**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**JSON Serialization và Model Classes**

**Sinh viên thực hiện:**

**01.Ngô Trung Chinh Lớp: 22KTMT1**  **MSSV:** 106220211

**02. Nguyễn Hữu Duy**  **Lớp: 22KTMT1 MSSV:**  106220215

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

**THUYẾT MINH**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**JSON Serialization và Model Classes**

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 11 | Ngô Trung Chinh | - Nghiên cứu yêu cầu và nguyên lý hoạt động của JSON Serialization trong Flutter. - Thiết kế các model class chính - Xử lý dữ liệu lồng nhau (nested objects) và mảng (arrays). - Viết và kiểm thử các phương thức fromJson() và toJson() cho từng model. | 50% |
| 15 | Nguyễn Hữu Duy | - Cài đặt và cấu hình các thư viện json\_annotation, build\_runner. - Tạo và quản lý code sinh tự động bằng json\_serializable. - Thực hiện kiểm thử quá trình serialize/deserialize với dữ liệu mẫu. - Ứng dụng best practices trong việc quản lý model phức tạp, tổ chức thư mục | 50% |

 Link code github: <https://github.com/huuduy26/LTDNT-DUYCHINH-10>

**Mục lục**

**NỘI DUNG**

**1. Cơ sở lý thuyết** ............................................................................................................. 3  
1.1. JSON là gì? ................................................................................................................. 3  
1.2. JSON Serialization trong Dart/Flutter ........................................................................ 3  
1.3. Tại sao cần JSON Serialization .................................................................................. 4

**2. Tạo Model Classes với toJson() và fromJson()** ......................................................... 4  
2.1. Khái niệm Model Class............................................................................................... 4  
2.2. Cấu trúc của một Model Class .................................................................................... 5  
2.3. Ví dụ cơ bản và kết quả .............................................................................................. 5

**3. Xử lý Nested Objects và Arrays** ................................................................................. 5  
3.1. Nested Objects (Đối tượng lồng nhau) ....................................................................... 6  
3.2. Ví dụ minh họa ........................................................................................................... 6  
3.3. Làm việc với danh sách (List / Array) ........................................................................ 7  
3.4. Một số lỗi thường gặp ................................................................................................. 7

**4. Sử dụng json\_annotation và build\_runner** ................................................................ 8  
4.1. Cấu trúc hoạt động ...................................................................................................... 8  
4.2. Các bước thực hiện ..................................................................................................... 9

**5. Best Practices cho xử lý dữ liệu phức tạp trong Flutter** ........................................... 9  
5.1. Nguyên tắc thiết kế Model hiệu quả ........................................................................... 9  
5.2. Sử dụng Annotation nâng cao trong JSON ............................................................... 10  
5.3. Chiến lược xử lý Null-safety ..................................................................................... 10  
5.4. Tối ưu hiệu suất serialize / deserialize ...................................................................... 10

**NỘI DUNG**

# 1. Cơ sở lý thuyết

## 1.1 JSON là gì?

* **JSON (JavaScript Object Notation)** là một dạng định dạng dữ liệu có cấu trúc nhẹ, giúp trao đổi thông tin giữa các hệ thống một cách dễ dàng. Dữ liệu trong JSON được tổ chức dưới dạng các cặp *key–value*, rất dễ đọc và xử lý bởi con người cũng như máy tính.
* JSON được ứng dụng phổ biến trong các hệ thống web và mobile, đặc biệt là khi truyền dữ liệu qua **API** giữa **client** và **server**.

Cấu trúc JSON bao gồm:

**Object:** Tập hợp các cặp *key–value:*

*A screen shot of a computer code

AI-generated content may be incorrect.*

**Array:** Danh sách giá trị, ví dụ:

A close-up of a name

AI-generated content may be incorrect.

Đặc điểm của JSON:

## Có thể sử dụng với mọi ngôn ngữ lập trình.

## Dạng text nên dễ truyền qua mạng.

## Cấu trúc rõ ràng, dễ dàng chuyển đổi sang các dạng dữ liệu khác.

## 1.2. JSON Serialization trong Dart/Flutter

* JSON Serialization là quá trình chuyển đổi qua lại giữa dữ liệu JSON và đối tượng Dart trong ứng dụng Flutter.
* Nói cách khác, đây là cầu nối giúp Flutter **hiểu và làm việc với dữ liệu** mà backend gửi đến.

Quá trình này bao gồm hai hướng chính:

+ Serialization: Là việc chuyển đổi một đối tượng (object) trong Dart thành chuỗi JSON để có thể:

* + Gửi dữ liệu đó lên server thông qua API.
  + Lưu trữ dữ liệu cục bộ (local storage, file, SharedPreferences, v.v.).
  + Giao tiếp giữa các tiến trình hoặc lưu cache tạm thời.

+ Deserialization: Là việc chuyển đổi dữ liệu JSON (thường là chuỗi string nhận từ server) trở lại thành đối tượng Dart (model class), để có thể:

* + Hiển thị thông tin lên giao diện (UI).
  + Thao tác, tính toán, hoặc xử lý dữ liệu.
  + Lưu trữ vào cơ sở dữ liệu nội bộ.

## 1.3. Tại sao cần JSON Serialization

# Trong các ứng dụng Flutter hiện đại, dữ liệu trả về từ backend gần như luôn ở dạng chuỗi JSON. Vì Dart không thể thao tác trực tiếp với chuỗi này, ta cần chuyển nó thành object Dart để tiện xử lý, truy cập và kiểm soát kiểu dữ liệu.

# Ưu điểm của việc dùng JSON Serialization:

# Giúp code dễ hiểu, dễ bảo trì.

# Tăng độ an toàn kiểu dữ liệu (*type safety*).

# Thuận tiện trong việc mở rộng, tái sử dụng, kiểm thử.

# Dễ dàng tích hợp với API và database.

# 2.   Tạo model classes với toJson() và fromJson()

## 2.1. Khái niệm Model Class

**Model Class** (hay *Data Model*) là một lớp Dart mô tả cấu trúc dữ liệu của đối tượng mà ứng dụng làm việc với.  
Nó là cầu nối giữa dữ liệu dạng JSON và các đối tượng cụ thể trong ứng dụng Flutter.

## 2.2. Cấu trúc của một Model Class

## Một model class thông thường bao gồm:

## Fields (thuộc tính): Đại diện cho dữ liệu (thường dùng final).

## Constructor: Khởi tạo đối tượng.

## fromJson(): Hàm chuyển JSON → Object.

## toJson(): Hàm chuyển Object → JSON.

## 2.3. Ví dụ cơ bản và kết quả

A screen shot of a computer program

AI-generated content may be incorrect.

Kết quả:

A white background with black text

AI-generated content may be incorrect.

# 3. Xử lý nested objects và arrays

## Trong thực tế, dữ liệu JSON không chỉ chứa giá trị đơn giản mà còn có thể chứa đối tượng lồng nhau (nested objects) hoặc mảng đối tượng (list/array). Đây là dạng dữ liệu phổ biến trong các API phức tạp.

## 3.1. Nested Objects (Đối tượng lồng nhau)

Đây là tình huống khi một object có chứa object khác làm thuộc tính con, ví dụ:

A computer code with text

AI-generated content may be incorrect.

## Ở đây, address là một object con. Khi deserialize, ta cần tạo một class riêng Address và gọi Address.fromJson() trong User.fromJson() để xử lý.

## 3.2. Ví dụ minh họa

|  |  |
| --- | --- |
|  |  |

Kết quả:

A close up of a text

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Bước** | **Mô tả** |
| 1 | Nhận dữ liệu JSON có chứa object con. |
| 2 | Trong fromJson() của class cha, gọi tiếp fromJson() của class con. |
| 3 | Sau khi parse, có thể truy cập user.address.city hoặc user.address.street. |
| 4 | Khi xuất lại JSON, toJson() của class cha gọi toJson() của class con. |

## 3.3. Làm việc với danh sách (List / Array)

Array (hoặc List) trong JSON là một danh sách gồm nhiều phần tử, mỗi phần tử có thể là:

* Giá trị đơn giản (int, String, bool, …)
* Hoặc object phức tạp (ví dụ danh sách Order, Product, User, …)

Trong Dart, ta biểu diễn bằng kiểu List<T> — trong đó T là kiểu dữ liệu của từng phần tử.

## Khi JSON chứa danh sách đối tượng:

## {

## "orders": [

## {"id": 1, "price": 100000},

## {"id": 2, "price": 200000}

## ]

## }

## Xử lý trong Dart:

## orders = (json['orders'] as List)

## .map((item) => Order.fromJson(item))

## .toList();

**3.4. Một số lỗi thường gặp**

|  |  |  |
| --- | --- | --- |
| **Lỗi** | **Nguyên nhân** | **Cách khắc phục** |
| Null check operator used on a null value | JSON thiếu object hoặc danh sách | Dùng nullable hoặc kiểm tra null trước |
| type 'List' is not a subtype of type 'List' | Không ép kiểu đúng khi map dữ liệu | Dùng .map((e) => Model.fromJson(e)).toList() |
| String is not a subtype of int | Sai kiểu dữ liệu | Kiểm tra kiểu từng trường hoặc ép kiểu |
| NoSuchMethodError: '[]' was called on null | Key không tồn tại | Kiểm tra bằng containsKey() |
| Dữ liệu không đồng nhất | JSON trả về sai cấu trúc | Kiểm tra response thực tế trước khi parse |

# 4. Sử dụng json\_annotation và build\_runner

Khi số lượng model tăng, việc tự viết fromJson() và toJson() cho từng class sẽ tốn thời gian và dễ nhầm lẫn.  
Dart cung cấp công cụ **tự sinh code** thông qua các thư viện:

|  |  |
| --- | --- |
| **Thư viện** | **Vai trò** |
| json\_annotation | Cung cấp annotation cho model |
| json\_serializable | Sinh mã tự động |
| build\_runner | Thực thi việc sinh mã |

Các thư viện này giúp phân tích cấu trúc model và tự động sinh code fromJson() và toJson() một cách chuẩn xác, đồng nhất, và hiệu năng cao.

## 4.1 Cấu trúc hoạt động

## Model Class + @JsonSerializable()

## ↓

## json\_serializable

## ↓

## Sinh file .g.dart tự động

## ↓

## Có sẵn fromJson() và toJson()

## 4.2 Các bước thực hiện

**Bước 1: Thêm vào pubspec.yaml các package cần thiết.**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**Bước 2: Tạo model và thêm annotation**

**A computer screen shot of a code

AI-generated content may be incorrect.**

**Bước 3: Chạy lệnh sinh code**

**“dart run build\_runner build”**

**File book.g.dart sẽ chứa:**

A screen shot of a computer code

AI-generated content may be incorrect.

# 5. Best Practices cho Complex Data Structures trong Flutter

## 5.1. Nguyên tắc thiết kế Model

## Mỗi model chỉ nên mô tả một thực thể dữ liệu duy nhất.

## Tận dụng model con trong nhiều model cha để giảm trùng lặp.

## Đặt tên class rõ ràng, dễ hiểu (ví dụ: UserInfo, OrderDetail).

## Ưu tiên các thuộc tính final để dữ liệu bất biến.

## 5.2. Sử dụng Annotation nâng cao

Các annotation trong json\_serializable giúp xử lý dữ liệu JSON linh hoạt hơn:

* @JsonKey(name: 'field\_name') → ánh xạ key JSON khác với tên biến.
* ignore → bỏ qua trường không cần serialize.
* defaultValue → gán giá trị mặc định.
* nullable → cho phép giá trị null.

**Ví dụ:**

A screen shot of a computer code

AI-generated content may be incorrect.

## 5.3. Chiến lược xử lý Null-safety

## Dùng ? cho biến có thể null: String? email;

## Dùng ?? để gán giá trị mặc định:

## print(user.email ?? "Không có email");

## Dùng late nếu chắc chắn giá trị sẽ được gán sau.

## 5.4. Tối ưu hiệu suất serialize / deserialize

* Tránh decode JSON nhiều lần.
* Dùng const constructors khi có thể.
* Hạn chế nested JSON quá sâu.
* Ưu tiên json\_serializable để tối ưu tốc độ và độ chính xác.

**TÀI LIỆU THAM KHẢO**

[1] Flutter, *“JSON and serialization,”* Flutter Documentation, 2025.

[2] Dart Dev Team, *“dart:convert library,”* Dart SDK Documentation, Google Developers, 2025.

[3] Google Developers, *“Flutter JSON serialization with json\_serializable,”* Codelab, Google, 2025.

[4] M. Smith and L. Johnson, *“Data Modeling and Serialization in Flutter Applications,”* *Journal of Mobile Development*, vol. 8, no. 2, pp. 45–53, 2023.