BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG

TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM

KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM

BÁO CÁO ĐỒ ÁN MÔN
CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG

VIẾT APP ỨNG DỤNG CHAT CƠ BẢN

GVHD ThS. Nguyễn Thanh Truyền

Thành viên nhóm

Nguyễn Phạm Thảo Ngân 0850070031

Lại Thị Phương Nhung 0850070036

Trần Thanh Phong 0850070039

Nguyễn Ngọc Quang 0850070045

Võ Hoàng Triều 0850070060

Lớp 08_ĐH_TTMT

TP. Hồ Chí Minh, tháng 8 năm 2023

BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG

TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM

KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM

BÁO CÁO ĐỒ ÁN MÔN
CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG

VIẾT APP ỨNG DỤNG CHAT CƠ BẢN

GVHD ThS. Nguyễn Thanh Truyền

Thành viên nhóm

Nguyễn Phạm Thảo Ngân 0850070031

Lại Thị Phương Nhung 0850070036

Trần Thanh Phong 0850070039

Nguyễn Ngọc Quang 0850070045

Võ Hoàng Triều 0850070060

Lớp 08_ĐH_TTMT

TP. Hồ Chí Minh, tháng 8 năm 2023

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành nhất đến thầy ThS. Nguyễn Thanh Truyền giảng viên bộ môn Công nghệ lập trình đa nền tảng cho ứng dụng di động trường Đại học Tài nguyên và Môi trường Thành phố Hồ Chí Minh. Trong quá trình học tập, thực hành và tìm hiểu nhóm chúng em đã nhận được sự quan tâm giúp đỡ, hướng dẫn rất tận tình, tâm huyết của thầy. Thầy đã giúp nhóm chúng em tích lũy thêm nhiều kiến thức để có cái nhìn sâu sắc và hoàn thiện hơn trong môn học.

Từ những kiến thức mà thầy đã truyền tải, nhóm chúng em đã hiểu thêm được những khúc mắc trong môn học này. Thông qua bài báo cáo đồ án môn học này, em xin trình bày lại những gì mà mình đã tìm hiểu được về môn học và cho ra đề tài

Có lẽ kiến thức là vô hạn mà sự tiếp nhận kiến thức của bản thân mỗi người luôn tồn tại những hạn chế nhất định. Do đó, trong quá trình hoàn thành bài tiểu luận, chắc chắn không tránh khỏi những thiếu sót. Bản thân nhóm chúng em rất mong nhận được những góp ý đến từ thầy để bải tiểu luận của nhóm chúng em được hoàn thiện hơn.

Lời cuối, nhóm chúng em kính chúc thầy có thật nhiều sức khỏe, hạnh phúc và thành công trên con đường sự nghiệp giảng dạy.

Nhóm chúng em xin chân thành cảm ơn

MỞ ĐẦU

Theo thống kê, dân số Việt Nam hiện đang xấp xỉ 100 triệu người, có khoảng 70% hiện đang trong độ tuổi lao động, và 40% trong số đó dưới 25 tuổi. Hơn thế nữa, có hơn một nửa quốc gia được kết nối với Wi-Fi và một nửa trong số đó là người dùng Internet thông qua các thiết bị di động (smartphone). Nói cách khác, Việt Nam được đánh giá là một quốc gia rộng lớn, trẻ trung và ngày càng hiểu biết về công nghệ.

App mobile được tạo ra nhằm hướng đến các nhu cầu trong cuộc sống từ giải trí, thể thao, hay bất kỳ thứ gì cần thiết trong cuộc sống đặc biệt là nhu cầu mua sắm. Hiện nay có đến khoảng 89% người sử dụng các ứng dụng điện thoại ở Việt Nam. Đa phần mọi người ở trong độ tuổi từ 18 – 24 là nhóm sẵn sàng mua hàng qua các ứng dụng chiếm khoảng 94%.

Bên cạnh đó việc phát triển các app mobile, người ta còn thiết kế cả các web tùy theo mục đích sử dụng riêng. Tuy nhiên so với những trang website thì việc ứng dụng điện thoại nhằm cung cấp nhiều tương tác hơn, trình bày những thông tin chi tiết về sản phẩm, hàng hóa và dịch vụ được thực hiện trên app iOS/Android.

Như vậy, công nghệ lập trình đa nền tảng cho ứng dụng di động mục đích nhằm phục vụ cho nhu cầu của người dùng ngày càng tăng và theo kịp xu hướng của công nghệ hiện nay.

Với nhu cầu sử dụng điện thoại thông minh phổ biến như hiện nay việc liên lạc giữa người với người là vô cùng tất yếu và đóng một vai trò quan trọng trong cuộc sống. Không những giúp chúng ta trong việc liên lạc, mà còn giúp cho chúng ta giải quyết được những nhu cầu về mặt giải trí và những lợi ích xã hội khác.

Chính vì vậy, để đáp ứng những nhu cầu cấp thiết trên việc xây dựng một ứng dụng chat mà ở đó mọi người có thể kết nối thông qua ứng dụng mà không có bất kỳ giới hạn nào, chỉ cần có ứng dụng và kết nối mạng Internet là chúng ta đã có thể kết nối đến mọi người ở khắp mọi nơi. Với mong muốn mọi người có thể cảm thấy thoải mái hơn khi liên lạc với nhau mà không có bất kỳ trở ngại, giúp mọi người có thể gần gũi và gắn kết hơn với nhau. Đó cũng chính là điều mà tôi muốn nhất trong việc lựa chọn đề tài này.

Mục đích đề tài

Với mục tiêu là giúp cho mọi người có thể liên lạc với nhau một cách thuận tiện hơn, ứng dụng không chỉ giúp cho mọi người kết nối được với nhau dễ dàng hơn, mà còn giúp cho mọi người có thể kết bạn và tạo hội nhóm với nhau.

Tiếp đó, ứng dụng còn cung cấp nơi cho chúng ta có thể đưa những bức ảnh đại diện của mình lên và nhờ đó mà chúng ta có thể tìm được nhau và kết bạn với nhau một cách dễ dàng nhất. Ngoài ra nó còn được bảo mật một cách an toàn giúp cho thông tin của người dùng không bị lộ ra ngoài.

Đối tượng nghiên cứu ở đây là những người có nhu cầu sử dụng các phần mềm trên điện thoại để kết nối, liên lạc

Kết cấu đề tài

Đề tài bao gồm 5 chương , trong đó:

Chương 1: Tổng quan

Chương 2: Cơ sở lý luận

Chương 3: Các bước cài đặt

Chương 4: Các bước tiến hành

Chương 5: Kết luận

MỤC LỤC

MỤC LỤC HÌNH ẢNH

LỊCH LÀM VIỆC

Tuần	Nội dung công việc được giao
1 + 2	- Tìm hiểu đề tài - Tìm hiểu phần lý thuyết - Tải các ứng dụng liên quan
3	- Tổng hợp lý thuyết - Triển khai phần thực hành trên ứng dụng
4	- Tiếp tục khiển khai phần thực hành trên ứng dụng - Viết báo cáo tổng hợp
5	- Chỉnh sửa và báo cáo

CHƯƠNG 1: TỔNG QUAN

1.1. Tổng quan môn học

1.1.1. Đa nền tảng

Đa nền tảng là một thuật ngữ để chỉ các phần mềm hay phương thức điện toán được vận hành cùng nhau trên nhiều nền tảng. Như vậy, một phần mềm được gọi là đa nền tảng khi và chỉ khi nó có khả năng hoạt động trên nhiều hơn một hệ điều hành hay kiến trúc máy tính.

Hiện nay có khá nhiều loại phần mềm đa nền tảng khác nhau được tạo ra nhưng nhìn chung chúng ta có thể phân chúng thành 2 loại chính là:

- Loại phần mềm có thể trực tiếp chạy trên bất cứ nền tảng nào mà không cần đến các bước biên dịch/ thông dịch. Để làm được điều này, các lập trình viên cần viết phần mềm bằng một loại ngôn ngữ thông dịch hoặc đã dịch phần mềm trước sang mã bytecode…
- Phần mềm đa nền tảng còn lại đơn giản, dễ thiết kế hơn. Tuy nhiên nó yêu cầu có thêm bước thiết kế hoặc biên dịch từng phiên bản cho mỗi nền tảng mà nó hỗ trợ.

Hình 1: Đa nền tảng

Mỗi loại phần mềm đều có ưu nhược điểm riêng phụ thuộc vào mục đích của người dùng. Tuy nhiên, tất cả các phần mềm đa nền tảng trên thị trường đều có một điểm chung là nó có thể chạy trên các nền tảng khác nhau nhưng không thể tự động làm việc trên tất cả kiến trúc mà hệ điều hành đó hỗ trợ.

1.1.2. Lập trình đa nền tảng

Lập trình đa nền tảng là việc viết code để tạo ra các phần mềm, ứng dụng, hoặc chương trình hoạt động trên nhiều hệ điều hành hoặc nền tảng khác nhau, ví dụ như Windows, MacOS, iOS, Android,...Nói cách khác, ứng dụng hoặc phần mềm đó được viết một lần và có thể chạy trên nhiều hệ điều hành mà không cần thay đổi mã nguồn.

Lập trình đa nền tảng là phương pháp giúp doanh nghiệp tiết kiệm thời gian và chi phí khi phát hành một ứng dụng. Theo đó, nhà phát triển sẽ dễ dàng đẩy nhanh tiến độ tạo ra một phần mềm hoặc ứng dụng mới khi xây dựng chúng dựa trên lập trình đa nền tảng.

Hình 2: Lập trình đa nền tảng là gì

Các ngôn ngữ lập trình đa nền tảng được các lập trình viên ưa thích gồm có như Java, C ++ và JavaScript.

1.1.2.1. Ưu điểm

Xu hướng lập trình đa nền tảng đã lan rộng ra toàn thế giới bởi nó mang đến nhiều tiện ích nổi bật, một số có thể kể đến như:

- **Dễ dàng tiếp cận khách hàng**

Lợi ích đầu tiên mà ai cũng có thể nhận thấy đó chính là lập trình mobile đa nền tảng giúp tiếp cận khách hàng dễ dàng và nhanh chóng mở rộng thị trường. Ứng dụng của bạn có thể hoạt động trên nhiều nền tảng khác nhau đồng nghĩa với việc độ phủ sóng đang ngày càng mở rộng. Và như thế, không khó để tiếp cận với đối tượng khách hàng mục tiêu. Bên cạnh đó, khi sở hữu các ứng dụng đa nền tảng, bạn sẽ thuận tiện trong việc marketing thương hiệu/ sản phẩm trên các phương tiện truyền thông khác nhau.

- **Tiết kiệm thời gian và chi phí**

Thay vì phải tốn nhiều nhân lực và thời gian trong việc tạo ra nhiều ứng dụng cho các hệ điều hành khác nhau thì giờ đây bạn chỉ cần viết code một lần – quản lý tập trung cho ứng dụng của mình trên tất cả các nền tảng. Không những thế, việc tự do tiếp thị trên các phương tiện truyền thông cũng giúp bạn giảm thiểu chi phí quảng cáo.

- **Sử dụng công nghệ lập trình cao**

Trong khi lập trình đa nền tảng, bạn có thể sử dụng các công nghệ lập trình tiên tiến để nâng cao trải nghiệm người dùng cũng như tăng độ mượt mà cho ứng dụng. Bên cạnh đó thì bạn cũng hoàn toàn có thể sử dụng các công nghệ cũ, quen thuộc để viết phần mềm đa nền tảng.

Đây cũng chính là lợi thế về sự linh hoạt trong lập trình. Với lợi thế này, bạn vừa có thể tối ưu nguồn lực sẵn có và cũng có thể tận dụng chúng cho việc khai thác nâng cao để tạo ra các phần mềm cao cấp hơn!

- **Tạo ra và duy trì sự đồng bộ hoá**

Và một lợi thế khác của lập trình đa nền tảng chính là khả năng đồng bộ hoá "cực đỉnh". Thay vì phải cập nhật dữ liệu và đồng bộ thủ công giữa ứng dụng của các nền tảng khác nhau thì giờ đây bạn có thể ngay lập tức đồng bộ hoá dữ liệu trên tất cả các nền tảng.

1.1.2.2. Nhược điểm

- **Hạn chế về sự linh hoạt**

Trước hết, một ứng dụng đa nền tảng chắc chắn sẽ bị hạn chế về sự linh hoạt hơn so với các ứng dụng phục vụ cho một nền tảng nhất định. Mỗi nền tảng sẽ có đặc điểm, yêu cầu, phương thức hoạt động, cách hiển thị và nhu cầu người dùng khác nhau. Những yếu tố này sẽ đặt ứng dụng đa nền tảng vào một phạm vi bất lợi nhất định.

Trong khi đó, yêu cầu của người dùng ngày càng khắt khe hơn, bởi thế bạn phải cân nhắc lựa chọn kỹ lưỡng giữa việc lập trình ứng dụng đa nền tảng hay thiết kế nhiều ứng dụng cho các nền tảng khác nhau.

- **Việc đảm bảo chất lượng gặp khó khăn**

Việc thiết kế ra một ứng dụng hoạt động tốt được trên nhiều nền tảng đòi hỏi khả năng và công nghệ lập trình cao. Có rất nhiều yếu tố cần cân nhắc trong quá trình lập trình là:

- Giao diện người dùng khác nhau;
- Khác biệt về công cụ và ngôn ngữ trong mỗi nền tảng API;
- Tích hợp ứng dụng với thiết lập cục bộ giữa các nền tảng;
- Tùy chọn lưu trữ;
- Xử lý yêu cầu từ các bên liên quan…

1.1.3. Lợi ích của công cụ đa nền tảng phổ biến

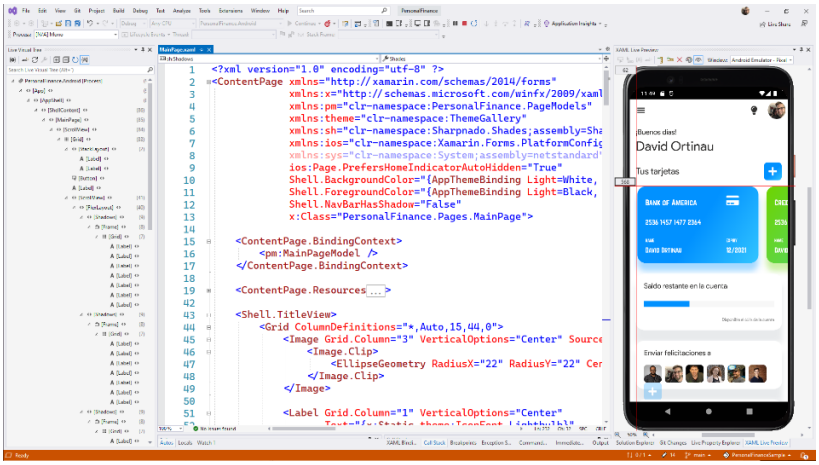
Đa nền tảng là một framework mà qua đó một ứng dụng di động có thể chạy trên nhiều nền tảng như Android, iOS và Windows. Một số lợi ích khác của việc sử dụng các franework đa nền tảng để phát triển ứng dụng dành cho thiết bị di động bao gồm:

- Khả năng tái sử dụng mã
- Tích hợp đám mây
- Hiệu quả về chi phí
- Lưu trữ dễ dàng
- Ít rào cản kỹ thuật hơn

1.1.4. Công cụ đa nền tảng phổ biến

1.1.4.1. Xamarin

Các ứng dụng được phát triển bằng cách sử dụng framework này được xây dựng bằng C # . Công cụ này được sử dụng để thiết kế ứng dụng cho nhiều nền tảng.



Hình 3: Xamarin

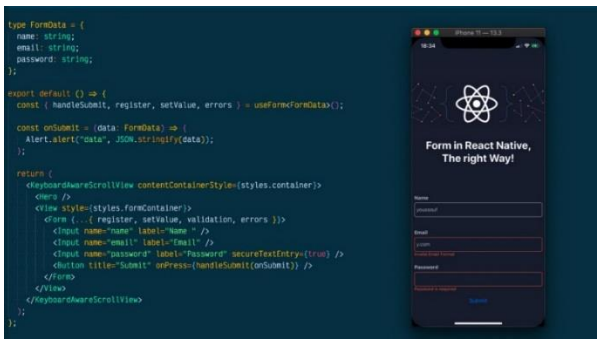
1.1.4.2. PhoneGap

PhoneGap, một công cụ phát triển đa nền tảng đã được Adobe tung ra để xây dựng các ứng dụng dành cho thiết bị di động được hỗ trợ bởi công nghệ web mở. Sử dụng các ngôn ngữ như HTML5, CSS và Javascript để phát triển ứng dụng

Hình 4: PhoneGap

1.1.4.3. React Native

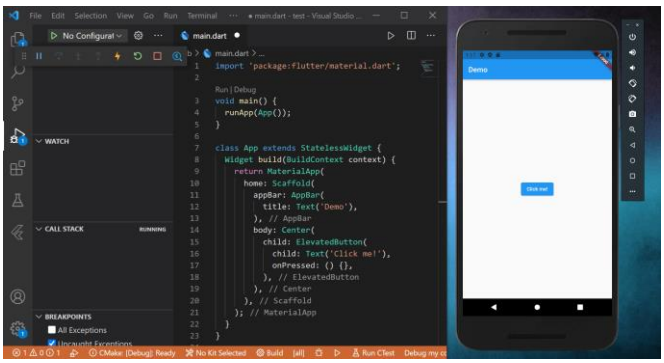
Một framework đa nền tảng khác cung cấp cùng một mã để phát triển ứng dụng di động cho bất kỳ nền tảng nào là React Native. Quá trình phát triển ứng dụng dành cho thiết bị di động trên nền tảng này thực sự nhanh chóng và doanh nghiệp có thể khởi chạy ứng dụng của mình trên cả hai nền tảng mà không ảnh hưởng đến chất lượng.



Hình 5: React Native

1.1.4.4. Flutter

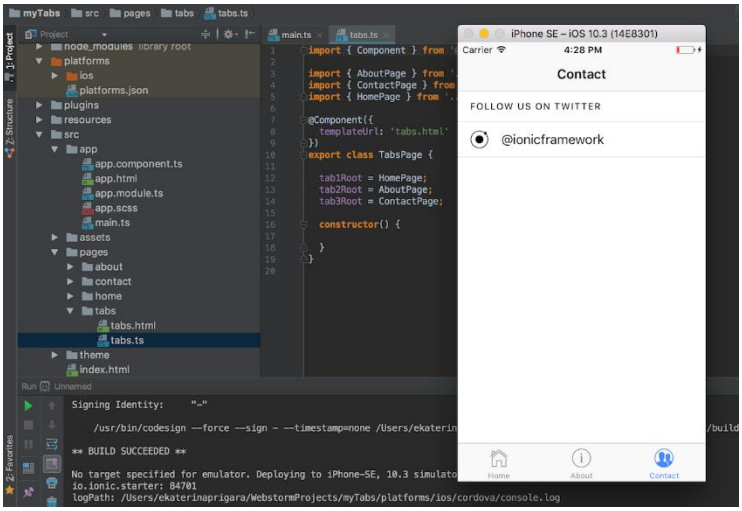
Flutter là một frameword ứng dụng đa nền tảng của Google. Các doanh nghiệp có thể nhận được hai ứng dụng di động có thể chạy trên nền tảng Android và iOS mà không cần đầu tư thêm tiền và thời gian sử dụng công cụ này.



Hình 6: Flutter

1.1.4.5. Ionic

Ionic là một framework đa nền tảng mã nguồn mở để phát triển ứng dụng di động kết hợp. Framework cho phép người dùng chọn bất kỳ khung giao diện người dùng nào như React và Angular.



Hình 7: Ionic

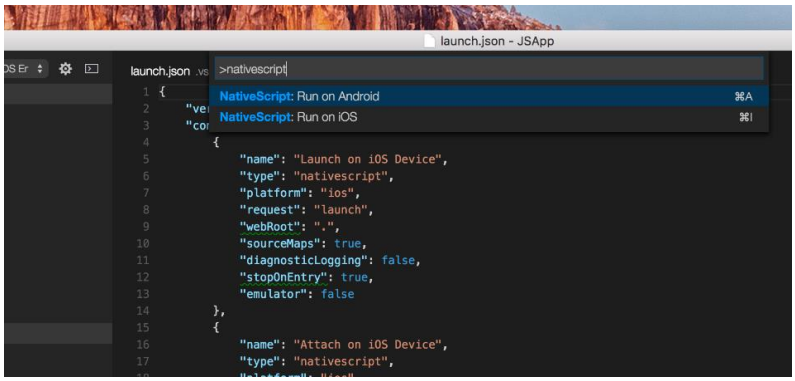
1.1.4.6. Sencha Touch

Sencha Touch cho phép các ứng dụng chạy nhất quán trên các trình duyệt và ứng dụng dành cho thiết bị di động. Một nhóm sử dụng framework này có thể ngồi ở bất cứ đâu và làm việc với sự trợ giúp của các yếu tố được tạo sẵn. Nó đã được hợp nhất với ext JS. để xây dựng các ứng dụng web sử dụng nhiều dữ liệu.

Hình 8: Sencha Touch

1.1.4.7. Native Script

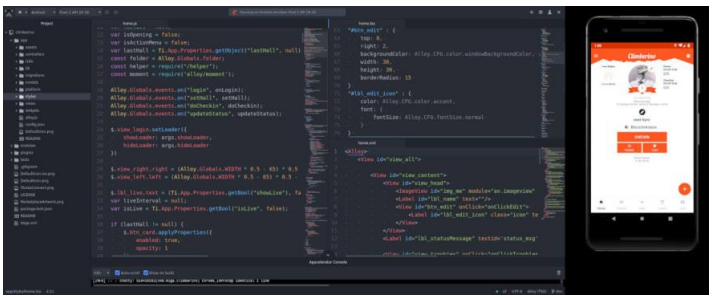
Native Script là một framework đa nền tảng miễn phí để phát triển các ứng dụng di động cho các nền tảng Android và iOS. Các ứng dụng dành cho thiết bị di động được tạo bằng JavaScript. Sử dụng framework này, các nhà phát triển có thể tùy chỉnh giao diện người dùng cho các thiết bị cụ thể. Framework cung cấp cho các nhà phát triển các plugin cho tất cả các loại giải pháp.



Hình 9: Native Script

1.1.4.8. Appcelerator Titanium

Appcelerator là một công cụ phát triển ứng dụng đa nền tảng cho phép tạo các ứng dụng di động gốc cho các nền tảng như Android, iOS và Windows. Nó cung cấp nhiều công cụ cho các nhà phát triển để có quá trình phát triển ứng dụng nhanh chóng. Sử dụng ứng dụng này, các nhà phát triển có thể xây dựng một ứng dụng dành cho thiết bị di động trong thời gian ngắn hơn và ít nỗ lực hơn.



Hình 10: Appcelerator Titanium

1.2. Tổng quan về ứng dụng di động

1.2.1. Hệ điều hành Android

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Ban đầu, Android được phát triển bởi Android Inc. với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005.

Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào năm 2008.

Hình 11: Hệ điều hành Android

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache không có nhiều ràng buộc nên Android ngày càng trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới.

Vào quý 3 năm 2012, Android chiếm 75% thị phần về điện thoại thông minh trên toàn thế giới với con số tổng cộng khoảng 500 triệu thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Tuy nhiên, hiện nay Android đang có sự cạnh tranh với iOS từ

Apple nhưng Android vẫn dẫn vị thế thượng phong trên cuộc chiến điện thoại thông minh.

1.2.2. Giao diện và ứng dụng Android

1.2.2.1. Giao diện

Giao diện của Android sử dụng cảm ứng chạm, tác động trực tiếp lên màn hình như vuốt, chạm, phóng to và thu lại để xử lý các đối tượng trên màn hình.

Những ứng dụng do bên thứ 3 cung cấp có trên Google Play cho phép người dùng thay đổi chủ đề của màn hình chính tương tự như Windows Phone hay iOS. Tuy nhiên, những nhà sản xuất đều thay đổi hình dáng Android một cách linh hoạt để dễ dàng phân biệt chúng với các hệ điều hành khác.

Các phiên bản Android sẽ được Google cập nhật theo chu kỳ từ 6 đến 9 tháng. Hiện nay (tính đến tháng 08/2021), bản cập nhật chính thức mới nhất của Android là Android 11.

Hình 12: Phiên bản mới nhất Android 11

1.2.2.2. Ứng dụng

Các ứng dụng do bên thứ ba có trên Google Play để người dùng có thể tải về. Các ứng dụng trên Play Store cho phép người dùng tải về và cập nhật các ứng dụng do Google và các nhà phát triển phát hành.

Đối với những ứng dụng mất phí tải về, nếu người dùng mua một ứng dụng mà họ cảm thấy không hài lòng thì họ được hoàn trả tiền sau 15 phút kể từ lúc tải về.

Tính đến tháng 10/2012, đã có hơn 700.000 ứng dụng trên Android và số lượt tải về từ cửa hàng ứng dụng chính của Android (Google Play) chiếm khoảng 25 tỷ lượt. Đến nay, con số này đã lên tới 3 triệu ứng dụng.

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java và sử dụng Bộ phát triển phần mềm Android (SDK). Bộ phát triển này gồm có công cụ gỡ lỗi, thư viện phần mềm,... hỗ trợ với công suất tối đa cho nhu cầu của các thiết bị.

1.2.2.3. Ưu điểm

- **Kho ứng dụng đa dạng**

Với hệ thống cửa hàng ứng dụng Google Play, hệ điều hành Android có thể đáp ứng các nhu cầu từ chơi game cho đến làm việc với hơn 3 triệu ứng dụng để bạn lựa chọn.

- **Mẫu mã đa dạng**

Với nhiều nhà sản xuất lớn như Samsung, OPPO, Xiaomi, Huawei, Sony, Nokia,... bạn có thể lựa chọn giữa nhiều mẫu mã thiết bị khác nhau, từ các mẫu giá rẻ cho đến các mẫu cao cấp

- **Có thể mở rộng bộ nhớ bằng thẻ nhớ**

Với các thiết bị của Apple, bạn chỉ có thể sử dụng bộ nhớ trong có sẵn của máy. Còn với phần lớn các thiết bị Android, bạn sẽ có lựa chọn mở rộng bộ nhớ có sẵn với các loại thẻ nhớ dung lượng cao.

- **Khả năng tùy biến cao**

Do bản chất nguồn mở của hệ điều hành Android, ai cũng có thể lấy được mã nguồn của hệ điều hành này. Điều này cũng đồng nghĩa là các nhà sản xuất, cũng như là các lập trình viên độc lập, có thể tự do tùy biến Android để có được hiệu năng tốt nhất hoặc bỏ đi những tính năng không cần thiết.

- **Người dùng ưa chuộng nhiều**

Android có cộng đồng người dùng và lập trình viên độc lập khá lớn, nên khi bạn gặp vấn đề về thiết bị hay về phiên bản Android của bạn, bạn sẽ được hỗ trợ rất nhiệt tình từ phía cộng đồng.

1.2.2.4. Nhược điểm

- **Nhiều ứng dụng chạy ngầm làm chậm máy**

So với iOS, Android tối ưu hóa bộ nhớ RAM có phần kém hơn, dẫn đến việc nhiều ứng dụng chạy ngầm gây chậm máy hoặc thậm chí là đơ máy.

- **Một số ứng dụng chưa được tối ưu hóa tốt**

Do có quá nhiều mẫu mã khác nhau, các nhà phát triển không thể tối ưu hóa ứng dụng cho tất cả các mẫu thiết bị Android trên thị trường, nên các ứng dụng có thể gặp các lỗi như không hiển thị được toàn màn hình hoặc không thể tận dụng hết sức mạnh phần cứng của máy.

- **Chất lượng một số ứng dụng còn kém**

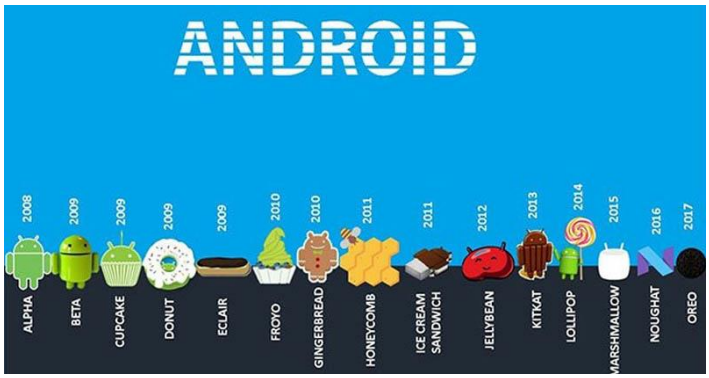
Một số ứng dụng trên Google Play có chất lượng khá kém với hàng loạt các quảng cáo khó chịu và không có các chức năng hữu dụng, gây cản trở cho công việc hay thời gian giải trí của bạn.

- **Dễ bị virus xâm nhập**

1.2.3. Sự phát triển của các phiên bản hệ điều hành Android

Android đã có hơn 10 năm trong quá trình phát triển của hệ điều hành. Chiếc điện thoại đầu tiên chạy Android là HTC Dream được bán vào ngày 22/10/2008.

Từ năm 2008 đến nay, Android đã trải qua 20 lần cập nhật để dần dần cải tiến hệ điều hành. Phiên bản mới nhất hiện nay là Android 11, ra mắt vào tháng 09/2020 với nội dung khá hoàn chỉnh hơn so với bản cập nhật trước.



Hình 13: Sự phát triển của hệ điều hành Android

Phiên bản Android	Tên mã	Ngày phát hành
1.6	Donut	15/09/2009
2.0 – 2.1	Eclair	26/10/2009
2.2	Froyo	20/05/2010
2.3 - 2.3.2	Gingerbread	06/12/2010
2.3.3 - 2.3.7	Gingerbread	09/02/2011
3.1	Honeycomb	10/05/2011
3.2	Honeycomb	15/07/2011
4.0.x	Ice Cream Sandwich	16/12/2011
4.1.x	Jelly Bean	09/07/2012
4.2.x	Jelly Bean	13/11/2012
4.3	Jelly Bean	25/07/2013
4.4	KitKat	10/2013
5.0	Lollipop	07/2014
6.0	Marshmallow	05/10/2015
7.0	Nougat	22/08/2016
8	Oreo	21/08/2017
9	Pie	07/08/2018
10	Android 10	03/09/2019
11	Android 11	09/2020

Hình 14: Sự phát triển của hệ điều hành Android

1.2.4. Tổng quan về ứng dụng di động

Ứng dụng di động còn được gọi là app mobile hay mobile application, là một chương trình ứng dụng dành riêng cho thiết bị di động như tablet hay smartphone cho phép người dùng có thể sử dụng để truy cập vào nội dung mà bạn mong muốn ngay trên thiết bị điện thoại di động đó.

Hình 15: Ứng dụng di động

Mục đích chính mà nhà lập trình muốn hướng đến khi thiết kế các sản phẩm Mobile Apps là cung cấp đa dạng tính năng như lướt web, xem phim, mua sắm, xem tin tức,… trên một không gian thu nhỏ. Những ứng dụng này cho phép người sử dụng có thể truy cập vào nhiều nội dung khác nhau ngay trên thiết bị di động.

1.2.5. Phân loại Mobile App

Hiện nay, Mobile App được phân biệt thành các loại chính là:

- **Native Mobile App (Ứng dụng gốc) :** Đây là dạng ứng dụng di động "gốc", hỗ trợ người dùng tải các dữ liệu hay nội dung xuống thiết bị di động và bạn chỉ việc truy cập để sử dụng
- **Hybrid Mobile App (Ứng dụng lai) :** Đây là loại ứng dụng di động tích hợp những đặc tính của Native App và Web App. Chúng được lập trình bởi các ngôn ngữ như HTML5, CSS3, JavaScript,… và được gói lại bởi lớp vỏ container để có thể được tải về từ kho ứng dụng di động của người dùng.
- **Web App (Ứng dụng web) :** Được xây dựng để tương tác, điều hướng và các tuỳ chỉnh khác, web app là các phần mềm trên thiết bị di động sử dụng công nghệ javascript hoặc HTML5 và trái ngược hoàn toàn với native app.
- **Progressive Web App :** Là xu hướng phát triển ứng dụng web mới nhất hiện nay, Progressive Web App được sử dụng phổ biến trên hệ điều hành android và IOS. Ứng dụng này được ra đời với mục đích gia tăng khả năng tiếp cận người dùng và những cái mới cho website bằng cách thêm một vài tính năng cho các ứng dụng web.

1.2.6. Tầm quan trọng của Moblie App

- **Độ phổ biến của Smartphone:**

Smartphone đang ngày càng phổ biến và khả năng thay thế được các feature phone tại cửa hàng là rất lớn. Hầu hết tất cả mọi người bao gồm doanh nhân, social networker và cả game thủ đều sử dụng các app trên smartphone.

- **Khả năng hiển thị:**

Smartphone ngày càng có sức hút và được sử dụng nhiều với các lượng công việc lớn mà trước đây bị hạn chế bằng các thiết bị máy tính xách tay hay máy tính để bàn. Các mobile app có khả năng hiển thị tốt. Vì vậy mà các doanh nghiệp thường ứng dụng nó để quảng bá thương hiệu, sản phẩm hay để cung cấp quyền truy cập cho sản phẩm đó.

- **Phổ biến của App Store:**

Các thiết bị di động sẽ phụ thuộc khá nhiều vào App Store. Vì vậy thời điểm hiện tại là cơ hội để phát triển, xây dựng và bổ sung ứng dụng của mình vào các App Store. Người dùng có thể dễ dàng tiếp cận, tìm thấy và tải xuống ứng dụng nếu có nhu cầu. Đây là kênh tiếp thị và phát triển thương hiệu rất hiệu quả mà doanh nghiệp nên tận dụng.

- **Gia tăng trải nghiệm cho khách hàng:**

Mobile app mang đến cho khách hàng những trải nghiệm khác biệt mà không một mobile web hay bất kì công cụ nào khác có thể có được. Ngoài ra, các thiết bị smartphone mang tính cá nhân sẽ gần gũi hơn với khách hàng so với các thiết bị khác. Điều này giúp doanh nghiệp có thể dễ dàng gia tăng trải nghiệm cho khách hàng của mình.

- **Đa dạng trong quảng bá thương hiệu:**

Sở hữu mobile app, doanh nghiệp có thể dễ dàng quảng bá cũng như tăng độ nhận biết thương hiệu của doanh nghiệp mình đến với người dùng một cách đa dạng, nhanh chóng và có hiệu quả.

CHƯƠNG 2: CƠ SỞ LÝ LUẬN

2.1. Lý thuyết

2.1.1. Flutter

2.1.1.1. Khái niệm

Flutter là một khung nguồn mở do Google phát triển và hỗ trợ. Các nhà phát triển frontend và fullstack sử dụng Flutter để xây dựng giao diện người dùng (UI) của ứng dụng cho nhiều nền tảng chỉ với một nền mã duy nhất.

Flutter bao gồm Reactive framework và công nghệ hiển thị 2D (2D rendering engine)và các công cụ phát trển(development tool). Các thành phần này làm việc cùng nhau giúp ta thiết kế, xây dựng, test, debug ứng dụng.

Hình 16: Flutter

2.1.1.2. Lịch sử hình thành

Flutter được ra mắt lần đầu năm 2015 và chính thức phát hành vào tháng 5/2017. Cho đến nay, flutter được biết đến với khá nhiều phiên bản:

- Phiên bản đầu tiên của Flutter được gọi là "Sky" chạy trên hệ điều hành Android. Được công bố tại hội nghị nhà phát triển Dart 2015.
- Tháng 5/2020 bộ công cụ phát triển phần mềm Dart(SDK) phiên bản 2.8 và Flutter 1.17/0 đã được phát hành bổ sung thêm bộ hỗ trợ Metal API giúp cải thiện 50% hiệu suất trên thiết bị IOS.
- Phiên bản Flutter 2.0 được phát hành 2/2021 trong một sự kiện trực tuyến flutter Engage. Phiên bản mới này chính thức hỗ trợ phát triển ứng dụng trên web, windows, MAC và Linux.

- Phiên bản Flutter 2.5 và Dart 2.14 được phát hành bởi Google vào tháng 9/2021, phiên bản này đã cải tiến chế độ toàn màn hình của android và phiên bản Material Design mới nhất của Google có tên là Material You.
- Phiên bản Flutter 3 và Dart 2.17 được Google công bố vào tháng 5/2022. Phiên bản này đã nâng cấp hỗ trợ 6 nền tảng bao gồm hỗ trợ ổn định cho Linux và MacOS trên cả bộ vị xử lý Intel và Apple Silicon.

2.1.1.3. Ưu điểm

- Hiệu suất gần với phát triển ứng dụng gốc. Flutter sử dụng ngôn ngữ lập trình Dart và biên dịch thành mã máy. Các thiết bị máy chủ hiểu được mã này, điều này đảm bảo hiệu suất nhanh và hiệu quả.
- Kết xuất nhanh, nhất quán và có thể tùy chỉnh. Thay vì dựa vào các công cụ kết xuất theo nền tảng, Flutter sử dụng thư viện đồ họa Skia nguồn mở của Google để kết xuất UI. Điều này mang đến cho người dùng phương tiện trực quan nhất quán cho dù họ sử dụng nền tảng nào để truy cập ứng dụng.
- Công cụ thân thiện với nhà phát triển. Google đã xây dựng Flutter chú trọng vào tính dễ sử dụng. Với các công cụ như tải lại nóng, nhà phát triển có thể xem trước các thay đổi mã sẽ như thế nào mà không bị mất trạng thái. Các công cụ khác như widget inspector giúp dễ dàng trực quan hóa và giải quyết các vấn đề với bố cục UI.

2.1.1.4. Nhược điểm

- Có một rào cản lớn cho những người chỉ học JS, hoặc đến từ RN. Do thực tế là Dart được sử dụng với sự thừa kế, đa hình của nó và tất cả các thứ về OOP.
- Không có JSX mà tất cả chúng ta đã quen khi làm RN. Nhìn vào tệp Dart trong đó không có phân chia thành template, styling và data, nó trở nên khó chịu.
- Styling là một nhược điểm. Thực tế là không có sự tách biệt thành Styles, templates và controller, có một vấn đề khi mô tả một thành phần thì chúng ta cũng cần mô tả các styles cùng một lúc.
- Animation còn khó khăn hơn. Mặc dù nó có một animation tốt, nhưng nó sẽ khó hơn khi tạo hiệu ứng động, không giống như trong RN.
- Tối ưu hóa. Trong Fluter, chỉ có thừa kế widget với các điều kiện có thể thay đổi (trạng thái) và không thể thay đổi (không trạng thái). Trong khi ở React-Native

chúng ta có thể quản lý vòng đời. Ngoài ra, một nhược điểm lớn là thiếu các công cụ để lưu trạng thái của ứng dụng, vấn đề này có thể, tuy nhiên, được giải quyết bằng cách tuần tự hóa trạng thái hiện tại.

2.1.2. Dart

2.1.2.1. Khái niệm

Dart là ngôn ngữ lập trình cho Flutter- bộ công cụ giao diện người dùng của Google để xây dựng các ứng dụng Mobile, Web và Desktop app đẹp, được biên dịch nguyên bản từ một cơ sở mã code duy nhất.

Dart là ngôn ngữ lập trình đa mục đích ban đầu được phát triển bởi Google và sau đó được Ecma (ECMA-408) phê chuẩn làm tiêu chuẩn. Nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động. Dart là một ngôn ngữ hướng đối tượng, được xác định theo lớp, với cơ chế garbage-collected, sử dụng cú pháp kiểu C để dịch mã tùy ý sang JavaScript. Nó hỗ trợ interface, mixin, abstract, generic, static typing và sound type

Hình 17: Ngôn ngữ Dart

2.1.2.2. Lịch sử hình thành

Dart được tiết lộ lần đầu tiên trong hội nghị GOTO vào tháng 10 – 12 tháng 10 năm 2011 tại Aarhus, Đan Mạch. Ban đầu nó được thiết kế bởi Lars và Kespar và được phát triển bởi Google.

- Phiên bản 1.0 đầu tiên của Dart được phát hành vào ngày 14 tháng 11 năm 2013, nhằm mục đích thay thế JavaScript.
- Vào tháng 7 năm 2014, ấn bản đầu tiên của ngôn ngữ Dart đã được Ecma International thông qua tại Đại hội đồng lần thứ 107 của tổ chức này.

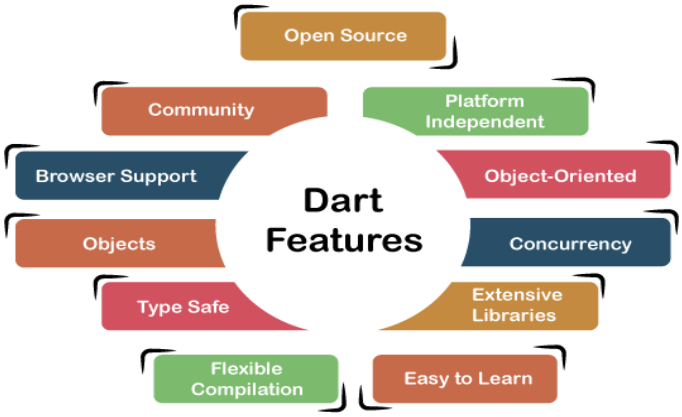
- Phiên bản đầu tiên đã bị chỉ trích do sự cố trên web và kế hoạch này đã bị loại bỏ vào năm 2015 với bản phát hành 1.9 của Dart.
- Phiên bản thứ hai của Dart 2.0 được phát hành vào tháng 8, bao gồm một hệ thống âm thanh.
- Phiên bản gần đây Dart 2.7 được bổ sung thêm phương thức mở rộng, cho phép thêm bất kỳ loại chức năng nào.

2.1.2.3. Đặc điểm của Dart

- Dart là một ngôn ngữ độc lập với nền tảng và hỗ trợ tất cả các hệ điều hành như Windows, Mac, Linux,…
- Nó là một ngôn ngữ mã nguồn mở, có nghĩa là nó có sẵn miễn phí cho tất cả mọi người. Nó đi kèm với giấy phép BSD và được công nhận bởi tiêu chuẩn ECMA.
- Nó là một ngôn ngữ lập trình hướng đối tượng và hỗ trợ tất cả các tính năng của oops như kế thừa, giao diện và các tính năng kiểu tùy chọn.
- Dart rất hữu ích trong việc xây dựng các ứng dụng thời gian thực vì tính ổn định.
- Dart đi kèm với trình biên dịch dar2js để truyền mã Dart thành mã JavaScript chạy trên tất cả các trình duyệt web hiện đại.
- Máy ảo Dart độc lập cho phép mã Dart chạy trong môi trường giao diện dòng lệnh.

2.1.2.4. Tính năng của Dart

Dart là một ngôn ngữ lập trình hướng đối tượng, mã nguồn mở, chứa nhiều tính năng hữu ích. Đây là ngôn ngữ lập trình mới và hỗ trợ một loạt các tiện ích lập trình như giao diện, bộ sưu tập, lớp, kiểu gõ động và tùy chọn. Nó được phát triển cho máy chủ cũng như trình duyệt.



Hình 18: Tính năng của Dart

- **Mã nguồn mở :** Dart là một ngôn ngữ lập trình mã nguồn mở, có nghĩa là nó có sẵn miễn phí. Nó được phát triển bởi Google, được phê duyệt bởi tiêu chuẩn ECMA và đi kèm với giấy phép BSD.
- **Nền tảng độc lập :** Dart hỗ trợ tất cả các hệ điều hành chính như Windows, Linux, Macintosh, v.v. Dart có Máy ảo riêng được gọi là Dart VM, cho phép chúng tôi chạy mã Dart trong mọi hệ điều hành.
- **Hướng đối tượng :** Dart là một ngôn ngữ lập trình hướng đối tượng và hỗ trợ tất cả các khái niệm oops như lớp, kế thừa, giao diện và các tính năng gõ tùy chọn. Nó cũng hỗ trợ các khái niệm nâng cao như mixin, abstract, các lớp, hệ thống kiểu chung được sửa đổi và mạnh mẽ.
- **Đồng nhất :** Dart là một ngôn ngữ lập trình không đồng bộ, có nghĩa là nó hỗ trợ đa luồng sử dụng Isolates. Các vùng cách ly là các thực thể độc lập có liên quan đến các luồng nhưng không chia sẻ bộ nhớ và thiết lập giao tiếp giữa các quá trình bằng cách truyền thông điệp. Thông điệp nên được nối tiếp nhau để tạo hiệu quả truyền thông. Việc tuần tự hóa thông báo được thực hiện bằng cách sử dụng một ảnh chụp nhanh được tạo ra bởi đối tượng đã cho và sau đó truyền đến một vùng cách ly khác để giải mã.
- **Thư viện mở rộng :** gồm nhiều thư viện tích hợp hữu ích bao gồm SDK (Bộ phát triển phần mềm), lõi, toán học, không đồng bộ, toán học, chuyển đổi, html, IO, v.v. Nó cũng cung cấp cơ sở để tổ chức mã Dart thành các thư viện với không gian tên riêng. Nó có thể sử dụng lại bằng câu lệnh nhập.

- **Dễ học :** Cú pháp của Dart tương tự như Java, C #, JavaScript, kotlin, v.v. nếu bạn biết bất kỳ ngôn ngữ nào trong số này thì bạn có thể học Dart dễ dàng.
- **Biên dịch linh hoạt :** Nó hỗ trợ hai loại quy trình biên dịch, AOT (Ahead of Time) và JIT (Just-in-Time). Mã Dart được truyền bằng ngôn ngữ khác có thể chạy trong các nhà sản xuất web hiện đại.
- **Nhập an toàn :** Dart là ngôn ngữ an toàn kiểu, có nghĩa là nó sử dụng cả kiểm tra kiểu tĩnh và kiểm tra thời gian chạy để xác nhận rằng giá trị của một biến luôn khớp với kiểu tĩnh của biến, đôi khi nó được gọi là kiểu gõ âm thanh. Mặc dù loại là bắt buộc, nhưng chú thích loại là tùy chọn vì loại nhiễu. Điều này làm cho mã dễ đọc hơn. Ưu điểm khác của ngôn ngữ an toàn kiểu chữ là khi chúng ta thay đổi phần mã, hệ thống sẽ cảnh báo chúng ta về sửa đổi mà chúng ta đã sửa trước đó.
- **Hỗ trợ trình duyệt :** Dart hỗ trợ tất cả các trình duyệt web hiện đại. Nó đi kèm với trình biên dịch dart2js để chuyển đổi mã Dart thành mã JavaScript được tối ưu hóa phù hợp với tất cả các loại trình duyệt web.

2.1.2.5. Đối tượng

Trong vài năm gần đây, Flutter là một nền tảng mới nổi, có thể viết ứng dụng cho cả iOS & Android và phát triển được với Web hay Desktop.

2.1.2.6. Flutter sử dụng Dart

Flutter sử dụng Dart vì Dart cho phép Flutter tránh được sự cần thiết của một ngôn ngữ bố cục khai báo riêng biệt như JSX và XML. Bố cục của Dart là khai báo và có lập trình, và nó giúp các nhà phát triển dễ dàng đọc và hình dung nó một cách nhanh chóng và dễ dàng. Ngoài ra, Flutter có thể dễ dàng cung cấp thêm công cụ vì bố cục bằng một ngôn ngữ và thông dụng.

2.2. Kỹ thuật

2.2.1. Android Studio

2.2.1.1. Khái niệm

Android Studio là Môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android. Nhờ có công cụ cho nhà phát triển và trình soạn thảo mã mạnh mẽ của

IntelliJ IDEA, Android Studio cung cấp thêm nhiều tính năng giúp bạn nâng cao năng suất khi xây dựng ứng dụng Android, chẳng hạn như:

- Một hệ thống xây dựng linh hoạt dựa trên Gradle
- Một trình mô phỏng nhanh và nhiều tính năng
- Một môi trường hợp nhất nơi bạn có thể phát triển cho mọi thiết bị Android
- Tính năng Live Edit (Chỉnh sửa trực tiếp) để cập nhật các thành phần kết hợp trong trình mô phỏng và thiết bị thực theo thời gian thực
- Mã mẫu và quá trình tích hợp GitHub để giúp bạn xây dựng các tính năng ứng dụng phổ biến cũng như nhập mã mẫu
- Đa dạng khung và công cụ thử nghiệm
- Công cụ tìm lỗi mã nguồn (lint) để nắm bắt hiệu suất, khả năng hữu dụng, khả năng tương thích với phiên bản và các vấn đề khác
- Hỗ trợ C++ và NDK
- Tích hợp sẵn tính năng hỗ trợ Google Cloud Platform, giúp dễ dàng tích hợp Google Cloud Messaging và App Engine

Hình 19: Android Studio

2.2.1.2. Lịch sử hình thành Android Studio

Android Studio đã được công bố vào năm 2013 tại hội nghị Google I/O và được phát hành vào năm 2014 sau nhiều phiên bản khác nhau.

Trước đó, thì các nhà phát triển của Android thường sử dụng các công cụ như Eclipse IDE hoặc một IDE Java chung để hỗ trợ cho nhiều ngôn ngữ lập trình khác. Android Studio giúp cho việc tạo ứng dụng dễ dàng hơn so với các phần mềm chuyên dụng.

Với người mới, sẽ có rất nhiều thứ phải học và nhiều thông tin có sẵn. Thậm chí, chúng còn thông qua nhiều kênh chính thức hoặc có thể có lỗi khiến người dùng hoang mang.

2.2.1.3. Tính năng của Android Studio

- Build được các biến và tạo được nhiều file APK.
- Code of template to support are features of the information app.
- Gradle support – based một cách linh hoạt.
- Với GitHub tích hợp giúp bạn xây dựng được các ứng dụng tính năng một cách phổ biến.
- Chỉnh sửa được bố cục một cách đa dạng với các hoạt động kéo thả linh hoạt.
- Tích hợp lên google Cloud Platform, giúp bạn có thể dễ dàng tích hợp được app engine và google cloud Messaging.
- Giúp mô phỏng được phần mềm để tiến hành sửa chữa và nâng cấp được các sản phẩm khi cần.
- Các trình soạn thảo mã và công cụ Intell cung cấp các tính năng cao
- Instant Run giúp thay đổi các ứng dụng đang chạy mà không cần xây dựng APK mới.
- Hỗ trợ được C++ và NDK.
- Giúp Sâu firebase và các ứng dụng sau click chuột.
- Công cụ build dựa trên Gradle.
- Các wizard tích hợp giúp các lập trình viên có thể tạo ứng dụng từ các mẫu có sẵn.
- Chức năng dò và sửa lỗi nhanh để hướng Android.

2.2.1.4. Ưu điểm

- Được phát triển bởi Google, cũng là chủ sở hữu của hệ điều hành Android
- Các công cụ hỗ trợ và được cập nhật mới nhất và đầy đủ
- Tính năng dễ làm quen và giao diện thân thiện, nó là điểm cộng lớn.
- Có tài liệu tham khảo và hướng dẫn đầy đủ cùng các diễn đàn dành cho lập trình viên Android.
- Được đào tạo thông qua các khóa học lập trình Android cơ bản, nâng cao, khóa học Tester

2.2.1.5. Nhược điểm

- Là công cụ hỗ trợ tích hợp tất cả nên dữ liệu phải phát triển tối ưu nhất. Lượng lớn dữ liệu chiếm nhiều không gian bộ nhớ máy tính của bạn.
- Có thể kiểm tra được cash hoạt động của app thông qua giả lập của Android studio. Điều này làm đơ máy, lag, nóng máy tính và gây tiêu tốn nhiều pin.
- Android là một công cụ lập trình hỗ trợ mạnh mẽ với các hỗ trợ và cập nhật các tính năng mới nhất từ google. Google đã khắc phục vấn đề tối ưu tài nguyên máy tính giúp giảm bớt Android Studio trên máy tính cũ.

2.2.2. SQLite

2.2.2.1. Khái niệm

SQLite là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) tương tự như Mysql, SQL Server, Ocracle, PostgreSQL ..., được phát triển bởi Richard Hipp vào năm 2000. Đặc điểm nổi bật của SQLite so với các DBMS khác là gọn, nhẹ, đơn giản, đặt biệt không cần mô hình server-client, không cần cài đặt, cấu hình hay khởi động nên không có khái niệm user, password hay quyền hạn trong SQLite Database. Dữ liệu cũng được lưu ở một file duy nhất. SQLite được sử dụng vào rất nhiều chương trình từ desktop đến mobile hay là website.

SQLite thường không được sử dụng với các hệ thống lớn nhưng với những hệ thống ở quy mô vùa và nhỏ thì SQLite không thua các DBMS khác về chức năng hay tốc độ. Vì không cần cài đặt hay cấu hình nên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm … vì tránh được những rắc rối trong quá trình cài đặt.

Hình 20: SQLite

2.2.2.2. Lịch sử hình thành

- Năm 2000: Phiên bản đầu tiên của SQLite (phiên bản 1.0) được phát hành vào ngày 17 tháng 8 năm 2000. SQLite 1.0 đã hỗ trợ một số tính năng cơ bản của cơ sở dữ liệu quan hệ và là một phiên bản đơn giản. D.Richard Hipp đã thiết kế SQLite dưới dạng thư viện bằng ngôn ngữ lập trình C với mục đích không cần quản trị để điều hành một chương trình.
- Năm 2004: SQLite 2.0 ra mắt với nhiều cải tiến, bao gồm hỗ trợ cho các kiểu dữ liệu thêm vào và giao diện ứng dụng nhiều luồng.
- Năm 2009: SQLite 3.6.11 được phát hành, hỗ trợ giao dịch và các tính năng mới liên quan đến an toàn cơ sở dữ liệu.
- Năm 2010: SQLite 3.7.0 giới thiệu một số tính năng mới như cơ chế WRITE-AHEAD LOGGING (WAL) cho hiệu suất ghi tăng và cải thiện khả năng đồng thời.
- Năm 2011: Hipp bổ sung UNQI Interface cho SQLite DB và để phát triển UNQLite (là một Document Oriented Database).
- Năm 2013: SQLite 3.8.0 đưa ra tính năng "partial indexes", giúp tối ưu hóa truy vấn và giảm không gian lưu trữ cho các chỉ mục.
- Năm 2016: Phiên bản 3.15.0 của SQLite thêm tích hợp mở rộng JSON1, hỗ trợ các truy vấn và xử lý dữ liệu JSON.
- Năm 2020: SQLite 3.31.0 đưa ra tính năng mới "incremental BLOB I/O" để cải thiện hiệu suất ghi và đọc các đối tượng nhị phân lớn.

2.2.2.3. Tính năng

- Giao dịch trong SQLite tuân thủ theo nguyên tắc (ACID) ngay cả sau khi hệ thống treo và mất điện.
- Không cấu hình: Không cần thiết lập hoặc quản trị
- SQLite hỗ trợ với đầy đủ tính năng với các khả năng nâng cao như các chỉ mục 1 phần, các chỉ mục về các biểu thức, JSON và các biểu thức bảng chung.
- Một sở dữ liệu hoàn chỉnh được lưu trữ trong một tệp đa nền tảng duy nhất. Phù hợp với sử dụng dưới dạng định dạng tệp ứng dụng
- Hỗ trợ các cơ sở dữ liệu có kích thước terabyte và các chuỗi có kích thước gigabyte.

- API:
    - Đơn giản dễ sử dụng
    - Nhanh: Trong một số trường hợp, SQLite nhanh hơn hệ thống tệp tin trực tiếp I/O.
- Được viết bằng ANSI-C: Bindings cho hàng chục ngôn ngữ khác có sẵn 1 cách riêng biệt.
- Mã nguồn đầy, nguồn mở đủ có thể kiểm tra nhánh 100%.

Nền tảng đa nền tảng: SQLite là có sẵn trên Android, *BSD, iOS, Linux, Mac, Solaris, Windows,.. Dễ dàng dịch chuyển sang các hệ thống khác

2.2.2.4. Ứng dụng chủ yếu

- **Cơ sở dữ liệu cho Internet Of Things**

SQLite là lựa chọn phổ biến cho các công cụ cơ sở dữ liệu trong điện thoại di động, PDA, máy nghe nhạc mp3, hộp set-top, và các tiện ích điện tử khác.

- **Định dạng tệp ứng dụng**

Thay vì sử dụng fopen() để viết XML, JSON, CSV hoặc một số định dạng động quyền vào các tệp đĩa được ứng dụng của bạn sử dụng, hãy sử dụng SQLite.

- **Cơ sở dữ liệu cho web**

Bởi vì SQLite không yêu cầu cấu hình và lưu trữ thông tin trong các tệp đĩa thông thường nên SQLite là lựa chọn phổ biến làm cơ sở dữ liệu để quay lại các trang web vừa và nhỏ.

- **Stand-in cho một doanh nghiệp RDBMS**

SQLite được sử dụng như một RDBMS doanh nghiệp cho mục đích trình diễn hoặc để thử nghiệm vì SQLite nhanh và không yêu cầu thiết lập.

2.2.2.5. Ưu điểm

- SQLite không cần mô hình client – server để hoạt động.
- SQLite không cần phải cấu hình tức là bạn không cần phải cài đặt.
- Với SQLite database được lưu trữ trên một tập tin duy nhất.
- SQLite hỗ trợ hầu hết các tính năng của ngôn ngữ truy vấn SQL theo chuẩn SQL92.
- SQLite rất nhỏ gọn bản đầy đủ các tính năng nhỏ hơn 500kb, và có thể nhỏ hơn nếu lược bớt một số tính năng.

- Các thao tác dữ liệu trên SQLite chạy nhanh hơn so với các hệ quản trị cơ sở dữ liệu theo mô hình client – server.
- SQLite rất đơn giản và dễ dàng sử dụng.
- SQLite tuân thủ 4 tính chất ACID (là tính nguyên tổ (Atomic), tính nhất quán (Consistent), tính cô lập (Isolated), và tính bền vững (Durable)).

Với đặc tính nhỏ gọn, truy xuất dữ liệu nhanh SQLite thường được sử dụng để nhúng vào các dự án.

2.2.2.6. Nhược điểm

- Do sử dụng cơ chế coarse-gained locking nên trong cùng một thời điểm SQLite có thể hỗ trợ nhiều người đọc dữ liệu, nhưng chỉ có 1 người có thể ghi dữ liệu.
- SQLite không phải là lựa chọn hoàn hảo để đáp ứng các nhu cầu xử lý trên một khối lượng dữ liệu lớn, phát sinh liên tục.
- SQLite cũng thiếu các tính năng đo lường hiệu suất

2.2.2.7. Cú pháp trong SQLite

SQLite hỗ trợ gần như đầy đủ các cú pháp trong chuẩn SQL92.

Dưới đây là một số câu lệnh thường được sử dụng

STT	Cú pháp	Ý nghĩa
1	sqlite3 <name.db>	Tạo database
2	ATTACH DATABASE '<databasename>' As '<alias-name>';	Sử dụng database, có thể đặt alias cho database và sử dụng như tên của database
3	DETACH DATABASE '<name-name>';	Xóa cơ sở dữ liệu sử dụng với tên alias
4	CREATE TABLE <databasename.tablename>();	Tạo bảng
5	DROP TABLE database_name.table_name;	Xóa bảng

STT	Cú pháp	Ý nghĩa
6	INSERT INTO table_name [(column1, column2,..)] VALUES (value1, value2,..);	Thêm dữ liệu vào bảng
7	INSERT INTO table1 [(column…)] SELECT column FROM table2 [WHERE];	Chèn dữ liệu vào bảng từ một bảng khác
8	SELECT sql FROM table;	Hiển thị thông tin bảng
9	SELECT (12+8) AS ADDITION; #20	Thực hiện biểu thức số học
10	SELECT COUNT(*) AS “RECORDS” FROM table;	đếm bảng ghi trong bảng
11	SELECT CURRENT_TIMESTAMP;	Hiển thị thời gian hệ thống
12	UPDATE table_name SET column1 = value,... WHERE ..;	Update dữ liệu bảng
13	DELETE FROM table_name WHERE …;	Xóa bản ghi
14	PRAGMA pragma_name;	Điều khiển các biến môi trường đa dạng
15	PRAGMA pragma_name = value;	Thiết lập giá trị
16	SELECT ... FROM table1 CROSS JOIN table2 ...	Kết nối bảng đầu tiên với bảng thứ hai
17	SELECT ... FROM table1 [INNER] JOIN table2 ON conditional_expression ...	INNER JOIN
18	SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional_expression ...	OUTER JOIN: chỉ hỗ trợ LEFT JOIN

2.2.3. Sqlite trong Flutter

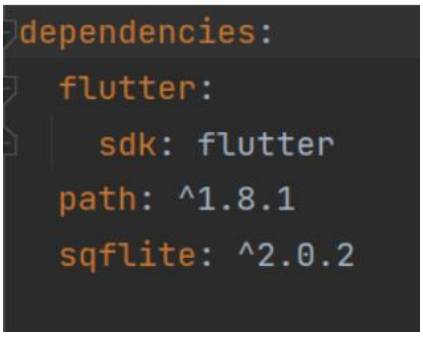
Trong Flutter chúng ta sử dụng thư viện *SQLite*, thư viện này sẽ hỗ trợ chúng ta:

- Sử dụng trên đa nền tảng Android, iOS, MacOS.
- Hỗ trợ transactions và batches.
- Tự động quản lý phiên bản (version) trong khi mở.
- Hỗ trợ các câu lệnh truy vấn CRUD (Create – Read - Update - Delete)
- Thực hiện các xử lý trên background của iOS và Android.

2.2.4. Cách sử dụng Sqlite trong Flutter

2.2.4.1. Thêm thư viện

Thêm thư viện sqflite và path (dùng để xác định vị trí lưu database trong bộ nhớ) vào phần dependencies trong file pubspec.yaml



Hình 21: Thêm thư viện

2.2.4.2. Cài đặt thư viện

Bước 1: Cần tạo một class model Student, đây sẽ là dữ liệu chúng ta dùng trong quá trình lưu trữ.

```
class Student {
  int id;
  String name;
  int grade;

  Student({required this.id, required this.name, required this.grade});
}
```

Hình 22: Tạo một class model Student

Bước 2: Khởi tạo database

```
class StudentDatabase {
  static Database? _database;

  static Future<Database> getInstance() async {
    _database ??= await openDatabase(

        /// use join to create path for db, then the path will be path/student.db
        join(await getDatabasesPath(), "student.db"),

        /// This function will be called in the first time database is created
        onCreate: (db, version) {
      return db.execute(
          "CREATE TABLE student(id INTEGER PRIMARY KEY, name TEXT, grade INTEGER)");
    },

        /// This version will use when you want to upgrade or downgrade the database
        version: 1,
        singleInstance: true);
    return _database!;
  }
}
```

Hình 23: Khởi tạo database

- Hàm *onCreate()* được gọi tại lần đầu mà database được khởi tạo
- *Version* chính là phiên bản của database, nếu bạn muốn thay đổi cấu trúc của *database* thì chúng ta phải thay đổi *version*.

Thêm các hàm phục vụ cho việc chuyển đổi dữ liệu khi muốn thêm vào database cũng như lấy dữ liệu ra từ database

```
class Student{
...

Map<String, dynamic> toMap() => {'id': id, 'name': name, 'grade': grade};

factory Student.fromMap(Map<String, dynamic> map) {
  return Student(id: map['id'], name: map['name'], grade: map['grade']);
}
...
}
```

Hình 24: Thêm các hàm phục vụ cho việc chuyển đổi dữ liệu

Bước 3: Thêm một trường dữ liệu vào bảng

```
Future<DataResult> insertStudent(Student student, String tableName) async {
    try{
      Database db = await StudentDatabase.getInstance();
      int lastInsertedRow = await db.insert(tableName, student.toMap(),
          conflictAlgorithm: ConflictAlgorithm.replace);
      return DataResult.success(lastInsertedRow);
    }catch(ex){
      return DataResult.failure(DatabaseFailure(ex.toString()));
    }
  }
```

Hình 25: Thêm một trường dữ liệu vào bảng

- Hàm *insert()* có hai tham số truyền vào là tên của bảng và dữ liệu chúng ta muốn thêm vào bảng, dữ liệu này đã được chuyển thành một map để có thể thực hiện thêm vào *database*.
- Tham số *conflictAlgorithm*: được sử dụng để xác định các xử lý khi có sự trùng lặp dữ liệu xảy ra, ở đây tham số này có giá trị là *ConflictAlgorithm.replace* có nghĩa là nếu trùng primary key thì giá trị cũ sẽ được thay thế bằng giá trị mới.
- Nếu hàm *insert* thực hiện thành công sẽ trả về id của hàng vừa được thêm.
- Nếu không thành công sẽ trả về giá trị 0.

Bước 4: Lấy thông tin của tất cả các trường

Trong Android Studio có một công cụ hỗ trợ xem được các cơ sở dữ liệu đang có trong app, đó chính là App Inspection.

```
Future<DataResult> getAllStudent() async{
    try{
      Database db = await StudentDatabase.getInstance();
      final List<Map<String,dynamic>> maps = await db.query("student");
      List<Student> students = maps.map((e) => Student.fromMap(e)).toList();
      return DataResult.success(students);
    }catch(ex){
      return DataResult.failure(DatabaseFailure(ex.toString()));
    }
  }
```

Hình 26: Lấy thông tin của tất cả các trường

cần sử dụng hàm query, truyền vào tham số là bảng mà chúng ta cần lấy dữ liệu.

- Dữ liệu trả về sẽ ở dạng *List<Map<String,dynamic>>*, vì vậy chúng ta cần chuyển đổi chúng sang dạng của lớp Student sử dụng hàm *fromMap()* đã được thêm trong class Student.

Bước 5: Thay đổi thông tin của một trường.

```
Future<DataResult> updateStudent(Student student) async{
    try{
      Database db = await StudentDatabase.getInstance();
      int numberOfChanges = await db.update("student", student.toMap(),where: "id = ?",whereArgs: [student.id]);
      return DataResult.success(numberOfChanges);
    }catch(ex){
      return DataResult.failure(DatabaseFailure(ex.toString()));
    }
  }
```

Hình 27: Thay đổi thông tin của một trường

Để thực hiện được việc thay đổi dữ liệu của một trường chúng ta cần sử dụng hàm *update()*

- Các tham số truyền vào gồm có tên bảng, dữ liệu thay đổi (được chuyển sang *map*), điều kiện thay đổi và tham số cho điều kiện đó.
- Có thể thấy phần thay đổi dữ liệu này khá giống với phần thêm dữ liệu, chỉ khác chúng ta có thêm hai tham số khá quan trọng là "where" và "whereArgs".
- Giá trị trả về của hàm update() là số lượng thay đổi đã diễn ra.

Bước 6: Xóa thông tin của một trường

```
Future<DataResult> deleteStudent(int id) async{
    try{
      Database db = await StudentDatabase.getInstance();
      int numberOfRowEffected = await db.delete("student",where: "id = ?",whereArgs: [id]);
      return DataResult.success(numberOfRowEffected);
    }catch(ex){
      return DataResult.failure(DatabaseFailure(ex.toString()));
    }
  }
```

Hình 28: Xóa thông tin của một trường

Để thực hiện việc xóa dữ liệu chúng ta sẽ cần dùng đến hàm *delete()*

- Gần giống với hàm update chúng ta cũng cần có hai tham số "where" và "whereArgs"
- Giá trị trả về sẽ là hàng bị thay đổi giá trị.

2.2.5. Firebase

2.2.5.1. Khái niệm

Firebase là một nền tảng giúp phát triển các ứng dụng di động trong web. Bên cạnh đó, Firebase còn được hiểu là một dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây cloud với hệ thống máy chủ mạnh mẽ của Google.

Cụ thể là những giao diện lập trình ứng dụng API đơn giản. Mục đích nhằm tăng số lượng người dùng và thu lại nhiều lợi nhuận hơn.

Hình 29: Firebase

2.2.5.2. Lịch sử hình thành và phát triển

- Vào năm 2011, James Tamplin và Andrew Lee đã cho ra mắt Evolve. Đây là một nền tảng có cấu trúc khá đơn giản chuyên cung cấp các API cần thiết để tích hợp tính năng trò chuyện vào các trang web. Tuy nhiên, họ nhận ra rằng nền tảng này đang được sử dụng để truyền dữ liệu ứng dụng chứ không đơn giản là chat. Sau đó, họ đã phát triển Envole và tạo nên Firebase.
- Đến tháng 4 năm 2012, Firebase đã được công bố trên toàn cầu dưới dạng một công ty riêng biệt. Những năm sau đó, Firebase đã thực hiện nhiều cuộc huy động vốn và ra mắt các sản phẩm mới.
- Đến tháng 10 năm 2014, Firebase đã chính thức được Google mua lại và trở thành một ứng dụng đa năng trên nền tảng di động và website.

2.2.5.3. Đặc điểm

- Firebase RealTime Database là một cơ sở dữ liệu NoSQL dưới dạng lưu trữ đám mây cho phép bạn lưu trữ và đồng bộ dữ liệu. Firebase thuộc sở hữu của Google.
- Khi bạn có một tài khoản Google thì đồng nghĩa bạn có một tài khoản Firebase
- Khi phát triển Mobile App sử dụng Firebase, tất cả các client sẽ chia sẻ một phiên bản Realtime Database, từ đó có thể tự động cập nhật dữ liệu mới nhất.

2.2.5.4. FireStore

- Cloud FireStore là cơ sở dữ liệu mới của Firebase phát triển cho các ứng dụng di động.
- Là sự kế thừa của Realtime Database với mô hình dữ liệu mới và trực quan hơn. Cloud Firestore phong phú hơn, nhanh hơn và có khả năng mở rộng siêu việt hơn so với Realtime Database.
- Firestore được sử dụng trong Firebase

Hình 30: FireStore

2.2.5.5. So sánh SQLite và Firebase

- SQLite là mã nguồn mở và Firebase là công nghệ độc quyền
- Firebase là nền tảng phát triển toàn diện
- SQLite là cơ sở dữ liệu
- Firebase cung cấp dịch vụ lưu trữ
- SQLite miễn phí tải xuống

Firebase	SQLite
Cung cấp cơ sở dữ liệu thời gian thực	SQLite là một hệ thống quản lý cơ sở dữ liệu quan hệ được nhúng trong quá trình (RDBMS).
Một nền tảng nguồn đóng	Một nền tảng mã nguồn mở.
Cloud Firestore và Cơ sở dữ liệu thời gian thực là tài liệu NoSQL và cơ sở dữ liệu đám mây.	SQLite là cơ sở dữ liệu Ngôn ngữ truy vấn có cấu trúc (SQL).
Đây là một nền tảng được lưu trữ trên đám mây.	Đây là một thư viện cơ sở dữ liệu serverless.
Thích hợp cho cả ứng dụng iOS và Android.	Thích hợp hơn cho các ứng dụng Android.
Nó là một nền tảng trả phí với các tùy chọn cấp miễn phí.	Nó là một công cụ cơ sở dữ liệu hoàn toàn miễn phí không cần bất kỳ giấy phép nào.

2.2.5.6. Các tính năng cốt lõi của Firebase

- **Cơ sở dữ liệu Firebase**

Firebase lưu trữ dữ liệu trên hai cơ sở dữ liệu quan trọng là Cơ sở dữ liệu thời gian thực và Cloud Firestore.

Đầu tiên, nếu nói về Firebase Realtime Database, nó là cơ sở dữ liệu NoSQL giúp tạo các ứng dụng serverless cho người dùng toàn cầu. Tự kết nối với Xác thực Firebase để bảo mật người dùng và có khả năng tự động đồng bộ hóa dữ liệu ngoại tuyến khi người dùng trực tuyến.

Tương tự, Cloud Firestore là một loại tài liệu của cơ sở dữ liệu NoSQL có thể sắp xếp dữ liệu với các bộ sưu tập.

Các nhà phát triển có thể dễ dàng tạo và truy vấn dữ liệu ở đây theo mong muốn của họ. Giống như cơ sở dữ liệu Thời gian thực, bạn cũng có thể kết nối cơ sở dữ liệu này một cách thuận tiện với Xác thực, Hàm đám mây và các thư viện SDK khác.

- **Lưu trữ đám mây**

Firebase cung cấp SDK tích hợp để tải xuống và tải lên nội dung một cách nhanh chóng. Nó cho phép các nhà phát triển tiếp tục và giữ các chuyển khoản này chỉ trong vài cú nhấp chuột.

Đồng thời, Firebase cũng có thể trình bày và thu thập nội dung do người dùng tạo một cách mạnh mẽ với sự hỗ trợ của Google Cloud.

- **Lưu trữ**

Cho dù bạn muốn lưu trữ một trang tĩnh một trang hay một ứng dụng web động, bạn có thể dễ dàng lưu trữ với Dịch vụ lưu trữ Firebase.

Các lập trình viên có thể kết nối dịch vụ lưu trữ với Chức năng đám mây để phục vụ và tạo nội dung động. Bộ nhớ được hỗ trợ bởi SSD và chứng chỉ SSL miễn phí cũng là những lợi ích khi sử dụng Dịch vụ lưu trữ Firebase.

- **Xác thực**

Người dùng và nhà phát triển ứng dụng có thể nhận được xác thực an toàn với danh tính đầu cuối. Người dùng có thể đăng nhập qua Twitter, Facebook, điện thoại, Auth, Google và GitHub để truy cập ứng dụng của bạn.

Ngược lại, các trang xác thực cũng có thể được tùy chỉnh đẹp mắt với giao diện người dùng thả vào do Google hỗ trợ. Có, nó là mã nguồn mở và hoàn toàn có thể tùy chỉnh.

- **Firebase Pricing**

Firebase là một nền tảng trả phí đi kèm với tùy chọn bậc miễn phí. Có, Spark Plan của nó hoàn toàn miễn phí sử dụng với nguồn lực hạn chế.

Các công ty khởi nghiệp có thể truy cập 1 GiB để lưu trữ, 10 GiB cho đầu ra mạng hàng tháng và 20.000 lần ghi tài liệu hàng ngày của Cloud Firestore mà không phải trả bất kỳ khoản phí nào. Hơn nữa, bạn cũng có thể lưu trữ và tải xuống dữ liệu 1 và 10 GB cho Cơ sở dữ liệu thời gian thực theo gói Spark này.

2.2.5.7. Các tính năng cốt lõi của SQLite

- **Không có máy chủ**

So với các cơ sở dữ liệu khác như PostgreSQL và MySQL, RDBMS này không cần một máy chủ độc lập. Có, SQLite dựa trên kiến trúc máy khách/máy chủ RDBMS nơi giao thức TCP/IP được tuân theo để phát hành và nhận yêu cầu.

Trong môi trường không có máy chủ này, các ứng dụng trực tiếp viết và đọc hướng dẫn từ các tệp DB có sẵn trên đĩa. Các cơ sở dữ liệu serverless này có năng suất cao, tiết kiệm chi phí và có thể mở rộng.

- **Không có cấu hình**

Với thuộc tính không cấu hình, các nhà phát triển không cần cài đặt công cụ cơ sở dữ liệu này để lưu trữ và truy vấn dữ liệu. SQLite cũng không phụ thuộc vào các tệp cấu hình cho các quy trình tiếp theo.

- **Nguồn mở & Đa nền tảng**

SQLite là một nền tảng nguồn mở và bạn có thể tìm thấy một số tài nguyên hữu ích trực tuyến. Nó đặc biệt hữu ích cho các công ty khởi nghiệp làm việc với các nhóm mới và muốn có cơ sở dữ liệu nguồn mở để dễ dàng xử lý.

Tương tự như vậy, SQLite là một DBMS đa nền tảng cho phép các lập trình viên xây dựng cơ sở dữ liệu trên một máy và sử dụng nó trên một số máy khác. Cơ sở dữ liệu này có khả năng chạy trên máy 64-bit và 32-bit.

- **Khép kín**

Độc lập là một tính năng nổi bật khác của cơ sở dữ liệu SQLite. Điều đó có nghĩa là các nhà phát triển không phải phụ thuộc vào các công cụ và thư viện của bên thứ ba khi sử dụng công cụ cơ sở dữ liệu SQLite. Về vấn đề này, mã nguồn ANSI-C cũng đóng một vai trò quan trọng.

- **Giao dịch**

SQLite cũng có thuộc tính ACID (tính nguyên tử, tính nhất quán, tính cách ly, độ bền) để đảm bảo tính ổn định của dữ liệu trong trường hợp có bất kỳ rủi ro nào. Có, trong các tình huống như sự cố ứng dụng, lỗi và mất điện, các giao dịch vẫn bền vững, nguyên tử, nhất quán và tách biệt.

2.2.6. Bloc

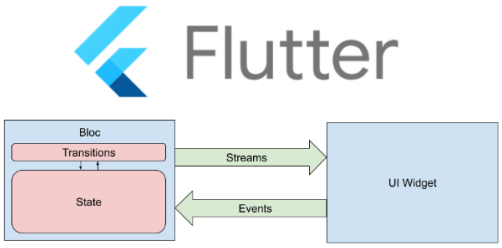
2.2.6.1. Khái niệm

Bloc là một lib để quản lý state cho Flutter application. Được tạo bởi Google và ra mắt 2018, được viết dựa trên Stream và Reactive Programing. B.L.o.C nghĩa là Business Logic Component. Nhận 'Event' như là input và trả về output là 'State'. Bloc được xây dựng dựa trên RxDart.

Chúng ta có thể chia Flutter application architecture thành 3 lớp sau:

- Biển diễn của View Layer : Có nhiệm vụ render chính nó dựa trên một hoặc nhiều state của bloc. Nó cũng xử lý các sự kiện input của user và lifecycle event của application.
- Bloc : Có nhiệm vụ nhận event từ lớp biểu diễn và trả về state mới. Hoạt động như một cầu nối giữa lớp data và lớp presentation.
- Data Layer : Có nhiệm vụ cung cấp data từ bất kể nguồn nào. Data Provider cung cấp data tho và repository sẽ là trình đóng gói một hoặc nhiều data providers.

Vậy có nghĩa là Bloc đứng giữa 2 lớp View và Data.



Hình 31: Bloc

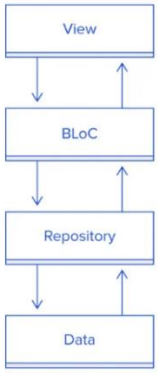
2.2.6.2. Kiến trúc BLoC

Kiến trúc BLoC dựa trên khái niệm lập trình phản ứng , trong đó trạng thái của ứng dụng được biểu diễn dưới dạng một luồng sự kiện. Nói cách khác, kiến trúc BLoC là về việc quản lý trạng thái của ứng dụng và phản ứng với những thay đổi trong trạng thái đó. Kiến trúc BLoC bao gồm ba thành phần chính:

- **Thành phần logic nghiệp vụ (BLoC)**: Đây là nơi chứa logic nghiệp vụ của ứng dụng của bạn. BLoC nhận các sự kiện từ giao diện người dùng và xử lý chúng. Sau đó, nó sẽ gửi các trạng thái mới tới giao diện người dùng, giao diện này sẽ cập nhật tương ứng.

- **Giao diện người dùng (UI):** Đây là lớp trình bày của ứng dụng của bạn. Nó nhận các trạng thái từ BLoC và tự cập nhật cho phù hợp.
- **Luồng sự kiện:** Luồng sự kiện này chảy từ giao diện người dùng đến BLoC. Nó thể hiện sự tương tác của người dùng với ứng dụng.

Ý tưởng chính đằng sau kiến trúc BLoC là tách logic nghiệp vụ khỏi giao diện người dùng, giúp ứng dụng trở nên mô-đun hơn và dễ bảo trì hơn. Với kiến trúc BLoC, bạn cũng có thể dễ dàng kiểm tra logic nghiệp vụ của ứng dụng một cách độc lập.



Hình 32: Kiến trúc BLoC

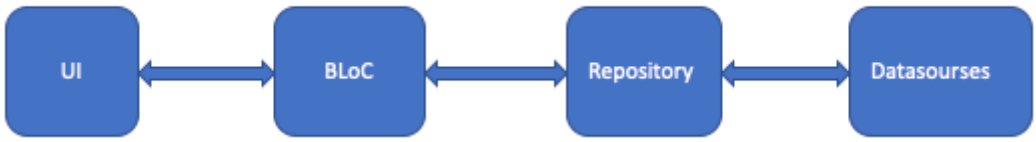
2.2.6.3. Kiến trúc Bloc trong Fluter

Về kiến trúc Bloc trong Flutter, có hai dạng mà bạn thường gặp đó là :

+ Xây dựng Bloc với RxDart

+ Xây dựng BLoC với Event – State

Nhưng dù dùng kiểu nào đi nữa, thì cấu trúc cũng theo một mô hình như bên dưới:



Hình 33: Kiến trúc Bloc trong Fluter

Trong kiến trúc BloC, chia làm 4 layers chính là : UI, BloC, Respository, DataSources:

- UI (User Interface) : Là những phần của Ứng dụng để hiển thị với người dùng.
- BloC: Luôn lắng nghe các sự kiện pass qua nó, ví dụ luôn lắng nghe data pass qua stream.

- Repository : Dùng để fetching data từ Data sources. Đầu ra của lớp này sẽ là đầu vào của khối Bloc, khi đó data sẽ được đặt trong các Stream
- Data Sources: Là khối cung cấp data cho ứng dụng như network, sqflite, shared_preferences

Khi xây dựng kiến trúc Bloc theo kiểu event – state; cần nhớ :

- Một Bloc được tạo ra từ việc kế thừa class Bloc của bloc package
- Bloc phải được khai báo state ban đầu : initial State
- Khi tạo bloc với sự kế thừa Bloc của Bloc package, bạn cần implement function mapEventToState, với tham số là event , đầu ra là stream của state.
- Sử dụng thuộc tính currentState để truy cập current state.
- Mỗi bloc cần phải có thuộc tính dispatch, hàm dispatch nhận event , trigger mapEventToState. Dispatch có vai trò như lớp presentation
- Hàm onTransition được gọi trước khi update state bloc
- Để bắt exception của bloc , chúng ta có thể override hàm onError

Các bước tạo một khối bloc và sử dụng bloc:

Tạo Events → Tạo State → Tạo Bloc → Tạo Bloc Provider → Sử dụng Bloc với state, event

2.2.7. BLoC Architecture

2.2.7.1. Khái niệm

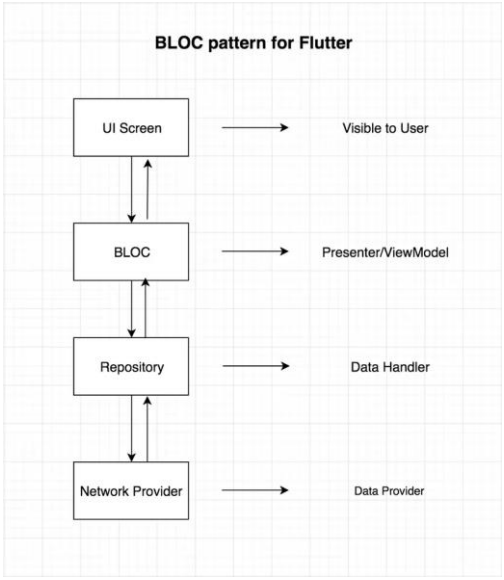
BLoC Architecture là một thành phần tách logic nghiệp vụ của ứng dụng ra khỏi giao diện người dùng thông qua việc sử dụng Streams, do đó hỗ trợ mô hình MVVM trong ứng dụng. Do đây không phải là một package mà chỉ là logic, nó cung cấp cho developer sự tự do để xây dựng ứng dụng theo cách họ mong muốn.

BLoC Architecture được xây dựng dựa trên Streams và vận hành bởi Inherited Widgets.

2.2.7.2. Mục tiêu

Để xây dựng 1 project có 1 screen dạng girl list của các item. Các item đấy sẽ được lấy từ server và lấy từ trạng Movie.

2.2.7.3. Sơ đồ Architect



Hình 34: Sơ đồ Architect

Diagram show cách lấy data tới Data layer và ngược lại. Bloc sẽ không bao giờ tham chiếu tới Widget in UI Screen. Màn Screen chỉ quan sát những thay đổi từ Bloc class.

2.2.8. Stream

Streams hoặc reactive là 1 cách tiếp cận. Dữ liệu sẽ được chuyển từ BLOC đến UI hoặc từ UI tơi BlOC dưới dạng stream.

Streams là một chuỗi các sự kiện không đồng bộ. Để hiểu rõ hơn, hãy thử tưởng tượng một đường ống chứa chất lỏng, khi thêm một màu sắc vào đầu này, nó sẽ cập nhật màu sắc của toàn bộ chất lỏng trong ống.

BLoC hoạt động tương tự trong Flutter. Để cập nhật Widget trong run time, chúng ta tạo ra Stream của các thuộc tính trong Widget, chúng sẽ thay đổi trong run time thông qua StreamController. Các thuộc tính này có thể là bất cứ thứ gì từ color, border, height, width... Sau khi một Stream được khởi tạo, nó có thể dễ dàng được thay đổi và lắng nghe thông qua các thuộc tính được phơi bày ra bởi Stream là sink và stream.

CHƯƠNG 3: CÁC BƯỚC CÀI ĐẶT

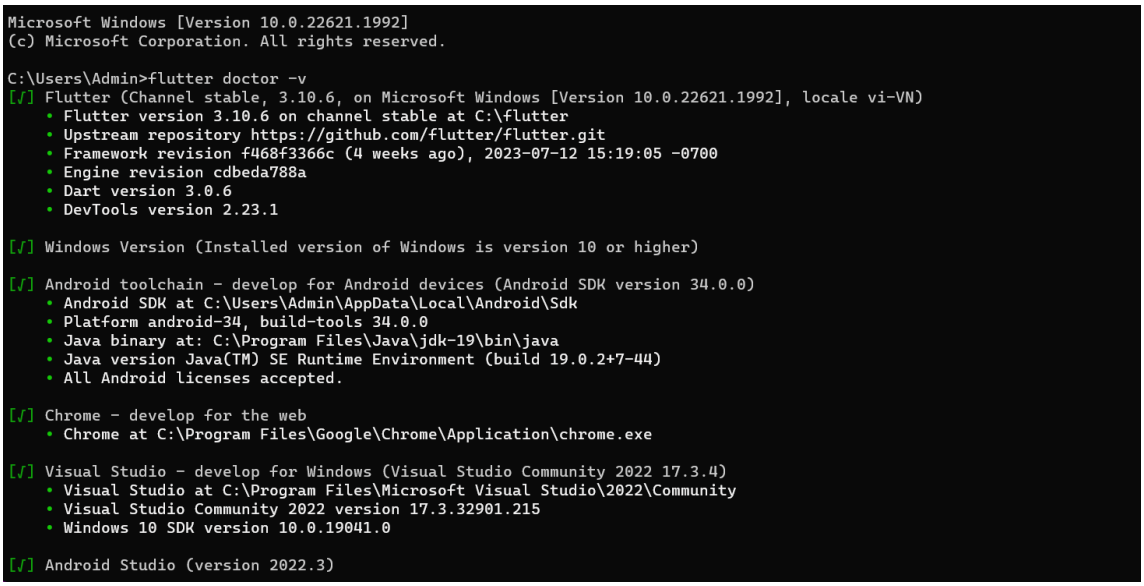
3.1. Cài đặt Android Studio:

Để cài đặt Android Studio trên Windows, hãy làm theo các bước sau: Nếu bạn đã tải một tệp .exe xuống (nên dùng), hãy nhấp đúp để mở tệp đó. Nếu bạn đã tải một tệp .zip xuống, thì hãy:

- Giải nén .zip.
- Sao chép thư mục android-studio vào thư mục Program Files (Tệp chương trình).
- Mở thư mục android-studio > bin.
- Chạy studio64.exe (đối với máy 64 bit) hoặc studio.exe (đối với máy 32 bit).
- Làm theo Trình hướng dẫn thiết lập trong Android Studio và cài đặt mọi gói SDK được đề xuất.

Tiến hành cài đặt theo link:
https://developer.android.com/studio/install?hl=vi&fbclid=IwAR1CqBtM4Vy4Hs_JR_2XJtvcbOgCKpm9yWQnPDAnDASdlq8wzlpf8w1ASf4

Tải thêm cái phần hỗ trợ với các thông số sau:

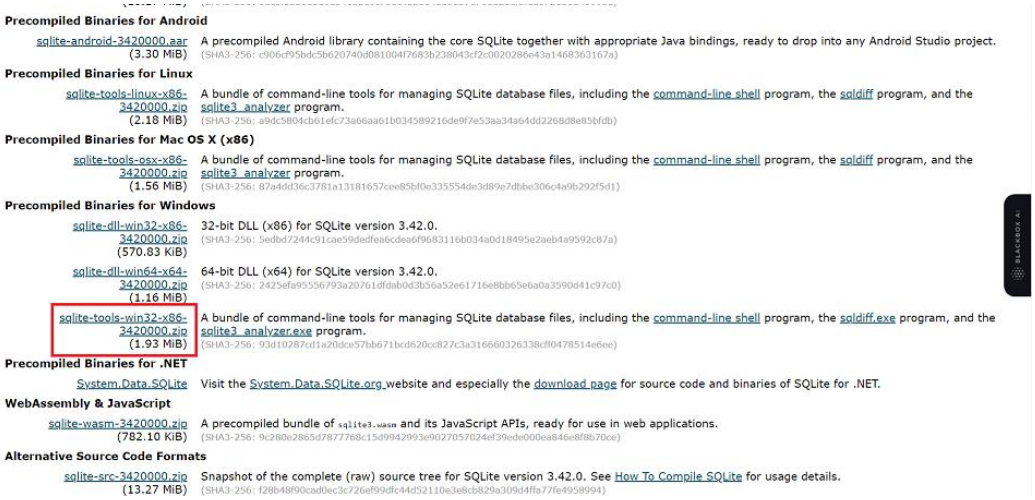


Hình 35: Phần mềm hỗ trợ

3.2. Cài đặt SQLite

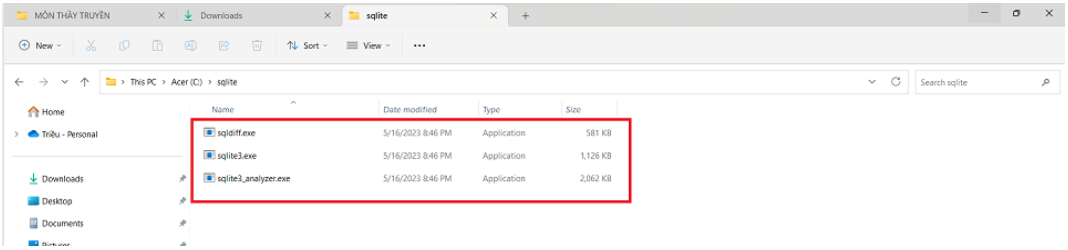
3.2.1. Tải SQLite

Bước 1: Vào trang https://www.sqlite.org/download.html**.**



Hình 36: Chọn version SQLite

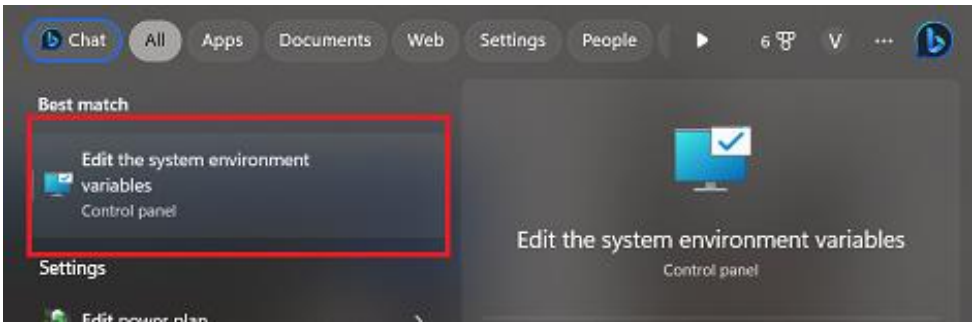
Bước 2: Sau khi download xong, ta tiến hành giải nén tệp đó, vào ổ C tạo thư mục có tên "sqlite", copy file mới giải nén bỏ vào tệp mới tạo.



Hình 37: Giải nén

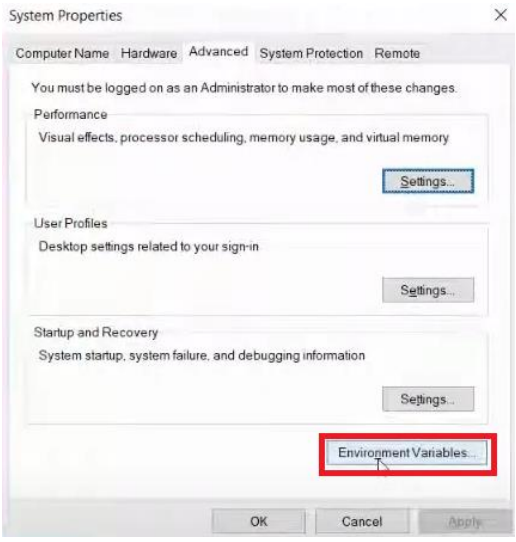
3.2.2. Tạo biến môi trường

Bước 1: Vào thanh Search tìm từ khóa **Edit the system environment variables.**



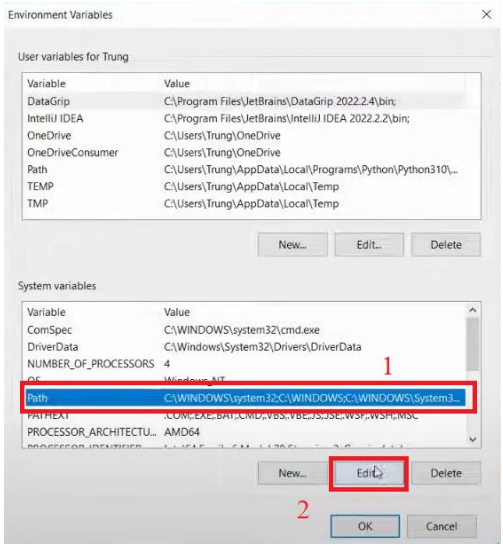
Hình 38: Vào thanh Search tìm từ khóa

Bước 2: Chọn Environment Variables...



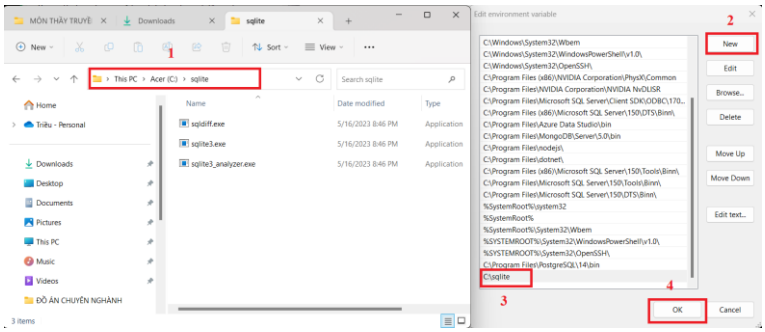
Hình 39: Chọn Environment Variables...

Bước 3: Chọn System Variables → chọn Path → Edit.



Hình 40: Chọn System Variables → chọn Path → Edit.

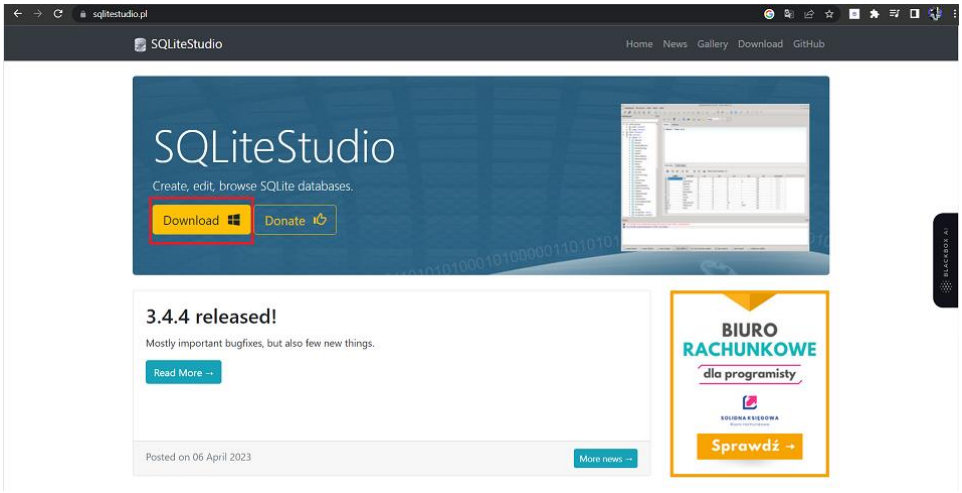
Bước 4: Copy đường dẫn của ổ C có chứa tệp của sqlite → chọn New → paste đường dẫn vào ô mới tạo → OK.



Hình 41: Cài biến môi trường

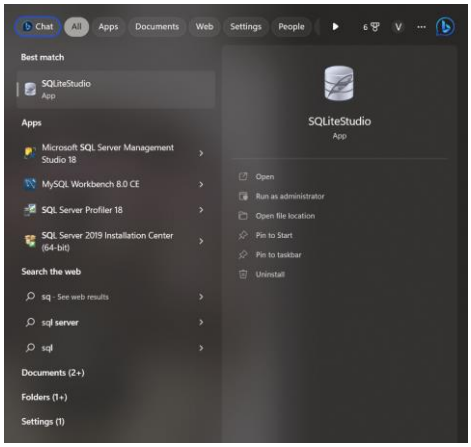
3.2.3. Cài đặt SQLiteStudio

Bước 1: Vào trang https://sqlitestudio.pl/ để download.



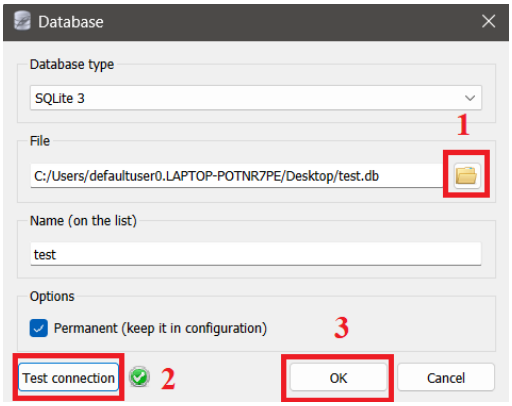
Hình 42: Cài đặt SQLiteStudio

Bước 2: Làm quen với SQLiteStudio



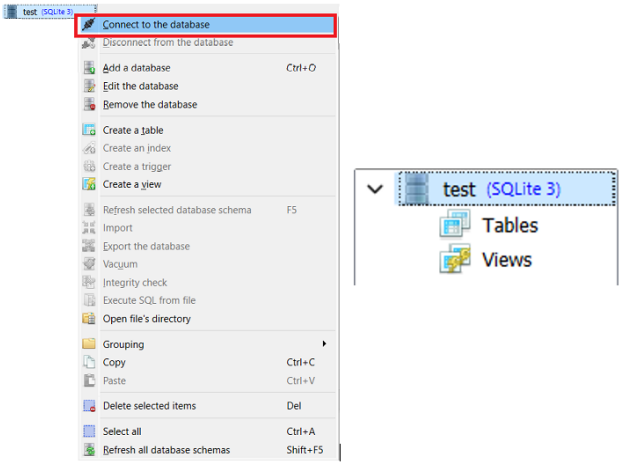
Hình 43: Giao diện SQLiteStudio

Bước 3: Tạo database (Ctrl+O)



Hình 44: Tạo database

Bước 4: Chuột phải chọn connect to database



Hình 45: Chuột phải chọn connect to database

CHƯƠNG 4: CÁC BƯỚC TIẾN HÀNH

4.1. Phân tích và thiết kế cơ sở dữ liệu

4.1.1. Giới thiệu

Trình bày nội dung và phân tích cách người dùng trò chuyện với nhau, các chức năng được sử dụng trong ứng dụng để phân tích ra các mô hình dữ liệu.

4.1.2. Khảo sát yêu cầu người dùng

Khả năng trò chuyện với bạn bè.

Hệ thống sẽ lưu trữ thông tin của người dùng.

4.1.3. Chức năng của hệ thống

- **Admin**

Chức năng thêm người dùng.

Chức năng xóa người dùng.

Chức năng sửa thông tin người dùng.

- **User**

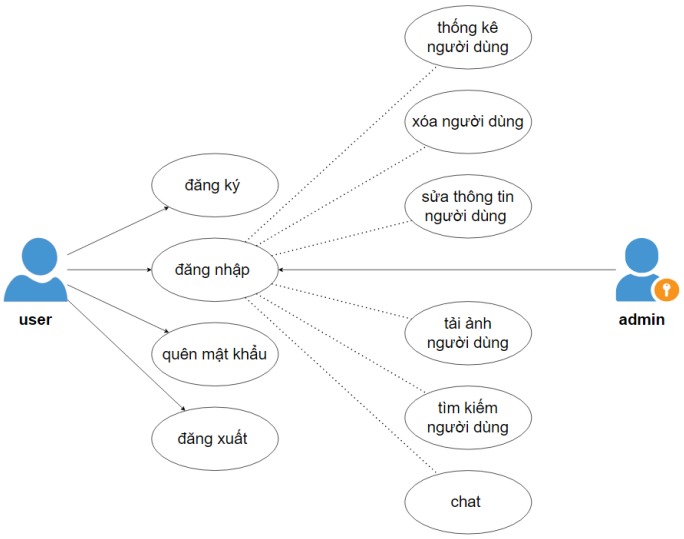
Chức năng đăng ký.

Chức năng đăng nhập.

Chức năng chat với bạn bè.

Chức năng thoát tài khoản.

4.2. Sơ đồ Usecase



Hình 46: Sơ đồ Usecase

4.3. Đặc tả Usecase

4.3.1. Đặc tả đăng nhập

Tên Use-Case	Đăng nhập
Actor chính	Người dùng
Actor phụ	Người quản trị

Mô tả chung: Cho phép người dùng truy cập vào hệ thống

- Bước 1: Nếu người dùng chưa có tài khoản đăng nhập, người dùng phải đăng ký trước.
- Bước 2: Người dùng sẽ đăng nhập trước khi truy cập vào hệ thống.
- Bước 3: Người dùng điền thông tin cá nhân như tên đăng nhập và mật khẩu.
- Bước 4: Người dùng bấm nút đăng nhập.
- Bước 5: Hệ thống sẽ kiểm tra thông tin khớp với đăng ký lúc đầu và hiển thị thông báo đăng nhập thành công (Nếu hệ thống kiểm tra thông tin không khớp với đăng ký lúc đầu, hệ thống sẽ hiện thị đăng nhập không thành công).
- Bước 6: Người dùng đã vào được hệ thống.

4.3.2. Đặc tả Usecase đăng xuất

Tên Use-case	Đăng nhập
Actor chính	Người dùng
Actor phụ	Người quản trị

Mô tả chung: Cho phép người dùng thoát khỏi hệ thống

- Bước 1: Người dùng bấm vào tài khoản tài khoản của mình.
- Bước 2: Hệ thống sẽ hiển thị giao diện tài khoản cá nhân.
- Bước 3: Người dùng nhấn vào nút đăng xuất tài khoản.
- Bước 4: Hệ thống sẽ hiển thị thông báo xác nhận đăng xuất hoặc không.
- Bước 5: Người dùng bấm xác nhận đăng xuất hoặc không.
- Bước 6: Người dùng sẽ ra ngoài trang giao diện đăng nhập. (Nếu người dùng không đăng xuất sẽ vẫn còn ở trong trang giao diện thông tin cá nhân).

4.3.3. Đặc tả Usecase đăng ký tài khoản

Tên Use-case	Đăng ký tài khoản
Actor chính	Người dùng

Actor phụ	Người quản trị

Mô tả chung: Cho phép người dùng đăng ký tài khoản cá nhân.

- Bước 1: Người dùng bấm nút đăng ký.
- Bước 2: Người dùng sẽ được đưa đến trang giao diện đăng ký.
- Bước 3: Người dùng phải điền các thông tin cá nhân (Tên đăng ký, mật khẩu, email, số điện thoại,..)
- Bước 4: Hệ thống sẽ kiểm tra hợp lệ hay không (nếu hợp lệ → đăng ký thành công, không hợp lệ → đăng ký không thành công).
- Bước 5: Người dùng sẽ vào được trang đăng nhập.

4.3.4. Đặc tả Usecase chat

Tên Use-case	Nhắn tin cá nhân
Actor chính	Người dùng
Actor phụ 1	Người quản trị
Actor phụ 2	Người dùng 2

Mô tả chung: Cho phép người dùng có thể nhắn tin với bạn bè đã kết bạn.

- Bước 1: Người dùng bấm vào khung tìm kiếm.
- Bước 2: Người dùng tìm kiếm tên bạn bè cần chat.
- Bước 3: Hệ thống sẽ gợi ý những người bạn cần tìm.
- Bước 4: Người dùng chọn một người bạn.
- Bước 5: Người dùng có thể thêm một hoặc nhiều người bạn khác vào nhóm.

4.3.5. Đặc tả Usecase tải ảnh người dùng

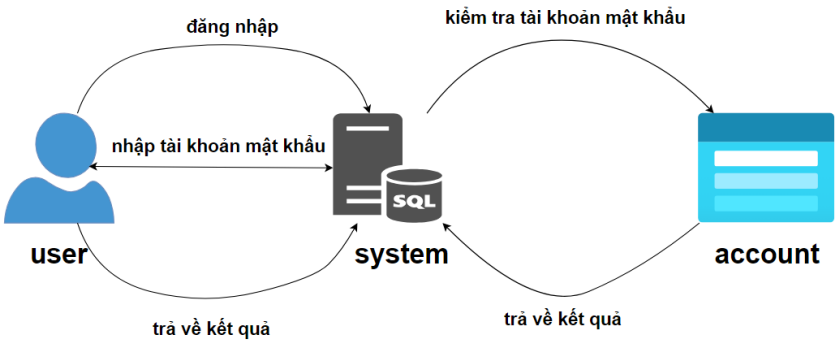
Tên Use-case	Nhắn tin cá nhân
Actor chính	Người dùng
Actor phụ	Người quản trị

Mô tả chung: Cho phép người dùng đăng tải hình của mình.

- Bước 1: Người dùng bấm chọn ảnh avatar.
- Bước 2: Hệ thống sẽ hiển thị giao diện avatar.
- Bước 3: Người dùng chọn ảnh sẽ tải lên.
- Bước 4: Hệ thống sẽ hỏi xác nhận người dùng.
- Bước 5: Hệ thống ứng dụng sẽ cập nhật hình avatar của người dùng.

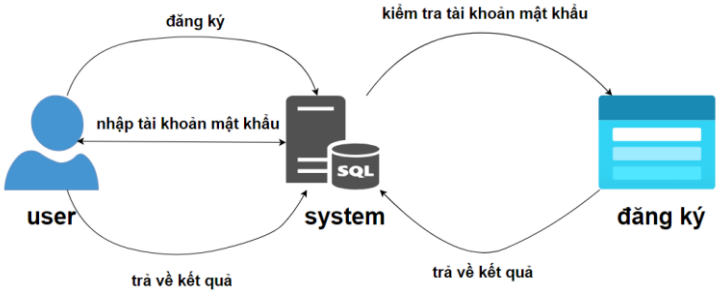
4.4. Sơ đồ tuần tự

4.4.1. Đăng nhập



Hình 47: Đăng nhập

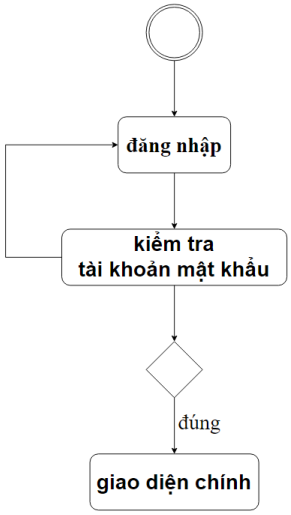
4.4.2. Đăng ký



Hình 48: Đăng ký

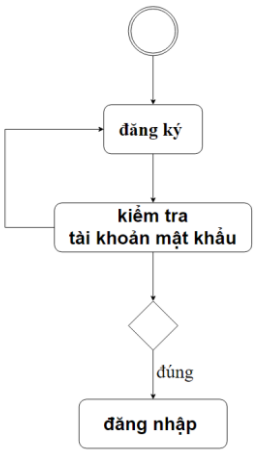
4.5. Sơ đồ hoạt động

4.5.1. Đăng nhập



Hình 49: Mô tả sơ đồ hoạt động đăng nhập

4.5.2. Đăng ký



Hình 50: Mô tả sơ đồ hoạt động đăng ký

4.6. Thiết kế giao diện

4.6.1. Giao diện trang đăng nhập

Hình 51: Giao diện trang đăng nhập

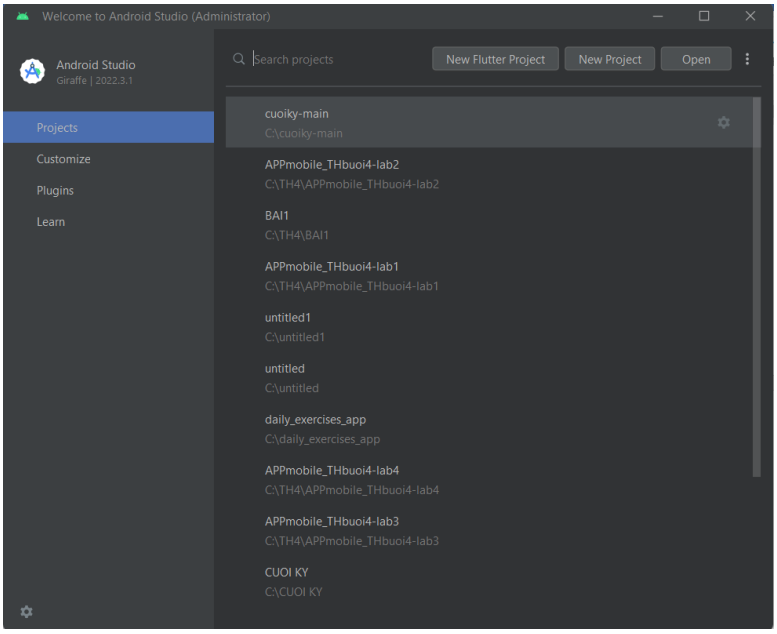
4.6.2. Giao diện trang đăng ký

Hình 52: Giao diện trang đăng ký

4.7. Triển khai

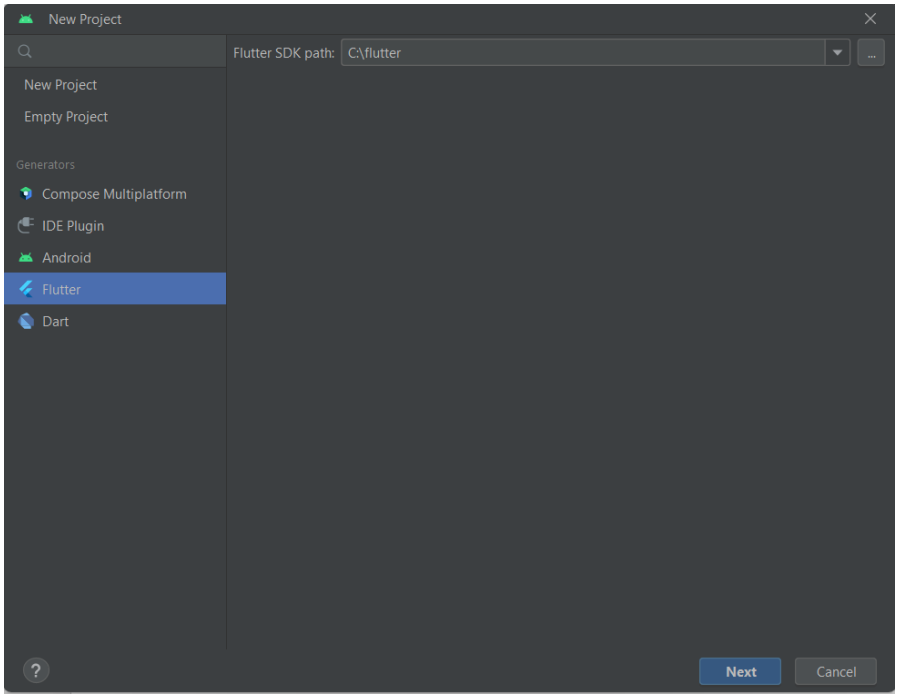
4.7.1. Tạo 1 project mới

Bước 1: Khởi động phần mềm Android Studio → chọn New Flutter Project



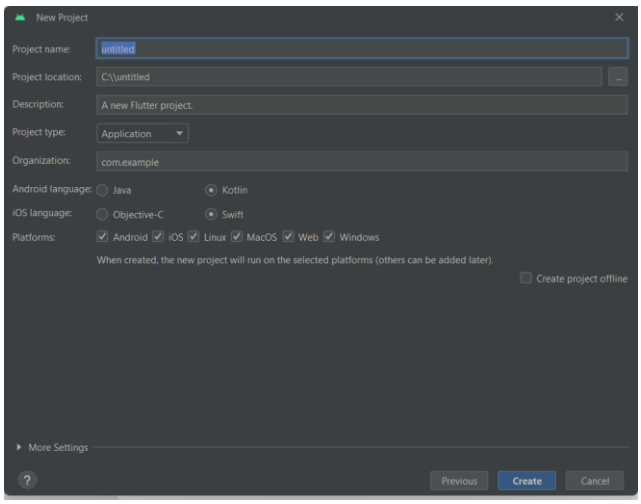
Hình 53: New Flutter Project

Bước 2: Tại hộp thoại New Project → chọn Flutter → Next



Hình 54:

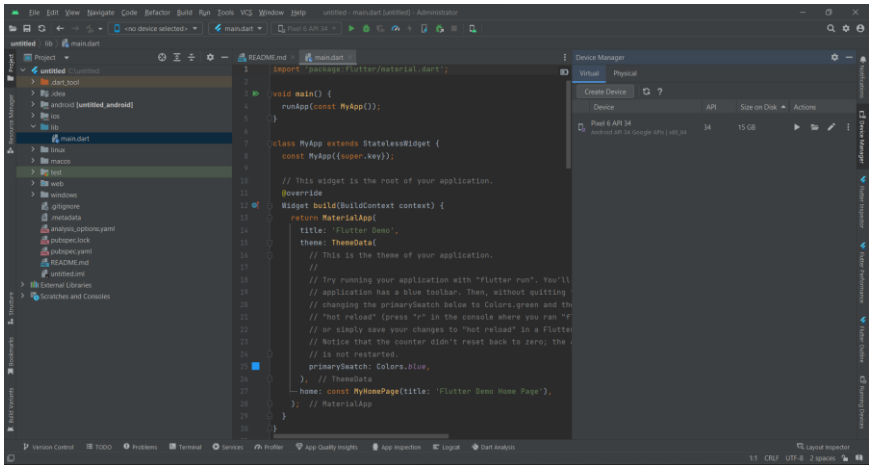
Bước 3: Đặt tên Project → Create



Hình 55: Đặt tên Project mới

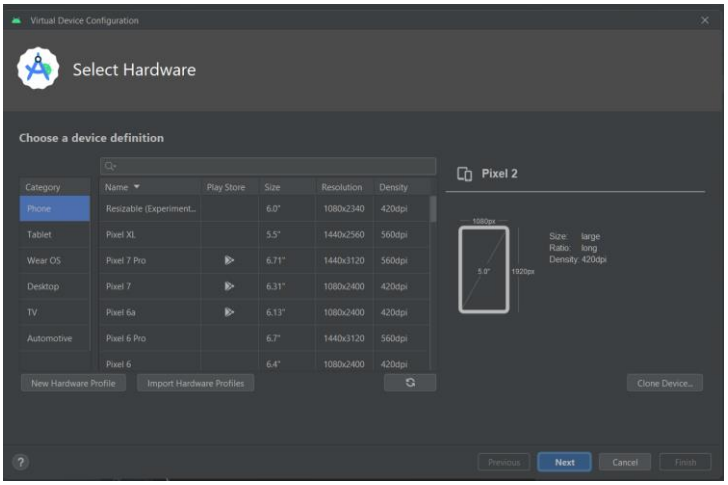
4.7.2. Thiết lập điện thoại cho Android Studio

Bước 1: Ở Device Manger → Chọn Create Device



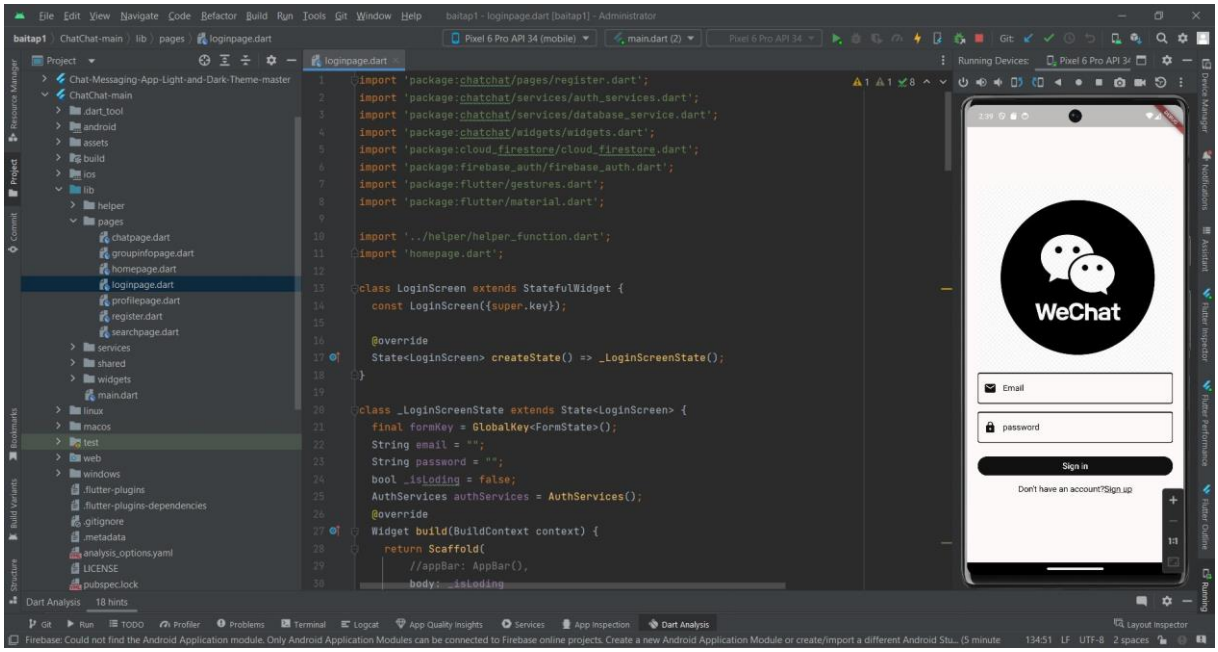
Hình 56: Thiết lập điện thoại cho Android Studio

Bước 2: Chọn thiết bị phù hợp → Create



Hình 57: Chọn thiết bị phù hợp → Create

4.7.3. Thực hiện triển khai Project



Hình 58: Thực hiện triển khai Project

4.8. Kết quả

4.8.1. Trang đăng nhập

Hình 59: Trang đăng nhập

4.8.2. Trang đăng ký

Hình 60: Trang đăng ký

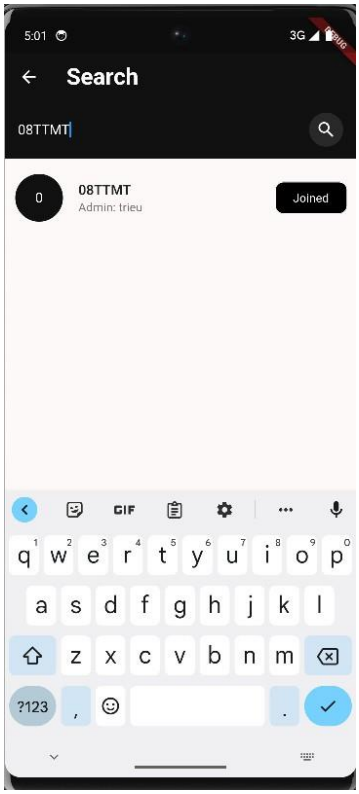
4.8.3. Trang chủ

Hình 61: Trang chủ

4.8.4. Trang Tìm kiếm

Hình 62: Trang Tìm kiếm

4.8.5. Trang Gia nhập nhóm

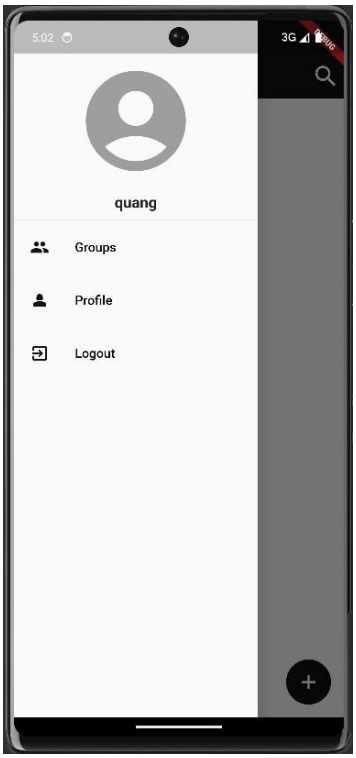


Hình 63: Trang Gia nhập nhóm

4.8.6. Trang Hội thoại

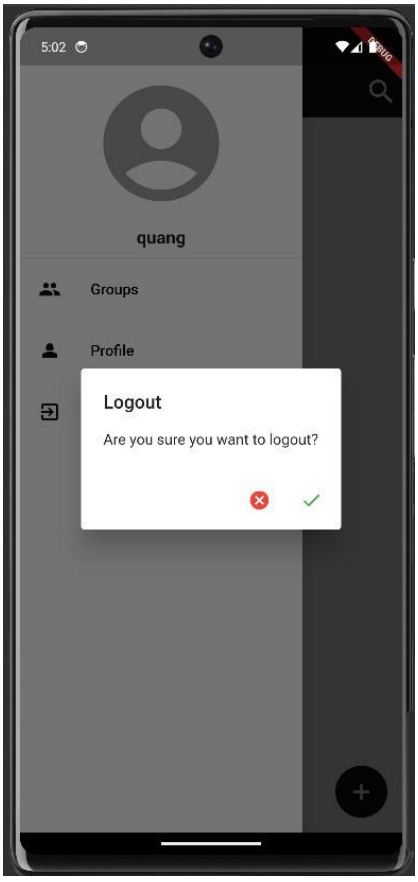
Hình 64: Trang Hội thoại

4.8.7. Trang Thông tin



Hình 65: Trang Thông tin

4.8.8. Trang Đăng xuất



Hình 66: Trang Đăng xuất

CHƯƠNG 5: KẾT LUẬN

Qua báo cáo môn học này, nhóm chúng em đã triển khai được:

- Giao diện App Chat đơn giản như:

- Đăng ký
- Đăng nhập
  - Xác định được khả năng tương thích với thông tin đăng ký
- Trang chủ
- Tạo groups
  - Có thể xem được thành viên trong group
- Tìm kiếm
- Trang thông tin
  - Thông tin người dùng
  - Email người dùng
- Đăng xuất

- Trờ về lại giao diện trang Đăng nhập (có thể có sự lựa chọn)

- Có thể đồng bộ hóa nhiều người dùng nhắn tin chung trên cùng 1 group hoặc nhiều group khác nhau.

Tuy nhiên, vì thời gian ngắn nên nhóm chúng em còn những lỗi chưa khắc phục được như:

- Chưa đồng bộ hóa về ngôn ngữ Tiếng Việt

- Chưa xóa/ thu hồi tin nhắn về phía người dùng

- Chưa có chức năng rời khỏi group

Trong tương lai, nhóm chúng em sẽ hoàn thiên hơn một số tính năng trên và triển khai App phù hợp và thân thiện với người dùng hơn.

TÀI LIỆU THAM KHẢO

[1]: https://bmdsolutions.vn/lap-trinh-da-nen-tang

[2]: https://vietnix.vn/mobile-app-la-gi/

[3]: https://bizfly.vn/techblog/lap-trinh-da-nen-tang.html

[4]: https://bmdsolutions.vn/lap-trinh-da-nen-tang/

[5]: https://websitehcm.com/gioi-thieu-ve-ngon-ngu-lap-trinh-dart/

[6]: https://viblo.asia/p/tong-quan-ve-flutter-Eb85oyAkZ2G

[7]: https://vi.wikipedia.org/wiki/Flutter

[8]: https://viblo.asia/p/su-dung-bloc-trong-flutter-de-quan-ly-state-eW65GWJa5DO

[9]: https://baoflutter.com/nghe-thuat-flutter-kien-truc-bloc-va-su-dung-blocprovider/

[10]: https://baoflutter.com/nghe-thuat-flutter-kien-truc-bloc-va-su-dung-blocprovider/

[11]: https://viblo.asia/p/flutter-quan-ly-trang-thai-cua-widget-RQqKLEGMZ7z

[12]: https://sqlitestudio.pl/?act=download%E3%80%82SQLiteStudio

BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ KẾT QUẢ

Họ tên	Nội dung công việc	%
Nguyễn Phạm Thảo Ngân (0850070031)	- Phân chia nhiệm vụ - Tìm hiểu tổng quan môn học - Tìm hiểu BloCArchitecture - Thiết kế trang đăng ký, đăng nhập, trang chat - Tổng hợp file word - Viết báo cáo - Làm powerpoint	25%
Lại Thị Phương Nhung (0850070036)	- Tìm hiểu BloCArchitecture - Thiết kế trang đăng ký, đăng nhập	15%
Trần Thanh Phong (0850070039)	- Tìm hiểu SQLite - Tìm hiểu SQLite in flutter	10%
Nguyễn Ngọc Quang (0850070045)	- Tìm hiểu tổng quan môn học - Tìm hiểu BloCArchitecture - Thiết kế trang đăng ký, đăng nhập, trang chat	25%
Võ Hoàng Triều (0850070060)	- Tìm hiểu SQLite - Tìm hiểu SQLite in flutter - Xây dựng CSDL - Thiết kế sơ đồ	25%