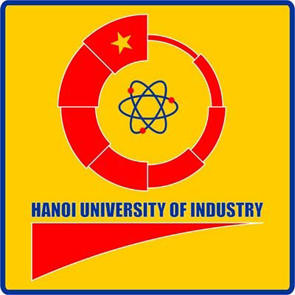


**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

****

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG DI ĐỘNG**

**QUẢN LÝ CHI TIÊU SỬ DỤNG FLUTTER**

**Giảng viên hướng dẫn: TS. Nguyễn Thị Mỹ Bình**

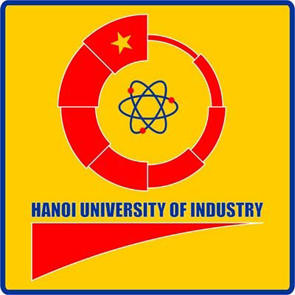
**Sinh viên: Trương Thị Thu Phương**

**Mã sinh viên: 2019605502**

**Hà Nội - Năm 20****23**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

****

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**TÊN ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG DI ĐỘNG**

**QUẢN LÝ CHI TIÊU SỬ DỤNG FLUTTER**

**Giảng viên hướng dẫn: TS. Nguyễn Thị Mỹ Bình**

**Sinh viên: Trương Thị Thu Phương**

**Mã sinh viên: 2019605502**

**Hà Nội - Năm 2023**

MỤC LỤC

[DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT i](#_Toc135163436)

[DANH MỤC HÌNH ẢNH ii](#_Toc135163437)

[DANH MỤC BẢNG BIỂU iii](#_Toc135163438)

[LỜI CẢM ƠN iv](#_Toc135163439)

[MỞ ĐẦU 1](#_Toc135163440)

[CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU 4](#_Toc135163441)

[1.1. Giới thiệu chung 4](#_Toc135163442)

[1.2. Cơ sở lý thuyết 4](#_Toc135163443)

[1.2.1. Tổng quan về lập trình di động 4](#_Toc135163444)

[1.2.2. Giới thiệu công cụ lập trình Visual Studio Code 7](#_Toc135163445)

[1.2.3. Ngôn ngữ lập trình Dart 8](#_Toc135163446)

[1.2.4. Một số framework trong lập trình Flutter 9](#_Toc135163447)

[1.2.5. SQLite Database 10](#_Toc135163448)

[1.2.6. Tổng quan về mô hình MVC 12](#_Toc135163449)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 15](#_Toc135163450)

[2.1. Khảo sát người dùng 15](#_Toc135163451)

[2.2. Giới thiệu về hệ thống 16](#_Toc135163452)

[2.3. Phân tích yêu cầu 17](#_Toc135163453)

[2.3.1. Về hệ thống 17](#_Toc135163454)

[2.3.2. Về người sử dụng 17](#_Toc135163455)

[2.3.3. Yêu cầu về chức năng 17](#_Toc135163456)

[2.4. Thiết kế hệ thống 18](#_Toc135163457)

[2.4.1. Mô hình hóa Usecase 18](#_Toc135163458)

[2.4.2. Mô tả chi tiết các Usecase 21](#_Toc135163459)

[2.4.3. Biểu đồ lớp hệ thống 39](#_Toc135163460)

[2.5. Thiết kế giao diện 40](#_Toc135163461)

[2.5.1. Giao diện Tổng quan 40](#_Toc135163462)

[2.5.2. Giao diện Quản lý giao dịch 41](#_Toc135163463)

[2.5.3. Giao diện Quản lý sự kiện 43](#_Toc135163464)

[2.5.4. Giao diện Quản lý nguồn tiền 44](#_Toc135163465)

[2.5.5. Giao diện Xem thông báo 45](#_Toc135163466)

[2.6. Thiết kế cơ sở dữ liệu 46](#_Toc135163467)

[CHƯƠNG 3. KẾT QUẢ 47](#_Toc135163468)

[3.1. Cài đặt môi trường 47](#_Toc135163469)

[3.2. Cài đặt chương trình 48](#_Toc135163470)

[3.3. Kết quả thu được 52](#_Toc135163471)

[3.3.1. Quản lý loại giao dịch 52](#_Toc135163472)

[3.3.2. Quản lý ví tiền 53](#_Toc135163473)

[3.3.3. Thống kê 53](#_Toc135163474)

[3.3.4. Quản lý sự kiện 53](#_Toc135163475)

[3.3.5. Thông báo 53](#_Toc135163476)

[3.4. Kết quả kiểm thử 54](#_Toc135163477)

[3.4.1. Kỹ thuật kiểm thử hộp đen (Black box testing) 54](#_Toc135163478)

[3.4.2. Kiểm thử chức năng 55](#_Toc135163479)

[KẾT LUẬN 62](#_Toc135163480)

[TÀI LIỆU THAM KHẢO 64](#_Toc135163481)

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Viết tắt** | **Dịch nghĩa** |
| 1 | HTML/CSS | HyperText Markup Language/Cascading Style Sheets |
| 2 | SQL | Structured Query Language |
| 3 | XML | Extensible Markup Language |
| 4 | RDBMS | Hệ quản trị cơ sở dữ liệu quan hệ |
| 5 | MVC | Model-View-Controller |
| 6 | CSDL | Cơ sở dữ liệu |
| 7 | SDK | Software Development Kit |

DANH MỤC HÌNH ẢNH

[Hình 1.1: Mô hình MVC 12](#_Toc135163196)

[Hình 2.1: Biểu đồ Usecase chính 18](#_Toc135163197)

[Hình 2.2: Phân rã Usecase Xem Thông tin tổng quan 19](#_Toc135163198)

[Hình 2.3: Phân rã Usecase Quản lý giao dịch 20](#_Toc135163199)

[Hình 2.4: Phân rã Usecase Quản lý sự kiện 20](#_Toc135163200)

[Hình 2.5: Phân rã Usecase Quản lý nguồn tiền 21](#_Toc135163201)

[Hình 2.6: Phân rã Usecase Xem thông báo 21](#_Toc135163202)

[Hình 2.7: Biểu đồ trình tự Xem thông tin tổng quan 23](#_Toc135163203)

[Hình 2.8: Mô tả Usecase Quản lý giao dịch 27](#_Toc135163204)

[Hình 2.9: Biểu đồ trình tự Usecase Quản lý giao dịch 28](#_Toc135163205)

[Hình 2.10: Biểu đồ trình tự Usecase Quản lý sự kiện 31](#_Toc135163206)

[Hình 2.11: Biểu đồ trình tự Usecase Quản lý sự kiện 32](#_Toc135163207)

[Hình 2.12: Biểu đồ trình tự Usecase Quản lý nguồn tiền 36](#_Toc135163208)

[Hình 2.13: Biểu đồ trình tự Xem thông báo 38](#_Toc135163209)

[Hình 2.14: Biểu đồ lớp hệ thống 39](#_Toc135163210)

[Hình 2.15: Giao diện tổng quan 40](#_Toc135163211)

[Hình 2.16: Giao diện Quản lý giao dịch 42](#_Toc135163212)

[Hình 2.17: Giao dịch quản lý sự kiện 43](#_Toc135163213)

[Hình 2.18: Giao diện quản lý nguồn tiền 44](#_Toc135163214)

[Hình 2.19: Giao diện Xem thông báo 45](#_Toc135163215)

[Hình 3.1: Visual Studio Code 47](#_Toc135163216)

[Hình 3.2: Cài đặt Flutter 48](#_Toc135163217)

[Hình 3.3: Mô tả cách cài đặt chương trình 49](#_Toc135163218)

[Hình 3.4: Mô tả cách cài đặt chương trình 49](#_Toc135163219)

[Hình 3.5: Mô tả cách cài đặt chương trình 50](#_Toc135163220)

[Hình 3.6: Mô tả cách cài đặt chương trình 51](#_Toc135163221)

[Hình 3.7: Mô tả cách cài đặt chương trình 52](#_Toc135163222)

[Hình 3.8: Mô tả cách cài đặt chương trình 52](#_Toc135163223)

DANH MỤC BẢNG BIỂU

[Bảng 2.1: Mô tả Usecase Xem thông tin tổng quan 21](#_Toc134540465)

[Bảng 2.2: Mô tả Usecase Quản lý giao dịch 23](#_Toc134540466)

[Bảng 2.3: Mô tả Usecase Quản lý sự kiện 29](#_Toc134540467)

[Bảng 2.4: Mô tả Usecase Quản lý nguồn tiền 33](#_Toc134540468)

[Bảng 2.5: Mô tả Usecase Xem thông báo 37](#_Toc134540469)

[Bảng 2.6: Thiết kế cơ sở dữ liệu 46](#_Toc134540470)

[Bảng 3.1: Bảng kiểm thử chức năng 54](#_Toc134540471)

LỜI CẢM ƠN

Trong suốt thời gian học tập và thực hiện đồ án tốt nghiệp em luôn nhận được sự hướng dẫn nhiệt tình từ phía nhà trường và các thầy cô giáo. Vì thế, lời đầu tiên em xin chân thành cảm ơn trường Đại học Công nghiệp Hà Nội, thầy cô giáo khoa Công nghệ thông tin đã tạo điều kiện thuận lợi và truyền dạy kiến thức cho em trong thời gian qua để em hoàn thành đề tài một cách tốt nhất. Và hơn hết em xin bày tỏ lòng biết ơn sâu sắc nhất tới cô giáo hướng dẫn **TS. Nguyễn Thị Mỹ Bình** đã tận tình giúp đỡ, định hướng em rất nhiều trong suốt quá trình tìm hiểu nghiên cứu và hoàn thành đồ án tốt nghiệp.

Khi thực hiện nghiên cứu đề tài, em đã hoàn thành tốt nhất báo cáo đồ án tốt nghiệp. Em rất mong nhận được sự đóng góp của tất cả các thầy cô giáo để đồ án của em được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

MỞ ĐẦU

1. **Lý do chọn đề tài**

Việc quản lý chi tiêu là một vấn đề quan trọng và cần thiết trong cuộc sống hàng ngày của mỗi người. Tuy nhiên, việc quản lý chi tiêu hiệu quả lại không phải là điều dễ dàng đối với nhiều người, đặc biệt là những người mới bắt đầu sống độc lập hoặc có thu nhập thấp.

Với sự phát triển của công nghệ thông tin, ứng dụng quản lý chi tiêu đã trở thành một công cụ hữu ích và tiện lợi cho người dùng. Flutter là một nền tảng phát triển ứng dụng di động được phát triển bởi Google, cho phép lập trình viên phát triển ứng dụng đa nền tảng với một lần viết code.

Chính vì vậy, em quyết định chọn đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" để tìm hiểu sâu hơn về việc phát triển ứng dụng di động đa nền tảng bằng Flutter cùng với việc giải quyết một vấn đề thực tế trong cuộc sống. Ngoài ra, đây là một đề tài phù hợp với khả năng và sở thích của em trong lĩnh vực lập trình di động.

1. **Đối tượng, mục tiêu và phạm vi nghiên cứu**
2. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" bao gồm:

* Nghiên cứu và phân tích các nhu cầu và yêu cầu của người dùng về ứng dụng quản lý chi tiêu.
* Tìm hiểu và áp dụng công nghệ xây dựng ứng dụng di động đa nền tảng Flutter để phát triển một ứng dụng quản lý chi tiêu đáp ứng các yêu cầu đặt ra.
* Xây dựng và thiết kế giao diện cho ứng dụng quản lý chi tiêu bằng Flutter với mục đích đảm bảo tính cơ động và dễ sử dụng cho người dùng.
* Đảm bảo tính hiệu quả của ứng dụng thông qua việc kiểm thử và đánh giá, đồng thời đảm bảo tính ổn định và bảo mật thông tin cá nhân của người dùng trong quá trình sử dụng ứng dụng.
* Tìm hiểu và áp dụng các công nghệ mới và tiên tiến trong quá trình phát triển ứng dụng để nâng cao tính năng và trải nghiệm của người dùng.

1. Đối tượng nghiên cứu

* Ngôn ngữ lập trình Dart để phát triển ứng dụng trên Flutter, cùng với các công cụ hỗ trợ như Visual Studio Code
* Mô hình MVC (Model-View-Controller) được sử dụng để phát triển ứng dụng, kèm theo đó là sử dụng các cơ sở dữ liệu như Firebase và SQLite để lưu trữ thông tin và dữ liệu cho ứng dụng.
* Quy trình phát triển phần mềm
* Các vấn đề liên quan đến quản lý chi tiêu

1. Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" bao gồm:

* Phát triển một ứng dụng di động đa nền tảng bằng Flutter để hỗ trợ người dùng quản lý chi tiêu hàng ngày.
* Các tính năng của ứng dụng bao gồm quản lý chi tiêu, quản lý thu nhập, thống kê chi tiêu và thu nhập theo thời gian và loại, quản lý tài khoản ngân hàng, tạo và quản lý các khoản vay, tạo và quản lý các mục tiêu tài chính, báo cáo tổng quan về tình hình tài chính.
* Phát triển giao diện người dùng thân thiện và dễ sử dụng.
* Tối ưu hiệu suất ứng dụng và tính đa nền tảng bằng Flutter.

1. **Phương pháp nghiên cứu**

* Phương pháp nghiên cứu về mặt lý thuyết:
* Tiến hành nghiên cứu và thu thập các tài liệu liên quan đến đề tài, bao gồm các tài liệu về quản lý chi tiêu, ứng dụng di động, Flutter, và các công nghệ liên quan.
* Tổng hợp và phân tích các tài liệu thu thập được.
* Chọn lọc và sử dụng các tài liệu nghiên cứu để báo cáo đồ án tốt nghiệp.
* Phương pháp nghiên cứu trong thực nghiệm:
* Tìm hiểu các tiêu chí kiểm thử chức năng và cách thức thực hiện.
* Phân tích các ứng dụng có chức năng tương tự để tìm hiểu các tiêu chuẩn thiết kế và triển khai ứng dụng.
* Tiến hành xây dựng ứng dụng theo phân tích và yêu cầu thực tế.
* Tìm hiểu và sử dụng các công nghệ liên quan đến Flutter như Firebase để xây dựng tính năng đăng nhập và đồng bộ dữ liệu.
* Tổng hợp kiến thức đã học và hoàn thành báo cáo đồ án tốt nghiệp.

# TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU

## Giới thiệu chung

Ứng dụng quản lý chi tiêu là một ứng dụng rất hữu ích trong cuộc sống hiện đại, giúp người dùng có thể quản lý tài chính cá nhân một cách khoa học và hiệu quả. Flutter là một nền tảng phát triển ứng dụng di động rất phổ biến, được sử dụng để xây dựng các ứng dụng trên nhiều nền tảng khác nhau.

Việc sử dụng Flutter để phát triển ứng dụng quản lý chi tiêu có nhiều ưu điểm. Đầu tiên, Flutter cung cấp khả năng phát triển ứng dụng nhanh chóng và hiệu quả, giúp giảm thời gian và chi phí phát triển. Ngoài ra, Flutter còn cung cấp thư viện widget đa dạng và dễ sử dụng, giúp tạo ra giao diện ứng dụng đẹp và thân thiện với người dùng.

Với ứng dụng quản lý chi tiêu bằng Flutter, người dùng có thể dễ dàng ghi lại các khoản chi tiêu của mình, theo dõi tình hình thu chi và tạo kế hoạch chi tiêu. Ngoài ra, ứng dụng còn cung cấp nhiều tính năng hữu ích khác như báo cáo tài chính, nhắc nhở thanh toán hóa đơn, quản lý tài khoản ngân hàng và nhiều tính năng khác.

Trong tổng quan, việc sử dụng Flutter để phát triển ứng dụng quản lý chi tiêu là một giải pháp hiệu quả và tiện lợi. Với các tính năng và tiện ích của nó, ứng dụng này giúp người dùng quản lý tài chính cá nhân một cách dễ dàng và hiệu quả hơn.

## Cơ sở lý thuyết

### Tổng quan về lập trình di động

Hiện nay có 3 hướng chính xây dựng và phát triển 1 ứng dụng di động là: Native app, Web App và Hybrid app. Mỗi hướng đều có ưu và nhược điểm và kỹ năng riêng.

#### Hybrid App

Hybrid App là ứng dụng kết hợp của cả Mobile Web App và Native App. Theo đó, các phần cơ bản của ứng dụng vẫn viết bằng ngôn ngữ web, nhưng được đặt trong native container, nên vẫn có thể đưa lên các App Store.

- Ưu điểm:

+ Chỉ cần có kiến thức về HTML, CSS, JavaScript …

+ Viết một lần dùng được nhiều nơi.

+ Tận dụng được các chức năng của hệ thống.

+ Có thể chạy được ở chế độ offline

- Nhược điểm:

+ Hiệu năng chậm

+ Không ổn định do khó debug. Framework sẽ dịch code thành code native sửa lỗi khá khó khăn không biết được dịch như thế nào

#### Native App

Native App là những ứng dụng được viết riêng cho một loại nền tảng như iOS, Android, Windows Phone bằng các ngôn ngữ tương ứng của mọi nền tảng đó ví dụ Java trên Android, Object C trên iOS.

Mỗi Native App chỉ chạy được trên một nền tảng và không thể mang sang các nền tảng khác

- Ưu điểm:

+ Tận dụng được tính năng có sẵn trên thiết bị như: GPS, Camera, thiết bị thu âm ...

+ Hiệu năng cao vì code native chạy trực tiếp trên máy

+ Có thế chạy được ở chế độ offline

- Nhược điểm:

+ Không thể kết hợp nhiều nền tảng. Mỗi một ứng dụng chỉ chạy trên 1 nền tảng nhất định

+ Mỗi hệ điều hành cân phải viết ứng dụng riêng khó đồng bộ giữa các ứng dụng

+ Việc bảo trì hay nâng cấp sẽ làm mất nhiều thời gian. Do phải sửa chữa từng app trên từng hệ điều hành

+ Xây dựng ứng dụng cần dùng các phần mềm riêng biệt theo hệ điều hành (dùng Xcode trên Mac để phát triển ứng dụng IOS, Android Studio để phát triển ứng dụng Android)

#### Web App

Web app được phát triển khi đã có sẵn một website đang hoạt động. Ta tạo thêm một trang web riêng cho các thiết bị di động sử dụng HTML, CSS và một số thư viện khác hỗ trợ.

Web app được thiết kế chạy trên nền tảng web hoặc các trình duyệt của thiết bị di động cho phép người dùng thao tác như thao tác trang web giống nhu thao tác ứng dụng.

- Ưu điểm:

+ Có thể chạy trên tất cả trình duyệt của mobile hỗ trợ phiên bản HTML và javascript

+ Không cần cài đặt trên máy miễn là máy có trình duyệt web

+ Một phiên bản duy nhất cho tất cả, nên giảm chi phí và thời gian cho phát triển, bảo trì, cũng như nâng cấp sau này.

+ Phiên bản được cập nhật liên tục không cần phải cập nhật trên chợ

- Nhược điểm:

+ Hiệu năng không được tốt như native app và luôn phải chạy online.

+ Không thể dùng được các tính năng tích hợp của di động: Notification, chụp hình, nghiêng máy, định vị GPS, các sensor…

+ Với một số máy đời cũ, Web app sẽ bị vỡ giao diện, hiển thị sai, hoặc javascript không chạy.

### Giới thiệu công cụ lập trình Visual Studio Code

Visual Studio Code (hay gọi tắt là VSCode) là một trình biên tập mã nguồn mở, đa nền tảng và rất phổ biến, được phát triển bởi Microsoft. VSCode hỗ trợ nhiều ngôn ngữ lập trình và cung cấp nhiều tính năng hữu ích giúp cho việc phát triển phần mềm trở nên dễ dàng hơn.

Dưới đây là một số tính năng và công cụ hỗ trợ của VSCode:

- Đa nền tảng: VSCode có sẵn cho Windows, macOS và Linux, giúp cho các nhà phát triển có thể sử dụng trình biên tập này trên nhiều hệ điều hành khác nhau.

- Hỗ trợ nhiều ngôn ngữ lập trình: VSCode hỗ trợ nhiều ngôn ngữ lập trình như C#, C++, Java, Python, JavaScript, TypeScript, v.v.

- Công cụ IntelliSense: VSCode có tính năng IntelliSense giúp cho việc gợi ý mã và tự động hoàn thành mã trở nên dễ dàng hơn. IntelliSense cũng giúp cho việc phát hiện lỗi và sửa lỗi nhanh chóng.

- Hỗ trợ Git: VSCode tích hợp sẵn với Git, giúp cho việc quản lý mã nguồn và hợp nhất mã trở nên dễ dàng hơn.

- Thư viện mở rộng (Extensions): VSCode có thể được mở rộng bằng cách cài đặt các extension, giúp cho việc phát triển phần mềm trở nên linh hoạt hơn. Các extension này bao gồm các công cụ hỗ trợ cho các ngôn ngữ lập trình, quản lý dự án, kiểm thử, v.v.

- Tích hợp bộ nhớ đệm (Workspace): VSCode cho phép lưu trữ các tệp tin và dữ liệu cấu hình trong bộ nhớ đệm (Workspace) để có thể truy cập lại sau này.

Tóm lại, VSCode là một công cụ lập trình đa nền tảng và rất phổ biến, với nhiều tính năng hữu ích giúp cho việc phát triển phần mềm trở nên dễ dàng hơn.

### Ngôn ngữ lập trình Dart

Dart là một ngôn ngữ lập trình được phát triển bởi Google vào năm 2011, đặc biệt được sử dụng để phát triển ứng dụng di động và web. Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn. Dart hỗ trợ kiểu tĩnh và kiểu động, cú pháp đơn giản và dễ đọc, hỗ trợ lập trình hướng đối tượng, hàm bậc cao, và bộ thu gom rác tự động.

Dart cũng hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web. Flutter, một framework phát triển ứng dụng di động đa nền tảng, cũng được phát triển bởi Google và sử dụng Dart làm ngôn ngữ lập trình chính. Flutter hỗ trợ phát triển ứng dụng di động với tốc độ và hiệu suất cao, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.

* Ưu điểm:

- Tính năng hiện đại: Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn, bao gồm hỗ trợ kiểu tĩnh và kiểu động, hàm bậc cao, và bộ thu gom rác tự động.

- Đơn giản và dễ đọc: Cú pháp của Dart đơn giản và dễ đọc, giúp cho các nhà phát triển dễ dàng hiểu và sử dụng ngôn ngữ này.

- Lập trình hướng đối tượng: Dart hỗ trợ lập trình hướng đối tượng, cho phép phát triển ứng dụng với cấu trúc logic rõ ràng và dễ bảo trì.

- Viết code đa nền tảng: Dart hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web.

- Flutter: Flutter, một framework phát triển ứng dụng di động đa nền tảng của Google, được sử dụng Dart làm ngôn ngữ lập trình chính. Flutter hỗ trợ phát triển ứng dụng di động với tốc độ và hiệu suất cao, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.

Với những ưu điểm trên, Dart đang trở thành một ngôn ngữ lập trình ngày càng phổ biến và được nhiều nhà phát triển tin tưởng sử dụng cho các dự án phát triển ứng dụng di động và web.

### Một số framework trong lập trình Flutter

#### Flutter SDK

Flutter SDK là một SDK hỗ trợ phát triển ứng dụng di động đa nền tảng sử dụng Flutter, cung cấp các tính năng như widget, animation, routing, và hỗ trợ viết ứng dụng đa nền tảng cho cả Android và iOS.

#### AngularDart

AngularDart là một framework phát triển ứng dụng web, sử dụng Dart làm ngôn ngữ lập trình chính. AngularDart cung cấp các tính năng như mô hình MVC, dependency injection, routing, và có thể tích hợp với các thư viện khác để phát triển ứng dụng web phong phú.

#### StageXL

StageXL là một framework phát triển trò chơi web, sử dụng Dart làm ngôn ngữ lập trình chính. StageXL cung cấp các tính năng như sprite, animation, sound, và hỗ trợ viết trò chơi web đa nền tảng.

### SQLite Database

SQLite là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) nhỏ gọn, được thiết kế để tích hợp vào các ứng dụng. Nó là một thư viện mã nguồn mở, được viết bằng ngôn ngữ C, và cung cấp các tính năng quản lý cơ sở dữ liệu như các hệ quản trị cơ sở dữ liệu lớn hơn như MySQL, PostgreSQL hay Oracle.

SQLite được phát triển bởi D. Richard Hipp vào năm 2000 và đã trở thành một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới. SQLite được tích hợp sẵn trên hầu hết các hệ điều hành phổ biến như Windows, macOS và Linux, và cũng được hỗ trợ trên các nền tảng di động như Android và iOS.

* Cấu trúc của SQLite

SQLite là một thư viện mã nguồn mở và được viết bằng ngôn ngữ C. Nó bao gồm một bộ máy truy vấn SQL, một bộ máy quản lý tệp và các thành phần bổ sung khác. Các đối tượng trong SQLite bao gồm bảng, cột, khóa chính và các ràng buộc.

* Bảng: Là một đối tượng trong SQLite để lưu trữ dữ liệu. Bảng bao gồm các cột và hàng, trong đó mỗi cột đại diện cho một kiểu dữ liệu và mỗi hàng đại diện cho một bản ghi.
* Cột: Là một đối tượng trong SQLite để đại diện cho một kiểu dữ liệu trong bảng. Các kiểu dữ liệu có thể là INTEGER, REAL, TEXT hoặc BLOB.
* Khóa chính: Là một cột hoặc tập hợp các cột trong bảng, được sử dụng để định danh duy nhất cho mỗi hàng trong bảng.
* Ràng buộc: Là các quy tắc được áp dụng trên bảng để đảm bảo tính toàn vẹn của dữ liệu. Ví dụ, ràng buộc NOT NULL sẽ đảm bảo rằng giá trị của cột không được phép là null.
* Cách thức hoạt động của SQLite

SQLite hoạt động như một thư viện được tích hợp vào các ứng dụng. Khi một ứng dụng sử dụng SQLite, nó sẽ tạo ra một tệp tin cơ sở dữ liệu, trong đó sẽ lưu trữ các bảng, cột và các bản ghi. Ứng dụng có thể truy vấn và thay đổi dữ liệu trong cơ sở dữ liệu này bằng cách sử dụng các lệnh SQL.

SQLite hỗ trợ các lệnh SQL như SELECT, INSERT, UPDATE và DELETE để truy vấn và thay đổi dữ liệu trong cơ sở dữ liệu. Nó cũng hỗ trợ các câu lệnh để tạo bảng, chỉnh sửa bảng và xóa bảng. SQLite cũng có thể thực hiện các hoạt động như tạo chỉ mục, ràng buộc và trigger.

Trong ứng dụng của chúng ta, chúng ta sẽ sử dụng các package hỗ trợ của Flutter như sqflite và path\_provider để tương tác với cơ sở dữ liệu SQLite. Sqflite là một package hỗ trợ cho phép tương tác với cơ sở dữ liệu SQLite trong Flutter, trong khi path\_provider là một package cho phép truy cập vào các thư mục và tệp tin trong hệ thống tệp của thiết bị.

Với việc sử dụng cơ sở dữ liệu SQLite, chúng ta có thể lưu trữ thông tin và dữ liệu của người dùng một cách hiệu quả và tiện lợi. Ví dụ, chúng ta có thể lưu trữ thông tin về các khoản chi tiêu của người dùng và tạo báo cáo thống kê chi tiêu theo tháng, quý, năm, v.v. từ các dữ liệu được lưu trữ trong cơ sở dữ liệu SQLite.

* Ưu điểm của SQLite Database:
* Kích thước nhỏ gọn: SQLite có dung lượng nhỏ gọn và khả năng tích hợp cao, giúp dễ dàng tích hợp vào các ứng dụng.
* Không cần server: SQLite không yêu cầu một server đặc biệt để chạy, vì vậy nó rất tiện lợi cho các ứng dụng đơn giản.
* Tốc độ truy vấn nhanh: SQLite có tốc độ truy vấn nhanh và khả năng xử lý dữ liệu tốt.
* Hỗ trợ nhiều ngôn ngữ lập trình: SQLite được hỗ trợ bởi nhiều ngôn ngữ lập trình như C/C++, Python, Java, PHP, v.v.

### Tổng quan về mô hình MVC

Mô hình MVC (Model-View-Controller) là một mô hình thiết kế phần mềm được sử dụng để phát triển các ứng dụng có giao diện người dùng. Mô hình này bao gồm ba thành phần chính: Model, View và Controller.



Hình 1.1: Mô hình MVC

* *Model*

Model là thành phần chịu trách nhiệm xử lý dữ liệu. Nó đại diện cho các đối tượng dữ liệu và quy trình xử lý dữ liệu. Model không biết gì về giao diện người dùng, nó chỉ tập trung vào việc xử lý dữ liệu.

* *View*

View là thành phần chịu trách nhiệm hiển thị dữ liệu và tương tác với người dùng. View đại diện cho giao diện người dùng, bao gồm các thành phần như các nút bấm, trường nhập liệu và các thành phần khác. View không biết gì về dữ liệu và quy trình xử lý, nó chỉ tập trung vào việc hiển thị và tương tác với người dùng.

* *Controller*

Controller là thành phần chịu trách nhiệm điều khiển quá trình tương tác giữa Model và View. Controller đóng vai trò là trung gian giữa Model và View, nó nhận lệnh từ View, xử lý dữ liệu từ Model và cập nhật lại View để hiển thị kết quả cho người dùng. Controller là thành phần duy nhất có thể tương tác với cả Model và View.

Mô hình MVC giúp tách biệt logic xử lý dữ liệu và giao diện người dùng, giúp cho việc phát triển và bảo trì ứng dụng trở nên dễ dàng hơn. Nó cũng giúp cho việc thay đổi giao diện người dùng hoặc cơ sở dữ liệu không ảnh hưởng đến logic xử lý dữ liệu và ngược lại.

* *Một ứng dụng thực tế có thể được thiết kế theo mô hình MVC*

- Model: Chứa các đối tượng dữ liệu như thông tin người dùng, thông tin sản phẩm, danh sách đơn hàng, v.v. Nó cũng chứa các phương thức để truy xuất và xử lý dữ liệu.

- View: Chứa các màn hình giao diện người dùng để hiển thị thông tin và tương tác với người dùng, ví dụ như màn hình đăng nhập, màn hình danh sách sản phẩm, màn hình đặt hàng, v.v.

- Controller: Chứa các phương thức để điều khiển quá trình tương tác giữa Model và View, ví dụ như phương thức để xử lý đăng nhập, phương thức để lấy danh sách sản phẩm, phương thức để đặt hàng, v.v.

Với việc sử dụng mô hình MVC, việc phát triển và bảo trì ứng dụng sẽ trở nên dễ dàng hơn và có thể được thực hiện bởi các nhóm phát triển khác nhau độc lập với nhau.

***\* Ưu điểm***

+ Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view

+ MVC sẽ tạo sự tương tác hiệu quả giữa designer và developer

+ Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều

+ Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…

***\* Nhược điểm***

+ Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML

+ Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần

+ Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel

+ Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Khảo sát người dùng

* Câu 1: Bạn gặp khó khăn hay vấn đề gì trong việc quản lý chi tiêu?

Trả lời:

* + Bạn đang lập ngân sách mà không có mục tiêu rõ ràng
  + Bạn quên lập ngân sách cho các khoản chi tiêu không cố định
  + Thiếu kiểm soát trong chi tiêu thường nhật
* Câu 2: Bạn có thường dễ quên và bỏ lỡ việc ghi lại chi tiêu những thứ nhỏ không?

Trả lời:

* + Có
* Câu 3: Bạn thường dùng cách nào để quản lý chi tiêu khoa học?

Trả lời:

* + Sổ ghi chép, app hoặc tự nhớ
  + Dùng google sheet
  + Ghi lại trên ứng dụng Note
* Câu 4: Bạn đã dùng ứng dụng nhắc nhở nào?
  + Google sheet
  + Chưa dùng app nào
  + Dùng các ứng dụng sẵn của điện thoại
* Câu 5: Bạn hay gặp khó khăn gì khi sử dụng ứng dụng quản lý chi tiêu?
* Giao diện hơi khó dùng
* Chưa phù hợp với bản thân nên khó trong việc sử dụng
* Câu 6: Bạn thích nhất chức năng gì trong một ứng dụng quản lý chi tiêu?
* Thống kê tổng chi tiêu trong tháng
* Có thể thay đổi linh hoạt, nhiều chức năng để tích hợp với từng hoàn cảnh, người dùng khác nhau
* Thông báo trước một khoảng thời gian trước khi sự kiện đã lập ngân sách diễn ra
* Giao diện đẹp, trực quan.

## Giới thiệu về hệ thống

Đồ án tốt nghiệp của bạn về đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" là một hệ thống ứng dụng di động được phát triển bằng Flutter, cho phép người dùng quản lý chi tiêu cá nhân một cách dễ dàng và tiện lợi.

Hệ thống sẽ cho phép người dùng nhập thông tin chi tiêu hàng ngày, theo các danh mục khác nhau như thực phẩm, điện thoại, xăng xe, giải trí, và nhiều hơn nữa. Người dùng có thể xem chi tiêu của mình theo thời gian, theo danh mục, và theo các tiêu chí khác như số tiền đã chi, số tiền còn lại, và tổng số tiền chi tiêu trong một khoảng thời gian nhất định.

Hệ thống cũng sẽ cung cấp tính năng đặt mục tiêu tiết kiệm cho người dùng, giúp họ đặt mục tiêu số tiền muốn tiết kiệm cho một khoảng thời gian nhất định, và theo dõi tiến độ đạt được mục tiêu.

Ngoài ra, hệ thống cũng cung cấp tính năng nhắc nhở cho người dùng về các khoản chi tiêu cần phải thanh toán trong thời gian sắp tới, giúp họ không quên các khoản chi tiêu quan trọng.

Hệ thống sẽ được xây dựng trên nền tảng Flutter, một framework phổ biến cho việc phát triển ứng dụng di động đa nền tảng. Hệ thống sẽ được thiết kế với giao diện đơn giản, dễ sử dụng và tối ưu hóa hiệu suất để đáp ứng nhu cầu của người dùng.

## Phân tích yêu cầu

### Về hệ thống

Ứng dụng Quản lý chi tiêu hướng tới một hệ thống có giao diện dễ nhìn, dễ sử dụng, dễ quản lý và đem lại sự trải nghiệm tuyệt vời nhất cho người sử dụng. Bên cạnh đó, ứng dụng cũng hướng tới sự tối ưu hiệu suất, tiết kiệm thời gian cho người sử dụng.

### Về người sử dụng

Người dùng nắm bắt được các giao dịch chi tiêu hằng ngày. Ngoài ra có thể xem các danh sách thông tin các ví tiền, các sự kiện sử dụng tiền, lọc các thông tin theo nhiều cách.

### Yêu cầu về chức năng

* Các chức năng chính của hệ thống:
* Xem thông tin tổng quan
* Quản lý giao dịch
* Quản lý sự kiện
* Quản lý nguồn tiền
* Xem thông báo
* Yêu cầu phi chức năng:
* Giao diện đồ họa dễ hình, dễ thao tác
* Hoạt động của hệ thống ổn định
* Độ tin cậy của ứng dụng
* Hiệu năng của ứng dụng

## Thiết kế hệ thống

### Mô hình hóa Usecase

#### Xác định các tác nhân

* Người dùng

#### Xác định các Usecase

* Xem thông tin tổng quan
* Quản lý giao dịch
* Quản lý sự kiện
* Quản lý nguồn tiền
* Xem thông báo

#### Biểu đồ Usecase

***Biểu đồ Usecase chính***

******

Hình 2.1: Biểu đồ Usecase chính

***Biểu đồ Usecase thứ cấp***

*Phân rã Usecase <Xem Thông tin tổng quan>*

******

Hình 2.2: Phân rã Usecase Xem Thông tin tổng quan

*Phân rã Usecase <Quản lý Giao Dịch>*

******

Hình 2.3: Phân rã Usecase Quản lý giao dịch

*Phân rã Usecase <Quản Lý Sự kiện>*

******

Hình 2.4: Phân rã Usecase Quản lý sự kiện

*Phân rã Usecase <Quản lý Nguồn Tiền>*

******

Hình 2.5: Phân rã Usecase Quản lý nguồn tiền

*Phân rã Usecase <Xem Thông Báo>*

******

Hình 2.6: Phân rã Usecase Xem thông báo

### Mô tả chi tiết các Usecase

#### Mô tả Usecase <Xem thông tin tổng quan>

Bảng 2.1: Mô tả Usecase Xem thông tin tổng quan

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng xem thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu trong năm, các giao dịch gần đây. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Usecase bắt đầu khi người dùng mở ứng dụng hoặc bấm vào nút “Tổng quan” trên màn hình.   * Khi người dùng mở ứng dụng hoặc bấm vào nút “Tổng quan” trên màn hình, hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây. * Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



Hình 2.7: Biểu đồ trình tự Xem thông tin tổng quan

#### Mô tả Usecase <Quản lý giao dịch>

Bảng 2.2: Mô tả Usecase Quản lý giao dịch

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng quản lý và xem thông tin các giao dịch. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Usecase bắt đầu khi người dùng bấm vào “Lịch sử”. Khi đó trên màn hình chính của ứng dụng hiển thị lên màn hình quản lý giao dịch  Xem danh sách giao dịch   * Khi người dùng bấm vào “Lịch sử” trên màn hình, hệ thống hiển thị danh sách giao dịch trong tháng này. * Xem danh sách giao dịch tháng trước: Khi người dùng bấm vào “Tháng trước” hệ thống hiển thị các giao dịch trong tháng trước. * Xem danh sách giao dịch tháng này: Khi người dùng bấm vào “Tháng này” hệ thống hiển thị các giao dịch trong tháng này. * Xem danh sách giao dịch tương lai: Khi người dùng bấm vào “Tương lai” hệ thống hiển thị các giao dịch trong tương lai. * Xem danh sách giao dịch trong khoảng thời gian: Khi người dùng bấm vào nút “Lọc”, hệ thống hiển thị form, người dùng nhập vào ngày đầu và ngày cuối rồi nhấn áp dụng, hệ thống hiển thị tất cả giao dịch trong khoảng thời gian người dùng vừa nhập. * Xem chi tiết giao dịch: Khi người dùng bấm vào một giao dịch bất kỳ, hệ thống hiển thị thông tin chi tiết giao dịch đó.   Thêm giao dịch   * Người dùng bấm vào nút thêm giao dịch (Dấu cộng màu xanh lam) trên màn hình, hệ thống hiển thị lên màn hình thêm giao dịch. * Người dùng nhập thông tin và chọn “Lưu” để hoàn thành. * Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu   Sửa giao dịch   * Trên màn hình “Chi tiết giao dịch”, người dùng nhập thông tin cần thay đổi và nhấn “Sửa”. * Hệ thống cập nhật vào cơ sở dữ liệu.   Xóa giao dịch   * Trong màn hình hiển thị các giao dịch, kéo giao dịch sang trái sẽ hiển thị nút “Xóa”. * Bấm “Xóa” để thực hiện xóa giao dịch. * Hệ thống cập nhật thông tin vào cơ sở dữ liệu.   Tìm kiếm giao dịch   * Trên màn hình quản lý giao dịch, người dùng nhấn vào thanh tìm kiếm và nhập tên loại giao dịch hoặc ghi chú của giao dịch cần tìm. * Hệ thống hiển thị màn hình danh sách chứa các giao dịch chứa các từ khóa tìm kiếm của người dùng đã nhập. * Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | * Luồng 1.1: Trong Usecase xem danh sách giao dịch trong khoảng thời gian, khi đang điền form, nếu người dùng nhấn nút “Bỏ lọc”, hệ thống quay về màn hình “Quản lý giao dịch” * Luồng 1.2: Trong Usecase xem danh sách giao dịch trong khoảng thời gian, khi đang điền form, nếu người dùng nhấn nút “Xóa”, hệ thống xóa toàn bộ giá trị ngày bắt đầu và ngày kết thúc mà Người Dùng đã nhập. * Luồng 2.1: Người dùng nhập thiếu thông tin giao dịch: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc. * Luồng 3.1: Người dùng nhập thiếu thông tin giao dịch để chỉnh sửa: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc. |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



Hình 2.8: Mô tả Usecase Quản lý giao dịch



Hình 2.9: Biểu đồ trình tự Usecase Quản lý giao dịch

#### Mô tả Usecase <Quản lý sự kiện>

Bảng 2.3: Mô tả Usecase Quản lý sự kiện

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng quản lý và xem thông tin các sự kiện. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Usecase bắt đầu khi người dùng bấm vào nút “Sự kiện” trên màn hình  Thêm sự kiện   * Người dùng bấm vào nút thêm sự kiện (Dấu cộng màu xanh lục) trên màn hình “Quản lý sự kiện”, hệ thống hiển thị lên màn hình thêm sự kiện. * Người dùng nhập thông tin vào chọn “Lưu” để hoàn thành. * Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu   Xem danh sách sự kiện   * Khi người dùng bấm vào nút “Sự kiện” trên màn hình, hệ thống hiển thị danh sách các sự kiện.   Xem chi tiết sự kiện   * Ở màn hình danh sách sự kiện, khi người dùng bấm vào một sự kiện, hệ thống hiển thị màn hình “Chi tiết sự kiện” bao gồm tên sự kiện, số tiền cho phép của sự kiện, thời gian diễn ra, số tiền đã chi tiêu, danh sách các giao dịch trong sự kiện.   Sửa sự kiện   * Trên màn hình “Chi tiết sự kiện”, người dùng nhập thông tin cần thay đổi và nhấn “Sửa”. * Hệ thống kiểm tra lại thông tin và cập nhật vào cơ sở dữ liệu. * Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | * Luồng 1.1: Người dùng nhập thiếu thông tin sự kiện: Hệ thống không cho phép lưu và hiển thị thông báo các trường bắt buộc. |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



Hình 2.10: Biểu đồ trình tự Usecase Quản lý sự kiện



Hình 2.11: Biểu đồ trình tự Usecase Quản lý sự kiện

#### Mô tả Usecase <Quản lý nguồn tiền>

Bảng 2.4: Mô tả Usecase Quản lý nguồn tiền

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng quản lý và xem thông tin các nguồn tiền. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Usecase bắt đầu khi người dùng bấm vào “Xem tất cả” trong mục “Ví của tôi” trên màn hình tổng quan.  Xem danh sách các nguồn tiền   * Khi người dùng bấm vào “Xem tất cả” trong mục “Ví của tôi” trên màn hình tổng quan, hệ thống hiển thị danh sách các nguồn tiền.   Thêm nguồn tiền  - Người dùng bấm vào nút thêm giao dịch (Dấu cộng) trên màn hình “Quản lý nguồn tiền”, hệ thống hiển thị lên màn hình thêm nguồn tiền.  - Người dùng nhập thông tin vào chọn “Lưu” để hoàn thành.  - Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu  Xem danh sách giao dịch sử dụng nguồn tiền   * Trên màn hình hiển thị danh sách các nguồn tiền, khi người dùng bấm vào một nguồn tiền, hệ thống hiển thị ra các giao dịch sử dụng nguồn tiền đó. * Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



Hình 2.12: Biểu đồ trình tự Usecase Quản lý nguồn tiền

#### Mô tả Usecase <Xem thông báo>

Bảng 2.5: Mô tả Usecase Xem thông báo

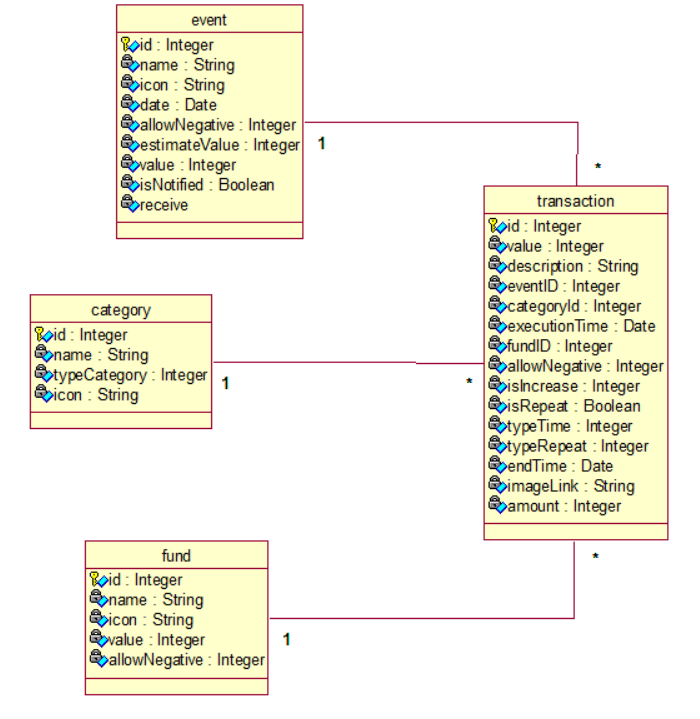
|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase cho phép người dùng theo dõi các thông báo về các giao dịch. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Usecase bắt đầu khi người dùng bấm vào nút “Thông báo” trên màn hình   * Sau khi người dùng bấm nút “Thông báo”, hệ thống hiển thị danh sách thông báo. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



Hình 2.13: Biểu đồ trình tự Xem thông báo

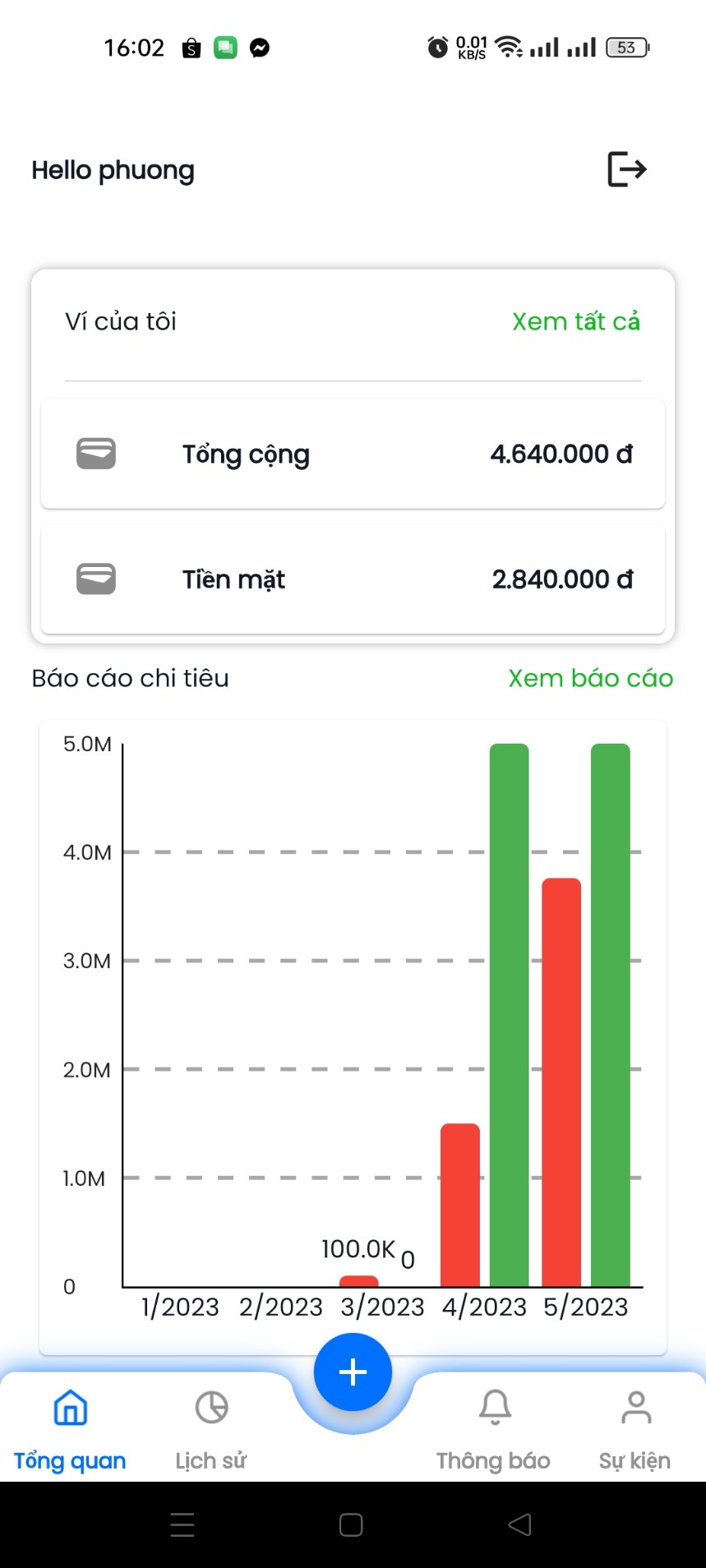
### Biểu đồ lớp hệ thống



Hình 2.14: Biểu đồ lớp hệ thống

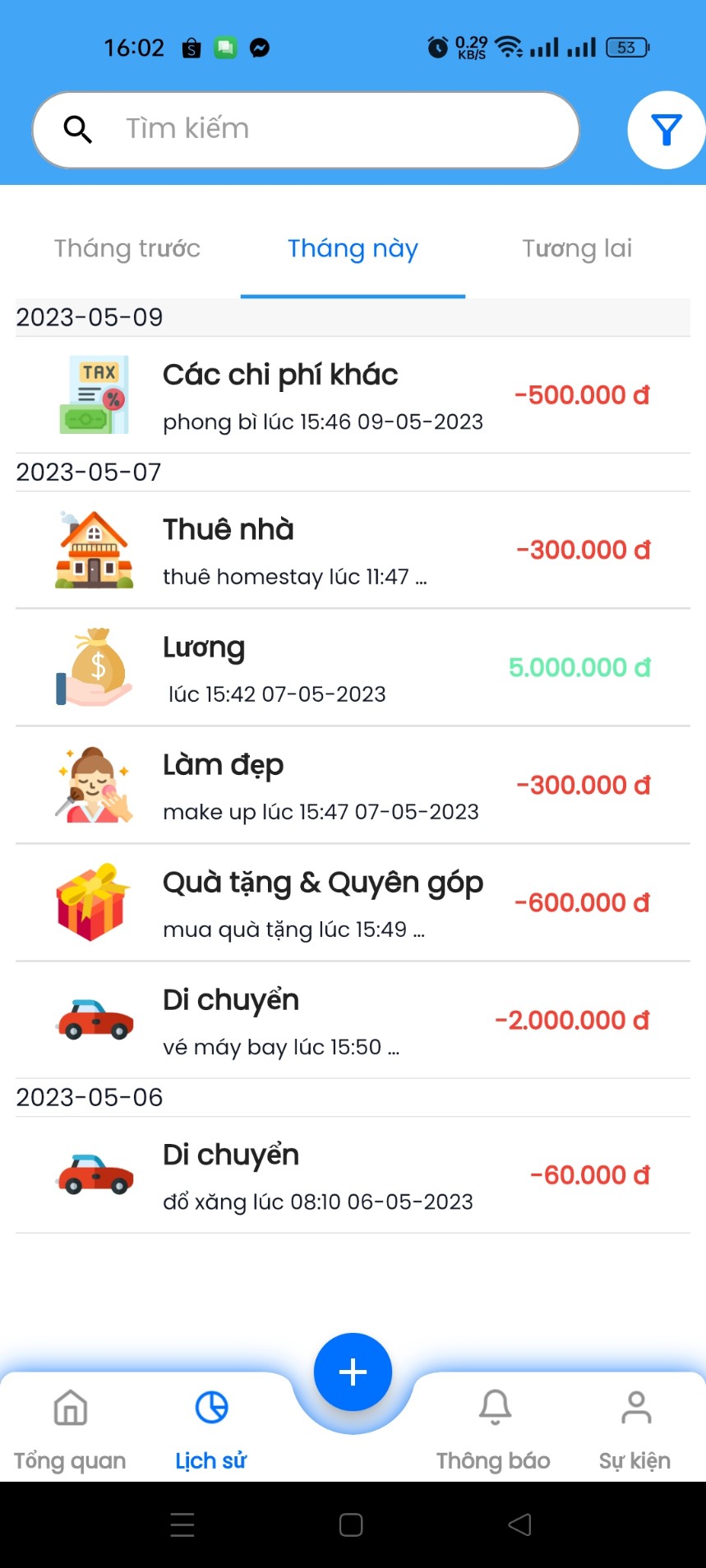
## Thiết kế giao diện

### Giao diện Tổng quan



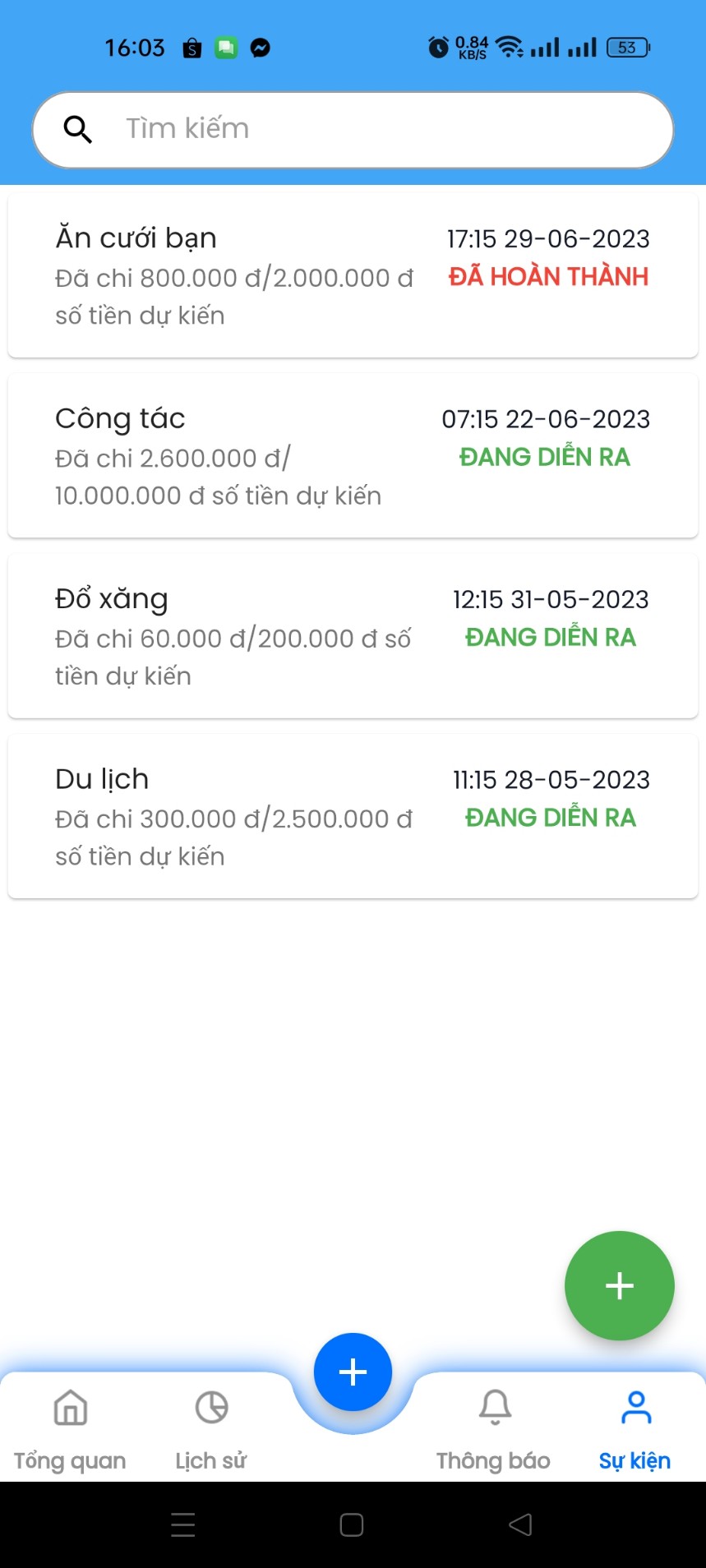
Hình 2.15: Giao diện tổng quan

### Giao diện Quản lý giao dịch



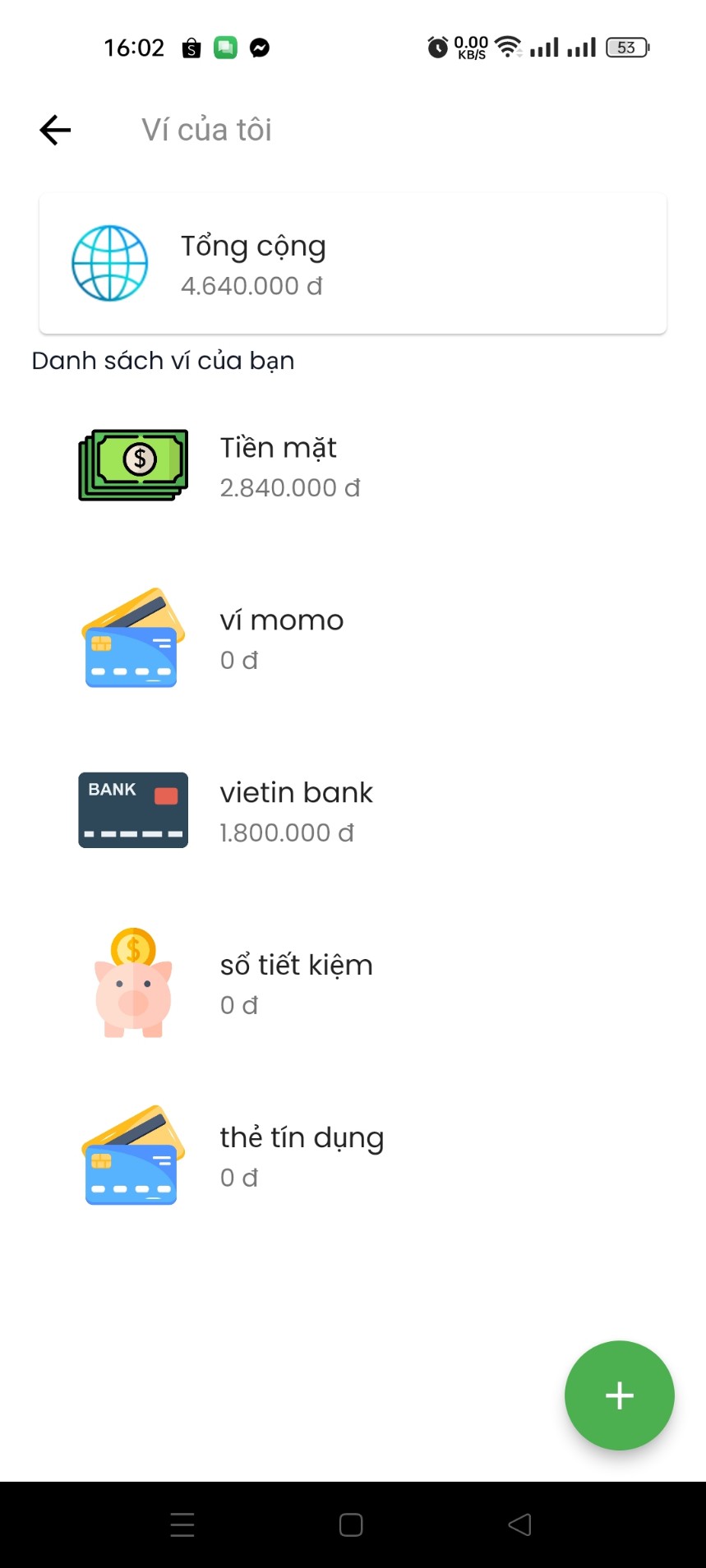
Hình 2.16: Giao diện Quản lý giao dịch

### Giao diện Quản lý sự kiện



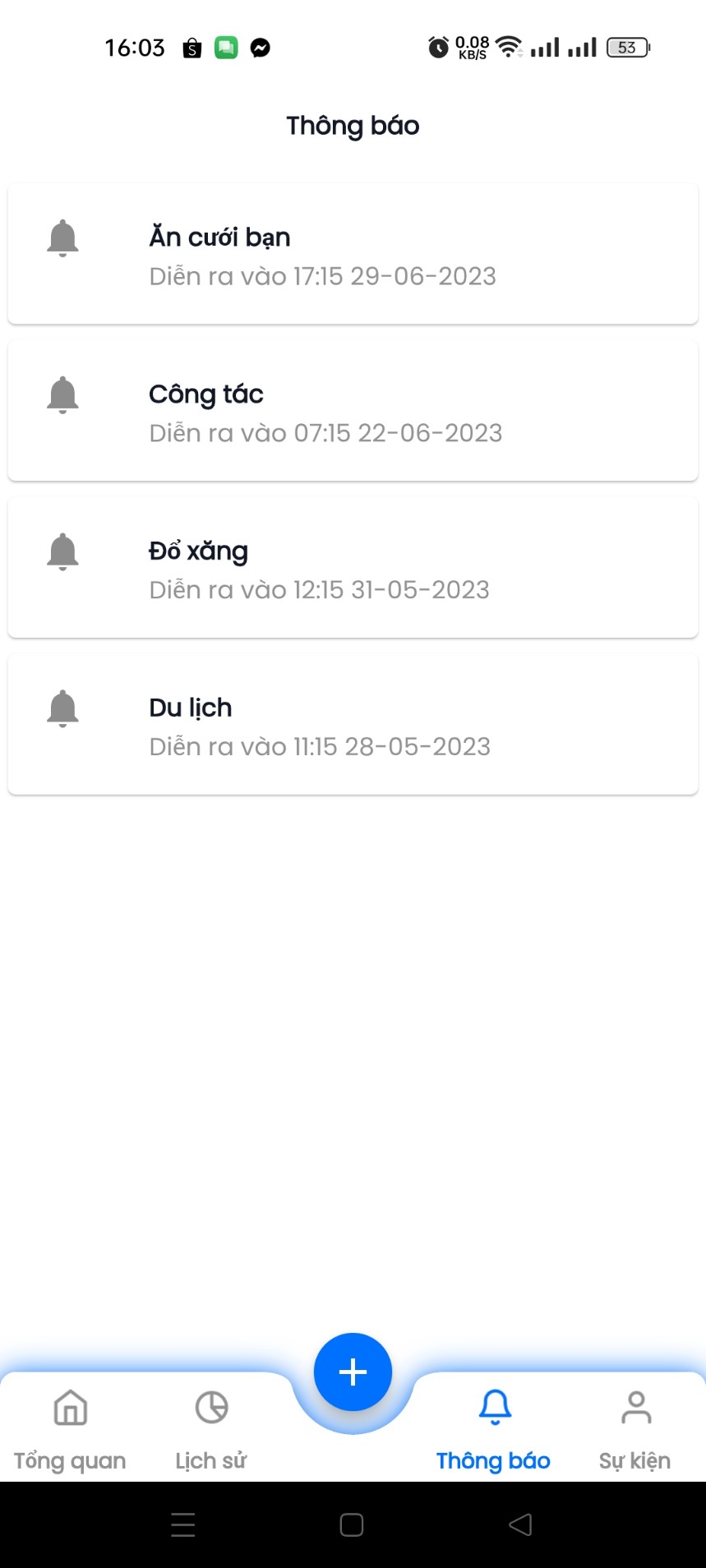
Hình 2.17: Giao dịch quản lý sự kiện

### Giao diện Quản lý nguồn tiền



Hình 2.18: Giao diện quản lý nguồn tiền

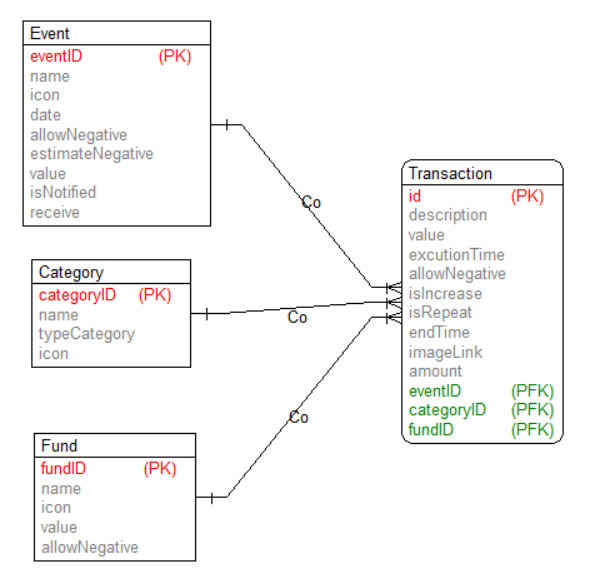
### Giao diện Xem thông báo



Hình 2.19: Giao diện Xem thông báo

## Thiết kế cơ sở dữ liệu

Bảng 2.6: Thiết kế cơ sở dữ liệu



# KẾT QUẢ

## Cài đặt môi trường

* Cài đặt VS Code
* Bước 1: Truy cập vào link <https://code.visualstudio.com/> để tải xuống phiên bản VS Code
* Bước 2: : Ấn nút download để thực hiện tải và cài đặt phiên bản mới nhất

A screenshot of a computer

Description automatically generated

Hình 3.1: Visual Studio Code

* Cài đặt Flutter và Dart plugin
  + Mở VS Code.
  + Chọn View > Command Palette….
  + Nhấn “install”, sau đó chọn Extensions: Install Extensions.
  + Tìm kiếm “flutter” trong khung tìm kiếm extensions, chọn Flutter trong danh sách, sau đó nhấn “install”.

A screenshot of a computer

Description automatically generated

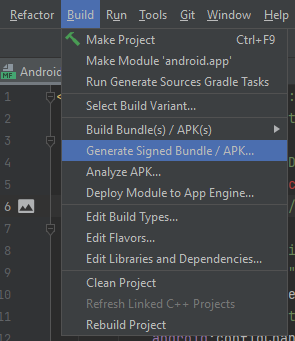
Hình 3.2: Cài đặt Flutter

* Xác thực cài đặt bằng Flutter doctor
* Chọn View > Command Palette….
* Nhập “doctor” và chọn “Flutter: Run Flutter Doctor”
* Xem lại đầu ra trong khung OUTPUT để biết bất kỳ sự cố nào.

## Cài đặt chương trình

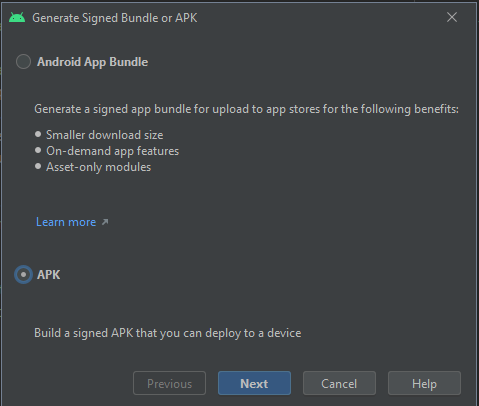
**Bước 1:** Tạo file keystore

* + - Tại menu Build / chọn Generate Signed APK…



Hình 3.3: Mô tả cách cài đặt chương trình

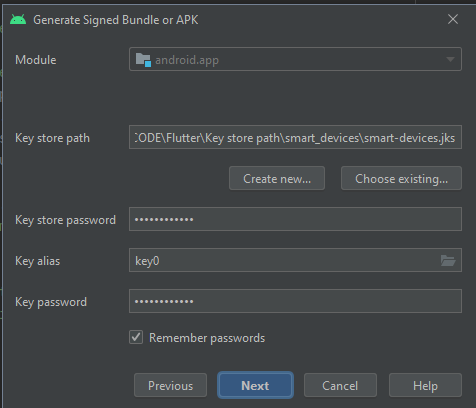
* + - Lựa chọn APK tại cửa sổ Generate Signed Bundle or APK và chọn Next



Hình 3.4: Mô tả cách cài đặt chương trình

**Bước 2:** Nhúng keystore vào file apk

* + - Cửa sổ hiện lên chọn **Create new…** để tạo **Key Store**



Hình 3.5: Mô tả cách cài đặt chương trình

* Chọn đường dẫn tại mục Key store path và điền các thông tin cần thiết

A screenshot of a computer

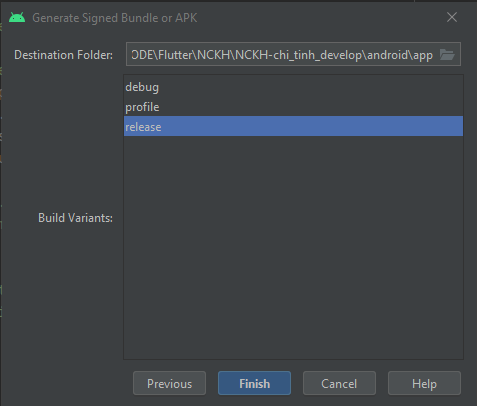
Description automatically generated

Hình 3.6: Mô tả cách cài đặt chương trình

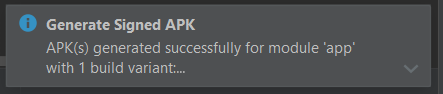
* Click “**OK**” để hoàn thành tạo Keystore, sau đó chọn “**Next**”.

**Bước 3:** Chuyển file apk từ debug sang release

* + - Tiếp tục chọn **release** để và **Finish** để Generate file .apk của app



Hình 3.7: Mô tả cách cài đặt chương trình



Hình 3.8: Mô tả cách cài đặt chương trình

Quá trình Generate hoàn thành và tạo được file cài đặt trên hệ điều hành Android.

## Kết quả thu được

Hệ thống được xây dựng với những chức năng, đó là:

### Quản lý loại giao dịch

* Phân loại có 2 loại giao dịch

+ Giao dịch lặp lại (Lặp theo ngày/tuần/tháng, sẽ tự sinh ra giao dịch theo từng loại lặp lại)

+ Giao dịch không lặp lại

* Thêm, sửa, xóa giao dịch :

+ Xác thực các trường dữ liệu như số tiền, danh mục, thời gian, nguồn tiền trước khi thêm/sửa

+ Xác nhận xóa/sửa

* Tìm kiếm theo tên, mô tả
* Lọc theo thời gian

### Quản lý ví tiền

* Thêm mới ví tiền : Xác thực tên ví trước khi thêm

### Thống kê

* Thống kê giao dịch, tổng thu, tổng chi theo từng ngày, theo ví tiền, theo sự kiện
* Thống kê chi tiêu theo tất cả các tháng, lọc theo từng ví
* Tổng hợp 5 giao dịch gần nhất

### Quản lý sự kiện

* Thêm mới, sửa , đánh dấu hoàn thành hoặc chưa hoàn thành sự kiện

+ Xác thực tên, số tiền trước khi thêm/sửa

+ Xác nhận sửa, cập nhật trạng thái hoàn thành, chưa hoàn thành

### Thông báo

* Thông báo trực tiếp những về thông tin sự kiện đã diễn ra trong khi dùng ứng dụng
* Thông báo từ hệ thống trước đúng một ngày diễn ra sự kiện đối với sự kiện diễn ra sau thời điểm thêm một ngày và thông báo ngay sau đó một phút với sự kiện diễn ra trước thời điểm sau thời điểm hiện tại một ngày

## Kết quả kiểm thử

### Kỹ thuật kiểm thử hộp đen (Black box testing)

Kiểm thử hộp đen (Black box testing) là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó. Mục đích chính của kiểm thử hộp đen chỉ là để xem phần mềm có hoạt động như dự kiến trong tài liệu yêu cầu và liệu nó có đáp ứng được sự mong đợi của người dùng hay không.

* Các loại kiểm thử hộp đen
* Kiểm thử chức năng (Functional Testing): Là loại kiểm thử có liên quan tới chức năng của toàn bộ hệ thống bên trong sản phẩm, phần mềm. Trong quá trình kiểm thử với Functional Testing chức năng sẽ được kiểm tra kỹ càng trong luồng dữ liệu đầu vào, sau đó đánh giá kết quả đầu ra mà không lo bị ảnh hưởng tới cấu trúc bên trong ứng dụng.
* Kiểm thử phi năng (Non - Functional Testing): Là loại kiểm thử nhằm mục đích kiểm thử các vấn đề về hiệu suất, độ tin cậy và mức độ có thể sử dụng trong 1 sản phẩm phần mềm, ứng dụng bất kỳ.
* Kiểm thử hồi quy (Regression testing): Là kiểm thử hồi quy thường được sử dụng khi đã xác định được lỗi, đã nâng cấp và bảo trì hệ thống thành công.
* Kiểm thử hộp đen có 8 kỹ thuật phổ biến và có nhiều ưu điểm nhất:
* Phân vùng tương đương (equivalence partitioning)
* Phân tích giá trị biên (boundary value analysis)
* Bảng quyết định (decision tables)
* Kiểm thử chuyển đổi trạng thái (state transition testing).
* Kỹ thuật kiểm thử các bộ n thần kỳ (Pairwise)
* Kỹ thuật phân tích vùng miền (Domain analysis)
* Kỹ thuật dựa trên đặc tả Use case (Use case)
* Kỹ thuật dùng lược đồ quan hệ nhân quả (Cause- Effect Diagram)

### Kiểm thử chức năng

Phần mềm được thực hiện kiểm thử chức năng (Functional Testing). Kiểm thử chức năng là một loại kiểm thử hộp đen (Black box testing) và test case của nó được dựa trên đặc tả của ứng dụng phần mềm/thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị và kiểm tra kết quả đầu ra, ít quan tâm đến cấu trúc bên trong của ứng dụng.

Bảng 3.1: Bảng kiểm thử chức năng

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Chức năng | Trường hợp kiểm thử | Bước thực hiện | Dữ liệu kiểm thử | Đầu ra thực tế | Đầu ra mong muốn | Kết quả |
| TQ1 | Xem thông tin tổng quan | Xem thông tin tổng quan | 1. Bấm vào nút “Tổng quan” trên màn hình |  | Hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây | Hệ thống hiển thị các thông tin tổng quan về Nguồn tiền, báo cáo chi tiêu và các giao dịch gần đây | Pass |
| SK1 | Quản lý sự kiện | Thêm sự kiện | 1. bấm vào nút thêm sự kiện (Dấu cộng màu xanh lục) trên màn hình “Quản lý sự kiện”  2. Điền thông tin về tên sự kiện, số tiền, chọn ngày, chọn giờ  3. Nhấn vào “Lưu” bên góc phải trên của màn hình | Tên sự kiện: Ăn uống  Số tiền: 100.000đ  Ghi chú: Tiền ăn  Chọn ngày: 07/05/2023  Chọn thời gian: 07:15 | Màn hình hiển thị “Thêm mới sự kiện thành công” | Màn hình hiển thị “Thêm mới sự kiện thành công” | Pass |
| SK2 | Xem danh sách sự kiện | 1. bấm vào nút “Sự kiện” trên màn hình | Thông tin sự kiện vừa thêm ở trên | Màn hình hiển thị các sự kiện | Màn hình hiển thị các sự kiện | Pass |
| SK3 | Xem chi tiết sự kiện | 1. Ở màn hình danh sách sự kiện, khi người dùng bấm vào một sự kiện, hệ thống hiển thị màn hình “Chi tiết sự kiện” |  | Hiển thị thông tiên chi tiết của sự kiện | Hiển thị thông tiên chi tiết của sự kiện | Pass |
| SK4 | Sửa sự kiện | 1. Trên màn hình “Chi tiết sự kiện”, người dùng nhập thông tin cần thay đổi và nhấn “Sửa” |  | Màn hình hiển thị “Sửa sự kiện thành công” | Màn hình hiển thị “Sửa sự kiện thành công” | Pass |
| GD1 | Quản lý giao dịch | Thêm giao dịch | 1. Bấm vào nút thêm giao dịch (Dấu cộng màu xanh lam)  2. Điền thông tin về số tiền, nhập ghi chú, chọn danh mục, thời gian, nguồn tiền, sự kiện, hình ảnh  3. Nhấn vào “Lưu” bên góc phải trên của màn hình | Số tiền: 100.000đ  Ghi chú: Tiền ăn  Chọn danh mục: Ăn uống  Chọn thời gian: 07/05/2023 – 07:15  Chọn nguồn tiền: Tiền mặt  Chọn sự kiện: Ăn uống | Màn hình hiển thị “Thêm giao dịch thành công” | Màn hình hiển thị “Thêm giao dịch thành công” | Pass |
| GD2 | Xem giao dịch | 1. Bấm vào “Lịch sử” | Xem được thông tin giao dịch vừa thêm ở trên | Màn hình hiển thị thông tin các cuộc giao dịch trong tháng này | Màn hình hiển thị thông tin các cuộc giao dịch trong tháng này | Pass |
| GD3 | Sửa giao dịch | 1. Trên màn hình “Chi tiết giao dịch”, người dùng nhập thông tin cần thay đổi và nhấn “Sửa”.  2. Sửa thông tin  3. Trên màn hình xác bấm nhấn vào nút “Sửa” | Sửa số tiền thành: 150.000đ | Màn hình hiển thị “Sửa giao dịch thành công” | Màn hình hiển thị “Sửa giao dịch thành công” | Pass |
| GD4 | Xoá giao dịch | 1. Trong màn hình hiển thị các giao dịch, kéo giao dịch sang trái sẽ hiển thị nút “Xóa”.  2. Nhấn vào “Xoá”  3. Tại màn hình xác nhận nhấn nút “Xoá” | Xoá giao dịch vừa thêm ở trên | Màn hình hiển thị “Xoá giao dịch thành công” | Màn hình hiển thị “Xoá giao dịch thành công” | Pass |
| GD5 | Tìm kiếm giao dịch | 1. Trên màn hình quản lý giao dịch, người dùng nhấn vào thanh tìm kiếm và nhập tên loại giao dịch | Thông tin tìm kiếm: Ăn uống | Màn hình hiển thị thông tin giao dịch “Ăn uống” | Màn hình hiển thị thông tin giao dịch “Ăn uống” | Pass |
| NT1 | Quản lý nguồn tiền | Xem danh sách các nguồn tiền | 1. bấm vào “Xem tất cả” trong mục “Ví của tôi” trên màn hình tổng quan |  | Màn hình hiển thị danh sách các nguồn tiền. | Màn hình hiển thị danh sách các nguồn tiền. | Pass |
| NT2 | Thêm nguồn tiền | 1. bấm vào nút thêm giao dịch (Dấu cộng) trên màn hình “Quản lý nguồn tiền”  2. Nhập thông tin nguồn tiền  3. Bấm “Lưu” | Tên ví: Momo  Số tiền: 1.000.000đ  Chọn loại ví: Ví cơ bản | Màn hình hiển thị “Thêm mới Ví tiền thành công” | Màn hình hiển thị “Thêm mới Ví tiền thành công” | Pass |
| NT3 | Xem danh sách giao dịch sử dụng nguồn tiền | 1. Trên màn hình hiển thị danh sách các nguồn tiền, khi người dùng bấm vào một nguồn tiền |  | Màn hình hiển thị ra các giao dịch sử dụng nguồn tiền đó | Màn hình hiển thị ra các giao dịch sử dụng nguồn tiền đó | Pass |
| TB1 | Xem thông báo | Xem thông báo | 1. bấm vào nút “Thông báo” trên màn hình |  | Màn hình hiển thị danh sách thông báo. | Màn hình hiển thị danh sách thông báo. | Pass |

KẾT LUẬN

* **Đánh giá kết quả**

Dựa trên những thông tin em cung cấp, đánh giá kết quả đề tài "Ứng dụng quản lý chi tiêu bằng Flutter" như sau:

- Phần hoàn thành:

+ Đã nắm bắt và áp dụng thành công quy trình để xây dựng một ứng dụng đa nền tảng, đồng thời sử dụng thành thạo trình soạn thảo VScode.

+ Hoàn thiện bài khảo sát và phân tích thiết kế hệ thống " Ứng dụng quản lý chi tiêu bằng Flutter ".

+ Phát triển thành thạo với ngôn ngữ Dart và framework Flutter.

- Phần chưa hoàn thành:

+ Tuy nhiên, cần hoàn thiện thêm giao diện để khách hàng dễ sử dụng. Điều này rất quan trọng vì giao diện đóng vai trò quan trọng trong trải nghiệm người dùng và ảnh hưởng đến sự thành công của ứng dụng. Em có thể đưa thêm các tính năng và cải thiện trải nghiệm người dùng để tăng tính hữu dụng của ứng dụng.

Tóm lại, em đã hoàn thành một phần lớn của đề tài "Ứng dụng quản lý chi tiêu bằng Flutter", đồng thời đã có kiến thức và kinh nghiệm để phát triển các ứng dụng đa nền tảng. Tuy nhiên, để đạt được sự thành công về mặt thương mại, em cần phải hoàn thiện giao diện và trải nghiệm người dùng của ứng dụng.

* **Hướng phát triển**
* Thiết kế và lập trình mở rộng các chức năng: Đồng bộ tự động tích hợp hệ thống Back End, Tích hợp thêm quản lý nguồn tiền đến các thẻ ngân hàng và các loại ví điện tử
* Hỗ trợ thêm chế độ Dark mode.
* Hỗ trợ giao diện dạng widget cho ứng dụng trên màn hình chính của điện thoại.
* Hỗ trợ giao diện ứng dụng trên IOS, Window.
* Cải thiện sự mượt mà của ứng dụng. Nâng cao trải nghiệm người dùng.
* Em rất mong tiếp tục nhận được sự giúp đỡ và tạo điều kiện của thầy cô và nhà trường để em có cơ hội phát triển, hoàn thiện ứng dụng tốt hơn trong thời gian tới.

TÀI LIỆU THAM KHẢO

[1]. Tổ HTTT Đại học Công Nghiệp Hà Nội – Giáo trình hệ quản trị cơ sở dữ liệu SQL.

[2]. Tổ HTTT Đại học Công Nghiệp Hà Nội – Giáo trình phân tích thiết kế hệ thống.

[3]. Vũ Thị Dương, Phùng Đức Hòa, Nguyễn Thị Hương Lan (2015), “Giáo trình Phân tích thiết kế hướng đối tượng”, NXB Khoa học và Kỹ thuật.

{Thomas, 2000 #1}