

MobileApp/lib/bio/bio_reg_login.dart

```
1 import 'dart:async';
2 import 'dart:convert';
3 import 'package:flutter/material.dart';
4 import 'package:camera/camera.dart';
5 import 'package:project_9/utils/constants.dart';
6 import 'package:project_9/services/api.dart';
7 import 'package:http/http.dart' as http;
8 import 'package:path/path.dart' as path;
9 import 'package:project_9/screens/homepage.dart'; // Certifique-se de importar a
  página inicial
10
11 class BioLoginCreatePage extends StatefulWidget {
12   final String email;
13
14   const BioLoginCreatePage({required this.email});
15
16   @override
17   _BioLoginCreatePageState createState() => _BioLoginCreatePageState();
18 }
19
20 class _BioLoginCreatePageState extends State<BioLoginCreatePage> {
21   CameraController? _controller;
22   String? _videoPath;
23   bool _isRecording = false;
24   String _apiResponse = '';
25   String _frase = 'A carregar frase...'; // Label inicial
26   CameraDescription? _camera;
27   bool _isCameraInitialized = false;
28   bool _showSuccessButton = false; // Variável para controlar a exibição do botão
29
30   // Variáveis de estado para as respostas da API
31   bool? _faceResult;
32   bool? _phraseResult;
33   bool? _voiceResult;
34   bool? _livenessResult;
35
36   @override
37   void initState() {
38     super.initState();
39     _fetchPhrase();
40   }
41
42   Future<void> _initializeCamera() async {
43     try {
44       final cameras = await availableCameras();
45       if (cameras.isNotEmpty) {
46         _camera = cameras.firstWhere(
47           (camera) => camera.lensDirection == CameraLensDirection.front,
48           orElse: () => cameras.first,
49         );
50
51         _controller = CameraController(
```

```
52         _camera!,
53         ResolutionPreset.medium,
54     );
55
56     await _controller!.initialize();
57     if (!mounted) return;
58     setState(() {
59         _isCameraInitialized = true;
60     });
61 } else {
62     setState(() {
63         _apiResponse = 'No cameras available';
64     });
65 }
66 } catch (e) {
67     setState(() {
68         _apiResponse = 'Error initializing camera: $e';
69     });
70 }
71 }
72
73 Future<void> _fetchPhrase() async {
74     try {
75         final data = await API.fetchPhrase();
76         setState(() {
77             _frase = data['frase'] ?? 'Falha ao obter frase';
78         });
79     } catch (e) {
80         setState(() {
81             _frase = 'Erro de conexão à API: $e';
82         });
83     }
84 }
85
86 Future<void> startVideoRecording() async {
87     _faceResult = null;
88     _phraseResult = null;
89     _voiceResult = null;
90     _livenessResult = null;
91     _apiResponse = "A Capturar...";
92     if (!_isCameraInitialized || !_controller!.value.isInitialized) {
93         return;
94     }
95
96     if (_controller!.value.isRecordingVideo) {
97         return;
98     }
99
100    try {
101        await _controller!.startVideoRecording();
102        setState(() {
103            _isRecording = true;
104        });
105    } catch (e) {
```

```
106     print(e);
107   }
108 }
109
110 Future<void> stopRecordingVideo() async {
111   if (!_controller!.value.isRecordingVideo) {
112     return;
113   }
114
115   try {
116     final videoFile = await _controller!.stopVideoRecording();
117     setState(() {
118       _isRecording = false;
119       _videoPath = videoFile.path;
120       _apiResponse = "A Verificar...";
121     });
122     processAndUploadVideo(); // Enviar o vídeo após parar a gravação
123     setState(() {
124       _isCameraInitialized = false;
125       _controller = null;
126     });
127   } catch (e) {
128     print(e);
129   }
130 }
131
132 Future<void> processAndUploadVideo() async {
133   if (_videoPath == null || _videoPath!.isEmpty) {
134     return;
135   }
136
137   String email = widget.email; // Use o email passado como parâmetro
138   String url = '$API_ADDRESS/camera_save';
139   String videoFile = path.basename(_videoPath!);
140   try {
141     var request = http.MultipartRequest('POST', Uri.parse(url))
142       ..fields['email'] = email
143       ..fields['frase'] = _frase
144       ..fields['videofile'] = videoFile
145       ..files.add(await http.MultipartFile.fromPath(
146         'video',
147         _videoPath!,
148       ));
149     var response = await request.send();
150     if (response.statusCode == 200) {
151     } else {
152     }
153   } catch (e) {
154     print(e);
155   }
156
157   // Registrar o utilizador
158   registerUser(email, videoFile, _frase);
159 }
```

```
160
161 Future<void> registerUser(String email, String videoFile, String frase) async {
162   String url = '$API_ADDRESS/validate_user';
163
164   try {
165     var response = await http.post(
166       Uri.parse(url),
167       headers: {"Content-Type": "application/json"},
168       body: jsonEncode({"email": email, "videofile": videoFile, "frase": frase}),
169     );
170     var data = jsonDecode(response.body);
171
172     // Certifique-se de que a resposta contém os campos corretos
173     if (response.statusCode == 200 && data['success'] == true) {
174       setState(() {
175         _faceResult = data['face'] ?? false;
176         _phraseResult = data['phrase'] ?? false;
177         _voiceResult = data['voice'] ?? false;
178         _livenessResult = data['liveness'] ?? false;
179         _apiResponse = data['message'] ?? '';
180         _showSuccessButton = true;
181         _apiResponse = ''; // Mostrar o botão de sucesso
182         _frase = '';
183
184       });
185     } else {
186       setState(() {
187         _faceResult = data['face'] ?? false;
188         _phraseResult = data['phrase'] ?? false;
189         _voiceResult = data['voice'] ?? false;
190         _livenessResult = data['liveness'] ?? false;
191         _apiResponse = data['message'] ?? 'Falha no registo';
192       });
193     }
194   } catch (e) {
195     setState(() {
196       _apiResponse = 'Erro de conexão à API: $e';
197     });
198   }
199 }
200
201 @override
202 void dispose() {
203   _controller?.dispose();
204   super.dispose();
205 }
206
207 Color _getButtonColor(bool? result) {
208   if (result == null) {
209     return Colors.white;
210   } else if (result) {
211     return Colors.green;
212   } else {
213     return Colors.red;
```

```
214     }
215   }
216
217   @override
218   Widget build(BuildContext context) {
219     return Scaffold(
220       appBar: AppBar(
221         title: const Text('Validação Biométrica', style: TextStyle(color:
Colors.black)),
222         backgroundColor: Colors.white,
223         elevation: 0,
224         centerTitle: true,
225         iconTheme: const IconThemeData(color: Colors.black),
226       ),
227       body: Column(
228         children: [
229           if (_isCameraInitialized && _controller != null &&
_controller!.value.isInitialized) {
230             Expanded(
231               flex: 2,
232               child: CameraPreview(_controller!),
233             )
234           else
235             const Expanded(
236               flex: 2,
237               child: Center(
238                 child: Image(
239                   image: AssetImage('assets/images/face_logo.png'),
240                 ),
241               ),
242             ),
243           Padding(
244             padding: const EdgeInsets.all(8.0),
245             child: Text(
246               _frase,
247               style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold,
color: Colors.red),
248               textAlign: TextAlign.center,
249             ), // Label para mostrar a frase obtida da API
250           ),
251           Padding(
252             padding: const EdgeInsets.all(8.0),
253             child: ElevatedButton(
254               onPressed: _showSuccessButton ? null : () async {
255                 if (_isRecording) {
256                   await stopRecordingVideo();
257                 } else {
258                   await _initializeCamera();
259                   await startVideoRecording();
260                 }
261               },
262               style: ElevatedButton.styleFrom(
263                 foregroundColor: Colors.white, backgroundColor: _showSuccessButton ?
Colors.grey : Colors.blueAccent,
```

```

264         padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 12),
265         shape: RoundedRectangleBorder(
266           borderRadius: BorderRadius.circular(8),
267         ),
268       ),
269       child: Text(
270         _isRecording ? 'Terminar' : 'Validar',
271         style: const TextStyle(
272           fontSize: 18,
273           fontWeight: FontWeight.bold,
274         ),
275       ),
276     ),
277   ),
278   Expanded(
279     flex: 1,
280     child: Center(child: Text(_isRecording ? 'A Capturar...' :
_apiResponse)),
281   ),
282   if (_showSuccessButton)
283     Padding(
284       padding: const EdgeInsets.all(8.0),
285       child: ElevatedButton(
286         onPressed: () {
287           Navigator.pushReplacement(
288             context,
289             MaterialPageRoute(builder: (context) => HomePage(email:
widget.email)),
290           );
291         },
292         style: ElevatedButton.styleFrom(
293           foregroundColor: Colors.white, backgroundColor: Colors.green,
294           padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 16),
295           shape: RoundedRectangleBorder(
296             borderRadius: BorderRadius.circular(8),
297           ),
298         ),
299         child: const Text(
300           'Avançar',
301           style: TextStyle(
302             fontSize: 18,
303             fontWeight: FontWeight.bold,
304           ),
305         ),
306       ),
307     ),
308   Row(
309     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
310     children: [
311       TextButton(
312         onPressed: null, // Botões desabilitados
313         style: TextButton.styleFrom(
314           foregroundColor: Colors.white, backgroundColor:
_getButtonColor(_faceResult),

```

```

315         shape: RoundedRectangleBorder(
316           borderRadius: BorderRadius.circular(8),
317         ),
318       ),
319       child: const Text('Face'),
320     ),
321     TextButton(
322       onPressed: null,
323       style: TextButton.styleFrom(
324         foregroundColor: Colors.white, backgroundColor:
325         _getButtonColor(_phraseResult),
326         shape: RoundedRectangleBorder(
327           borderRadius: BorderRadius.circular(8),
328         ),
329         child: const Text('Frase'),
330       ),
331       TextButton(
332         onPressed: null,
333         style: TextButton.styleFrom(
334           foregroundColor: Colors.white, backgroundColor:
335           _getButtonColor(_voiceResult),
336           shape: RoundedRectangleBorder(
337             borderRadius: BorderRadius.circular(8),
338           ),
339           child: const Text('Voz'),
340         ),
341         TextButton(
342           onPressed: null,
343           style: TextButton.styleFrom(
344             foregroundColor: Colors.white, backgroundColor:
345             _getButtonColor(_livenessResult),
346             shape: RoundedRectangleBorder(
347               borderRadius: BorderRadius.circular(8),
348             ),
349             child: const Text('Prova Vida'),
350           ),
351         ],
352       ),
353       const Padding(
354         padding: EdgeInsets.all(10.0),
355         child: Text(
356           '""',
357           style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color:
Colors.white),
358           textAlign: TextAlign.center,
359         ), // Label não utilizada
360       ),
361     ],
362   ),
363   backgroundColor: Colors.white,
364 );

```

```
365 |   }  
366 | }
```