**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ - VIỄN THỐNG**

🙚  🙘

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**ĐỀ TÀI:**

**Future và Async/Await trong Flutter**

|  |  |
| --- | --- |
| **Sinh viên thực hiện:** | **Lê Xuân Nam (106220226)**  **Nguyễn Nam Phương (106220266)** |
| **Lớp học phần:** | **22.44** |
| **Lớp sinh hoạt:** | **22KTMT1-22KTMT2** |

**Giảng Viên Hướng Dẫn: TS.Nguyễn Duy Nhật Viễn**

🙚  🙘

**Đà Nẵng, 28/10/2025**

**Mục lục**

1.Future………………………………………………………………………………3

2.Async/await​………………………………………………………………………...4

3.FutureBuilder​………………………………………………………………………5

4.Error Handling với try-catch​……………………………………………………….7

5.Kết hợp nhiều Future với Future.wait()……………………………………………8

6.Kết luận…………………………………………………………………………….11

**BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | Lê Xuân Nam | Error handling với try-catch  Combine multiple Futures với Future.wait() | 50% |
| 02 | Nguyễn Nam Phương | Future, async/await concepts  Demo FutureBuilder widget | 50% |

 Link code github: <https://github.com/leemalek4/LTDNT_11>

1. **Future**

Là 1 đối tượng đại diện cho kết quả của hoạt động bất đồng bộ sẽ có trong tương lai​

Ví dụ: Khi đặt hàng, cửa hàng sẽ hứa giao hàng sau 3 phút, sau 3 phút thì nhận được món hàng​. Lời hứa của cửa hàng được coi là là 1 Future, món hàng nhận được là kết quả của Future đó

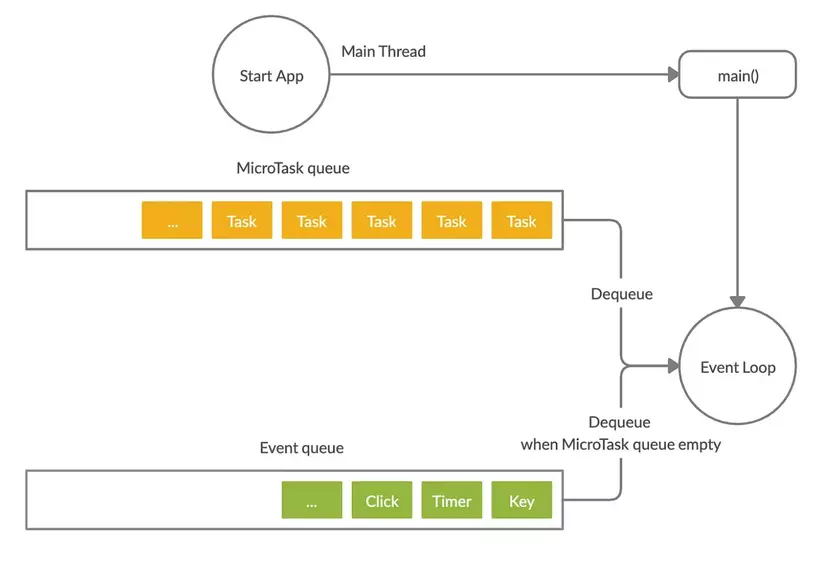
- Future có 2 trạng thái chính:​

+ Chưa hoàn thành: Future vừa được khởi tạo, hoạt động đang được thực hiện ​

+ Hoàn thành: Trả về giá trị hoặc lỗi​

Dart sử dụng Event Loop để quản lý các tác vụ bất đồng bộ bao gồm các tác vụ liên quan tới Future

Mô hình EventLoop:



Khi ứng dụng khởi động từ Start App qua Main Thread và hàm main(), luồng chính được kích hoạt để khởi chạy toàn bộ ứng dụng. Event Loop đóng vai trò trung tâm, điều phối hai hàng đợi quan trọng: Microtask Queue, nơi xử lý các tác vụ ưu tiên như callback của Future và Event Queue, nơi chứa các sự kiện như click từ người dùng, timer hết hạn, hay dữ liệu từ API trả về. Khi một Future được thực thi, nó gửi tín hiệu qua Event Queue sau khi tác vụ hoàn tất, sau đó Event Loop nhanh chóng chuyển các bước xử lý tiếp theo, như callback .then(), sang Microtask Queue để đảm bảo thực thi ngay lập tức, giữ cho giao diện người dùng luôn mượt mà. Sự phối hợp nhịp nhàng giữa hai hàng đợi này giúp ứng dụng xử lý bất đồng bộ hiệu quả, không gián đoạn.



Trong đoạn code trên, hàm tenHocSinh là 1 Future trả về chuỗi “Nguyen Van A”, trong hàm main, lệnh gọi tenHocSinh được gắn với .then để hiển thị kết quả, dòng "tuoi: 16" được in trực tiếp

Khi chạy, "tuoi: 16" hiện lên trước vì nó chạy ngay lập tức. Sau đó, khi Future hoàn tất, "Nguyen Van A" xuất hiện nhờ xử lý bất đồng bộ trong .then.

1. **Async/Await**

- Async là một từ khóa trong Dart dùng để đánh dấu một hàm là bất đồng bộ ​

- Hàm async luôn trả về một Future, ngay cả khi không khai báo rõ ràng​

- Ví dụ:

Hàm tenHocSinh có từ khóa async sẽ luôn trả về 1 Future mặc dù ko khai báo trả về Future

- Await là từ khóa dùng bên trong hàm async để tạm dừng thực thi hàm cho đến khi một Future hoàn thành, lấy giá trị từ Future đó.​

 - Ví dụ:​



Khi lấy giá trị của hàm tenHocSinh qua **await** luồng thực thi trong main tạm dừng, chờ Future từ hàm tenHocSinh hoàn thành, rồi mới tiếp tục in "Nguyen Van A". Ngay sau đó, "tuoi: 16" được in theo thứ tự code.

1. **Future Builder**

- Là widget giúp xây dựng UI dựa trên kết quả của 1 Future​

- Tự động cập nhật giao diện khi Future thay đổi trạng thái​

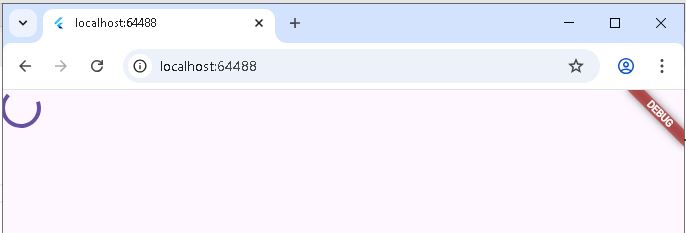
- Giúp xử lý các trạng thái UI ( loading, error, empty, content) một cách dễ dàng

Code demo FutureBuilder:

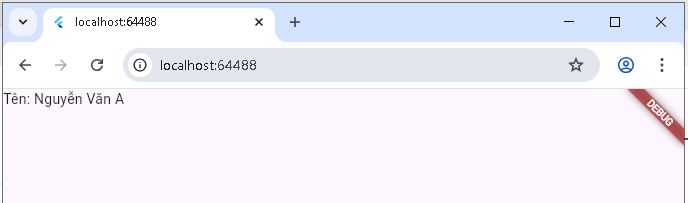


Hàm tenHocSinh được đánh dấu async, sử dụng Future.delayed để mô phỏng một tác vụ bất đồng bộ kéo dài 2 giây, rồi trả về chuỗi "Nguyen Van A". Bên trong runApp, FutureBuilder được dùng để xây dựng giao diện

Kết quả:



FutureBuilder nhận một future là tenHocSinh() và một builder function để định hình giao diện. Trong builder, kiểm tra snapshot.connectionState để nhận biết trạng thái hiện tại. Khi mới khởi động, ConnectionState.waiting được kích hoạt, và giao diện hiển thị CircularProgressIndicator (biểu tượng loading tròn), cho thấy ứng dụng đang chờ dữ liệu.



Sau 2 giây, khi Future hoàn thành, snapshot.hasData trở thành true, và giao diện chuyển sang hiển thị "Tên: Nguyen Van A".

1. **Error Handling với try–catch.**

Trong các ứng dụng Flutter, phần lớn thao tác với dữ liệu diễn ra bất đồng bộ (asynchronous): gọi API, đọc/ghi file, truy vấn cơ sở dữ liệu, v.v. Điều này dẫn đến hai nhu cầu bắt buộc: (1) xử lý lỗi an toàn để ứng dụng không bị crash khi Future thất bại; (2) tối ưu thời gian tải bằng cách kết hợp nhiều Future chạy song song

Lỗi bất đồng bộ (Exception) có thể xảy ra mọi lúc: timeout, mất mạng, dữ liệu sai định dạng… → cần try–catch để không crash.

Độ trễ lớn khi phải gọi nhiều tác vụ độc lập → nên dùng Future.wait() để chạy song song và chỉ chờ tác vụ chậm nhất.

**4.1 Mục đích**

Ngăn crash khi Future thất bại; cho phép phản hồi có kiểm soát (thông báo, log, retry).

Giữ UI ổn định: cập nhật trạng thái “đã bắt lỗi” thay vì treo/đứng.

**4.2 Cách khai báo và cách hoạt động**

*Khai báo:*

**A close up of words

AI-generated content may be incorrect.**

*Cách hoạt động:*

*B1: Vào try, chờ await.*

*B2:* *Nếu Future trả value → chạy tiếp trong try.*

*B3:* *Nếu Future ném Exception → nhảy sang catch*

**4.3 Ví dụ minh hoạ**

*Hàm giả lập lỗi :*

A computer code with colorful text

AI-generated content may be incorrect.

*Xử lý khi ấn nút:*

*A screen shot of a computer code

AI-generated content may be incorrect.*

*Kết quả:*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.*

1. **Kết hợp nhiều Future với Future.wait()​.**

**5.1 Mục đích.**

Chạy song song các tác vụ độc lập để giảm thời gian chờ tổng thể.

Chỉ cập nhật UI khi có đủ dữ liệu (tất cả Future hoàn thành).

**5.2 Cách khai báo và cách hoạt động**

*Cách khai báo:*

**

**5.3 Ví dụ minh hoạ**

*A screen shot of a computer code

AI-generated content may be incorrect.*

*Kết quả :*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

*Sau ~2 giây, hiển thị “User: Nam | Age: 36”. Trên màn hình   
So với chạy tuần tự (≈3 giây), Future.wait() nhanh hơn rõ rệt.*

1. **Kết Luận**

*Future* đại diện cho kết quả bất đồng bộ. Sử dụng .then() hoặc async/await để xử lý kết quả.

*Async/Await* làm cho code bất đồng bộ dễ đọc, giống như code đồng bộ. Luôn đi kèm với Future.

*FutureBuilder* là Widget mạnh mẽ để xây dựng UI dựa trên trạng thái của Future. Kiểm tra connectionState, hasData, và hasError để xử lý các trường hợp.

*Error handling* với *try–catch* là lớp an toàn bắt buộc khi làm với Future → giúp ứng dụng không crash, phản hồi có kiểm soát, dễ bảo trì.

*Future.wait()* giúp tối ưu thời gian bằng cách chạy song song các tác vụ độc lập → cải thiện trải nghiệm người dùng.

**Tài liệu tham khảo**

*[1] Flutter Documentation – “Asynchronous programming: futures, async, await,” dart.dev.*​  
*A*vailable *:* [*https://dart.dev/libraries/async/async-await*](https://dart.dev/libraries/async/async-await)​

*[2] Flutter Documentation – “Error handling,” dart.dev.*​  
*A*vailable*:* [*https://dart.dev/language/error-handling*](https://dart.dev/language/error-handling)​

*[3] Danish Sharma – “Efficiently Handle Concurrent Tasks in Flutter with Future.wait,” Medium, 2023.*​  
*A*vailable*:* [*https://medium.com/@danishsharma13255/efficiently-handle-concurrent-tasks-in-flutter-with-future-wait-7221076fcff0*](https://medium.com/@danishsharma13255/efficiently-handle-concurrent-tasks-in-flutter-with-future-wait-7221076fcff0)​

*[4] Jhon Munoz – “How to Use Future.wait in Dart for Better Futures Handling,” Medium, 2022.*​  
*A*vailable*:* [*https://medium.com/@jhonmunozromero/how-to-use-future-wait-in-dart-for-better-futures-handling-0e180e65ef22*](https://medium.com/@jhonmunozromero/how-to-use-future-wait-in-dart-for-better-futures-handling-0e180e65ef22)​

*[5] TS.Nguyễn Duy Nhật Viễn – “Tài liệu hướng dẫn Flutter (Slide bộ môn ),” Trường Đại học Bách Khoa Đại Học Đà Nẵng, 2025.*​  
*A*vailable*:* [*https://drive.google.com/file/d/1PMQm5DOBg1ZM6Z6BWeE5G6vBzOg6-4Ib/view*](https://drive.google.com/file/d/1PMQm5DOBg1ZM6Z6BWeE5G6vBzOg6-4Ib/view)​