Programação Para Dispositivos Móveis

Aula 10

Paulo Tomé

Escola Superior de Tecnologia e Gestão de Viseu

ptome@estgv.ipv.pt

Ano letivo 2024-2025

Sumário

Armazenamento de dados de forma persistente

- O armazenamento de dados pode ser realizado em:
  - Bases de Dados;
  - Ficheiros;
  - Shared Preferences;
  - Cloud.

Bases de Dados

- O armazenamento de dados numa Base de Dados no dispositivo é uma opção frequente das aplicações.

- Não há necessidade de conetividade permanente (dados ou rede sem fios).

- Bibliotecas de disponíveis para utilizar em Flutter:
  - SQFLite;
  - Hive;
  - Floor;
  - Sembast (NoSQL);
  - Drift (Moor);
  - ObjectBox (NoSQL);
  - sqlite3;
  - Isar.

Guardar ficheiros em DART

- Os ficheiros de uma aplicação podem ser guardados na pasta de ficheiros da aplicação. Poderão utilizar-se pastas para organizar os ficheiros.

- A função seguinte guarda o conjunto de dados passados como parâmetro num ficheiro com a designação estabelecida pelo segundo parâmetro.

```
...
import 'dart:io';
...
Future<void> guardafic(dados,fich) async {
      final dir = await getTemporaryDirectory();
      var nomefich = '${dir.path}/$fich';
      final file = File(nomefich);
      await file.writeAsBytes(dados);
      }
```

Shared Preferences

- As *Shared Preferences* são armazenadas na forma de atributo/valor.

- Tem de se adicionar a dependência *shared_preferences*:

```
dependencies:
  ..
  shared_preferences:
```

- Tem de se realizar o *import*:

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

- Podem ser armazenados: Strings, inteiros, reais, booleanos ou listas de strings.

- Exemplos:

```
await prefs.setString('utilizador', 'Paulo');
await prefs.setInt('idade', 25);
await prefs.setDouble('altura', 1.8);
await prefs.setBool('segunda', true);
await prefs.setStringList('diasdasemana', ['segunda', 'terça', '...']);
```

Shared Preferences

- Os métodos set permitem registar.
- Os métodos get permitem ler. Exemplo:

```
String? log = prefs.getString('utilizador');
```

Shared Preferences

- Classe com um método para registar e outro para ler:

```
import 'package:shared_preferences/shared_preferences.dart';
class Sharpref
{
Future<void> registaUt(lo,pa) async {
  final SharedPreferences prefs = await SharedPreferences.getInstance();

  await prefs.setString('login', lo);
  await prefs.setString('pass', pa);
}
Future<String> lerUt(lo,pa) async {
  final SharedPreferences prefs = await SharedPreferences.getInstance();

  String? log = prefs.getString('login');
  String? pas = prefs.getString('pass');
  if (log == lo && pas == pa)
  {
    return log!;
  }
  else
  {
    return('Utilizador não autenticado');
  }
  }

}
```

Shared Preferences

- Pode-se apagar um atributo através do método *remove*:

```
Future<void> remove(atr) async {
  final SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.remove(atr);
}
```

- Podem-se apagar todos os atributos:

```
Future<void> removetodas() async {
  final SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.clear();
}
```

Transferência de ficheiros em DART

- Uma das formas de transferir dados de ficheiros de uma API é através da devolução por esta do conteúdo em bytes.
- O atributo *bodyBytes* permite ter acesso ao conteúdo em bytes dos dados devolvidos pela API. Exemplo:

```
final http.Response response = await http.get(Uri.parse(lista[i]['imagem']));
guardafic(response.bodyBytes,'fich1.jpg');
```

Ligação com autenticação a uma API (Consulta)

- Vulgarmente o acesso a dados de API pressupõe a utilização de algumas regras de segurança. Uma das regras poderá passar pelo registo e utilização de um código de acesso.

- O departamento de Informática disponibiliza uma API com autenticação:

  - URL: https://apoiodi.estgv.ipv.pt/ws_pdm/
  - nmec= abcde;
  - token = ((a+b+c+d+e)+abcde)*2025

Ligação com autenticação a uma API (Consulta)

- O acesso para consulta pode fazer-se através de:

```
Future<void> rece_list(context) async {
var headers = {
   'Cookie': 'PHPSESSID=mv490gtafp2di20rptvp6d69ro'
 };
var request = http.MultipartRequest('POST',
Uri.parse(url));
 request.fields.addAll({
'nmec': '12345',
 'token': '25016640',
 'imagens': ''
 });
 request.headers.addAll(headers);
 http.StreamedResponse response = await request.send();

 if (response.statusCode == 200) {
    res = await response.stream.bytesToString();
 }
 else {
 print(response.reasonPhrase);
 }
}
```

Ligação com autenticação a uma API (Upload)

- O acesso para upload pode fazer-se através de:

```
Future<void> envio_fich(fich) async
{
  var headers = {
 'Cookie': 'PHPSESSID=mv490gtafp2di20rptvp6d69ro'
 };
 var request = http.MultipartRequest('POST',Uri.parse(url));
 request.fields.addAll({
 'nmec': '12345',
 'token': '25016640',
 'nome': 'xxxxx',
 'imagem': fich
 });

 request.headers.addAll(headers);
 http.StreamedResponse response = await request.send();
 if (response.statusCode == 200) {
 print(await response.stream.bytesToString());
 }
 else {
 print(response.reasonPhrase);
 }
}
```

- A função anterior envia o ficheiro recebido como parâmetro para a API.

Ligação com autenticação a uma API (Upload)

- A função é invocada da forma seguinte:

```
//código colocado no botão de registo de foto
final image = await _controller.takePicture();
final bytes = Io.File(image.path).readAsBytesSync();
String img64 = base64Encode(bytes);
final Servidor se = Servidor();
se.envio_fich(img64);
```

Encriptação de dados em Flutter

- O Flutter tem uma biblioteca para encriptar/desencriptar dados com suporte dos seguintes algoritmos:
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
  - SHA-512/224
  - SHA-512/256
  - MD5
  - HMAC (i.e. HMAC-MD5, HMAC-SHA1, HMAC-SHA256)

- Tem de se adicionar a dependência:

```
dependencies:
    ...
    crypto:
```

- Tem que se fazer o *import*:

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

Encriptação de dados em Flutter

- A encriptação faz-se:

```
final enc = algoritmo.convert(valor);
```

- Podendo o algoritmo ser:
  - sha1
  - sha256
  - hmacSha256
  - md5
  - ...

- Exemplo de método que encripta para md5:

```
String criaMd5(String valor) {
  return md5.convert(utf8.encode(valor)).toString();
}
```

Tarefa

- Elabore uma aplicação em:
  - No método de ”arranque” da aplicação registe numa *shared preference* registe um utilizador com lgin e password. Tenha em consideração que a password tem de ser codificada.
  - Tenha uma *route* em que o utilizador faz login e mostre, através de um *widget* assíncrono, se o utilizador o fez com sucesso ou não.