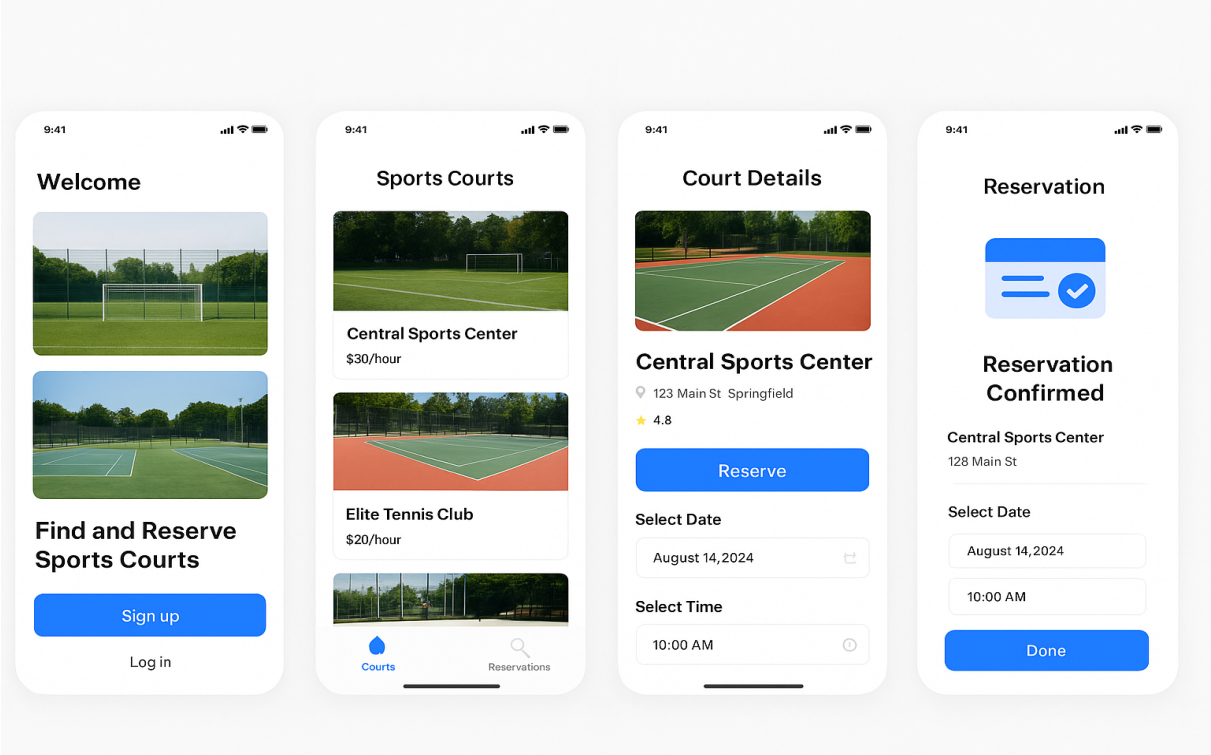
App de Reserva de Canchas y Deportes

Proyecto completo en React Native (Expo) – Paso a paso sin omitir nada



Autor: ChatGPT • Versión: 1.0 • Licencia: Uso educativo/demostración

Índice

1. Objetivo y alcance

Construir una app móvil de reserva de canchas (fútbol, tenis, pádel, básquet, etc.) usando Expo/React Native, sin backend propio. Los datos se almacenan localmente con AsyncStorage. Opcionalmente se puede sincronizar con Google Sheets en otra iteración. El objetivo es mostrar un producto moderno, con alto impacto visual (Swiper y animaciones fluidas) y un flujo completo de reserva.

2. Stack, librerías y por qué (GSAP vs Reanimated)

- Expo (React Native) para acelerar desarrollo y build.
- React Navigation para navegación en stack y tabs.
- react-native-swiper o reanimated-carousel para carruseles.
- react-native-reanimated + react-native-gesture-handler para animaciones de alto rendimiento (equivalente a GSAP en web).
- AsyncStorage para persistencia local sin backend.
- (Opcional) react-native-maps para ver clubes en un mapa.
- (Opcional) date-fns para manejo de fechas.

Nota importante sobre GSAP: GSAP está orientado a web/DOM. En React Native nativo no hay DOM, por lo tanto se usan Reanimated y Gesture Handler para lograr efectos equivalentes de forma performante.

3. Setup del entorno (Expo)

```
# Requisitos previos
# - Node.js LTS instalado
# - Expo CLI (usá npx para no instalar global):

npx create-expo-app reservapp --template
# Elegí 'Blank (TypeScript)' para tipado.

cd reservapp

# Instalar dependencias principales
npx expo install react-native-gesture-handler react-native-reanimated @react-navigation/native @rea
npx expo install react-native-safe-area-context react-native-screens
npm i react-native-swiper --save
npm i @react-native-async-storage/async-storage date-fns

# Configuración Reanimated (babel)
# en babel.config.js agregá el plugin 'react-native-reanimated/plugin' al final del array de plugin

# Habilitar Reanimated en index.js si corresponde (ya viene con Expo).
```

4. Estructura de carpetas

```
reservapp/
  app/                    # (si usás Expo Router) o src/
  src/
    assets/
      images/             # imágenes locales (fallback si falla API de fotos)
      lottie/
    components/
      CourtCard.tsx
      Header.tsx
      SwiperHero.tsx
```

```
    screens/
      HomeScreen.tsx
      CourtsScreen.tsx
      CourtDetailScreen.tsx
      DateTimeScreen.tsx
      ConfirmScreen.tsx
      MyBookingsScreen.tsx
      MapScreen.tsx        # opcional
    context/
      BookingContext.tsx
    data/
      mockData.ts
    lib/
      storage.ts
      api.ts               # (opcional) integraciones con APIs de fotos
    theme/
      colors.ts
      spacing.ts
    navigation/
      index.tsx
  app.json / app.config.ts
  babel.config.js
  package.json
```

5. Diseño UI/UX (Swiper, paleta, tipografía)

Paleta sugerida: azul 600 (#2563eb), azul 700 (#1d4ed8), gris 900 (#0f172a), grises neutros para fondos. Tipografía por defecto del sistema (San Francisco / Roboto). Bordes redondeados (12-16) y sombras suaves. Hero con Swiper en Home y cards con fotos grandes; CTA primario 'Reservar'.

```
// SwiperHero.tsx (ejemplo básico)
import React from 'react';
import { View, Image, Dimensions } from 'react-native';
import Swiper from 'react-native-swiper';

const { width } = Dimensions.get('window');

export default function SwiperHero() {
  return (
    <View style={{ height: 220 }}>
      <Swiper autoplay loop showsPagination>
        {[
          require('../assets/images/cancha1.jpg'),
          require('../assets/images/cancha2.jpg'),
          require('../assets/images/cancha3.jpg'),
        ].map((src, i) => (
          <Image key={i} source={src} style={{ width, height: 220 }} resizeMode="cover" />
        ))}
      </Swiper>
    </View>
  );
}
```

6. Modelado de datos y tipos

```
// Tipos base
export type Sport = 'futbol' | 'tenis' | 'padel' | 'basquet';

export interface Venue {
  id: string;
  name: string;
```

```
  address: string;
  lat: number;
  lng: number;
  photos: string[];     // URLs o require(...)
  rating?: number;
}

export interface Court {
  id: string;
  venueId: string;
  sport: Sport;
  name: string;
  pricePerHour: number;
  surface?: string;     // 'sintético', 'cemento', 'madera'
  availableSlots: string[]; // '2025-09-05T18:00:00-03:00'
}

export interface Booking {
  id: string;
  courtId: string;
  venueId: string;
  userName: string;
  dateISO: string;      // fecha/hora ISO
  durationHours: number;
  totalPrice: number;
  notes?: string;
}
```

7. Gestión de estado y persistencia (Context + AsyncStorage)

```
// context/BookingContext.tsx
import React, { createContext, useContext, useReducer, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { Booking } from '../types';

type Action =
  | { type: 'ADD_BOOKING'; payload: Booking }
  | { type: 'CANCEL_BOOKING'; payload: string }
  | { type: 'LOAD'; payload: Booking[] };

interface State { bookings: Booking[]; }
const initialState: State = { bookings: [] };

function reducer(state: State, action: Action): State {
  switch (action.type) {
    case 'ADD_BOOKING':
      return { bookings: [...state.bookings, action.payload] };
    case 'CANCEL_BOOKING':
      return { bookings: state.bookings.filter(b => b.id !== action.payload) };
    case 'LOAD':
      return { bookings: action.payload };
    default:
      return state;
  }
}

const Ctx = createContext<{state: State; dispatch: React.Dispatch<Action>}>({state: initialState, d

export function BookingProvider({ children }: { children: React.ReactNode }) {
  const [state, dispatch] = useReducer(reducer, initialState);
```

```
  useEffect(() => {
    (async () => {
      const raw = await AsyncStorage.getItem('BOOKINGS');
      if (raw) dispatch({ type: 'LOAD', payload: JSON.parse(raw) });
    })();
  }, []);

  useEffect(() => { AsyncStorage.setItem('BOOKINGS', JSON.stringify(state.bookings)); }, [state.boo

  return <Ctx.Provider value={{ state, dispatch }}>{children}</Ctx.Provider>;
}

export const useBookings = () => useContext(Ctx);
```

8. Navegación (React Navigation)

```
// navigation/index.tsx
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import HomeScreen from '../screens/HomeScreen';
import CourtsScreen from '../screens/CourtsScreen';
import MyBookingsScreen from '../screens/MyBookingsScreen';

const Tab = createBottomTabNavigator();

export default function RootNavigation() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="Inicio" component={HomeScreen} />
        <Tab.Screen name="Canchas" component={CourtsScreen} />
        <Tab.Screen name="Mis reservas" component={MyBookingsScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

9. Pantallas y flujo completo

• HomeScreen: hero con Swiper, CTA 'Buscar canchas'. • CourtsScreen: listado de canchas (CourtCard) con foto, precio, rating y botón 'Ver detalle'. • CourtDetailScreen: galería, info, selector de fecha/hora y botón 'Reservar'. • DateTimeScreen: selección exacta de fecha (DateTimePicker) y duración. • ConfirmScreen: resumen + botón Confirmar → guarda en AsyncStorage → animación success. • MyBookingsScreen: lista de reservas, cancelación con confirmación. • MapScreen (opcional): react-native-maps con marcadores de venues.

```
// components/CourtCard.tsx
import React from 'react';
import { View, Text, Image, Pressable } from 'react-native';

export default function CourtCard({ photo, title, price, onPress }: any) {
  return (
    <Pressable onPress={onPress} style={{ backgroundColor: 'white', borderRadius: 16, overflow: 'hi
      <Image source={photo} style={{ height: 160, width: '100%' }} resizeMode="cover" />
      <View style={{ padding: 12 }}>
        <Text style={{ fontSize: 18, fontWeight: '700' }}>{title}</Text>
        <Text style={{ color: '#334155', marginTop: 4 }}>${'{'}price{'}'}/hora</Text>
      </View>
    </Pressable>
  );
```

```
}
```

10. Animaciones (Reanimated) – equivalentes a GSAP

```
// Ejemplo: aparecer con opacidad y translateY
import Animated, { useSharedValue, useAnimatedStyle, withTiming, withSpring } from 'react-native-re
import { useEffect } from 'react';
import { View } from 'react-native';

export default function FadeUp({ children }: any) {
  const o = useSharedValue(0);
  const y = useSharedValue(20);
  const style = useAnimatedStyle(() => ({
    opacity: o.value,
    transform: [{ translateY: y.value }],
  }));

  useEffect(() => {
    o.value = withTiming(1, { duration: 400 });
    y.value = withSpring(0, { damping: 16, stiffness: 120 });
  }, []);

  return <Animated.View style={style}>{children}</Animated.View>;
}
```

11. Carga de imágenes (Unsplash / Pexels)

Para obtener fotos reales de canchas podés usar las APIs gratuitas de Unsplash o Pexels. Crea una cuenta de desarrollador, obtené una API Key y realizá búsquedas por términos como 'tennis court', 'soccer field', 'padel court', 'basketball court'. Respetá sus términos de uso y atribución.

```
// lib/api.ts (Unsplash de ejemplo)
const UNSPLASH_KEY = 'TU_API_KEY';

export async function searchUnsplash(query: string) {
  const res = await fetch(`https://api.unsplash.com/search/photos?query=${query}&per_page=10`, {
    headers: { Authorization: `Client-ID ${UNSPLASH_KEY}` },
  });
  const json = await res.json();
  return json.results?.map((p: any) => p.urls.small) as string[];
}
```

12. Validaciones y edge cases

• Bloquear selección de horarios pasados. • Evitar solapes: si un horario ya está reservado, no permitir duplicados. • Confirmación antes de cancelar una reserva. • Modo offline: mostrar placeholder de imágenes si falla la API. • Accesibilidad: tamaños mínimos táctiles y labels claros.

13. Pruebas y checklist de QA

1) Crear reserva completa (feliz). 2) Cancelar reserva. 3) Reintentar imágenes si no cargan. 4) Rotación de pantalla. 5) Rendimiento en listas largas. 6) Persistencia tras cerrar y reabrir la app.

14. Build y publicación

```
# Compilación
npx expo prebuild
npx expo run:android
npx expo run:ios
```

```
# Producción con EAS (opcional)
# https://docs.expo.dev/eas/
```

15. Roadmap de mejoras

• Integración con Google Calendar (enlace ICS). • Confirmación con QR. • Notificaciones push (Expo Notifications). • Filtros por superficie, precio y rating. • Mapas y geolocalización. • Pasarela de pago (solo demo/redirección).

16. Créditos, fuentes y medidas de canchas (resumen)

Deporte	Medidas oficiales (aprox.)	Fuente
Fútbol (internacional)	100–110 m x 64–75 m	IFAB Laws of the Game
Fútbol (recomendación FIFA)	105 m x 68 m	FIFA Stadium Guidelines
Tenis (pista)	23.77 m x 8.23 m (singles) / 10.97 m (dobles)	ITF
Básquet (FIBA)	28 m x 15 m	FIBA
Pádel	20 m x 10 m	FIP
Vóley (indoor)	18 m x 9 m	FIVB
Beach vóley	16 m x 8 m	FIVB

Fotos: Usá Unsplash o Pexels con sus APIs (atribución requerida). Consultá sus documentos para los términos. Incluí siempre créditos dentro de tu app (por ej., en el detalle de la cancha).