ricomincia da qui correggi importazione da event list screen a event map screen

# header fisso

import React from "react";

import {

  View,

  Text,

  StyleSheet,

  StatusBar,

  Image,

  Platform,

} from "react-native";

export const HeaderFisso: React.FC = () => {

  return (

    <>

      {/\* StatusBar blu con icone bianche \*/}

      <StatusBar backgroundColor="#0D65BE" />

      <View style={[styles.headerContainer]}>

        <View style={styles.headerContent}>

          <Text style={styles.headerText}>

            Protezione Civile | Regione Calabria

          </Text>

          <Image source={require("./Logo.png")} style={styles.logo} />

        </View>

      </View>

    </>

  );

};

const styles = StyleSheet.create({

  headerContainer: {

    backgroundColor: "#0D65BE",

    paddingVertical: 15,

    paddingHorizontal: 15,

    justifyContent: "center",

  },

  headerContent: {

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "space-between",

  },

  headerText: {

    color: "#fff",

    fontSize: 15,

    textAlign: "left",

  },

  logo: {

    width: 35,

    height: 35,

  },

});

# config js corretto

export const CONFIG = {

  // URL base per le API

  API\_BASE\_URL\_DEV: "https://pc2.dev.schema31.it",

  API\_BASE\_URL\_PROD: "https://pc2.protezionecivilecalabria.it",

  // Endpoint API

  ENDPOINTS: {

    LOGIN: "/api/users/\_session",

    REGISTER: "/registra",

    VERIFY\_EMAIL: "/api/email/verify",

    VERIFY\_PHONE: "/api/cellular/verify",

    EVENTS\_FIND: "/api/events/\_find",

    EVENTS\_CREATE: "/api/events",

    EVENTS\_VIEW: "/api/events",

    EVENTS\_UPDATE: "/api/events",

  },

  // Credenziali di test

  TEST\_CREDENTIALS: {

    fiscalCode: "LLNRBL80A01G273E",

    password: "89411809",

  },

  // URL esterni

  EXTERNAL\_URLS: {

    WEATHER\_BULLETIN:

      "https://protezionecivilecalabria.it/bollettino-meteo/bollettino-meteo.php",

    REGION\_WEBSITE: "https://www.regione.calabria.it",

    PRIVACY\_POLICY: "https://protezionecivilecalabria.it/privacy",

    CONTACTS: "https://protezionecivilecalabria.it/contatti",

  },

  // Configurazione mappa

  MAP\_CONFIG: {

    INITIAL\_REGION: {

      latitude: 39.3081, // Centro Calabria

      longitude: 16.2539,

      latitudeDelta: 2.0,

      longitudeDelta: 2.0,

    },

  },

  // Ambiente corrente (dev/prod)

  ENVIRONMENT: \_\_DEV\_\_ ? "dev" : "prod",

};

// Funzione per ottenere l'URL base corretto

export const getBaseUrl = () => {

  return CONFIG.ENVIRONMENT === "dev"

    ? CONFIG.API\_BASE\_URL\_DEV

    : CONFIG.API\_BASE\_URL\_PROD;

};

# app navigator salvatore

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import { Ionicons } from '@expo/vector-icons';

// Import delle schermate (da creare)

import HomeScreen from '../screens/HomeScreen';

import LoginScreen from '../screens/LoginScreen';

import RegisterScreen from '../screens/RegisterScreen';

import EventsMapScreen from '../screens/EventsMapScreen';

import EventsListScreen from '../screens/EventsListScreen';

import CreateReportScreen from '../screens/CreateReportScreen';

import WeatherBulletinScreen from '../screens/WeatherBulletinScreen';

import CommunicationsScreen from '../screens/CommunicationsScreen';

import ProfileScreen from '../screens/ProfileScreen';

const Stack = createStackNavigator();

const Tab = createBottomTabNavigator();

// Navigazione principale con tab

function MainTabNavigator() {

  return (

    <Tab.Navigator

      screenOptions={({ route }) => ({

        tabBarIcon: ({ focused, color, size }) => {

          let iconName;

          if (route.name === 'Eventi') {

            iconName = focused ? 'map' : 'map-outline';

          } else if (route.name === 'Segnala') {

            iconName = focused ? 'add-circle' : 'add-circle-outline';

          } else if (route.name === 'Bollettino') {

            iconName = focused ? 'cloud' : 'cloud-outline';

          } else if (route.name === 'Comunicazioni') {

            iconName = focused ? 'chatbubbles' : 'chatbubbles-outline';

          } else if (route.name === 'Profilo') {

            iconName = focused ? 'person' : 'person-outline';

          }

          return <Ionicons name={iconName} size={size} color={color} />;

        },

        tabBarActiveTintColor: '#FF6B35',

        tabBarInactiveTintColor: 'gray',

        headerStyle: {

          backgroundColor: '#FF6B35',

        },

        headerTintColor: '#fff',

        headerTitleStyle: {

          fontWeight: 'bold',

        },

      })}

    >

      <Tab.Screen

        name="Eventi"

        component={EventsMapScreen}

        options={{ title: 'Eventi' }}

      />

      <Tab.Screen

        name="Segnala"

        component={CreateReportScreen}

        options={{ title: 'Segnala' }}

      />

      <Tab.Screen

        name="Bollettino"

        component={WeatherBulletinScreen}

        options={{ title: 'Bollettino' }}

      />

      <Tab.Screen

        name="Comunicazioni"

        component={CommunicationsScreen}

        options={{ title: 'Comunicazioni' }}

      />

      <Tab.Screen

        name="Profilo"

        component={ProfileScreen}

        options={{ title: 'Profilo' }}

      />

    </Tab.Navigator>

  );

}

// Navigazione principale dell'app

export default function AppNavigator() {

  return (

    <NavigationContainer>

      <Stack.Navigator

        initialRouteName="Home"

        screenOptions={{

          headerStyle: {

            backgroundColor: '#FF6B35',

          },

          headerTintColor: '#fff',

          headerTitleStyle: {

            fontWeight: 'bold',

          },

        }}

      >

        <Stack.Screen

          name="Home"

          component={HomeScreen}

          options={{

            title: 'Protezione Civile Calabria',

            headerShown: false

          }}

        />

        <Stack.Screen

          name="Login"

          component={LoginScreen}

          options={{ title: 'Accedi' }}

        />

        <Stack.Screen

          name="Register"

          component={RegisterScreen}

          options={{ title: 'Registrati' }}

        />

        <Stack.Screen

          name="Main"

          component={MainTabNavigator}

          options={{ headerShown: false }}

        />

        <Stack.Screen

          name="EventsList"

          component={EventsListScreen}

          options={{ title: 'Lista Eventi' }}

        />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

# //app navigator ok appnavigator.js

import React from "react";

import { NavigationContainer } from "@react-navigation/native";

import { createStackNavigator } from "@react-navigation/stack";

import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";

import { Ionicons } from "@expo/vector-icons";

// Import delle schermate (da creare)

import HomeScreen from "../screens/HomeScreen";

import LoginScreen from "../screens/LoginScreen";

import RegisterScreen from "../screens/RegisterScreen";

import EventsMapScreen from "../screens/EventsMapScreen";

import EventsListScreen from "../screens/EventsListScreen";

import CreateReportScreen from "../screens/CreateReportScreen";

import WeatherBulletinScreen from "../screens/WeatherBulletinScreen";

import CommunicationsScreen from "../screens/CommunicationsScreen";

import ProfileScreen from "../screens/ProfileScreen";

const Stack = createStackNavigator();

const Tab = createBottomTabNavigator();

// Navigazione principale con tab

function MainTabNavigator() {

  return (

    <Tab.Navigator

      screenOptions={({ route }) => ({

        tabBarIcon: ({ focused, color, size }) => {

          let iconName;

          if (route.name === "Eventi") {

            iconName = focused ? "map" : "map-outline";

          } else if (route.name === "Segnala") {

            iconName = focused ? "add-circle" : "add-circle-outline";

          } else if (route.name === "Bollettino") {

            iconName = focused ? "cloud" : "cloud-outline";

          } else if (route.name === "Comunicazioni") {

            iconName = focused ? "chatbubbles" : "chatbubbles-outline";

          } else if (route.name === "Profilo") {

            iconName = focused ? "person" : "person-outline";

          }

          return <Ionicons name={iconName} size={size} color={color} />;

        },

        tabBarActiveTintColor: "#FF6B35",

        tabBarInactiveTintColor: "gray",

      })}

    >

      <Tab.Screen

        name="Eventi"

        component={EventsMapScreen}

        options={{

          title: "Eventi",

          headerShown: false,

        }}

      />

      <Tab.Screen

        name="Segnala"

        component={CreateReportScreen}

        options={{ title: "Segnala", headerShown: false }}

      />

      <Tab.Screen

        name="Bollettino"

        component={WeatherBulletinScreen}

        options={{ title: "Bollettino", headerShown: false }}

      />

      <Tab.Screen

        name="Comunicazioni"

        component={CommunicationsScreen}

        options={{ title: "Comunicazioni", headerShown: false }}

      />

      <Tab.Screen

        name="Profilo"

        component={ProfileScreen}

        options={{ title: "Profilo", headerShown: false }}

      />

    </Tab.Navigator>

  );

}

// Navigazione principale dell'app

export default function AppNavigator() {

  return (

    <NavigationContainer>

      <Stack.Navigator

        initialRouteName="Home"

        screenOptions={{

          headerStyle: {

            backgroundColor: "#FF6B35",

          },

          headerTintColor: "#fff",

          headerTitleStyle: {

            fontWeight: "bold",

          },

        }}

      >

        <Stack.Screen

          name="Home"

          component={HomeScreen}

          options={{

            title: "Protezione Civile Calabria",

            headerShown: false,

          }}

        />

        <Stack.Screen

          name="Login"

          component={LoginScreen}

          options={{ title: "Accedi", headerShown: false }}

        />

        <Stack.Screen

          name="Register"

          component={RegisterScreen}

          options={{ title: "Registrati", headerShown: false }}

        />

        <Stack.Screen

          name="Main"

          component={MainTabNavigator}

          options={{ headerShown: false }}

        />

        <Stack.Screen

          name="EventsList"

          component={EventsListScreen}

          options={{ title: "Lista Eventi" }}

        />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

# auth navigator

/\*import React from "react";

import { createStackNavigator } from "@react-navigation/stack";

// Tipi

import { AuthStackParamList } from "../types";

// Schermate

import HomeScreen from "../screens/HomeScreen";

import { LoginScreen } from "../screens/LoginScreen";

import { RegisterScreen } from "../screens/RegisterScreen";

const Stack = createStackNavigator<AuthStackParamList>();

export const AuthNavigator: React.FC = () => {

  return (

    <Stack.Navigator

      initialRouteName="Home"

      screenOptions={{

        headerStyle: {

          height: 100, // Explicitly set header height

          paddingTop: 10, // Add some padding on top if needed

        },

        headerTintColor: "#0D65BE",

        headerTitleStyle: {

          fontWeight: "bold",

          fontSize: 22, // Adjust font size if needed

        },

        headerTitleAlign: "center", // Title remains centered

      }}

    >

      <Stack.Screen

        name="Home"

        component={HomeScreen}

        options={{ title: "Accesso" }}

      />

      <Stack.Screen

        name="Login"

        component={LoginScreen}

        options={{ title: "Accedi" }}

      />

      <Stack.Screen

        name="Register"

        component={RegisterScreen}

        options={{ title: "Registrati" }}

      />

    </Stack.Navigator>

  );

};

\*/

# main navigator

/\*// MainNavigator.tsx

import React from "react";

import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";

import { createNativeStackNavigator } from "@react-navigation/native-stack";

import { Ionicons } from "@expo/vector-icons";

import EventScreen from "../screens/EventScreen";

import { ReportScreen } from "../screens/ReportScreen";

//import { ProfileScreen } from "../screens/ProfileScreen";

type TabParamList = {

  EventTab: undefined;

  ReportTab: undefined;

  ProfileTab: undefined;

};

type ReportStackParamList = {

  ReportMain: undefined;

};

const Tab = createBottomTabNavigator<TabParamList>();

const ReportStack = createNativeStackNavigator<ReportStackParamList>();

// Stack per la tab "Segnala"

const ReportStackScreen: React.FC = () => (

  <ReportStack.Navigator screenOptions={{ headerShown: false }}>

    <ReportStack.Screen name="ReportMain" component={ReportScreen} />

  </ReportStack.Navigator>

);

export const MainNavigator: React.FC = () => {

  return (

    <Tab.Navigator

      screenOptions={({ route }) => ({

        tabBarIcon: ({ color, size }) => {

          let iconName: keyof typeof Ionicons.glyphMap;

          if (route.name === "EventTab") iconName = "map-outline";

          else if (route.name === "ReportTab") iconName = "warning-outline";

          else if (route.name === "ProfileTab") iconName = "person-outline";

          else iconName = "help-outline";

          return <Ionicons name={iconName} size={size} color={color} />;

        },

        tabBarActiveTintColor: "#FF6B35",

        tabBarInactiveTintColor: "gray",

        headerShown: false,

      })}

    >

      <Tab.Screen

        name="EventTab"

        component={EventScreen}

        options={{ tabBarLabel: "Eventi" }}

      />

      <Tab.Screen

        name="ReportTab"

        component={ReportStackScreen}

        options={{ tabBarLabel: "Segnala" }}

      />

    </Tab.Navigator>

  );

};

\*/

# types.ts

/\*/ navigation/types.ts

export type ReportTabParamList = {

  ReportMain: undefined;

  ReportDescription: undefined;

  ReportDetails?: undefined; // Se aggiungi step 3

  ReportSummary?: undefined; // Se aggiungi step 4

};

export type RootStackParamList = {

  EventsScreen: undefined;

  ReportTab: {

    screen: keyof ReportTabParamList;

    params?: any;

  };

};

\*/

# bulletin screen

/\*import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  StyleSheet,

  SafeAreaView,

  ScrollView,

  ActivityIndicator,

  RefreshControl,

  TouchableOpacity,

  Alert,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import { apiService } from "../services/api";

import { HeaderFisso } from "../components/HeaderFisso";

interface WeatherData {

  location: string;

  date: string;

  temperature: {

    current: number;

    min: number;

    max: number;

  };

  condition: string;

  humidity: number;

  windSpeed: number;

  windDirection: string;

  pressure: number;

  visibility: number;

  uvIndex: number;

  forecast: DayForecast[];

  alerts: WeatherAlert[];

}

interface DayForecast {

  date: string;

  day: string;

  condition: string;

  icon: string;

  tempMin: number;

  tempMax: number;

  precipitation: number;

}

interface WeatherAlert {

  id: string;

  type: "warning" | "watch" | "advisory";

  title: string;

  description: string;

  severity: "low" | "medium" | "high";

  validFrom: string;

  validTo: string;

}

export const BulletinScreen: React.FC = () => {

  const [weatherData, setWeatherData] = useState<WeatherData | null>(null);

  const [isLoading, setIsLoading] = useState(true);

  const [refreshing, setRefreshing] = useState(false);

  const [selectedCity, setSelectedCity] = useState("Catanzaro");

  const calabrianCities = [

    "Catanzaro",

    "Cosenza",

    "Reggio Calabria",

    "Crotone",

    "Vibo Valentia",

  ];

  useEffect(() => {

    loadWeatherData();

  }, [selectedCity]);

  const loadWeatherData = async () => {

    try {

      const response = await apiService.getWeatherBulletin();

      setWeatherData(response.data);

    } catch (error) {

      console.error("Errore nel caricamento bollettino meteo:", error);

      // Dati mock per il testing

      setWeatherData(mockWeatherData);

    } finally {

      setIsLoading(false);

      setRefreshing(false);

    }

  };

  const onRefresh = () => {

    setRefreshing(true);

    loadWeatherData();

  };

  const getWeatherIcon = (condition: string): string => {

    const iconMap: { [key: string]: string } = {

      sunny: "sunny",

      cloudy: "cloudy",

      "partly-cloudy": "partly-sunny",

      rainy: "rainy",

      stormy: "thunderstorm",

      snowy: "snow",

      foggy: "cloudy-night",

    };

    return iconMap[condition] || "partly-sunny";

  };

  const getAlertColor = (severity: string): string => {

    switch (severity) {

      case "high":

        return "#FF4444";

      case "medium":

        return "#FF8800";

      case "low":

        return "#FFD700";

      default:

        return "#666";

    }

  };

  const getAlertIcon = (type: string): string => {

    switch (type) {

      case "warning":

        return "warning";

      case "watch":

        return "eye";

      case "advisory":

        return "information-circle";

      default:

        return "alert-circle";

    }

  };

  const renderCurrentWeather = () => {

    if (!weatherData) return null;

    return (

      <View style={styles.currentWeatherCard}>

        <View style={styles.currentWeatherHeader}>

          <Text style={styles.locationText}>{weatherData.location}</Text>

          <Text style={styles.dateText}>{weatherData.date}</Text>

        </View>

        <View style={styles.currentWeatherMain}>

          <View style={styles.temperatureSection}>

            <Text style={styles.currentTemp}>

              {weatherData.temperature.current}°

            </Text>

            <Text style={styles.condition}>{weatherData.condition}</Text>

          </View>

          <Ionicons

            name={getWeatherIcon(weatherData.condition.toLowerCase()) as any}

            size={80}

            color="#2E7D32"

          />

        </View>

        <View style={styles.tempRange}>

          <Text style={styles.tempRangeText}>

            Min: {weatherData.temperature.min}° | Max:{" "}

            {weatherData.temperature.max}°

          </Text>

        </View>

        <View style={styles.weatherDetails}>

          <View style={styles.detailItem}>

            <Ionicons name="water" size={16} color="#666" />

            <Text style={styles.detailText}>

              Umidità: {weatherData.humidity}%

            </Text>

          </View>

          <View style={styles.detailItem}>

            <Ionicons name="speedometer" size={16} color="#666" />

            <Text style={styles.detailText}>

              Vento: {weatherData.windSpeed} km/h {weatherData.windDirection}

            </Text>

          </View>

          <View style={styles.detailItem}>

            <Ionicons name="barbell" size={16} color="#666" />

            <Text style={styles.detailText}>

              Pressione: {weatherData.pressure} hPa

            </Text>

          </View>

          <View style={styles.detailItem}>

            <Ionicons name="eye" size={16} color="#666" />

            <Text style={styles.detailText}>

              Visibilità: {weatherData.visibility} km

            </Text>

          </View>

        </View>

      </View>

    );

  };

  const renderForecast = () => {

    if (!weatherData?.forecast) return null;

    return (

      <View style={styles.forecastCard}>

        <Text style={styles.sectionTitle}>Previsioni 5 giorni</Text>

        <ScrollView horizontal showsHorizontalScrollIndicator={false}>

          {weatherData.forecast.map((day, index) => (

            <View key={index} style={styles.forecastDay}>

              <Text style={styles.forecastDayName}>{day.day}</Text>

              <Text style={styles.forecastDate}>{day.date}</Text>

              <Ionicons

                name={getWeatherIcon(day.condition) as any}

                size={32}

                color="#2E7D32"

              />

              <Text style={styles.forecastTemp}>{day.tempMax}°</Text>

              <Text style={styles.forecastTempMin}>{day.tempMin}°</Text>

              <Text style={styles.forecastPrecip}>{day.precipitation}%</Text>

            </View>

          ))}

        </ScrollView>

      </View>

    );

  };

  const renderAlerts = () => {

    if (!weatherData?.alerts || weatherData.alerts.length === 0) return null;

    return (

      <View style={styles.alertsCard}>

        <Text style={styles.sectionTitle}>Allerte Meteo</Text>

        {weatherData.alerts.map((alert) => (

          <TouchableOpacity

            key={alert.id}

            style={[

              styles.alertItem,

              { borderLeftColor: getAlertColor(alert.severity) },

            ]}

            onPress={() => Alert.alert(alert.title, alert.description)}

          >

            <View style={styles.alertHeader}>

              <Ionicons

                name={getAlertIcon(alert.type) as any}

                size={20}

                color={getAlertColor(alert.severity)}

              />

              <Text

                style={[

                  styles.alertTitle,

                  { color: getAlertColor(alert.severity) },

                ]}

              >

                {alert.title}

              </Text>

            </View>

            <Text style={styles.alertDescription} numberOfLines={2}>

              {alert.description}

            </Text>

            <Text style={styles.alertValidity}>

              Valido dal {alert.validFrom} al {alert.validTo}

            </Text>

          </TouchableOpacity>

        ))}

      </View>

    );

  };

  const renderCitySelector = () => (

    <View style={styles.citySelectorCard}>

      <Text style={styles.sectionTitle}>Seleziona Città</Text>

      <ScrollView horizontal showsHorizontalScrollIndicator={false}>

        {calabrianCities.map((city) => (

          <TouchableOpacity

            key={city}

            style={[

              styles.cityButton,

              selectedCity === city && styles.selectedCityButton,

            ]}

            onPress={() => setSelectedCity(city)}

          >

            <Text

              style={[

                styles.cityButtonText,

                selectedCity === city && styles.selectedCityButtonText,

              ]}

            >

              {city}

            </Text>

          </TouchableOpacity>

        ))}

      </ScrollView>

    </View>

  );

  if (isLoading) {

    return (

      <SafeAreaView style={styles.container}>

        <View style={styles.loadingContainer}>

          <ActivityIndicator size="large" color="#2E7D32" />

          <Text style={styles.loadingText}>

            Caricamento bollettino meteo...

          </Text>

        </View>

      </SafeAreaView>

    );

  }

  return (

    <SafeAreaView style={styles.container}>

      <ScrollView

        style={styles.scrollView}

        refreshControl={

          <RefreshControl refreshing={refreshing} onRefresh={onRefresh} />

        }

        showsVerticalScrollIndicator={false}

      >

        <View style={styles.content}>

          <View style={styles.header}>

            <Text style={styles.title}>Bollettino Meteo</Text>

            <Text style={styles.subtitle}>Previsioni per la Calabria</Text>

          </View>

          {renderCitySelector()}

          {renderCurrentWeather()}

          {renderForecast()}

          {renderAlerts()}

        </View>

      </ScrollView>

    </SafeAreaView>

  );

};

// Dati mock per il testing

const mockWeatherData: WeatherData = {

  location: "Catanzaro, Calabria",

  date: new Date().toLocaleDateString("it-IT", {

    weekday: "long",

    year: "numeric",

    month: "long",

    day: "numeric",

  }),

  temperature: {

    current: 24,

    min: 18,

    max: 28,

  },

  condition: "Partly Cloudy",

  humidity: 65,

  windSpeed: 12,

  windDirection: "NE",

  pressure: 1013,

  visibility: 10,

  uvIndex: 6,

  forecast: [

    {

      date: "15/09",

      day: "Oggi",

      condition: "partly-cloudy",

      icon: "partly-sunny",

      tempMin: 18,

      tempMax: 28,

      precipitation: 20,

    },

    {

      date: "16/09",

      day: "Dom",

      condition: "sunny",

      icon: "sunny",

      tempMin: 19,

      tempMax: 30,

      precipitation: 5,

    },

    {

      date: "17/09",

      day: "Lun",

      condition: "rainy",

      icon: "rainy",

      tempMin: 16,

      tempMax: 22,

      precipitation: 80,

    },

    {

      date: "18/09",

      day: "Mar",

      condition: "stormy",

      icon: "thunderstorm",

      tempMin: 15,

      tempMax: 20,

      precipitation: 90,

    },

    {

      date: "19/09",

      day: "Mer",

      condition: "cloudy",

      icon: "cloudy",

      tempMin: 17,

      tempMax: 25,

      precipitation: 40,

    },

  ],

  alerts: [

    {

      id: "1",

      type: "warning",

      title: "Allerta Temporali",

      description:

        "Possibili temporali intensi con grandine nelle zone interne della Calabria. Si raccomanda prudenza negli spostamenti.",

      severity: "medium",

      validFrom: "17/09 14:00",

      validTo: "18/09 06:00",

    },

    {

      id: "2",

      type: "advisory",

      title: "Vento Forte",

      description:

        "Venti forti lungo le coste tirreniche con raffiche fino a 60 km/h.",

      severity: "low",

      validFrom: "16/09 18:00",

      validTo: "17/09 12:00",

    },

  ],

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f5f5f5",

  },

  scrollView: {

    flex: 1,

  },

  content: {

    paddingHorizontal: 16,

    paddingVertical: 20,

  },

  header: {

    alignItems: "center",

    marginBottom: 20,

  },

  title: {

    fontSize: 24,

    fontWeight: "bold",

    color: "#2E7D32",

    marginBottom: 8,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

  },

  loadingContainer: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

  },

  loadingText: {

    marginTop: 16,

    fontSize: 16,

    color: "#666",

  },

  citySelectorCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    marginBottom: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  cityButton: {

    paddingHorizontal: 16,

    paddingVertical: 8,

    marginRight: 8,

    borderRadius: 20,

    backgroundColor: "#f0f0f0",

  },

  selectedCityButton: {

    backgroundColor: "#2E7D32",

  },

  cityButtonText: {

    fontSize: 14,

    color: "#666",

    fontWeight: "500",

  },

  selectedCityButtonText: {

    color: "#fff",

  },

  currentWeatherCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 20,

    marginBottom: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  currentWeatherHeader: {

    alignItems: "center",

    marginBottom: 20,

  },

  locationText: {

    fontSize: 18,

    fontWeight: "600",

    color: "#333",

  },

  dateText: {

    fontSize: 14,

    color: "#666",

    marginTop: 4,

  },

  currentWeatherMain: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    marginBottom: 16,

  },

  temperatureSection: {

    alignItems: "flex-start",

  },

  currentTemp: {

    fontSize: 64,

    fontWeight: "bold",

    color: "#2E7D32",

  },

  condition: {

    fontSize: 18,

    color: "#666",

    marginTop: -8,

  },

  tempRange: {

    alignItems: "center",

    marginBottom: 20,

  },

  tempRangeText: {

    fontSize: 16,

    color: "#666",

  },

  weatherDetails: {

    flexDirection: "row",

    flexWrap: "wrap",

    justifyContent: "space-between",

  },

  detailItem: {

    flexDirection: "row",

    alignItems: "center",

    width: "48%",

    marginBottom: 8,

  },

  detailText: {

    fontSize: 14,

    color: "#666",

    marginLeft: 8,

  },

  forecastCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    marginBottom: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  sectionTitle: {

    fontSize: 18,

    fontWeight: "600",

    color: "#333",

    marginBottom: 16,

  },

  forecastDay: {

    alignItems: "center",

    marginRight: 16,

    paddingVertical: 8,

    minWidth: 70,

  },

  forecastDayName: {

    fontSize: 14,

    fontWeight: "600",

    color: "#333",

  },

  forecastDate: {

    fontSize: 12,

    color: "#666",

    marginBottom: 8,

  },

  forecastTemp: {

    fontSize: 16,

    fontWeight: "600",

    color: "#333",

    marginTop: 8,

  },

  forecastTempMin: {

    fontSize: 14,

    color: "#666",

  },

  forecastPrecip: {

    fontSize: 12,

    color: "#4A90E2",

    marginTop: 4,

  },

  alertsCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    marginBottom: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  alertItem: {

    borderLeftWidth: 4,

    paddingLeft: 12,

    paddingVertical: 12,

    marginBottom: 12,

    backgroundColor: "#f9f9f9",

    borderRadius: 8,

  },

  alertHeader: {

    flexDirection: "row",

    alignItems: "center",

    marginBottom: 8,

  },

  alertTitle: {

    fontSize: 16,

    fontWeight: "600",

    marginLeft: 8,

  },

  alertDescription: {

    fontSize: 14,

    color: "#666",

    marginBottom: 8,

    lineHeight: 20,

  },

  alertValidity: {

    fontSize: 12,

    color: "#999",

    fontStyle: "italic",

  },

});

\*/

# comunication screen

/\*// CommunicationsScreen.tsx

import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  StyleSheet,

  SafeAreaView,

  ScrollView,

  ActivityIndicator,

  RefreshControl,

  TouchableOpacity,

  Alert,

  Linking,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import { apiService } from "../services/api";

import { HeaderFisso } from "../components/HeaderFisso";

interface Communication {

  id: string;

  type: "alert" | "news" | "announcement" | "emergency";

  title: string;

  content: string;

  summary: string;

  priority: "low" | "medium" | "high" | "critical";

  publishedAt: string;

  validUntil?: string;

  author: string;

  category: string;

  attachments?: Attachment[];

  readStatus: boolean;

}

interface Attachment {

  id: string;

  name: string;

  type: "pdf" | "image" | "video" | "link";

  url: string;

  size?: string;

}

export const CommunicationsScreen: React.FC = () => {

  const [communications, setCommunications] = useState<Communication[]>([]);

  const [isLoading, setIsLoading] = useState(true);

  const [refreshing, setRefreshing] = useState(false);

  const [selectedFilter, setSelectedFilter] = useState<string>("all");

  const filterOptions = [

    { id: "all", label: "Tutte", icon: "list" },

    { id: "alert", label: "Allerte", icon: "warning" },

    { id: "news", label: "Notizie", icon: "newspaper" },

    { id: "announcement", label: "Avvisi", icon: "megaphone" },

    { id: "emergency", label: "Emergenze", icon: "alert-circle" },

  ];

  useEffect(() => {

    loadCommunications();

  }, []);

  const loadCommunications = async () => {

    try {

      const response = await apiService.getCommunications();

      setCommunications(response.data);

    } catch (error) {

      console.error("Errore nel caricamento comunicazioni:", error);

      setCommunications(mockCommunications); // mock fallback

    } finally {

      setIsLoading(false);

      setRefreshing(false);

    }

  };

  const onRefresh = () => {

    setRefreshing(true);

    loadCommunications();

  };

  const getPriorityColor = (priority: string): string => {

    switch (priority) {

      case "critical":

        return "#FF0000";

      case "high":

        return "#FF4444";

      case "medium":

        return "#FF8800";

      case "low":

        return "#4CAF50";

      default:

        return "#666";

    }

  };

  const getTypeIcon = (type: string): string => {

    switch (type) {

      case "alert":

        return "warning";

      case "news":

        return "newspaper";

      case "announcement":

        return "megaphone";

      case "emergency":

        return "alert-circle";

      default:

        return "information-circle";

    }

  };

  const getTypeColor = (type: string): string => {

    switch (type) {

      case "alert":

        return "#FF8800";

      case "news":

        return "#2196F3";

      case "announcement":

        return "#9C27B0";

      case "emergency":

        return "#FF0000";

      default:

        return "#666";

    }

  };

  const formatDate = (dateString: string): string => {

    const date = new Date(dateString);

    return date.toLocaleDateString("it-IT", {

      day: "2-digit",

      month: "2-digit",

      year: "numeric",

      hour: "2-digit",

      minute: "2-digit",

    });

  };

  const markAsRead = (communicationId: string) => {

    setCommunications((prev) =>

      prev.map((comm) =>

        comm.id === communicationId ? { ...comm, readStatus: true } : comm

      )

    );

  };

  const openAttachment = async (attachment: Attachment) => {

    try {

      const supported = await Linking.canOpenURL(attachment.url);

      if (supported) {

        await Linking.openURL(attachment.url);

      } else {

        Alert.alert("Errore", "Impossibile aprire il collegamento");

      }

    } catch (error) {

      Alert.alert("Errore", "Errore nell'apertura del collegamento");

    }

  };

  const showCommunicationDetails = (communication: Communication) => {

    markAsRead(communication.id);

    Alert.alert(communication.title, communication.content, [

      {

        text: "Chiudi",

        style: "cancel",

      },

      ...(communication.attachments && communication.attachments.length > 0

        ? [

            {

              text: "Allegati",

              onPress: () => showAttachments(communication.attachments!),

            },

          ]

        : []),

    ]);

  };

  const showAttachments = (attachments: Attachment[]) => {

    const attachmentList = attachments

      .map((att, index) => `${index + 1}. ${att.name}`)

      .join("\n");

    Alert.alert("Allegati", attachmentList, [

      ...attachments.map((att) => ({

        text: att.name,

        onPress: () => {

          void openAttachment(att);

        },

      })),

      { text: "Chiudi", style: "cancel" },

    ]);

  };

  const filteredCommunications = communications.filter((comm) => {

    if (selectedFilter === "all") return true;

    return comm.type === selectedFilter;

  });

  const renderFilterButtons = () => (

    <View style={styles.filterContainer}>

      <ScrollView horizontal showsHorizontalScrollIndicator={false}>

        {filterOptions.map((filter) => (

          <TouchableOpacity

            key={filter.id}

            style={[

              styles.filterButton,

              selectedFilter === filter.id && styles.selectedFilterButton,

            ]}

            onPress={() => setSelectedFilter(filter.id)}

          >

            <Ionicons

              name={filter.icon as any}

              size={16}

              color={selectedFilter === filter.id ? "#fff" : "#2E7D32"}

            />

            <Text

              style={[

                styles.filterButtonText,

                selectedFilter === filter.id && styles.selectedFilterButtonText,

              ]}

            >

              {filter.label}

            </Text>

          </TouchableOpacity>

        ))}

      </ScrollView>

    </View>

  );

  const renderCommunicationCard = (communication: Communication) => (

    <TouchableOpacity

      key={communication.id}

      style={[

        styles.communicationCard,

        !communication.readStatus && styles.unreadCard,

      ]}

      onPress={() => showCommunicationDetails(communication)}

    >

      <View style={styles.cardHeader}>

        <View style={styles.typeContainer}>

          <Ionicons

            name={getTypeIcon(communication.type) as any}

            size={20}

            color={getTypeColor(communication.type)}

          />

          <Text

            style={[

              styles.typeText,

              { color: getTypeColor(communication.type) },

            ]}

          >

            {communication.category}

          </Text>

        </View>

        <View style={styles.priorityContainer}>

          <View

            style={[

              styles.priorityDot,

              { backgroundColor: getPriorityColor(communication.priority) },

            ]}

          />

          {!communication.readStatus && <View style={styles.unreadDot} />}

        </View>

      </View>

      <Text style={styles.cardTitle} numberOfLines={2}>

        {communication.title}

      </Text>

      <Text style={styles.cardSummary} numberOfLines={3}>

        {communication.summary}

      </Text>

      <View style={styles.cardFooter}>

        <View style={styles.authorContainer}>

          <Ionicons name="person" size={14} color="#666" />

          <Text style={styles.authorText}>{communication.author}</Text>

        </View>

        <Text style={styles.dateText}>

          {formatDate(communication.publishedAt)}

        </Text>

      </View>

      {communication.attachments && communication.attachments.length > 0 && (

        <View style={styles.attachmentsIndicator}>

          <Ionicons name="attach" size={14} color="#666" />

          <Text style={styles.attachmentsText}>

            {communication.attachments.length} allegato

            {communication.attachments.length > 1 ? "i" : ""}

          </Text>

        </View>

      )}

      {communication.validUntil && (

        <View style={styles.validityContainer}>

          <Ionicons name="time" size={14} color="#FF8800" />

          <Text style={styles.validityText}>

            Valido fino al {formatDate(communication.validUntil)}

          </Text>

        </View>

      )}

    </TouchableOpacity>

  );

  if (isLoading) {

    return (

      <SafeAreaView style={styles.container}>

        <View style={styles.loadingContainer}>

          <ActivityIndicator size="large" color="#2E7D32" />

          <Text style={styles.loadingText}>Caricamento comunicazioni...</Text>

        </View>

      </SafeAreaView>

    );

  }

  return (

    <SafeAreaView style={styles.container}>

      <View style={styles.header}>

        <Text style={styles.title}>Comunicazioni</Text>

        <Text style={styles.subtitle}>Avvisi e notizie ufficiali</Text>

      </View>

      {renderFilterButtons()}

      <ScrollView

        style={styles.scrollView}

        refreshControl={

          <RefreshControl refreshing={refreshing} onRefresh={onRefresh} />

        }

        showsVerticalScrollIndicator={false}

      >

        <View style={styles.content}>

          {filteredCommunications.length === 0 ? (

            <View style={styles.emptyContainer}>

              <Ionicons name="mail-open" size={64} color="#ccc" />

              <Text style={styles.emptyText}>

                Nessuna comunicazione trovata

              </Text>

            </View>

          ) : (

            filteredCommunications.map(renderCommunicationCard)

          )}

        </View>

      </ScrollView>

    </SafeAreaView>

  );

};

// Dati mock per il testing

const mockCommunications: Communication[] = [

  {

    id: "1",

    type: "emergency",

    title: "Allerta Rossa per Maltempo Estremo",

    content:

      "È stata emessa un'allerta rossa per maltempo estremo su tutta la regione Calabria. Si prevedono piogge torrenziali, venti forti e possibili allagamenti. Si raccomanda di evitare spostamenti non necessari e di prestare massima attenzione.",

    summary:

      "Allerta rossa per maltempo estremo su tutta la Calabria. Evitare spostamenti non necessari.",

    priority: "critical",

    publishedAt: "2025-09-15T08:00:00Z",

    validUntil: "2025-09-16T18:00:00Z",

    author: "Protezione Civile Calabria",

    category: "Emergenza Meteo",

    readStatus: false,

    attachments: [

      {

        id: "1",

        name: "Bollettino Meteo Dettagliato",

        type: "pdf",

        url: "https://example.com/bollettino.pdf",

        size: "2.3 MB",

      },

    ],

  },

  {

    id: "2",

    type: "alert",

    title: "Rischio Incendi Boschivi",

    content:

      "A causa delle alte temperature e dei venti secchi, è stato dichiarato il rischio elevato di incendi boschivi nelle province di Cosenza e Catanzaro. È vietato accendere fuochi all'aperto.",

    summary:

      "Rischio elevato incendi boschivi. Vietato accendere fuochi all'aperto.",

    priority: "high",

    publishedAt: "2025-09-14T14:30:00Z",

    validUntil: "2025-09-17T23:59:00Z",

    author: "Corpo Forestale",

    category: "Prevenzione Incendi",

    readStatus: true,

  },

  {

    id: "3",

    type: "news",

    title: "Nuove Procedure di Evacuazione",

    content:

      "Sono state aggiornate le procedure di evacuazione per le zone a rischio idrogeologico. Tutti i cittadini sono invitati a prendere visione delle nuove modalità operative.",

    summary:

      "Aggiornate le procedure di evacuazione per zone a rischio idrogeologico.",

    priority: "medium",

    publishedAt: "2025-09-13T10:15:00Z",

    author: "Ufficio Comunicazione",

    category: "Aggiornamenti",

    readStatus: false,

    attachments: [

      {

        id: "2",

        name: "Guida Evacuazione",

        type: "pdf",

        url: "https://example.com/evacuazione.pdf",

        size: "1.8 MB",

      },

      {

        id: "3",

        name: "Video Informativo",

        type: "video",

        url: "https://example.com/video.mp4",

        size: "15.2 MB",

      },

    ],

  },

  {

    id: "4",

    type: "announcement",

    title: "Esercitazione di Protezione Civile",

    content:

      "Il 20 settembre si terrà un'esercitazione di protezione civile nella provincia di Reggio Calabria. L'esercitazione simulerà uno scenario di emergenza sismica.",

    summary:

      "Esercitazione protezione civile il 20 settembre a Reggio Calabria.",

    priority: "low",

    publishedAt: "2025-09-12T16:45:00Z",

    author: "Coordinamento Regionale",

    category: "Esercitazioni",

    readStatus: true,

  },

  {

    id: "5",

    type: "news",

    title: "Aggiornamento App Mobile",

    content:

      "È disponibile un nuovo aggiornamento dell'app mobile della Protezione Civile Calabria con nuove funzionalità per la segnalazione di emergenze.",

    summary:

      "Nuovo aggiornamento app con funzionalità migliorate per segnalazioni.",

    priority: "low",

    publishedAt: "2025-09-11T09:20:00Z",

    author: "Team Sviluppo",

    category: "Tecnologia",

    readStatus: false,

  },

];

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f5f5f5",

  },

  header: {

    alignItems: "center",

    paddingVertical: 20,

    paddingHorizontal: 16,

    backgroundColor: "#fff",

    borderBottomWidth: 1,

    borderBottomColor: "#eee",

  },

  title: {

    fontSize: 24,

    fontWeight: "bold",

    color: "#2E7D32",

    marginBottom: 8,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

  },

  loadingContainer: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

  },

  loadingText: {

    marginTop: 16,

    fontSize: 16,

    color: "#666",

  },

  filterContainer: {

    backgroundColor: "#fff",

    paddingVertical: 12,

    paddingHorizontal: 16,

    borderBottomWidth: 1,

    borderBottomColor: "#eee",

  },

  filterButton: {

    flexDirection: "row",

    alignItems: "center",

    paddingHorizontal: 12,

    paddingVertical: 6,

    marginRight: 8,

    borderRadius: 16,

    backgroundColor: "#f0f0f0",

    gap: 4,

  },

  selectedFilterButton: {

    backgroundColor: "#2E7D32",

  },

  filterButtonText: {

    fontSize: 14,

    color: "#2E7D32",

    fontWeight: "500",

  },

  selectedFilterButtonText: {

    color: "#fff",

  },

  scrollView: {

    flex: 1,

  },

  content: {

    paddingHorizontal: 16,

    paddingVertical: 16,

  },

  emptyContainer: {

    alignItems: "center",

    justifyContent: "center",

    paddingVertical: 60,

  },

  emptyText: {

    fontSize: 16,

    color: "#999",

    marginTop: 16,

  },

  communicationCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    marginBottom: 12,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  unreadCard: {

    borderLeftWidth: 4,

    borderLeftColor: "#2E7D32",

  },

  cardHeader: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    marginBottom: 12,

  },

  typeContainer: {

    flexDirection: "row",

    alignItems: "center",

    gap: 6,

  },

  typeText: {

    fontSize: 14,

    fontWeight: "600",

  },

  priorityContainer: {

    flexDirection: "row",

    alignItems: "center",

    gap: 8,

  },

  priorityDot: {

    width: 8,

    height: 8,

    borderRadius: 4,

  },

  unreadDot: {

    width: 10,

    height: 10,

    borderRadius: 5,

    backgroundColor: "#2E7D32",

  },

  cardTitle: {

    fontSize: 16,

    fontWeight: "600",

    color: "#333",

    marginBottom: 8,

    lineHeight: 22,

  },

  cardSummary: {

    fontSize: 14,

    color: "#666",

    lineHeight: 20,

    marginBottom: 12,

  },

  cardFooter: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

  },

  authorContainer: {

    flexDirection: "row",

    alignItems: "center",

    gap: 4,

  },

  authorText: {

    fontSize: 12,

    color: "#666",

  },

  dateText: {

    fontSize: 12,

    color: "#999",

  },

  attachmentsIndicator: {

    flexDirection: "row",

    alignItems: "center",

    gap: 4,

    marginTop: 8,

    paddingTop: 8,

    borderTopWidth: 1,

    borderTopColor: "#f0f0f0",

  },

  attachmentsText: {

    fontSize: 12,

    color: "#666",

  },

  validityContainer: {

    flexDirection: "row",

    alignItems: "center",

    gap: 4,

    marginTop: 8,

    paddingHorizontal: 8,

    paddingVertical: 4,

    backgroundColor: "#FFF3E0",

    borderRadius: 4,

  },

  validityText: {

    fontSize: 12,

    color: "#FF8800",

    fontWeight: "500",

  },

});

\*/

# //event list screen

import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  FlatList,

  TouchableOpacity,

  StyleSheet,

  RefreshControl,

  Alert,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import { eventsService } from "../services/api";

export default function EventsListScreen({ navigation }) {

  const [events, setEvents] = useState([]);

  const [loading, setLoading] = useState(false);

  const [refreshing, setRefreshing] = useState(false);

  const [showMyReports, setShowMyReports] = useState(false);

  useEffect(() => {

    loadEvents();

  }, [showMyReports]);

  const loadEvents = async () => {

    setLoading(true);

    try {

      // Simula dati eventi per la demo

      const mockEvents = [

        {

          id: "1",

          title: "Allagamento Strada Provinciale",

          description:

            "Segnalazione di allagamento sulla SP 123 all'altezza del km 15. La strada risulta impraticabile per i veicoli.",

          type: "flood",

          status: "open",

          location: "Cosenza, SP 123",

          createdAt: "2024-09-17T08:30:00Z",

          priority: "high",

          hasInjured: false,

          hasVictims: false,

        },

        {

          id: "2",

          title: "Frana Località Monte Alto",

          description:

            "Frana che blocca completamente la viabilità sulla strada comunale. Necessario intervento urgente.",

          type: "landslide",

          status: "open",

          location: "Catanzaro, Monte Alto",

          createdAt: "2024-09-17T07:15:00Z",

          priority: "critical",

          hasInjured: true,

          hasVictims: false,

        },

        {

          id: "3",

          title: "Incendio Boschivo",

          description:

            "Principio di incendio in zona boschiva. Fumo visibile dalla strada statale.",

          type: "fire",

          status: "open",

          location: "Reggio Calabria, Aspromonte",

          createdAt: "2024-09-17T06:45:00Z",

          priority: "high",

          hasInjured: false,

          hasVictims: false,

        },

        {

          id: "4",

          title: "Albero Caduto",

          description:

            "Albero caduto sulla carreggiata a causa del vento forte.",

          type: "obstacle",

          status: "resolved",

          location: "Crotone, Via Roma",

          createdAt: "2024-09-16T18:20:00Z",

          priority: "medium",

          hasInjured: false,

          hasVictims: false,

        },

      ];

      // Filtra per le proprie segnalazioni se necessario

      const filteredEvents = showMyReports

        ? mockEvents.filter((event) => event.id === "1") // Simula che solo l'evento 1 sia dell'utente

        : mockEvents;

      setEvents(filteredEvents);

    } catch (error) {

      console.error("Errore caricamento eventi:", error);

      Alert.alert("Errore", "Impossibile caricare gli eventi");

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  const onRefresh = () => {

    setRefreshing(true);

    loadEvents();

  };

  const getEventIcon = (eventType) => {

    switch (eventType) {

      case "flood":

        return "water";

      case "landslide":

        return "triangle";

      case "fire":

        return "flame";

      case "obstacle":

        return "warning";

      default:

        return "alert-circle";

    }

  };

  const getEventColor = (eventType) => {

    switch (eventType) {

      case "flood":

        return "#007AFF";

      case "landslide":

        return "#FF9500";

      case "fire":

        return "#FF3B30";

      case "obstacle":

        return "#8E8E93";

      default:

        return "#FF6B35";

    }

  };

  const getPriorityColor = (priority) => {

    switch (priority) {

      case "critical":

        return "#FF3B30";

      case "high":

        return "#FF9500";

      case "medium":

        return "#FFCC00";

      case "low":

        return "#34C759";

      default:

        return "#8E8E93";

    }

  };

  const getStatusText = (status) => {

    switch (status) {

      case "open":

        return "Aperto";

      case "in\_progress":

        return "In corso";

      case "resolved":

        return "Risolto";

      default:

        return "Sconosciuto";

    }

  };

  const formatDate = (dateString) => {

    const date = new Date(dateString);

    return date.toLocaleDateString("it-IT", {

      day: "2-digit",

      month: "2-digit",

      year: "numeric",

      hour: "2-digit",

      minute: "2-digit",

    });

  };

  const handleEventPress = (event) => {

    Alert.alert(

      event.title,

      `${event.description}\n\nLocalità: ${event.location}\nData: ${formatDate(

        event.createdAt

      )}`,

      [

        { text: "Chiudi", style: "cancel" },

        { text: "Dettagli", onPress: () => showEventDetails(event) },

      ]

    );

  };

  const showEventDetails = (event) => {

    // Naviga ai dettagli dell'evento

    console.log("Mostra dettagli evento:", event);

  };

  const toggleFilter = () => {

    setShowMyReports(!showMyReports);

  };

  const renderEventItem = ({ item }) => (

    <TouchableOpacity

      style={styles.eventCard}

      onPress={() => handleEventPress(item)}

    >

      <View style={styles.eventHeader}>

        <View style={styles.eventIconContainer}>

          <Ionicons

            name={getEventIcon(item.type)}

            size={24}

            color={getEventColor(item.type)}

          />

        </View>

        <View style={styles.eventInfo}>

          <Text style={styles.eventTitle}>{item.title}</Text>

          <Text style={styles.eventLocation}>{item.location}</Text>

        </View>

        <View style={styles.eventStatus}>

          <View

            style={[

              styles.priorityBadge,

              { backgroundColor: getPriorityColor(item.priority) },

            ]}

          >

            <Text style={styles.priorityText}>

              {item.priority.toUpperCase()}

            </Text>

          </View>

        </View>

      </View>

      <Text style={styles.eventDescription} numberOfLines={2}>

        {item.description}

      </Text>

      <View style={styles.eventFooter}>

        <Text style={styles.eventDate}>{formatDate(item.createdAt)}</Text>

        <View style={styles.eventBadges}>

          {item.hasInjured && (

            <View style={styles.warningBadge}>

              <Ionicons name="medical" size={12} color="#fff" />

              <Text style={styles.badgeText}>Feriti</Text>

            </View>

          )}

          {item.hasVictims && (

            <View style={styles.criticalBadge}>

              <Ionicons name="skull" size={12} color="#fff" />

              <Text style={styles.badgeText}>Vittime</Text>

            </View>

          )}

          <View

            style={[

              styles.statusBadge,

              {

                backgroundColor:

                  item.status === "resolved" ? "#34C759" : "#FF6B35",

              },

            ]}

          >

            <Text style={styles.badgeText}>{getStatusText(item.status)}</Text>

          </View>

        </View>

      </View>

    </TouchableOpacity>

  );

  return (

    <View style={styles.container}>

      {/\* Header \*/}

      <View style={styles.header}>

        <TouchableOpacity

          style={styles.backButton}

          onPress={() => navigation.goBack()}

        >

          <Ionicons name="arrow-back" size={24} color="#fff" />

        </TouchableOpacity>

        <TouchableOpacity style={styles.filterButton} onPress={toggleFilter}>

          <Ionicons

            name={showMyReports ? "person" : "people"}

            size={20}

            color="#fff"

          />

          <Text style={styles.filterText}>

            {showMyReports ? "Le mie segnalazioni" : "Eventi"}

          </Text>

        </TouchableOpacity>

        <TouchableOpacity

          style={styles.mapButton}

          onPress={() => navigation.goBack()}

        >

          <Ionicons name="map" size={20} color="#fff" />

          <Text style={styles.mapButtonText}>Mappa</Text>

        </TouchableOpacity>

      </View>

      {/\* Events List \*/}

      <FlatList

        data={events}

        renderItem={renderEventItem}

        keyExtractor={(item) => item.id}

        contentContainerStyle={styles.listContainer}

        refreshControl={

          <RefreshControl

            refreshing={refreshing}

            onRefresh={onRefresh}

            colors={["#FF6B35"]}

          />

        }

        ListEmptyComponent={

          <View style={styles.emptyContainer}>

            <Ionicons name="document-text-outline" size={60} color="#ccc" />

            <Text style={styles.emptyText}>

              {showMyReports

                ? "Nessuna segnalazione trovata"

                : "Nessun evento disponibile"}

            </Text>

          </View>

        }

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f8f9fa",

  },

  header: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    backgroundColor: "#FF6B35",

    paddingHorizontal: 15,

    paddingVertical: 10,

    paddingTop: 50,

  },

  backButton: {

    padding: 5,

  },

  filterButton: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "rgba(255, 255, 255, 0.2)",

    paddingHorizontal: 12,

    paddingVertical: 8,

    borderRadius: 20,

  },

  filterText: {

    color: "#fff",

    marginLeft: 5,

    fontSize: 14,

    fontWeight: "600",

  },

  mapButton: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "rgba(255, 255, 255, 0.2)",

    paddingHorizontal: 12,

    paddingVertical: 8,

    borderRadius: 20,

  },

  mapButtonText: {

    color: "#fff",

    marginLeft: 5,

    fontSize: 14,

    fontWeight: "600",

  },

  listContainer: {

    padding: 15,

  },

  eventCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 15,

    marginBottom: 15,

    elevation: 3,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 3.84,

  },

  eventHeader: {

    flexDirection: "row",

    alignItems: "flex-start",

    marginBottom: 10,

  },

  eventIconContainer: {

    width: 40,

    height: 40,

    borderRadius: 20,

    backgroundColor: "#f8f9fa",

    justifyContent: "center",

    alignItems: "center",

    marginRight: 12,

  },

  eventInfo: {

    flex: 1,

  },

  eventTitle: {

    fontSize: 16,

    fontWeight: "bold",

    color: "#333",

    marginBottom: 4,

  },

  eventLocation: {

    fontSize: 14,

    color: "#666",

  },

  eventStatus: {

    alignItems: "flex-end",

  },

  priorityBadge: {

    paddingHorizontal: 8,

    paddingVertical: 4,

    borderRadius: 12,

  },

  priorityText: {

    fontSize: 10,

    fontWeight: "bold",

    color: "#fff",

  },

  eventDescription: {

    fontSize: 14,

    color: "#666",

    lineHeight: 20,

    marginBottom: 12,

  },

  eventFooter: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

  },

  eventDate: {

    fontSize: 12,

    color: "#999",

  },

  eventBadges: {

    flexDirection: "row",

    gap: 6,

  },

  warningBadge: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "#FF9500",

    paddingHorizontal: 6,

    paddingVertical: 2,

    borderRadius: 10,

    gap: 2,

  },

  criticalBadge: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "#FF3B30",

    paddingHorizontal: 6,

    paddingVertical: 2,

    borderRadius: 10,

    gap: 2,

  },

  statusBadge: {

    paddingHorizontal: 6,

    paddingVertical: 2,

    borderRadius: 10,

  },

  badgeText: {

    fontSize: 10,

    fontWeight: "600",

    color: "#fff",

  },

  emptyContainer: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

    paddingVertical: 60,

  },

  emptyText: {

    fontSize: 16,

    color: "#999",

    marginTop: 15,

    textAlign: "center",

  },

});

# event screen

/\*// EventScreen.tsx

import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  StyleSheet,

  SafeAreaView,

  TouchableOpacity,

  FlatList,

  ActivityIndicator,

} from "react-native";

import MapView, { Marker } from "react-native-maps";

import { Ionicons } from "@expo/vector-icons";

import { Event, EventsSearchRequest } from "../types";

import { apiService } from "../services/api";

import { HeaderFisso } from "../components/HeaderFisso";

import { NativeStackNavigationProp } from "@react-navigation/native-stack";

import { RootStackParamList } from "../navigation/AppNavigator"; // o "types" se lo hai spostato lì

type EventScreenNavigationProp = NativeStackNavigationProp<

  RootStackParamList,

  "Event"

>;

type Props = {

  navigation: EventScreenNavigationProp;

};

const EventScreen: React.FC<Props> = ({ navigation }) => {

  const [activeTab, setActiveTab] = useState<"map" | "list">("map");

  const [events, setEvents] = useState<Event[]>([]);

  const [isLoading, setIsLoading] = useState(false);

  const [filter, setFilter] = useState<"all" | "events" | "reports">("all");

  useEffect(() => {

    loadEvents();

  }, [filter]);

  const loadEvents = async () => {

    setIsLoading(true);

    try {

      const searchRequest: EventsSearchRequest = {

        selector: { Type: "Event", event: { status: "Event:Status:Open" } },

        limit: 40,

        sort: [{ "lifespan.start": "desc" }],

        fields: [

          "SubType",

          "createdAt",

          "updatedAt",

          "lifespan.start",

          "event",

          "register.serial",

          "locations",

          "engagements",

          "requests",

          "operations",

          "officialReport",

          "isPublic",

        ],

        use\_index: ["IdxEventList", "eventList"],

      };

      const response = await apiService.searchEvents(searchRequest);

      setEvents(response.data.docs || []);

    } catch (error) {

      console.error("Errore nel caricamento eventi:", error);

      setEvents(mockEvents);

    } finally {

      setIsLoading(false);

    }

  };

  const filteredEvents = events.filter((event) => {

    if (filter === "all") return true;

    if (filter === "events")

      return ["Alluvione", "Incendio", "Frana"].includes(event.SubType);

    if (filter === "reports") return event.SubType === "Report";

    return true;

  });

  const renderEventItem = ({ item }: { item: Event }) => (

    <TouchableOpacity style={styles.eventItem}>

      <Text style={styles.eventTitle}>{item.SubType || "Evento"}</Text>

      <Text style={styles.eventDate}>

        {item.lifespan?.start

          ? new Date(item.lifespan.start).toLocaleDateString("it-IT")

          : "Data non disponibile"}

      </Text>

      <Text style={styles.eventDescription}>

        Seriale: {item.register?.serial || "N/A"}

      </Text>

      <Text style={styles.eventStatus}>

        Stato: {item.event.status.replace("Event:Status:", "")}

      </Text>

    </TouchableOpacity>

  );

  return (

    <View style={{ flex: 1 }}>

      <HeaderFisso />

      <SafeAreaView style={{ flex: 1 }}>

        <View style={styles.tabContainer}>

          <TouchableOpacity

            style={[styles.tab, activeTab === "map" && styles.activeTab]}

            onPress={() => setActiveTab("map")}

          >

            <Ionicons

              name={activeTab === "map" ? "map" : "map-outline"}

              size={20}

              color={activeTab === "map" ? "#2E7D32" : "#666"}

            />

            <Text

              style={[

                styles.tabText,

                activeTab === "map" && styles.activeTabText,

              ]}

            >

              Mappa

            </Text>

          </TouchableOpacity>

          <TouchableOpacity

            style={[styles.tab, activeTab === "list" && styles.activeTab]}

            onPress={() => setActiveTab("list")}

          >

            <Ionicons

              name={activeTab === "list" ? "list" : "list-outline"}

              size={20}

              color={activeTab === "list" ? "#2E7D32" : "#666"}

            />

            <Text

              style={[

                styles.tabText,

                activeTab === "list" && styles.activeTabText,

              ]}

            >

              Lista

            </Text>

          </TouchableOpacity>

        </View>

        {activeTab === "map" ? (

          <MapView

            style={{ flex: 1 }}

            initialRegion={{

              latitude: 39.0,

              longitude: 16.5,

              latitudeDelta: 2.0,

              longitudeDelta: 2.0,

            }}

          >

            {filteredEvents.map((event, index) =>

              event.locations && event.locations.length > 0 ? (

                <Marker

                  key={event.\_id || index}

                  coordinate={{

                    latitude: event.locations[0].latitude || 0,

                    longitude: event.locations[0].longitude || 0,

                  }}

                  title={event.SubType || "Evento"}

                  description={event.register?.serial || "N/A"}

                />

              ) : null

            )}

          </MapView>

        ) : isLoading ? (

          <View style={styles.loadingContainer}>

            <ActivityIndicator size="large" color="#2E7D32" />

            <Text>Caricamento eventi...</Text>

          </View>

        ) : (

          <FlatList

            data={filteredEvents}

            renderItem={renderEventItem}

            keyExtractor={(item, index) =>

              item.\_id ?? `${item.SubType}-${index}`

            }

            contentContainerStyle={{ padding: 16 }}

            ListEmptyComponent={

              <View style={styles.emptyContainer}>

                <Text>Nessun evento disponibile</Text>

              </View>

            }

          />

        )}

      </SafeAreaView>

    </View>

  );

};

// ✅ Esportazione corretta per import default

export default EventScreen;

// 👇 Mock dati (in caso di fallback)

const mockEvents: Event[] = [

  {

    \_id: "1",

    SubType: "Alluvione",

    createdAt: "",

    updatedAt: "",

    lifespan: { start: "2024-09-15T10:00:00Z" },

    event: { status: "Event:Status:Open" },

    register: { serial: "ALL-2024-001" },

    locations: [{ latitude: 39.3, longitude: 16.2, address: "Cosenza, CS" }],

    engagements: [],

    requests: [],

    operations: [],

    officialReport: null,

    isPublic: true,

  },

];

const styles = StyleSheet.create({

  tabContainer: {

    flexDirection: "row",

    borderBottomWidth: 1,

    borderBottomColor: "#ddd",

  },

  tab: {

    flex: 1,

    flexDirection: "row",

    justifyContent: "center",

    alignItems: "center",

    paddingVertical: 10,

  },

  activeTab: {

    borderBottomWidth: 2,

    borderBottomColor: "#2E7D32",

  },

  tabText: {

    marginLeft: 8,

    color: "#666",

  },

  activeTabText: {

    color: "#2E7D32",

    fontWeight: "bold",

  },

  eventItem: {

    backgroundColor: "#f5f5f5",

    padding: 12,

    borderRadius: 6,

    marginBottom: 12,

  },

  eventTitle: {

    fontWeight: "bold",

    fontSize: 16,

    color: "#2E7D32",

  },

  eventDate: {

    fontSize: 12,

    color: "#666",

  },

  eventDescription: {

    fontSize: 14,

    marginBottom: 4,

  },

  eventStatus: {

    fontSize: 12,

    color: "#888",

  },

  loadingContainer: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

  },

  emptyContainer: {

    alignItems: "center",

    marginTop: 32,

  },

});

\*/

# profile screen

/\*// ProfileScreen.tsx

import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  StyleSheet,

  SafeAreaView,

  ScrollView,

  TouchableOpacity,

  Alert,

  Linking,

  Switch,

  TextInput,

  ActivityIndicator,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import { useAuth } from "../utils/AuthContext";

import { apiService } from "../services/api";

import { APP\_CONFIG, API\_CONFIG } from "../constants/config";

interface UserSettings {

  notifications: boolean;

  locationTracking: boolean;

  emergencyAlerts: boolean;

  weatherAlerts: boolean;

}

interface AppInfo {

  version: string;

  buildNumber: string;

  lastUpdate: string;

}

export const ProfileScreen: React.FC = () => {

  const { user, logout } = useAuth();

  const [isEditing, setIsEditing] = useState(false);

  const [isLoading, setIsLoading] = useState(false);

  const [settings, setSettings] = useState<UserSettings>({

    notifications: true,

    locationTracking: true,

    emergencyAlerts: true,

    weatherAlerts: true,

  });

  const [editedUser, setEditedUser] = useState({

    name: user?.name || "",

    surname: user?.surname || "",

    email: user?.email || "",

    phoneNumber: user?.phoneNumber || "",

  });

  const appInfo: AppInfo = {

    version: APP\_CONFIG.VERSION,

    buildNumber: "1",

    lastUpdate: new Date().toLocaleDateString("it-IT"),

  };

  useEffect(() => {

    loadUserSettings();

  }, []);

  const loadUserSettings = async () => {

    try {

      // In un'app reale, questi dati verrebbero caricati dal server

      // Per ora usiamo valori di default

      setSettings({

        notifications: true,

        locationTracking: true,

        emergencyAlerts: true,

        weatherAlerts: true,

      });

    } catch (error) {

      console.error("Errore nel caricamento impostazioni:", error);

    }

  };

  const handleSaveProfile = async () => {

    if (!user) return;

    setIsLoading(true);

    try {

      await apiService.updateUserProfile(user.fiscalCode, editedUser);

      Alert.alert("Successo", "Profilo aggiornato con successo");

      setIsEditing(false);

    } catch (error) {

      console.error("Errore nell'aggiornamento profilo:", error);

      Alert.alert(

        "Errore",

        "Impossibile aggiornare il profilo. Riprova più tardi."

      );

    } finally {

      setIsLoading(false);

    }

  };

  const handleSettingChange = (setting: keyof UserSettings, value: boolean) => {

    setSettings((prev) => ({ ...prev, [setting]: value }));

    // In un'app reale, qui si salverebbero le impostazioni sul server

  };

  const handleLogout = () => {

    Alert.alert("Conferma Logout", "Sei sicuro di voler uscire dall'app?", [

      { text: "Annulla", style: "cancel" },

      {

        text: "Esci",

        style: "destructive",

        onPress: () => {

          logout();

        },

      },

    ]);

  };

  const openExternalLink = async (url: string, title: string) => {

    try {

      const supported = await Linking.canOpenURL(url);

      if (supported) {

        await Linking.openURL(url);

      } else {

        Alert.alert("Errore", `Impossibile aprire ${title}`);

      }

    } catch (error) {

      Alert.alert("Errore", `Errore nell'apertura di ${title}`);

    }

  };

  const renderUserInfo = () => (

    <View style={styles.section}>

      <View style={styles.sectionHeader}>

        <Text style={styles.sectionTitle}>Informazioni Personali</Text>

        <TouchableOpacity

          style={styles.editButton}

          onPress={() => setIsEditing(!isEditing)}

        >

          <Ionicons

            name={isEditing ? "close" : "pencil"}

            size={20}

            color="#2E7D32"

          />

        </TouchableOpacity>

      </View>

      <View style={styles.userCard}>

        <View style={styles.avatarContainer}>

          <View style={styles.avatar}>

            <Ionicons name="person" size={40} color="#fff" />

          </View>

          <Text style={styles.userName}>

            {user?.name} {user?.surname}

          </Text>

          <Text style={styles.userRole}>Cittadino</Text>

        </View>

        <View style={styles.userDetails}>

          <View style={styles.detailRow}>

            <Text style={styles.detailLabel}>Codice Fiscale</Text>

            <Text style={styles.detailValue}>{user?.fiscalCode}</Text>

          </View>

          <View style={styles.detailRow}>

            <Text style={styles.detailLabel}>Nome</Text>

            {isEditing ? (

              <TextInput

                style={styles.editInput}

                value={editedUser.name}

                onChangeText={(text) =>

                  setEditedUser((prev) => ({ ...prev, name: text }))

                }

                placeholder="Nome"

              />

            ) : (

              <Text style={styles.detailValue}>{user?.name}</Text>

            )}

          </View>

          <View style={styles.detailRow}>

            <Text style={styles.detailLabel}>Cognome</Text>

            {isEditing ? (

              <TextInput

                style={styles.editInput}

                value={editedUser.surname}

                onChangeText={(text) =>

                  setEditedUser((prev) => ({ ...prev, surname: text }))

                }

                placeholder="Cognome"

              />

            ) : (

              <Text style={styles.detailValue}>{user?.surname}</Text>

            )}

          </View>

          <View style={styles.detailRow}>

            <Text style={styles.detailLabel}>Email</Text>

            {isEditing ? (

              <TextInput

                style={styles.editInput}

                value={editedUser.email}

                onChangeText={(text) =>

                  setEditedUser((prev) => ({ ...prev, email: text }))

                }

                placeholder="Email"

                keyboardType="email-address"

                autoCapitalize="none"

              />

            ) : (

              <Text style={styles.detailValue}>{user?.email}</Text>

            )}

          </View>

          <View style={styles.detailRow}>

            <Text style={styles.detailLabel}>Telefono</Text>

            {isEditing ? (

              <TextInput

                style={styles.editInput}

                value={editedUser.phoneNumber}

                onChangeText={(text) =>

                  setEditedUser((prev) => ({ ...prev, phoneNumber: text }))

                }

                placeholder="Numero di telefono"

                keyboardType="phone-pad"

              />

            ) : (

              <Text style={styles.detailValue}>{user?.phoneNumber}</Text>

            )}

          </View>

        </View>

        {isEditing && (

          <TouchableOpacity

            style={styles.saveButton}

            onPress={handleSaveProfile}

            disabled={isLoading}

          >

            {isLoading ? (

              <ActivityIndicator color="#fff" />

            ) : (

              <>

                <Ionicons name="checkmark" size={20} color="#fff" />

                <Text style={styles.saveButtonText}>Salva Modifiche</Text>

              </>

            )}

          </TouchableOpacity>

        )}

      </View>

    </View>

  );

  const renderSettings = () => (

    <View style={styles.section}>

      <Text style={styles.sectionTitle}>Impostazioni</Text>

      <View style={styles.settingsCard}>

        <View style={styles.settingRow}>

          <View style={styles.settingInfo}>

            <Ionicons name="notifications" size={20} color="#666" />

            <View style={styles.settingText}>

              <Text style={styles.settingTitle}>Notifiche Push</Text>

              <Text style={styles.settingDescription}>

                Ricevi notifiche per aggiornamenti

              </Text>

            </View>

          </View>

          <Switch

            value={settings.notifications}

            onValueChange={(value) =>

              handleSettingChange("notifications", value)

            }

            trackColor={{ false: "#ccc", true: "#2E7D32" }}

            thumbColor={settings.notifications ? "#2E7D32" : "#f4f3f4"}

          />

        </View>

        <View style={styles.settingRow}>

          <View style={styles.settingInfo}>

            <Ionicons name="location" size={20} color="#666" />

            <View style={styles.settingText}>

              <Text style={styles.settingTitle}>Geolocalizzazione</Text>

              <Text style={styles.settingDescription}>

                Permetti accesso alla posizione

              </Text>

            </View>

          </View>

          <Switch

            value={settings.locationTracking}

            onValueChange={(value) =>

              handleSettingChange("locationTracking", value)

            }

            trackColor={{ false: "#ccc", true: "#2E7D32" }}

            thumbColor={settings.locationTracking ? "#2E7D32" : "#f4f3f4"}

          />

        </View>

        <View style={styles.settingRow}>

          <View style={styles.settingInfo}>

            <Ionicons name="alert-circle" size={20} color="#666" />

            <View style={styles.settingText}>

              <Text style={styles.settingTitle}>Allerte Emergenza</Text>

              <Text style={styles.settingDescription}>

                Notifiche per situazioni di emergenza

              </Text>

            </View>

          </View>

          <Switch

            value={settings.emergencyAlerts}

            onValueChange={(value) =>

              handleSettingChange("emergencyAlerts", value)

            }

            trackColor={{ false: "#ccc", true: "#2E7D32" }}

            thumbColor={settings.emergencyAlerts ? "#2E7D32" : "#f4f3f4"}

          />

        </View>

        <View style={styles.settingRow}>

          <View style={styles.settingInfo}>

            <Ionicons name="cloudy" size={20} color="#666" />

            <View style={styles.settingText}>

              <Text style={styles.settingTitle}>Allerte Meteo</Text>

              <Text style={styles.settingDescription}>

                Notifiche per condizioni meteorologiche

              </Text>

            </View>

          </View>

          <Switch

            value={settings.weatherAlerts}

            onValueChange={(value) =>

              handleSettingChange("weatherAlerts", value)

            }

            trackColor={{ false: "#ccc", true: "#2E7D32" }}

            thumbColor={settings.weatherAlerts ? "#2E7D32" : "#f4f3f4"}

          />

        </View>

      </View>

    </View>

  );

  const renderExternalLinks = () => (

    <View style={styles.section}>

      <Text style={styles.sectionTitle}>Link Utili</Text>

      <View style={styles.linksCard}>

        <TouchableOpacity

          style={styles.linkRow}

          onPress={() =>

            openExternalLink(

              API\_CONFIG.EXTERNAL\_URLS.REGIONE\_CALABRIA,

              "Regione Calabria"

            )

          }

        >

          <Ionicons name="globe" size={20} color="#2E7D32" />

          <Text style={styles.linkText}>Sito Regione Calabria</Text>

          <Ionicons name="chevron-forward" size={20} color="#ccc" />

        </TouchableOpacity>

        <TouchableOpacity

          style={styles.linkRow}

          onPress={() =>

            openExternalLink(

              API\_CONFIG.EXTERNAL\_URLS.BOLLETTINO\_METEO,

              "Bollettino Meteo"

            )

          }

        >

          <Ionicons name="cloudy" size={20} color="#2E7D32" />

          <Text style={styles.linkText}>Bollettino Meteo Ufficiale</Text>

          <Ionicons name="chevron-forward" size={20} color="#ccc" />

        </TouchableOpacity>

        <TouchableOpacity

          style={styles.linkRow}

          onPress={() => openExternalLink("tel:115", "Numero Emergenza")}

        >

          <Ionicons name="call" size={20} color="#FF4444" />

          <Text style={styles.linkText}>Chiama 115 - Vigili del Fuoco</Text>

          <Ionicons name="chevron-forward" size={20} color="#ccc" />

        </TouchableOpacity>

        <TouchableOpacity

          style={styles.linkRow}

          onPress={() => openExternalLink("tel:112", "Numero Emergenza")}

        >

          <Ionicons name="call" size={20} color="#FF4444" />

          <Text style={styles.linkText}>

            Chiama 112 - Numero Unico Emergenze

          </Text>

          <Ionicons name="chevron-forward" size={20} color="#ccc" />

        </TouchableOpacity>

      </View>

    </View>

  );

  const renderAppInfo = () => (

    <View style={styles.section}>

      <Text style={styles.sectionTitle}>Informazioni App</Text>

      <View style={styles.appInfoCard}>

        <View style={styles.appInfoRow}>

          <Text style={styles.appInfoLabel}>Nome App</Text>

          <Text style={styles.appInfoValue}>{APP\_CONFIG.APP\_NAME}</Text>

        </View>

        <View style={styles.appInfoRow}>

          <Text style={styles.appInfoLabel}>Versione</Text>

          <Text style={styles.appInfoValue}>{appInfo.version}</Text>

        </View>

        <View style={styles.appInfoRow}>

          <Text style={styles.appInfoLabel}>Build</Text>

          <Text style={styles.appInfoValue}>{appInfo.buildNumber}</Text>

        </View>

        <View style={styles.appInfoRow}>

          <Text style={styles.appInfoLabel}>Ultimo Aggiornamento</Text>

          <Text style={styles.appInfoValue}>{appInfo.lastUpdate}</Text>

        </View>

      </View>

    </View>

  );

  return (

    <SafeAreaView style={styles.container}>

      <ScrollView

        style={styles.scrollView}

        showsVerticalScrollIndicator={false}

      >

        <View style={styles.content}>

          <View style={styles.header}>

            <Text style={styles.title}>Profilo</Text>

            <Text style={styles.subtitle}>

              Gestisci il tuo account e le impostazioni

            </Text>

          </View>

          {renderUserInfo()}

          {renderSettings()}

          {renderExternalLinks()}

          {renderAppInfo()}

          <TouchableOpacity style={styles.logoutButton} onPress={handleLogout}>

            <Ionicons name="log-out" size={20} color="#FF4444" />

            <Text style={styles.logoutButtonText}>Esci dall'App</Text>

          </TouchableOpacity>

        </View>

      </ScrollView>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f5f5f5",

  },

  scrollView: {

    flex: 1,

  },

  content: {

    paddingHorizontal: 16,

    paddingVertical: 20,

  },

  header: {

    alignItems: "center",

    marginBottom: 30,

  },

  title: {

    fontSize: 24,

    fontWeight: "bold",

    color: "#2E7D32",

    marginBottom: 8,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

  },

  section: {

    marginBottom: 24,

  },

  sectionHeader: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    marginBottom: 16,

  },

  sectionTitle: {

    fontSize: 18,

    fontWeight: "600",

    color: "#333",

  },

  editButton: {

    padding: 8,

  },

  userCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 20,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  avatarContainer: {

    alignItems: "center",

    marginBottom: 20,

  },

  avatar: {

    width: 80,

    height: 80,

    borderRadius: 40,

    backgroundColor: "#2E7D32",

    justifyContent: "center",

    alignItems: "center",

    marginBottom: 12,

  },

  userName: {

    fontSize: 20,

    fontWeight: "bold",

    color: "#333",

    marginBottom: 4,

  },

  userRole: {

    fontSize: 14,

    color: "#666",

  },

  userDetails: {

    gap: 16,

  },

  detailRow: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

  },

  detailLabel: {

    fontSize: 14,

    color: "#666",

    fontWeight: "500",

    flex: 1,

  },

  detailValue: {

    fontSize: 14,

    color: "#333",

    flex: 2,

    textAlign: "right",

  },

  editInput: {

    flex: 2,

    borderWidth: 1,

    borderColor: "#ddd",

    borderRadius: 6,

    paddingHorizontal: 12,

    paddingVertical: 8,

    fontSize: 14,

    textAlign: "right",

  },

  saveButton: {

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    backgroundColor: "#2E7D32",

    paddingVertical: 12,

    borderRadius: 8,

    marginTop: 20,

    gap: 8,

  },

  saveButtonText: {

    color: "#fff",

    fontSize: 16,

    fontWeight: "600",

  },

  settingsCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  settingRow: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    paddingVertical: 12,

    borderBottomWidth: 1,

    borderBottomColor: "#f0f0f0",

  },

  settingInfo: {

    flexDirection: "row",

    alignItems: "center",

    flex: 1,

    gap: 12,

  },

  settingText: {

    flex: 1,

  },

  settingTitle: {

    fontSize: 16,

    color: "#333",

    fontWeight: "500",

  },

  settingDescription: {

    fontSize: 12,

    color: "#666",

    marginTop: 2,

  },

  linksCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  linkRow: {

    flexDirection: "row",

    alignItems: "center",

    paddingVertical: 12,

    borderBottomWidth: 1,

    borderBottomColor: "#f0f0f0",

    gap: 12,

  },

  linkText: {

    fontSize: 16,

    color: "#333",

    flex: 1,

  },

  appInfoCard: {

    backgroundColor: "#fff",

    borderRadius: 12,

    padding: 16,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  appInfoRow: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    paddingVertical: 8,

  },

  appInfoLabel: {

    fontSize: 14,

    color: "#666",

    fontWeight: "500",

  },

  appInfoValue: {

    fontSize: 14,

    color: "#333",

  },

  logoutButton: {

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    backgroundColor: "#fff",

    paddingVertical: 16,

    borderRadius: 8,

    marginTop: 20,

    gap: 8,

    borderWidth: 1,

    borderColor: "#FF4444",

  },

  logoutButtonText: {

    color: "#FF4444",

    fontSize: 16,

    fontWeight: "600",

  },

});

\*/

# //register screen originale

import React, { useState } from "react";

import {

  View,

  Text,

  TextInput,

  TouchableOpacity,

  StyleSheet,

  SafeAreaView,

  Alert,

  ActivityIndicator,

  KeyboardAvoidingView,

  Platform,

  ScrollView,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import DateTimePicker from "@react-native-community/datetimepicker";

import { authService } from "../services/api";

export default function RegisterScreen({ navigation }) {

  const [formData, setFormData] = useState({

    fiscalCode: "",

    screenName: "",

    email: "",

    phoneNumber: "",

    birthDate: new Date(),

    password: "",

    confirmPassword: "",

  });

  const [loading, setLoading] = useState(false);

  const [showDatePicker, setShowDatePicker] = useState(false);

  const [showPassword, setShowPassword] = useState(false);

  const [showConfirmPassword, setShowConfirmPassword] = useState(false);

  const [emailVerified, setEmailVerified] = useState(false);

  const [phoneVerified, setPhoneVerified] = useState(false);

  const updateFormData = (field, value) => {

    setFormData((prev) => ({ ...prev, [field]: value }));

  };

  const handleDateChange = (event, selectedDate) => {

    setShowDatePicker(false);

    if (selectedDate) {

      updateFormData("birthDate", selectedDate);

    }

  };

  const verifyEmail = async () => {

    if (!formData.email.trim()) {

      Alert.alert("Errore", "Inserisci un indirizzo email");

      return;

    }

    setLoading(true);

    try {

      await authService.verifyEmail(formData.email);

      setEmailVerified(true);

      Alert.alert(

        "Email inviata",

        "Controlla la tua casella di posta per verificare l'email"

      );

    } catch (error) {

      console.error("Errore verifica email:", error);

      Alert.alert("Errore", "Impossibile inviare l'email di verifica");

    } finally {

      setLoading(false);

    }

  };

  const verifyPhone = async () => {

    if (!formData.phoneNumber.trim()) {

      Alert.alert("Errore", "Inserisci un numero di telefono");

      return;

    }

    setLoading(true);

    try {

      await authService.verifyPhone(formData.phoneNumber);

      setPhoneVerified(true);

      Alert.alert(

        "SMS inviato",

        "Controlla i tuoi messaggi per il codice di verifica"

      );

    } catch (error) {

      console.error("Errore verifica telefono:", error);

      Alert.alert("Errore", "Impossibile inviare l'SMS di verifica");

    } finally {

      setLoading(false);

    }

  };

  const handleRegister = async () => {

    // Validazione form

    if (

      !formData.fiscalCode.trim() ||

      !formData.screenName.trim() ||

      !formData.email.trim() ||

      !formData.password.trim()

    ) {

      Alert.alert("Errore", "Compila tutti i campi obbligatori");

      return;

    }

    if (formData.password !== formData.confirmPassword) {

      Alert.alert("Errore", "Le password non coincidono");

      return;

    }

    if (formData.password.length < 6) {

      Alert.alert("Errore", "La password deve essere di almeno 6 caratteri");

      return;

    }

    setLoading(true);

    try {

      const userData = {

        \_id: formData.fiscalCode,

        screenName: formData.screenName,

        Type: "Person",

        fiscalCode: formData.fiscalCode,

        email: formData.email,

        phoneNumber: formData.phoneNumber,

        birthDate: formData.birthDate.toISOString(),

        roles: ["citizen"],

      };

      await authService.register(userData);

      Alert.alert(

        "Registrazione completata",

        "Il tuo account è stato creato con successo. Ora puoi effettuare il login.",

        [

          {

            text: "Vai al Login",

            onPress: () => navigation.navigate("Login"),

          },

        ]

      );

    } catch (error) {

      console.error("Errore registrazione:", error);

      Alert.alert(

        "Errore di registrazione",

        "Impossibile completare la registrazione. Riprova più tardi."

      );

    } finally {

      setLoading(false);

    }

  };

  return (

    <SafeAreaView style={styles.container}>

      <KeyboardAvoidingView

        behavior={Platform.OS === "ios" ? "padding" : "height"}

        style={styles.keyboardView}

      >

        <ScrollView contentContainerStyle={styles.scrollContent}>

          {/\* Header \*/}

          <View style={styles.header}>

            <Text style={styles.title}>Registrati</Text>

            <Text style={styles.subtitle}>

              Crea il tuo account per accedere a tutti i servizi

            </Text>

          </View>

          {/\* Form \*/}

          <View style={styles.form}>

            {/\* Codice Fiscale \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Codice Fiscale \*</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="card"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.fiscalCode}

                  onChangeText={(value) => updateFormData("fiscalCode", value)}

                  placeholder="Inserisci il tuo codice fiscale"

                  autoCapitalize="characters"

                  maxLength={16}

                />

              </View>

            </View>

            {/\* Nome Utente \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Nome Utente \*</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="person"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.screenName}

                  onChangeText={(value) => updateFormData("screenName", value)}

                  placeholder="Inserisci il tuo nome utente"

                  autoCorrect={false}

                />

              </View>

            </View>

            {/\* Email \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Email \*</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="mail"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.email}

                  onChangeText={(value) => updateFormData("email", value)}

                  placeholder="Inserisci la tua email"

                  keyboardType="email-address"

                  autoCapitalize="none"

                  autoCorrect={false}

                />

                <TouchableOpacity

                  onPress={verifyEmail}

                  style={styles.verifyButton}

                  disabled={loading || emailVerified}

                >

                  <Text

                    style={[

                      styles.verifyButtonText,

                      emailVerified && styles.verifiedText,

                    ]}

                  >

                    {emailVerified ? "Verificata" : "Verifica"}

                  </Text>

                </TouchableOpacity>

              </View>

            </View>

            {/\* Numero di Telefono \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Numero di Cellulare</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="call"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.phoneNumber}

                  onChangeText={(value) => updateFormData("phoneNumber", value)}

                  placeholder="+39 123 456 7890"

                  keyboardType="phone-pad"

                />

                <TouchableOpacity

                  onPress={verifyPhone}

                  style={styles.verifyButton}

                  disabled={loading || phoneVerified}

                >

                  <Text

                    style={[

                      styles.verifyButtonText,

                      phoneVerified && styles.verifiedText,

                    ]}

                  >

                    {phoneVerified ? "Verificato" : "Verifica"}

                  </Text>

                </TouchableOpacity>

              </View>

            </View>

            {/\* Data di Nascita \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Data di Nascita</Text>

              <TouchableOpacity

                style={styles.inputWrapper}

                onPress={() => setShowDatePicker(true)}

              >

                <Ionicons

                  name="calendar"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <Text style={styles.dateText}>

                  {formData.birthDate.toLocaleDateString("it-IT")}

                </Text>

              </TouchableOpacity>

            </View>

            {showDatePicker && (

              <DateTimePicker

                value={formData.birthDate}

                mode="date"

                display="default"

                onChange={handleDateChange}

                maximumDate={new Date()}

              />

            )}

            {/\* Password \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Password \*</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="lock-closed"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.password}

                  onChangeText={(value) => updateFormData("password", value)}

                  placeholder="Inserisci la password"

                  secureTextEntry={!showPassword}

                  autoCorrect={false}

                />

                <TouchableOpacity

                  onPress={() => setShowPassword(!showPassword)}

                  style={styles.eyeIcon}

                >

                  <Ionicons

                    name={showPassword ? "eye-off" : "eye"}

                    size={20}

                    color="#666"

                  />

                </TouchableOpacity>

              </View>

            </View>

            {/\* Conferma Password \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Conferma Password \*</Text>

              <View style={styles.inputWrapper}>

                <Ionicons

                  name="lock-closed"

                  size={20}

                  color="#666"

                  style={styles.inputIcon}

                />

                <TextInput

                  style={styles.input}

                  value={formData.confirmPassword}

                  onChangeText={(value) =>

                    updateFormData("confirmPassword", value)

                  }

                  placeholder="Conferma la password"

                  secureTextEntry={!showConfirmPassword}

                  autoCorrect={false}

                />

                <TouchableOpacity

                  onPress={() => setShowConfirmPassword(!showConfirmPassword)}

                  style={styles.eyeIcon}

                >

                  <Ionicons

                    name={showConfirmPassword ? "eye-off" : "eye"}

                    size={20}

                    color="#666"

                  />

                </TouchableOpacity>

              </View>

            </View>

            {/\* Register Button \*/}

            <TouchableOpacity

              style={[

                styles.registerButton,

                loading && styles.registerButtonDisabled,

              ]}

              onPress={handleRegister}

              disabled={loading}

            >

              {loading ? (

                <ActivityIndicator color="#fff" />

              ) : (

                <>

                  <Ionicons name="person-add" size={20} color="#fff" />

                  <Text style={styles.registerButtonText}>Registrati</Text>

                </>

              )}

            </TouchableOpacity>

            {/\* Login Link \*/}

            <View style={styles.loginContainer}>

              <Text style={styles.loginText}>Hai già un account? </Text>

              <TouchableOpacity onPress={() => navigation.navigate("Login")}>

                <Text style={styles.loginLink}>Accedi qui</Text>

              </TouchableOpacity>

            </View>

          </View>

        </ScrollView>

      </KeyboardAvoidingView>

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f8f9fa",

  },

  keyboardView: {

    flex: 1,

  },

  scrollContent: {

    flexGrow: 1,

    paddingHorizontal: 20,

    paddingVertical: 20,

  },

  header: {

    alignItems: "center",

    marginBottom: 30,

  },

  title: {

    fontSize: 28,

    fontWeight: "bold",

    color: "#333",

    marginTop: 15,

    marginBottom: 10,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

  },

  form: {

    flex: 1,

  },

  inputContainer: {

    marginBottom: 20,

  },

  label: {

    fontSize: 16,

    fontWeight: "600",

    color: "#333",

    marginBottom: 8,

  },

  inputWrapper: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "#fff",

    borderRadius: 12,

    borderWidth: 1,

    borderColor: "#ddd",

    paddingHorizontal: 15,

    height: 50,

  },

  inputIcon: {

    marginRight: 10,

  },

  input: {

    flex: 1,

    fontSize: 16,

    color: "#333",

  },

  eyeIcon: {

    padding: 5,

  },

  verifyButton: {

    backgroundColor: "#FF6B35",

    paddingHorizontal: 12,

    paddingVertical: 6,

    borderRadius: 6,

  },

  verifyButtonText: {

    color: "#fff",

    fontSize: 12,

    fontWeight: "600",

  },

  verifiedText: {

    color: "#28a745",

  },

  dateText: {

    flex: 1,

    fontSize: 16,

    color: "#333",

  },

  registerButton: {

    backgroundColor: "#FF6B35",

    paddingVertical: 15,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    marginTop: 20,

    marginBottom: 20,

    elevation: 3,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.25,

    shadowRadius: 3.84,

  },

  registerButtonDisabled: {

    backgroundColor: "#ccc",

  },

  registerButtonText: {

    color: "#fff",

    fontSize: 16,

    fontWeight: "bold",

    marginLeft: 8,

  },

  loginContainer: {

    flexDirection: "row",

    justifyContent: "center",

    alignItems: "center",

    marginTop: 20,

  },

  loginText: {

    fontSize: 14,

    color: "#666",

  },

  loginLink: {

    fontSize: 14,

    color: "#FF6B35",

    fontWeight: "600",

  },

});

# register description

/\*// screens/ReportDescription.tsx

import React from "react";

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

export const ReportDescription: React.FC = () => {

  return (

    <View style={styles.container}>

      <Text style={styles.text}>ciao</Text>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

  },

  text: {

    fontSize: 28,

    fontWeight: "bold",

  },

});

\*/

# report screen

/\*import React, { useState } from "react";

import {

  View,

  Text,

  StyleSheet,

  TouchableOpacity,

  ScrollView,

} from "react-native";

import { NativeStackNavigationProp } from "@react-navigation/native-stack";

import { RootStackParamList } from "../navigation/types";

type NavigationProp = NativeStackNavigationProp<

  RootStackParamList,

  "ReportTab"

>;

type Props = {

  navigation: NavigationProp;

};

export const ReportScreen: React.FC<Props> = ({ navigation }) => {

  const [selectedOption, setSelectedOption] = useState<string | null>(null);

  const options = [

    { label: "Incendio", icon: "fire" },

    { label: "Frana", icon: "mountain" },

    { label: "Alluvione", icon: "water" },

    { label: "Tsunami", icon: "wave-square" },

    { label: "Terremoto", icon: "house-damage" },

    { label: "Tempesta", icon: "cloud-showers-heavy" },

    { label: "Vento Forte", icon: "wind" },

    { label: "Neve", icon: "snowflake" },

    { label: "Blocco Stradale", icon: "road" },

    { label: "Ricerca e Soccorso", icon: "search" },

  ];

  const handleNext = () => {

    if (selectedOption) {

      navigation.navigate("ReportTab", { screen: "ReportDescription" });

    }

  };

  return (

    <View style={styles.container}>

      <View style={styles.header}>

        <Text style={styles.headerText}>Nuova Segnalazione</Text>

        <Text style={styles.stepText}>1/4</Text>

      </View>

      <ScrollView contentContainerStyle={styles.scrollContainer}>

        {options.map((option, index) => (

          <TouchableOpacity

            key={index}

            style={[

              styles.optionItem,

              selectedOption === option.label && styles.selectedOptionItem,

            ]}

            onPress={() => setSelectedOption(option.label)}

          >

            <Text style={styles.optionText}>{option.label}</Text>

            <View style={styles.radioCircle}>

              {selectedOption === option.label && (

                <View style={styles.radioDot} />

              )}

            </View>

          </TouchableOpacity>

        ))}

      </ScrollView>

      <TouchableOpacity

        style={[

          styles.nextButton,

          !selectedOption && { backgroundColor: "#ccc" },

        ]}

        onPress={handleNext}

        disabled={!selectedOption}

      >

        <Text style={styles.nextButtonText}>Avanti</Text>

      </TouchableOpacity>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#fff",

  },

  header: {

    flexDirection: "row",

    justifyContent: "space-between",

    paddingHorizontal: 16,

    paddingVertical: 20,

    backgroundColor: "#0D65BE",

  },

  headerText: {

    fontSize: 18,

    color: "#fff",

    fontWeight: "bold",

  },

  stepText: {

    fontSize: 16,

    color: "#fff",

  },

  scrollContainer: {

    paddingHorizontal: 16,

    paddingBottom: 20,

  },

  optionItem: {

    flexDirection: "row",

    alignItems: "center",

    padding: 14,

    backgroundColor: "#f9f9f9",

    borderRadius: 8,

    marginBottom: 12,

    borderWidth: 1,

    borderColor: "#ddd",

  },

  selectedOptionItem: {

    backgroundColor: "#e6f0fb",

    borderColor: "#0D65BE",

  },

  icon: {

    marginRight: 16,

    width: 24,

    textAlign: "center",

  },

  optionText: {

    flex: 1,

    fontSize: 16,

    color: "#333",

  },

  radioCircle: {

    width: 20,

    height: 20,

    borderRadius: 10,

    borderWidth: 2,

    borderColor: "#0D65BE",

    alignItems: "center",

    justifyContent: "center",

  },

  radioDot: {

    width: 10,

    height: 10,

    borderRadius: 5,

    backgroundColor: "#0D65BE",

  },

  nextButton: {

    backgroundColor: "#0D65BE",

    paddingVertical: 14,

    marginHorizontal: 20,

    borderRadius: 30,

    marginBottom: 20,

    alignItems: "center",

  },

  nextButtonText: {

    color: "#fff",

    fontSize: 18,

    fontWeight: "bold",

  },

});

\*/

# //app.js salvatore app js

import React from 'react';

import { StatusBar } from 'expo-status-bar';

import AppNavigator from './src/navigation/AppNavigator';

export default function App() {

  return (

    <>

      <StatusBar style="light" backgroundColor="#FF6B35" />

      <AppNavigator />

    </>

  );

}

# //app js ok fino alle 10 e 55 del 24 set 25

import React from 'react';

import { StatusBar } from 'expo-status-bar';

import AppNavigator from './src/navigation/AppNavigator';

export default function App() {

  return (

    <>

      <StatusBar style="light" backgroundColor="#FF6B35" />

      <AppNavigator />

    </>

  );

}

# //app navigator js ok fino a 14 e 27 del 24-9-25

import React from "react";

import { NavigationContainer } from "@react-navigation/native";

import { createStackNavigator } from "@react-navigation/stack";

import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";

import { Ionicons } from "@expo/vector-icons";

// Import delle schermate (da creare)

import HomeScreen from "../screens/HomeScreen";

import LoginScreen from "../screens/LoginScreen";

import RegisterScreen from "../screens/RegisterScreen";

import EventsMapScreen from "../screens/EventsMapScreen";

import EventsListScreen from "../screens/EventsListScreen";

import CreateReportScreen from "../screens/CreateReportScreen";

import WeatherBulletinScreen from "../screens/WeatherBulletinScreen";

import CommunicationsScreen from "../screens/CommunicationsScreen";

import ProfileScreen from "../screens/ProfileScreen";

const Stack = createStackNavigator();

const Tab = createBottomTabNavigator();

// Navigazione principale con tab

function MainTabNavigator() {

  return (

    <Tab.Navigator

      screenOptions={({ route }) => ({

        tabBarIcon: ({ focused, color, size }) => {

          let iconName;

          if (route.name === "Eventi") {

            iconName = focused ? "map" : "map-outline";

          } else if (route.name === "Segnala") {

            iconName = focused ? "add-circle" : "add-circle-outline";

          } else if (route.name === "Bollettino") {

            iconName = focused ? "cloud" : "cloud-outline";

          } else if (route.name === "Comunicazioni") {

            iconName = focused ? "chatbubbles" : "chatbubbles-outline";

          } else if (route.name === "Profilo") {

            iconName = focused ? "person" : "person-outline";

          }

          return <Ionicons name={iconName} size={size} color={color} />;

        },

        tabBarActiveTintColor: "#FF6B35",

        tabBarInactiveTintColor: "gray",

      })}

    >

      <Tab.Screen

        name="Eventi"

        component={EventsMapScreen}

        options={{

          title: "Eventi",

          headerShown: false,

        }}

      />

      <Tab.Screen

        name="Segnala"

        component={CreateReportScreen}

        options={{ title: "Segnala", headerShown: false }}

      />

      <Tab.Screen

        name="Bollettino"

        component={WeatherBulletinScreen}

        options={{ title: "Bollettino", headerShown: false }}

      />

      <Tab.Screen

        name="Comunicazioni"

        component={CommunicationsScreen}

        options={{ title: "Comunicazioni", headerShown: false }}

      />

      <Tab.Screen

        name="Profilo"

        component={ProfileScreen}

        options={{ title: "Profilo", headerShown: false }}

      />

    </Tab.Navigator>

  );

}

// Navigazione principale dell'app

export default function AppNavigator() {

  return (

    <NavigationContainer>

      <Stack.Navigator

        initialRouteName="Home"

        screenOptions={{

          headerStyle: {

            backgroundColor: "#FF6B35",

          },

          headerTintColor: "#fff",

          headerTitleStyle: {

            fontWeight: "bold",

          },

        }}

      >

        <Stack.Screen

          name="Home"

          component={HomeScreen}

          options={{

            title: "Protezione Civile Calabria",

            headerShown: false,

          }}

        />

        <Stack.Screen

          name="Login"

          component={LoginScreen}

          options={{ title: "Accedi", headerShown: false }}

        />

        <Stack.Screen

          name="Register"

          component={RegisterScreen}

          options={{ title: "Registrati", headerShown: false }}

        />

        <Stack.Screen

          name="Main"

          component={MainTabNavigator}

          options={{ headerShown: false }}

        />

        <Stack.Screen

          name="EventsList"

          component={EventsListScreen}

          options={{ title: "Lista Eventi", headerShown: false }}

        />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

# //app json app.json

{

  "expo": {

    "name": "ProtezioneCivileCalabria",

    "slug": "ProtezioneCivileCalabria",

    "version": "1.0.0",

    "orientation": "portrait",

    "icon": "./src/components/Logo.png",

    "userInterfaceStyle": "light",

    "newArchEnabled": true,

    "splash": {

      "image": "./src/components/Logo.png",

      "resizeMode": "contain",

      "backgroundColor": "#ffffff"

    },

    "ios": {

      "supportsTablet": true

    },

    "android": {

      "adaptiveIcon": {

        "foregroundImage": "./src/components/Logo.png",

        "backgroundColor": "#ffffff"

      },

      "edgeToEdgeEnabled": true

    },

    "web": {

      "favicon": "./assets/favicon.png"

    }

  }

}

# //event list screen

import React, { useState, useEffect } from "react";

import {

  View,

  Text,

  FlatList,

  TouchableOpacity,

  StyleSheet,

  RefreshControl,

  Alert,

  Image,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

export default function EventsListScreen({ navigation }) {

  const [events, setEvents] = useState([]);

  const [loading, setLoading] = useState(false);

  const [refreshing, setRefreshing] = useState(false);

  const [showMyReports, setShowMyReports] = useState(false);

  const [selectedType, setSelectedType] = useState("Eventi"); // Stato per gestire "Eventi" o "Segnalazioni"

  const [isDropdownVisible, setIsDropdownVisible] = useState(false); // Stato per la visibilità del menu a tendina

  useEffect(() => {

    loadEvents();

  }, [showMyReports, selectedType]);

  const loadEvents = async () => {

    setLoading(true);

    try {

      const mockEvents = [

        {

          id: "1",

          title: "Allagamento Strada Provinciale",

          description:

            "Segnalazione di allagamento sulla SP 123 all'altezza del km 15. La strada risulta impraticabile per i veicoli.",

          type: "flood",

          status: "open",

          location: "Cosenza, SP 123",

          createdAt: "2024-09-17T08:30:00Z",

          priority: "high",

          hasInjured: false,

          hasVictims: false,

        },

        {

          id: "2",

          title: "Frana Località Monte Alto",

          description:

            "Frana che blocca completamente la viabilità sulla strada comunale. Necessario intervento urgente.",

          type: "landslide",

          status: "open",

          location: "Catanzaro, Monte Alto",

          createdAt: "2024-09-17T07:15:00Z",

          priority: "critical",

          hasInjured: true,

          hasVictims: false,

        },

        {

          id: "3",

          title: "Incendio Boschivo",

          description:

            "Principio di incendio in zona boschiva. Fumo visibile dalla strada statale.",

          type: "fire",

          status: "open",

          location: "Reggio Calabria, Aspromonte",

          createdAt: "2024-09-17T06:45:00Z",

          priority: "high",

          hasInjured: false,

          hasVictims: false,

        },

        {

          id: "4",

          title: "Albero Caduto",

          description:

            "Albero caduto sulla carreggiata a causa del vento forte.",

          type: "obstacle",

          status: "resolved",

          location: "Crotone, Via Roma",

          createdAt: "2024-09-16T18:20:00Z",

          priority: "medium",

          hasInjured: false,

          hasVictims: false,

        },

      ];

      const filteredEvents =

        selectedType === "Eventi"

          ? mockEvents

          : mockEvents.filter((event) => event.id === "1"); // Solo "Segnalazioni" (Le mie segnalazioni) se selezionato

      setEvents(filteredEvents);

    } catch (error) {

      console.error("Errore caricamento Eventi:", error);

      Alert.alert("Errore", "Impossibile caricare gli Eventi");

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  const onRefresh = () => {

    setRefreshing(true);

    loadEvents();

  };

  const getEventicon = (eventType) => {

    switch (eventType) {

      case "flood":

        return "water";

      case "landslide":

        return "triangle";

      case "fire":

        return "flame";

      case "obstacle":

        return "warning";

      default:

        return "alert-circle";

    }

  };

  const getEventColor = (eventType) => {

    switch (eventType) {

      case "flood":

        return "#007AFF";

      case "landslide":

        return "#FF9500";

      case "fire":

        return "#FF3B30";

      case "obstacle":

        return "#8E8E93";

      default:

        return "#FF6B35";

    }

  };

  const getPriorityColor = (priority) => {

    switch (priority) {

      case "critical":

        return "#FF3B30";

      case "high":

        return "#FF9500";

      case "medium":

        return "#FFCC00";

      case "low":

        return "#34C759";

      default:

        return "#8E8E93";

    }

  };

  const getStatusText = (status) => {

    switch (status) {

      case "open":

        return "Aperto";

      case "in\_progress":

        return "In corso";

      case "resolved":

        return "Risolto";

      default:

        return "Sconosciuto";

    }

  };

  const handleEventPress = (event) => {

    Alert.alert(

      event.title,

      `${event.description}\n\nLocalità: ${event.location}\nData: ${formatDate(

        event.createdAt

      )}`,

      [

        { text: "Chiudi", style: "cancel" },

        { text: "Dettagli", onPress: () => showEventDetails(event) },

      ]

    );

  };

  const showEventDetails = (event) => {

    console.log("Mostra dettagli evento:", event);

  };

  const formatDate = (dateString) => {

    const date = new Date(dateString);

    return date.toLocaleDateString("it-IT", {

      day: "2-digit",

      month: "2-digit",

      year: "numeric",

      hour: "2-digit",

      minute: "2-digit",

    });

  };

  const renderEventitem = ({ item }) => (

    <TouchableOpacity

      style={styles.eventCard}

      onPress={() => handleEventPress(item)}

    >

      <View style={styles.eventHeader}>

        <View style={styles.EventiconContainer}>

          <Ionicons

            name={getEventicon(item.type)}

            size={24}

            color={getEventColor(item.type)}

          />

        </View>

        <View style={styles.Eventinfo}>

          <Text style={styles.eventTitle}>{item.title}</Text>

          <Text style={styles.eventLocation}>{item.location}</Text>

        </View>

        <View style={styles.eventStatus}>

          <View

            style={[

              styles.priorityBadge,

              { backgroundColor: getPriorityColor(item.priority) },

            ]}

          >

            <Text style={styles.priorityText}>

              {item.priority.toUpperCase()}

            </Text>

          </View>

        </View>

      </View>

      <Text style={styles.eventDescription} numberOfLines={2}>

        {item.description}

      </Text>

      <View style={styles.eventFooter}>

        <Text style={styles.eventDate}>{formatDate(item.createdAt)}</Text>

        <View style={styles.eventBadges}>

          {item.hasInjured && (

            <View style={styles.warningBadge}>

              <Ionicons name="medical" size={12} color="#fff" />

              <Text style={styles.badgeText}>Feriti</Text>

            </View>

          )}

          {item.hasVictims && (

            <View style={styles.criticalBadge}>

              <Ionicons name="skull" size={12} color="#fff" />

              <Text style={styles.badgeText}>Vittime</Text>

            </View>

          )}

          <View

            style={[

              styles.statusBadge,

              {

                backgroundColor:

                  item.status === "resolved" ? "#34C759" : "#FF6B35",

              },

            ]}

          >

            <Text style={styles.badgeText}>{getStatusText(item.status)}</Text>

          </View>

        </View>

      </View>

    </TouchableOpacity>

  );

  return (

    <View style={styles.container}>

      {/\* Header \*/}

      <View style={styles.header2}>

        <Text style={styles.headerTitle2}>

          Protezione Civile | Regione Calabria

        </Text>

        <Image

          source={require("../components/Logo.png")}

          style={styles.reportButtonImage}

        />

      </View>

      <View style={styles.header}>

        <Text style={styles.headerTitle}>ProCiv Calabria</Text>

        <TouchableOpacity

          onPress={() => navigation.navigate("Main", { screen: "Segnala" })}

          activeOpacity={1}

          style={styles.reportButton}

        >

          <Text style={styles.reportButtonText}>Segnala</Text>

        </TouchableOpacity>

      </View>

      <View style={styles.tabContainer}>

        <TouchableOpacity

          style={styles.tab}

          onPress={() => navigation.navigate("Main", { screen: "Eventi" })}

        >

          <Text style={[styles.tabText]}>Mappa</Text>

        </TouchableOpacity>

        <TouchableOpacity

          style={styles.tab}

          onPress={() => navigation.navigate("EventsList")}

        >

          <Text style={(styles.tabText, styles.activeTabText)}>Lista</Text>

        </TouchableOpacity>

      </View>

      {/\* Dropdown personalizzato per Eventi / Le mie Segnalazioni \*/}

      <View style={styles.dropdownContainer}>

        <Text style={styles.dropdownTitle}>EVENTI/LE MIE SEGNALAZIONI</Text>

        <TouchableOpacity

          style={styles.dropdownButton}

          onPress={() => setIsDropdownVisible(!isDropdownVisible)} // Toggle dropdown visibility

        >

          <Text style={styles.dropdownText}>{selectedType}</Text>

          <Ionicons

            name={isDropdownVisible ? "chevron-up" : "chevron-down"}

            size={20}

            color="#007AFF"

          />

        </TouchableOpacity>

        {isDropdownVisible && (

          <View style={styles.dropdownMenu}>

            <Text>Seleziona tipologia</Text>

            <TouchableOpacity onPress={() => setSelectedType("Eventi")}>

              <Text style={styles.dropdownItem}>Eventi</Text>

            </TouchableOpacity>

            <TouchableOpacity onPress={() => setSelectedType("Segnalazioni")}>

              <Text style={styles.dropdownItem}>Segnalazioni</Text>

            </TouchableOpacity>

          </View>

        )}

      </View>

      {/\* Lista degli eventi \*/}

      <FlatList

        data={events}

        renderItem={renderEventitem}

        keyExtractor={(item) => item.id}

        contentContainerStyle={styles.listContainer}

        refreshControl={

          <RefreshControl

            refreshing={refreshing}

            onRefresh={onRefresh}

            colors={["#FF6B35"]}

          />

        }

        ListEmptyComponent={

          <View style={styles.emptyContainer}>

            <Text style={styles.emptyText}>

              {selectedType === "Segnalazioni"

                ? "Nessuna segnalazione trovata"

                : "Nessun evento disponibile"}

            </Text>

          </View>

        }

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: { marginTop: 20, flex: 1 },

  dropdownContainer: { paddingVertical: 15, paddingHorizontal: 20 },

  dropdownTitle: { fontSize: 18, marginBottom: 10, fontWeight: "bold" },

  dropdownButton: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    padding: 10,

    backgroundColor: "#f0f0f0",

    borderRadius: 8,

  },

  dropdownText: { fontSize: 16, color: "#333" },

  dropdownMenu: {

    marginTop: 5,

    backgroundColor: "#f0f0f0",

    borderRadius: 8,

    overflow: "hidden",

  },

  dropdownItem: {

    paddingVertical: 10,

    paddingHorizontal: 15,

    fontSize: 16,

    color: "#333",

  },

  listContainer: { paddingBottom: 20 },

  eventCard: {

    backgroundColor: "#fff",

    marginBottom: 15,

    padding: 15,

    borderRadius: 8,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 4,

    elevation: 3,

  },

  eventHeader: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

  },

  EventiconContainer: {

    backgroundColor: "#f0f0f0",

    padding: 10,

    borderRadius: 50,

  },

  Eventinfo: { flex: 1, marginLeft: 10 },

  eventTitle: { fontWeight: "bold", fontSize: 16 },

  eventLocation: { color: "#555" },

  eventStatus: { flexDirection: "row", alignItems: "center" },

  priorityBadge: {

    borderRadius: 12,

    paddingVertical: 5,

    paddingHorizontal: 10,

    marginTop: 5,

  },

  priorityText: { color: "#fff", fontSize: 12 },

  eventDescription: { marginTop: 10, color: "#555" },

  eventFooter: {

    marginTop: 10,

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

  },

  eventDate: { fontSize: 12, color: "#888" },

  eventBadges: { flexDirection: "row", alignItems: "center" },

  warningBadge: {

    backgroundColor: "#FF3B30",

    paddingVertical: 3,

    paddingHorizontal: 6,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    marginRight: 5,

  },

  criticalBadge: {

    backgroundColor: "#FF9500",

    paddingVertical: 3,

    paddingHorizontal: 6,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    marginRight: 5,

  },

  statusBadge: { paddingVertical: 3, paddingHorizontal: 6, borderRadius: 12 },

  badgeText: { color: "#fff", fontSize: 10, marginLeft: 3 },

  emptyContainer: {

    justifyContent: "center",

    alignItems: "center",

    padding: 20,

  },

  emptyText: { fontSize: 16, color: "#888" },

  reportButton: {

    backgroundColor: "#FF6B35",

    paddingVertical: 8,

    paddingHorizontal: 16,

    borderRadius: 5,

    borderColor: "white",

    borderWidth: 3,

  },

  reportButtonText: { color: "#fff", fontSize: 16, fontWeight: "bold" },

  header: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    backgroundColor: "#0091D6",

    paddingHorizontal: 20,

    paddingVertical: 10,

  },

  header2: {

    flexDirection: "row",

    justifyContent: "space-between",

    alignItems: "center",

    backgroundColor: "#0091D6",

    paddingHorizontal: 20,

    paddingVertical: 10,

  },

  headerTitle2: { fontSize: 16, color: "white" },

  headerTitle: { fontSize: 20, color: "white", fontWeight: "bold" },

  tabContainer: {

    flexDirection: "row",

    borderBottomWidth: 1,

    borderBottomColor: "#ddd",

  },

  tab: {

    flex: 1,

    flexDirection: "row",

    justifyContent: "center",

    alignItems: "center",

    paddingVertical: 10,

  },

  tabText: {

    marginLeft: 8,

    color: "#666",

  },

  activeTabText: {

    color: "blue",

    fontWeight: "bold",

  },

  reportButtonImage: { width: 50, height: 50, resizeMode: "contain" },

});

# //register screen ok fino alle 14 e 16 del 24 set 25

pulsante data di nascita ok

pulsante verifica mail, verifica numero e procedi ancora da sistemare ricomincia da qui

pulsante annulla ok

## //codice

//register screen ok fino alle 10 e 48 del 24 set 25

import React, { useState } from "react";

import {

  Button,

  View,

  Text,

  TextInput,

  TouchableOpacity,

  StyleSheet,

  SafeAreaView,

  Alert,

  ActivityIndicator,

  KeyboardAvoidingView,

  Platform,

  ScrollView,

} from "react-native";

import DateTimePicker from "@react-native-community/datetimepicker";

import { authService } from "../services/api";

export default function RegisterScreen({ navigation }) {

  const [isPressed, setIsPressed] = useState(false);

  const [formData, setFormData] = useState({

    nome: "",

    cognome: "",

    codiceFiscale: "",

    emailDiRegistrazione: "",

    phoneNumber: "",

    birthDate: null,

    sesso: "",

  });

  const [loading, setLoading] = useState(false);

  const [showDatePicker, setShowDatePicker] = useState(false);

  const updateFormData = (field, value) => {

    setFormData((prev) => ({ ...prev, [field]: value }));

  };

  const isFormValid = () => {

    const {

      nome,

      cognome,

      codiceFiscale,

      emailDiRegistrazione,

      birthDate,

      sesso,

    } = formData;

    return (

      nome.trim() &&

      cognome.trim() &&

      codiceFiscale.trim() &&

      emailDiRegistrazione.trim() &&

      birthDate &&

      sesso

    );

  };

  const handleRegister = async () => {

    if (!isFormValid()) {

      Alert.alert("Errore", "Compila tutti i campi obbligatori");

      return;

    }

    setLoading(true);

    try {

      await authService.register({

        Nome: formData.nome,

        Cognome: formData.cognome,

        codiceFiscale: formData.codiceFiscale,

        email: formData.emailDiRegistrazione,

        phoneNumber: formData.phoneNumber,

        birthDate: formData.birthDate.toISOString(),

        sesso: formData.sesso,

      });

      Alert.alert(

        "Registrazione completata",

        "Il tuo account è stato creato con successo. Ora puoi effettuare il login.",

        [{ text: "Vai al Login", onPress: () => navigation.navigate("Login") }]

      );

    } catch (error) {

      console.error("Errore registrazione:", error);

      Alert.alert(

        "Errore di registrazione",

        "Impossibile completare la registrazione. Riprova più tardi."

      );

    } finally {

      setLoading(false);

    }

  };

  const handleDateChange = (event, selectedDate) => {

    setShowDatePicker(false);

    if (selectedDate) {

      updateFormData("birthDate", selectedDate);

    }

  };

  const renderSubmitButton = (label, extraStyle = {}) => (

    <TouchableOpacity

      style={[

        styles.registerButton,

        loading && styles.registerButtonDisabled,

        extraStyle,

      ]}

      onPress={handleRegister}

      disabled={loading}

    >

      {loading ? (

        <ActivityIndicator color="#fff" />

      ) : (

        <Text style={styles.registerButtonText}>{label}</Text>

      )}

    </TouchableOpacity>

  );

  return (

    <SafeAreaView style={styles.container}>

      <KeyboardAvoidingView

        behavior={Platform.OS === "ios" ? "padding" : "height"}

        style={styles.keyboardView}

      >

        <ScrollView contentContainerStyle={styles.scrollContent}>

          {/\* Header \*/}

          <View style={styles.header}>

            <Text style={styles.title}>Registrati</Text>

            <Text style={styles.subtitle}>

              I campi contrassegnati da \* sono obbligatori

            </Text>

          </View>

          {/\* Form \*/}

          <View style={styles.form}>

            {/\* Nome \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Nome \*</Text>

              <TextInput

                style={styles.input}

                value={formData.nome}

                onChangeText={(text) => updateFormData("nome", text)}

              />

            </View>

            {/\* Cognome \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Cognome \*</Text>

              <TextInput

                style={styles.input}

                value={formData.cognome}

                onChangeText={(text) => updateFormData("cognome", text)}

              />

            </View>

            {/\* Codice Fiscale \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Codice Fiscale \*</Text>

              <TextInput

                style={styles.input}

                value={formData.codiceFiscale}

                onChangeText={(text) => updateFormData("codiceFiscale", text)}

                maxLength={16}

              />

            </View>

            {/\* Data di Nascita \*/}

            <TouchableOpacity

              style={styles.dateButton}

              onPress={() => setShowDatePicker(true)}

            >

              <Text style={styles.dateButtonText}>Data di Nascita</Text>

            </TouchableOpacity>

            {showDatePicker && (

              <DateTimePicker

                value={formData.birthDate || new Date()}

                mode="date"

                display="default"

                onChange={handleDateChange}

              />

            )}

            {formData.birthDate && (

              <Text style={styles.selectedDateText}>

                {formData.birthDate.toLocaleDateString()}

              </Text>

            )}

            {/\* Sesso \*/}

            <View style={styles.sessoContainer}>

              <Text style={styles.label}>Sesso \*</Text>

              <View style={styles.sessoOptions}>

                {["Maschio", "Femmina"].map((option) => (

                  <TouchableOpacity

                    key={option}

                    style={styles.radioButton}

                    onPress={() => updateFormData("sesso", option)}

                  >

                    <View

                      style={[

                        styles.radioCircle,

                        formData.sesso === option && styles.selectedCircle,

                      ]}

                    />

                    <Text style={styles.radioText}>{option}</Text>

                  </TouchableOpacity>

                ))}

              </View>

            </View>

            <View style={styles.horizontalLine} />

            {/\* Email \*/}

            <Text style={styles.title2}>Email di registrazione \*</Text>

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Email</Text>

              <TextInput

                style={styles.input}

                value={formData.emailDiRegistrazione}

                onChangeText={(text) =>

                  updateFormData("emailDiRegistrazione", text)

                }

                keyboardType="email-address"

                autoCapitalize="none"

              />

            </View>

            {renderSubmitButton("Verifica email")}

            <View style={styles.horizontalLine} />

            {/\* Numero di cellulare \*/}

            <Text style={styles.title2}>Numero di cellulare</Text>

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Numero di cellulare</Text>

              <TextInput

                style={styles.input}

                value={formData.phoneNumber}

                onChangeText={(text) => updateFormData("phoneNumber", text)}

                keyboardType="phone-pad"

              />

            </View>

            {renderSubmitButton("Verifica numero")}

            {renderSubmitButton("Procedi", styles.registerButton2)}

            <View style={styles.bottomContainer}>

              <TouchableOpacity

                onPress={() => navigation.goBack()}

                style={styles.cancelButton}

              >

                <Text style={styles.cancelButtonText}>Annulla</Text>

              </TouchableOpacity>

            </View>

          </View>

        </ScrollView>

      </KeyboardAvoidingView>

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f8f9fa",

  },

  scrollContent: {

    paddingHorizontal: 20,

    paddingVertical: 40,

  },

  header: {

    marginBottom: 30,

  },

  title: {

    fontSize: 28,

    fontWeight: "bold",

    color: "#333",

    marginBottom: 10,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    marginTop: 10,

  },

  title2: {

    fontSize: 20,

    color: "#666",

    marginBottom: 20,

    textAlign: "center",

  },

  form: {

    flex: 1,

  },

  label: {

    fontSize: 16,

    fontWeight: "600",

    color: "#333",

  },

  input: {

    height: 50,

    fontSize: 16,

    color: "#333",

    borderBottomWidth: 2,

    borderBottomColor: "#333",

    paddingHorizontal: 0,

    marginBottom: 20,

  },

  dateButton: {

    backgroundColor: "#28a745",

    paddingVertical: 12,

    paddingHorizontal: 25,

    borderRadius: 12,

    alignItems: "center",

    justifyContent: "center",

    marginBottom: 20,

    alignSelf: "center",

  },

  dateButtonText: {

    color: "#fff",

    fontSize: 16,

    fontWeight: "600",

  },

  cancelButton: {

    alignItems: "center",

    marginBottom: 40,

  },

  cancelButtonText: {

    color: "#000000ff",

    fontSize: 16,

    fontWeight: "bold",

  },

  selectedDateText: {

    fontSize: 24,

    color: "#1E3A8A",

    textAlign: "center",

  },

  sessoContainer: {

    marginBottom: 10,

  },

  sessoOptions: {

    flexDirection: "column",

    justifyContent: "space-between",

  },

  radioButton: {

    flexDirection: "row",

    alignItems: "center",

    marginBottom: 10,

  },

  radioCircle: {

    height: 20,

    width: 20,

    borderRadius: 10,

    borderWidth: 2,

    borderColor: "#333",

    marginRight: 10,

  },

  selectedCircle: {

    backgroundColor: "#28a745",

  },

  radioText: {

    fontSize: 16,

    color: "#333",

    paddingVertical: 15,

  },

  bottomContainer: {

    justifyContent: "flex-end",

    paddingHorizontal: 20,

  },

  registerButton: {

    backgroundColor: "#c8c8c8",

    paddingVertical: 15,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    marginTop: 10,

    marginBottom: 5,

    elevation: 3,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.25,

    shadowRadius: 3.84,

  },

  registerButton2: {

    backgroundColor: "#c8c8c8",

    paddingVertical: 15,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    marginTop: 150,

    marginBottom: 30,

    elevation: 3,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.25,

    shadowRadius: 3.84,

  },

  registerButtonDisabled: {

    backgroundColor: "#ccc",

  },

  registerButtonText: {

    color: "#393939",

    fontSize: 16,

    fontWeight: "bold",

    marginLeft: 8,

  },

  horizontalLine: {

    borderBottomWidth: 1,

    borderBottomColor: "#000",

    marginVertical: 30,

  },

});