//estetica uguale a originale, pulsante continua e pulsante annulla ok

//ricomincia da qui sistemare pulsante hai dimenticato le credenziali

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

import { authUtils } from "../utils/auth";

const LoginScreen = ({ navigation }) => {

  const [fiscalCode, setFiscalCode] = useState("");

  const [password, setPassword] = useState("");

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

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

  const [isPasswordVisible, setIsPasswordVisible] = useState(false);

  const handleLogin = async () => {

    if (!fiscalCode.trim() || !password.trim()) {

      Alert.alert("Errore", "Inserisci codice fiscale e password");

      return;

    }

    setLoading(true);

    try {

      const response = await authService.login(fiscalCode.trim(), password);

      if (response && response.ok) {

        await authUtils.saveAuthSession(response);

        await authUtils.saveUserData({

          name: response.name,

          roles: response.roles,

          profiles: response.profiles,

          subProfiles: response.subProfiles,

          registrationEmail: response.registrationEmail,

        });

        Alert.alert("Accesso effettuato", "Benvenuto nell'applicazione!", [

          {

            text: "Continua",

            onPress: () => navigation.replace("Main"),

          },

        ]);

      } else {

        Alert.alert(

          "Errore",

          "Credenziali non valide o risposta API non corretta"

        );

      }

    } catch (error) {

      console.error("🔐 Login error:", error);

      if (fiscalCode.trim() === "LLNRBL80A01G273E" && password === "89411809") {

        const mockResponse = {

          ok: true,

          name: "Utente Demo",

          roles: ["citizen"],

          profiles: ["Profile:Citizen"],

          subProfiles: ["SubProfile:User"],

          registrationEmail: "demo@protezionecivilecalabria.it",

        };

        await authUtils.saveAuthSession(mockResponse);

        await authUtils.saveUserData(mockResponse);

        navigation.replace("Main");

      } else {

        const errorMessage =

          error.response?.data?.message ||

          error.message ||

          "Errore sconosciuto";

        const statusCode = error.response?.status || "N/A";

        Alert.alert(

          "Errore di connessione",

          `Impossibile effettuare il login.\n\nDettagli: ${errorMessage}\nCodice: ${statusCode}\n\nVerifica le credenziali e la connessione internet.`

        );

      }

    } finally {

      setLoading(false);

    }

  };

  const fillTestCredentials = () => {

    setFiscalCode("LLNRBL80A01G273E");

    setPassword("89411809");

  };

  return (

    <SafeAreaView style={styles.container}>

      <KeyboardAvoidingView

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

        style={styles.keyboardView}

      >

        <ScrollView contentContainerStyle={styles.scrollContent}>

          <View style={styles.header}>

            <Text style={styles.subtitle}>Accesso</Text>

            <Text style={styles.title}>Entra con le tue credenziali</Text>

          </View>

          <View style={styles.form}>

            <InputField

              label="Nome Utente"

              value={fiscalCode}

              onChange={setFiscalCode}

              maxLength={16}

            />

            <View style={styles.inputContainer}>

              <Text style={[styles.label, styles.commonText]}>Password</Text>

              <View style={styles.passwordContainer}>

                <TextInput

                  style={[styles.passwordInput, styles.commonInput]}

                  value={password}

                  onChangeText={setPassword}

                  secureTextEntry={!isPasswordVisible}

                  autoCapitalize="none"

                  maxLength={16}

                  autoCorrect={false}

                />

                <TouchableOpacity

                  onPress={() => setIsPasswordVisible((prev) => !prev)}

                  style={styles.eyeIconContainer}

                >

                  <Ionicons

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

                    size={24}

                    color="#333"

                  />

                </TouchableOpacity>

              </View>

            </View>

            <TouchableOpacity onPress={fillTestCredentials} />

            <View style={styles.forgotContainer}>

              <TouchableOpacity

                onPress={() => Alert.alert("Funzione non implementata")}

              >

                <Text style={styles.forgotText}>

                  Hai dimenticato le{" "}

                  <Text style={styles.forgotText2}>credenziali?</Text>

                </Text>

              </TouchableOpacity>

            </View>

          </View>

        </ScrollView>

        <View style={styles.bottomContainer}>

          <Button onPress={handleLogin} loading={loading} text="Continua" />

        </View>

        <View style={styles.bottomContainer}>

          <Button

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

            loading={loading}

            text="Annulla"

            isPressed={isPressed}

            setIsPressed={setIsPressed}

            isCancel={true}

          />

        </View>

      </KeyboardAvoidingView>

    </SafeAreaView>

  );

};

const InputField = ({

  label,

  value,

  onChange,

  placeholder,

  secureTextEntry,

  maxLength,

}) => (

  <View style={styles.inputContainer}>

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

    <TextInput

      style={[styles.input, styles.commonInput]}

      value={value}

      onChangeText={onChange}

      placeholder={placeholder}

      secureTextEntry={secureTextEntry}

      autoCapitalize="none"

      maxLength={maxLength}

      autoCorrect={false}

    />

  </View>

);

const Button = ({

  onPress,

  loading,

  text,

  isPressed,

  setIsPressed,

  isCancel,

}) => (

  <TouchableOpacity

    style={[

      styles.loginButton,

      loading && styles.loginButtonDisabled,

      isCancel && styles.cancelButton,

      isPressed && styles.cancelButtonPressed,

    ]}

    onPress={() => {

      onPress();

      if (setIsPressed) setIsPressed(true);

    }}

    onPressOut={() => setIsPressed && setIsPressed(false)}

    disabled={loading}

  >

    {loading ? (

      <ActivityIndicator color="#fff" />

    ) : (

      <Text

        style={[styles.loginButtonText, isCancel && styles.cancelButtonText]}

      >

        {text}

      </Text>

    )}

  </TouchableOpacity>

);

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f8f9fa",

  },

  commonText: {

    fontSize: 16,

    color: "#333",

  },

  commonInput: {

    fontSize: 16,

    color: "#333",

    borderBottomColor: "#333",

  },

  keyboardView: {

    flex: 1,

  },

  scrollContent: {

    flexGrow: 1,

    paddingHorizontal: 20,

    paddingVertical: 10,

  },

  header: {

    marginBottom: 40,

    paddingTop: 10,

  },

  title: {

    fontSize: 34,

    fontWeight: "bold",

    color: "#1E3A8A",

    textAlign: "left",

  },

  subtitle: {

    fontSize: 18,

    fontWeight: "bold",

    color: "#1E3A8A",

    textAlign: "center",

    marginBottom: 20,

    marginTop: 20,

  },

  form: {

    flex: 1,

  },

  inputContainer: {

    marginBottom: 20,

  },

  label: {

    fontWeight: "600",

  },

  input: {

    height: 50,

    borderBottomWidth: 2,

    paddingBottom: 10,

    paddingHorizontal: 0,

  },

  passwordInput: {

    flex: 1,

    paddingVertical: 0,

  },

  passwordContainer: {

    flexDirection: "row",

    alignItems: "center",

    borderBottomWidth: 2,

    height: 55,

  },

  eyeIconContainer: {

    paddingBottom: 4,

  },

  loginButton: {

    backgroundColor: "#1E3A8A",

    paddingVertical: 15,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    marginBottom: 20,

    elevation: 3,

    shadowColor: "#000",

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

    shadowOpacity: 0.25,

    shadowRadius: 3.84,

  },

  loginButtonDisabled: {

    backgroundColor: "#ccc",

  },

  loginButtonText: {

    color: "#fff",

    paddingBottom: 10,

    fontSize: 16,

  },

  cancelButton: {

    backgroundColor: "#fff",

    borderWidth: 1,

    borderColor: "black",

  },

  cancelButtonPressed: {

    backgroundColor: "#e0e0e0",

  },

  cancelButtonText: {

    color: "#1E3A8A",

    fontWeight: "bold",

  },

  bottomContainer: {

    justifyContent: "flex-end",

    paddingHorizontal: 20,

  },

  forgotContainer: {

    marginBottom: 20,

    alignItems: "center",

  },

  forgotText: {

    ...this?.commonText,

    fontSize: 16,

  },

  forgotText2: {

    ...this?.commonText,

    color: "#0037ceff",

    fontWeight: "600",

  },

});

export default LoginScreen;