



# 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:**

```

dragon-ball-app/
├── app/                                # Rutas de la aplicación (Expo Router)
│   ├── (tabs)/                           # Rutas con navegación por tabs
│   │   └── _layout.tsx                  # Configuración de tabs
│   ├── index.tsx                         # Tab 1: Lista de personajes
│   └── explore.tsx                      # Tab 2: Lista de planetas
│       └── character/
│           └── [id].tsx                 # Pantalla de detalle (ruta dinámica)
│
│   └── components/                      # Componentes reutilizables
│       ├── CharacterCard.tsx          # Tarjeta de personaje
│       ├── PlanetCard.tsx            # Tarjeta de planeta
│       ├── LoadingState.tsx          # Estado de carga
│       └── ErrorState.tsx            # Estado de error
│
└── src/                                 # Lógica de negocio
    ├── data/                            # Capa de datos
    └── services/
        ├── api.config.ts               # Configuración de Axios
        ├── character.service.ts
        └── planet.service.ts
    └── domain/                          # Modelos de dominio
        └── models/
            ├── Character.model.ts
            ├── Transformation.model.ts
            └── Planet.model.ts
    └── presentation/                   # Capa de presentación
        ├── hooks/                      # Custom hooks
        ├── useCharacters.ts
        ├── useCharacterDetail.ts
        └── usePlanets.ts
    └── styles/
        └── globalStyles.ts             # Estilos compartidos
└── package.json

```

**DRAGON-BALL-APP**

- > .claude
- > .expo
- > .vscode
- > app
  - > (tabs)
    - \_layout.tsx
    - explore.tsx
    - index.tsx
  - > character
    - [id].tsx
    - \_layout.tsx
- > assets
- > components
  - CharacterCard.tsx
  - ErrorState.tsx
  - LoadingState.tsx
  - PlanetCard.tsx
- > hooks
- > node\_modules
- > src
  - > data\services
    - TS api.config.ts
    - TS character.service.ts
    - TS planet.service.ts
  - > domain\models
    - TS Character.model.ts
    - TS Planet.model.ts
    - TS Transformation.model.ts
  - > presentation
    - > hooks
      - TS useCharacterDetail.ts
      - TS useCharacters.ts
      - TS usePlanets.ts
    - > styles
      - TS globalStyles.ts
- > .gitignore
- > app.json
- > eslint.config.js
- TS expo-env.d.ts
- ▶ package-lock.json
- ▶ package.json

## 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:

**Personajes**

	<b>Goku</b> Saiyan • Male Ki: 60.000.000
	<b>Vegeta</b> Saiyan • Male Ki: 54.000.000
	<b>Piccolo</b> Namekian • Male Ki: 2.000.000
	<b>Bulma</b> Human • Female Ki: 0
	<b>Freezer</b> Frieza Race • Male Ki: 530.000
	<b>Zarbon</b> Frieza Race • Male Ki: 20.000

**← character/[id]**

**Vegeta**  
Saiyan

**Información**

Género:	Male
Ki Base:	54.000.000
Ki Máximo:	19.84 Septillion
Afiliación:	Z Fighters

**Descripción**

Príncipe de los Saiyans, inicialmente un villano, pero luego se une a los Z Fighters. A pesar de que a inicios de Dragon Ball Z, Vegeta cumple un papel antagonico, poco despues decide rebelarse ante el Imperio de Freeza, volviéndose un aliado clave para los Guerreros Z. Con el paso del tiempo llegaría a cambiar su manera

## Planetas



### Namek

⚠ Destruido

Planeta natal de los Namekianos. Escenario de importantes batallas y la obtención de las Dragon Balls de Namek.

## Transformaciones (5)



**Vegeta SSJ**

Ki: 330.000.000



**Vegeta SSJ2**

Ki: 24 Billion



**Vegeta SSJ4**

Ki: 1.8 Trillion



**Vegeta SSJB**

Ki: 100 Quintillion



**Vegeta Mega Instinc Evil**

Ki: 19.84 Septillion

```
Android Bundled 7691ms node_modules\expo-router\entry.js (1500 modules)
WARN [Layout children]: No route named "modal" exists in nested children:
LOG 🔍 Petición: GET /characters
LOG ✅ Respuesta exitosa de: /characters
LOG 🔍 Petición: GET /characters
LOG ✅ Respuesta exitosa de: /characters
LOG 🔍 Petición: GET /planets
LOG ✅ Respuesta exitosa de: /planets
LOG 🔍 Petición: GET /characters/2
LOG ✅ Respuesta exitosa de: /characters/2
LOG 📈 Transformaciones del personaje Vegeta: 5
LOG 🔍 Petición: GET /characters/2
LOG ✅ Respuesta exitosa de: /characters/2
LOG 📈 Transformaciones del personaje Vegeta: 5
LOG
```

## Captura de Pantalla del estudiante, adjuntada como evidencia

```
Talleres > Taller5_11_08_2025 > dragon-ball-app > app > _layout.tsx > ...
  1 import { Stack } from 'expo-router';
  2 import { StatusBar } from 'expo-status-bar';
  3 import React from 'react';
  4 import 'react-native-reanimated';
  5 import { Colors } from '../src/presentation/styles/globalStyles';
  6
  7 export default function RootLayout() {
  8   return (
  9     <>
10       <Stack screenOptions={{ headerStyle: { backgroundColor: Colors.cardBackground, }, headerTitle: '' }}>
11         <Stack.Screen name="(tabs)" options={{ headerShown: false, }} />
12         <Stack.Screen name="characters/[id]" options={{ title: "Detalle del Personaje" }} />
13       </Stack>
14       <StatusBar style="light" />
15     </>
16   );
17 }
```

```
Talleres > Taller5_11_08_2025 > dragon-ball-app > app > characters > [id].tsx > ...
1 import { ErrorState } from "@/components/ErrorState";
2 import { LoadingState } from "@/components>LoadingState";
3 import { Image } from "expo-image";
4 import { useLocalSearchParams } from "expo-router";
5 import React from "react";
6 import { FlatList, ScrollView, Text, View } from "react-native";
7 import { Transformation } from "../../src/domain/models/Transformation.model";
8 import { useCharacterDetail } from "../../src/presentation/hooks/useCharacterDetail";
9 import { globalStyles } from "../../src/presentation/styles/globalStyles";
10
11 /**
12 * Pantalla de detalle de personaje
13 */
14 export default function CharacterDetailScreen() {
15     const { id } = useLocalSearchParams<{ id: string }>();
16
17     // Validación del ID
18     if (!id) {
19         return <LoadingState message="Cargando detalles..." />;
20     }
21
22     const characterId = parseInt(id as string, 10);
23
24     if (isNaN(characterId)) {
25         return <ErrorState message="ID de personaje inválido" />;
26     }
27
28     const { character, transformations, loading, error } =
29         useCharacterDetail(characterId);
30
31     if (loading) {
32         return <LoadingState message="Cargando detalles..." />;
33     }
34 }
```

```
Talleres > Taller5_11_08_2025 > dragon-ball-app > components > CharacterCard.tsx > ...
1 import { Image } from "expo-image";
2 import { useRouter } from "expo-router";
3 import React from "react";
4 import { Text, TouchableOpacity, View } from "react-native";
5 import { Character } from "../../src/domain/models/Character.model";
6 import { globalStyles } from "../../src/presentation/styles/globalStyles";
7
8 interface CharacterCardProps {
9     character: Character;
10 }
11
12 export const CharacterCard: React.FC<CharacterCardProps> = ({ character }) => {
13     const router = useRouter();
14
15     const handlePress = () => {
16         router.push(`/characters/${character.id}`);
17     };
18
19     return (
20         <TouchableOpacity
21             style={globalStyles.characterCard}
22             onPress={handlePress}
23             activeOpacity={0.7}
24         >
25             <Image
26                 source={{ uri: character.image }}
27                 style={globalStyles.characterImage}
28                 contentFit="contain"
29                 transition={300}
30             />
31             <View style={globalStyles.characterInfo}>
32                 <Text style={globalStyles.characterName} numberOfLines={1}>
33                     {character.name}
34                 </Text>
35             </View>
36         </TouchableOpacity>
37     );
38 }
```

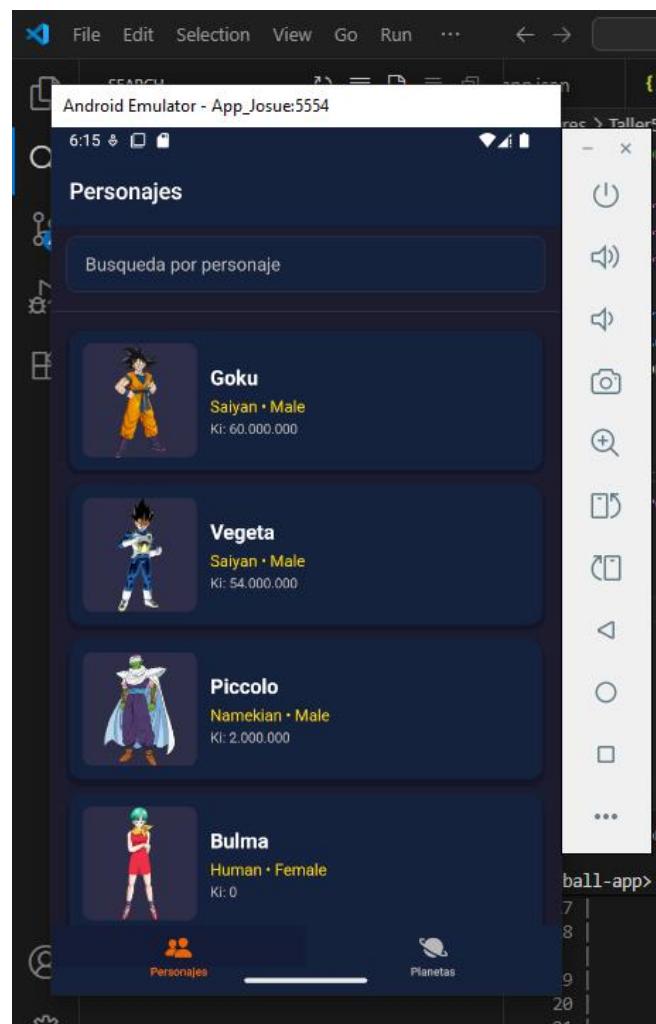
```
Talleres > Taller5_11_08_2025 > dragon-ball-app > components > ErrorState.tsx > ...
1 import React from "react";
2 import { Text, View } from "react-native";
3 import { globalStyles } from "../src/presentation/styles/globalStyles";
4
5 interface ErrorStateProps {
6   message: string;
7 }
8
9 export const ErrorState: React.FC<ErrorStateProps> = ({ message }) => {
10   return (
11     <View style={globalStyles.errorContainer}>
12       <Text style={globalStyles.errorText}>{message}</Text>
13     </View>
14   );
15 };
16
```

```
Talleres > Taller5_11_08_2025 > dragon-ball-app > components > PlanetCard.tsx > ...
1 import { Image } from "expo-image";
2 import React from "react";
3 import { Text, View } from "react-native";
4 import { Planet } from "../src/domain/models/Planet.model";
5 import { globalStyles } from "../src/presentation/styles/globalStyles";
6
7 interface PlanetCardProps {
8   planet: Planet;
9 }
10
11 export const PlanetCard: React.FC<PlanetCardProps> = ({ planet }) => {
12   return (
13     <View style={globalStyles.planetCard}>
14       <Image
15         source={{ uri: planet.image }}
16         style={globalStyles.planetImage}
17         contentFit="cover"
18         transition={300}
19       />
20
21       <View style={globalStyles.planetInfo}>
22         <Text style={globalStyles.planetName}>{planet.name}</Text>
23         <Text style={globalStyles.planetDescription} numberOfLines={3}>
24           {planet.description}
25         </Text>
26
27         <View style={globalStyles.planetStatus}>
28           <View
29             style={[
30               globalStyles.statusBadge,
31               planet.isDestroyed
32                 ? globalStyles.statusDestroyed
33                 : globalStyles.statusActive,
34             ]}
35         </View>
36       </View>
37     </View>
38   );
39 };
40
```

## RETO DEL ESTUDIANTE(Obligatorio)



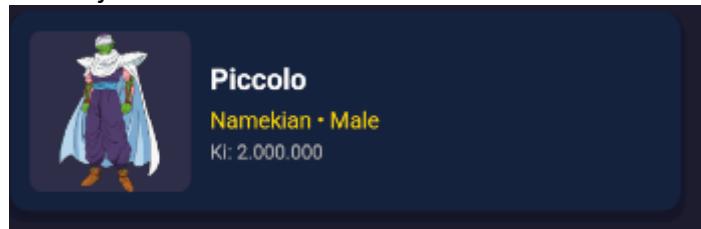
1. Agregar un campo de búsqueda para filtrar personajes



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\\_1\\_08\\_2025/dragon-ball-app](https://github.com/josdank/App_Moviiles-Josue/tree/7e27c8a343470c7a5d0b961728d7e76e1bc6e2d0/Talleres/Taller5_1_08_2025/dragon-ball-app)