FitnessApp: Desarrollo y una aplicación de gestión deportiva

2º de DAM (Desarrollo de Aplicaciones Multiplataforma)
Desarrollo de Interfaces

GONZALO SÁNCHEZ ÁLVAREZ

Resumen Ejecutivo

Fitness App es una aplicación diseñada para optimizar y estructurar entrenamientos en el gimnasio. A diferencia de aplicaciones como Google Fit o MyFitnessPal, que se enfocan en el seguimiento general de la actividad física, Fitness App permite registrar ejercicios de manera detallada, incluyendo peso, repeticiones y evolución a lo largo del tiempo.

Actualmente, no existen aplicaciones accesibles que combinen planificación personalizada de entrenamientos con una interfaz intuitiva y optimizada para el usuario. Fitness App cubre esta necesidad con una plataforma funcional, fácil de usar y enfocada en mejorar la experiencia de entrenamiento.

Oportunidad en el Mercado

El sector fitness ha experimentado un crecimiento significativo en la digitalización, pero la mayoría de las aplicaciones no permiten un registro estructurado de entrenamientos. Muchos usuarios aún dependen de notas en papel o archivos de texto desorganizados, lo que dificulta la medición de su progreso.

Si bien la aplicación está enfocada en el usuario individual, existe un potencial de crecimiento para desarrollar funcionalidades que permitan la gestión de entrenamientos por parte de entrenadores y gimnasios, lo que ampliaría aún más su alcance.

Aunque actualmente Fitness App no permite la administración de clientes por parte de entrenadores, esta es una funcionalidad planificada para futuras versiones, lo que aumenta su potencial dentro del mercado digital del fitness.

Fitness App se distribuirá a través de un sistema de licencias escalables, ofreciendo planes adaptados a distintos tipos de usuarios y necesidades.

- **Licencia Individual (€4,99/mes o €49,99/año):** Permite gestionar y registrar entrenamientos personales sin restricciones.
- **Licencia para Gimnasios (€29,99/mes por instalación):** En desarrollo, ofrecerá funcionalidades específicas para centros deportivos con múltiples usuarios.
- **Licencia Corporativa (Precio según volumen de usuarios):** Dirigida a grandes gimnasios y cadenas de fitness que buscan integrar la aplicación en sus sistemas internos.

Este modelo garantiza ingresos recurrentes y permite la escalabilidad a medida que crece la base de usuarios.

Conclusión

Fitness App representa una solución innovadora para la gestión y optimización del entrenamiento personal. Su enfoque en la planificación detallada de ejercicios la diferencia de otras aplicaciones en el mercado. Aunque la gestión para entrenadores aún no está disponible, la aplicación ya cubre una necesidad clave en el ámbito del fitness digital.

Con un modelo basado en licencias escalables, Fitness App se perfila como una herramienta clave en la transformación digital del sector, con oportunidades de crecimiento tanto en el mercado individual como en el ámbito profesional del fitness.

1. Creación de Componentes Visuales en la Aplicación

1.1. Componentes y Eventos

La interfaz de usuario de la aplicación ha sido diseñada utilizando **Android Studio**, en la que se emplearon archivos **XML** para definir los componentes visuales, y **Java** para manejar la lógica y los eventos asociados a estos componentes. Entre los componentes visuales utilizados se encuentran:

- **Botones**: Para realizar acciones como la carga y edición de los ejercicios, la navegación entre pantallas, y el envío de datos a la base de datos, además de botones de registro e inicio de sesión.
- **Campos de entrada de texto**: Utilizados para introducir el nombre del ejercicio, el peso, las repeticiones, y descripciones de los ejercicios además de esto todos los campos de registro e inicio de sesión, como los campos para editar datos.
- **Formularios**: Incluyendo el formulario de registro de ejercicios en que hay que rellenar los campos anteriormente comentados.
- **Imágenes**: Asociadas a cada ejercicio para ilustrar el tipo de actividad.
- **Perfil de usuario**: Una sección donde el usuario puede ver y editar su información personal.

Cada uno de estos componentes está asociado a eventos específicos como, por ejemplo:

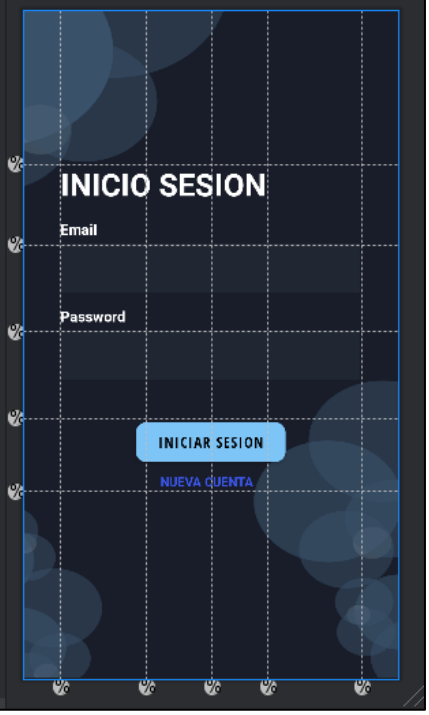
- **Botones de navegación**: Al hacer clic en estos, se navega entre diferentes actividades dentro de la app.
- **Botones de envío de datos**: Al hacer clic, los datos de los formularios son validados y enviados a la base de datos.
- **Campos de texto**: Detectan cuando el usuario introduce o edita los valores de los ejercicios.

1.2. Propiedades y Atributos

Cada componente tiene atributos definidos en los archivos XML de la interfaz. A continuación, algunos ejemplos como la interfaz de inicio de sesión:

```
<TextView
    android:id="@+id/l_txv_email"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="5dp"
    android:fontFamily="@font/roboto_bold"
    android:text="Email"
    android:textColor="#FFFFFF"
    android:textSize="16sp"
    app:layout_constraintBottom_toTopOf="@+id/l_inpt_email"
    app:layout_constraintStart_toStartOf="@+id/v_left_gl" />

<Button
    android:id="@+id/l_btn_iniciarSesion"
    android:layout_width="164dp"
    android:layout_height="55dp"
    android:backgroundTint="#7DC4F7"
    android:fontFamily="@font/open_sans_condensed_bold"
    android:onClick="checkEmptyInputsLogin"
    android:text="INICIAR SESION"
    android:textColor="#000000"
    android:textSize="16sp"
    app:cornerRadius="30px"
    app:layout_constraintEnd_toStartOf="@+id/v_right_gl"
    app:layout_constraintStart_toStartOf="@+id/v_left_gl"
    app:layout_constraintTop_toTopOf="@+id/guideline2" />
```



La actividad presentada corresponde a una pantalla de inicio de sesión (**LoginActivity**), donde se han implementado los siguientes componentes básicos:

TextView: Para mostrar etiquetas como "Email", "Password" y el título "INICIO SESIÓN".

TextInputEditText: Para que el usuario introduzca su correo electrónico y contraseña.

Button: Para ejecutar acciones como iniciar sesión o cambiar a la actividad de registro.

Guidelines: Utilizadas para organizar y alinear los elementos de forma consistente dentro de un **ConstraintLayout**.

Botón para iniciar sesión

```
<Button
    android:id="@+id/l_btn_iniciarSesion"
    android:layout_width="164dp"
    android:layout_height="55dp"
    android:backgroundTint="#7DC4F7"
    android:fontFamily="@font/open_sans_condensed_bold"
    android:onClick="checkEmptyInputsLogin"
    android:text="INICIAR SESION"
    android:textColor="#000000"
    android:textSize="16sp"
    app:cornerRadius="30px"
    app:layout_constraintEnd_toStartOf="@+id/v_right_gl"
    app:layout_constraintStart_toStartOf="@+id/v_left_gl"
    app:layout_constraintTop_toTopOf="@+id/guideline2" />
```

Este botón llama al método ***checkEmptyInputsLogin*** al ser pulsado, verificando si los campos de correo y contraseña están vacíos.

Campo de entrada para el correo electrónico

```
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/l_inpt_email"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="328dp"
    android:layout_height="54dp"
    android:background="#202733"
    android:inputType="text"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textColor="#FFFFFF"
    android:textColorHint="@color/white"
    app:layout_constraintEnd_toStartOf="@+id/v_right_gl"
    app:layout_constraintStart_toStartOf="@+id/v_left_gl"
    app:layout_constraintTop_toTopOf="@+id/GLTopInputNameInicioSesion" />
```

Este campo permite al usuario introducir su dirección de correo electrónico, con un fondo oscuro y texto blanco.

En esta actividad, se han configurado propiedades clave para personalizar los componentes visuales.

1. **TextView: Configuración de fuente y color**

Propiedades destacadas:

- ***android:fontFamily****:* Define la tipografía personalizada.
- ***android:textSize:*** Especifica el tamaño de la fuente en sp.
- ***android:textColor****:* Configura el color del texto.

2. **Botón: Color de fondo y esquinas redondeadas**

Propiedades destacadas:

- *android:backgroundTint:* Establece el color del fondo del botón.
- *app:cornerRadius:* Aplica bordes redondeados al botón.

3. **Guidelines: Organización de componentes**

Las Guidelines ayudan a posicionar los elementos en el diseño. En este ejemplo, la guía se encuentra al 10% del ancho de la pantalla.

1.3. Listeners

Listener para confirmar la edición de un ejercicio

```
btnyes.setOnClickListener(new View.OnClickListener() {
    gontugithub
    @Override
    public void onClick(View view) {

        editExercise();


    }
});
```

1. *btnYes:* Es el identificador del botón que captura el evento de clic.
2. *setOnClickListener:* Se utiliza para escuchar el evento de clic en el botón.
3. *editExercise():* Método llamado al pulsar el botón, que contiene la lógica necesaria para realizar la edición del ejercicio.

Este Listener asegura que, al pulsar el botón "Yes" (o "Sí"), se confirme la acción de editar un ejercicio en la aplicación. La lógica del método *editExercise* puede incluir la

validación de datos, la actualización de la base de datos, y la retroalimentación al usuario, como se muestra a continuación:

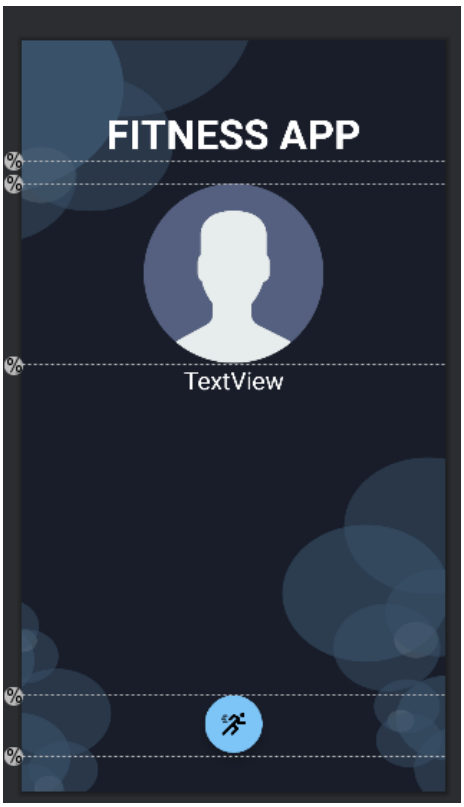
1.4 Métodos y Eventos

En **Fitness App**, los métodos permiten gestionar diversas acciones dentro de la aplicación, como la navegación entre actividades. Esto es crucial para proporcionar una experiencia de usuario fluida y estructurada.

```
public void changeHomeActivity(View view){
    startActivity(new Intent( packageContext: LoginActivity.this, HomeActivity.class ));

}
```

- *public void changeHomeActivity(View view):* Método público que se ejecuta cuando se vincula a un evento de clic en un botón (por ejemplo, al iniciar sesión).
- *Intent:* Es una clase que permite iniciar nuevas actividades. En este caso, se utiliza para abrir la actividad HomeActivity.
- *startActivity:* Método que lanza la nueva actividad especificada en el Intent.
- *LoginActivity.this:* Hace referencia a la actividad actual desde la que se está llamando el método.
- *HomeActivity.class:* Indica la actividad que se abrirá.



1.5 Persistencia

En **Fitness App**, se utiliza una combinación de métodos de persistencia para garantizar la disponibilidad y accesibilidad de los datos, tanto en línea como localmente:

1. **Firebase Realtime Database**: Gestiona los datos principales de la aplicación, como los ejercicios, entrenamientos y perfiles de los usuarios, ofreciendo sincronización en tiempo real.
2. **Base de datos auxiliar en SQLite**: Se utiliza para almacenar localmente los datos de inicio de sesión, permitiendo un acceso rápido y seguro cuando no hay conexión a internet.

1.5.1. Firebase Realtime Database

Firebase permite almacenar y sincronizar datos en tiempo real entre los usuarios. En **Fitness App**, Firebase se utiliza para guardar los ejercicios y entrenamientos.

```
FirebaseFirestore database = FirebaseFirestore.getInstance();
```

Se crea una instancia de **Firebase Firestore** para poder interactuar con la base de datos.

```
database.collection("users").document(userId)
.get(src).addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() { ... });
```

- Se accede a la colección **users** y se obtiene el documento del usuario correspondiente mediante su *userId*.
- Se usa *Source.SERVER* para asegurarse de obtener los datos actualizados desde el servidor.

```
ArrayList<HashMap> exerciseArray = (ArrayList<HashMap>) data.get("exercises");
```

Se obtiene la lista de ejercicios almacenada en el documento del usuario.

Se verifica si la lista de ejercicios está vacía antes de agregar uno nuevo. Entonces si está vacía agregamos, el Hashmap del nuevo ejercicio

```
if (exerciseArray.isEmpty()) {
```

```
HashMap<Object, Object> nuevoEjercicio = new HashMap<>();
nuevoEjercicio.put("name", exercisename.getText().toString());
nuevoEjercicio.put("repetition", Integer.parseInt(reps.getText().toString()));
nuevoEjercicio.put("weight", Double.parseDouble(weight.getText().toString()));
```

1.5.2 Base de Datos Local en SQLite)

En **Fitness App**, además del uso de Firebase, se ha implementado una base de datos local con **SQLite** para almacenar datos de sesión del usuario. Esto permite que la aplicación mantenga la información de inicio de sesión sin tener que acceder a Firebase.

Este sería el código para poder crear la base de datos local

```java
package com.example.fitnessapp;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class DataBaseAux extends SQLiteOpenHelper {
    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "REGISTRO_USUARIOS";

    public DataBaseAux(@Nullable Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    // Se ejecuta la primera vez que se crea la base de datos
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("CREATE TABLE users (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "email TEXT NOT NULL, " +
                "name TEXT NOT NULL, " +
                "password TEXT NOT NULL)");
    }

    // Se ejecuta cuando hay una actualización de versión de la base de datos
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS users");
        onCreate(sqLiteDatabase);
    }
}
```

Extiende de la clase SQLiteOpenHelper:
Esto permite gestionar la base de datos de manera eficiente.

Estructura de la base de datos:

- users: Tabla para almacenar los datos de los usuarios.
- Columnas: id (clave primaria), email, name, password.

Método onCreate:

- Se ejecuta cuando se crea la base de datos por primera vez.
- Se define la estructura de la tabla users.

Método onUpgrade:

- Se usa cuando se actualiza la versión de la base de datos.
- Elimina la tabla existente (DROP TABLE IF EXISTS) y la recrea.

1.6 Empaquetado formato APK

El **APK (Android Package)** es el archivo utilizado para instalar aplicaciones en dispositivos **Android**. En **Fitness App**, la generación del APK se realizó con **Android Studio**, permitiendo su instalación y distribución.

1.6.1. Proceso de generación del APK en Android Studio

Pasos que seguir para crear la APK:

1. **Abrir el proyecto en Android Studio.**
2. Ir a **Build** → *Generate Signed Bundle / APK...*.
3. Seleccionar **APK** y hacer clic en *Next*.
4. **Elegir la clave de firma** (KeyStore) o crear una nueva.
5. Seleccionar la **variante de compilación**:
    - **Debug APK** (para pruebas).
    - **Release APK** (para distribución).
6. Hacer clic en **Finish**.
7. El APK generado se encuentra en:
    - **Modo Release:** app/build/outputs/apk/release/app-release.apk
    - **Modo Debug:** app/build/outputs/apk/debug/app-debug.apk

Una vez generada la APK procederemos a instalarla en nuestro dispositivo móvil Android.

1.6.2. Instalación de la APK

Para hacer la instalación hay 2 formas de hacerlo:

Instalación manual

1. Transferir el archivo APK al móvil.
2. Habilitar **Orígenes desconocidos** en **Ajustes** > **Seguridad**.
3. Abrir el APK y hacer clic en **Instalar**.

Instalación con ADB

Si el dispositivo está conectado al ordenador con **depuración USB activada**, se puede instalar usando la terminal y ejecutando este comando, con el nombre que tenga la APK.

```
adb install app-release.apk
```

2. Usabilidad en Fitness App

La usabilidad en **Fitness App** ha sido diseñada para que los usuarios puedan **navegar e interactuar con facilidad**, asegurando una experiencia fluida e intuitiva. Se han aplicado principios de **accesibilidad, eficiencia y confort visual**, basándose en estándares de usabilidad y ergonomía.

2.1. Objetivos de Uso y Estándares de Usabilidad

El diseño de **Fitness App** sigue estos principios clave de usabilidad:

- **Accesibilidad mejorada**: Se ha colocado el menú en la zona inferior, ya que es más fácil de alcanzar con el dedo al sostener el teléfono.
- **Navegación intuitiva**: Cada sección de la app (Home, Ejercicios, Registrar Ejercicios) está representada con iconos visuales, permitiendo al usuario identificar rápidamente la función de cada botón.
- **Evitar fatiga visual**: Se ha utilizado un fondo oscuro con tonos azulados, reduciendo la fatiga ocular en el uso prolongado de la aplicación.
- **Tipografía clara y legible**: La fuente elegida facilita la lectura sin causar esfuerzo visual innecesario.

6.2. Aplicación de Normas Internacionales

ISO / IEC 9126-1 (Calidad del Producto)

- Se han optimizado los tiempos de carga y la respuesta rápida de la interfaz, asegurando eficiencia y rendimiento en diferentes dispositivos.

ISO / IEC 9241 (Guía de Usabilidad)

- El menú inferior y la disposición de los elementos permiten que los usuarios interactúen de forma rápida y natural.
- Se ha reducido la sobrecarga de información en cada pantalla, facilitando la navegación.

ISO / IEC 14915 (Ergonomía del Software y Diseño Visual)

- Uso de iconos representativos en el menú, minimizando la necesidad de texto explicativo.
- Tamaño de botones óptimo para evitar toques accidentales y mejorar la interacción.

2.2. Tipos de Usuarios en Fitness App

1. **Usuario Registrado**
    - Puede registrar y gestionar entrenamientos.
    - Accede a su perfil y datos sincronizados en la nube.

2.3. Características de Usabilidad Aplicadas en Fitness App

En **Fitness App**, se han tenido en cuenta diversas características para mejorar la experiencia del usuario, asegurando accesibilidad, claridad y optimización para distintos tipos de dispositivos y usuarios.

Capacidades Cognitivas y Perceptivas

- Interfaz clara y minimalista con iconos intuitivos.
- Mensajes de error y confirmación para guiar al usuario.
- Botones diferenciados y bien ubicados para evitar confusión.

Si un usuario intenta guardar un ejercicio sin completar los datos, aparece un mensaje indicando qué falta.

Factores Culturales

- Diseño neutro sin referencias culturales específicas.
- Posibilidad de adaptar unidades y formatos en futuras actualizaciones.

Opción de cambiar entre kilogramos y libras según preferencia del usuario.

Consideraciones para Discapacidades

- Contraste adecuado para mejorar la legibilidad.
- Botones grandes y bien espaciados para facilitar la interacción.
- Compatible con lectores de pantalla como TalkBack en Android.

Los botones de "Guardar" y "Cancelar" tienen suficiente tamaño para evitar errores al pulsar.

Factores Tecnológicos

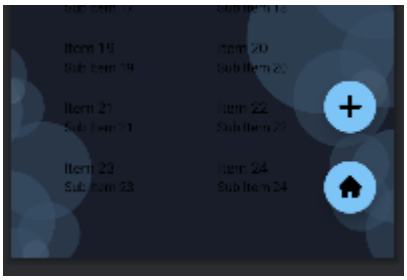
- Optimizada para dispositivos de gama baja y media.
- Sincronizando datos cuando hay internet.
- Fluidez en la navegación con bajo consumo de batería y memoria.

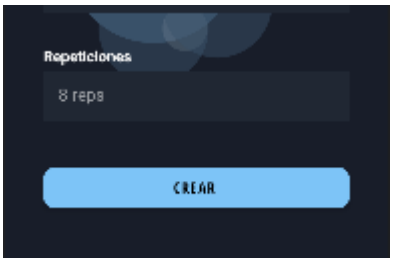
2.4. Pautas de Diseño en Fitness App

Para facilitar la navegación, el menú principal se ha ubicado en la **parte inferior** de la pantalla, permitiendo un acceso rápido con el pulgar.

El menú inferior cuenta con tres botones representados por iconos:

- **Inicio:** Accede a la pantalla principal.
- **Ejercicios:** Acceso a los ejercicios registrados
- **Registrar Ejercicios:** Acceso a un formulario para registrar nuevos ejercicios.





Ventanas y Cuadros de Diálogo

Tengo varias ventanas, o actividades en este caso cada una con sus cuadros de diálogos adaptados a sus funciones de registro, iniciar sesión, nuevo ejercicio etc..

Pautas de Diseño Relativas al Aspecto

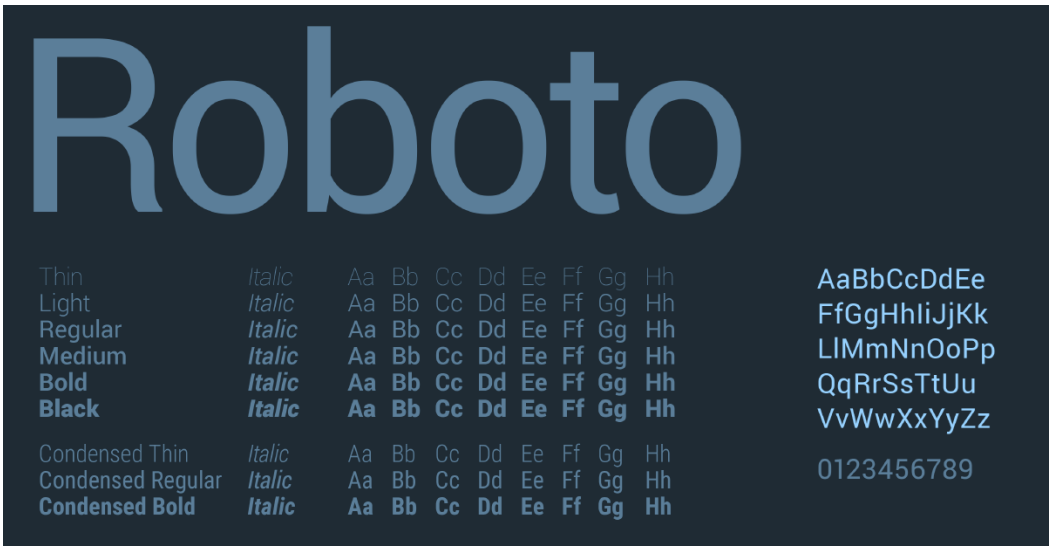
Iconos: Se utilizan para representar funciones de forma visual y reducir la cantidad de texto en la interfaz.

Colores: Fondo oscuro con tonos azulados para reducir fatiga visual.

Botones de acción en colores llamativos (azul claro) para destacarlos.

El botón de "Iniciar sesión" es azul claro con texto negro, destacándose sobre el fondo oscuro.

Fuente: Se ha elegido una tipografía clara y de tamaño adecuado para facilitar la lectura. La fuente concretamente es ROBOTO



Elementos Interactivos

Títulos intuitivos: Cada pantalla tiene un encabezado claro que indica su función.

Acciones comprensibles: Los botones tienen textos directos como "Guardar ejercicio" o "Ver entrenamiento".

Opciones diferenciadas: Cada elemento tiene su propio estilo visual para evitar confusión.

En la pantalla de ejercicios, los botones "Agregar" y "Eliminar" tienen colores distintos para diferenciarlos.

Presentación de Datos

En **Fitness App**, los datos de los ejercicios se presentan de manera clara y visualmente atractiva. Cada ejercicio se muestra en una tarjeta que incluye:

- **Nombre del ejercicio**: Título en texto blanco destacado para identificar fácilmente la actividad.
- **Peso utilizado**: Resaltado en un tamaño de fuente mayor, acompañado de la unidad de medida en este caso (kg).
- **Repeticiones**: Mostradas en color amarillo para diferenciarlas de otros datos.
- **Opciones interactivas**: Iconos visibles para editar (icono azul) o salir del ejercicio (icono amarillo) el ejercicio.
- **Paleta de color:** La imagen de la derecha contiene los colores utilizados en la app

Accesibilidad

Hacer **Fitness App** accesible para personas con algun tipo de discapacidad implica implementar diversas estrategias que mejoren la experiencia de los usuarios con discapacidades visuales, auditivas o motoras.

En los elementos visuales importantes (botones, iconos, imágenes). Podríamos añadir una propiedad llamada ContentDescription que permite a través del TalkBack de Android a los usuarios que se les pueda describir el contenido. Y este sería un ejemplo de cómo se implementaría. (En un botón).

```
<ImageButton
    android:id="@+id/edit_button"
    android:contentDescription="Editar ejercicio"
    android:src="@drawable/ic_edit" />
```

Para personas con discapacidad motora:

- La aplicación es compatible con dispositivos como **Switch Access** o teclados adaptados, permitiendo una navegación sencilla.
- Todas las interacciones se pueden realizar con toques únicos, evitando gestos complicados.
- Los botones son grandes y están bien separados, facilitando su selección y reduciendo errores.

Análisis y verificación de la usabilidad

La usabilidad en **Fitness App** se ha verificado en distintos elementos clave de la aplicación.

- **Visibilidad del estado del sistema**: Los mensajes de confirmación y error aparecen en **formularios de registro y edición de ejercicios**, asegurando que el usuario reciba retroalimentación inmediata.
- **Control y libertad del usuario**: En la pantalla de **ejercicios**, los usuarios pueden **cancelar o eliminar** acciones mediante cuadros de diálogo de confirmación.
- **Prevención de errores**: En los **formularios de entrada de datos**, se validan los campos en tiempo real, evitando el registro de información incorrecta.
- **Navegación intuitiva**: El **menú inferior** facilita el acceso a las secciones principales sin pasos adicionales, asegurando un uso fluido.

Estos elementos garantizan que la aplicación sea clara, accesible y fácil de utilizar.

2.5 Análisis y verificación del proceso de desarrollo de interfaces

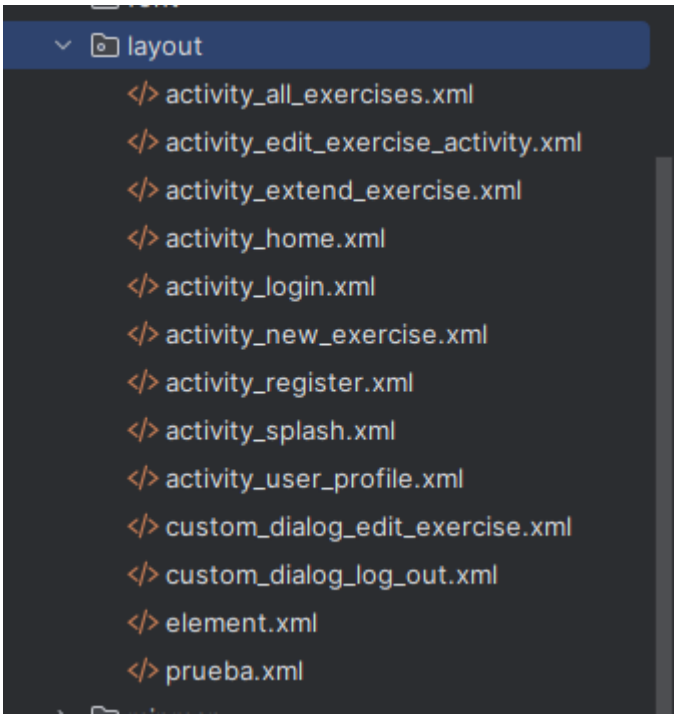
El desarrollo de Fitness App siguió un proceso estructurado en varias fases, asegurando que la aplicación cumpliera con sus objetivos y ofreciera una experiencia de usuario fluida. Desde la planificación hasta el mantenimiento, se utilizaron herramientas como Miro para el diseño inicial, Android Studio para la implementación, GitHub para el control de versiones, además de ello ser realizaron pruebas en múltiples dispositivos y emuladores.

Fase de Planificación

El diseño inicial se realizó en Miro, donde se definió el flujo de actividades, especificando cómo el usuario navegaría entre las diferentes pantallas. En esta etapa, también se planteó la estructura general de la aplicación, incluyendo las funcionalidades principales y la interacción entre módulos.

Fase de Diseño

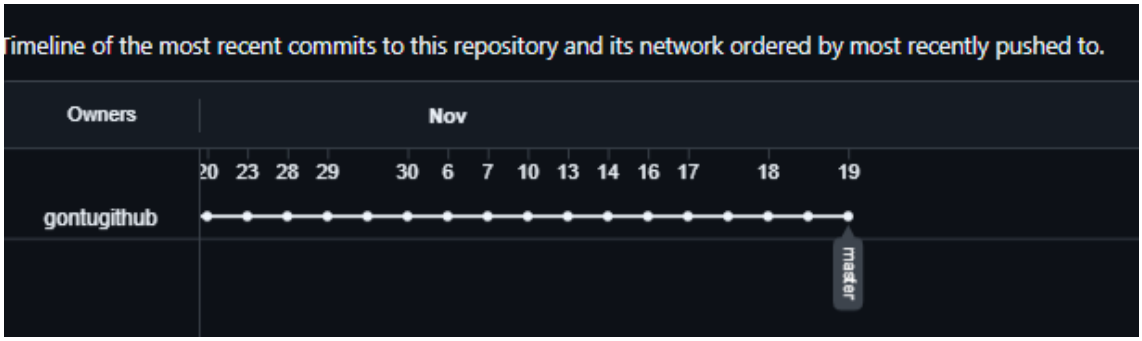
Se crearon los layout en XML dentro de Android Studio, asegurando una estructura visual clara y accesible. Se establecieron principios de diseño como el menú inferior para facilitar la navegación y el uso de colores oscuros para reducir la fatiga visual. También se planificó la estructura de los datos y la comunicación con Firebase.



Fase de Implementación

El desarrollo se llevó a cabo en Android Studio utilizando Java para la lógica y Firebase para la base de datos.

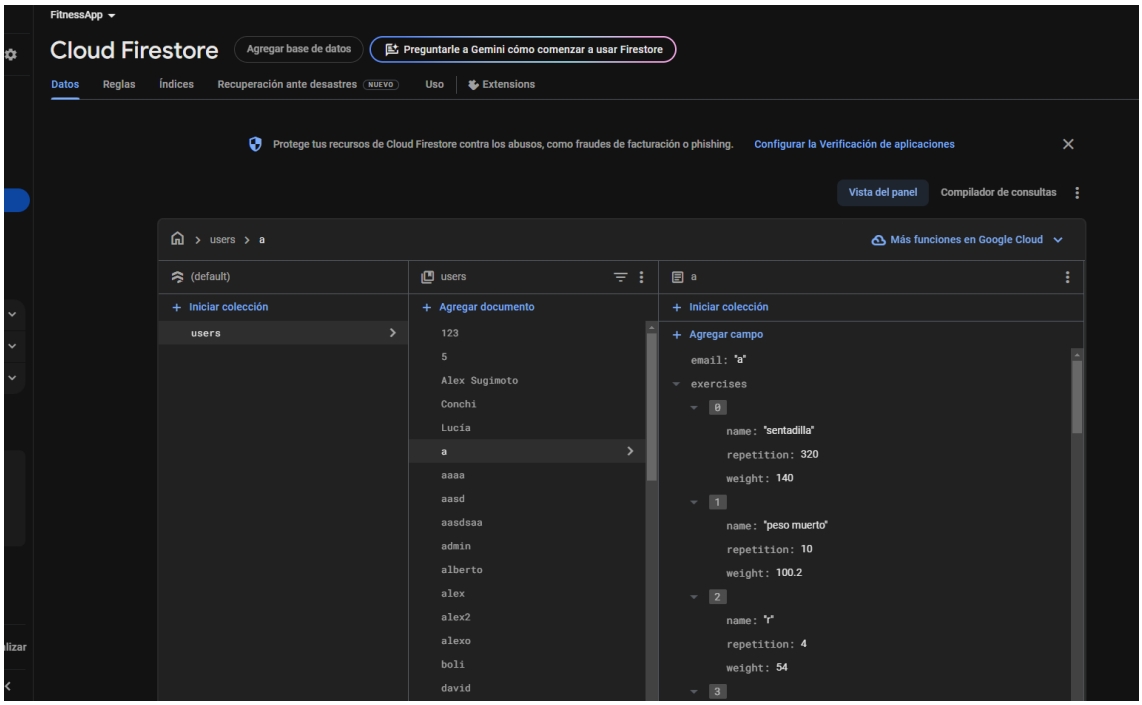
- Se implementó un Singleton para gestionar el estado del usuario registrado de manera eficiente.
- El desarrollo se organizó en ramas de GitHub, permitiendo trabajar en diferentes funcionalidades de forma independiente antes de integrarlas en la versión principal.



Fase de Evaluación

Se realizaron pruebas en diferentes entornos para garantizar la estabilidad de la aplicación:

- Se probaron las funcionalidades en varios emuladores de Android y en dispositivos físicos de compañeros de curso.
- Se verificó el correcto funcionamiento del flujo de navegación y la integración con Firebase.



Fase de Puesta en Producción

Se genero un APK la cual distribuí a familiares y amigos para que testearan la aplicación

Fase de Mantenimiento y Seguimiento

Actualmente han aparecido nuevos bugs, y se sigue desarrollando código para su mejora.

Proceso Cíclico

Si en cualquier fase se detectan errores o mejoras necesarias, se regresa a la fase de **Diseño e Implementación** para realizar ajustes antes de volver a probar la aplicación.

3. Confección de informes

Para ello seguiría un enfoque basado en un **servidor remoto** debido a que JasperReports no se ejecuta directamente en Android.

3.1. Preparación y Diseño de los Informes

Primero, diseñaría las plantillas de los informes en iReports o Jaspersoft Studio. Crearía un archivo .jrxml que definiría la estructura del informe, incluyendo:

- Un encabezado con el nombre del usuario y la fecha del informe.
- Una tabla dinámica donde se listarían los ejercicios registrados, con datos como nombre, peso y repeticiones.
- Un gráfico de barras o líneas para visualizar el progreso del usuario en el tiempo.

3.2. Implementación del Servidor para Generar Informes

Dado que JasperReports no puede ejecutarse directamente en Android, configuraría un servidor en Java con Spring Boot o Tomcat, que sería el encargado de procesar los datos y generar los informes en distintos formatos (PDF, CSV, XLSX).

1. Configuraría JasperReports en el servidor y almacenaría las plantillas .jrxml.
2. Implementaría un API REST que recibiría solicitudes desde FitnessApp.
3. Extraería los datos de Firebase en el backend y los pasaría a JasperReports.
4. Generaría el informe y devolvería el archivo al usuario en el formato deseado.

3.3. Integración en Fitness App

En la aplicación, añadiría una sección donde el usuario pueda generar y descargar su informe. Incluiría una interfaz con:

- Un botón para "Generar Informe", que enviaría una solicitud al servidor.
- Opciones para seleccionar el formato del informe (PDF, Excel, CSV).
- Una barra de progreso mientras se genera el informe.
- Una notificación para descargar el archivo cuando esté listo.

3.4. Exportación y Distribución de los Informes

Cuando el usuario reciba su informe, le daría opciones para:

- **Descargar el archivo** en su dispositivo.
- **Compartirlo por correo electrónico** o **Google Drive**.
- **Visualizarlo directamente en la app** usando una librería de PDF como **PDFView**.

4. Documentación de la Aplicación

Para asegurar un desarrollo estructurado y un mantenimiento eficiente, **Fitness App** cuenta con distintos tipos de documentación. Desde la documentación técnica interna hasta los manuales de usuario, cada sección tiene un propósito clave en la comprensión y uso de la aplicación.

Documentación Técnica

Interna (Código comentado):

- Se añadieron comentarios en puntos clave del código para facilitar su comprensión y mantenimiento.
- Se documentaron variables, funciones y procedimientos sin caer en comentarios redundantes.

```
// Método para editar un ejercicio existente en la base de datos
public void editExercise() {

    // Declaración de variables
    String actual_name;
    TextInputEditText name, weigth, reps;

    // Se obtiene la instancia del usuario registrado mediante el Singleton RegisterUser
    RegisterUser registerUser = RegisterUser.getInstance();
    String userId = registerUser.getUser().getName(); // Se obtiene el ID del usuario actual

    // Se obtiene el nombre actual del ejercicio que se va a editar desde el intent
    actual_name = getIntent().getStringExtra("name").toLowerCase();

    // Se referencian los campos de entrada de datos en la interfaz
    name = findViewById(R.id.e_inpt_name);
    weigth = findViewById(R.id.e_inpt_weigth);
    reps = findViewById(R.id.e_inpt_reps);

    // Se verifica si alguno de los campos está vacío antes de proceder con la edición
    if (isEmpty(name) || isEmpty(weigth) || isEmpty(reps)) {
        Toast.makeText(EditExerciseActivity.this, "RELLENA TODOS LOS CAMPOS", Toast.LENGTH_SHORT).show();
    } else {

        // Se obtiene la instancia de la base de datos de Firebase
        FirebaseFirestore database = FirebaseFirestore.getInstance();
        Source src = Source.SERVER;
```

Externa (Manual): "*ANEXO 1 Manual de usuario***"**

Para facilitar el uso de la aplicación, se ha elaborado un **manual de navegación** que explica detalladamente cómo moverse por **FitnessApp**, cubriendo todas las funciones principales y resolviendo posibles dudas del usuario.

El manual incluye:

- Explicación paso a paso sobre cómo acceder a las distintas secciones de la aplicación.
- Instrucciones para registrar, editar y eliminar ejercicios.

Además, la **instalación y configuración** de **Fitness App** ya ha sido detallada en la sección correspondiente del trabajo, asegurando que los usuarios puedan instalar la aplicación sin inconvenientes.

Gracias a esta documentación, cualquier usuario podrá comprender rápidamente el funcionamiento de la aplicación sin necesidad de conocimientos técnicos previos.

5. Distribución de Aplicaciones

El proceso de distribución de **FitnessApp** ya fue explicado en el apartado de instalación del trabajo, donde se detalló cómo generar y desplegar el **APK en dispositivos Android**. Sin embargo, en este apartado se vuelve a explicar con mayor profundidad para reforzar la importancia de un empaquetado eficiente.

Para instalar **FitnessApp**, los pasos son los siguientes:

1. **Descarga del APK en el dispositivo.**
2. **Habilitación de fuentes desconocidas si es necesario.**
3. **Ejecución del instalador con una pantalla de confirmación.**
4. **Finalización con un mensaje de éxito y opción para abrir la aplicación.**

Este método permite que la aplicación pueda ser instalada en cualquier dispositivo Android sin necesidad de utilizar la Play Store.

Además, **FitnessApp** cuenta con un empaquetado adecuado, incluyendo sus **librerías, ficheros ejecutables y elementos multimedia**, garantizando que la aplicación funcione de manera óptima en distintos dispositivos.

5.1. Componentes de la Aplicación y Empaquetado

Para distribuir la aplicación de manera eficiente, se han considerado los siguientes elementos:

- **Librerías y Bibliotecas:** Se han integrado librerías externas, como **Firebase**, para el almacenamiento de datos y autenticación.
- **Ficheros Ejecutables:** La aplicación se distribuye en formato **APK**, asegurando compatibilidad con dispositivos Android.
- **Elementos Multimedia:** Se han optimizado imágenes y gráficos en formatos **PNG y SVG** para mejorar el rendimiento.

Estos componentes permiten que la aplicación funcione sin problemas en distintos dispositivos.

6. Generación de interfaces a partir de documentos XML

En **FitnessApp**, las interfaces han sido diseñadas utilizando **XML en Android Studio**, lo que permite una estructura clara y escalable para definir la apariencia y disposición de los elementos en pantalla.

6.1. Análisis y Edición del Documento XML en FitnessApp

El diseño de las pantallas en **FitnessApp** se basa en archivos XML que contienen:

- **Etiquetas:** Definen cada componente de la interfaz, como TextView, Button o EditText.
- **Atributos:** Especifican propiedades como el tamaño, color, o márgenes de los elementos.
- **Valores:** Determinan los textos, dimensiones y comportamientos de los componentes.
- **Eventos:** Se configuran en XML para asociar acciones a botones y otros elementos interactivos.

```
<EditText
    android:id="@+id/login_email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce tu email"
    android:textColor="@color/white"
    android:inputType="textEmailAddress"/>
```

- La etiqueta <EditText> representa un campo de entrada de texto.
- Los atributos **definen el tamaño, el color del texto y el tipo de entrada**.
- El valor del **hint** indica el texto de ayuda que aparecerá en el campo antes de que el usuario escriba.

6.2. Generación de Código para Diferentes Plataformas

Si bien **FitnessApp** ha sido desarrollada exclusivamente para Android, el uso de **XML** facilita su adaptación a otras plataformas en **futuras mejoras**.

En caso de expandir la aplicación, se podrían considerar las siguientes opciones:

- **Conversión a una aplicación web**, transformando los archivos XML en **HTML + CSS** para crear una versión accesible desde navegadores.
- **Uso de Flutter**, donde los layouts en XML podrían ser traducidos a archivos **Dart**, permitiendo que la aplicación sea compatible tanto con **iOS como con Android**.
- **Exportación de gráficos en formato SVG**, asegurando que los elementos visuales sean reutilizables en otros entornos sin perder calidad.

Estas posibilidades permitirían que **FitnessApp** evolucione y llegue a más usuarios sin necesidad de rediseñar completamente su interfaz.

7. Realización de Pruebas

Para garantizar el correcto funcionamiento de **FitnessApp**, se llevaron a cabo diversas pruebas durante el desarrollo. Estas pruebas permitieron detectar errores, mejorar la estabilidad y optimizar la experiencia del usuario antes de su distribución.

7.1. El Proyecto de Desarrollo y Objetivos de las Pruebas

Durante el desarrollo de **FitnessApp**, se siguieron las siguientes fases:

1. **Planificación:** Definición de funcionalidades clave.
2. **Diseño:** Creación de interfaces y estructura de datos en Firebase.
3. **Implementación:** Desarrollo del código en Android Studio con Java
4. **Pruebas:** Evaluación del correcto funcionamiento de la aplicación.
5. **Puesta en Producción:** Generación del APK final.

El objetivo principal de las pruebas fue garantizar que todas las funcionalidades operaran correctamente y sin errores. Se emplearon dos enfoques:

- **Pruebas de caja negra:** Se verificó que las funciones respondieran correctamente sin analizar el código interno.
- **Pruebas de caja blanca:** Se revisó la ejecución interna del código para detectar posibles fallos en la lógica de programación.

7.2. Tipos de Pruebas Realizadas

Para evaluar la calidad de la aplicación, se realizaron diferentes pruebas:

1. Depuración con Debugger

El proceso de pruebas se llevó a cabo mediante el **depurador de Android Studio** y la **depuración USB** en un dispositivo Android real. Esto permitió monitorear el comportamiento del código en tiempo real y detectar errores con mayor precisión.

2. Pruebas de Integración

Se verificó la comunicación entre módulos, asegurando que la integración con Firebase funcionara correctamente en tiempo real. Se probaron acciones como el registro de usuario, el guardado de ejercicios y la actualización de datos.

3. Pruebas de Regresión

Cada vez que se realizaba una modificación en el código, se volvía a ejecutar la aplicación en el emulador y en un dispositivo físico para garantizar que los cambios no afectaran otras funciones.

6. Pruebas en Emuladores y Dispositivos Físicos

Se probó la aplicación en varios emuladores de Android Studio y en dispositivos físicos mediante depuración USB, asegurando su funcionamiento en diferentes entornos y resolviendo problemas específicos en cada uno.

7. Pruebas de Usuario

Se compartió el APK con compañeros del curso para que probaran la aplicación en sus propios dispositivos, identificando posibles mejoras en la interfaz y la usabilidad.

Las pruebas realizadas mediante **depuración en emuladores y dispositivos físicos** fueron fundamentales para detectar errores y mejorar la estabilidad de **FitnessApp**. Gracias a este proceso, se pudo optimizar la aplicación antes de su versión final, asegurando una mejor experiencia de usuario.

8. Imágenes y Software de Gestión de Recursos Gráficos en FitnessApp

Para el diseño de la aplicación y su documentación, utilicé herramientas como **Miro y Canva**, lo que permitió una planificación visual efectiva de la interfaz y la creación de un manual de usuario bien estructurado.

8.1. Herramientas de Gestión de Recursos Gráficos

En el desarrollo de **FitnessApp**, utilicé:

- **Miro:** Para organizar y diseñar el flujo de navegación de la aplicación.
- **Canva:** Para crear el **manual de usuario y la portada del trabajo**, asegurando un diseño atractivo y claro.

8.2. Tipos de Recursos Gráficos Utilizados

Para mantener una interfaz limpia y optimizada, utilicé diferentes tipos de imágenes:

- **Rasterizadas (PNG, JPEG):** Usadas en elementos visuales de la app.
- **Vectoriales (SVG):** Para iconos sin pérdida de calidad.

Garantizando una **buena calidad visual sin afectar el rendimiento de la aplicación**.

8.3. Optimización de Imágenes

Para evitar un consumo excesivo de memoria, se aplicaron buenas prácticas como:

- **Uso de imágenes en formatos adecuados:** PNG para gráficos con transparencia y JPEG para imágenes más livianas.

8.4. Integración de Recursos Gráficos

Para mejorar la experiencia del usuario en la aplicación, consideré:

- **Uso de recursos locales:** Se almacenaron imágenes en la aplicación para evitar cargas innecesarias desde internet, como las imágenes comunes del perfil del usuario o los iconos usados en los botones.
- **Carga eficiente:** Opté por que las imágenes se mostraran solo cuando fueran necesarias, mejorando el rendimiento.

Una posible mejora a futuro sería implementar **lazy loading** para cargar imágenes bajo demanda y optimizar aún más la app.

Gracias a **Miro y Canva**, pude estructurar mejor tanto el diseño de la app como la documentación. Además, al optimizar los recursos gráficos, se logró un equilibrio entre **calidad visual y rendimiento**, asegurando una experiencia fluida para el usuario.