PeriferiApp

Aplicación móvil desarrollada con **React Native**, orientada a la gestión y visualización de información de usuarios, favoritos y detalles, siguiendo buenas prácticas de arquitectura y desarrollo de software moderno.

Tecnologías principales

- **React Native 0.80.1**: Framework principal para desarrollo multiplataforma.
- **TypeScript**: Tipado estático para mayor robustez y mantenibilidad.
- **React Navigation**: Navegación entre pantallas (stack y tabs).
- **Zustand**: Manejo de estado global simple y eficiente.
- **React Query**: Gestión de datos remotos, caché y sincronización.
- **AsyncStorage**: Persistencia local de datos.
- **React Native Vector Icons**: Iconografía moderna y personalizable.
- **React Hook Form**: Manejo de formularios y validaciones.
- **Jest & Testing Library**: Pruebas unitarias y de hooks.
- **ESLint & Prettier**: Linting y formateo automático de código.

Arquitectura y estructura

- **Modular y escalable**: Separación clara por capas y dominios.
  - `/src/presentation`: Pantallas, componentes UI y navegación.
  - `/src/hooks`: Hooks personalizados para lógica reutilizable.
  - `/src/store`: Estado global (Zustand) para autenticación, usuarios y favoritos.
  - `/src/data/api`: Llamadas y hooks para consumo de APIs.
  - `/src/data/storage`: Persistencia local (AsyncStorage).
  - `/src/domain/models`: Tipos y modelos de negocio.
- **Principio de responsabilidad única**: Cada módulo tiene una función clara.
- **Separación de lógica de UI y datos**: La lógica de negocio y acceso a datos está desacoplada de la interfaz.
- **Uso de hooks**: Para lógica compartida, side effects y gestión de estado local/global.

Buenas prácticas y principios aplicados

- **Clean Code**: Nombres descriptivos, funciones pequeñas y legibles.
- **DRY (Don't Repeat Yourself)**: Reutilización de lógica mediante hooks y utilidades.
- **KISS (Keep It Simple, Stupid)**: Soluciones simples y directas.
- **Testing**: Pruebas unitarias para hooks, stores y lógica de almacenamiento.
- **Formateo y linting**: Uso de ESLint y Prettier en todo el proyecto.
- **Control de versiones**: `.gitignore` bien configurado para excluir archivos temporales y dependencias.
- **Internacionalización**: Preparado para soportar múltiples idiomas si se requiere.

- **Accesibilidad**: Uso de componentes accesibles y buenas prácticas de UI.

Dependencias principales

- `react-native`
- `react`
- `@react-navigation/native` y stacks/tabs
- `@tanstack/react-query`
- `zustand`
- `@react-native-async-storage/async-storage`
- `react-native-vector-icons`
- `react-hook-form`

Pruebas

- **Jest** para pruebas unitarias.
- **@testing-library/react-native** para pruebas de hooks y componentes.
- Cobertura de pruebas para lógica de autenticación, favoritos y hooks personalizados.

Ejecución y despliegue

Android

```
npm run android
```

iOS

```
npm run ios
```

Generar APK de producción

```
cd android
./gradlew assembleRelease
```

El APK estará en `android/app/build/outputs/apk/release/app-release.apk`

Estructura de carpetas

```
periferiapp/
├── android/
├── ios/
├── src/
│   ├── presentation/
│   ├── hooks/
│   ├── store/
│   ├── data/
│   │   ├── api/
│   │   ├── storage/
│   ├── domain/
│   │   └── models/
├── App.js / App.tsx
├── package.json
├── README.md
└── ...
```

Recursos y enlaces útiles

- React Native Docs
- Zustand
- React Query
- React Navigation
- Testing Library
- AsyncStorage
