# Guia de Implementação de Recursos Nativos em Flutter

## Sumário



## Configuração Inicial

### pubspec.yaml

```
dependencies:

flutter:

sdk: flutter

geolocator: ^10.0.0

camera: ^0.10.5+5

image_picker: ^1.0.4

permission_handler: ^11.0.1
```

```
path_provider: ^2.1.1

path: ^1.8.3
```

### Estrutura do Projeto

```
lib/

├── services/

| ├── location_service.dart

| └── camera_service.dart

├── screens/

| ├── location_screen.dart

| └── camera_screen.dart

└── main.dart
```

## Serviço de Localização

### location_service.dart

```
import  'package:geolocator/geolocator.dart';

import  'package:permission_handler/permission_handler.dart';


class  LocationService {

// Solicita permissão de localização
```

```
Future<bool> requestLocationPermission() async {
final status =  await  Permission.location.request();
return status.isGranted;
}

// Obtém localização atual
Future<Position> getCurrentLocation() async {
bool serviceEnabled =  await  Geolocator.isLocationServiceEnabled();
if (!serviceEnabled) {
throw  Exception('Serviços de localização desativados');
}

LocationPermission permission =  await  Geolocator.checkPermission();
if (permission ==  LocationPermission.denied) {
permission =  await  Geolocator.requestPermission();
if (permission ==  LocationPermission.denied) {
throw  Exception('Permissão de localização negada');
}
}

if (permission ==  LocationPermission.deniedForever) {
throw  Exception('Permissões de localização permanentemente negadas');
```

```
}


return  await  Geolocator.getCurrentPosition(

desiredAccuracy:  LocationAccuracy.high,

);

}


// Stream de atualizações de localização

Stream<Position> getLocationStream() {

return  Geolocator.getPositionStream(

locationSettings:  LocationSettings(

accuracy:  LocationAccuracy.high,

distanceFilter:  10, // Atualiza a cada 10 metros

),

);

}

}
```

## Serviço de Câmera

### camera_service.dart

```dart
import 'package:camera/camera.dart';
import 'package:image_picker/image_picker.dart';
import 'package:path_provider/path_provider.dart';
import 'package:path/path.dart' as path;
import 'dart:io';

class CameraService {
CameraController? controller;
List<CameraDescription> cameras = [];

// Inicialização da câmera
Future<void> initialize() async {
final status = await Permission.camera.request();
if (!status.isGranted) {
throw Exception('Permissão da câmera negada');
}

cameras = await availableCameras();
if (cameras.isEmpty) {
throw Exception('Nenhuma câmera encontrada');
}
```

```
controller =  CameraController(

cameras[0],

ResolutionPreset.high,

enableAudio:  false,

);



await controller!.initialize();

}



// Tirar foto

Future<String> takePicture() async {

if (controller ==  null  ||  !controller!.value.isInitialized) {

throw  Exception('Câmera não inicializada');

}



final  Directory appDir =  await  getApplicationDocumentsDirectory();

final  String dirPath =  '${appDir.path}/Photos';

await  Directory(dirPath).create(recursive:  true);



final  String filePath = path.join(

dirPath,

'${DateTime.now().millisecondsSinceEpoch}.jpg',
```

```
);

try {
XFile photo =  await controller!.takePicture();
await photo.saveTo(filePath);
return filePath;
} catch (e) {
throw  Exception('Erro ao tirar foto: $e');
}
}

// Selecionar imagem da galeria
Future<String?> pickImageFromGallery() async {
final  ImagePicker picker =  ImagePicker();
final  XFile? image =  await picker.pickImage(
source:  ImageSource.gallery,
maxWidth:  1800,
maxHeight:  1800,
);
if (image !=  null) {
return image.path;
}
```

```
return  null;

}


void  dispose() {

controller?.dispose();

}

}
```

# Telas

## location_screen.dart

```
class  LocationScreen  extends  StatefulWidget {

@override

_LocationScreenState  createState() =>  _LocationScreenState();

}


class  _LocationScreenState  extends  State<LocationScreen> {

final  LocationService _locationService =  LocationService();

Position? _currentPosition;

StreamSubscription<Position>? _positionStreamSubscription;
```

```
@override

void  initState() {

super.initState();

_setupLocationUpdates();

}


Future<void> _setupLocationUpdates() async {

try {

await _locationService.requestLocationPermission();

_currentPosition =  await _locationService.getCurrentLocation();

setState(() {});


_positionStreamSubscription = _locationService

.getLocationStream()

.listen((Position position) {

setState(() {

_currentPosition = position;

});

});

} catch (e) {

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content:  Text('Erro: $e')),
```

```
);
}
}

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

@override
Widget  build(BuildContext context) {
return  Scaffold(
appBar:  AppBar(title:  Text('GPS Real')),
body:  Center(
child:  Column(
mainAxisAlignment:  MainAxisAlignment.center,
children: [
if (_currentPosition !=  null) ...[
Text(
'Localização Atual:',
style:  Theme.of(context).textTheme.titleLarge,
```

```
),
SizedBox(height:  20),
Text(
'Latitude: ${_currentPosition!.latitude.toStringAsFixed(6)}',
style:  Theme.of(context).textTheme.bodyLarge,
),
Text(
'Longitude: ${_currentPosition!.longitude.toStringAsFixed(6)}',
style:  Theme.of(context).textTheme.bodyLarge,
),
Text(
'Altitude: ${_currentPosition!.altitude.toStringAsFixed(2)}m',
style:  Theme.of(context).textTheme.bodyLarge,
),
Text(
'Velocidade: ${_currentPosition!.speed.toStringAsFixed(2)}m/s',
style:  Theme.of(context).textTheme.bodyLarge,
),
] else
CircularProgressIndicator(),
],
),
```

```
),
);
}
}
```

## camera_screen.dart

```
class CameraScreen extends StatefulWidget {
@override
_CameraScreenState createState() => _CameraScreenState();
}

class _CameraScreenState extends State<CameraScreen> {
final CameraService _cameraService = CameraService();
List<String> _photos = [];
bool _isInitialized = false;

@override
void initState() {
super.initState();
_initializeCamera();
}
```

```
Future<void> _initializeCamera() async {

try {

await _cameraService.initialize();

setState(() {

_isInitialized =  true;

});

} catch (e) {

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content:  Text('Erro ao inicializar câmera: $e')),

);

}

}



Future<void> _takePicture() async {

try {

final  String photoPath =  await _cameraService.takePicture();

setState(() {

_photos.add(photoPath);

});

} catch (e) {

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content:  Text('Erro ao tirar foto: $e')),
```

```
);
}
}

@override
void  dispose() {
_cameraService.dispose();
super.dispose();
}

@override
Widget  build(BuildContext context) {
if (!_isInitialized) {
return  Scaffold(
appBar:  AppBar(title:  Text('Câmera')),
body:  Center(child:  CircularProgressIndicator()),
);
}

return  Scaffold(
appBar:  AppBar(
title:  Text('Câmera Real'),
```

```
actions: [
IconButton(
icon:  Icon(Icons.photo_library),
onPressed: _pickFromGallery,
),
],
),
body:  Column(
children: [
Expanded(
flex:  2,
child:  CameraPreview(_cameraService.controller!),
),
Expanded(
flex:  1,
child: _photos.isEmpty
?  Center(child:  Text('Nenhuma foto tirada'))
:  GridView.builder(
gridDelegate:  SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount:  3,
crossAxisSpacing:  4,
mainAxisSpacing:  4,
```

```
),
itemCount: _photos.length,
itemBuilder: (context, index) {
return  Image.file(
File(_photos[index]),
fit:  BoxFit.cover,
);
},
),
),
],
),
floatingActionButton:  FloatingActionButton(
onPressed: _takePicture,
child:  Icon(Icons.camera),
),
floatingActionButtonLocation:  FloatingActionButtonLocation.centerFloat,
);
}
}
```

## Configurações Nativas

### Android (android/app/src/main/AndroidManifest.xml)

```
<!-- Adicione dentro da tag <manifest> -->
<uses-permission  android:name="android.permission.ACCESS_FINE_LOCATION"
<uses-permission  android:name="android.permission.ACCESS_COARSE_LOCATIOI
<uses-permission  android:name="android.permission.CAMERA"  />
<uses-permission  android:name="android.permission.WRITE_EXTERNAL_STORAGI
<uses-permission  android:name="android.permission.READ_EXTERNAL_STORAGE
```

### iOS (ios/Runner/Info.plist)

```
<!-- Adicione dentro da tag <dict> -->
<key>NSLocationWhenInUseUsageDescription</key>
<string>Este aplicativo precisa de acesso à localização para fornecer fui
<key>NSLocationAlwaysUsageDescription</key>
<string>Este aplicativo precisa de acesso à localização para fornecer fui
<key>NSCameraUsageDescription</key>
<string>Este aplicativo precisa de acesso à câmera para tirar fotos</str
<key>NSPhotoLibraryUsageDescription</key>
<string>Este aplicativo precisa de acesso à galeria para selecionar fotos
```

## Dicas de Implementação

1. **Gerenciamento de Permissões**:

- Sempre verifique as permissões antes de usar recursos nativos
- Forneça feedback claro ao usuário quando permissões forem negadas
- Implemente um fluxo para solicitar permissões novamente se necessário

2. **Uso de Recursos**:

- Lembre-se de liberar recursos no dispose()
- Gerencie o ciclo de vida dos controllers adequadamente
- Considere o consumo de bateria ao usar GPS continuamente

3. **Tratamento de Erros**:

- Implemente tratamento de erros robusto
- Forneça feedback adequado ao usuário
- Considere diferentes estados do dispositivo (GPS desligado, sem câmera, etc.)

4. **Otimizações**:

- Ajuste a precisão do GPS conforme necessidade
- Otimize a qualidade das fotos baseado no uso
- Implemente cache de imagens quando apropriado

5. **Testes**:

- Teste em diferentes dispositivos
- Verifique diferentes versões do Android e iOS
- Teste cenários de permissões negadas