**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**Chủ đề: SharedPreferences trong Flutter**

**Sinh viên thực hiện:**

1. **Đặng Quang Lâm Lớp:22KTMT2**  **MSSV:**  106220256
2. **Nguyễn Hữu Nam** **Lớp: 22KTMT2** **MSSV:**  106220262

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

**THUYẾT MINH**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**Chủ đề: Shared Preferences trong Flutter**

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | Đặng Quang Lâm | Tìm hiểu lý thuyết về tổng quan widget Image & Icon, so sánh NetworkImage vs AssetImage, cách cấu hình assets trong, tìm hiểu các thuộc tính Image properties, chuẩn bị slide và báo cáo… | 50% |
| 02 | Nguyễn Hữu Nam | Tìm hiểu lý thuyết về tổng quan widget Image & Icon, So sánh Material Icons và Cupertino Icons, tìm hiểu kỹ thuật xử lý lỗi/loading, demo code, chuẩn bị slide và báo cáo... | 50% |

 Link code github: [github](https://github.com/quanglam19082004/SharedPreferences)

**Mục lục**

[**1.** **Định nghĩa SharedPreferences** 4](#_Toc212485623)

[**1.1.** **Định nghĩa** 4](#_Toc212485624)

[**1.2.** **Cài đặt và sử dụng SharedPreferences** 4](#_Toc212485625)

[**2.** **Nguyên lý hoạt động** 4](#_Toc212485626)

[**3.** **Ứng dụng lưu trữ cài đặt dark mode, font size** 5](#_Toc212485627)

[**4.** **SharedPreferences với các phương thức lưu trữ khác** 7](#_Toc212485628)

[**5. Các lỗi thường gặp** 8](#_Toc212485629)

[**5.1. Lỗi đọc/ghi không đồng bộ (Async)** 8](#_Toc212485630)

[**5.2. Lỗi dữ liệu chưa tồn tại (null)** 9](#_Toc212485631)

[**5.3. Lỗi ghi đè nhầm key** 9](#_Toc212485632)

[**6. Cách thực hiện tốt nhất** 10](#_Toc212485633)

[**6.1. Khởi tạo SharedPreferences đúng chỗ** 10](#_Toc212485634)

[**6.2. Gói chức năng lưu trữ vào Service riêng** 10](#_Toc212485635)

[**6.3. Không lưu dữ liệu nhạy cảm** 11](#_Toc212485636)

[**6.4. Sử dụng kiểu dữ liệu phù hợp** 11](#_Toc212485637)

1. **Định nghĩa SharedPreferences:**
   1. **Định nghĩa:**

SharedPreferences là một cơ chế lưu trữ dữ liệu dạng key-value (cặp khóa–giá trị) trên thiết bị.

Thích hợp cho các dữ liệu đơn giản như: cài đặt người dùng (dark mode, font size...), token, trạng thái đăng nhập...

Đặc điểm chính:

* Đơn giản: Dễ sử dụng với API trực quan
* Nhẹ: Không phù hợp cho dữ liệu lớn
* Không đồng bộ: Hỗ trợ async/await
* Platform-specific: Mỗi nền tảng có cách lưu trữ riêng
  1. **Cài đặt và sử dụng SharedPreferences:**- Trong file pubspect.yaml:  
      dependencies:  
      shared preferences: ^2.2.2

- Chạy lệnh: flutter pub get để cập nhật các gói.

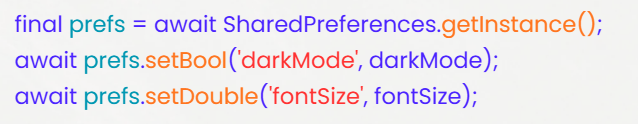
- Import:

import 'package:shared\_preferences/shared\_preferences.dart' ;

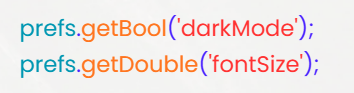
1. **Nguyên lý hoạt động**

Gồm 5 bước:

Bước 1:Flutter gọi hàm set / get

****

Nếu cần đọc lại, sẽ gọi:

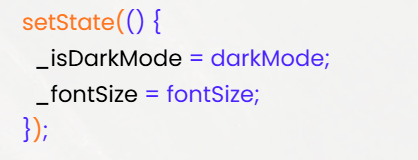
****

Bước 2:Plugin (SharedPreferences) xử lí yêu cầu và lưu tạm trong cache, gửi lệnh qua Method Channel

Bước 3:Native Platform ghi dữ liệu xuống file hệ thống

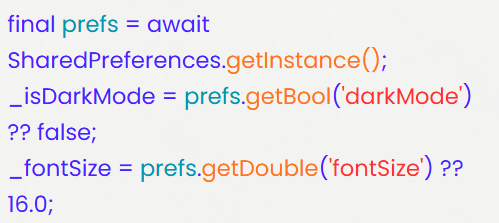
Bước 4:Dữ liệu được lưu trong thư mục: data/data/<app>/shared\_prefs/

Bước 5:SharedPreferences ghi thành công, Flutter cập nhật giao diện

****

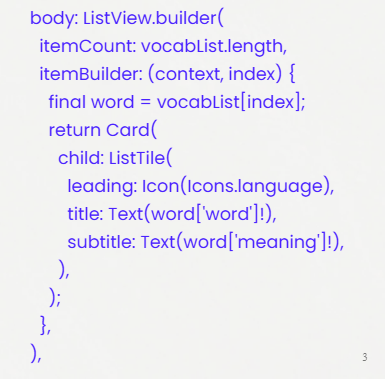
1. **Ứng dụng lưu trữ cài đặt dark mode, font size:**Gồm 4 file:  
   File: main.dart - Entry point của ứng dụng
   * Là điểm bắt đầu của chương trình Flutter (void main()).
   * Khởi tạo EnglishVocabApp – toàn bộ ứng dụng.
   * Quản lý theme (Dark/Light) và font size của toàn app.
   * Lưu và tải lại các thiết lập của người dùng bằng SharedPreferences.

Code:

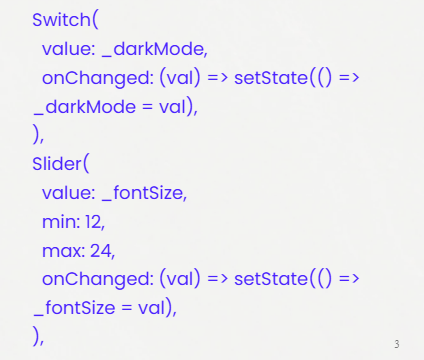


File: home\_screen.dart - Màn hình chính

* + Là màn hình chính của ứng dụng học từ vựng.
  + Duyệt qua danh sách từ (vocabList) và hiển thị mỗi từ trong Card có icon và màu sắc.
  + Có nút ⚙️ để mở màn hình cài đặt (Settings).
  + Nhận và áp dụng lại thay đổi theme / font khi người dùng quay lại.

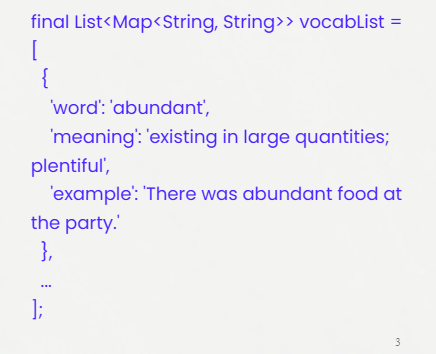
Code:  
   
 File: settings\_screen.dart - Màn hình cài đặt

* + Cho phép người dùng bật/tắt chế độ Dark Mode.
  + Điều chỉnh cỡ chữ (font size) bằng Slider.
  + Khi người dùng nhấn “Save Settings”, dữ liệu được gửi ngược lại về main.dart để lưu bằng SharedPreferences.



File: vocab\_data.dart - Dữ liệu từ vựng

* + Chứa danh sách các từ vựng mẫu để hiển thị trong app.
  + Mỗi từ gồm: word, meaning, example.
  + Dạng dữ liệu: List<Map<String, String>>.



1. **SharedPreferences với các phương thức lưu trữ khác**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phương thức** | **Mô tả** | **Ưu điểm** | **Hạn chế** | **Dùng khi** |
| SharedPreferences | Lưu key-value nhỏ | Nhanh, dễ dùng, đa nền tảng | Không lưu được object phức tạp | Cài đặt, token, chế độ |
| File Storage (dart:io) | Lưu file text hoặc JSON | Tùy chỉnh linh hoạt | Cần tự quản lý đọc/ghi, parse dữ liệu | Lưu nhiều dữ liệu cấu trúc |
| SQLite (sqflite) | CSDL quan hệ cục bộ | Lưu dữ liệu lớn, có truy vấn | Cần thiết kế bảng, phức tạp hơn | Dữ liệu dạng danh sách, quản lý |
| Hive / ObjectBox | CSDL NoSQL native | Hiệu năng cao, lưu object dễ | Thêm thư viện, hơi nặng | Lưu dữ liệu có cấu trúc phức tạp, offline app |

# **5. Các lỗi thường gặp:**

## **5.1. Lỗi đọc/ghi không đồng bộ (Async):**

- SharedPreferences hoạt động bất đồng bộ (async) - việc đọc/ghi dữ liệu cần thời gian I/O, cần await

- Nếu không dùng await, lệnh tiếp theo có thể chạy trước khi dữ liệu được tải hoặc lưu xong

- Lỗi xảy ra:

* Dữ liệu đọc ra là null (do chưa kịp load)
* Việc ghi chưa hoàn tất nhưng line code khác đã chạy => nhận giá trị cũ

- Ví dụ:

+ Block code 1:

*void saveSetting(bool darkMode) {*

*final prefs = SharedPreferences.getInstance();*

*prefs.setBool('darkMode', darkMode);*

*}*

=> Không có await, có thể gây mất đồng bộ hoặc lỗi giá trị null

+ Block code 2:

*Future<void> saveSetting(bool darkMode) async {*

*final prefs = await SharedPreferences.getInstance();*

*await prefs.setBool('darkMode', darkMode);*

*}*

=> Cách sử dụng đúng, có await chờ thời gian I/O cho quá trình đọc/ghi dữ liệu

## **5.2. Lỗi dữ liệu chưa tồn tại (null):**

- Khi gọi *prefs.getX('key')*, nếu key đó chưa từng được lưu => phương thức sẽ trả về giá trị null

- Nếu ta sử dụng dữ liệu này ngay (biến bool, int hay hiển thị giao diện) => Crash hoặc lỗi ứng dụng

- Cách giải quyết:

+ Dùng toán tử ?? để cung cấp giá trị mặc định

Ví dụ: *bool darkMode = prefs.getBool('darkMode') ?? false;*

+ Kiểm tra key trước khi đọc

Ví dụ:

*if (prefs.containsKey('darkMode')) {*

*darkMode = prefs.getBool('darkMode')!;*

*} else {*

*darkMode = false;*

*}*

=> Cách đơn giản, tường minh và có phần “kỹ tính” nhưng cực kỳ chắc chắn về mặt logic code

+ Đặt mặc định khi tạo ứng dụng

Ví dụ: *prefs.setBool('darkMode', false);*

## **5.3. Lỗi ghi đè nhầm key:**

- Các phần khác nhau trong ứng dụng sử dụng cùng tên key để lưu dữ liệu

- SharedPreferences không có namespace hoặc phân vùng, nên nếu 2 module cùng ghi "user", giá trị sau sẽ ghi đè giá trị trước

- Không có cảnh báo => khó phát hiện lỗi, đặc biệt khi debug

- Cách giải quyết:

+ Quy ước đặt tên key rõ ràng, có prefix theo chức năng

Ví dụ: *await prefs.setString('auth\_userToken', 'token\_ABC');*

*await prefs.setString('profile\_userName', 'Nguyễn Văn A');*

+ Tạo class quản lý key tập trung

Ví dụ: *class PrefKeys {*

*static const userToken = 'auth\_userToken';*

*static const userName = 'profile\_userName';*

*static const darkMode = 'settings\_darkMode';*

*}*

=> Giúp dễ tra cứu, tránh gõ nhầm, dễ bảo trì khi app mở rộng

# **6. Cách thực hiện tốt nhất:**

## **6.1. Khởi tạo SharedPreferences đúng chỗ:**

- Trong Flutter, SharedPreferences.getInstance() là một thao tác bất đồng bộ (async) và chỉ nên gọi một lần duy nhất trong quá trình khởi động ứng dụng

- Việc khởi tạo sai chỗ - ví dụ gọi trong nhiều widget - có thể dẫn đến hiệu năng kém, lag UI, hoặc lỗi truy cập dữ liệu chưa khởi tạo

- Cách làm đúng là tạo một bước khởi tạo sớm (init) trong main() hoặc trong lớp service riêng:

*void main() async {*

*WidgetsFlutterBinding.ensureInitialized();*

*await PrefsService.instance.init();*

*runApp(MyApp());*

*}*

*=>* Đảm bảo toàn bộ dữ liệu trong SharedPreferences được nạp sẵn và có thể truy cập ở mọi nơi trong ứng dụng mà không gây trễ

## **6.2. Gói chức năng lưu trữ vào Service riêng:**

- Không nên thao tác SharedPreferences trực tiếp ở nhiều màn hình khác nhau, code sẽ bị trùng lặp, khó kiểm soát key và khó bảo trì khi muốn thay đổi logic lưu trữ

- Tạo một lớp duy nhất chịu trách nhiệm tương tác với SharedPreferences.

- Lớp này đóng vai trò trung gian - UI hoặc ViewModel không cần biết cách lưu trữ, chỉ cần gọi phương thức có ý nghĩa

- Ưu điểm cách làm này:

* Tập trung - dễ bảo trì
* Dễ mở rộng & tái sử dụng
* Code gọn hơn, dễ đọc hơn
* Dễ kiểm thử (Unit Test)

Ví dụ:

*class PrefsService {*

*static final PrefsService instance = PrefsService.\_internal();*  *PrefsService.\_internal();*

*late SharedPreferences \_prefs;*

*//Khởi tạo một lần trong main()*

*Future init() async { \_*

*prefs = await SharedPreferences.getInstance();*

*}*

*//Các phương thức lưu trữ*

*Future setDarkMode(bool value) async {*

*await \_prefs.setBool('dark\_mode', value);*

*}*

*bool getDarkMode() {*

*return \_prefs.getBool('dark\_mode') ?? false;*

*}*

*Future setFontSize(double size) async {*

*await \_prefs.setDouble('font\_size', size);*

*}*

*double getFontSize() {*

*return \_prefs.getDouble('font\_size') ?? 14.0;*

*}*

*}*

## **6.3. Không lưu dữ liệu nhạy cảm:**

- SharedPreferences chỉ lưu dữ liệu dưới dạng plain text (không mã hóa)

- Điều này nghĩa là bất kỳ ai truy cập được file lưu trữ đều có thể đọc nội dung, vì vậy ta không nên lưu các thông tin như token, mật khẩu, hoặc dữ liệu cá nhân quan trọng

- Nếu cần lưu trữ dữ liệu quan trọng, có thể sử dụng flutter\_secure\_storage - dùng Keychain (iOS) / Android Keystore (Android). Thư viện này lưu dữ liệu an toàn hơn bằng cách mã hóa trên hệ thống

## **6.4. Sử dụng kiểu dữ liệu phù hợp:**

- SharedPreferences chỉ hỗ trợ các kiểu dữ liệu cơ bản:

* int
* double
* bool
* String
* List<String>

- Nên chọn kiểu phù hợp với mục đích lưu trữ - ví dụ:

* Lưu trạng thái dark mode => bool
* Lưu cỡ chữ => double
* Lưu danh sách chủ đề yêu thích => List<String>

**TÀI LIỆU THAM KHẢO**

[1]. Luan Dang, “Tìm hiểu về SharedPreferences trong Flutter: Lưu trữ dữ liệu cục bộ đơn giản”, Viblo, 2025.

[2]. Pham Xuan Lu, “Shared Preferences trong Android”, Viblo, 2018.

[3]. Developers, “Lưu trữ dữ liệu đơn giản bằng Shared Preferences”, Developers, 2025.

[4]. Flutter.dev, “Shared preferences plugin”, pub.dev, 2025

[5]. Kteam, “Lưu trữ dữ liệu với SharedPreferences”, Kteam.