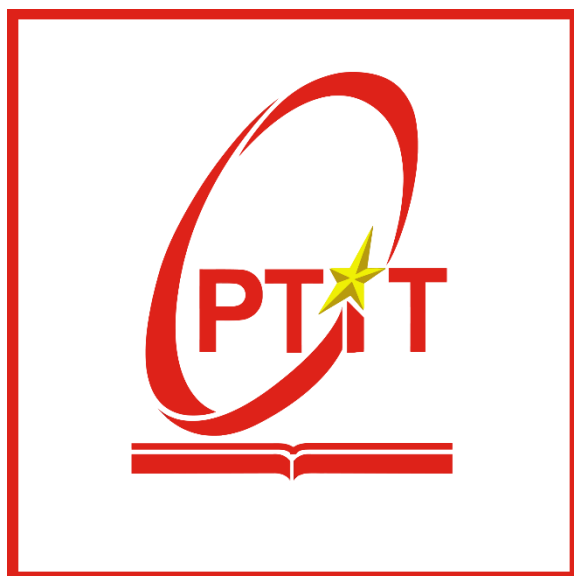


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

-----o0o-----



BÁO CÁO MÔN THỰC TẬP CƠ SỞ
ĐỀ TÀI:
LẬP TRÌNH ANDROID ỨNG DỤNG QUẢN LÝ CHI TIÊU CÁ NHÂN

Giáo viên hướng dẫn : PGS. TS. Nguyễn Mạnh Hùng

Họ và tên sinh viên : Nguyễn Phúc Sơn

Mã sinh viên : B20DCCN581

Lớp : D20CQCN05-B

Hà Nội – 2023

QUÁ TRÌNH LÀM BÁO CÁO

Tên tuần	Nội dung đã làm được
Tuần 1 – 25/2/2023	Hoàn thiện bìa, outline, chương 1, tiếp tục hoàn thiện chương 2
Tuần 2 – 4/3/2023	<ul style="list-style-type: none"> - Cập nhật thêm trong phần 2.2 chương 2 - Thêm phần kiến trúc ứng dụng và mô hình MVVM (2.3 và 2.4) - Sửa lỗi ở tuần 1: thêm phần tài liệu tham khảo
Tuần 3 – 11/3/2023	<ul style="list-style-type: none"> - Sửa lỗi chú thích tài liệu tham khảo - Cập nhật outline - Thêm các cơ sở lý thuyết của ứng dụng Android 2.3, 2.6 - Tìm hiểu thêm về thư viện Room 2.4.2
Tuần 4 – 18/3/2023	<ul style="list-style-type: none"> - Thêm phần 3: cài đặt ứng dụng, bao gồm <ul style="list-style-type: none"> + tạo dự án mới trên Android Studio + thiết kế cơ sở dữ liệu để lưu trữ trên SQLite
Tuần 5 – 25/3/2023	<ul style="list-style-type: none"> - Cập nhật phần 3.2 - Thêm phần cài đặt cơ sở dữ liệu trên dự án Android Studio với thư viện Room
Tuần 6 – 1/4/2023	<ul style="list-style-type: none"> - Thêm 3.3, hoàn thành cơ bản các giao diện màn hình của ứng dụng - Hoàn thành chức năng hiển thị cơ bản của HomeScreen - Hiện tại đang tìm hiểu và sửa các lỗi liên qua đến việc cài đặt chức năng lọc và tìm kiếm trên HomeScreen trong ViewModel và Room Database
Tuần 7 – 8/4/2023	<ul style="list-style-type: none"> - Thêm chi tiết các thành phần và bố cục UI trong Jetpack Compose (2.4.1) - Thay các dòng code bằng ảnh giúp code trực quan hơn - Hoàn thiện chức năng và cập nhật logic màn hình HomeScreen (3.3.1) - Đang sửa lại các ViewModel
Tuần 8 – 15/4/2023	- Code và logic điều hướng bằng Jetpack Compose (3.3.1 b)
Tuần 9 – 22/4/2023	- Thêm các thành phần ViewModel và chức năng của chúng trong từng màn hình (đang hoàn thiện - 3.3.2)
Tuần nộp	<ul style="list-style-type: none"> - Bỏ các mục code copy, các lý thuyết dài dòng dư thừa - Hoàn thiện một bài báo cáo hoàn chỉnh

MỤC LỤC

MỞ ĐẦU	3
1. GIỚI THIỆU ĐỀ TÀI	4
1.1 Các vấn đề trong bài toán quản lý chi tiêu cá nhân thông qua ứng dụng	4
1.2 Các giải pháp đề xuất cho ứng dụng quản lý chi tiêu	4
1.3 Đề xuất giải pháp của luận văn để giải quyết bài toán	5
2. CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH THIẾT KẾ.....	5
2.1 Cơ sở lý thuyết	5
2.1.1 Hệ điều hành Android.....	5
2.1.2 Bộ công cụ hỗ trợ phát triển phần mềm Android	5
2.1.3 Nền tảng của một ứng dụng Android.....	9
2.1.4 Android Jetpack	12
2.1.5 Kiến trúc ứng dụng Android	17
2.1.6 Mô hình Model, View, ViewModel (MVVM).....	21
2.2 Phân tích thiết kế.....	23
2.2.1 Phân tích thiết kế dữ liệu	23
2.2.2 Chức năng các màn hình.....	27
3. ỨNG DỤNG THỰC NGHIỆM	28
3.1. Home Screen (Màn hình chính).....	28
3.2 Màn hình thêm lịch sử giao dịch.....	28
3.3 Màn hình xem chi tiết lịch sử giao dịch.....	30
3.4 Màn hình chỉnh sửa lịch sử giao dịch	30
3.5 Màn hình thống kê lịch sử giao dịch.....	31
KẾT LUẬN	32
TÀI LIỆU THAM KHẢO.....	33

MỞ ĐẦU

Trong thời đại công nghệ số hiện nay, điện thoại di động đã trở thành một thiết bị không thể thiếu trong cuộc sống của chúng ta. Việc sử dụng điện thoại di động không chỉ đơn thuần là để liên lạc với bạn bè, đồng nghiệp hay để giải trí, mà nó còn có thể giúp chúng ta quản lý tài chính cá nhân một cách dễ dàng và hiệu quả hơn.

Với mục đích giúp người dùng tiết kiệm thời gian và công sức trong việc quản lý chi tiêu cá nhân, ứng dụng quản lý chi tiêu cá nhân trên điện thoại di động đã được phát triển và trở thành một trong những ứng dụng phổ biến nhất trên thị trường hiện nay. Đặc biệt, với hệ điều hành Android được sử dụng rộng rãi trên điện thoại di động, việc phát triển ứng dụng quản lý chi tiêu cá nhân trên nền tảng này đã thu hút sự quan tâm của nhiều nhà phát triển.

Trong báo cáo này, tôi sẽ giới thiệu cơ sở lý thuyết và demo ứng dụng quản lý chi tiêu cá nhân trên nền tảng Android. Nội dung báo cáo sẽ bao gồm các chức năng cơ bản của ứng dụng, các công nghệ sử dụng để xây dựng ứng dụng, và các thử nghiệm và đánh giá hiệu suất của ứng dụng. Tôi hi vọng báo cáo này sẽ cung cấp cho người đọc những kiến thức cơ bản về lập trình Android và phát triển ứng dụng trên thiết bị di động, đặc biệt là trong lĩnh vực quản lý tài chính cá nhân.

1. GIỚI THIỆU ĐỀ TÀI

1.1 Các vấn đề trong bài toán quản lý chi tiêu cá nhân thông qua ứng dụng

Bài toán mà đề tài lập trình Android ứng dụng quản lý chi tiêu cá nhân giải quyết là giúp người dùng quản lý các khoản thu chi cá nhân một cách chi tiết, khoa học, hiệu quả và thuận tiện hơn. Thông qua ứng dụng, người dùng có thể dễ dàng đăng nhập, theo dõi, phân loại và thống kê chi tiêu của mình để có được cái nhìn tổng quan về tình hình tài chính của mình. Bên cạnh đó, ứng dụng còn có thể cung cấp các chức năng hữu ích khác như đặt mục tiêu tiết kiệm, nhắc nhở lịch sử giao dịch, nhắc nhở thực hiện ghi lại giao dịch, tạo ghi chú cho giao dịch, giúp người dùng quản lý và sắp xếp ngân sách cá nhân một cách hiệu quả. Với bài toán này, đề tài hy vọng sẽ giúp cho người dùng có thể quản lý tài chính của mình một cách thông minh và tiện lợi hơn.

1.2 Các giải pháp đề xuất cho ứng dụng quản lý chi tiêu

Hiện nay, trên thị trường đã xuất hiện nhiều ứng dụng quản lý chi tiêu cá nhân trên điện thoại, với nhiều tính năng hữu ích để hỗ trợ người dùng. Sau đây là một số giải pháp và nghiên cứu đáng chú ý:

- Quản lý thu chi tự động: nhiều ứng dụng quản lý chi tiêu hiện nay đã có tính năng tự động phân loại và theo dõi chi tiêu của người dùng thông qua các giao dịch tài chính được thực hiện trên thẻ tín dụng hoặc tài khoản ngân hàng. Điều này giúp người dùng tiết kiệm thời gian và nỗ lực trong việc nhập thông tin.
- Tính năng ghi nhớ và phân loại giao dịch: một số ứng dụng cho phép người dùng tùy chỉnh và phân loại các giao dịch chi tiêu của mình theo từng danh mục cụ thể, ví dụ như thức ăn, đi lại, giải trí, tiền thuê nhà, tiền điện nước... điều này giúp người dùng có cái nhìn chi tiết hơn về ngân sách của mình và tìm ra những khoản chi tiêu không cần thiết.
- Tính năng tạo ngân sách: một số ứng dụng cung cấp tính năng tạo ngân sách cá nhân cho người dùng, giúp họ quản lý và theo dõi số tiền họ có thể chi tiêu cho từng danh mục và ngăn chặn việc chi tiêu quá mức.
- Tính năng phân tích số liệu: một số ứng dụng có khả năng phân tích và tạo ra báo cáo về chi tiêu của người dùng, giúp họ có cái nhìn tổng quan hơn về tình hình tài chính của mình, cũng như tìm ra những mẫu chi tiêu và xu hướng chi tiêu của mình.
- Tính năng nhắc nhở: nhiều ứng dụng còn cung cấp tính năng nhắc nhở cho người dùng về các khoản chi tiêu cần thanh toán, lịch sử giao dịch, hay những khoản chi tiêu quan trọng, giúp họ tránh quên và giữ được sự ổn định trong tài chính cá nhân.
- Tính năng đồng bộ: một số ứng dụng cho phép đồng bộ hóa thông tin đã lưu trên thiết bị di động hiện tại của họ lên nền tảng cloud. Khi đó họ vẫn có thể tiếp tục cập nhật thông tin giao dịch của mình trên các thiết bị khác mà không bị mất các lịch sử giao dịch cũ.

Một số ứng dụng nổi bật như Mint, PocketGuard, Expensify và Spendee. Tuy nhiên, mỗi ứng dụng lại có những ưu nhược điểm riêng, phù hợp với nhu cầu và thói quen chi tiêu của từng người dùng.

1.3 Đề xuất giải pháp của luận văn để giải quyết bài toán

Tính năng mà bài báo cáo này đưa ra sẽ là:

- Ghi nhớ và phân loại giao dịch: đọc-thêm-sửa-xoá thông tin của lịch sử giao dịch, tìm kiếm và phân loại giao dịch, giúp người dùng dễ dàng theo dõi và quản lý lịch sử giao dịch của mình
- Thống kê số liệu: giúp người dùng có cái nhìn tổng quan hơn về lịch sử chi tiêu của mình

2. CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH THIẾT KẾ

2.1 Cơ sở lý thuyết

2.1.1 Hệ điều hành Android

Hệ điều hành Android là một trong những hệ điều hành phổ biến nhất hiện nay, được sử dụng trên hàng tỷ thiết bị trên toàn thế giới. Android được phát triển bởi Google và dựa trên nền tảng mã nguồn mở Linux, cho phép các nhà phát triển thiết kế và phát triển ứng dụng trên nền tảng Android. Với nhiều phiên bản phát hành, Android luôn cải tiến và cung cấp nhiều tính năng mới để đáp ứng nhu cầu người dùng, cũng như hỗ trợ cho các nhà phát triển phát triển các ứng dụng đa dạng và phong phú. Hơn nữa, Android cũng được tích hợp với nhiều dịch vụ của Google như Google Play, Google Maps, Google Drive, Google Photos... giúp cho việc sử dụng và quản lý thiết bị trở nên tiện lợi và đơn giản hơn.

2.1.2 Bộ công cụ hỗ trợ phát triển phần mềm Android

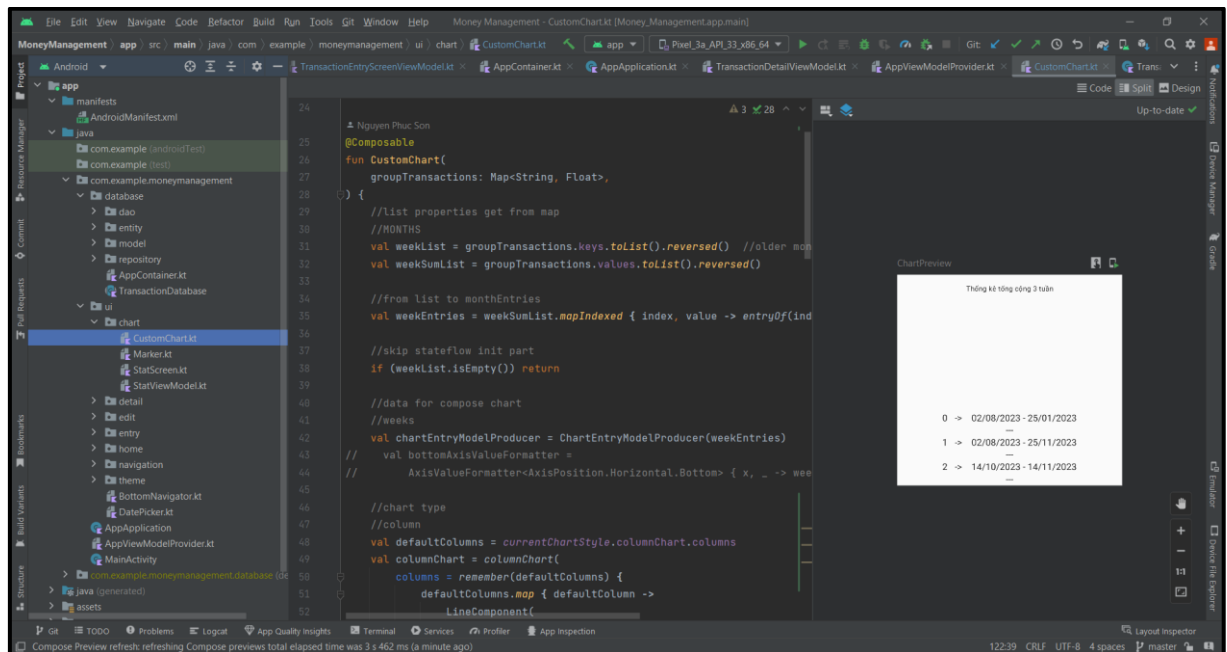
2.1.2.1. Android Software Development (Android SDK)

Android SDK là một bộ công cụ phần mềm cung cấp các công cụ cần thiết để phát triển ứng dụng Android. Android SDK bao gồm một loạt các thư viện, công cụ, trình biên dịch và các tài liệu hướng dẫn để phát triển ứng dụng cho hệ điều hành Android. [1]

Các thành phần chính của Android SDK bao gồm:

- Android Studio: Là môi trường phát triển tích hợp cho phép lập trình viên phát triển ứng dụng Android. Android Studio bao gồm trình biên dịch, trình quản lý dự án, trình gỡ lỗi và một loạt các công cụ hỗ trợ phát triển ứng dụng.
- Android Debug Bridge (ADB): Là một công cụ cho phép truy cập vào các thiết bị Android để thực hiện các tác vụ như cài đặt ứng dụng, gỡ lỗi và kiểm tra các thông số kỹ thuật của thiết bị.
- Android Emulator: Là một môi trường giả lập thiết bị Android để phát triển và kiểm thử ứng dụng.

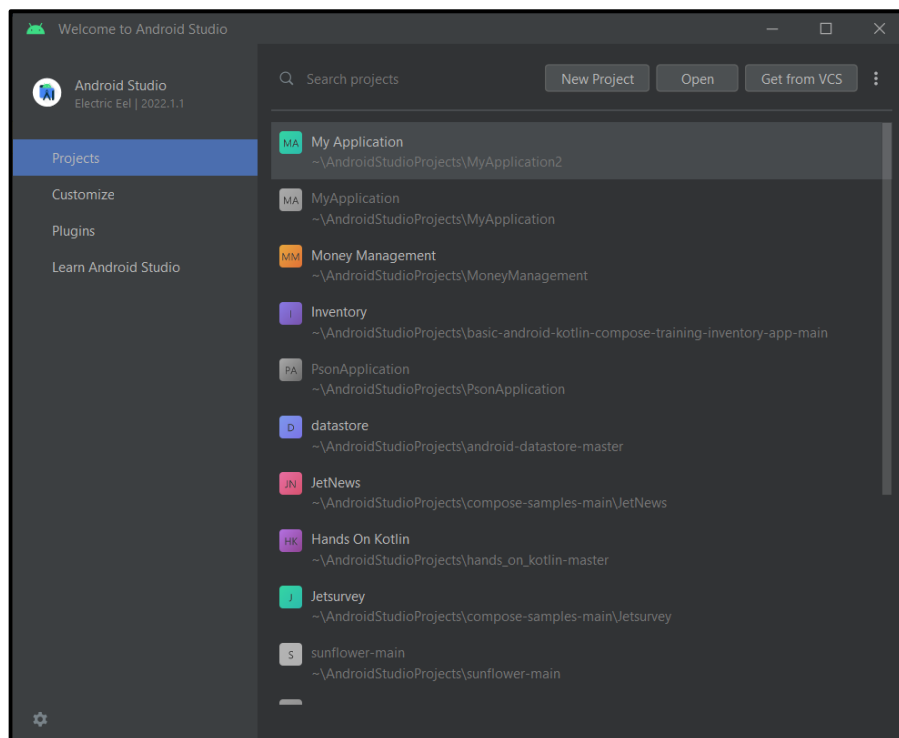
- Android SDK Platform: Là bộ thư viện phát triển Android chứa các API và thư viện hỗ trợ phát triển ứng dụng Android.
- Android Support Library: Là một bộ thư viện hỗ trợ cho các phiên bản Android cũ hơn, giúp phát triển ứng dụng tương thích với các phiên bản cũ hơn của Android.



Hình 2.1: Giao diện Android Studio

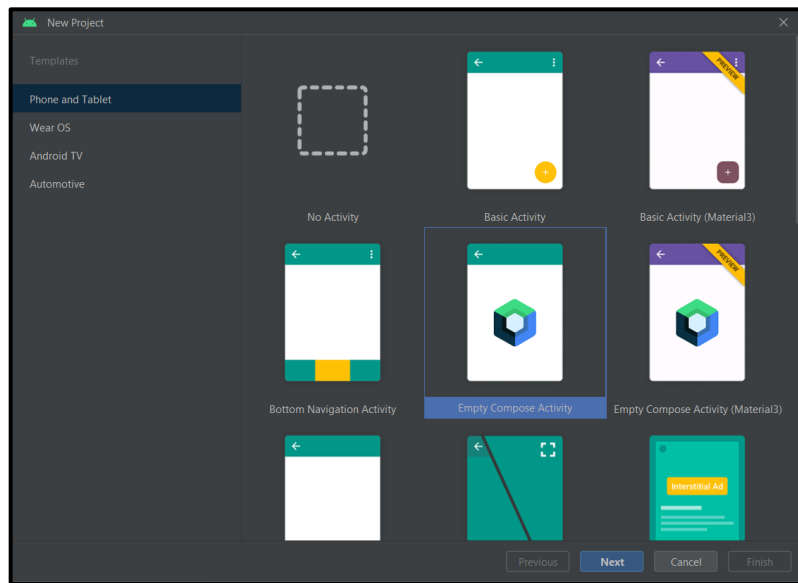
Android SDK được cung cấp miễn phí bởi Google và có sẵn để tải xuống trên trang web của Android Developer. Các lập trình viên có thể sử dụng Android SDK để phát triển ứng dụng Android trên các nền tảng khác nhau như Windows, macOS và Linux.

Bước 1: Mở Android Studio và chọn "New Project"



Hình 2.2: Cửa sổ chào mừng khi mở Android Studio

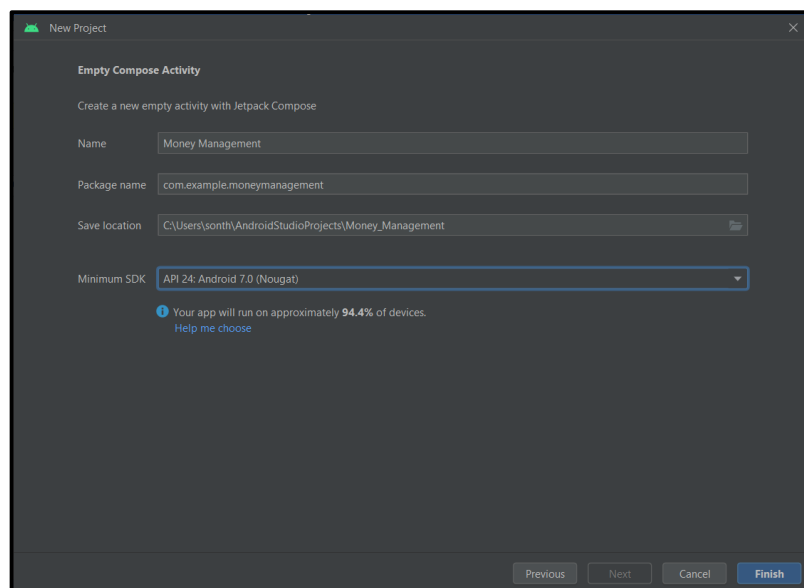
Bước 2: Trong hộp thoại "New Project", chọn "Empty Compose Activity" và bấm "Next".



Hình 2.3: Lựa chọn thiết lập Compose rồi được định nghĩa sẵn

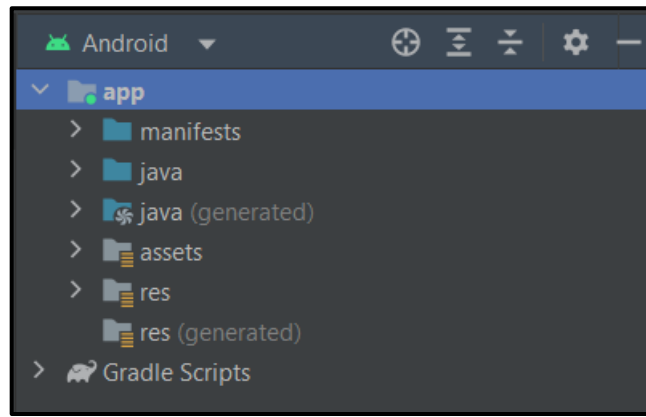
Bước 3:

- Đặt tên cho project của bạn và chọn nơi lưu trữ cho project.
- Chọn phiên bản Android và kiểu thiết bị mà bạn muốn hỗ trợ.
- Điền các thông tin cần thiết cho project, bao gồm tên ứng dụng, tên gói, ngôn ngữ lập trình và mức độ hỗ trợ.
- Chọn "Finish" để hoàn tất việc tạo project.



Hình 2.5: Bước cuối cùng để tạo thư mục dự án cho ứng dụng

Sau bước khởi tạo ban đầu, Android Studio sẽ tạo ra cho ta thư mục dự án



Hình 2.6: Các thư mục của dự án Android Studio khởi tạo

Trong hình 2.6, ta sẽ chủ yếu thao tác với thư mục “app”

- app/manifest: thư mục Manifest trong một dự án Android Studio chứa tệp manifest.xml. Theo cơ sở lý thuyết đã đề cập mục 2.3, manifest.xml chứa các thông tin quan trọng về ứng dụng của bạn để hệ điều hành Android có thể hiểu và quản lý các thành phần và quyền truy cập của ứng dụng.
- app/java: Nơi ta tiến hành cài đặt giao diện, cơ sở dữ liệu, các logic nghiệp vụ cho ứng dụng
- app/Gradle Scripts: Gradle Scripts là một thư mục trong dự án Android chứa các tệp cấu hình bản dựng cho dự án. Các tệp này được sử dụng để cấu hình bản dựng cho dự án, bao gồm các thành phần phụ thuộc (dependencies), công cụ bản dựng (build tools) và các cài đặt khác cần thiết để bản dựng và chạy ứng dụng.

2.1.2.2 Ngôn ngữ lập trình Kotlin

Kotlin là ngôn ngữ lập trình chính được khuyến khích và sử dụng rộng rãi trong lập trình ứng dụng Android hiện nay là Kotlin. Kotlin là một ngôn ngữ lập trình hiện đại, tương thích hoàn toàn với Java và được phát triển bởi JetBrains. Kotlin có cú pháp đơn giản và cung cấp nhiều tính năng tiện ích giúp lập trình viên tối ưu hóa quá trình phát triển ứng dụng Android.

Một số tính năng nổi bật của Kotlin bao gồm: [2]

- Null safety: Kotlin có hỗ trợ chặt chẽ về kiểm tra null (giá trị rỗng), giúp giảm thiểu lỗi truy xuất đến giá trị null trong quá trình chạy ứng dụng.
- Extension functions: Kotlin cho phép định nghĩa thêm các hàm mở rộng (extension functions) cho các lớp có sẵn trong hệ thống, giúp giảm thiểu việc lặp lại code.
- Data classes: Kotlin có hỗ trợ định nghĩa các lớp dữ liệu (data classes) với cú pháp đơn giản, giúp tạo ra các đối tượng dữ liệu dễ dàng hơn.
- Lambda expressions: Kotlin có hỗ trợ định nghĩa các hàm lambda (lambda expressions), giúp rút ngắn code và giảm thiểu việc lặp lại code.

- Coroutines: Lập trình không đồng bộ hoặc không chặn là một phần quan trọng trong quá trình phát triển phần mềm. Kotlin giải quyết vấn đề này một cách linh hoạt bằng cách cung cấp hỗ trợ coroutine ở cấp độ ngôn ngữ và ủy thác hầu hết chức năng cho các thư viện.

Với những tính năng tiện ích trên, Kotlin đã trở thành một lựa chọn phổ biến và được Google khuyến khích và ủng hộ trong việc phát triển ứng dụng Android.

2.1.3 Nền tảng của một ứng dụng Android

2.1.3.1 Các thành phần của ứng dụng

Các thành phần ứng dụng là các khối xây dựng thiết yếu của một ứng dụng Android. Mỗi thành phần là một điểm vào mà qua đó hệ thống hoặc người dùng có thể vào ứng dụng của bạn. Một số thành phần phụ thuộc vào những thành phần khác.

Có bốn loại thành phần ứng dụng khác nhau: [3]

- Activities

Một hoạt động (activity) là điểm tương tác với người dùng. Một hoạt động đại diện cho một màn hình duy nhất với giao diện người dùng. Ví dụ: một ứng dụng email có thể có một hoạt động hiển thị danh sách email mới, một hoạt động khác để soạn email và một hoạt động khác để đọc email. Mặc dù các hoạt động hoạt động cùng nhau để tạo thành trải nghiệm người dùng gắn kết trong ứng dụng email, nhưng mỗi hoạt động đều độc lập với các hoạt động khác.

Một hoạt động tạo điều kiện cho các tương tác chính sau đây giữa hệ thống và ứng dụng:

- Theo dõi những gì người dùng hiện đang quan tâm (những gì trên màn hình) để đảm bảo rằng hệ thống tiếp tục chạy quy trình lưu trữ hoạt động.
- Biết rằng các quy trình đã sử dụng trước đây chứa những thông tin mà người dùng có thể quay lại (các hoạt động đã dừng). Ưu tiên cao hơn để giữ các quy trình đó xung quanh.
- Giúp ứng dụng xử lý việc quá trình của nó bị tắt để người dùng có thể quay lại các hoạt động với trạng thái trước đó được khôi phục.
- Cung cấp một cách để các ứng dụng triển khai các luồng người dùng tương tác lẫn nhau và để hệ thống điều phối các luồng này. (Ví dụ như chia sẻ.)

Để triển khai một ứng dụng ta cần tạo ít nhất một lớp con kế thừa lớp “Activity” mặc định của hệ thống.

Đối với đề tài ứng dụng quản lý chi tiêu người dùng không quá phức tạp nên chỉ sử dụng duy nhất một hoạt động để đơn giản hoá quá trình cài đặt.

Vòng đời của một activity (hoạt động)

Khi người dùng điều hướng qua lại trong ứng dụng, Activity trong ứng dụng sẽ chuyển qua các trạng thái khác nhau trong vòng đời của chúng. Lớp Activity cung cấp một số

- + onStart(): Sẽ được gọi khi nó hiện hữu với người dùng.
- + onResume(): Sẽ được gọi khi người dùng tương tác với các ứng dụng.
- + onPause(): Tạm dừng một activity, không nhận dữ liệu do người dùng nhập vào và không thể thực thi lệnh nào. Phương thức này được gọi khi activity hiện tại đang được tạm dừng, và activity trước đó đang được tiếp tục.
- + onStop(): Được gọi khi một activity đã không được nhìn thấy trong thời gian dài.
- + onDestroy(): Được gọi trước khi hệ thống hủy activity.
- + onRestart(): Được gọi khi activity cần được dùng trở lại sau khi bị gọi onStop().

- Services

Dịch vụ (Service) là một điểm vào có mục đích chung để giữ cho ứng dụng chạy trong nền vì mọi lý do. Nó là một thành phần chạy trong nền để thực hiện các hoạt động chạy dài hoặc để thực hiện công việc cho các quy trình từ xa. Một dịch vụ không cung cấp giao diện người dùng. Ví dụ: một dịch vụ có thể phát nhạc ở chế độ nền trong khi người dùng đang ở trong một ứng dụng khác hoặc dịch vụ đó có thể tìm nạp dữ liệu qua mạng mà không chặn tương tác của người dùng với một hoạt động. Một thành phần khác, chẳng hạn như một hoạt động, có thể khởi động dịch vụ và cho phép nó chạy hoặc liên kết với nó để tương tác với nó.

- Broadcast receivers

Một broadcast receiver là một thành phần cho phép hệ thống gửi các sự kiện đến ứng dụng ngoài luồng sử dụng bình thường của người dùng, cho phép ứng dụng phản hồi với các thông báo phát sóng trên toàn hệ thống. Với broadcast receivers hệ thống có thể gửi thông báo phát sóng đến các ứng dụng mà hiện không chạy. Ví dụ, một ứng dụng có thể lên lịch một thông báo để đăng một thông báo cho người dùng về sự kiện sắp tới... Bằng cách gửi thông báo đó đến một BroadcastReceiver của ứng dụng, không cần ứng dụng phải chạy đến khi báo động được kích hoạt. Nhiều thông báo phát sóng bắt nguồn từ hệ thống - ví dụ, một thông báo thông báo rằng màn hình đã tắt, pin đang yếu hoặc một hình ảnh đã được chụp. Broadcast receivers không hiển thị giao diện người dùng, chúng có thể tạo ra một thông báo trên thanh trạng thái để cảnh báo người dùng khi một sự kiện phát sóng xảy ra.

- Content providers

Một nhà cung cấp nội dung (content provider) quản lý một tập hợp dữ liệu ứng dụng được lưu trữ trên hệ thống tập tin, trong cơ sở dữ liệu SQLite, trên web hoặc bất kỳ vị trí lưu trữ bền vững nào mà ứng dụng của bạn có thể truy cập. Qua nhà cung cấp nội dung, các ứng dụng khác có thể truy vấn hoặc sửa đổi dữ liệu nếu nhà cung cấp nội dung cho phép. Ví dụ, hệ thống Android cung cấp một nhà cung cấp nội dung quản lý thông tin liên lạc của người dùng. Do đó, bất kỳ ứng dụng nào có quyền truy cập thích hợp đều có thể truy vấn nhà cung cấp nội dung và lấy dữ liệu cần sử dụng.

Trước khi hệ thống Android có thể khởi động một thành phần ứng dụng, hệ thống phải biết rằng thành phần đó tồn tại bằng cách đọc tệp kê khai “AndroidManifest.xml” của ứng dụng. Ứng dụng phải khai báo tất cả các thành phần của nó trong tệp này, tệp này phải nằm ở thư mục gốc của thư mục dự án ứng dụng.

2.1.3.2 Tài nguyên ứng dụng

Một ứng dụng Android không chỉ bao gồm mã mà còn yêu cầu các tài nguyên tách biệt với mã nguồn, chẳng hạn như hình ảnh, tệp âm thanh và bất kỳ thứ gì liên quan đến phần trình bày trực quan của ứng dụng. [3] Ví dụ, ta có thể xác định hoạt ảnh, menu, kiểu, màu sắc và bố cục của giao diện người dùng hoạt động bằng các tệp XML. Việc sử dụng tài nguyên ứng dụng giúp bạn dễ dàng cập nhật các đặc điểm khác nhau của ứng dụng mà không cần sửa đổi mã. Việc cung cấp các bộ tài nguyên thay thế cho phép bạn tối ưu hóa ứng dụng của mình cho nhiều cấu hình thiết bị khác nhau, chẳng hạn như các ngôn ngữ và kích thước màn hình khác nhau.

Một trong những khía cạnh quan trọng nhất của việc cung cấp tài nguyên tách biệt với mã nguồn của bạn là khả năng cung cấp tài nguyên thay thế cho các cấu hình thiết bị khác nhau. Ví dụ: bằng cách xác định các chuỗi giao diện người dùng trong XML, bạn có thể dịch các chuỗi này sang các ngôn ngữ khác và lưu các chuỗi đó trong các tệp riêng biệt. Sau đó, Android áp dụng các chuỗi ngôn ngữ thích hợp cho giao diện người dùng của bạn dựa trên bộ định danh ngôn ngữ mà bạn thêm vào tên của thư mục tài nguyên và cài đặt ngôn ngữ của người dùng.

2.1.4 Android Jetpack

Android Jetpack là một bộ công cụ (toolkit) được phát triển bởi Google giúp cho việc phát triển ứng dụng Android trở nên dễ dàng hơn. Nó bao gồm một tập hợp các thành phần phần mềm (software components) được thiết kế để giúp cho các lập trình viên xây dựng ứng dụng Android một cách nhanh chóng, hiệu quả và dễ bảo trì hơn. [5]

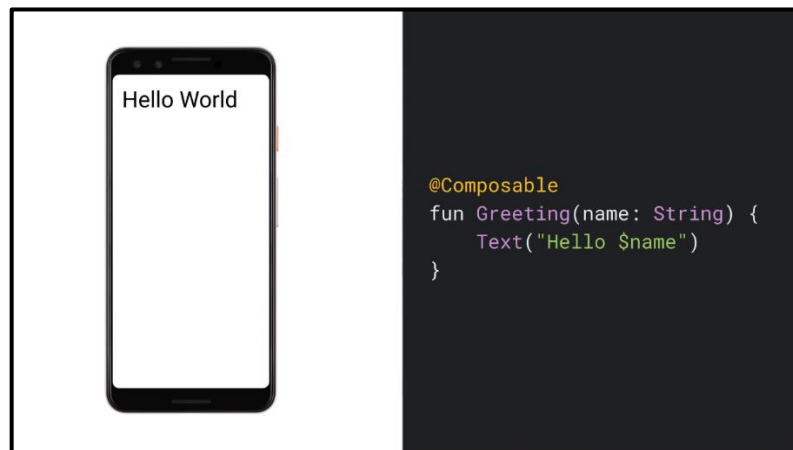
Android Jetpack bao gồm các thành phần phần mềm như Room, ViewModel, LiveData, Navigation, WorkManager... và nhiều thành phần khác nữa. Những thành phần này cung cấp các tính năng phổ biến và hữu ích cho việc phát triển ứng dụng Android như lưu trữ dữ liệu, quản lý vòng đời của các thành phần giao diện người dùng, điều hướng giữa các màn hình trong ứng dụng, lập lịch các tác vụ cần thực hiện, và rất nhiều tính năng khác nữa.

Sử dụng Android Jetpack giúp cho các lập trình viên có thể tập trung vào việc phát triển các tính năng chính của ứng dụng mà không phải lo lắng quá nhiều về những vấn đề phức tạp của việc quản lý vòng đời và các thành phần khác. Bên cạnh đó, Android Jetpack còn được cập nhật thường xuyên và được hỗ trợ bởi Google, đảm bảo tính ổn định và tiếp tục phát triển trong tương lai.

Ứng dụng quản lý chi tiêu cá nhân sẽ áp dụng một số thành phần của Android Jetpack.

2.1.4.1 Android Jetpack Compose

Compose là một trong những công nghệ mới của Android, giúp cho việc phát triển giao diện người dùng trở nên dễ dàng hơn bao giờ hết. Nó cho phép lập trình viên xây dựng giao diện bằng cách sử dụng các hàm và các thành phần được cung cấp bởi Compose. Với Jetpack Compose, việc xây dựng giao diện trở nên tương đối đơn giản và dễ hiểu, với việc sử dụng một hàm để biểu diễn giao diện gọi là Composable function. Ngoài ra, Jetpack Compose còn cung cấp các tính năng như hoạt ảnh, tự tùy chỉnh giao diện, quản lý trạng thái, và nhiều hơn nữa để giúp cho việc xây dựng ứng dụng trở nên hiệu quả và nhanh chóng hơn. Từ phiên bản Android Studio 4.0 trở đi, Jetpack Compose đã được tích hợp sẵn vào Android Studio và hỗ trợ việc phát triển ứng dụng Android Compose một cách nhanh chóng và thuận tiện. [6]



Hình 2.8: Hiển thị một dòng chữ chỉ bằng một hàm Composable. [6]

Một vài điều đáng chú ý về chức năng trong trường hợp trong ảnh 2.8 này:

- Chú thích `@Composable`: Tất cả các hàm Composable phải có chú thích này; chú thích này thông báo cho trình biên dịch Compose rằng chức năng này nhằm mục đích chuyển đổi dữ liệu thành giao diện người dùng.
- Các tham số đầu vào: Các hàm này có thể chấp nhận các tham số, cho phép logic ứng dụng mô tả giao diện người dùng. Trong trường hợp này, hàm Composable lấy tham số là một chuỗi để hiển thị đúng theo tên người dùng.
- Chức năng hiển thị văn bản trong giao diện người dùng: Để làm như vậy, ta gọi hàm Composable `Text()`, hàm này thực sự tạo ra phân tử giao diện người dùng văn bản. Các hàm Composable tạo ra hệ thống phân cấp giao diện người dùng bằng cách gọi các hàm Composable khác.
- Hàm Composable không trả về bất cứ thứ gì: Các chức năng soạn thảo phát ra giao diện người dùng không cần trả lại bất kỳ thứ gì, vì chúng mô tả trạng thái màn hình mong muốn thay vì xây dựng các tiện ích giao diện người dùng.

Các ưu điểm Compose sở hữu :

- Ít mã hơn: Thực hiện được nhiều chức năng hơn với ít mã hơn và tránh toàn bộ các lớp lỗi, giúp mã trở nên đơn giản và dễ bảo trì hơn.

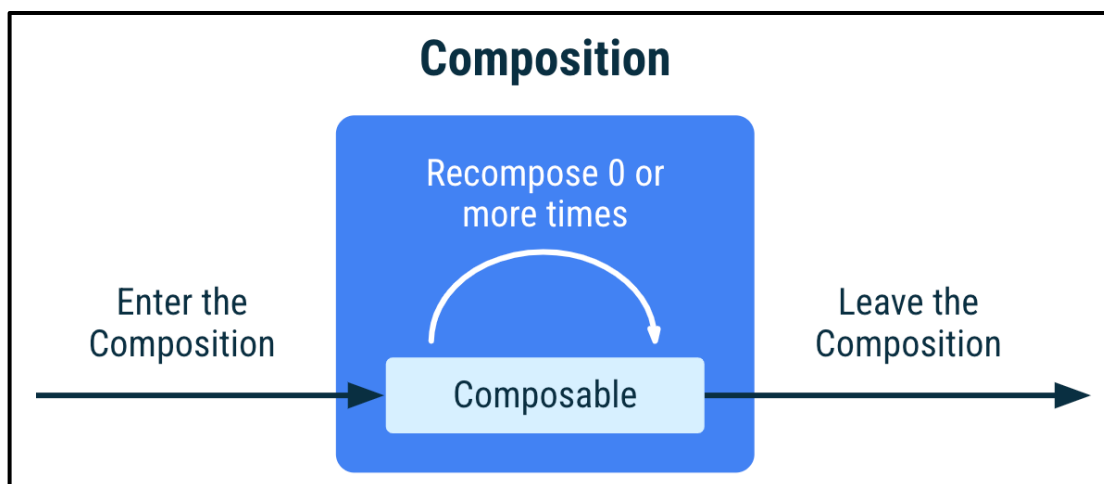
- **Thực quan:** Chỉ cần mô tả giao diện người dùng và Compose sẽ làm phần việc còn lại. Khi trạng thái ứng dụng thay đổi, giao diện người dùng sẽ tự động cập nhật trong phần hiển thị trước trong Android Studio.
- **Đẩy nhanh quá trình phát triển:** Tương thích với tất cả mã cũ, có thể áp dụng mọi lúc, mọi nơi. Lắp lại nhanh với bản xem trước trực tiếp và hỗ trợ Android Studio đầy đủ.
- **Mạnh mẽ:** Tạo các ứng dụng tuyệt vời có quyền truy cập trực tiếp vào API nền tảng Android và hỗ trợ tích hợp đối với Material Design, giao diện tối, ảnh động, v.v.

Vòng đời của Compose

Composition (tạm dịch là một bản tổng hợp) mô tả giao diện người dùng trên ứng dụng và được tạo ra bằng cách chạy các composables (được mô tả bởi các hàm Composable). [7] Một Composition là một cây cấu trúc của các composables mô tả giao diện người dùng.

Khi Jetpack Compose chạy các composables được lập trình lần đầu tiên, trong quá trình sơ tác ban đầu, nó sẽ theo dõi các composables mà ta gọi để mô tả giao diện người dùng của bạn trong một Composition. Sau đó, khi trạng thái của ứng dụng của bạn thay đổi, Jetpack Compose lên lịch một quá trình tái tạo. Quá trình tái tạo là khi Jetpack Compose thực hiện lại các composables có thể đã thay đổi để phản hồi với các trạng thái mới, và sau đó cập nhật Composition để phản ánh bất kỳ thay đổi nào.

Một Composition chỉ có thể được tạo ra bằng sơ tác ban đầu và được cập nhật bằng quá trình tái tạo. Cách duy nhất để sửa đổi một Composition là thông qua quá trình tái tạo.



Hình 2.9: Quá trình tổng hợp/ tái tạo lại một hàm Composable khi trạng thái mới được cập nhật. [7]

Một số thành phần xây dựng giao diện cơ bản của Jetpack Compose

a) Các thành phần UI

- **Text:** đây là thành phần cơ bản nhất để hiển thị văn bản trên màn hình. Các thuộc tính quan trọng của Text bao gồm font, kích thước, màu sắc và nội dung văn bản.

- **Button**: đây là thành phần cho phép người dùng tương tác với ứng dụng bằng cách nhấn vào nó. Các thuộc tính quan trọng của Button bao gồm văn bản trên nút, màu sắc và hành động khi người dùng nhấn vào nút (onClick).

- **Image**: đây là thành phần cho phép hiển thị hình ảnh trên màn hình. Các thuộc tính quan trọng của Image bao gồm đường dẫn đến hình ảnh, kích thước và các thuộc tính khác liên quan đến định dạng ảnh.

- **TextField**: đây là thành phần cho phép người dùng nhập liệu vào ứng dụng. Các thuộc tính quan trọng của TextField bao gồm tiêu đề, giá trị hiện tại và các hành động liên quan đến việc nhập liệu (onChange - khi đoạn văn trong ô nhập liệu thay đổi).

b) Bố cục (Layout) của các thành phần cơ bản

Các thành phần Layout trong Jetpack Compose giúp xác định cách sắp xếp và tổ chức các thành phần UI trên màn hình. Các thành phần cơ bản bao gồm Column, Row, LazyColumn và LazyRow.

- **Column**: Là một thành phần Layout cho phép các thành phần con được sắp xếp theo chiều dọc. Các thuộc tính quan trọng của Column bao gồm modifier (để thay đổi vị trí, kích thước, độ rộng, độ cao, ...), verticalArrangement (xác định cách sắp xếp các thành phần con theo chiều dọc), và horizontalAlignment (xác định cách sắp xếp các thành phần con theo chiều ngang).

- **LazyColumn**: Là một thành phần tương tự như Column, nhưng được thiết kế để xử lý dữ liệu động hoặc dữ liệu lớn hơn. Nó hiển thị chỉ một phần của nội dung trên màn hình và tải dữ liệu mới khi người dùng cuộn xuống dưới.

- **Row**: Là một thành phần Layout cho phép các thành phần con được sắp xếp theo chiều ngang. Các thuộc tính quan trọng của Row bao gồm modifier, horizontalArrangement (xác định cách sắp xếp các thành phần con theo chiều ngang), và verticalAlignment (xác định cách sắp xếp các thành phần con theo chiều dọc).

- **LazyRow**: Là một thành phần tương tự như Row và cơ chế hoạt động như LazyColumn.

c) Tham số Modifier trong các hàm Compose:

Trong Jetpack Compose, Modifier là một tham số quan trọng trong các hàm Compose. Nó giúp thay đổi và tùy chỉnh các thuộc tính của một thành phần UI, bao gồm vị trí, kích thước, độ rộng, độ cao, màu sắc, khoảng cách, độ trong suốt và hình dạng.

Một số phương thức thường được sử dụng của Modifier:

- **size**: thiết lập kích thước của một thành phần UI. Ta sử dụng nó để đặt kích thước tùy chỉnh hoặc chỉ định kích thước cố định cho một thành phần UI.

- **padding**: tạo khoảng cách giữa các thành phần UI. Ta sử dụng nó để đặt khoảng cách giữa các thành phần hoặc định vị các thành phần UI trong không gian bố trí.

- **background**: đặt màu nền cho một thành phần UI. Ta sử dụng nó để tạo nền màu đơn giản hoặc sử dụng các hiệu ứng hình ảnh phức tạp để tạo nền.
- **clickable**: tạo sự kiện click cho một thành phần UI. Ta sử dụng nó để thêm các hành động click cho các thành phần như nút, hộp chọn, trường văn bản và nhiều hơn nữa.
- **border**: thêm đường viền cho một thành phần UI. Ta sử dụng nó để tạo các hiệu ứng đường viền cho các thành phần như hộp chọn, nút, trường văn bản...
- **fillMaxSize**: điều chỉnh kích thước của một thành phần UI để nó phủ khắp không gian bố trí. Ta sử dụng nó để làm cho một thành phần UI đầy đủ kích thước hoặc để đặt kích thước cho các thành phần UI theo tỷ lệ nhất định.
- **align**: canh chỉnh một thành phần UI đến một vị trí cụ thể trong không gian bố trí. Ta sử dụng nó để canh chỉnh vị trí của các thành phần UI bên trong một bố cục Column hoặc Row.

2.1.4.2 Cơ sở dữ liệu trong ứng dụng Android

a) Về SQLite

SQLite là một công cụ cơ sở dữ liệu được viết bằng ngôn ngữ lập trình C. [8] SQLite không phải là một ứng dụng độc lập mà đúng hơn là một thư viện mà các nhà phát triển phần mềm nhúng vào ứng dụng của họ. Như vậy, SQLite thuộc họ cơ sở dữ liệu nhúng. Đây là công cụ cơ sở dữ liệu được triển khai rộng rãi nhất, vì nó được sử dụng bởi một số trình duyệt web, hệ điều hành, ứng dụng điện thoại di động - trong đó có ứng dụng chạy trên hệ điều hành Android và các hệ thống nhúng hàng đầu khác.

SQLite được thiết kế để cho phép chương trình hoạt động mà không cần cài đặt hệ thống quản lý cơ sở dữ liệu hoặc yêu cầu quản trị viên cơ sở dữ liệu. Thay vào đó, trình liên kết tích hợp thư viện SQLite — tĩnh hoặc động — vào một chương trình ứng dụng sử dụng chức năng của SQLite thông qua các lệnh gọi hàm đơn giản, giảm độ trễ trong các thao tác cơ sở dữ liệu; với các truy vấn đơn giản ít xảy ra đồng thời, hiệu năng SQLite thu được lợi nhuận từ việc tránh được chi phí giao tiếp giữa các quá trình.

b) Android Room Database

Thư viện Android Room Database là một phần của Android Jetpack và cung cấp một giải pháp lưu trữ cục bộ nhằm hỗ trợ cho ứng dụng của Android. [9] Nó giúp đơn giản hóa việc lưu trữ và truy cập dữ liệu, đồng thời giảm thiểu thủ tục lập trình và giúp quản lý cơ sở dữ liệu một cách hiệu quả.

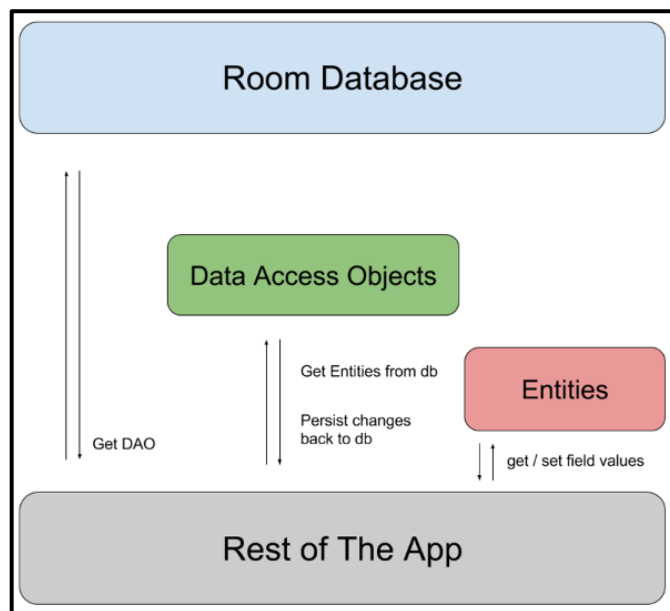
Room cho phép lập trình viên tạo các đối tượng cơ sở dữ liệu, các truy vấn dữ liệu và thực hiện các thao tác cơ bản như chèn, cập nhật và xóa dữ liệu. Nó sử dụng cấu trúc dữ liệu của SQLite nhưng cung cấp một lớp trừu tượng hơn để tương tác với cơ sở dữ liệu.

Room sử dụng cú pháp của các annotation để tạo các đối tượng cơ sở dữ liệu, các truy vấn dữ liệu và quản lý cơ sở dữ liệu. Các annotation này cho phép Room tự động tạo ra các đoạn mã để thực hiện các hoạt động cơ bản trên cơ sở dữ liệu và giúp giảm thiểu thời gian phát triển.

Có ba thành phần chính trong Room:

- Lớp cơ sở dữ liệu (database class) giữ cơ sở dữ liệu và đóng vai trò là điểm truy cập chính cho kết nối cơ sở dữ liệu bên dưới của ứng dụng của bạn.
- Thực thể dữ liệu (entity class) đại diện cho các bảng trong cơ sở dữ liệu của ứng dụng của bạn.
- Đối tượng truy cập dữ liệu (DAO – Data Access Object) cung cấp các phương thức mà ứng dụng của bạn có thể sử dụng để truy vấn, cập nhật, chèn và xóa dữ liệu trong cơ sở dữ liệu.

Lớp cơ sở dữ liệu cung cấp cho ứng dụng của bạn các phiên bản của các DAO liên quan đến cơ sở dữ liệu đó. Lúc đó, ứng dụng có thể sử dụng các DAO để truy xuất dữ liệu từ cơ sở dữ liệu dưới dạng các đối tượng thực thể dữ liệu liên quan. Ứng dụng cũng có thể sử dụng các thực thể dữ liệu được xác định để cập nhật các hàng từ các bảng tương ứng hoặc tạo hàng mới để chèn.



Hình 2.10: Mối quan hệ giữa các thành phần khác nhau của Room. [9]

2.1.5 Kiến trúc ứng dụng Android

Khi nói đến kiến trúc của một ứng dụng Android, người ta thường đề cập đến các nguyên tắc sau: [10]

- Nguyên tắc chia nhỏ các vấn đề
 - + Nguyên tắc quan trọng nhất là chia nhỏ các vấn đề ra. Điều thường gặp là viết toàn bộ mã trong một Activity hoặc Fragment (một lớp mô tả hành vi của một màn hình). Những lớp dựa trên giao diện người dùng này chỉ nên chứa các logic xử lý giao diện người dùng và tương tác hệ điều hành. Bằng cách giữ các lớp này càng nhẹ càng tốt, ta có thể tránh được nhiều vấn đề liên quan đến vòng đời của thành phần và cải thiện khả năng kiểm tra các lớp này.

+ Người lập trình viên không sở hữu các triển khai của Activity và Fragment; thay vào đó, chúng chỉ là các lớp dán giữ đại diện cho hợp đồng giữa hệ điều hành Android và ứng dụng của bạn. Hệ điều hành có thể phá hủy chúng bất cứ lúc nào dựa trên tương tác của người dùng hoặc vì các điều kiện hệ thống như bộ nhớ thấp. Để cung cấp trải nghiệm người dùng thỏa đáng và trải nghiệm bảo trì ứng dụng dễ quản lý hơn, tốt nhất là giảm thiểu sự phụ thuộc của bạn vào chúng.

- Thiết kế giao diện người dùng từ mô hình dữ liệu:

+ Một nguyên tắc quan trọng khác là bạn nên thiết kế giao diện người dùng từ các mô hình dữ liệu, ưu tiên hơn hết là các mô hình bền vững. Mô hình dữ liệu đại diện cho dữ liệu của một ứng dụng. Chúng không liên kết với giao diện người dùng và vòng đời của thành phần ứng dụng, nhưng vẫn sẽ bị phá hủy khi hệ điều hành quyết định loại bỏ quá trình của ứng dụng khỏi bộ nhớ.

+ Các mô hình bền vững lý tưởng cho những lý do sau:

* Người dùng của bạn sẽ không mất dữ liệu nếu hệ điều hành Android phá hủy ứng dụng của bạn để giải phóng tài nguyên.

* Ứng dụng của bạn tiếp tục hoạt động trong trường hợp kết nối internet chậm chạp hoặc không hoạt động

- Single Source of Truth – Một nơi lưu trữ dữ liệu duy nhất:

+ Khi một loại dữ liệu mới được định nghĩa trong ứng dụng của bạn, bạn nên chỉ định một Single Source of Truth (SSOT) cho nó. SSOT là chủ sở hữu của dữ liệu đó và chỉ SSOT mới có thể sửa đổi hoặc thay đổi nó. Để đạt được điều này, SSOT tiết lộ dữ liệu bằng cách sử dụng một kiểu không thay đổi, và để sửa đổi dữ liệu, SSOT tiết lộ các chức năng hoặc sự kiện để các loại khác có thể gọi.

+ Mô hình này mang lại nhiều lợi ích:

* Giúp tập trung tất cả các thay đổi vào một loại dữ liệu cụ thể trong một nơi.

* Bảo vệ dữ liệu để bên ngoài không thể can thiệp vào nó.

* Làm cho các thay đổi dữ liệu dễ theo dõi hơn. Do đó, các lỗi được phát hiện dễ dàng hơn.

Trong ứng dụng ưu tiên ngoại tuyến, nguồn đáng tin cậy cho dữ liệu ứng dụng thường là một cơ sở dữ liệu. Trong một số trường hợp khác, nguồn đáng tin cậy có thể là ViewModel hoặc thậm chí là giao diện người dùng.

- Luồng dữ liệu một chiều (Unidirectional Data Flow):

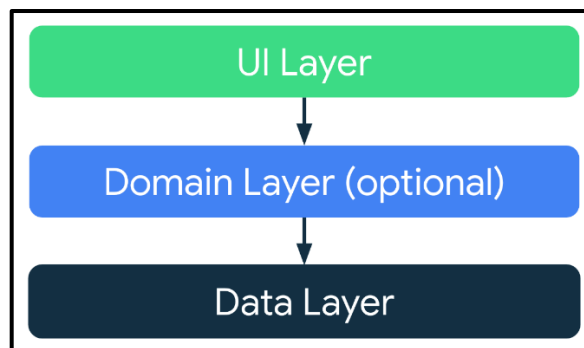
+ Trong nguyên tắc luồng dữ liệu một chiều (UDF), trạng thái chỉ chảy theo một hướng. Các sự kiện sửa đổi luồng dữ liệu theo hướng ngược lại.

+ Trong Android, trạng thái hoặc dữ liệu thường chảy từ các loại phân cấp có phạm vi cao hơn sang các loại có phạm vi thấp hơn. Các sự kiện thường được kích hoạt từ các loại có phạm vi thấp hơn cho đến khi chúng đạt đến SSOT cho loại dữ liệu tương ứng. Ví dụ: dữ liệu ứng dụng thường chuyển từ nguồn dữ liệu sang giao diện người dùng. Các sự kiện của người dùng, chẳng hạn như các lần nhấn nút, chuyển từ giao diện người dùng sang SSOT nơi dữ liệu ứng dụng được sửa đổi và hiển thị ở một loại không thay đổi.

+ Nguyên tắc này đảm bảo tính nhất quán của dữ liệu tốt hơn, ít bị lỗi hơn, dễ gỡ lỗi hơn và mang lại tất cả các lợi ích của mẫu SSOT.

Kiến trúc dữ liệu được khuyến dùng

Xem xét các nguyên tắc kiến trúc phổ biến được đề cập trong phần trước, mỗi ứng dụng nên có ít nhất hai lớp: [10]



Hình 2.11: Biểu đồ mô tả kiến trúc ứng dụng đặc trưng. [10]

- Lớp giao diện (UI Layer) người dùng hiển thị dữ liệu ứng dụng trên màn hình.
- Lớp dữ liệu (Data Layer) chứa logic xử lý của ứng dụng của bạn và hiển thị dữ liệu ứng dụng.

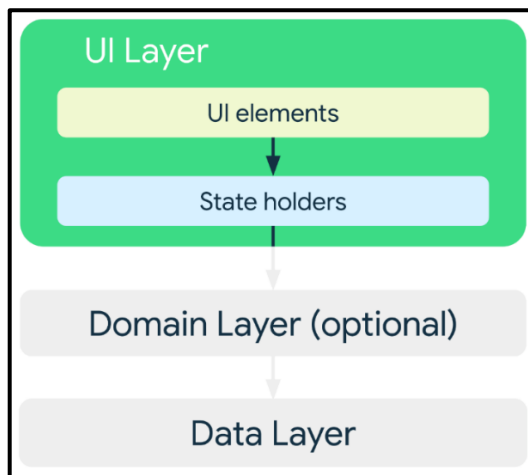
Có thể thêm một lớp bổ sung được gọi là lớp miền (Domain Layer) để đơn giản hóa và sử dụng lại các tương tác giữa giao diện người dùng và lớp dữ liệu. Ứng dụng quản lý chi tiêu trong bài báo cáo không sử dụng các logic nghiệp vụ quá phức tạp nên chưa cần thiết phải sử dụng lớp miền trong kiến trúc này.

a) Lớp giao diện (UI Layer)

Vai trò của lớp giao diện người dùng là hiển thị dữ liệu ứng dụng trên màn hình. Bất cứ khi nào dữ liệu thay đổi, do tương tác của người dùng (chẳng hạn như nhấn nút) hoặc đầu vào bên ngoài (chẳng hạn như phản hồi của mạng), giao diện người dùng sẽ cập nhật để phản ánh các thay đổi.

Lớp giao diện người dùng được tạo thành từ hai thứ:

- Các thành phần giao diện người dùng hiển thị dữ liệu trên màn hình.
- Lớp nắm giữ trạng thái (chẳng hạn như các lớp ViewModel), hiển thị các trạng thái đó cho giao diện người dùng và xử lý logic.



Hình 2.12: Mô tả chức năng của lớp giao diện người dùng trong kiến trúc ứng dụng. [10]

b) Lớp dữ liệu (Data Layer)

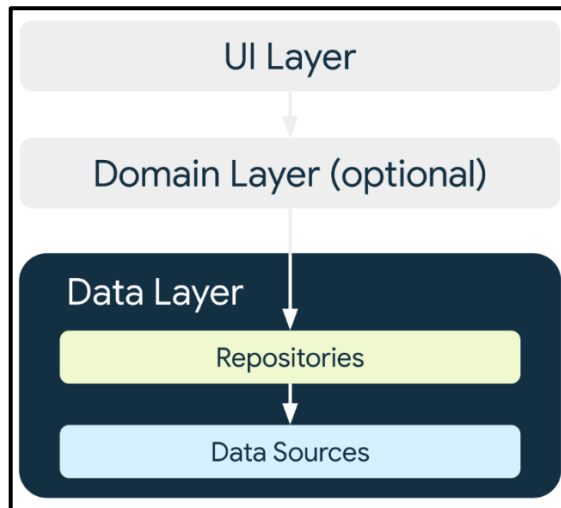
Lớp dữ liệu của ứng dụng chứa logic nghiệp vụ. Logic nghiệp vụ được tạo thành từ các quy tắc xác định cách ứng dụng của lập trình viên tạo, lưu trữ và thay đổi dữ liệu (cập nhật hoặc xóa).

Lớp dữ liệu được tạo thành từ các kho lưu trữ (repository) mà mỗi kho có thể chứa từ 0 đến nhiều nguồn dữ liệu. Nên tạo một lớp kho lưu trữ cho từng loại dữ liệu khác nhau cần xử lý trong một ứng dụng. Ví dụ: ta có thể tạo lớp “kho lưu trữ phim” (Movies Repository) cho dữ liệu liên quan đến phim hoặc lớp “kho lưu trữ giao dịch” (Transactions Repository) cho dữ liệu liên quan đến thanh toán.

Các lớp kho lưu trữ chịu trách nhiệm cho các nhiệm vụ sau:

- Hiện thị dữ liệu cho phần còn lại của ứng dụng.
- Tập trung các thay đổi vào dữ liệu.
- Giải quyết xung đột giữa nhiều nguồn dữ liệu.
- Tóm tắt các nguồn dữ liệu từ phần còn lại của ứng dụng.
- Chứa logic nghiệp vụ.

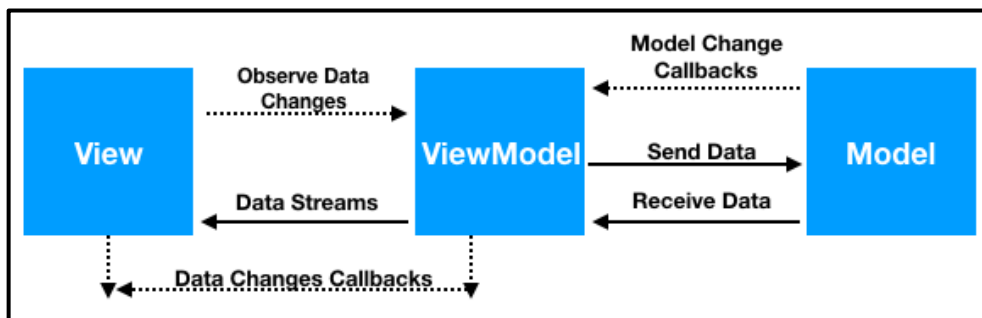
Mỗi lớp nguồn dữ liệu phải có trách nhiệm làm việc với chỉ một nguồn dữ liệu, có thể là tệp, nguồn mạng hoặc cơ sở dữ liệu cục bộ. Các lớp nguồn dữ liệu là cầu nối giữa ứng dụng và hệ thống cho các hoạt động dữ liệu.



Hình 2.13: Mô tả lớp dữ liệu trong kiến trúc ứng dụng [10]

2.1.6 Mô hình Model, View, ViewModel (MVVM)

MVVM (Model-View-ViewModel) là một mô hình thiết kế phần mềm được sử dụng rộng rãi và được các chuyên gia khuyên dùng trong lập trình ứng dụng Android. [11]



Hình 2.14: Mô tả mô hình thiết kế lập trình MVVM. [11]

MVVM chia ứng dụng thành ba phần chính:

- Model: Là phần chứa dữ liệu và logic xử lý dữ liệu. Trong phần này, ta thường sử dụng các thư viện hoặc kết nối tới server để lấy dữ liệu.
- View: Là phần hiển thị giao diện đồ họa cho người dùng. View sẽ liên kết với ViewModel để hiển thị dữ liệu và truyền tín hiệu điều khiển đến ViewModel.
- ViewModel: Là phần trung gian giữa Model và View. ViewModel chứa logic xử lý và xử lý tác vụ, đồng thời cũng cung cấp dữ liệu cho View hiển thị. ViewModel sử dụng các công nghệ quản lý bộ nhớ đệm để lưu trữ và xử lý dữ liệu và thông báo cho View khi có thay đổi dữ liệu.

ViewModel

Một lựa chọn khác với ViewModel là việc khai báo một lớp giữ toàn bộ dữ liệu được hiển thị trên giao diện người dùng. Điều này có thể trở thành một vấn đề khi điều hướng giữa các hoạt động của ứng dụng. Làm như vậy phá hủy toàn bộ dữ liệu tại thời điểm trước khi chuyển hướng và cần phải thiết lập cơ chế lưu giữ “instance” – thực thể của

lớp giữ liệu hiện tại, một cách thủ công. ViewModel cung cấp API thuận tiện cho sự tồn tại của dữ liệu để giải quyết vấn đề này. [12]

Những lợi ích chính của lớp ViewModel về được chia làm 2 thành phần: [12]

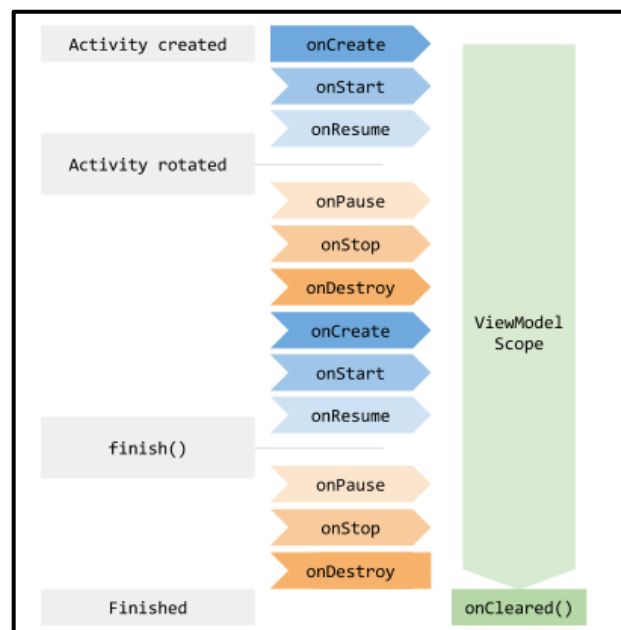
- Cho phép trạng thái hiển thị tồn tại liên tục: ViewModel cho phép dữ liệu tồn tại thông qua cả trạng thái mà ViewModel nắm giữ và hoạt động mà ViewModel kích hoạt. Cơ chế bộ nhớ đệm này giúp lập trình viên không phải tìm nạp lại dữ liệu thủ công thông qua các thay đổi cấu hình phổ biến, chẳng hạn như xoay màn hình.
- Cung cấp quyền truy cập vào công việc logic của dữ liệu: ViewModel là nơi lý tưởng để xử lý logic trong lớp giao diện người dùng. ViewModel cũng chịu trách nhiệm xử lý các sự kiện và ủy thác chúng cho các lớp khác của hệ thống phân cấp khi các logic cần được áp dụng để sửa đổi dữ liệu ứng dụng.

Vòng đời của ViewModel

Vòng đời của một ViewModel liên kết trực tiếp với phạm vi của nó. Một ViewModel vẫn tồn tại trong bộ nhớ cho đến khi ViewModelStoreOwner (Một giao diện – interface, để cài đặt lớp lưu trữ các ViewModel) mà nó được liên kết biến mất. Điều này có thể xảy ra trong các ngữ cảnh sau:

- Trong trường hợp của một activity, khi nó kết thúc.
- Trong trường hợp của một fragment (một mảnh của 1 activity), khi nó tách ra.
- Trong trường hợp của một điểm điều hướng, khi nó bị xóa khỏi back stack (một stack lưu trữ các trang chuyển hướng).

Điều này làm cho ViewModel trở thành một giải pháp tuyệt vời để lưu trữ dữ liệu tồn tại qua các thay đổi cấu hình.



Hình 2.15: trạng thái vòng đời khác nhau của một activity khi nó trải qua quá trình xoay và sau đó được kết thúc. [12]

Hình 2.15 minh họa các trạng thái vòng đời khác nhau của một activity khi nó trải qua quá trình xoay và sau đó được kết thúc. Hình minh họa cũng cho thấy thời gian tồn tại của ViewModel bên cạnh vòng đời của activity tương ứng. Sơ đồ này cụ thể minh họa các trạng thái của một activity. Các trạng thái cơ bản tương tự áp dụng cho vòng đời của một fragment. [12]

Thông thường ta yêu cầu một ViewModel lần đầu tiên khi hệ thống gọi phương thức onCreate() của đối tượng activity. Hệ thống có thể gọi onCreate() nhiều lần trong suốt sự tồn tại của một activity, chẳng hạn như khi màn hình thiết bị được xoay. ViewModel tồn tại từ khi bạn yêu cầu một ViewModel lần đầu tiên cho đến khi activity hoàn thành và bị hủy.

2.2 Phân tích thiết kế

2.2.1 Phân tích thiết kế dữ liệu

Các tập thực thể và thuộc tính

Thực thể	Thuộc tính	Kiểu dữ liệu	Chú thích
Category (Thẻ loại cha)	category_id (PK)	INTEGER	Đại diện cho thể loại của các giao dịch, mỗi bản ghi trong bảng này sẽ có thông tin về tên thẻ loại và biểu tượng của danh mục. Ví dụ, các danh mục có thể là "Ăn uống", "Di chuyển", "Giải trí", "Mua sắm"...
	category_name	TEXT	
	category_icon_name	TEXT	
Subcategory (Thẻ loại con)	subcategory_id (PK)	INTEGER	Đại diện cho các danh mục con trong từng danh mục, mỗi bản ghi trong bảng này sẽ có thông tin về tên danh mục con và danh mục chính mà nó thuộc về. Ví dụ, trong danh mục "Ăn uống" có thể có các danh mục con như "Ăn sáng", "Ăn trưa"...
	subcategory_name	TEXT	
	category_id (FK)	INTEGER	
transaction_table (Giao dịch)	transaction_id (PK)	INTEGER	Đại diện cho các giao dịch, mỗi bản ghi trong bảng này sẽ có thông tin về tên giao dịch, số tiền, ngày thực hiện giao dịch, ghi chú và danh mục, danh mục con mà giao dịch thuộc về. Ví dụ, một giao dịch có thể là "Mua sữa tắm tại siêu thị", với số tiền là 50.000đ, ngày thực hiện là 16/03/2023, và thuộc danh mục
	transaction_name	TEXT	
	transaction_amount	REAL	
	transaction_date	TEXT	

	transaction_note	TEXT	"Mua sắm" và danh mục con "Vật dụng nhà tắm".
	category_id (FK)	INTEGER	
	subcategory_id (FK)	INTEGER	

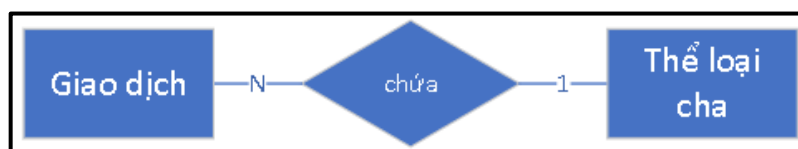
Các mối quan hệ giữa các thực thể

a) Category (Thẻ loại cha) – Subcategory (thẻ loại con)



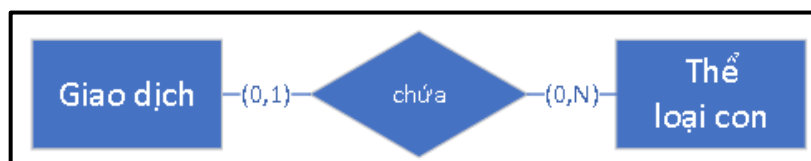
- Mỗi thẻ loại cha có thể chia ra thành nhiều thẻ loại con
- Mỗi một thẻ loại con chỉ thuộc về duy nhất 1 thẻ loại cha

b) Transaction (Giao dịch) – Category (Thẻ loại cha)



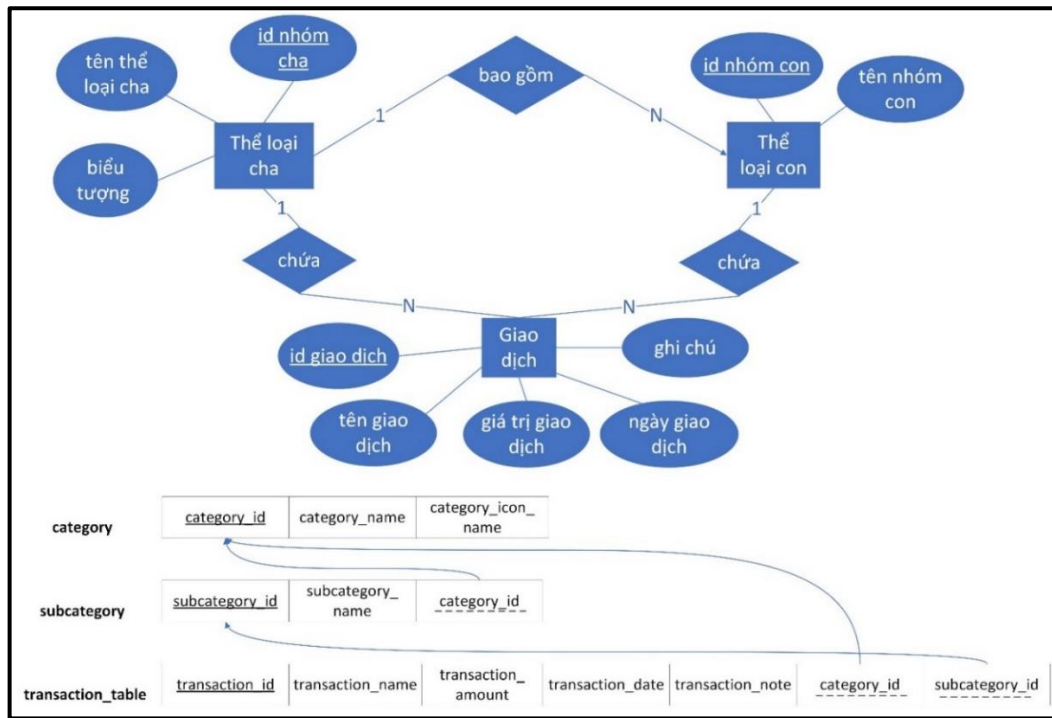
- Một giao dịch chỉ có thể được phân loại là một thẻ loại cha
- Một thẻ loại cha có thể tham gia vào nhiều giao dịch khác nhau

c) Transaction (Giao dịch) – Subcategory (Thẻ loại con)



- Một giao dịch chỉ có thể được phân loại là tối đa 1 thẻ loại con. Một giao dịch có thể không cần định nghĩa thẻ loại con, hay định nghĩa thẻ loại con cho một giao dịch là không bắt buộc
- Một thẻ loại con có thể tham gia vào nhiều giao dịch khác nhau, nhưng cũng có thể không tham gia vào giao dịch nào

Lược đồ ER và mô hình quan hệ



Hình 2.16: Lược đồ ER mô tả các thực thể, các thuộc tính và các quan hệ

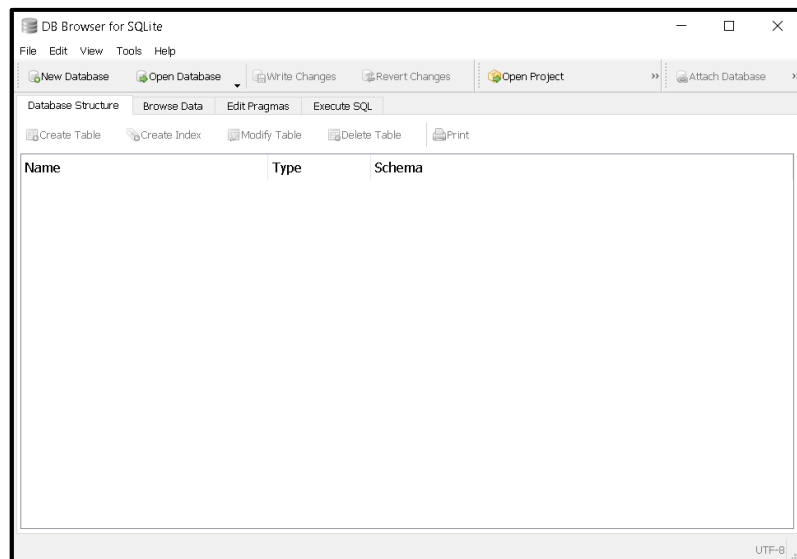
Cài đặt sẵn tệp cơ sở dữ liệu trên SQLite

Để thuận tiện cho việc cài đặt trên ứng dụng Android, ta sẽ cài đặt trước cơ sở dữ liệu trên SQLite, cùng với đó ta sẽ thêm một số dữ liệu mặc định trên cơ sở dữ liệu (ví dụ như một số bản ghi trong “category” và “subcategory”)

Để giúp cho việc thao tác cơ sở dữ liệu trở nên dễ dàng hơn, ta có thể sử dụng phần mềm [DB Browser for SQLite](#). Phần mềm này cung cấp giao diện đồ họa và dễ sử dụng để có thể thao tác trực tiếp với cơ sở dữ liệu qua màn hình.

Các bước cài đặt

Bước 1: Cài đặt và mở ứng dụng quản lý cơ sở dữ liệu [DB Browser for SQLite](#)

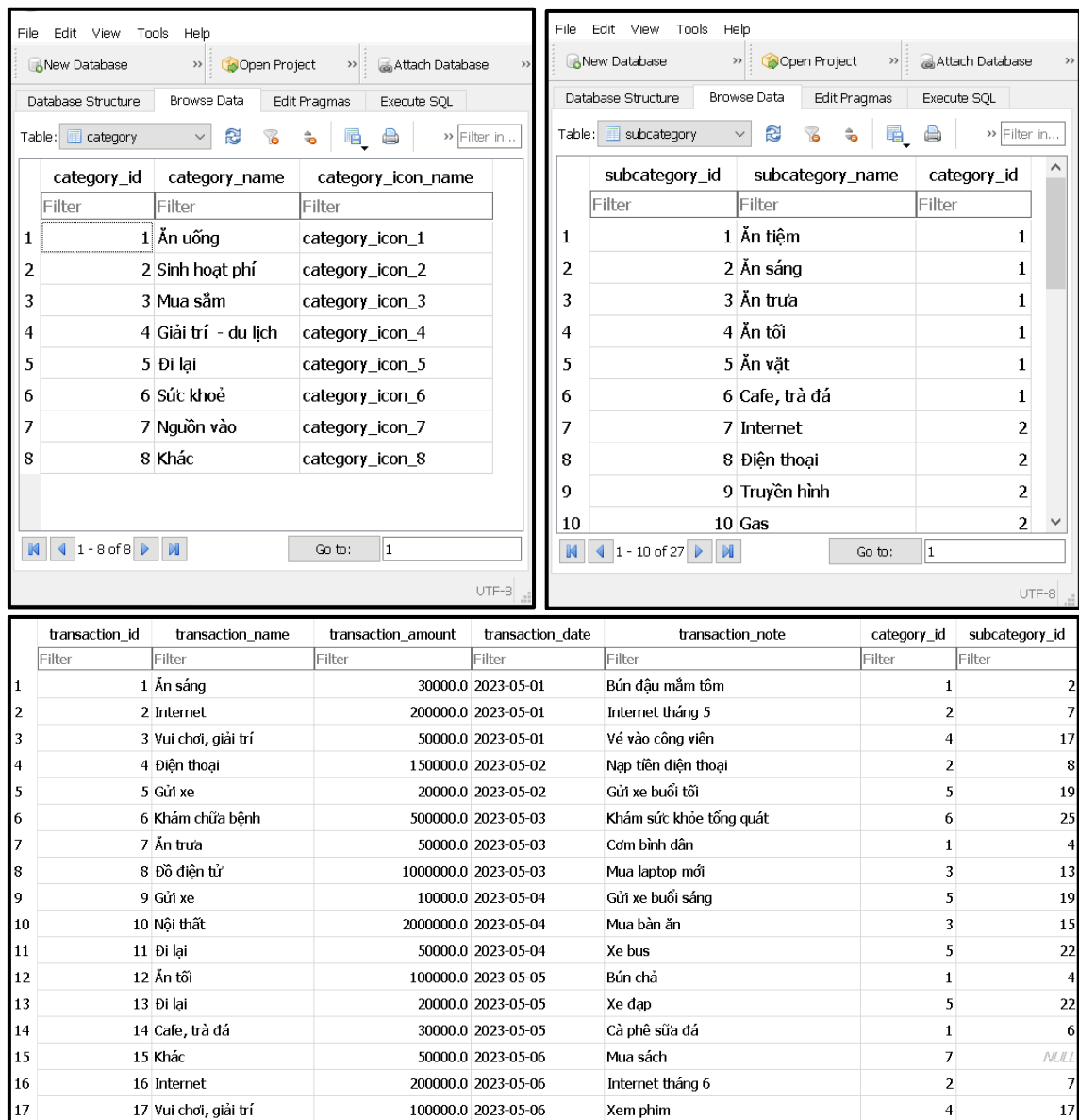


Hình 3.1: Giao diện người dùng của DB Browser for SQLite

Bước 2: Tạo cơ sở dữ liệu mới với tên “money_management.db” và lưu trữ ở một vị trí nào đó để sau này sử dụng và sử dụng SQL để tạo cơ sở dữ liệu hoàn chỉnh như đã mô tả và thiết kế ở phần trước .

Bước 3: Sử dụng câu lệnh “INSERT INTO” để tạo sẵn các bản ghi cho bảng “category” và các bản ghi trong bảng “subcategory” tương ứng. Để thuận tiện cho quá trình kiểm thử, ta tạo khoảng 50 bản ghi dữ liệu cho “transaction_table” .

Sau 3 bước trên, ta đã tạo được cơ sở dữ liệu “money_management.db” để nhúng vào ứng dụng Android. Kết quả của quá trình được hiển thị trên DB Browser for SQLite như sau:



category_id	category_name	category_icon_name
1	Ăn uống	category_icon_1
2	Sinh hoạt phí	category_icon_2
3	Mua sắm	category_icon_3
4	Giải trí - du lịch	category_icon_4
5	Đi lại	category_icon_5
6	Sức khỏe	category_icon_6
7	Nguồn vào	category_icon_7
8	Khác	category_icon_8

subcategory_id	subcategory_name	category_id
1	Ăn tiệm	1
2	Ăn sáng	1
3	Ăn trưa	1
4	Ăn tối	1
5	Ăn vặt	1
6	Cafe, trà đá	1
7	Internet	2
8	Điện thoại	2
9	Truyền hình	2
10	Gas	2

transaction_id	transaction_name	transaction_amount	transaction_date	transaction_note	category_id	subcategory_id
1	Ăn sáng	30000.0	2023-05-01	Bún đậu mắm tôm	1	2
2	Internet	200000.0	2023-05-01	Internet tháng 5	2	7
3	Vui chơi, giải trí	50000.0	2023-05-01	Vé vào công viên	4	17
4	Điện thoại	150000.0	2023-05-02	Nạp tiền điện thoại	2	8
5	Gửi xe	20000.0	2023-05-02	Gửi xe buổi tối	5	19
6	Khám chữa bệnh	500000.0	2023-05-03	Khám sức khỏe tổng quát	6	25
7	Ăn trưa	50000.0	2023-05-03	Cơm bình dân	1	4
8	Đồ điện tử	1000000.0	2023-05-03	Mua laptop mới	3	13
9	Gửi xe	10000.0	2023-05-04	Gửi xe buổi sáng	5	19
10	Nội thất	2000000.0	2023-05-04	Mua bàn ăn	3	15
11	Đi lại	50000.0	2023-05-04	Xe bus	5	22
12	Ăn tối	100000.0	2023-05-05	Bún chả	1	4
13	Đi lại	20000.0	2023-05-05	Xe đạp	5	22
14	Cafe, trà đá	30000.0	2023-05-05	Cà phê sữa đá	1	6
15	Khác	50000.0	2023-05-06	Mua sách	7	NULL
16	Internet	200000.0	2023-05-06	Internet tháng 6	2	7
17	Vui chơi, giải trí	100000.0	2023-05-06	Xem phim	4	17

Hình 3.2: Kết quả cài đặt cơ sở dữ liệu trên DB Browser for SQLite

2.2.2 Chức năng các màn hình

2.2.2.1 Màn hình chính (Home Screen)

- Hiện thị danh sách các giao dịch đã được thêm vào hệ thống, sắp xếp theo thứ tự thời gian từ mới nhất đến cũ hơn. Các lịch sử giao dịch có chung thời gian tháng trong năm được gom chung lại. Mỗi phần tử lịch sử giao dịch sẽ hiển thị: ảnh đại diện thể loại giao dịch cha, tên lịch sử giao dịch, ngày giao dịch và số tiền.
- Cung cấp chức năng tìm kiếm để người dùng có thể tìm kiếm các giao dịch theo tên.
- Đặt bộ lọc để người dùng có thể lọc danh sách lịch sử giao dịch theo thể loại lịch sử giao dịch cha/con, giúp người dùng dễ dàng nhìn thấy các giao dịch thuộc cùng một loại.

2.2.2.2 Màn hình thêm lịch sử giao dịch

- Hiện thị một giao diện gồm các trường nhập liệu và danh sách thông tin, cho phép người dùng nhập tên, ngày giao dịch, số tiền, ghi chú và thể loại về giao dịch mới.
- Kiểm tra các trường thông tin người dùng thêm vào là hợp lệ trước khi người dùng xác nhận thêm và hệ thống thêm giao dịch mới CSDL.

2.2.2.3 Màn hình xem chi tiết lịch sử giao dịch

- Hiện thị thông tin chi tiết về một giao dịch cụ thể, bao gồm: ảnh đại diện thể loại, tên giao dịch, số tiền, thể loại cha/con, ngày giao dịch và ghi chú.
- Cung cấp chức năng sửa lịch sử giao dịch, cho phép người dùng chỉnh sửa thông tin giao dịch được chọn.
- Cung cấp chức năng xoá lịch sử giao dịch, cho phép người dùng xoá giao dịch được chọn trong CSDL.

2.2.2.4 Màn hình sửa lịch sử giao dịch

Màn hình này tái sử dụng giao diện của màn hình thêm lịch sử giao dịch

- Hiện thị giao diện của màn hình thêm lịch sử giao dịch với các trường dữ liệu được điền sẵn dựa trên lịch sử giao dịch đã được chọn
- Kiểm tra các trường thông tin người dùng sửa đổi là hợp lệ trước khi người dùng xác nhận sửa đổi và hệ thống sửa thông tin của bản ghi giao dịch này trong CSDL.

2.2.2.5 Màn hình thống kê lịch sử giao dịch

- Hiện thị danh sách các lịch sử giao dịch và tổng số tiền đã chi tiêu trong khoảng thời gian từ ngày bắt đầu đến ngày kết thúc mà người dùng đã chọn.
- Hiện thị biểu đồ cột, thống kê tổng số tiền chi tiêu qua từng tuần từ ngày bắt đầu đến ngày kết thúc mà người dùng đã chọn.

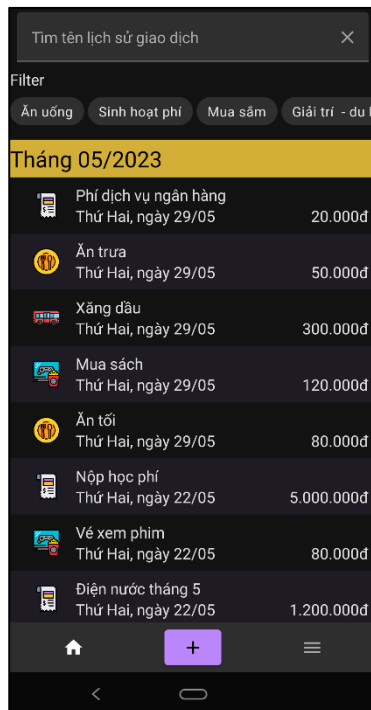
3. ỨNG DỤNG THỰC NGHIỆM

Các giao diện trên chương trình

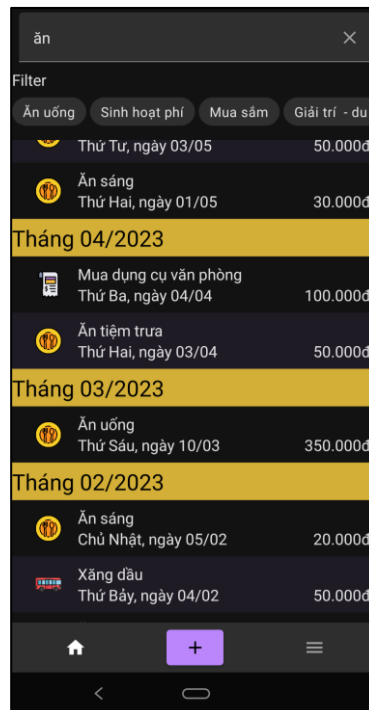
3.1. Home Screen (Màn hình chính)

Sau khi khởi động ứng dụng, hệ thống sẽ hiển thị màn hình HomeScreen. Màn hình này có các chức năng sau:

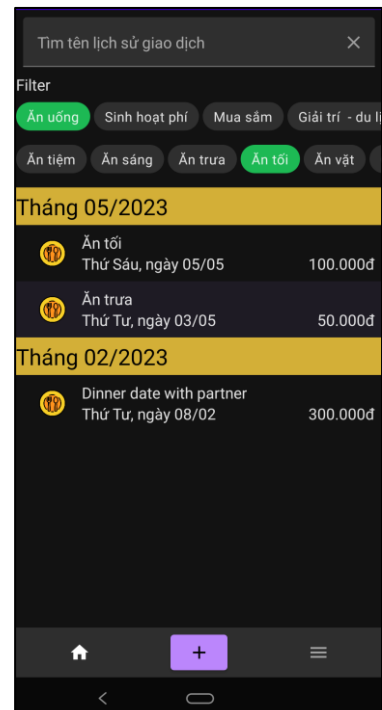
- Xem toàn bộ lịch sử giao dịch. Các giao dịch được nhóm theo tháng trong năm và hiển thị theo giao dịch mới nhất từ trên xuống dưới
- Tìm kiếm lịch sử giao dịch theo tên của giao dịch
- Lọc danh sách lịch sử giao dịch theo thể loại của giao dịch



Hình 3.3: Giao diện chính mặc định



Hình 3.4: Hiển thị quả tìm kiếm theo tên



Hình 3.5: Hiện thị qua bộ lọc theo thể loại

Màn hình này có thể được gọi khi người dùng bấm biểu tượng hình ngôi nhà ở thanh điều hướng bên dưới



3.2 Màn hình thêm lịch sử giao dịch

Màn hình này được kích hoạt khi người dùng bấm nút “+” ở thanh điều hướng bên dưới



Màn hình thêm lịch sử giao dịch giúp người dùng nhập những thông tin cần thiết mỗi khi đã bỏ ra một khoản chi tiêu. Các ô nhập liệu với tiêu đề có “*” là yêu cầu bắt buộc.

Để đảm bảo tính nhất quán của dữ liệu ngày giao dịch, ta sẽ cung cấp cho người dùng giao diện chọn ngày khi người dùng cần nhập trường dữ liệu này

Transaction Entry Screen

Tên giao dịch

Ngày giao dịch*

Số tiền*

Ghi chú

Phân loại giao dịch*

Ăn uống Sinh hoạt phí Mua sắm Giải trí - du lịch Đi lại

Mở rộng

testing
-1

Save

Hình 3.7: Giao diện mặc định khi truy cập

Transaction Entry Screen

Tên giao dịch

Ngày giao dịch*

Số tiền*

Ghi chú

Phân loại giao dịch*

Ăn uống Sinh hoạt phí Mua sắm Giải trí - du lịch Đi lại

Mở rộng

testing
-1

Save

Hình 3.8: Khi người dùng nhập ô ngày giao dịch

Để đảm bảo trường số tiền được người dùng giá trị hợp lệ (là một số và số này không âm), hệ thống sẽ thực hiện xác thực mỗi khi trường nhập liệu số tiền được nhập

Nút “Save” sẽ được hiện lên khi người dùng nhập đủ các trường thông tin yêu cầu (có đánh dấu *) và dữ liệu các trường này là hợp lệ. Khi người dùng click nút “Save”, hệ thống sẽ đóng gói và lưu thông tin của giao dịch vào CSDL và điều hướng về màn hình chính.

Transaction Entry Screen

Tên giao dịch

Ngày giao dịch*

Thứ Sáu, ngày 19/05

Số tiền*

-10000

Vui lòng nhập vào một số > 0.

Ghi chú

Phân loại giao dịch*

Ăn uống Sinh hoạt phí Mua sắm Giải trí - du lịch Đi lại

Mở rộng

Ăn tối

testing
1
4

Save

Hình 3.9: Hệ thống hiện lỗi khi input trường số tiền không hợp lệ

Transaction Entry Screen

Tên giao dịch

Ngày giao dịch*

Thứ Sáu, ngày 19/05

Số tiền*

10000

Ghi chú

Phân loại giao dịch*

Ăn uống Sinh hoạt phí Mua sắm Giải trí - du lịch Đi lại

Mở rộng

Ăn tối

testing
1
4

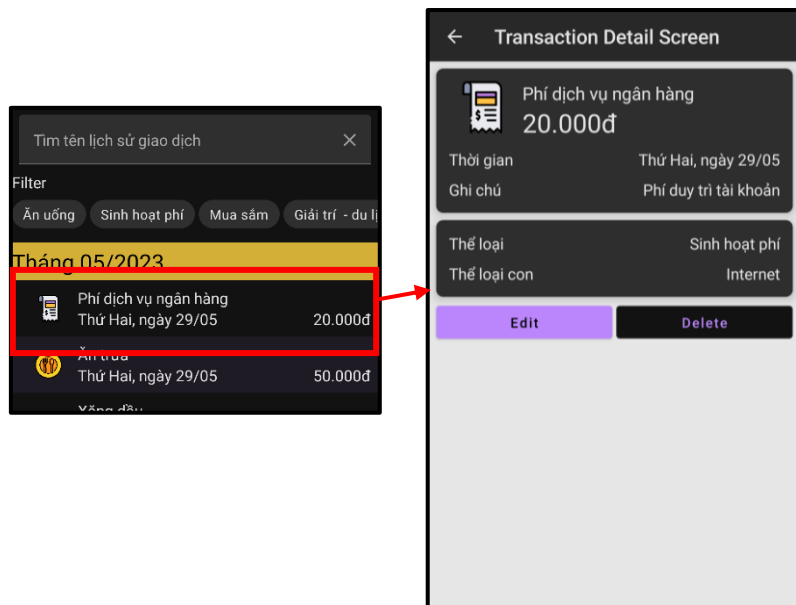
Save

Hình 3.10: Nút Save hiện lên khi các yêu cầu input tối thiểu được đáp ứng

3.3 Màn hình xem chi tiết lịch sử giao dịch

Màn hình này được kích hoạt khi người dùng chọn một lịch sử giao dịch bất kỳ trên danh sách lịch sử giao dịch ở màn hình chính

Màn hình này có nhiệm vụ hiển thị đầy đủ thông tin chi tiết của giao dịch đã được chọn. Đồng thời cung cấp 2 chức năng sửa hoặc xoá giao dịch trong CSDL



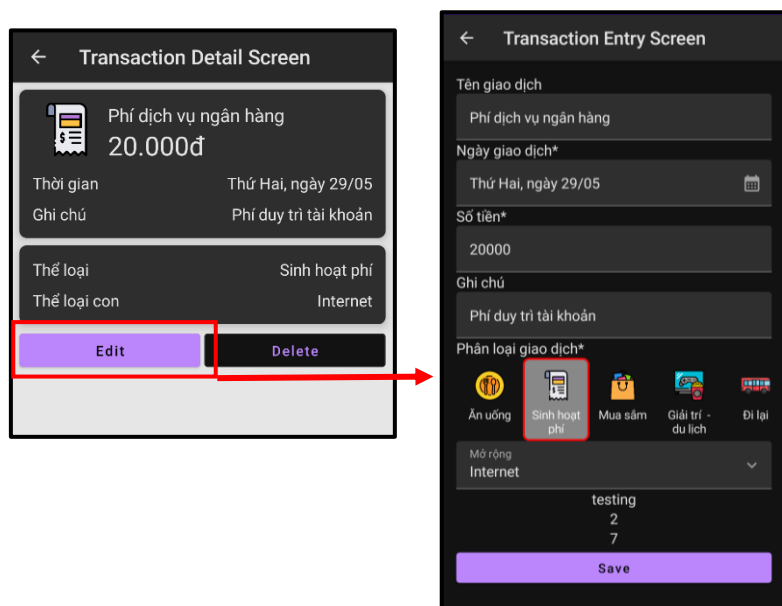
Hình 3.11: Giao diện chi tiết giao dịch

3.4 Màn hình chỉnh sửa lịch sử giao dịch

Màn hình này được gọi khi người dùng chọn nút “Edit” trên màn hình xem chi tiết lịch sử giao dịch.

Màn hình này kế thừa toàn bộ giao diện và chức năng của màn hình thêm lịch sử giao dịch. Tuy nhiên chức năng chính đó là thay đổi giao dịch đã được chọn ở trong CSDL.

Dữ liệu của giao dịch được điền trước tạo các trường nhập liệu và danh sách chọn.



Hình 3.12: Giao diện chỉnh sửa giao dịch

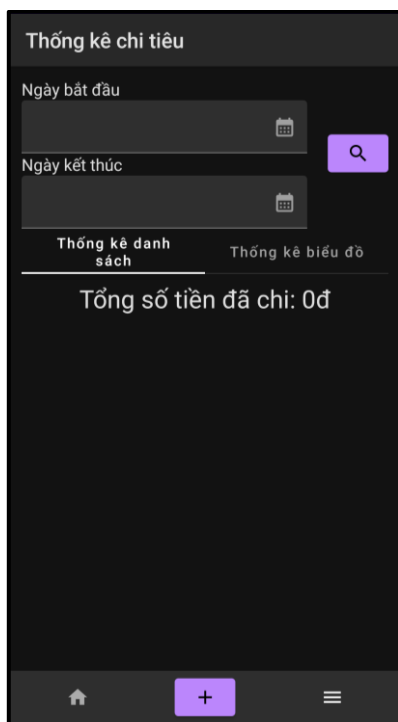
3.5 Màn hình thống kê lịch sử giao dịch

Màn hình này được gọi khi người dùng bấm biểu tượng “≡” ở thanh điều hướng bên dưới.

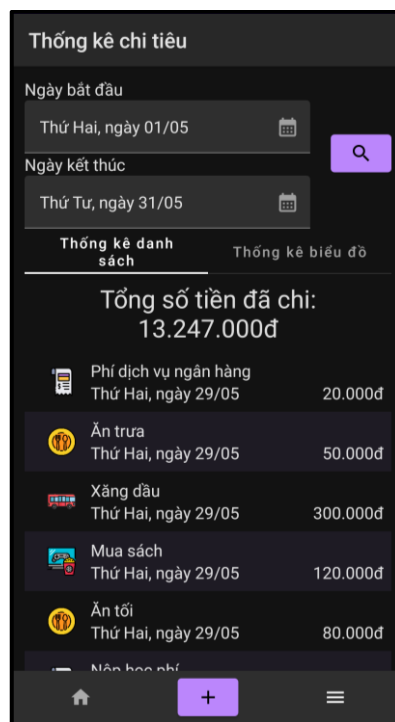


Màn hình này gồm:

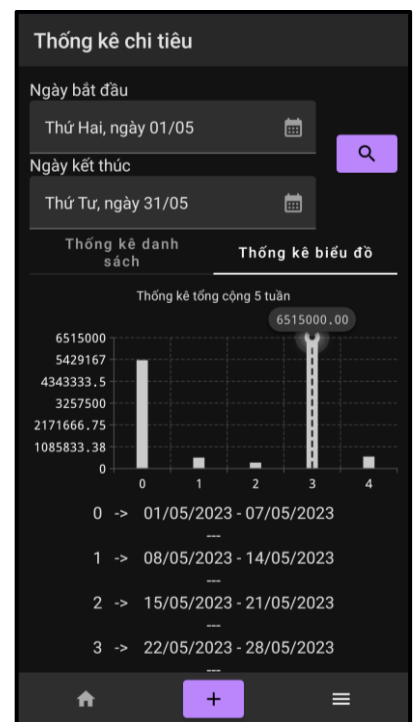
- Hai trường nhập liệu kiểu ngày tương ứng với ngày bắt đầu và kết thúc của các giao dịch mà người dùng muốn thống kê.
- Một nút kích hoạt tính năng tìm kiếm danh sách các lịch sử giao dịch có ngày giao dịch thỏa mãn nằm trong [ngày bắt đầu; ngày kết thúc].
- Hai thanh tab:
 - + Thống kê danh sách: hiển thị danh sách các lịch sử giao dịch đã được lọc theo ngày
 - + Thống kê biểu đồ: hiển thị một biểu đồ mô hình hoá các lịch sử giao dịch theo tuần tính từ ngày bắt đầu đến tuần cuối của ngày kết thúc. Trục x gồm các tuần và trục y mô tả tổng tiền đã tiêu trong mỗi tuần của trục x.



Hình 3.13: Giao diện thống kê mặc định



Hình 3.14: Khi người dùng nhập đủ ngày bắt đầu/kết thúc và chọn nút tìm kiếm (tab thống kê danh sách)



Hình 3.15: Khi người dùng nhập đủ ngày bắt đầu/kết thúc và chọn nút tìm kiếm (tab thống kê biểu đồ)

- Ứng dụng chạy với các thao tác khá mượt trên mẫu điện thoại LG V35 ThinQ. Trong quá trình thử nghiệm không xuất hiện tình trạng crash và hầu như không có khung ảnh bị khựng

KẾT LUẬN

Trong quá trình thực hiện bài báo cáo, tôi đã phát triển một ứng dụng quản lý chi tiêu trên hệ điều hành Android sử dụng các công nghệ và thư viện hiện đại như Android Studio, Jetpack Compose, Room và thư viện vẽ biểu đồ bên thứ ba Vico. Ứng dụng này có thể thêm, sửa, xóa và đọc dữ liệu, thống kê và tìm kiếm giao dịch theo tên và thể loại.

Trong quá trình phát triển, tôi đã áp dụng mô hình MVVM và tuân thủ kiến trúc Android, phân chia rõ ràng thành UI Layer và Data Layer. Điều này giúp tăng tính bảo trì, mở rộng và quản lý mã nguồn dễ dàng hơn.

Tuy nhiên, tôi nhận thấy ứng dụng còn một số nhược điểm. Giao diện vẫn chưa thực sự thân thiện với người dùng, và trong một số trường hợp, đã còn sử dụng hardcoded. Ngoài ra, các chức năng của ứng dụng còn có thể mở rộng thêm và thiếu một số tính năng quan trọng để trở thành một ứng dụng Android hiện đại, ví dụ như việc lưu trữ CSDL trên internet, hiển thị thông báo nhắc nhở,...

Trong tương lai, tôi sẽ tiếp tục phát triển ứng dụng, tập trung vào việc cải thiện giao diện để đáp ứng nhu cầu của người dùng và tăng tính tương tác của ứng dụng. Tôi cũng sẽ tối ưu hóa mã nguồn để loại bỏ hardcoded và tăng tính linh hoạt của ứng dụng. Đồng thời, tôi sẽ xem xét việc bổ sung các tính năng mới và khả năng lưu trữ CSDL trên internet, đặt lịch nhắc nhở ghi lại giao dịch,... để tạo ra một ứng dụng hoàn thiện giúp người dùng có thể quản lý chi tiêu dễ dàng và tiết kiệm thời gian.

Tổng kết lại, qua quá trình thực hiện bài báo cáo cho bộ môn thực tập cơ sở, tôi đã có cơ hội tìm hiểu kỹ hơn về lập trình Android và áp dụng các công nghệ mới nhất trong việc phát triển ứng dụng. Dù vẫn còn nhiều cải tiến cần được thực hiện, tôi đã xây dựng một cơ sở vững chắc để phát triển ứng dụng này thành một ứng dụng quản lý chi tiêu hoàn chỉnh và đáng tin cậy trong tương lai.

TÀI LIỆU THAM KHẢO

- [1] "Android SDK," Wikimedia Foundation, Inc, 1 Tháng 11 2022. [Online]. Available: https://en.wikipedia.org/wiki/Android_SDK#cite_note-2.
- [2] "Kotlin Docs | Kotlin Documentation," JetBrains, [Online]. Available: <https://kotlinlang.org/docs/home.html>.
- [3] "Application Fundamentals," Google Developers, [Online]. Available: <https://developer.android.com/guide/components/fundamentals#Components>.
- [4] "The activity lifecycle," Google Developers, [Online]. Available: <https://developer.android.com/guide/components/activities/activity-lifecycle>.
- [5] "Android Jetpack," Google Developers, [Online]. Available: <https://developer.android.com/jetpack>.
- [6] "Thinking in Compose | Android Developers," Google Developers, [Online]. Available: <https://developer.android.com/jetpack/compose/mental-model>.
- [7] "Lifecycle of composables | Compose | Android Developers," Google Developers, [Online]. Available: <https://developer.android.com/jetpack/compose/lifecycle>.
- [8] "SQLite," Wikimedia Foundation, Inc, 8 Tháng 3 2023. [Online]. Available: <https://en.wikipedia.org/wiki/SQLite>.
- [9] "Save data in a local database using Room | Android Developers," Google Developers, [Online]. Available: <https://developer.android.com/training/data-storage/room>.
- [10] "Guide to app architecture | Android Developers," Google Developers, [Online]. Available: <https://developer.android.com/topic/architecture>.
- [11] A. Chugh, "Android MVVM Design Pattern," DigitalOcean, 3 Tháng 8 2022. [Online]. Available: <https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern>.
- [12] "ViewModel overview | Android Developers," Google Developers, [Online]. Available: <https://developer.android.com/topic/libraries/architecture/viewmodel>.
- [13] P. & Patrick, "Vico," Patryk & Patrick, [Online]. Available: <https://github.com/patrykandpatrick/vico>.