**Họ tên; Nguyễn Phương Nga**

**Lớp: 08\_THMT**

**MSSV: 0850080033**

**Lý thuyết buổi 9**

1. **Storage**

"Storage" trong lập trình và công nghệ thông tin là quá trình lưu trữ và quản lý dữ liệu hoặc thông tin trên các thiết bị điện tử như máy tính, điện thoại di động, máy tính bảng và các thiết bị lưu trữ khác.

Điều này có thể bao gồm lưu trữ dữ liệu tạm thời hoặc lưu trữ dữ liệu lâu dài để sử dụng sau này. Storage cũng ám chỉ đến các phương tiện vật lý hoặc kỹ thuật số được sử dụng để lưu trữ dữ liệu, chẳng hạn như ổ cứng, bộ nhớ flash, thẻ nhớ và dịch vụ lưu trữ đám mây.

Trong ngữ cảnh của ứng dụng di động và phát triển ứng dụng, "storage" thường liên quan đến việc lưu trữ và quản lý các loại dữ liệu như cài đặt ứng dụng, thông tin người dùng, hình ảnh, video, tệp tin và dữ liệu khác. Các ứng dụng thường cần có cơ chế lưu trữ để đảm bảo rằng dữ liệu của người dùng được bảo quản một cách an toàn và có thể truy cập lại khi cần thiết.

Đối với trong ngôn ngữ lập trình Flutter, "storage" thường liên quan đến các phương pháp và thư viện để lưu trữ và quản lý dữ liệu trong ứng dụng, như Shared Preferences, cơ sở dữ liệu SQLite, quản lý trạng thái ứng dụng và các giải pháp khác.

Cách hoạt động của các phương pháp lưu trữ trong Flutter có thể được mô tả như sau:

* ***Shared Preferences***

- Shared Preferences hoạt động dựa trên cơ chế key-value.

- Khi bạn lưu trữ dữ liệu, bạn chỉ cần chỉ định một khóa (key) duy nhất và giá trị tương ứng.

- Dữ liệu sau đó được lưu trữ trong bộ nhớ cục bộ của thiết bị, thường là dưới dạng tệp tin JSON.

- Khi cần truy xuất dữ liệu, bạn chỉ cần cung cấp khóa tương ứng để truy cập giá trị.

* ***Local Database (SQLite)***

- SQLite là một hệ thống quản lý cơ sở dữ liệu quan hệ nhỏ gọn và hiệu quả.

- Bạn xác định các bảng và các cột trong bảng để lưu trữ dữ liệu.

- Dữ liệu được lưu trữ trong các bảng theo cấu trúc quan hệ.

- Bạn có thể thực hiện các thao tác như thêm, sửa, xóa và truy vấn dữ liệu trong cơ sở dữ liệu.

* ***File System***

- Khi bạn lưu trữ dữ liệu trong hệ thống tệp tin, bạn sẽ tạo ra các tệp tin trên thiết bị của người dùng.

- Dữ liệu có thể được lưu trữ dưới dạng văn bản hoặc dữ liệu nhị phân, chẳng hạn như hình ảnh, video, và tệp tin khác.

- Bạn có thể sử dụng các phương thức của thư viện Flutter để ghi và đọc tệp tin.

* ***Package Preferences***

- Các gói thư viện cho phép bạn lưu trữ dữ liệu một cách cấu trúc và hiệu quả hơn.

- Chúng thường sử dụng các cơ chế nền tảng như SQLite để lưu trữ dữ liệu, nhưng cung cấp các giao diện dễ sử dụng hơn cho lập trình viên.

- Các gói thư viện này có thể hỗ trợ tự động chuyển đổi dữ liệu thành các đối tượng Dart hoặc các cấu trúc dữ liệu khác.

* ***State Management Solutions***

- Các giải pháp quản lý trạng thái có thể lưu trữ dữ liệu trạng thái của ứng dụng và cung cấp cách thức để truy cập và cập nhật dữ liệu một cách hiệu quả.

- Thay vì lưu trữ dữ liệu một cách trực tiếp, bạn thường sẽ lưu trữ dữ liệu trong một trạng thái toàn cục và sử dụng các phương thức được cung cấp bởi giải pháp quản lý trạng thái để truy xuất và cập nhật dữ liệu.

Tùy thuộc vào mục đích và tính chất của dữ liệu, bạn có thể lựa chọn phương pháp lưu trữ phù hợp nhất cho ứng dụng của mình.

1. **3rd Party Librabies**Trong lập trình Flutter cũng vậy, "thư viện của bên thứ ba" (3rd Party Libraries) là các gói mã nguồn được phát triển bởi cộng đồng hoặc các tổ chức khác ngoài nhóm phát triển chính của Flutter. Các thư viện này cung cấp các chức năng, giao diện, hoặc tính năng khác mà bạn có thể sử dụng trong ứng dụng Flutter của mình để tiết kiệm thời gian và công sức.

Flutter có một hệ thống quản lý thư viện mạnh mẽ được gọi là "pub", cho phép bạn dễ dàng thêm và quản lý các thư viện của bên thứ ba trong dự án của mình. Bằng cách thêm các thư viện này vào dự án của bạn thông qua tệp `pubspec.yaml`, bạn có thể sử dụng các tính năng và chức năng mới mà các thư viện này cung cấp.

Một số ví dụ về thư viện của bên thứ ba trong cộng đồng Flutter bao gồm:

***http:*** Thư viện này cung cấp các công cụ để thực hiện các yêu cầu HTTP như tải dữ liệu từ các API.

***provider:*** Một thư viện quản lý trạng thái giúp bạn quản lý trạng thái ứng dụng của bạn một cách hiệu quả.

***firebase\_core và firebase\_auth***: Các thư viện kết nối ứng dụng Flutter với dịch vụ Firebase, giúp bạn thực hiện xác thực người dùng và quản lý dữ liệu.

***flutter\_svg:*** Cho phép bạn hiển thị các hình ảnh vector SVG trong ứng dụng Flutter của bạn.

***sqflite:*** Thư viện này giúp bạn làm việc với cơ sở dữ liệu SQLite trong ứng dụng Flutter.

***charts\_flutter:*** Cho phép bạn tạo các biểu đồ và đồ thị dễ dàng trong ứng dụng của bạn.

Khi sử dụng các thư viện của bên thứ ba, luôn kiểm tra tài liệu, đảm bảo tích hợp đúng cách và xem xét tương thích phiên bản để tránh các vấn đề khả năng xung đột hoặc bất kỳ vấn đề nào khác mà có thể xảy ra.

1. **Behavior Components**
2. **Sate Managements**
3. **Quality Assurance**
4. **Version Controlf**Là một hệ thống được sử dụng để quản lý và theo dõi sự thay đổi của mã nguồn trong quá trình phát triển phần mềm. Mục tiêu chính của Version Control là giúp các nhà phát triển làm việc cùng nhau trên cùng một dự án mà không gây ra xung đột hoặc mất dữ liệu, cung cấp khả năng theo dõi lịch sử thay đổi và giúp quản lý phiên bản của mã nguồn.

Các khái niệm chính trong Version Control bao gồm:

***Repository (Kho mã nguồn):*** Đây là nơi lưu trữ toàn bộ lịch sử thay đổi của mã nguồn. Kho mã nguồn có thể nằm trên máy cá nhân hoặc trên các dịch vụ lưu trữ từ xa như GitHub, GitLab, Bitbucket, và nhiều nền tảng khác.

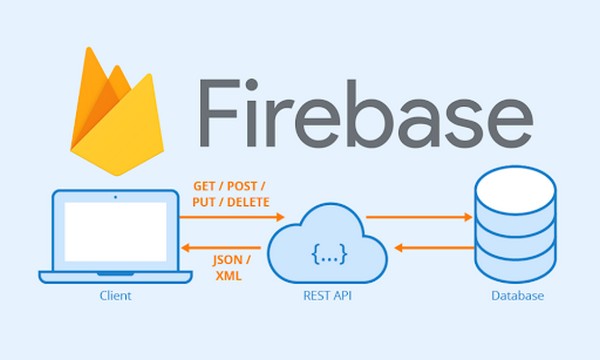
***Commit:*** Mỗi khi bạn thực hiện một sự thay đổi trên mã nguồn (thêm tệp mới, chỉnh sửa, xóa), bạn tạo ra một commit. Mỗi commit chứa thông tin về thay đổi, thông điệp mô tả và một số mã xác định để theo dõi.

***Branch (Nhánh)***: Các nhánh cho phép bạn tách nhánh ra từ phiên bản chính (thường là nhánh master hoặc main) để làm việc trên tính năng, sửa lỗi, hoặc thay đổi mà không ảnh hưởng đến phiên bản chính.

***Merge (Hợp nhất):*** Hợp nhất là quá trình kết hợp các thay đổi từ một nhánh vào nhánh khác, thường là để tích hợp tính năng hoặc sửa lỗi từ nhánh phụ vào nhánh chính.

***Conflict (Xung đột):*** Xung đột xảy ra khi hai người cùng làm việc trên cùng một phần của mã nguồn và thực hiện những thay đổi không tương thích. Hệ thống Version Control sẽ yêu cầu người dùng giải quyết xung đột này trước khi có thể hợp nhất.

Các hệ thống Version Control phổ biến bao gồm Git, Subversion (SVN), Mercurial và nhiều hệ thống khác. Git là một trong những hệ thống quản lý phiên bản phổ biến nhất và được sử dụng rộng rãi trong cộng đồng phát triển phần mềm.

1. **Firebase**

Firebase là gì đang được rất nhiều người thắc mắc, dựa theo định nghĩa thì nó là một nền tảng được sử dụng để phát triển các ứng dụng di động trong web. Nó được hoạt động trên nền tảng cloud, bao gồm các API đơn giản, mạnh mẽ nhưng không cần backend và server. Nó có chứa các CSDL với khả năng sẵn mang đến khả năng code nhanh chóng và thuận tiện

Chức năng chính của nó chính là giúp cho việc lập trình các ứng dụng một cách đơn giản hóa, bằng các thao tác với cơ sở dữ liệu. Điều đó giúp cho nó có khả năng code nhanh chóng và tiết kiệm thời gian triển khai cũng như quy mô mở rộng các ứng dụng đang thiết kế

Dưới đây là một số khía cạnh chính của Firebase:

***Lưu trữ dữ liệu (Firebase Realtime Database và Cloud Firestore)***: Firebase cung cấp hai dịch vụ lưu trữ dữ liệu phi cấu trúc, cho phép bạn lưu trữ và đồng bộ dữ liệu thời gian thực giữa các thiết bị. Firebase Realtime Database sử dụng mô hình dữ liệu JSON, trong khi Cloud Firestore sử dụng mô hình dữ liệu tài liệu.

***Xác thực người dùng***: Firebase Authentication giúp bạn quản lý xác thực người dùng thông qua nhiều phương thức như email/password, Google, Facebook, Twitter và nhiều phương thức khác. Điều này giúp đơn giản hóa việc xây dựng hệ thống đăng nhập và đăng ký trong ứng dụng của bạn.

***Phân tích ứng dụng:*** Firebase Analytics cung cấp dịch vụ theo dõi và phân tích hiệu suất ứng dụng của bạn. Bạn có thể theo dõi sự tương tác của người dùng, thống kê, dấu vết sự kiện, và hơn thế nữa để hiểu rõ hơn về cách người dùng sử dụng ứng dụng của bạn.

***Thông báo đẩy:*** Firebase Cloud Messaging (FCM) cho phép bạn gửi thông báo đẩy tới các thiết bị di động và máy tính bảng. Điều này giúp bạn thông báo với người dùng về các cập nhật mới, tin tức hoặc thông tin quan trọng khác.

***Lưu trữ tệp và ảnh***: Firebase cung cấp dịch vụ lưu trữ tệp và ảnh trực tiếp trên nền tảng của họ, giúp bạn quản lý việc lưu trữ và truy cập dữ liệu đa phương tiện.

***Phát triển ứng dụng web***: Firebase cung cấp các công cụ để xây dựng ứng dụng web thời gian thực, cho phép bạn tạo ứng dụng có khả năng đồng bộ dữ liệu và tương tác thời gian thực trực tiếp trên trình duyệt.

***Máy chủ hàm (Firebase Cloud Functions):*** Đây là dịch vụ cho phép bạn viết và triển khai mã JavaScript trên máy chủ mà không cần phải quản lý máy chủ riêng của mình. Điều này giúp bạn xử lý các tác vụ phức tạp như xử lý dữ liệu, gửi email hoặc thực hiện các hoạt động phía máy chủ.

**Ưu điểm của nền tảng firebase:**

* Cách tạo tài khoản và sử dụng dễ dàng: Ưu điểm đầu tiên của nền tảng firebase đối với người dùng đó chính là cho phép đăng ký tài khoản Google nhanh chóng và đơn giản. Nền tảng cung cấp miễn phí các tính năng để các nhà phát triển có thể sử dụng, tuy nhiên đối với những tính năng cao hơn thì bắt buộc người dùng phải trả phí.
* Sử dụng với tốc độ cao: Ưu điểm tiếp theo của các firebase đó chính là giúp cho người phát triển ứng dụng có thể tiết kiệm thời gian thiết kế. Các Dev có quyền truy cập và các server và host để để quản lý thời gian lập trình và hoàn tất công việc cụ thế.
* Là nền tảng đa dịch vụ: Là một trong những nền tảng đa dịch vụ bằng việc cung cấp đầy đủ các dịch vụ hỗ trợ cho việc phát triển web. Người tham gia lập trình có thể tùy cho database firebase hoặc realtime theo công việc và sở thích của mình.
* Là một thành phần của Google: Dựa theo lịch sử phát triển bạn có thể thấy nền firebase đã được Google mua lại và trở thành một phần của nó. Google đã phát triển và khai thác triệt để các sức mạnh và tính năng của firebase để tạo ra những dịch vụ sắn cho cho người dùng.
* Phát triển giao diện người dùng tốt: Một trong những ưu điểm mà ai cũng có thể nhìn nhận được tại nền tảng firebase đó chính và có khả năng tập trung vào giao diện người dùng. Xây dựng một kho backend với nhiều giao diện đa dang, phong phú giúp người dùng thoải mái lựa chọn.
* Không có máy chủ: Đặc điểm của firebase đó chính là hoạt động không dựa trên máy chủ, nên nó có khả năng tối ưu hóa hiệu suất làm việc bằng cách thức mở rộng database.
* Chức năng học máy: Đây được coi là một trong những chức năng vô cùng cần thiết kết và rất tốt cho các Dev khi mới tiếp cận nền tảng. Công dụng này có khả năng hỗ trợ tối đa cho việc phát triển ứng dụng trong quá trình làm việc của các lập trình viên.
* Tăng lượng traffic: Firebase có chức năng hỗ trợ tạo lập các chỉ mục giúp nâng cao thứ hạng của ứng dụng trên Google. Được coi là một trong những phương pháp giúp cho người dùng có thể dễ dàng tiếp cận với sản phẩm và tăng traffic.
* Phát hiện hỗi: Chức năng của firebase đó chính là theo dõi hệ thống, không chỉ là một công cụ phát triển nó còn có khả năng phát hiện lỗi, từ đó đưa ra hướng giải quyết hiệu quả.
* Sao lưu dữ liệu: Firebase sẽ tiến hành sao lưu thường xuyên tất cả dữ liệu, đảm bảo tính có sẵn khi Dev sử dụng.

**Nhược điểm của nền tảng firebase:**

* Không phải tất cả các dịch vụ firebase đều miễn phí, mức phí của nó cũng khá đắt và không ổn định..
* Vì firebase không phải là mã nguồn mở cho nên không có quyền truy cập mã nguồn.
* Firebase chỉ được cho phép hoạt động ở một vài quốc gia.
* Firebase chỉ chạy trên Google cloud và không cung cấp API GraphQL.
* Thời gian truy vấn của firebase chậm với kích thước tài liệu lớn.

1. **Native Integration**

Native integration là khả năng xây dựng quy trình làm việc kết hợp nhiều ứng dụng SaaS như Salesforce và Quickbooks. Chúng cung cấp cách trực tiếp để tích hợp và giao tiếp với nhau thông qua các giao diện lập trình ứng dụng (APIs). Dữ liệu được ánh xạ có thể dòng chảy giữa các ứng dụng SaaS trong thời gian thực và sẵn sàng cho các đội ngũ thích hợp.

APIs rất quan trọng trong việc native integration vì chúng là cấu trúc mã để đảm bảo sự tương tác giữa các ứng dụng.

Native integration có thể được phát triển bên trong hoặc thông qua một nền tảng tích hợp nhúng (embedded iPaaS) cung cấp các công cụ tích hợp cho người dùng xây dựng và triển khai tích hợp trong ứng dụng.

Tối ưu hóa quy trình với việc xây dựng tích hợp nhanh chóng

Native integration hữu ích cho tích hợp SaaS vì chúng cho phép các công ty SaaS kiểm soát toàn bộ vòng đời tích hợp. Từ phát triển, thiết kế đến triển khai, hỗ trợ và điều chỉnh.

Làm thế nào để thực hiện?

Sản phẩm SaaS mà bạn mua hoặc thuê như Slack thường có tích hợp native sẵn sàng sử dụng và có thể sử dụng ngay. Những tích hợp này được hỗ trợ bởi công ty mẹ và các nhóm phát triển của họ. Tuy nhiên, họ tập trung vào việc phát triển sản phẩm cốt lõi của họ chứ không phải là tích hợp.

Những tích hợp này thường thiếu mức độ tùy chỉnh hoặc trường hợp sử dụng cụ thể mà bạn yêu cầu. Đó là lý do tại sao việc xây dựng các tích hợp native của riêng bạn có thể hấp dẫn. Nhưng chúng có thể mất nhiều thời gian và tài nguyên. Tuy nhiên, chúng có thể dẫn đến mức độ tùy chỉnh cao nhưng giới hạn về khả năng mở rộng và linh hoạt.

Có thể sử dụng embedded iPaaS để nhanh chóng xây dựng các tích hợp native với các công cụ mã thấp trong một môi trường trắng làm nhãn riêng cho người dùng cài đặt và triển khai. Điều này có nghĩa là SaaS của bạn có thể có tất cả chức năng tích hợp native bạn cần cho khách hàng mà không cần phải đầu tư phát triển.

Do đó, việc thực hiện các tích hợp native linh hoạt và có khả năng mở rộng dễ dàng hơn bạn nghĩ. Bằng cách áp dụng một chiến lược tích hợp và nền tảng nhúng, như Cyclr, bạn có thể nhanh chóng cung cấp các tích hợp native trực tiếp cho người dùng trong ứng dụng SaaS của bạn.