MobileApp/lib/bio/bio_reg_create.dart

```dart
import 'dart:async';
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:project_9/utils/constants.dart';
import 'package:project_9/services/api.dart';
import 'package:http/http.dart' as http;
import 'package:path/path.dart' as path;
import 'package:project_9/bio/bio_reg_login.dart';

class BioRegCreatePage extends StatefulWidget {
  final String email;

  const BioRegCreatePage({required this.email});

  @override
  _BioRegCreatePageState createState() => _BioRegCreatePageState();
}

class _BioRegCreatePageState extends State<BioRegCreatePage> {
  CameraController? _controller;
  String? _videoPath;
  bool _isRecording = false;
  String _apiResponse = '';
  String _frase = 'A carregar frase...'; // Label inicial
  CameraDescription? _camera;
  bool _isCameraInitialized = false;
  bool _showSuccessButton = false; // Variável para controlar a exibição do botão

  // Variáveis de estado para as respostas da API
  bool? _faceResult;
  bool? _phraseResult;
  bool? _voiceResult;
  bool? _livenessResult;

  @override
  void initState() {
    super.initState();
    _fetchPhrase();
  }

  Future<void> _initializeCamera() async {
    try {
      final cameras = await availableCameras();
      if (cameras.isNotEmpty) {
        _camera = cameras.firstWhere(
          (camera) => camera.lensDirection == CameraLensDirection.front,
          orElse: () => cameras.first,
        );

        _controller = CameraController(
```

```
          _camera!,
          ResolutionPreset.medium,
        );

        await _controller!.initialize();
        if (!mounted) return;
        setState(() {
          _isCameraInitialized = true;
        });
      } else {
        setState(() {
          _apiResponse = 'No cameras available';
        });
      }
    } catch (e) {
      setState(() {
        _apiResponse = 'Error initializing camera: $e';
      });
    }
  }

  Future<void> _fetchPhrase() async {
    try {
      final data = await API.fetchPhrase();
      setState(() {
        _frase = data['frase'] ?? 'Falha ao obter frase';
      });
    } catch (e) {
      setState(() {
        _frase = 'Erro de conexão à API: $e';
      });
    }
  }

  Future<void> startVideoRecording() async {
    _faceResult = null;
    _phraseResult = null;
    _voiceResult = null;
    _livenessResult = null;
    _apiResponse = "A Capturar...";
    if (!_isCameraInitialized || !_controller!.value.isInitialized) {
      return;
    }

    if (_controller!.value.isRecordingVideo) {
      return;
    }

    try {
      await _controller!.startVideoRecording();
      setState(() {
        _isRecording = true;
      });
    } catch (e) {
```

```
      print(e);
    }
  }

  Future<void> stopRecordingVideo() async {
    if (!_controller!.value.isRecordingVideo) {
      return;
    }

    try {
      final videoFile = await _controller!.stopVideoRecording();
      setState(() {
        _isRecording = false;
        _videoPath = videoFile.path;
        _apiResponse = "A Verificar...";
      });
      processAndUploadVideo(); // Enviar o vídeo após parar a gravação
      setState(() {
        _isCameraInitialized = false;
        _controller = null;
      });
    } catch (e) {
      print(e);
    }
  }

  Future<void> processAndUploadVideo() async {
    if (_videoPath == null || _videoPath!.isEmpty) {
      return;
    }

    String email = widget.email; // Use o email passado como parâmetro
    String url = '$API_ADDRESS/camera_save';
    String videoFile = path.basename(_videoPath!);
    try {
      var request = http.MultipartRequest('POST', Uri.parse(url))
        ..fields['email'] = email
        ..fields['frase'] = _frase
        ..fields['videofile'] = videoFile
        ..files.add(await http.MultipartFile.fromPath(
          'video',
          _videoPath!,
        ));
      var response = await request.send();
      if (response.statusCode == 200) {
      } else {
      }
    } catch (e) {
      print(e);
    }

    // Registar o utilizador
    registerUser(email, videoFile, _frase);
  }
```

```dart

  Future<void> registerUser(String email, String videoFile, String frase) async {
    String url = '$API_ADDRESS/register_user';

    try {
      var response = await http.post(
        Uri.parse(url),
        headers: {"Content-Type": "application/json"},
        body: jsonEncode({"email": email, "videofile": videoFile, "frase": frase}),
      );
      var data = jsonDecode(response.body);

      // Certifique-se de que a resposta contém os campos corretos
      if (response.statusCode == 200 && data['success'] == true) {
        setState(() {
          _faceResult = data['face'] ?? false;
          _phraseResult = data['phrase'] ?? false;
          _voiceResult = data['voice'] ?? false;
          _livenessResult = data['liveness'] ?? false;
          _apiResponse = data['message'] ?? '';
          _showSuccessButton = true;
          _apiResponse = "";// Mostrar o botão de sucesso
          _frase = "";

        });
      } else {
        setState(() {
          _faceResult = data['face'] ?? false;
          _phraseResult = data['phrase'] ?? false;
          _voiceResult = data['voice'] ?? false;
          _livenessResult = data['liveness'] ?? false;
          _apiResponse = data['message'] ?? 'Falha no registo';
        });
      }
    } catch (e) {
      setState(() {
        _apiResponse = 'Erro de conexão à API: $e';
      });
    }
  }

  @override
  void dispose() {
    _controller?.dispose();
    super.dispose();
  }

  Color _getButtonColor(bool? result) {
    if (result == null) {
      return Colors.white;
    } else if (result) {
      return Colors.green;
    } else {
      return Colors.red;
```

```
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Registo Biométrico', style: TextStyle(color:
Colors.black)),
        backgroundColor: Colors.white,
        elevation: 0,
        centerTitle: true,
        iconTheme: const IconThemeData(color: Colors.black),
      ),
      body: Column(
        children: [
          if (_isCameraInitialized && _controller != null &&
_controller!.value.isInitialized)
            Expanded(
              flex: 2,
              child: CameraPreview(_controller!),
            )
          else
            const Expanded(
              flex: 2,
              child: Center(
                child: Image(
                  image: AssetImage('assets/images/face_logo.png'),
                ),
              ),
            ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              _frase,
              style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold,
color: Colors.red),
              textAlign: TextAlign.center,
            ), // Label para mostrar a frase obtida da API
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: ElevatedButton(
              onPressed: _showSuccessButton ? null : () async {
                if (_isRecording) {
                  await stopRecordingVideo();
                } else {
                  await _initializeCamera();
                  await startVideoRecording();
                }
              },
              style: ElevatedButton.styleFrom(
                foregroundColor: Colors.white, backgroundColor: _showSuccessButton ?
Colors.grey : Colors.blueAccent,
```

```
                padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 12),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(8),
                ),
              ),
              child: Text(
                _isRecording ? 'Terminar' : 'Iniciar',
                style: const TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ),
          Expanded(
            flex: 1,
            child: Center(child: Text(_isRecording ? 'A Capturar...' : _apiResponse)),
          ),
          if (_showSuccessButton)
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: ElevatedButton(
                onPressed: () {
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(builder: (context) => BioLoginCreatePage(email: widget.email)),
                  );
                },
                style: ElevatedButton.styleFrom(
                  foregroundColor: Colors.white, backgroundColor: Colors.green,
                  padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 16),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                ),
                child: const Text(
                  'Avançar',
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              TextButton(
                onPressed: null, // Botões desabilitados
                style: TextButton.styleFrom(
                  foregroundColor: Colors.white, backgroundColor: _getButtonColor(_faceResult),
```

```
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                ),
                child: const Text('Face'),
              ),
              TextButton(
                onPressed: null,
                style: TextButton.styleFrom(
                  foregroundColor: Colors.white, backgroundColor: _getButtonColor(_phraseResult),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                ),
                child: const Text('Frase'),
              ),
              TextButton(
                onPressed: null,
                style: TextButton.styleFrom(
                  foregroundColor: Colors.white, backgroundColor: _getButtonColor(_voiceResult),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                ),
                child: const Text('Voz'),
              ),
              TextButton(
                onPressed: null,
                style: TextButton.styleFrom(
                  foregroundColor: Colors.white, backgroundColor: _getButtonColor(_livenessResult),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                ),
                child: const Text('Prova Vida'),
              ),
            ],
          ),
          const Padding(
            padding: EdgeInsets.all(10.0),
            child: Text(
              "",
              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color: Colors.white),
              textAlign: TextAlign.center,
            ), // Label não utilizada
          ),
        ],
      ),
      backgroundColor: Colors.white,
    );
```

```
365     }
366 }
```