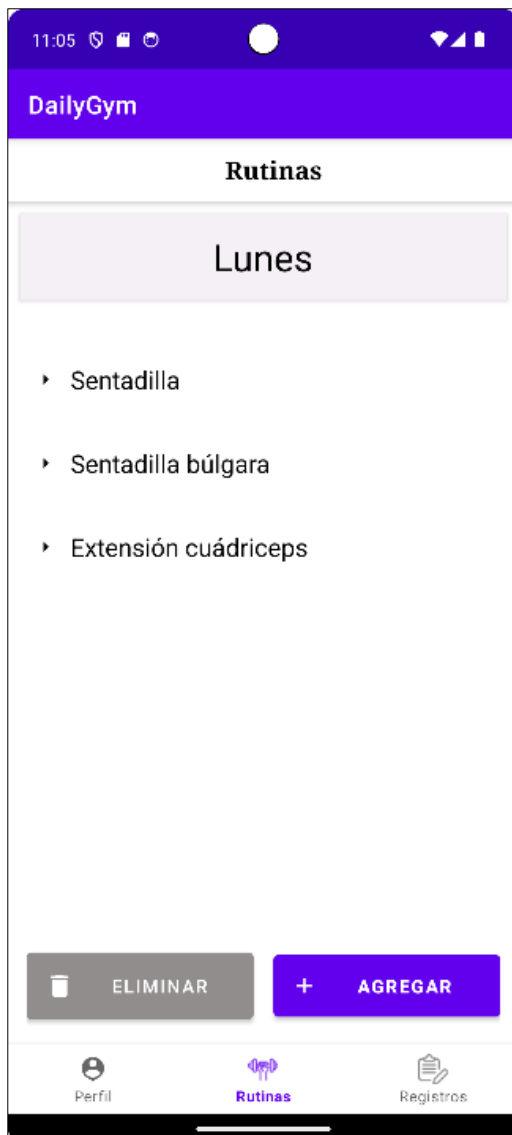


Figura 8

*Agregar Ejercicio*

The screenshot shows the 'Agregar Ejercicio' (Add Exercise) screen in the 'DailyGym' app. The app's status bar at the top shows the time as 11:13 and various system icons. The app's title bar is blue and displays 'DailyGym'. Below the title bar, the screen is titled 'Rutinas' (Routines). The form consists of three text input fields: 'Nombre' (Name), 'Descripción' (Description), and 'Músculo' (Muscle). A blue button labeled 'GUARDAR' (Save) is positioned below the 'Músculo' field. At the bottom of the screen, there is a navigation bar with three icons: a person icon for 'Perfil' (Profile), a dumbbell icon for 'Rutinas' (Routines), and a document icon for 'Registros' (Records). The 'Rutinas' icon is highlighted in blue.

Figura 9

*Eliminar Ejercicios*

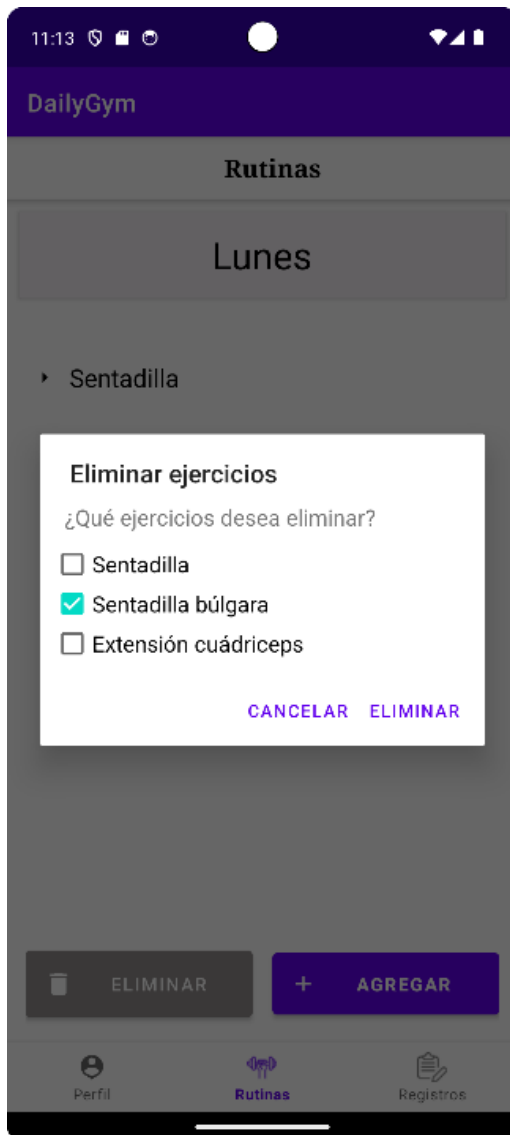


Figura 10

*Detalle Ejercicio*



Figura 11

*Agregar Registro*

9:51

DailyGym

Rutinas

Ejercicio: Sentadilla

Con barra

Tren inferior

NUEVO REGISTRO

Peso (kg)

Repeticiones

Series

GUARDAR

Peso: 30.0 kg Repeticiones: 12 Series: 2

Perfil Rutinas Registros

Figura 12

*Eliminar Registros*

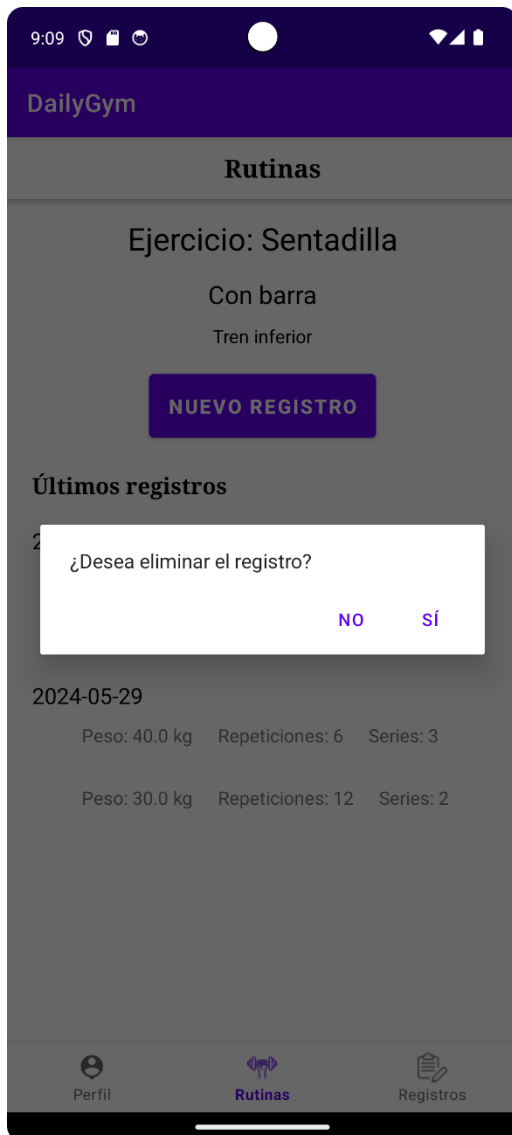
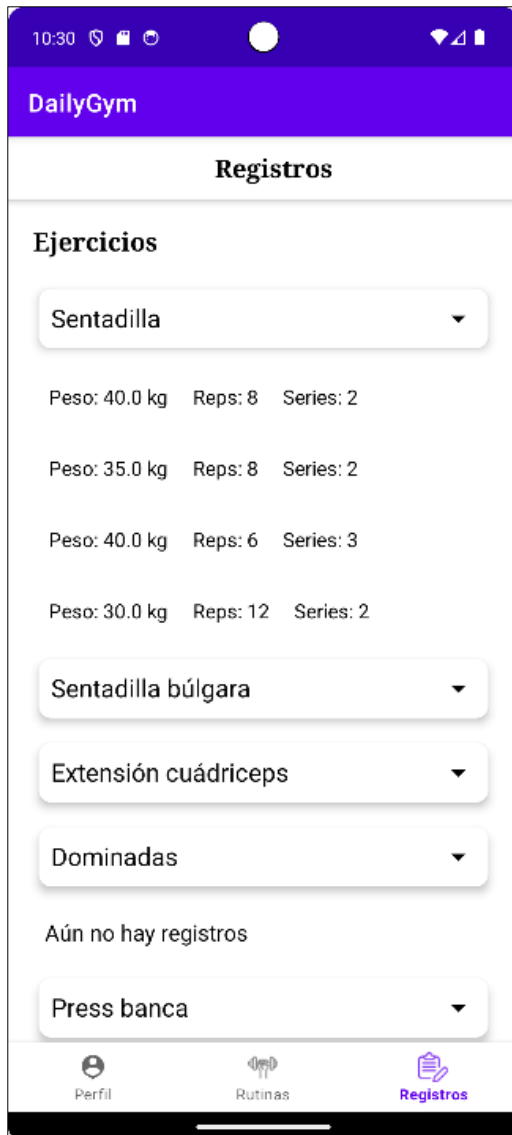


Figura 13

*Sección Registros*



## Anexo II: Conjunto de tablas en base de datos

Tabla 2

*Tabla Rutinas*

RUTINAS	ID
	NOMBRE
	DESCRIPCION
	DIAS_ENTRENO
	AUTOR

Tabla 3

*Tabla Ejercicios*

EJERCICIOS	ID_EJERCICIO
	NOMBRE_EJERCICIO
	DESCRIPCION_EJERCICIO
	MUSCULO_PRINCIPAL
	ID_RUTINA (fk)
	ID_DIA_ENTRENO

Tabla 4

*Tabla Registros*

REGISTROS	ID_REGISTRO
	ID_EJERCICIO (fk)
	ID_RUTINA (fk)
	ID_DIA_ENTRENO
	PESO
	REPETICIONES
	SERIES
	FECHA



### Anexo III: Calendario Planificación de Sprints

Tabla 5

#### *Planificación de Sprints*

Calendario	Sprint
8-10 abril	Definir secciones principales y gestionar navegación entre ellas
11-14 abril	Diseño de la sección Rutinas
15-17 abril	Diseñar y configurar la actividad para crear rutinas
18-23 abril	Persistencia de las rutinas creadas a través de base de datos SQL
24-26 abril	Configurar el listado de rutinas de la sección principal
27-29 abril	Crear y diseñar la sección individual de las rutinas y permitir el borrado de rutinas
30 abril - 3 mayo	Configurar la navegación y vista en detalle de cada rutina particular, añadir botón para definir una rutina como predeterminada
4-7 mayo	Configurar la navegación y vista en detalle de cada día de entreno
8-12 mayo	Diseñar e incluir funcionalidad para agregar y eliminar ejercicios, persistencia de los ejercicios en base de datos SQL
13-16 mayo	Configurar navegación y vista en detalle de cada ejercicio
17-20 mayo	Incluir funcionalidad para agregar registros, persistencia de los registros en base de datos SQL
21-26 mayo	Recuperar los registros tanto en la vista en detalle por ejercicio como en la sección Registros. Incluir opción de eliminar registros
27-28 mayo	Configurar sección Perfil

## Anexo IV: Distribución de clases según el patrón MVC

Tabla 6

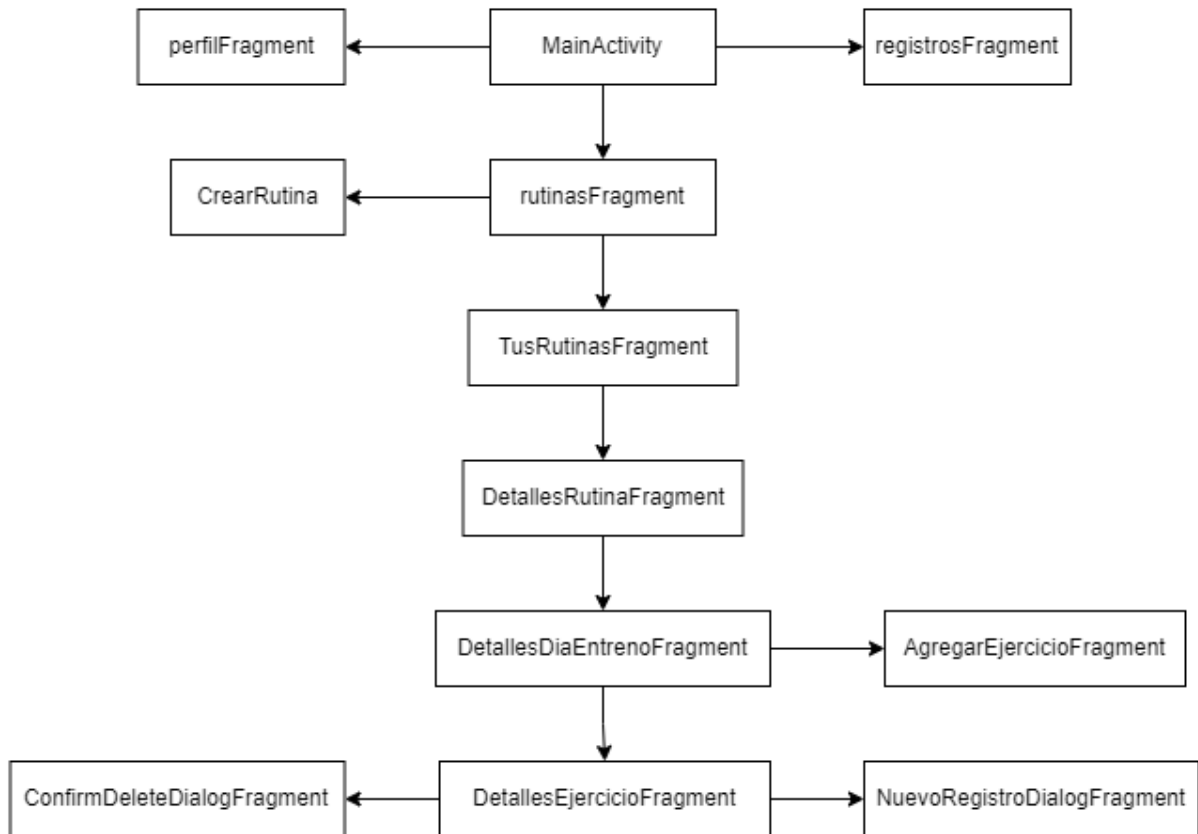
### Agrupación MVC

Clase	Modelo	Vista	Controlador
Rutinas			
DiasEntreno			
Ejercicios			
Registro			
BaseDatos			
UserProfile			
PreferenceManager			
MainActivity			
perfilFragment			
rutinasFragment			
registrosFragment			
CrearRutina			
TusRutinasFragment			
DetallesRutinaFragment			
DetallesDiaEntrenoFragment			
DetallesEjercicioFragment			
AgregarEjerciciosFragment			
NuevoRegistroDialogFragment			
ConfirmDeleteDialogFragment			
Adaptador			
MyAdapter			
RutinasAdapter			
EjerciciosAdapter			
RegistrosAdapter			
AllEjerciciosAdapter			
AllRegistrosAdapter			

## Anexo V: Diagrama de flujo entre fragmentos

Figura 14

Diagrama



## **Anexo VI: Estructura de repositorios y carpetas**

- app
  - manifests
  - java
    - com.example.dailygym
    - com.example.dailygym (androidTest)
    - com.example.dailygym (test)
  - java (generated)
  - res
    - drawable
    - layout
    - menú
    - mipmap
    - navigation
    - values
    - xml
  - res (generated)
- Gradle Scripts