Roteiro de Aula Prática: Recursos Nativos no Flutter (Zapp.run)

Nível: Intermediário

Ambiente: https://zapp.run

Objetivos

- Compreender a implementação de recursos nativos em Flutter
- Desenvolver uma aplicação que simule o uso de GPS e câmera
- Aprender sobre gerenciamento de estados e interfaces responsivas

Parte 1: Configuração do Ambiente

1.1 Preparação

1. Acesse https://zapp.run
2. Clique em “New Project”
3. Escolha “Flutter”

1.2 Configuração do pubspec.yaml

```
dependencies:

flutter:

sdk: flutter
```

```
cupertino_icons: ^1.0.2

provider: ^6.0.5

shared_preferences: ^2.2.2



dev_dependencies:

flutter_test:

sdk: flutter

flutter_lints: ^2.0.0
```

Parte 2: Estrutura do Projeto

2.1 Organização de Arquivos

Criar a seguinte estrutura no projeto:

```
lib/

├── models/

| ├── location_model.dart

| └── photo_model.dart

├── services/

| ├── location_service.dart

| └── camera_service.dart

├── screens/

| ├── home_screen.dart
```

```
| ├── location_screen.dart
| └── camera_screen.dart
└── main.dart
```

Parte 3: Implementação dos Modelos

3.1 Location Model

```dart
// lib/models/location_model.dart
class LocationData {
final double latitude;
final double longitude;
final String timestamp;

LocationData({
required this.latitude,
required this.longitude,
required this.timestamp,
});

Map<String, dynamic> toJson() {
return {
```

```
'latitude': latitude,

'longitude': longitude,

'timestamp': timestamp,

};

}


factory  LocationData.fromJson(Map<String, dynamic> json) {

return  LocationData(

latitude: json['latitude'],

longitude: json['longitude'],

timestamp: json['timestamp'],

);

}

}
```

3.2 Photo Model

```
// lib/models/photo_model.dart

class  PhotoData {

final  String id;

final  String path;

final  String timestamp;

final  LocationData? location;
```

```
PhotoData({

required  this.id,

required  this.path,

required  this.timestamp,

this.location,

});



Map<String, dynamic> toJson() {

return {

'id': id,

'path': path,

'timestamp': timestamp,

'location': location?.toJson(),

};

}



factory  PhotoData.fromJson(Map<String, dynamic> json) {

return  PhotoData(

id: json['id'],

path: json['path'],

timestamp: json['timestamp'],
```

```
location: json['location'] != null
? LocationData.fromJson(json['location'])
: null,
);
}
}
```

Parte 4: Implementação das Telas

4.1 Home Screen

```
// lib/screens/home_screen.dart
import 'package:flutter/material.dart';

class HomeScreen extends StatelessWidget {
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text('Recursos Nativos Demo'),
),
body: Center(
```

```
child:  Column(
mainAxisAlignment:  MainAxisAlignment.center,
children: [
ElevatedButton(
onPressed: () {
Navigator.pushNamed(context, '/location');
},
child:  Text('GPS Simulado'),
),
SizedBox(height:  20),
ElevatedButton(
onPressed: () {
Navigator.pushNamed(context, '/camera');
},
child:  Text('Câmera Simulada'),
),
],
),
),
);
}
}
```

4.2 Location Screen (Simulação de GPS)

```
// lib/screens/location_screen.dart

import 'package:flutter/material.dart';

import 'dart:math';


class LocationScreen extends StatefulWidget {

@override

_LocationScreenState createState() => _LocationScreenState();

}


class _LocationScreenState extends State<LocationScreen> {

double _latitude = 0.0;

double _longitude = 0.0;


void _simulateLocationUpdate() {

// Simula mudança de localização

setState(() {

_latitude += Random().nextDouble() * 0.001 * (Random().nextBool() ?

_longitude += Random().nextDouble() * 0.001 * (Random().nextBool() ?

});

}
```

```
@override

Widget  build(BuildContext context) {

return  Scaffold(

appBar:  AppBar(

title:  Text('GPS Simulado'),

),

body:  Center(

child:  Column(

mainAxisAlignment:  MainAxisAlignment.center,

children: [

Text(

'Localização Atual:',

style:  Theme.of(context).textTheme.titleLarge,

),

SizedBox(height:  20),

Text(

'Latitude: ${_latitude.toStringAsFixed(6)}',

style:  Theme.of(context).textTheme.bodyLarge,

),

Text(

'Longitude: ${_longitude.toStringAsFixed(6)}',
```

```
style:  Theme.of(context).textTheme.bodyLarge,

),

SizedBox(height:  30),

ElevatedButton(

onPressed: _simulateLocationUpdate,

child:  Text('Atualizar Localização'),

),

],

),

),

);

}

}
```

4.3 Camera Screen (Simulação de Câmera)

```
// lib/screens/camera_screen.dart

import  'package:flutter/material.dart';

import  'dart:math';


class  CameraScreen  extends  StatefulWidget {

@override

_CameraScreenState  createState() =>  _CameraScreenState();
```

```
}

class _CameraScreenState extends State<CameraScreen> {
List<String> _photos = [];

void _simulateTakePhoto() {
setState(() {
// Simula uma foto tirada gerando um ID único
_photos.add('Photo_${DateTime.now().millisecondsSinceEpoch}');
});
}

@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text('Câmera Simulada'),
),
body: Column(
children: [
Expanded(
child: _photos.isEmpty
```

```
? Center(child: Text('Nenhuma foto tirada'))
: GridView.builder(
gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount: 2,
crossAxisSpacing: 10,
mainAxisSpacing: 10,
),
itemCount: _photos.length,
padding: EdgeInsets.all(10),
itemBuilder: (context, index) {
return Container(
color: Colors.grey[300],
child: Center(
child: Text(
'Foto ${index + 1}',
style: Theme.of(context).textTheme.titleMedium,
),
),
);
},
),
),
```

```
Padding(

padding:  EdgeInsets.all(16.0),

child:  ElevatedButton(

onPressed: _simulateTakePhoto,

child:  Text('Tirar Foto'),

),

),

],

),

);

}

}
```

Parte 5: Exercícios Práticos

5.1 Exercícios

1. Adicionar histórico de localizações
2. Implementar filtro de fotos por data
3. Criar visualização em mapa (pode ser simulada com um grid)
4. Adicionar sistema de favoritos

Desafios Extras

1. Implementar tema escuro/claro
2. Adicionar animações nas transições
3. Criar filtros para as “fotos”
4. Implementar persistência de dados usando shared_preferences

Dicas de Desenvolvimento no Zapp.run

- Use o console do navegador (F12) para debug
- Salve seu trabalho frequentemente
- Teste em diferentes tamanhos de tela usando o redimensionamento do preview
- Utilize o hot reload para ver alterações rapidamente