ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

Desarrollo Aplicaciones Móviles

ASIGNATURA:	Desarrollo de Aplicaciones Móviles
PROFESOR:	Ing. Juan Carlos Gonzalez MSc.
PERÍODO ACADÉMICO:	2025-B

Taller 6

Consumo de APIS con React Native Expo

NOMBRES:

Josué Eduard Guerra Lovato

OBJETIVO GENERAL

Desarrollar una aplicación móvil multiplataforma con React Native Expo que permita a los usuarios explorar y visualizar información detallada sobre personajes y planetas del universo Dragon Ball, implementando una arquitectura limpia escalable y buenas prácticas de desarrollo.

OBJETIVOS ESPECÍFICOS

- Implementar un sistema de consumo de API REST utilizando Axios con interceptores para el manejo centralizado de peticiones HTTP, estados de carga y errores, permitiendo la visualización paginada de personajes y planetas desde la Dragon Ball API.
- Desarrollar una interfaz de usuario intuitiva y responsive con navegación por pestañas (tabs), rutas dinámicas para detalles de personajes, componentes reutilizables, y funcionalidades de interacción como paginación infinita y pull-to-refresh para mejorar la experiencia del usuario.

Taller Práctico: Aplicación Dragon Ball con React Native y Expo

En este taller construiremos una aplicación móvil completa para visualizar información sobre personajes y planetas del universo Dragon Ball. Aprenderemos:

- Arquitectura limpia en React Native
-Consumo de APIs REST con Axios
- Hooks personalizados para lógica reutilizable
- Navegación con Expo Router
- Gestión de estados de carga y errores
- Diseño responsive y atractivo

La app consume tres endpoints REST de la Dragon Ball API:

- GET /characters?page={page}&limit={limit} para listar personajes paginados,
- GET /characters/{id} para el detalle completo (incluye transformaciones)
- GET /planets?page={page}&limit={limit} para listar planetas con paginación.

API utilizada: Dragon Ball API (*https://dragonball-api.com/api*)

Esta aplicación no usa UseCases porque la lógica es simple (solo consumir y mostrar datos de la API). Los **UseCases son necesarios cuando tienes lógica de negocio compleja**, múltiples fuentes de datos, validaciones elaboradas o reglas de negocio que combinan varios servicios. Para esta app educativa CRUD, la arquitectura **Hook → Service → API es suficiente** y más fácil de entender.

Configuración Inicial del Proyecto

```
npx create-expo-app@latest dragon-ball-app

cd dragon-ball-app
```

Vamos a organizar nuestro código siguiendo **Clean Architecture (Borramos carpetas como constants, scripts, la carpeta components nativa que viene con la creación del proyecto y dejamos así**:

Paso 1: Instalación de Dependencias

npm install axios

¿Qué es cada paquete?

axios

- Cliente HTTP para hacer peticiones a APIs

- Más completo que *fetch* nativo

- Interceptores para logging y manejo de errores

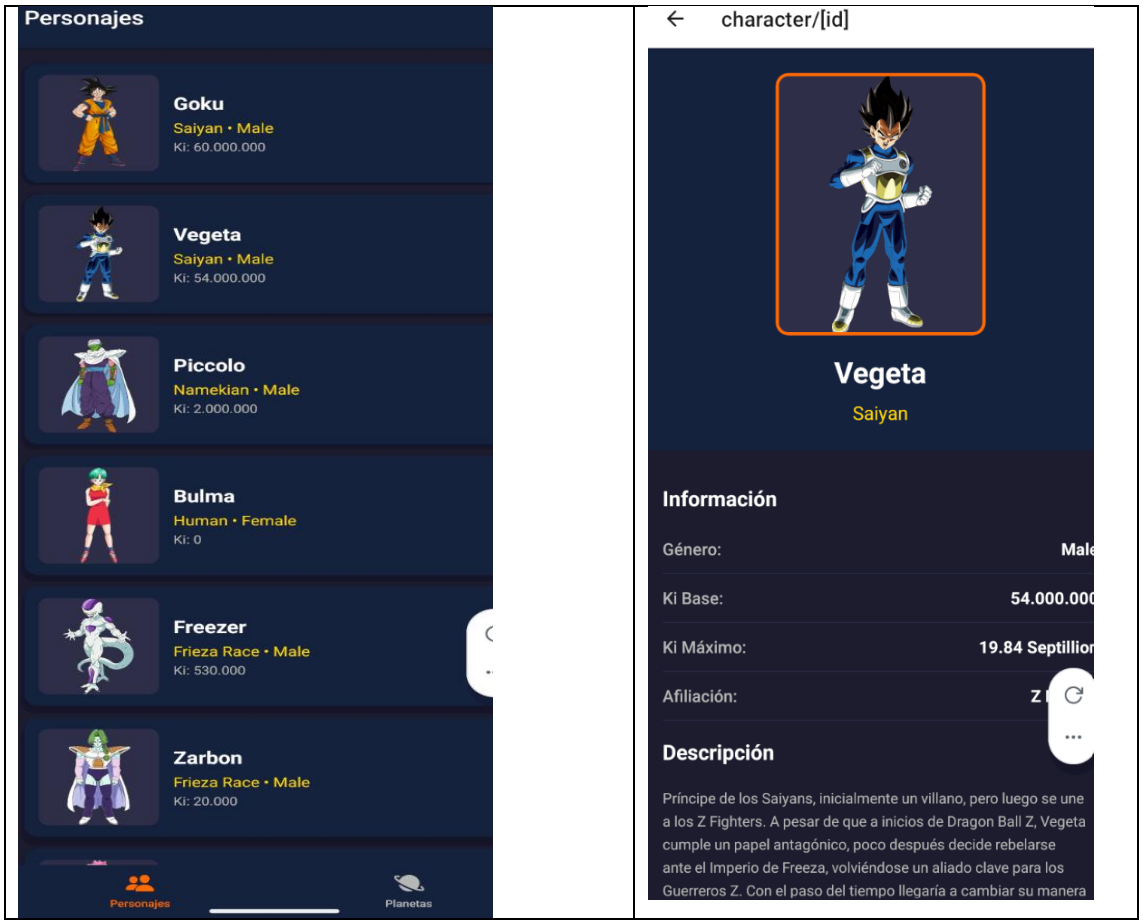
- Transformación automática de JSON

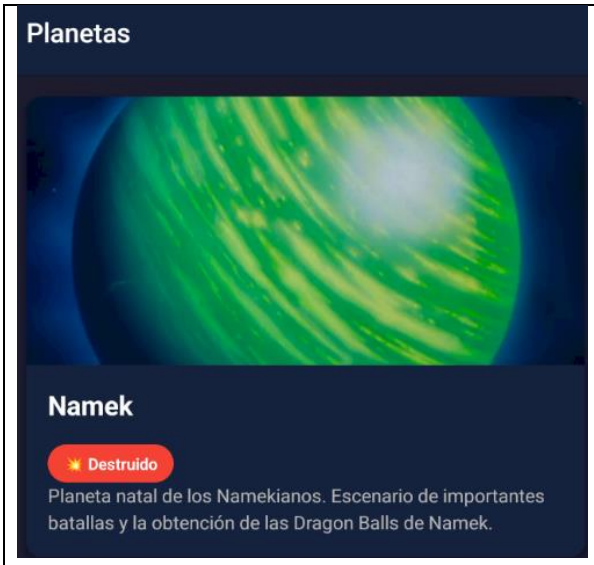
Alternativas:

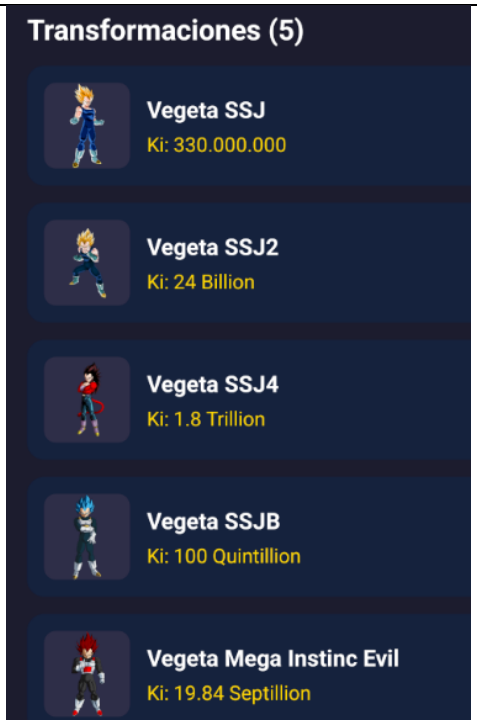
-*fetc`* (nativo, pero más verboso)

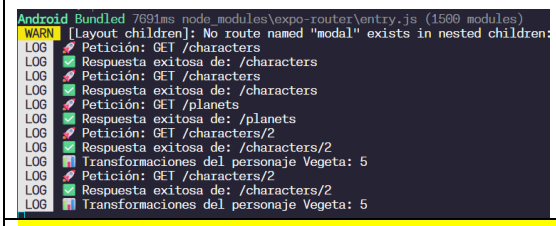
- *react-query* (para caché avanzado)

Ejecutar la aplicación , de haber realizado correctamente los pasos debe mostrarse lo siguiente:









Captura de Pantalla del estudiante, adjuntada como evidencia

Talleres > Taller5_11_08_2025 > dragon-ball-app > app > _layout.tsx > ...

```tsx
import { Stack } from 'expo-router';
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import 'react-native-reanimated';
import { Colors } from '../src/presentation/styles/globalStyles';

export default function RootLayout() {
    return (
        <>
            <Stack screenOptions={{ headerStyle: { backgroundColor: Colors.cardBackground, }, heade

                <Stack.Screen name="(tabs)" options={{ headerShown: false, }} />

                <Stack.Screen name="characters/[id]" options={{ title: "Detalle del Personaje" }} /

            </Stack>

            <StatusBar style="light" />
        </>
    );
}
```

Talleres > Taller5_11_08_2025 > dragon-ball-app > app > characters > [id].tsx > ...

```tsx
import { ErrorState } from "@/components/ErrorState";
import { LoadingState } from "@/components/LoadingState";
import { Image } from "expo-image";
import { useLocalSearchParams } from "expo-router";
import React from "react";
import { FlatList, ScrollView, Text, View } from "react-native";
import { Transformation } from "../../src/domain/models/Transformation.model";
import { useCharacterDetail } from "../../src/presentation/hooks/useCharacterDetail";
import { globalStyles } from "../../src/presentation/styles/globalStyles";

/**
 * Pantalla de detalle de personaje
 */
export default function CharacterDetailScreen() {
    const { id } = useLocalSearchParams<{ id: string }>();

    // Validación del ID
    if (!id) {
        return <LoadingState message="Cargando detalles..." />;
    }

    const characterId = parseInt(id as string, 10);

    if (isNaN(characterId)) {
        return <ErrorState message="ID de personaje inválido" />;
    }

    const { character, transformations, loading, error } =
        useCharacterDetail(characterId);

    if (loading) {
        return <LoadingState message="Cargando detalles..." />;
    }
```

Talleres > Taller5_11_08_2025 > dragon-ball-app > components > CharacterCard.tsx > ...

```tsx
import { Image } from "expo-image";
import { useRouter } from "expo-router";
import React from "react";
import { Text, TouchableOpacity, View } from "react-native";
import { Character } from "../src/domain/models/Character.model";
import { globalStyles } from "../src/presentation/styles/globalStyles";

interface CharacterCardProps {
    character: Character;
}

export const CharacterCard: React.FC<CharacterCardProps> = ({ character }) => {
    const router = useRouter();

    const handlePress = () => {
        router.push(`/characters/${character.id}`);
    };

    return (
        <TouchableOpacity
            style={globalStyles.characterCard}
            onPress={handlePress}
            activeOpacity={0.7}
        >
            <Image
                source={{ uri: character.image }}
                style={globalStyles.characterImage}
                contentFit="contain"
                transition={300}
            />

            <View style={globalStyles.characterInfo}>
                <Text style={globalStyles.characterName} numberOfLines={1}>
                    {character.name}
```

Talleres > Taller5_11_08_2025 > dragon-ball-app > components > ErrorState.tsx > ...

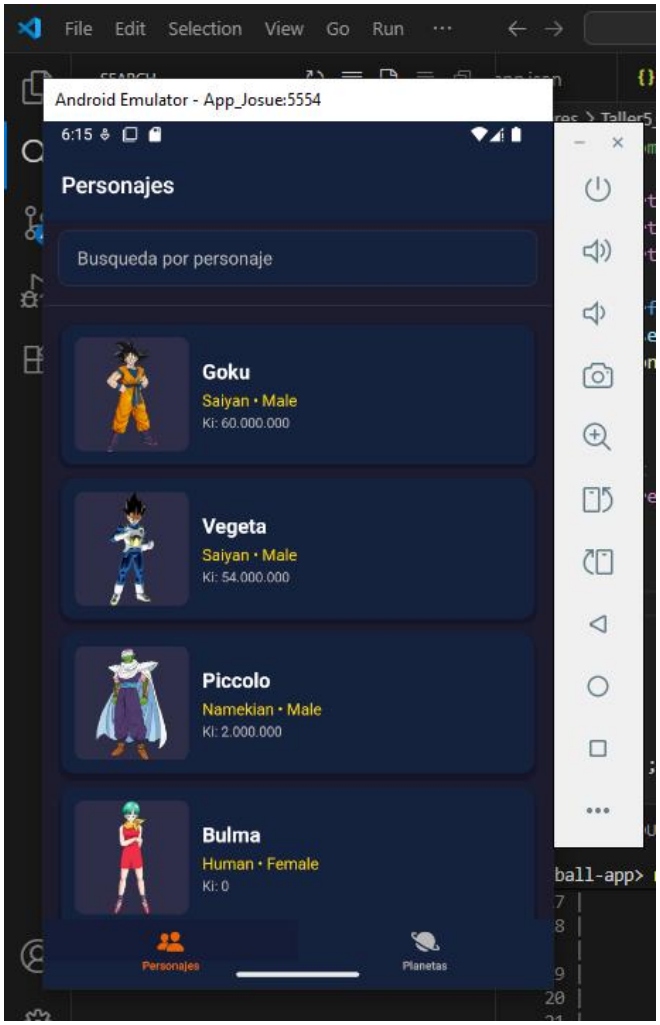
```tsx
import React from "react";
import { Text, View } from "react-native";
import { globalStyles } from "../src/presentation/styles/globalStyles";

interface ErrorStateProps {
    message: string;
}

export const ErrorState: React.FC<ErrorStateProps> = ({ message }) => {
    return (
        <View style={globalStyles.errorContainer}>
            <Text style={globalStyles.errorText}>{message}</Text>
        </View>
    );
};

```

Talleres > Taller5_11_08_2025 > dragon-ball-app > components > PlanetCard.tsx > ...

```tsx
import { Image } from "expo-image";
import React from "react";
import { Text, View } from "react-native";
import { Planet } from "../src/domain/models/Planet.model";
import { globalStyles } from "../src/presentation/styles/globalStyles";

interface PlanetCardProps {
    planet: Planet;
}

export const PlanetCard: React.FC<PlanetCardProps> = ({ planet }) => {
    return (
        <View style={globalStyles.planetCard}>
            <Image
                source={{ uri: planet.image }}
                style={globalStyles.planetImage}
                contentFit="cover"
                transition={300}
            />

            <View style={globalStyles.planetInfo}>
                <Text style={globalStyles.planetName}>{planet.name}</Text>
                <Text style={globalStyles.planetDescription} numberOfLines={3}>
                    {planet.description}
                </Text>

                <View style={globalStyles.planetStatus}>
                    <View
                        style={[
                            globalStyles.statusBadge,
                            planet.isDestroyed
                                ? globalStyles.statusDestroyed
                                : globalStyles.statusActive,
                        ]}
```

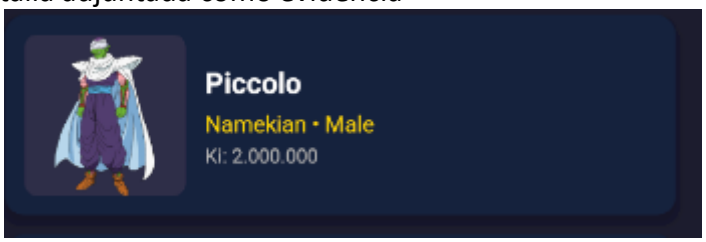
RETO DEL ESTUDIANTE(Obligatorio)

1. Agregar un campo de búsqueda para filtrar personajes



2. Mostrar más información de cada personaje en sus tarjetas

Captura de Pantalla adjuntada como evidencia



Entregables:

- Archivo pdf, con las evidencias de terminación del taller y de los dos retos
- Enlace de GitHub del repositorio con la aplicación https://github.com/josdank/App_Moviiles-Josue/tree/7e27c8a343470c7a5d0b961728d7e76e1bc6e2d0/Talleres/Taller5_11_08_2025/dragon-ball-app