**Coding Standard**

**Fitness app - Group C**

**Member:**

*1959019 - Phạm Lê Hoàng Minh*

*2159001 - Võ Quang Dũng*

*2159011 - Nguyễn Ngọc Phú*

***TABLE OF CONTENT***

[**1. Thiết kế module 1**](#_5bwu28pd74c7)

[**2. Cách tổ chức source code 1**](#_637hyoi36iws)

[**3. Quy cách code 3**](#_96zw3fiosnq3)

[a. Quy cách đặt tên 3](#_hgbeizcfq47g)

[b. Quy cách viết code 4](#_ie4zb7pj8jfv)

[**4. Handle error và Logging 5**](#_mk8cj1uvhw4e)

[**5. Tool và IDE: 6**](#_fc58guboichz)

# Thiết kế module

* ***Nguyên tắc SRP (Single Responsibility Principle):*** *Mỗi module chỉ nên có một trách nhiệm duy nhất, rõ ràng và dễ hiểu.*
* ***Sử dụng mô hình MVC hoặc MVVM:*** *Giúp phân tách rõ ràng giữa các lớp model, view và controller, tăng tính modularity và khả năng test.*
* ***Thiết kế API module rõ ràng:*** *Xác định rõ các API của module, bao gồm các hàm, lớp và cấu trúc dữ liệu.*

# Cách tổ chức source code

* **Sử dụng thư mục hợp lý:** Sắp xếp source code theo chức năng, component hoặc feature.
* **Sử dụng tên thư mục và file mô tả:** Đặt tên thư mục và file theo cách dễ hiểu, phản ánh nội dung bên trong.
* **Sử dụng IDE hỗ trợ sắp xếp code:** Sử dụng các tính năng sắp xếp code tự động của IDE để giữ cho code luôn gọn gàng.
* Tách code thành các thư mục khác nhau dựa trên chức năng hoặc tính năng, giúp điều hướng và định vị các tệp cụ thể dễ dàng hơn.

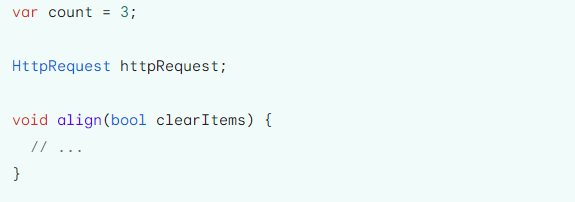


*Ví dụ: các pages sẽ nằm trong thư mục pages, và mỗi page sẽ có 2 file là page\_model.dart và page\_widget.dart.*

# Quy cách code

## Quy cách đặt tên

* **Đặt tên biến/hàm phải đi liền với ý nghĩa sử dụng**: string uusername = “Simon” , user có name là “Simon”, function getExercises(string execref) - phương thức lấy thông tin của bài tập từ đường dẫn reference.
* **Tên hàm nên là động từ và tên biến là danh từ:** DataType exercises, function getDates()...
* **Các constant sẽ** lowerCamel**Case toàn bộ, mỗi chữ cách nhau bởi dấu** gạch dưới “\_”:
* daysInWeek={“Mon”,”Tue”,”Wed”,”Thu”,”Fri”,”Sat”,”Sun”}
* Sử dụng camelCase cho biến, hàm và method: userName, getUserData, saveWorkout.
* Sử dụng PascalCase cho class và interface: UserData, ApiService.
* **Các biến tạm, hay đặt tên rõ ràng, ko sử dụng temp.**
* Sử dụng SCREAMING\_SNAKE\_CASE cho constants: API\_URL, MAX\_WORKOUT\_DURATION.
* Sử dụng snake\_case cho tên file: user\_data.dart, api\_service.dart.
* Sử dụng lowercase\_with\_underscores cho name import:
  + Ví dụ:
    - import 'dart:math' as math;
    - import 'package:angular\_components/angular\_components.dart' as angular\_components;
    - import 'package:js/js.dart' as js;
* Sử dụng lowerCamelCase cho tên cho các identifiers: Class members, top-level definitions, variables, parameters, and named parameters nên được đặt tên viết hoa chữ cái đầu tiên của mỗi từ ngoại trừ từ đầu tiên và không sử dụng dấu phân cách.
  + Ví dụ:



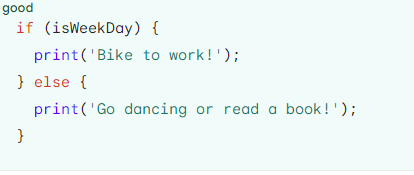
## 

## Quy cách viết code

* Giữa các tên biến, value, property và các kí tự “=”, “:”,.. phải có khoảng trắng: exerName = “Squat”
* Mỗi "phần" phải được phân tách bằng một dòng trống:



* Sử dụng code formatter tự động để định dạng code theo quy cách thống nhất: có thể dùng [dart format](https://dart.dev/tools/dart-format).
* Xem xét việc thay đổi mã của bạn để làm cho mã thân thiện hơn với người định dạng:
  + Ví dụ: rút ngắn tên biến cục bộ hoặc đưa một biểu thức vào một biến cục bộ mới.
* Hạn chế viết dòng code dài hơn 80 từ: khó theo dõi và quản lý.
* Sử dụng dấu ngoặc nhọn cho tất cả các câu lệnh điều khiển luồng:
  + Ví dụ:



* Tách code phức tạp thành các hàm nhỏ, dễ quản lý và test hơn.
* Sử dụng unit test để kiểm tra code, viết unit test cho các phần code quan trọng để đảm bảo chất lượng code.

# Handle error và Logging

* Handle error:
* Triển khai các cơ chế xử lý lỗi thích hợp, chẳng hạn như try-catch blocks, để xử lý các ngoại lệ và lỗi.
* Sử dụng ErrorWidget hoặc FlutterError.onError của Flutter để cung cấp khả năng xử lý lỗi nhẹ nhàng và hiển thị các thông báo lỗi có ý nghĩa cho người dùng.
* Logging:
  + Ghi log lỗi đầy đủ, bao gồm nguyên nhân lỗi, thời gian xảy ra lỗi và stack trace.
  + Sử dụng logging framework phù hợp để quản lý và theo dõi log hiệu quả.

Ghi log là để lưu lại quá trình thực thi của chương trình.

- Nếu ghi log cho lỗi (thường là lỗi runtime) thì khi chương trình có

lỗi có thể nhìn được lỗi gì, ở đoạn code nào và dễ dàng trong

việc tìm phương án xử lý. Nhiều trường hợp chương trình gặp

lỗi trên môi trường production mà không thể tái hiện lại được,

khi đó sẽ xem log để tìm nguyên nhân.

- Nếu ghi log cho quá trình sử dụng ứng dụng, sẽ biết user nào

vào truy cập vào ứng dụng và thao tác gì, từ đó tìm được thủ

phạm của những tác nhân gậy hại đến ứng dụng

- Nếu ghi log ở các đoạn code phức tạp, các tiến trình chạy ngầm,

sẽ biết được trạng thái của tiến trình đó đang chạy như thế nào,

biết được tốc độ xử lý của các nghiệp vụ phức tạp, có cơ sở để

tìm ra các đoạn code gây cao tải cho hệ thống…

Các cách ghi log:

* Print() ra console:
* Sử dụng Flutter’s logging library: Flutter cung cấp một thư viện logging riêng biệt cho phép bạn ghi log thông tin ở nhiều mức độ khác nhau, bao gồm debug, info, warning, error và fatal.
  + Ví dụ:



* Sử dụng logging packages: Ngoài thư viện logging của Flutter, bạn cũng có thể sử dụng các logging packages của bên thứ ba như logging hoặc pretty\_logger. Các packages này cung cấp nhiều tính năng nâng cao hơn như định dạng log tùy chỉnh, ghi log ra file,...
* Lưu ý:
  + Nên sử dụng mức log phù hợp với thông tin bạn muốn ghi. Ví dụ: sử dụng debug cho thông tin chi tiết, info cho thông tin chung, warning cho các vấn đề tiềm ẩn và error cho lỗi.
  + Nên ghi log thông tin đầy đủ bao gồm thời gian, tên class, tên method và thông tin lỗi (nếu có).
  + Nên sử dụng logging framework để quản lý và theo dõi log hiệu quả.

# Tool và IDE:

* Sử dụng IDE hỗ trợ code như Visual Studio Code, IntelliJ IDEA hoặc WebStorm để hỗ trợ viết code, refactor code và debug.
* Sử dụng Firebase/Supabase để quản lý dữ liệu.
* Sử dụng code formatter tự động để định dạng code theo quy cách thống nhất.
* Sử dụng static code analyzer để phát hiện lỗi tiềm ẩn trong code trước khi compile.
* Sử dụng version control system như Git để quản lý phiên bản code và cộng tác hiệu quả.