**GRADO EN INGENIERÍA INFORMÁTICA**

PROGRAMACIÓN DE DISPOSITIVOS MÓVILES


UNIVERSIDAD
DE GRANADA

# Práctica 1 : Aplicación Android: Generador de ejercicios por acrhivo

David Serrano Domínguez (*davidserrano07@correo.ugr.es*)

5 de mayo de 2025

# Índice



# Objetivos

La práctica consiste en desarrollar una aplicación móvil que funcione completamente de forma local, sin necesidad de conexión a un servidor externo. Para esta segunda entrega, se acordó implementar una app de entrenamiento que permita a los usuarios importar un archivo enviado por su entrenador, el cual contendrá la rutina de ejercicios. La aplicación deberá permitir iniciar, pausar y finalizar cada ejercicio, llevando un control detallado del tiempo dedicado a cada uno. Al finalizar el entrenamiento, los resultados podrán ser exportados y compartidos con el entrenador, quien podrá revisar los tiempos individuales y el tiempo total del entrenamiento.

ENLACE AL REPOSITORIO DE GITHUB : Repositorio de las prácticas

# 1. Descripción de las funcionalidades de la aplicación

El primer paso a la hora de realizar esta práctica fue establecer las funcionalidades básicas que quería que se tuvieran, es decir, lo que se llama el **MVP(Producto Mínimo Viable)**, en este caso se decidió que lo mínimo sería poder importar una rutina de ejercicios desde un archivo json del móvil y que al cargarlos en la aplicación mostrara para cada uno de estos ejercicios su información que en este caso será:

- **Nombre**
- **Orden de realización**
- **Descripción**
- **Tiempo de realización**
- **Duración estimada del ejercicio**
- **Estado del ejercicio(empezado, pausado o finalizado**

Además de la importación de la rutina, el poder controlar el entrenamiento, esto implica el poder iniciar, pausar o terminar cada ejercicio, además de poder ver el tiempo total de entreno que se lleva, la cantidad de ejercicios realizados del total y si hay algún ejercicio en progreso en ese momento.
Por último, el poder exportar los ejercicios con los valores del entrenamiento de manera rápida y sencilla para así poder compartirlo con cualquier persona, sea el entrenador o cualquier otra persona.

# 2. Tecnologias Usadas para la implementación

A continuación se explicará las tecnologías escogidas para la realización de la aplicación móvil y se explicará el motivo de la elección de las mismas.

Para el desarrollo de la app se decidió usar la librería de **Javascript** llamada **React**, concretamente **React Native** que es una versión de dicha librería enfocada al **desarrollo móvil** y que permite compilar las aplicaciones para ejecutarlas en **cualquier sistema operativo**.

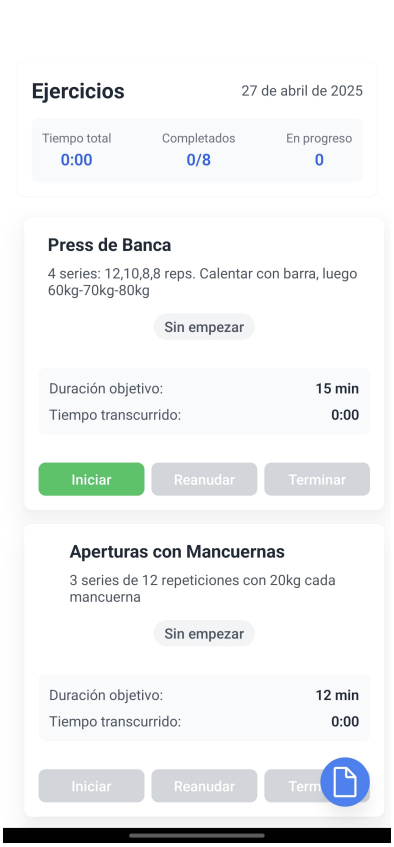
Además, tras leer la **documentación oficial** ellos mismos recomiendan usar el framework de **Expo** para crear aplicaciones móviles de **React Native** de manera más sencilla y rápida; por tanto, se decidió seguir dicha recomendación y usar dicho **framework** para el desarrollo.

A continuación se deja la documentación de ambas tecnologías:

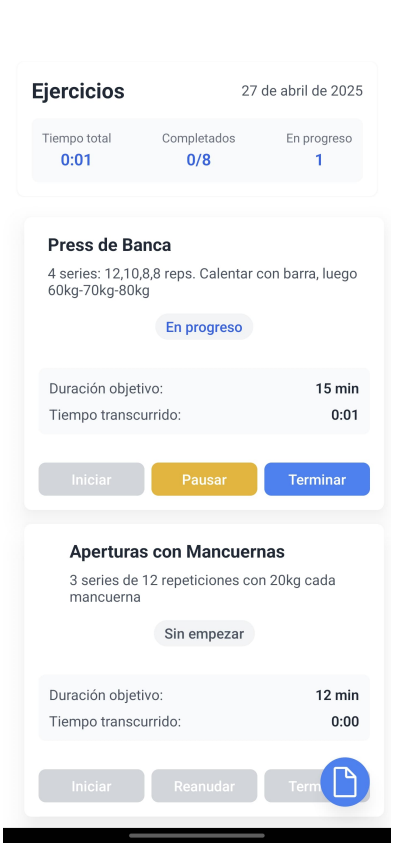
- **Documentación de React Native**
- **Documentación de Expo**

# 3. Pantallas principales de la aplicación e implementación

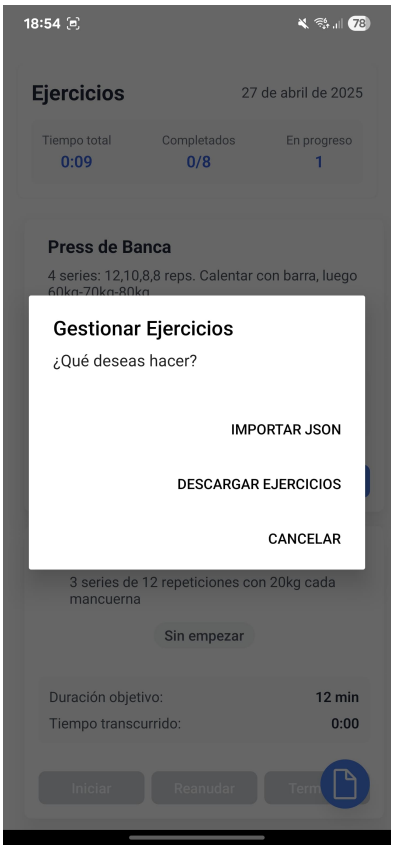
La aplicación de una única pantalla principal en la que se encuentra el grueso de la aplicación.
Nos encontramos en al parte superior con los datos del entrenamiento actual, donde podemos ver **el tiempo total**, **ejercicios realizados** y **si hay algún ejercicio en progreso**




Pantalla principal




Control de ejercicios




Menú de archivos

En esta página tendríamos la lista de ejercicios a realizar que se importarían al **subir el archivo a la aplicación** y que nos permitiría ver de manera rápida información general de todo el entrenamiento en la parte superior y luego información específica de cada ejercicio como sería **nombre, descripción, duración estimada y duración real** de cada uno de los ejercicios.

Por otro lado nos encontramos con los controles de cada uno de los ejercicios los cuales serían 3 principalmente que son **iniciar**, **pausar** y **terminar** que permitirían al usuario de manera fácil y sencilla poder tener el control del ejercicio durante su entrenamiento de manera **cómoda e intuitiva**

## 3.1. Componentes principales

Vamos a explicar los dos componente principales de la aplicación que permiten su funcionamiento.

### 3.1.1. CardExercise

El primer componente y el más importante es el de de **CardExercise** ya que este será el componente que tendrá la información de cada uno de los ejercicios y que nos va a permitir su control, es decir, **inicio, pausa y fin** permitiendonos así manter durante el entrenamiento el control de cada uno de los ejercicios a realizar. A continuación vamos a proceder a explicar las **cuatro funciones principales** que nos permiten dicho comportamiento.


Tarjeta de ejercicio

```
    const updateGlobalExercise = useCallback((updatedExercise: Partial<Exercise>) => {
        setExercises(prevExercises =>
            prevExercises.map(ex =>
                ex.id === currentExercise.id
                    ? { ...ex, ...updatedExercise }
                    : ex
            )
        );
    }, [currentExercise.id, setExercises]);

    const handleStart = () => {
        try {
            if (!isPreviousExerciseCompleted()) {
                alert('Debes completar el ejercicio anterior primero');
                return;
            }

            const updates = {
                start: true,
                paused: false,
                elapsedTime: 0
            };

            setExercise(prev => ({
                ...prev,
                ...updates
            }));
            setElapsedTime(0);
            setDoingExercise(true);
            updateGlobalExercise(updates);
        } catch(error) {
```

```
            console.error('Error in handleStart:', error);
        }
    };

    const handlePause = () => {
        try {
            const updates = {
                paused: !exercise.paused,
                elapsedTime: elapsedTime
            };
            setExercise(prev => ({
                ...prev,
                ...updates
            }));
            setDoingExercise(!doingExercise);
            updateGlobalExercise(updates);
        } catch(error) {
            console.error('Error in handlePause:', error);
        }
    };

    const handleFinish = () => {
        try {
            const updates = {
                start: true,
                finish: true,
                paused: false,
                elapsedTime: elapsedTime
            };
            setExercise(prev => ({
                ...prev,
                ...updates
            }));
            setDoingExercise(false);
            updateGlobalExercise(updates);
        } catch(error) {
            console.error('Error in handleFinish:', error);
        }
    };
```

Empezaremos explicando **updateGlobalExercise** el cual es el método encargado de cada vez que se actualice un ejercicio guardar su estado permitiéndonos así no perderlo, ya que al modificar este estado en el componente principal de la función lo detectará y lo almacenará en el **AsynStorage** teniendo así guardado el último estado de cada uno de los ejercicios cuando estos modifiquen cualquiera de sus valores.
Para ellos hacemos uso un **useEffect** que tenemos en el componente padre que haría lo siguiente:

```
const saveExercises = async (newExercises: Exercise[]) => {
  try {
    await AsyncStorage.setItem(STORAGE_KEY, JSON.stringify(newExercises));
  } catch (error) {
    console.error('Error saving exercises:', error);
  }
};

useEffect(() => {
  if(exercises){
    saveExercises(exercises);
  }
},[exercises]);
```

Cuando se ejecutará el efecto comprobamos si hay ejercicios y en caso de que sí llamaremos a la función **saveExercises** que se encargaría de guardarlo en nuestro **almacenamiento.**

Una vez explicado la función de **updateGlobalExercise** pasaremos a explicar las funciones de **handleStart, handlePause y handleFinish** que tienen comportamientos parecidos y lo que varía es el estado que modifican. En el caso de **start**, establecería el estado del tiempo del ejercicio a 0, el estado de iniciado a true y llamar a la función de actualizar los **ejercicios** con los nuevos valores del ejercicio.

Por otro lado, la función de **finish** lo que haría sería establecer el booleano de **finalizado** a true, almacenar el tiempo realizado del ejercicio y de nuevo llamar a la función de actualizar los ejercicios que explicamos al principio.

Por último, tenemos la función de pausar que establece el estado de pausa del ejercicio a true.

De esta manera, gracias a estas cuatro funciones, tendríamos el control y valores de los ejercicios en todo momento de manera sencilla.

### 3.1.2. HeadInfoElement

El segundo y último componente que tendremos será el de **HeadInfoElement** que permitirá al usuario tener información general del usuario en cada momento de manera **rápida** y en un solo vistazo.


Tarjeta de ejercicio

Como podemos observar esta formado por **tres** valores diferentes que para representar cada uno de ellos se usa el componente del que estamos hablando que esta formado por el siguiente código.

```
import { Text, View } from "react-native";

export default function HeadInfoElement({text, data} : {text: String, data: String}){

    return(
        <View className="text-center">
            <Text className="text-sm text-gray-500">{text}</Text>
            <Text className="text-lg text-center font-semibold text-blue-600">
                {data}
            </Text>
        </View>
    )
}
```

Y luego en el componente padre se pasaría se organizaría y pasaría la información de esta manera.

```
<View className="flex-row justify-between items-center mt-2 bg-gray-50 p-3 rounded-lg">
    <HeadInfoElement text="Tiempo total" data={formatTime(totalElapsedTime)} />
    <HeadInfoElement text="Completados" data={`${exercises.filter(ex => ex.finish).length}/
    ${exercises.length}`}  />
    <HeadInfoElement text="En progreso" data={`${exercises.filter(ex => ex.start && !ex.
    finish).length}`} />
</View>
```

El primer elemento sería el que representa el tiempo y para mostrarlo que hacemos será pasar el **tiempo total** que son segundos a un formato de **minutos y segundos** para mostrarlo de manera más fácil al usuario.
El segundo elemento sería el de **ejercicios completados** vs **totales** para ello mostramos por un lado los ejercicios con **estado finish** a true y por otro lado los ejercicios totales para que asía el usuario sepa en cada mommento los ejercicios que ha completado y que le quedan.

El último sería el de ejercicios en progreso que siempre tendrá valor **0 o 1** ya que nunca podremos tener más de un ejercicio en progreso a la vez, pero permitira al usuario de manera rápida saber si en ese momento tiene algún ejercicio activo o no.

## 3.2. Implementación de Control de archivos

Para manejar la **importación** y **exportación** de archivos se ha hecho uso de tres librerías nativas del propio framework de expo que son **expo-file-system, expo-sharing y expo-document-picker** la cuales nos permiten el poder importar **archivos** y poder **compartirlos a la hora de exportarlos**, empezaremos explicando como se ha manejado la importación de archivos en la **aplicación**

### 3.2.1. Importación de archivos

```
const handleImportExercises = async () => {
  try {
      const result = await DocumentPicker.getDocumentAsync({
          type: 'application/json',
          copyToCacheDirectory: true
      });

      if (!result.assets || !result.assets[0]) {
          Alert.alert('Error', 'No se selecciono ningun archivo');
          return;
      }

      const fileUri = result.assets[0].uri;
      const fileContent = await FileSystem.readAsStringAsync(fileUri);

      let exercises;
      try {
          exercises = JSON.parse(fileContent);
      } catch (error) {
          Alert.alert('Error', 'El archivo no contiene un JSON valido');
          return;
      }


      if (!Array.isArray(exercises)) {
          Alert.alert('Error', 'El archivo debe contener un array de ejercicios');
          return;
      }

      const orderedExercises = [...exercises].sort((a, b) => a.order - b.order);

      Alert.alert(
          'Confirmar importacion',
          ` Deseas  importar ${exercises.length} ejercicios? Esto reemplazara los
    ejercicios existentes.`,
          [
              {
                  text: 'Cancelar',
                  style: 'cancel'
              },
              {
                  text: 'Importar',
                  onPress: async () => {
                      try {

                          await AsyncStorage.setItem(STORAGE_KEY, JSON.stringify(
    orderedExercises));

                          setExercises(orderedExercises);


                          Alert.alert(
                              'Exito',
                              `Se importaron ${exercises.length} ejercicios correctamente`
```

```
                            );
                        } catch (error) {
                            Alert.alert('Error', 'No se pudieron guardar los ejercicios');
                        }
                    }
                }
            ]
        );

    } catch (error) {
        console.error('Error importing exercises:', error);
        Alert.alert(
            'Error',
            'Ocurrio un error al importar los ejercicios'
        );
    }
};
```

Lo primero que se hace para la importación del archivo es mediante el uso de la librería **DocumentPicker** obtener del sistema de archivos del móvil el archivo deseado y decirle que es de tipo **JSON** además le decimos que si lo queremos copiar en una carpeta temporal, para poder **leerlo de manera segura** evitando problemas en caso de que se **modificara o eliminara** dicho archivo cuando se este seleccionando.

Posteriormente se pasa a **comprobar** que al menos se ha **seleccionado un archivo** y leemos su contenido verificando también de que **contiene un array** ya que es el contenido que esperamos.
Una vez comprobado esto procedemos a ordenar los ejercicios por su **atributo order** que es el encargado de establecer el **orden en el que se deben** realizar los ejercicios del entrenamiento.

Una vez realizada todo esto le mostramos un **mensaje de confirmación** al usuario de que si seguro quiere importar los ejercicios y en caso de que si los almacenamos en el **AsynStorage** y guardamos en el estado actual de la aplicación.

### 3.2.2. Exportación de archivos

```
const handleDownloadExercises = async () => {
  try {
      const exercisesJson = JSON.stringify(exercises, null, 2);

      const date = new Date();
      const fileName = `ejercicios_${date.getDate()}-${date.getMonth() + 1}-${date.
  getFullYear()}.json`;

      const fileUri = `${FileSystem.documentDirectory}${fileName}`;

      await FileSystem.writeAsStringAsync(fileUri, exercisesJson);

      const canShare = await Sharing.isAvailableAsync();

      if (canShare) {
          await Sharing.shareAsync(fileUri, {
              mimeType: 'application/json',
              dialogTitle: 'Descargar Ejercicios',
              UTI: 'public.json'
          });

          Alert.alert(
              'Exito',
              'Los ejercicios se han exportado correctamente'
          );
      } else {
          Alert.alert(
              'Error',
              'Tu dispositivo no soporta la funcion de compartir archivos'
          );
      }

      await FileSystem.deleteAsync(fileUri, { idempotent: true });

  } catch (error) {
      console.error('Error downloading exercises:', error);
      Alert.alert(
          'Error',
          'No se pudieron descargar los ejercicios'
      );
  }
};
```

Para la **exportación de archivos** lo primero es transformar el array de ejercicios en un **JSON** una vez hecho esto establecemos el **nombre del archivo** con la fecha del dia de hoy, para posteriormente escribir en el archivo el contenido del **JSON** con los **ejercicios**.
Una vez que tenemos el archivo comprobamos si el dispositivo permite compartir archivos y en caso de que si entonces mediante el uso de la libreria **Sharing** decimos que queremos compartir un archivo de tipo **JSON** y una vez compartido avisamos el usuario que el archivo se ha compartido correctamente y **borramos el archivo temporal** que habíamos creado para compartirlo

# 4. Conclusiones

Como conclusión diría que la aplicación es **bastante práctica** ya que mediante algo tan sencillo como que el entrenador mande una rutina solamente y que con **importarla** de manera rápida y sencilla como es el caso se cargarían todos los ejercicios y el único objetivo sería poner **foco en el entreno** y tener solo que **iniciar**, **pausar**(en caso de ser necesario) y **terminar** los ejercicios y una vez hecho esto compartirlo con nuestro entrenador para que pueda **ver tiempos** y ejercicios que se han **completado** durante el entreno.