**MỤC LỤC**

[PHẦN 1 MỞ ĐẦU 9](#_Toc358131824)

[PHẦN 2 GIỚI THIỆU VỀ CÔNG TY Covisof 11](#_Toc358131825)

[1 Giới thiệu sơ lược về công ty: 12](#_Toc358131826)

[2 Lịch sử hình thành công ty và phát triển của công ty: 12](#_Toc358131827)

[3 Lĩnh vực kinh doanh của công ty: 12](#_Toc358131828)

[4 Cơ cấu tổ chức nhân sự của công ty Covisof: 12](#_Toc358131829)

[4.1 Sơ đồ tổ chức công ty: 12](#_Toc358131830)

[4.2 Cơ cấu tổ chức chức năng của các bộ phận phòng ban trong công ty: 13](#_Toc358131831)

[PHẦN 3 TỔNG QUAN 15](#_Toc358131832)

[**CHƯƠNG 1: SƠ LƯỢC VỀ HỆ ĐIỀU HÀNH 16**](#_Toc358131837)

[1.1 SƠ LƯỢC VỀ CÁC HỆ ĐIỀU HÀNH 16](#_Toc358131833)

[1.2 ANDROID 17](#_Toc358131834)

[1.4. Kiến trúc hệ điều hành Android 26](#_Toc358131835)

[1.5. Các thành phần trong ứng dụng Android 30](#_Toc358131836)

[**CHƯƠNG 2: MÔ TẢ SƠ ĐỒ USECASE 59**](#_Toc358131837)

[2.1. Sơ đồ Use-case 59](#_Toc358131838)

[2.2. Danh sách các Actor 59](#_Toc358131839)

[2.3. Danh sách các Use-case 60](#_Toc358131840)

[2.4. Đặc tả Use-case 61](#_Toc358131841)

[2.4.1. Use-case Thêm khoản thu 61](#_Toc358131842)

[2.4.2. Use-case Xóa khoản thu 61](#_Toc358131843)

[2.4.3. Use-case Sửa khoản thu 62](#_Toc358131844)

[2.4.4. Use-case Thêm khoản chi 63](#_Toc358131845)

[2.4.5. Use-case Xóa khoản chi 64](#_Toc358131846)

[2.4.6. Use-case Sửa khoản chi 64](#_Toc358131847)

[2.4.7. Use-case Thêm khoản nợ 65](#_Toc358131848)

[2.4.8. Use-case Xóa khoản nợ 66](#_Toc358131849)

[2.4.9. Use-case Sửa khoản nợ 66](#_Toc358131850)

[2.4.10. Use-case Tìm kiếm 67](#_Toc358131851)

[2.4.11. Use-case Thêm loại chi 68](#_Toc358131852)

[2.4.12. Use-case Sửa loại chi 68](#_Toc358131853)

[2.4.13. Use-case Xóa loại chi 69](#_Toc358131854)

[2.4.14. Use-case Thêm loại thu 70](#_Toc358131855)

[2.4.15. Use-case Xóa loại thu 70](#_Toc358131856)

[2.4.16. Use-case Sửa loại thu 71](#_Toc358131857)

[2.4.17. Use-case Thêm khoản vay 72](#_Toc358131858)

[2.4.18. Use-case Sửa khoản vay 72](#_Toc358131859)

[2.4.19. Use-case Xóa khoản vay 73](#_Toc358131860)

[2.4.20. Use-case Thống kê 74](#_Toc358131861)

[**CHƯƠNG 3:SƠ ĐỒ PHÂN TÍCH 75**](#_Toc358131862)

[3.1. Sơ đồ lớp 75](#_Toc358131863)

[3.1.1. Sơ đồ lớp 75](#_Toc358131864)

[3.1.2. Các lớp đối tượng và quan hệ 75](#_Toc358131865)

[3.1.3. Mô tả chi tiết các lớp đối tượng 76](#_Toc358131866)

[3.2 . Sơ đồ tuần tự 78](#_Toc358131867)

[3.2.1 Thêm khoản thu 78](#_Toc358131868)

[3.2.2 Thêm khoản chi 80](#_Toc358131869)

[3.2.3 Thêm thể loại chi 80](#_Toc358131870)

[**CHƯƠNG 4: THIẾT KẾ GIAO DIỆN 82**](#_Toc358131871)

[4.1. Danh sách các màn hình 82](#_Toc358131872)

[4.2. Mô tả chi tiết mỗi màn hình 83](#_Toc358131873)

[4.2.1. Màn hình “Màn hình khởi động” 83](#_Toc358131874)

[4.2.2. Màn hình “Thêm khoản chi” 83](#_Toc358131875)

[4.2.3. Màn hình “Danh sách khoản chi” 86](#_Toc358131876)

[4.2.4. Màn hình “Thêm khoản thu**”** 88](#_Toc358131878)

[4.2.5. Màn hình “Danh sách khoản thu” 90](#_Toc358131879)

[4.2.6 Màn hình “Thể loại”**:** 92](#_Toc358131880)

[4.2.7 Màn hình “Thể loại chi”**:** 92](#_Toc358131881)

[4.2.8 Màn hình “Thể loại thu”: 95](#_Toc358131883)

[4.2.9. Màn hình “Thêm khoản vay” 98](#_Toc358131885)

[4.2.10. Màn hình “Thêm khoản nợ” 100](#_Toc358131886)

[4.2.11. Màn hình “thống kê” 104](#_Toc358131887)

[4.2.12. Màn hình “thống kê khoản thu” 105](#_Toc358131888)

[4.2.13. Màn hình “thống kê khoản chi” 106](#_Toc358131889)

[PHẦN 4 KẾT QUẢ THỰC HIỆN 107](#_Toc358131890)

[1. Môi trường phát triển và triển khai 108](#_Toc358131891)

[2. Kết quả đạt được 108](#_Toc358131892)

[3. Hướng phát triển 109](#_Toc358131893)

[TÀI LIỆU THAM KHẢO 110](#_Toc358131894)

**BẢNG VÀ CÁC HÌNH VẼ**

[Hình 1 Giao diện màn hình chính (android 4.2.2) 59](#_Toc322814604)7

[Hình 1.1 Các thành viên của liên minh di động 1](#_Toc322814605)9

[Hình 1.2 Cấu trúc Stack hệ thống android 75](#_Toc322814606)6

[Hình 1.3 Các thành phần trong một androidprokect 3](#_Toc322814607)0

[Hình 1.4 Lược đồ vòng đời của một Activity 3](#_Toc322814608)6

[Hình 1.5 Màn hình làm việc của addon SQLite browser 5](#_Toc322814609)6

[Hình 2.1 Sơ đồ Use-case 58](#_Toc322814610)

[Hình 3.1 Sơ đồ lớp 78](#_Toc322814611)

[Bảng 3.1 Sơ đồ các lớp đối tượng và quan hệ 78](#_Toc322814612)

[Bảng 3.2 Lớp khoản thu 75](#_Toc322814613)

[Bảng 3.3 Lớp khoản chi 75](#_Toc322814614)

[Bảng 3.4 Lớp thể loại thu 75](#_Toc322814615)

[Bảng 3.5 Lớp thể loại chi 76](#_Toc322814616)

[Bảng 3.6 lớp khoản nợ 76](#_Toc322814617)

[Bảng 3.7 lớp khoản vay 76](#_Toc322814618)

[Hình 3.2 Sơ đồ tuần tự thêm khoản thu 77](#_Toc322814619)

[Hình 3.3 Sơ đồ tuần tự thê khoản chi 78](#_Toc322814620)

[Hình 3.4 Sơ đồ tuần tự thêm thể loại chi 79](#_Toc322814621)

[Bảng 4.1 Danh sách các màn hình 80](#_Toc322814622)

[Hình 4.1 Màn hình khởi động 81](#_Toc322814623)

[Hình 4.2 Màn hình thêm khoản chi 82](#_Toc322814624)

[Hình 4.3 Màn hình chọn tên thể loại chi 83](#_Toc322814617)

[Hình 4.4 Màn hình thông tin khoản chi 84](#_Toc322814618)

[Hình 4.5 Màn hình Xóa khoản chi 85](#_Toc322814619)

[Hình 4.6 màn hình hỏi xóa khoản chi 86](#_Toc322814620)

[Hình 4.7 màn hình thêm khoản thu 87](#_Toc322814621)

[Hình 4.8 Màn hình thông tin khoản thu 88](#_Toc322814622)

[Hình 4.9 Màn hình sửa, xáo khoản thu 89](#_Toc322814623)

[Hình 4.10 Màn hình thể loại 90](#_Toc322814624)

[Hình 4.11 Màn hình thêm thể loại chi 91](#_Toc322814617)

[Hình 4.12 Màn hình sửa xóa thể loại chi 92](#_Toc322814618)

[Hình 4.13 Màn hình hỏi xóa thể loại chi 93](#_Toc322814619)

[Hình 4.14 Màn hình thêm thể lọa thu 94](#_Toc322814620)

[Hình 4.15 Màn hình sửa xóa thể loại thu 95](#_Toc322814621)

[Hình 4.16 Màn hình thêm khoản vay 96](#_Toc322814622)

[Hình 4.17 Màn hình thông tin khoản vay 97](#_Toc322814623)

[Hình 4.18 Màn hình xóa, sửa khoản vay 98](#_Toc322814604)

[Hình 4.19 Màn hình thêm khoản nợ 99](#_Toc322814605)

[Hình 4.20 Màn hình thông tin khoản nợ 100](#_Toc322814606)

[Hình 4.21 Màn hình xóa, sửa khoản nợ 101](#_Toc322814607)

[Hình 4.22 Màn hình thống kê 102](#_Toc322814608)

[Hình 4.23 Màn hình thống kê khoản thu 103](#_Toc322814609)

[Hình 4.24 Màn hình thống kê khoản chi 104](#_Toc322814609)

LỜI CẢM ƠN

Thực tập là quá trình tham gia học hỏi, so sánh, nghiên cứu và ứng dụng những kiến thức đã học vào thực tế công việc ở các cơ quan công ty.

Báo cáo thực tập vừa là cơ hội để sinh viên trình bày những nghiên cứu về vấn đề mình quan tâm trong quá trình thực tập, đồng thời cũng là một tài liệu quan trọng giúp giảng viên kiểm tra đánh giá quá trình học tập và kết quả thực tập của mỗi sinh viên.

Để hoàn thành báo cáo thực tập này, ngoài sự nỗ lực của bản thân, tôi trân trọng gửi lời cảm ơn sâu sắc đến:

- Các giảng viên của trường Cao Đẳng Công Nghệ Thông Tin TP.HCM đã tận tình giảng dạy, không chỉ truyền thụ cho tôi những kiến thức nền tảng mà còn là đạo đức và tinh thần của một sinh viên sắp ra trường trong tương lai.

- TH.S Trương Châu Long giảng viên phụ trách đã tận tình hướng dẫn, chỉ bảo tôi trước và trong quá trình thực tập, xây dựng báo cáo.

- Đặc biệt cảm ơn các anh chị đang công tác tại công ty COVISOF đã quan tâm, giúp đỡ, tin tưởng tạo điều kiện cho tôi tiếp xúc với công việc của quý cơ quan.

Xin chân thành cảm ơn!

TP.Hồ Chí Minh,ngày 9 tháng 6 năm 2013

Sinh Viên

**Nguyễn Hữu Phú**

|  |
| --- |
| NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN …………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………  *Tp.HCM, ngày…..tháng…….năm 2013*  ***Giáo viên hướng dẫn***  **TH.S Trương Châu Long** |

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

----------

**PHIẾU XÁC NHẬN SINH VIÊN THỰC TẬP**

Đơn vị: **Công Ty Covisof**

Xác nhận sinh viên: **Nguyễn Hữu Phú**

Lớp: C10CNPM

Khoa: Công Nghệ Thông Tin

Trường: **Cao Đẳng Công Nghệ Thông Tin TP.HCM**

Thời gian thực tập từ ngày 15/04/ 2013 đến ngày 09/06/ 2013

Trong thời gian thực tập tại công ty,em có ý thức chấp hành nghiêm chỉnh các nội quy, nề nếp, kỷ luật tại đơn vị thực tập, có tinh thần học hỏi và hoàn thành đầy đủ các công việc theo sự hướng dẫn, giúp đỡ của các nhân viên và các bộ phận tại đơn vị thực tập.

TP.HCM,Ngày 9 tháng 06 năm 2013

**XÁC NHẬN CỦA ĐƠN VỊ THỰC TẬP**

                                                                                      (Ký tên, đóng dấu)

|  |  |
| --- | --- |
| Quản Lý  Tài Chính | PHẦN 1 **Mở Đầu** |
|  | |

**1.Tính cấp thiết khi chọn đề tài**

Quản lý tài chính cá nhân trong quá trình tính được khoản thu chi là một khâu quan trọng trong hệ thống quản lý tài chính. Để có thể quản lý tốt khoản thu/chi, khoản vay, khoản nợ của một cá nhân việc đó phải đòi hỏi cá nhân đó phải tìm một giải pháp quản lí chi tiêu của mình cho phù hợp và đạt hiểu quả cao về vấn đề thu chi cho cả tháng.

Đối với mỗi cá nhân trong quá trình quản lí tài chính của mình,nếu muốn quản lý tốt, đạt hiểu quả cao thì phải có sự phối hợp thống nhất, chính xác khâu quản lý của mình. Trong đời sống của mỗi cá nhân việc đang đau đầu với các khoản thu/chi của mỗi cá nhân nào là tiền điện, nước, điện thoại,tiền tiêu vặt, và còn có mỗi cá nhân cũng cần phải thống kê cho mình về các khoản vay, khoản nợ của mình. Trong đời sống khi cho bạn bè, đối tượng đối khách.Bởi vì lẻ đó bạn cần phải biết quản lý các khoản thu của mình để phù hợp và đủ chi tiêu cho gia đình và cá nhân của bạn. Bạn cũng phải cần lập ra các khoản mục chi tiêu, từ đó có thể tổng kết việc chi tiêu hàng tuần, tháng hay thậm chí là một năm, bạn có thể rút ra cho mình chiến lược, hay đơn giản là cách sử dụng số tiền của mình sao cho hợp lý.

Em xin giới thiệu với các bạn một ***“phần mềm quản lý tài chính cá nhân”*** khá hữu ích. Đây là một phần mềm của Việt Nam, chính vì vậy nên giao diện, ngôn ngữ rất thân thiện, hơn nữa bạn có thể sử dụng hoàn toàn miễn phí.

**2.Mục đích nghiên cứu:**

Em hi vọng rằng qua thời gian nghiên cứu em có dịp tiếp cận với hệ điều hành android, một số ngôn ngữ mới , cụ thể là công ty Covisof để em có thể trao dồi, hiểu rõ hơn những kiến thức mà em đã được học ở trường.

**3.Đối tượng nghiên cứu:**

Với đề tài này em sẽ tìm hiểu về hệ điều hành android, ngôn ngữ Java và viết được ứng dụng trên nền tản android, tìm hiểu về cách đánh giá ngôn ngữ Java và viết code trên nền tản android với các ngôn ngữ khác như C, C++, C#. Hiểu được phương pháp để xây dựng một phần mềm như thế nào cho hợp lí với thị trường. Sau đó em sẽ nghiên cứu cụ thể vào ngôn ngữ Java được xem là nền tản của cuộc cách mạng lập trình .Qua đó để thấy được ưu điểm và nhược điểm của ***“phần mềm quản lý tài chính cá nhân”*** của riêng mình.

|  |  |
| --- | --- |
| Quản Lý  Tài Chính | PHẦN 2 **Giới Thiệu Về Công Ty Covisof** |
|  | |

**1 Giới thiệu sơ lược về công ty:**

* Tên doanh nghiệp: Công ty COVISOF
* Đơn vị chủ quản: Covisof ltd
* Thư viện điện tử: Staf@easylife.com.vn
* Trụ sở chính: Charm IT Center, Công viên phần mềm Quang Trung, phường Tân Chánh Hiệp, quận 12, Thành phố Hồ Chí Minh
* Tel/Fax: ( 08) 37154714
* Website công ty: [http://easylife.com.vn](http://easylife.com.vn/)

**2 Lịch sử hình thành công ty và phát triển của công ty:**

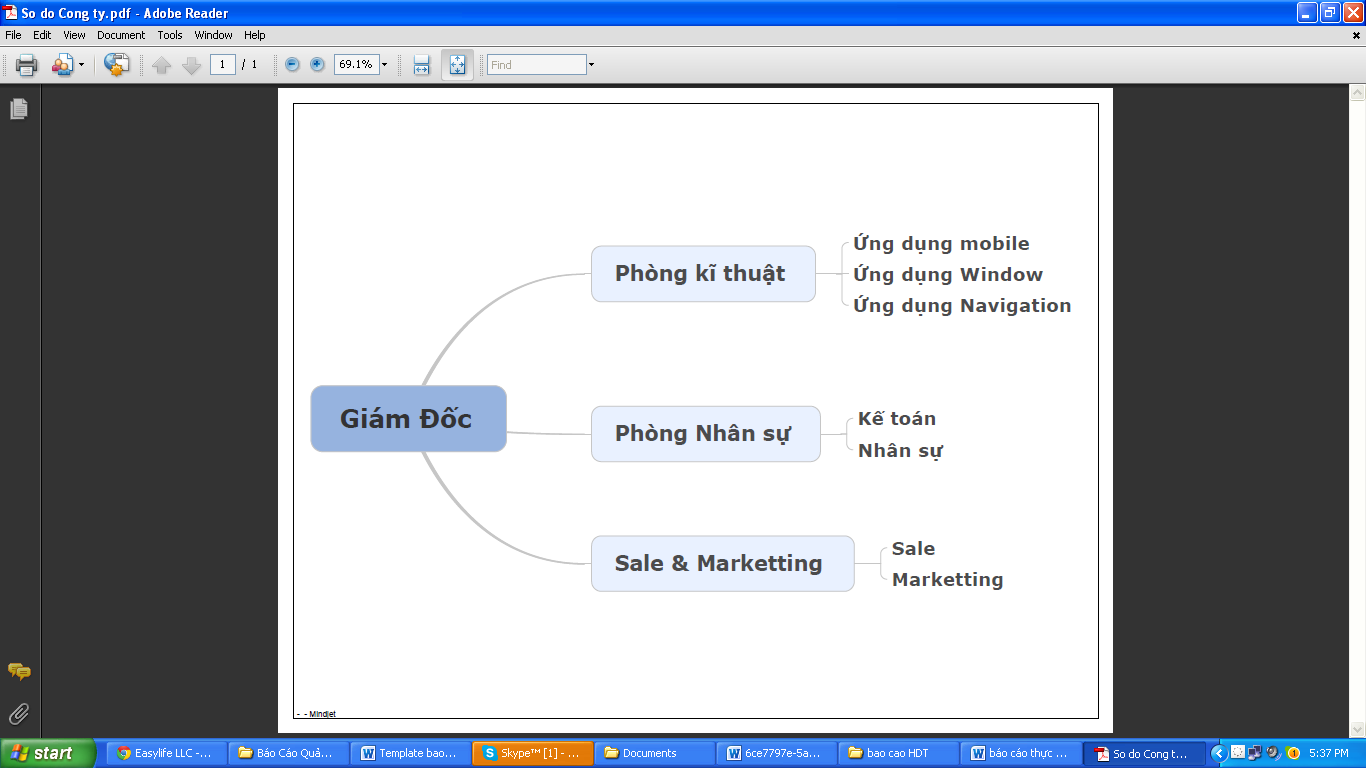
* Ngày cấp giấy phép kinh doanh: 19/4/2012.
* Ngày hoạt động: 19/4/2012.
* Người đại diện theo pháp luật của công ty:
* Bà: Đặng Thị Ngọc Giàu - Chức danh: Giám đốc công ty.

**3 Lĩnh vực kinh doanh của công ty:**

* Gia công các loại phần mềm nước ngoài.
* Sale Application do công ty phát triển
* Phần mềm Easylife là phần mềm đa phương tiện dành riêng cho di động sử dụng với hệ điều hành Android và IOS.

**4 Cơ cấu tổ chức nhân sự của công ty Covisof:**

**4.1 Sơ đồ tổ chức công ty:**



*Hình 2 Sơ đồ công ty Covisof*

**4.2 Cơ cấu tổ chức chức năng của các bộ phận phòng ban trong công ty:**

**Giám Đốc**:

* Là người chỉ đạo điều hành mọi hoạt động của công ty.
* **P. Sale & Marketting**:
* Là bộ phận rất quan trọng của công ty.
* Chịu trách nhiệm về nghiên cứu thị trường, đàm phán, kí kết hợp đồng.
* Tổ chức mua bán với các tổ chức kinh tế khác.
* Tiếp cận thị trường kịp thời.
* Nắm bắt thị hiếu của người tiêu dùng đáp ứng nhu cầu kinh doanh của công ty.
* Phối hợp với các phòng ban để thực hiện kế hoạch theo chỉ đạo của cấp trên.
* **P. nhân sự:**
* Có nhiệm vụ theo dõi ghi chép mọi hoạt động sản xuất kinh doanh của công ty.
* Đồng thời quản lý tốt các khoản thu chi công nợ, các khoản nộp ngân sách nhà nước.
* Những biến động về vốn nhằm báo cáo kịp thời cho ban giám đốc.
* Phối hợp với phòng sale & marketting tổ chức thanh toán cho các hợp đồng.
* Đồng thời lên kế hoạch tài chính trên cơ sở kế hoạch lưu chuyển hàng hóa của phòng kinh doanh.
* **P.kĩ thuật:**
* Chịu trách nhiệm và nghiên cứu các ứng dụng moblie, ứng dụng window, ứng dung Navigation
* Đồng thời kiểm tra hoàn thiện công trình từ khi viết ứng dụng đến debug cho đến khi hoàn thành và bàn giao cho khách hàng.

|  |  |
| --- | --- |
| Quản Lý  Tài Chính | PHẦN 3 **Tổng Quan** |
| Chương 1. Sơ Lược Về Hệ Điều Hành  [Chương 2.](#_2.2._Danh_sách) Mô Tả Về Sơ Đồ Usecase  [Chương 3.](#_2.3._Danh_sách) Sơ Đồ Phân Tích  [Chương 4.](#_2.4._Đặc_tả) Thiết Kế Giao Diện | |

**CHƯƠNG 1: SƠ LƯỢC VỀ HỆ ĐIỀU HÀNH**

**1.1 SƠ LƯỢC VỀ CÁC HỆ ĐIỀU HÀNH**

Hệ điều hành là chương trình chạy trên hệ thống máy tính, quản lý các tài nguyên trên máy tính và là môi trường cho các ứng dụng chạy trên nó.

Ngày nay, khái niệm hệ điều hành không chỉ là trên máy tính mà còn được mở rộng cho nhiều thiết bị điện tử khác chẳng hạng như điện thoại thông minh ( smart phone), các thiết bị cầm tay PDA v.v…

Như vậy hệ điều hành di động là hệ điều hành chạy trên hệ thống máy có tính di động cao. Với đặc thù đó, hệ điều hành di động phải có những khả năng đặc biệt mà những hệ điều hành thông thường không có được. Chẳng hạn như nó phải chạy trên hệ thống máy có cấu hình máy hạn chế về tốc độ bộ vi xử lý, bộ nhớ sử dụng, phải chạy được ổn định liên tục trong một thời gian dài mà chỉ sử dụng một lượng điện năng nhỏ, trong suốt thời gian chạy đó có thể duy trì các kết nối mạng không dây để đảm bảo liên lạc

Một số hệ điều hành tiêu biểu:

* Trên máy tính cá nhân: MS DOS, MS WINDOW , MACOS, LINUX, UNIX
* Trên điện thoại thông minh: Android, Sybian, Window Mobile, iPhone OS, BlackBerry, S60, Bada OS, Palm OS..
* Ngoài ra còn có các hệ điều hành chạy trên mainframe, server, thẻ chip…
* Trong phạm vi đồ án này chúng ta sẽ nói về hệ điều hành Android dành cho điện thoại.

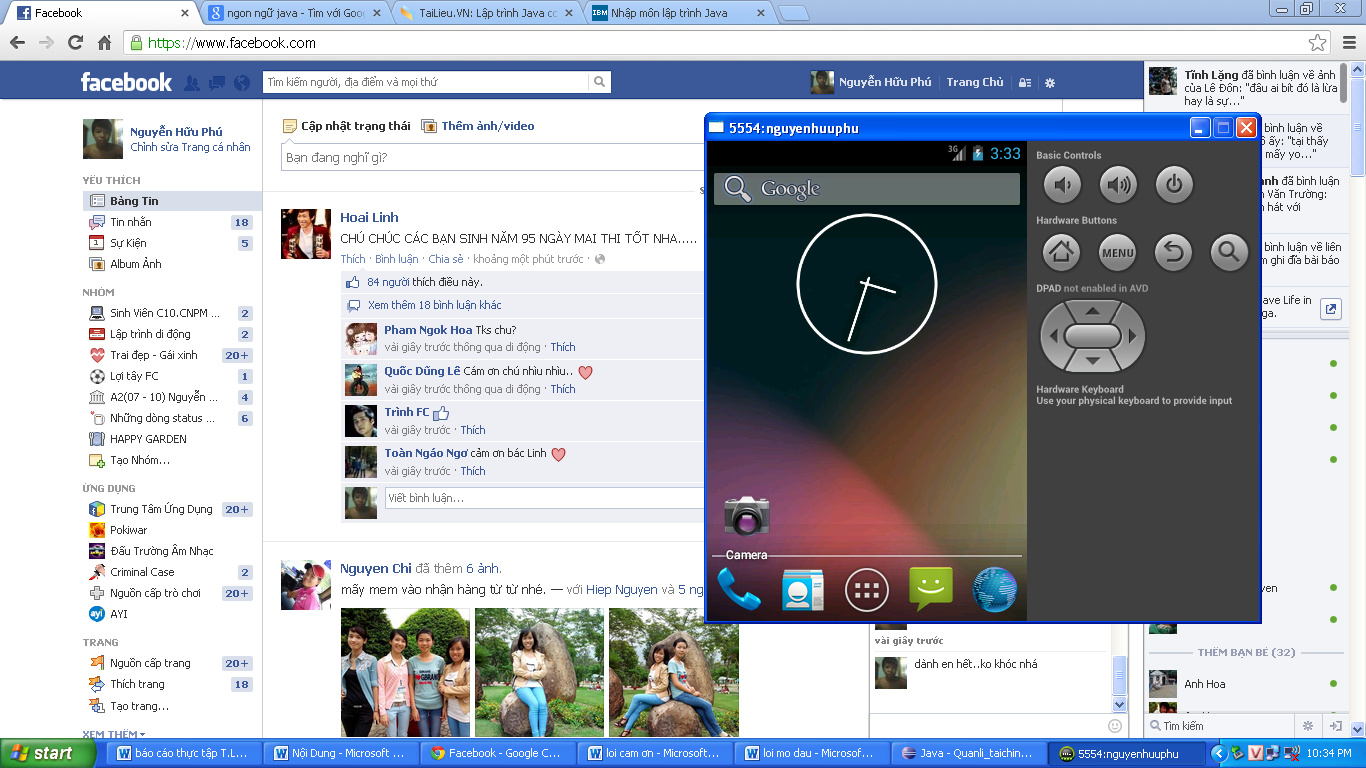
**1.2 ANDROID**

Android là hệ điều hành gồm 12 triệu dòng code

Các ứng dụng không được phép truy cập đến dữ liệu của nhau, cũng không được phép truy cậpđến dữ liệu của HĐH, trừ khi được chính ứng dụng đó cung cấp (thông qua các Content Provider mà sau này chúng ta sẽ được học).

Các ứng dụng hỗ trợ khả năng giao tiếp với người dùng đến đâu là do khả năng và thiết kế của người lập trình, còn bản thân Android đã hỗ trợ đầy đủ các giao tiếp cho ứng dụng.

Android – hệ điều hành dành cho điện thoại di động được phát triển bởi Google và ngày càng trở nên phổ biến với các hãng liên tục ra mắt các mẫu điện thoại sử dụng Android.



Hình 1: Giao diện màn hình chính (Android 4.2.2)

* Lịch sử phát triển của Android

Cách các nhà sản xuất điện thoại dùng Android là họ cho ra một dòng điện thoại, lấy bản open source của Android về, chỉnh sửa lại cho phù hợp với dòng điện thoại đó, rồi cài vào và tung ra thị trường. Do đó các bản Android khác nhau được gọi là các bản Room khác nhau của Android.

Thiết bị chuẩn của Android bao gồm 4 phím chính (cứng hoặc mềm) là home, back, search và menu (iPhone chỉ có duy nhất phím Home).

-          Phím Home sẽ quay trở về màn hình home

-          Phím back quay lại màn hình trước đó

-          Phím search và menu sẽ tùy từng ứng dụng mà chúng sẽ có action tương ứng.

Android được xây dựng trên nhân linux và được phân phối miễn phí. Không giống như Windows mobile và Apple iPhone, tuy cả hai đều cung cấp môi trường phát triển ứng dụng phong phú và đơn giản dễ tiếp cận nhưng luôn có sự ưu tiên cho các ứng dụng mặc định có sẵn của hệ điều hành (native applications). Với Android mọi ứng dụng đều được viết trên cùng một tập API, thế nên không có sự phân biệt giữa các ứng dụng mặc định và các ứng dụng của bên thứ ba. Người dùng hoàn toàn có thể thay thế mọi ứng dụng mặc định bằng các ứng dụng yêu thích của mình, thậm chí ngay cả màn hình thực hiện cuộc gọi mà màn hình nhà (home scream).

Các nhà phát triển ứng dụng cho Android dựa trên ngôn ngữ Java. Sự ra mắt của Android vào ngày 5 tháng 11 năm 2007 gắn với sự thành lập liên minh thiết bị cầm tay mã nguồn mở, bao gồm 78 công ty phần cứng và phần mềm viễn thông nhằm mục đích tạo nên chuẩn mở cho điện thoại di động trong tương lai. Google công bố hầu hết các mã nguồn Android theo bản cấp phép Apache.



Hình 1.1. Các thành viên của liên minh di động mở.

**Các ứng dụng có sẵn trên Android**

Một điện thoại Android thông thường sẽ đi kèm với một vài ứng dụng có sẵn, bao gồm:

* Một trình email tương thích với Gmail
* Chương trình quản lý tin nhắn SMS
* Chương trình quản lý thông tin cá nhân, bao gồm cả lịch làm việc, danh bạ và được đồng bộ hóa với dịch vụ Google
* Phiên bản thu gọn của Google Map cho điện thoại, bao gồm StreetView, tìm kiếm địa chỉ, chỉ đường, tình trạng giao thông…
* Trình duyệt Web dựa trên nhân Webkit.
* Chương trình tán gẫu (Chat).
* Trình đa phương tiện ( chơi nhạc, xem phim…).
* Android MarketPlace cho phép người dùng tải về và cài đặt các ứng dụng mới.

Tất cả các ứng dụng có sẵn đều được viết bằng ngôn ngữ Java và sử dụng Android SDK.

Các dữ liệu về thông tin người dùng được các ứng dụng có sẵn sử dụng như thông tin về danh bạ vẫn hoàn toàn có thể được sử dụng bởi các ứng dụng của bên thứ ba. Tương tự vậy, ứng dụng của bạn hoàn toàn có thể xử lý các sự kiện như các cuộc gọi đến, nhận một tin nhắn mới… thay cho các ứng dụng có sẵn.

* **Truy cập phần cứng**

Android bao gồm các thư viện API giúp đơn giản hóa tối đa việc sử dụng phần cứng của thiết bị. Điều đó đảm bảo rằng bạn không cần phải bận tâm nhiều đến việc ứng dụng của mình có thể chạy như mong đợi trên nhiều thiết bị khác nhau hay không, miễn là thiết bị đó có hỗ trợ Android.

Android SDK bao gồm các API cho phần cứng :GPS, Camera, kết nối mạng, WIFI, Bluetooth, con quay gia tốc, màn hình cảm ứng, quản lý năng lượng…

* **Dịch vụ chạy nền**

Android hỗ trợ các ứng dụng và dịch vụ được thiết kế chạy ẩn. Do kích thước nhỏ của màn hình điện thoại nên tại một thời điểm chỉ có thể thấy một ứng dụng. Dịch vụ chạy nền giúp tạo ra các thành phần ứng dụng “vô hình” để thực hiện tự động một tác vụ nào đó mà không cần phải có sự tương tác của người dùng. Ví dụ như một dịch vụ chạy nền có chức năng chặn cuộc gọi đến đối với các số điện thoại có trong “black list” chẳng hạn.

* **SQLite Database**

Bởi vì tính chất nhỏ gọn và bị hạn chế về phần cứng của điện thoại di động, cho nên đòi hỏi việc lưu trữ và truy xuất dữ liệu phải nhanh chóng và hiệu quả. Android hỗ trợ hệ quản trị nhỏ gọn SQLite, và cung cấp cho ứng dụng các API để thao tác. Mặc định mỗi ứng dụng đều được chạy trong SandBox (hộp cát) điều này có nghĩa là nội dung của từng database ứng với từng ứng dụng chỉ có thể truy cập bằng chính ứng dụng đó. Tuy nhiên cũng có các cơ chế để các ứng dụng chia sẽ, trao đổi các database với nhau.

* **Hệ thống thông báo**

Thông báo là cách thức tiêu chuẩn mà ở đó thiết bị báo cho người dùng đã có một sự kiện nào đó đã xảy ra. Chẳng hạn như có cuộc gọi tới, máy sắp hết pin… Sử dụng các API bạn có thể cho ứng dụng của mình thông báo tới người dùng bằng âm thanh, rung, hoặc thậm chí cả đèn LED của thiết bị.

* **Tối ưu hóa bộ nhớ và quản lý tiến trình**

Việc quản lý bộ nhớ và tiến trình trong Android cũng có một chút khác biệt giốn như công nghệ Java và .NET, Android sử dụng một bộ Run-time của riêng mình với công nghệ ảo hóa để quản lý bộ nhớ của các ứng dụng đang chạy. Không giống như những nền tản khác, Android Run-time cũng đồng thời quản lý luôn cả thời gian sống của ứng dụng. Android đảm bảo các ứng dụng đều được đáp ứng bằng cách dừng và hủy các tiến trình không cần thiết để giải phóng các tài nguyên cho các tiến trình có độ ưu tiên cao hơn.

Trong bối cảnh đó, độ ưu tiên được xác định tùy thuộc vào ứng dụng mà người dùng đang tương tác. Android đảm bảo rằng các ứng dụng được hủy một cách nhanh chóng, đồng thời cũng khởi động là nhanh cũng không kém nếu cần. Điều này thật sự quan trọng trong một môi trường mà ở đó bản thân ứng dụng không thể tự kiểm soát được thời gian sống cho mình.

**Android software development kit (SDK)**

Bộ SDK của Android bao gồm mọi thứ cần thiết giúp bạn có thể lập trình, debug, test ứng dụng Android.

* **Android API**: Cốt lõi của bộ SDK là thư viện các hàm API và Google cũng chỉ sử dụng bộ API này để xây dựng các ứng dụng có sẵn cho Android.
* **Development tool**: SDK bao gồm rất nhiều công cụ để giúp biên dịch, sửa lỗi và hỗ trợ trong việc lập trình ứng dụng.
* **Android Emulator**: Trình giả lập thiết bị chạy Android thực sự với nhiều Skin thay thế, cực kì tiện lợi cho việc test ứng dụng Android ngay trên máy tính mà không cần phải thông qua một thiết bị chạy Android thực.
* **Tài liệu**: SDK bao gồm một bộ tài liệu rất chi tiết, giải thích cặn kẽ chính xác những gì bao gồm trong mỗi page, class cùng với cách sử dụng chúng. Ngoài tài liệu về “code”, còn có những tài liệu dùng để “getting started” và giải thích các nguyên tắc và cơ chế hoạt động của ứng dụng trong Android.
* **Code mẫu**: SDK bao gồm các ứng dụng mẫu đơn giản minh họa cho các tính năng nổi bật trên Android, cũng như các ứng dụng demo cách sử dụng các tính năng của bộ API.

**Kiến trúc ứng dụng**

Ý tưởng của Android là việc khuyến khích tái sử dụng lại các thành phần đã có, cho phép ứng dụng của bạn có thể chia sẻ Activity, Service, Dữ liệu với các ứng dụng khác nhau trong giới hạn bạn đặt ra.

Sau đây là kiến trúc của mọi ứng dụng Android:

* *Activity Manager* : Kiểm soát vòng đời của Activity.
* *View* : Xây dựng giao diện người dùng cho Activity.
* *Notification Manager*: Cung cấp một cơ chế thống nhất và an toàn để ứng dụng có thể đưa ra các thông báo cho người dùng.
* *Content Provider*: Giúp trao đổi và chia sẽ dữ liệu giữa các ứng dụng với nhau.
* *Resource Manager*: Hỗ trợ quản lý các tài nguyên không là code như các chuỗi, hình ảnh, và âm thanh…

**Các thư viện của Android**

Android cung cấp các gói API để phát triển ứng dụng. Sau đây là các API mà tất cả các thiết bị Android đều tối thiểu phải hỗ trợ để giúp cho chúng ta một cái nhìn tổng quát về thư viện này.

* Android.util : Gói API lõi, chứa các class cấp thấp như container, string formatter, XML parsing.
* Android.os : Truy cập tới chức năng của hệ điều hành như : gởi nhận tin nhắn, giao tiếp nội bộ giữa các ứng dụng, thời gian…
* Android.graphics: Cung cấp các lớp liên quan tới xử lý đồ họa ở mức thấp. Hỗ trợ các hàm cơ bản như vẽ điểm, vẽ miền, tô màu.. trên khung canvas.
* Android.text: Cung cấp các hàm phân tích và xử lý chuỗi.
* Android.database: Cung cấp các lớp cấp thấp cần thiết để làm việc với database.
* Android.content: Dùng để quản lý các tài nguyên, các nội dung và các gói.
* Android.view: Views là lớp cha của mọi lớp giao diện người dùng.
* Android.widget: Được thừa kế từ lớp View, bao gồm các lớp cơ bản để xây dựng giao diện widget như: list, button, layout..
* Android.map: Gói API cấp cao, dùng để truy cập tới các chức năng của GoogleMap.
* Android.app: Gói API cấp cao, bao gồm các Activity và Service – hai lớp cơ sở cho mọi ứng dụng Android.
* Android.telephony: Cung cấp cho bạn khả năng tương tác trực tiếp với các chức năng cơ bản của một điện thoại như nghe, gọi, tin nhắn
* Android.webkit: cung cấp một webView control trên nền webkit để có thể nhúng ứng dụng, cùng với các API điều khiển cơ bản như stop, refresh, cookieManager…

**1.3. Delving với máy ảo DALVIK**

Dalvik là máy ảo giúp các ứng dụng Java chạy được trên các thiết bị di động Android. Nó chạy các ứng dụng đã được chuyển đổi thành một file thực thi Dalvik (dex). Định dạng phù hợp cho các hệ thống mà thường bị hạn chế về bộ nhớ và tốc độ xử lý. Dalvik đã được thiết kế và viết bởi Dan Bornstein, người đã đặt tên cho nó sau khi đến thăm một ngôi làng đánh cá nhỏ có tên là Dalvík ở đảo Eyjafjörður, nơi mà tổ tiên của ông sinh sống.

Từ góc nhìn của một nhà phát triển thì Dalvik trông giống như máy ảo java (Java Virtual Machine) nhưng thực tế thì hoàn toàn khác. Khi nhà phát triển viết một ứng dụng dành cho Android, anh ta thực hiện các đoạn mã trong môi trường Java. Sau đó, nó sẽ được biên dịch sang các byteCode của Java, tuy nhiên để thực thi được ứng dụng này trên Android thì nhà phát triển phải thực thi một công cụ có tên là dx. Đây là công cụ dùng để chuyển đổi byteCode sang một dạng gọi là dex bytecode. Dex là từ viết tắc của “Dalvik executable” đóng vai trò như cơ chế thực thi các ứng dụng Java.

## 1.4. Kiến trúc hệ điều hành Android

Mô hình sau thể hiện một cách tổng quát các thành phần của hệ điều hành Android. Mỗi phần sẽ được đặc tả một cách chi tiết dưới đây.



Hình 1.2 Cấu trúc Stack hệ thống Android

*1.4.1. Tầng ứng dụng (applications)*

Android được tích hợp sẵn một số ứng dụng cần thiết cơ bản như: Contacts, browser, camera, phone…Tất cả ứng dụng chạy trên hệ điều hành Android đều được viết bằng Java.

*1.4.2. Application Framework*

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa.

Nhà phát triển có thể truy cập vào các API cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế đơn giản hóa việc sử dụng lại các thành phần, bất kì ứng dụng có thể xuất bản khả năng của mình và ứng dụng nào khác sau đó có thể sử dụng những khả năng( có thể hạn chế được bảo mật thực thi bởi khuôn khổ). Cơ chế này cho phép các thành phần tương tự sẽ được thay thế bởi người sử dụng.

Cở bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống bao gồm:

* Một tập hợp rất nhiều các View có khả năng kế thừa lẫn nhau dùng để thiết kế phần giao diện ứng dụng như : gridView, tableView, linearLayout,…
* Một “Content Provider” cho phép tất cả ứng dụng truy xuất dữ liệu từ các ứng dụng khác ( chẳng hạn như Contact) hoặc là chia sẻ dữ liệu giữa các ứng dụng đó.
* Một “Resource Manager” cung cấp truy xuất tới các tài nguyên không phải là mã nguồn, chẳng hạn như: localized string, graphics, and layout file.
* Một “Notifycation Manager” cho tất cả các ứng dụng hiển thị các custom alerts trong status bar.

Activity Manager được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

*1.4.3. Library*

Android bao gồm một tập hợp các thư viện C/C++ được sử dụng bởi nhiều thành phần khác nhau trong hệ thống Android. Điều này được thể hiện thông qua nền tảng ứng dụng Android. Một số các thư viện cơ bản được liệt kê dưới đây:

* Hệ thống thư viện C: một thực hiện BSD có nguồn gốc từ hệ thống thư viện chuẩn C (libc), chuyển cho các thiết bị nhúng dựa trên Linux.

- Thư viện truyền thông - dựa trên PacketVideo của OpenCore, các thư viện hỗ trợ phát lại và ghi nhiều định dạng audio và video, cũng như các tập tin hình ảnh tĩnh, bao gồm: MPEG4, H, 264, MP3, AAC, AMG, JPG, và PNG

* Bề mặt quản lý – Quản lý việc truy xuất vào hệ thống hiển thị.

- LibWebCore - một công cụ trình duyệt web hiện đại, quyền hạn cả các trình duyệt Android và một xem web embadable.

- SGL - công cụ đồ họa cơ bản 2D.

- Thư viện 3D - một thực hiện dựa trên OpenGL ES 1.0 API, các thư viện sử dụng hoặc tăng tốc 3D phần cứng (nếu có) hoặc bao gồm, tối ưu hóa cao 3D rasterizer phần mềm.

- FreeType - bipmap và vector vẽ font.

* SQLite - một cơ sở dữ liệu quan hệ mạnh mẽ và trọng lượng nhẹ có sẵn cho tất cả các ứng dụng.

*1.4.4. Android runtime*

Android bao gồm tập hợp các thư viện cơ bản mà cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Tất cả các ứng dụng Android đều chạy trong tiến trình riêng. Máy sảo Dalvik đã được viết để cho một thiết bị có thể chạy nhiều máy ảo hiệu quả. Các VM Dalvik thực thi các tập tin thực thi Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là dựa trên register – base, và chạy các lớp đã được biên dịch bởi một trình biên dịch Java để chuyển đổi thành các định dạng dex. Các VM Dalvik dựa vào nhân Linux cho các chức năng cơ bản như luồng và quản lý bộ nhớ thấp.

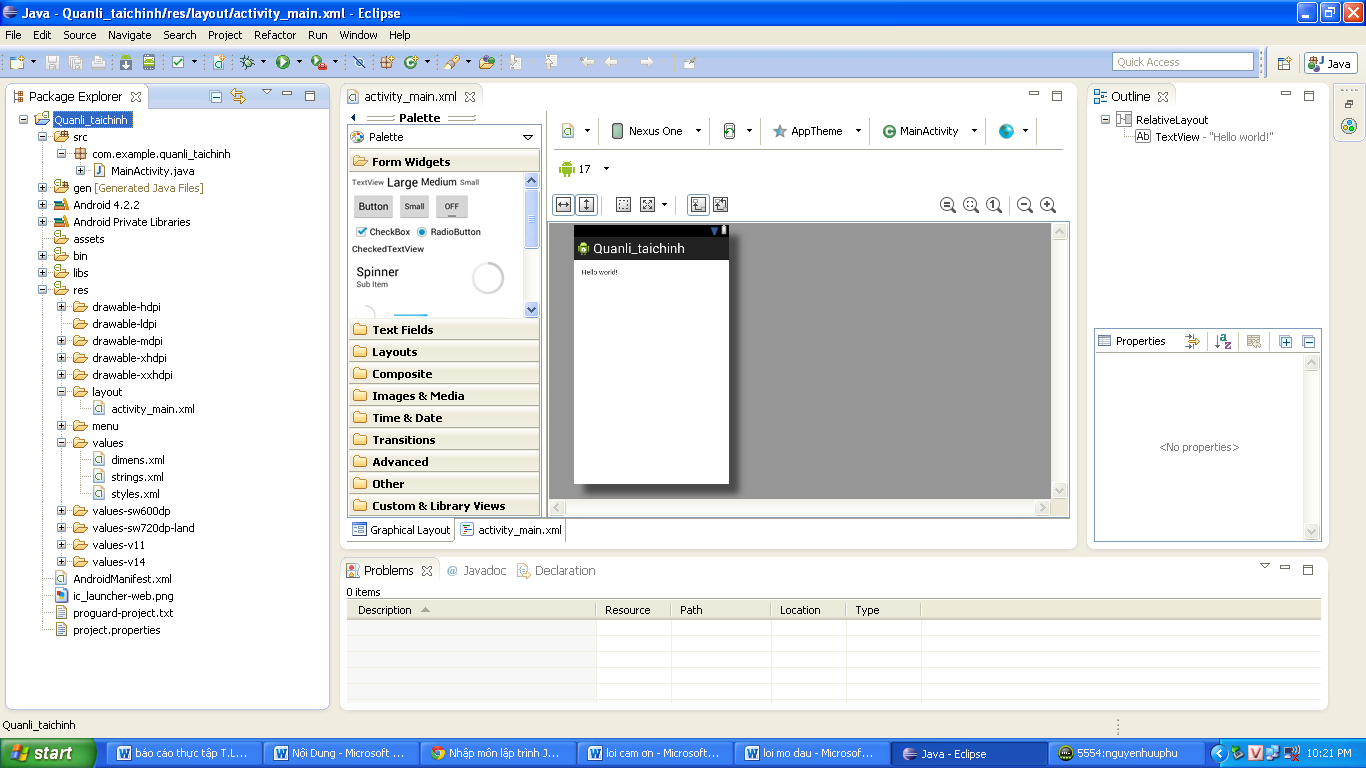
*1.4.5. Linux kernel*

Android dựa trên Linux phiên bản 2.6 cho hệ thống dịch vụ cốt lõi như security, memory managerment, process managerment, network stack, and driver model. Kernel Linux hoạt động như một lớp trừu tượng hóa giữa phần cứng và phần còn lại của phần mềm stack.

## 1.5. Các thành phần trong ứng dụng Android

Android project là một hệ thống thư mục file chứa toàn bộ source code, tài nguyên… mà mục đích cuối cùng là để đóng gói thành một file .apk duy nhất.

Trong một thư mục project, có một số thành phần (file, thư mục con) được tạo ra mặc định, còn lại phần lớn sẽ được tạo ra sau nếu cần trong phát triển ứng dụng.



Hình 1.3: Các thành phần trong một Android project

* ***Src/:*** Chứa toàn bộ source code (file .java hoặc .aidl)
* ***Bin/:*** Thư mục chứa file Output sau khi build. Đây là nơi bạn có thể tìm thấy file .apk
* ***Gen/:*** Chứa file .java tạo ra bởi ADT plug-in, như là file R.java hoặc các giao diện tạo ra từ file AIDL.
* ***Res/:*** Chứa các tài nguyên (resource) cho ứng dụng chẳng hạn như file hình ảnh, file layout, các chuỗi (string)…Dưới đây là các thư mục con của nó.
* ***Anim/:*** Chứa các file .xml dùng cho việc thiết lập các hiệu ứng động(animation).
* ***Color/:*** Chứa các file .xml dùng định nghĩa màu sắc.
* ***Drawable/:*** Chứa hình ảnh (png, jpeg, gif), file .xml định nghĩa cách vẽ các loại hình dạng khác nhau (shape).
* ***Layout/:*** Chứa file .xml dùng để dựng giao diện người dùng.
* ***Menu/:*** Chứa file .xml quy định application menu.
* ***Raw/:*** Chứa các file media, chẳng hạn như .mp3, .ogg
* ***Values/:*** Chứa file .xml định nghĩa các giá trị. Khác với các resource trong thư mục khác, resource ở thư mục này khi định danh trong lớp R thì sẽ không sử dụng file name để định danh mà sẽ được định danh theo quy định bên trong file .xml đó.
* ***Xml/:*** Dùng chứa các file .xml linh tinh khác, chẳng hạn như file .xml quy định app widget, search metadata,…
* ***Libs/:*** Chứa các thư viện riêng.
* ***AndroidManifest.xml/:*** File kiểm soát các thành phần trong ứng dụng như: activity, service, intent, receiver… tương tác với nhau, cách ứng dụng tương tác với ứng dụng khác, cũng như đăng kí các quyền hạn về sử dụng tài nguyên trong máy.
* ***Build.properties/:*** Tùy chỉnh các thiết lập cho hệ thống build, nếu bạn sử dụng Eclipse thì file này không cần thiết.
* ***Build.xml/:*** Chỉ sử dụng khi dùng dòng lệnh để kiến tạo project.
* ***Default.properties/:*** File này chứa các thiết lập cho project, chẳng hạn như build target, min SDK version…(tốt hơn hết là không nên chỉnh sửa file này bằng tay)
* **File AndroidManifest.xml**

Là  nền tảng của mọi ứng dụng Android, file AndroidManifest.xml được đặt trong thư mục root và cho biết những thành phần có trong ứng dụng của: các activities, các services,...cũng như cách các thành phần ấy gắn bó với nhau.

Mỗi file manifest đều bắt đầu với một thẻ manifest:

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.commonsware.android.search"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

...

</manifest>

Khi tạo file manifest, điều quan trọng nhất là phải cung cấp thuộc tính của package, tức tên của Java package dùng làm cơ sở cho ứng dụng của ta. Sau khi đã đặt tên package, sau này khi cần ghi tên lớp trong file manifest, ta có thể rút ngắn, chẳng hạn với lớp “com.yourapp.android.search.Someclass” ta chỉ cần ghi “.Someclass” là đủ.

Các thành phần manifest khác là :

* **uses-persmission:** chỉ định các quyền mà ứng dụng của ta đuợc cấp để hoạt động trôi chảy (như đã nói, các ứng dụng Android nằm dưới nhiều lớp bảo mật khác nhau).
* **permission**: chỉ định các quyền mà các activities hay services yêu cầu các ứng dụng khác phaỉ có mới được truy cập dữ liệu của ứng dụng của ta.
* **instrumentation**: chỉ định phần code cần được gọi khi xảy ra những sự kiện quan trọng (chẳng hạn khởi động activities) nhằm phục vụ việc ghi chú (logging) và tra soát (monitoring)
* **uses-library**: nhằm kết nối với các thành phần có sẵn của Android (như service tra bản đồ,...)
* **uses-sdk:** có thể có hoặc không, chỉ ra phiên bản củaAndroid mà ứng dụng này yêu cầu.
* **application**: định nghĩa phần trung tâm của ứng dụng của file manifest.

Tất nhiên, phần quan trọng của 1 file manifest chính là thành phần application. Mặc định, khi ta tạo 1 project Android mới, ta có sẵn 1 thành phần activity:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.example.quanli\_taichinh"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"8"*

android:targetSdkVersion=*"17"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.quanli\_taichinh.MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

</manifest>

Thành phần này cung cấp các thông tin sau

**android:name** : tên class hiện thực activity này.

**android:label** : tên activity.

**intent-filter** : Một thành phần con, chỉ ra dưới điều kiện nào thì activity này được hiển thị.

* **Các thành phần cơ bản của ứng dụng Android**
* **Activity:**

Là thành phần tối quan trọng của bất kỳ một ứng dụng Android nào. Thuật ngữ Activity chỉ một việc mà người dùng có thể thực hiện trong một ứng dụng Android. Do gần như mọi activity đều tương tác với người dùng, lớp Activity đảm nhận việc tạo ra một cửa sổ (window) để người lập trình đặt lên đó một giao diện UI với setContentView(View). Một activity có thể mang nhiều dạng khác nhau: Một cửa sổ toàn màn hình (full screen window), một cửa sổ floating (với windowsIsFloating) hay nằm lồng bên trong 1 activity khác (với ActivityGroup).

Có 2 phương thức mà gần như mọi lớp con của Activity đều phải hiện thực:

* **onCreate(Bundle)** - Nơi khởi tạo activity. Quan trọng hơn, đây chính người lập trình gọi setContentView(int) kèm theo layout để thể hiện UI của riêng mình. Đồng thời còn có findViewById(int) giúp gọi các widget (buttons, text boxes, labels,..) để dùng trong UI.
* **onPause()** - Nơi giải quyết sự kiện người dùng rời khỏi activity. Mọi dữ liệu được người dùng tạo ra tới thời điểm này cần phải được lưu vào ContentProvider.

*Vòng đời của Activity*

Các activity được quản lí dưới dạng các activity stack - First-In-Last-Out: Khi một activity mới được khởi tạo, nó sẽ được đưa lên trên cùng stack, các activity khác muốn chạy trên nền (foreground) trở lại thì cần phải chờ tới khi Activity mới này kết thúc.

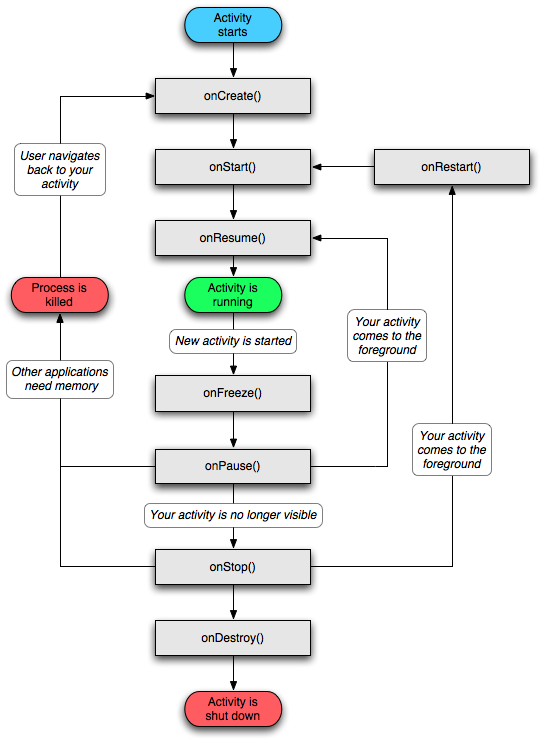
Một Activity có 4 trạng thái:

**Active** hay **Running:** Khi một activity đang chạy trên màn hình.

**Paused:** Khi một activity vẫn đang chạy trên màn hình nhưng đang bị một activity trong suốt (transparent) hay không chiếm toàn màn hình hiển thị phía trên. Tuy vẫn lưu trữ dữ liệu, nhưng các **paused activity** này sẽ bị hệ thống bắt chấm dứt khi đang thiếu bộ nhớ trầm trọng.

**Stopped:** Khi 1 activity bị che khuất hoàn toàn bởi 1 activity khác. Tuy vẫn lưu trữ dữ liệu, nhưng các stopped activity này sẽ thường xuyên bị hệ thống bắt chấm dứt để dành chỗ cho các tiến trình khác.

**Killed hay Shut down:** Khi 1 activity đang paused hay stopped, hệ thống sẽ xóa activity ấy ra khỏi bộ nhớ.



Hình 1.4 Lược đồ vòng đời của một Activity

Dựa vào lược đồ trên, thấy được có 3 vòng lặp quan trọng sau:

**Vòng đời toàn diện (Entire Lifetime):** Diễn ra từ lần gọi onCreate(Bundle) đầu tiên và kéo dài tới lần gọi onDestroy() cuối cùng.

**Vòng đời thấy được (Visible Lifetime):** Diễn ra từ khi gọi onStart() và kéo dài tới khi gọi onStop(). Ở vòng đời này, activity được hiển thị trên màn hinh nhưng có thế không tương tác với người dùng ở trên nền. Các phương thức onStart(0 và onStop() có thể được gọi nhiều lần.

**Vòng đời trên nền (Foreground Lifetime):** Diễn ra từ khi gọi onResume(0 và kéo dài tới khi gọi onPause(). Ở vòng đời này, activity nằm trên mọi activity khác và tương tác được với người dùng. 1 activity có thể liên tục thay đổi giữa 2 trạng thái paused và resumed, chẳng hạn khi thiết bị sleep hay 1 intent mới được đưa tới.

Toàn bộ vòng đời của 1 activity được định nghĩa nhờ các phương thức sau:

**public class Activity extends ApplicationContext {**

**protected void onCreate(Bundle savedInstanceState);**

*//Gọi khi mới tạo activity để setup các view, binding dữ liệu,...*

*//Kèm theo sau luôn là onStart().*

**protected void onRestart();**

*//Gọi sau khi activity bị stopped và trước khi được khởi động lại.*

*//Kèm theo sau luôn là onStart().*

**protected void onStart();**

*//Gọi khi activity hiện lên trước mắt người dùng.*

*//Kèm theo sau là onResume() nếu activity hiện lên nền hay onStop(0 nếu bị ẩn đi.*

**protected void onResume();**

*//Gọi khi activity bắt đầu tương tác với người dùng và đang trên cùng của activity stack.*

*//Kèm theo sau luôn là onPause().*

**protected void onPause();**

*//Gọi khi hệ thống sắp khởi động lại 1 activity khác trướcđó.*

*//kèotheo sau là onresume nếu activity trở lại trên cùng hay onStop() nếu bị ẩn đi.*

**protected void onStop();**

*//Gọi khi activity không còn hiển thị trước người dùng*

*//Kèm theo sau là onRestart() nếu activity hiện lên trở lại hay onDestroy nếu sắp xoá activity đi.*

**protected void onDestroy();**

*//Gọi ngay trước khi kết thúc activity, xảy ra khi hàm finish() được gọi hoặc khi hệ thống yêu cầu buộc phải kết thúc.*

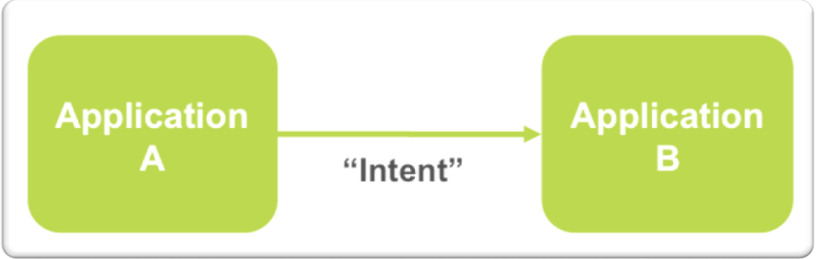
**}**

* **Intent:**

Khi **Tim Berners** phát minh ra giao thức Hypertext Transfer Protocol (HTTP), ông cũng đã phát minh ra một định dạng URLs chuẩn. Định dạng này là một hệ thống các động từ đi kèm các địa chỉ. Địa chỉ sẽ xác định nguồn tài nguyên như Web page, hình ảnh hay các server-side program. Động từ sẽ xác định cần phải làm cái gì với nguồn tài nguyên đó: GET để nhận dữ liệu về, POST để đưa dữ liệu cho nó để thực thi một công việc nào đó. Khái niệm **Intent** cũng tương tự, **Intent** là một mô tả trừu tượng của một hành động được thực thi. Nó đại diện cho một hành động đi kèm với một ngữ cảnh xác định. Với **Intent** thì có nhiều hành động và nhiều component (Một thể hiện của một class java dùng để thực thi các hành động được đặc tả trong **Intent**) dành cho **Intent** của Android hơn là so với HTTP verbs (POST, GET) và nguồn tài nguyên (hình ảnh, web page) của giao thức HTTP, tuy nhiên khái niệm vẫn tương tự nhau.

**Intent** được sử dụng với phương thức **startActivity()** để mở một **Activity**, và dùng với **broadcastIntent** để gởi nó đến bất kì **BroadcastReceiver** liên quan nào, và dùng với [**startService(Intent)**](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29), **bindService(Intent, ServiceConnection, int)** để giao tiếp với các **Service** chạy dưới nền.

Intent cung cấp một chức năng cho phép kết nối hai chương trình khác nhau trong quá trình thực thi (runtime) (Cung cấp khả năng cho phép hai chương trình khác nhau giao tiếp với nhau). Chức năng quan trọng và được sử dụng nhiều nhất của một **Intent**  là mở một **Activity**, nơi mà nó có thểđược dùng như một vật kết nối các Activity lại với nhau (Truyền thông tin giữa hai Activity khác nhau).

****

Sử dụng intent để trao đổi thông tin giữa hai chương trình

Thành phần chính của Intent bao gồm:

**Action:** Xác định hành động sẽ được thực thi, các hành động này có thể là: ACTION\_VIEW, ACTION\_EDIT, ACTION\_MAIN…

**Data:** Các dữ liệu được sử dụng để hành động (Action) thao tác trên nó, như bản ghi về một người nào đó trong cơ sở dữ liệu chứa các contact của thiết bị.

Ví dụ về thành phần action/data:

**ACTION\_VIEW content://contacts/people/1** – hiển thị thông tin liên lạc của người có mã là “1”.

**ACTION\_DIAL content://contacts/people/1** – gọi điện cho người có mã là “1”.

**ACTION\_DIAL tel:123** – gọi đến số điện thoại “123”

Ngoài ra, Intent còn có các thuộc tính phụ sau:

Category: thông tin chi tiết về hành động được thực thi, ví dụ như CATEGORY\_LAUNCHER có nghĩa là nó sẽ xuất hiện trong Launcher như ứng dụng có mức level cao (top-level application),  trong khi **CATEGORY\_ALTERNATIVE** chứa thông tin danh sách các hành động thay thế mà người dùng có thể thực thi trên phần dữ liệu mà **Intent** cung cấp.

**Type:** Chỉ định 1 kiểudữ liệu chính xác (kiểu MIME) được mang bởi intent. Thường thì type được suy ra từ chính dữ liệu. Bằng cách thiết lập thuộc tính type, bạn có thể vô hiệu hóa sự phỏng đoán kiểu dữ liệu và chỉ định rỏ một kiểu chính xác.

**Component:** Chỉ định rõ tên của lớp thành phần (Một thể hiện của một class java dùng để thực thi các hành động được đặc tả trong **Intent**) sử dụng cho Intent . Thông thường thì nó được xác định bằng cách tìm kiếm thông tin trong **Intent** (Các thông tin như Action, data/type, và category) và nối chúngvới các component (Một thể hiện của một Class java dùng để thực thi các hành động được đặc tả trong **Intent**) có thể đáp ứng được các yêu cầu sử lý của **Intent**.

**Extracts:** là một đối tượng **Bundle** dùng để chứa các thông tin kèm theo được dùng để cung cấp thông tin cần thiết cho component. Ví dụ: Nếu có một Intent đặc tả hành động send email thì những thông tin cần chứa trong đối tượng **Bundle** là subject, body…

**Inten Resolution**

Intent có 2 dạng chính sau:

**Explicit Intents**: Xác định rỏ một component (Một thể hiện của một class java dùng để thực thi các hành động được đặc tả trong **Intent**) (thông qua phương thức setComponent(ComponentName) hoặc setClass(Context, Class)) cung cấp lớp sẽ thực thi các hành động được đặc tả trong **Intent**. Thông thường thì những **Intent** này không chứa bất kỳ thông tin nào khác (như category, type) mà đơn giản chỉ là cách để ứng dụng mở các **Activity** khác bên trong một **Activity**.

**Implicit Intents**: Không chỉ định một component nào cả, thay vào đó, chúng sẽ chứa đủ thông tin để hệ thống có thể xác định component có sẵn nào là tốt nhất để thực thi hiệu quả cho **Intent** đó.

Khi sử dụng **Implicit intents**, do tính chất chuyên quyền của loại **Intent** này,ta cần phải biết phải làm gì với nó. Công việc này được đảm nhiệm bởi tiến trình của **Intent resolution**, nó giúp chỉ định **Intent** đến một **Actvity**,

**BroadcastReceiver**, hoặc **Service** (hoặc thỉnh thoảng có thểlà 2 hay nhiều hơn một activity/receiver) để có thể xử lý các hành động được đặc tả trong **Intent**.

Bất cứ thành phần nào (Activity, BroadcastReceiver, Service) khi muốn sử dụng trong ứng dụng đều phải được đăng kí trong file **AndroidManifest.xml**. Trong đó cầnđịnh nghĩa một thẻ **<intent-fillter>** cung cấp các thông tin để hệ thống có thể xác định được cái mà các component này (Activity, BroadcastReceiver, Service) có thể xử lý được (những action mà component này có thể thực hiện được).

**Intent Fillter** là bản đặc tả có cấu trúc của các giá trị của **Intent** dùng để xác định component phù hợp để thực hiệncác hành động được đặc tả trong **Intent**. Một **Intent Fillter** nối các hành động, categories và dữ liệu (hoặc thông qua type hoặc sheme) trong Intent. **Intent Fillter** được khai báo trong **AndroidManifest.xml** và sử dụng thẻ **intent-fillter**.

Một Intent Fillter có các thành phần chính sau:

**Action**: Tên hành động mà component có thể thực thi.

**Type:**Kiểu hành động mà component có thể thực thi.

**Category:** Phân nhóm các hành động.

Đối với những dữ liệu không phải là nội dung cụ thể (VD: URI) thì việc xem xét lựa chọn Intent phù hợp sẽ dựa vào lược đồ(Scheme) của dữ liệu được cung cấp (VD: http:// mailto: …)

* **Luật xác định component phù hợp với intent**

Để xác định một thành phần là phù hợp với một Intent hệ thống sẽ tiến hành xem xét từ trên xuống.

Trước tiên khi một Intent được gọi, Android sẽ tìm kiếm những component (Activity, BroadcastReceiver, Service) có action-name phù hợp với Intent.

Nếu có component phù hợp Android sẽmở component đó lên để thực thi các hành động theo yêu cầu.

Nếu có nhiều hơn 1 component có action-name phù hợp thì Android sẽ yêu cầu người dùng chọn component phù hợp.

Ngược lại nếu không có component nào phù hợp Android sẽ tiến hành xem xét kiểu dự liệu của Intent cung cấp xem có component nào có đủ năng lực để sử lý kiểu dữ liệu đó không. Nếu khôngđược Android sẽ tiến hành xem xét scheme của dữ liệu đó để tìm kiếm component phù hợp. Nếu vẫn không tìm được component phù hợp Android sẽ tiến hành xem xét các component có chung Category với Intent để xác định component.

* **Service :**

Một service là một thành phần của ứng dụng, thể hiện mong muốn ứng  dụng thực hiện các hành động trong khi không tương tác với người dùng hoặc cung cấp chức năng cho các ứng dụng khác sử dụng. Nói một cách đơn giản, service là các tác vụ (task) chạy ngầm dưới hệ thống nhằm thực hiện một nhiệm vụ nào đó.  Mỗi class Service phải chứa thẻ <service> được khai báo trong file **AndroidManifext.xml**. Services có thể được bắt đầu bởi Context.startService() và Context.bindservice()

Cũng như các đối tượng của ứng dụng khác, services chạy trên luồng (Thread) của tiến trình chính. Có nghĩa là nếu service của bạn dự định chạy các hành vi có cường độ lớn hoặc các hành vi có thể gây nghẽn mạch, nó sẽ tự sinh ra luồng (Thread) của chính nó để làm thực hiện các tác vụ được đặc tả trong Service.

Cần lưu ý:

Service không phải là một tiến trình độc lập. Đối tượng Service không chạy trên tiến trình (process) của riêng nó (trừ khi có một yêu cầu đặt biệt), Service chạy trên tiến trình của chương trình.

Service không phải là một luồng (thread).

Service có chứa 2 đặc trưng cơ bản sau:

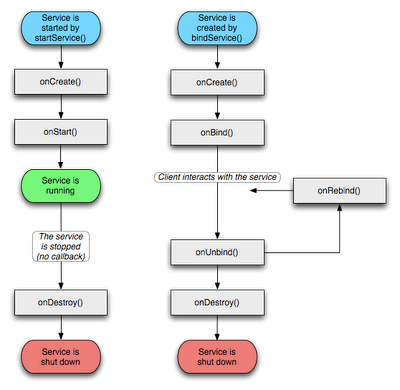
Là một chức năng để ứng dụng thông báo với hệ thống về việc mà nó muốn thực hiện ở phía dưới nền (ngay cả khi người dùng không tương tác trực tiếp tới ứng dụng). Điều này tương ứng với việc gọi phương thức Context.startservice(), nó sẽ yêu cầu hệ thống lên lịch cho service để chạy cho đến khi bị chính service hay người nào đó dừng nó lại.

Là một chức năng cho phép ứng dụng để cung cấp các chức năng của nó cho các ứng dụng khác. Điều đó tương ứng với việc gọi Context.bindService(), cho phép một long-standing connection được tạo ra để service có thể tương tác với nó.

Khi một Service được tạo ra, việc hệ thống cần làm là tạo ra thành phần và gọi hàm onCreate() và bất kỳ lệnh callback thích hợp nào trên luồng chình. Nó phụ thuộc vào Service thi hành hành vi thích hợp nào, chẳng hạn tạo luồng thứ hai để thực hiện tiến trình.

* **Vòng đời của một service**

Có 2 lý do để một service được chạy bởi hệ thống. Bằng cách gọi phương thức Context.startService() và Context.bindService(). Nếu người dùng gọi phương thức Context.startService() thì hệ thống sẽ nhận về một đối tượng **service** (tạo mới nó và gọi hàm onCreate() nếu cần thiết) và sau đó lại gọi phương thức onStartCommand(Intent, int ,int) với các biến được cung cấp bởi client. Service mới được tạo ra này sẽ chạy cho tới khi có lệnh gọi phương thức Context.stopService() hoặc stopSelf(). Dù cho một service có được start bao nhiêu lần đi nữa thì nó sẽ bị tắt ngay khi phương thức Context.stopService() hoặc phương thức stopSelf() được gọi.Service có thể dùng phương thức stopSelf(int) để đảm bảo rằng service sẽ không được tắt cho đến khi Intent đã khởi tạo nó được xử lý.

****

* **Lược đồ vòng đời của một service**

Dựa theo hình trên, 1 service được bắt đầu bởi phương thức startService() là service dùng để thực hiện các tác vụ ngầm dưới nền, service được bắt đầu bởi phương thức bindService() là service được dùng để cung cấp các chức năng cho các chương trình khác.

* **BroadcastReceiver**

Là lớp nhận về các intents được gửi bởi sendBroadcast(). Có thể tạo instance cho lớp này bằng 2 cách: hoặcvới Context.registerReceiver() hay thông qua tag <receiver> trong file AndroidManifest.xml.

Có hai lớp broadcast quan trọng cần xét đến:

Broadcasts bình thường (Normal broadcasts): Được gửi bởi  context.sendBroadcast tới các receivers hoàn toàn không theo 1 trật tự nào, dẫn tới đôi khi tồn tại cùng 1 lúc nhiều receivers khác nhau. Tuy nhanh nhưng không thể trao đổi kết quả nhận về giữa cac1 receivers với nhau.

Broadcasts có thứ tự (Ordered broadcasts) - Được gửi bởi Context.sendOrderedBroadcast() tới từng receiver một. Như vậy, kết quả nhận về của 1 receiver có thể trở thành input của 1 receiver khác, hay có thể ngay lập tức dừng việc broadcast tới các receiver khác. Trật tự chạy các receivers được quyết định bởi thuộc tính android:priority trong intent-filter tương ứng.

* **ContentProvider:**

Là nơi lưu trữ và cung cấp cách truy cập dữ liệu do các ứng dụng tạo nên. Đây là cách duy nhất mà các ứng dụng có thể chia sẻ dữ liệu của nhau.

Android cung cấp sẵn content providers cho 1 số kiểu dữ liệu thông dụng như âm thanh, video, thông tin sổ điện thoại,... (Xem thêm trong gói android.provider). Người lập trình cũng có thể tự tạo ra các class con (subclass) của Content Provider để lưu trữ kiểu dữ liệu của riêng mình.

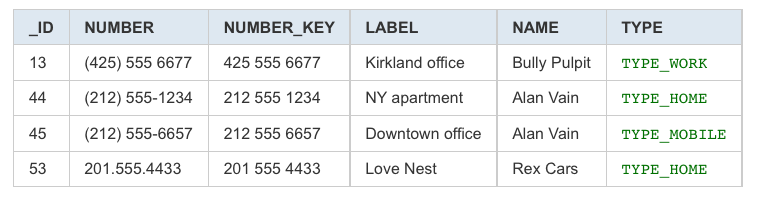
Cách thức hoạt động của Content Provider:

Mọi content provider, tuy có thể khác nhau về cách thức lưu trữ dữ liệu, đều sử dụng chung 1 interface để tìm và trả về dữ liệu, bao gồm việc thêm, bớt và sửa dữ liệu. Việc này được thực hiện thông qua các đối tượng ContentResolver khi gọi getContentResolver()

ContentResolver cr = getContentResolver();

*Mô hình dữ liệu*

Content providers sử dụng dữ liệu dưới dạng 1 table đơn giản, trong đó mỗi hàng là 1 record và mỗi cột là dữ liệu của 1 kiểu nhất định.



Ví dụ mô hình dữ liệu thông tin số điện thoại trong máy

Mỗi record có 1 trường “\_ID” độc nhất để liên kết với các bảng khác.

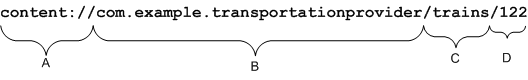
URI (Uniform Resource Identifier)

URI là dãy các ký tự string được dùng để:

Mở 1 trang web trong trình duyệt, bắt đầu với “http://”

Chỉ tới các file tài nguyên có sẵn trong máy (VD: 1 file video, danh sách contacts,...), bắt đầu với "content://"

Dựa vào hình trên



Các thành phần của một URL

A – Tiền tố mặc định của 1 URI, không thay đổi được.

B – Xác định content provider. Đối với các ứng dụng hãng thứ 3, đây chính là tên đầy đủ của 1 lớp.

C – Xác định loại dữ liệu nào cần lấy về. Nếu nguồn cung cấp chỉ cho lấy 1 loại duy nhất thì có thể để trống, hoặc nếu có nhiều loại thì có thể phân ra từng phần 1 (land/bus, land/train, sea/ship,…)

D – ID của 1 record, khi cần lấy về chỉ duy nhất 1 record.

Khi tạo 1 content provider, Android tự động tạo 1 constant cho URI của provider đó để đảm bảo code hiệu quả và dễ cập nhật sau này. Constant này - CONTENT\_URI - sẽ xác định cho ContentProvider phải liên kết với bảng nào của nguồn cung cấp.

Ví dụ:

URI cho bảng liên kết số điện thoại với tên người là:

android.provider.Contacts.Phones.CONTENT\_URI

URI cho bảng liên kết hình đại diện với tên người là:

android.provider.Contacts.Photos.CONTENT\_URI

Tìm kiếm 1 Content Provider

Để bắt đầu tìm kiếm (query) 1 content provider, trước nhất người lập trình cần có trong tay các yếu tố sau:

URI xác định provider.

Tên các trường (field) dữ liệu sẽ lấy về.

Các kiểu dữ liệu của những trường trên.

ID, chỉ khi cần tìm 1 record xác định nào đó.

Để bắt đầu tìm kiếm, người lập trình có thể dùng 1 trong 2 phương thức: ContentResolver.query() hoặc Activity.managedQuery(). Cả 2 phương thức trên đều có chung arguments và đều trả về 1 đối tượng con trỏ (Cursor)

|  |
| --- |
|  |

**public final Cursor query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)**

**public final Cursor managedQuery (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)**

Đầu tiên, dù với query() hay managedQuery() thì ta đều cần cung cấp URI của bên cung cấp, tức CONTENT\_URI. Chú ý nếu muốn giới hạn chỉ tìm kiếm 1 URI duy nhất, ta thêm ID vào cuốiURI như sau: “content://.../23”, với 23 là ID ta cần tím kiếm

Hoặc ta có thể dùng các hàm hỗ trợ như ContentUris.withAppended() và Uri.withAppendedPath().

Ví dụ:

**Uri myPerson = ContentUris.withAppendedId(People.CONTENT\_URI,23);**

Các biến truyền vào khác của query() và managedQuery() là:

projection - Tên của cột dữ liệu cần trả về. Nếu NULL thì trả về tất cả các cột.

selection - Tên các hàng dữ liệu sẽ trả về. Sử dụng với WHERE giống như khi truy vấn SQL. Nếu NULL thì trả về tất cả các hàng.

selectionArgs - quy định giá trị trả về, VD:  selectionArgs = {value\_as\_string} sẽ ép kiểu dữ liệu trả về dưới dạng string.

sortOrder - Xác định thứ tự sắp xếp các hàng. Sử dụng với ORDER BY giống như khi truy vấn SQL. Nếu NULL thì không theo thứ tự gì cả.

Tạo mới 1 Content provider

Để tạo mới 1 content provider, cần phải thoả mãn các điều kiện:

Thiết lập hệ thống lưu trữ dữ liệu như thế nào: bằng các file hay sử dụng cơ sở dữ liệu SQLite.

Extend lớp ContentProvider.

Khai báo 1 content provider trong file AndroidManifest.xml

Để định nghĩa 1 lớp con ContentProvider, ta cần hiện thực 6 phương thức trừu tượng (abstract) được khai báo trong lớp ContentProvider:

query()

insert()

update()

delete()

getType()

onCreate()

Phương thức query() phải trả về 1 đối tượng Cursor. Mặc dù Cursor về cơ bản là 1 interface, Android cung cấp sẵn 1 số đối tượng Cursor tạo sẵn. Ví dụ; SQLiteCursor dành cho cơ sở dữ  liệu SQLite, MatrixCursor cho dữ liệu không lưu trong SQLite.

Android cũng khuyên nên gọi ContentResover.notifyChange() để báo cho các listeners biếtdữ liệu sẽ được hiệu chỉnh. Ngoài ra người lập trình cũng nên thêm vài thao tác sau để giúp việc truy xuất các lớp hiệu quả hơn:

Định nghĩa một Uri dạng public static final, với tên là CONTENT\_URI (như đã trình bày ở trên).

Định nghĩa tên các cột mà content provider sẽ trả về. Thôg thường thì đây cũng chính là tên các cột trong cơ sở dữ liệu SQLite mà chúng đại diện. Tiếp theo, người lập trình nên định nghĩa các public static String constants để cac1 ứng dụng khác có thể ghi rõ tên các cột và các hướng dẫn khác 1 cột interger tên "\_id" (với constant \_ID) cũng cần được định nghia để sử dụng khi cần lấy về các record độc nhất. Trong SQLite, trường \_ID nên có kiểu như sau:INTEGER PRIMARY KEY AUTOINCREMENT, trong đó thành phần AUTOINCREMENT không nhất thiết phải có, nhưng nên được đưa vào để đảm bảo các ID luôn độc nhất, tránh trường hợp khi xóa 1 row thì ID của row tiếp theo sẽ được cấp ID giống hệ của row vừa bị xóa.

Các ghi chú giúp người khác ghi đọc biết đọc dữ liệu như thế nào.

Nếu content provider này xử lý 1 kiểu dữ liệu hoàn toàn mới, người lập trình cần định nghĩ 1 kiểu MIME mới để trả về trong sự thể hiện của ContentProvider.getType().

Để khai báo 1 Content Provider mới, trước nhất cần phải khai báo trong tag <provider> của file AndroidManifest.xml.

Trong đó, thuộc tính name là tên đầy đủ của lớp con ContentProvider, và authorities là phần xác định provider của URI "content:"

* **View:**

View là thành phần cơ bản để xây dựng giao diện người dùng cho 1 ứng dụng Android. View là 1 lớp căn bản của widgets (widgets được dùng để tạo các nút nhấn, text fields,...).

Lớp con ViewGroup là lớp căn bản của layouts, có thể coi như 1 cái hộp vô hình chứa nhiều Views hay ViewGroups khác và xác định các thuộc tính layout.

Sử dụng Views:

Trong 1 cửa sổ màn hình, mọi views đều được sắp xếp theo thứ tự trên 1 cây trong 1 hay nhiều file XML nằm trong res/layout.

Khi tạo 1 cây chứa các views, người lập trình cần xác định những giá trị sau:

Các thuộc tính - Chẳng hạn định xem 1 TextView sẽ hiển thị chữ gì. Mỗi view có những thuộc tính và phương thức khác nhau.

Mức độ tập trung (Focus) - Thay đổi sự tập trung tuỳ theo input của người dùng với requestFocus().

Listeners - Định các listeners để có hành động khi 1 sự kiện gì đó xảy ra với setOnFocusChangeListener(View.OnFocusChangeListener).

Một số Views thường dùng:

**TextView:** Hiển thị 1 khung text và cho phép người dùng thay đổi.

**ImageView:** Hiển thị 1 hình ảnh xác định từ file tài nguyên hay qua 1 content provider.

**Button:** Hiển thị 1 nút nhấn.

**Checkbox:** Hiển thị 1 nút nhấn với 2 trạng thái khác nhau.

**KeyboardView:** Hiển thị bàn phím ảo để nhập liệu.

**WebView:** Hiển thị các trang web bằng Webkit.

Một số ViewGroups thường dùng:

**LinearLayout:** Các view được xếp theo 1 hàng hay 1 cột duy nhất.

**AbsoluteLayout:** Cho phép xác định chính xác vị trí của từng view.

**TableLayout:** Sắp xếp các view theo các cột và hàng. Mỗi hàng có thể có nhiều view, mà mỗi view sẽ tạo nên 1 ô.

**RelativeLayout:** Cho phép xác định vị trí các view theo mối quan hệ giữa chúng (VD; canh trái, phải,...).

**FrameLayout:** Là 1 placeholder cho phép đặt lên đó 1 view duy nhất. View đặt lên FrameLayout luôn được canh lề trái phía trên.

**ScrollView:** Là 1 FrameLayout đặc biệt cho phép trượt (scroll) 1 danh sách dài hơn kích thước màn hình.

* **Lưu trữ dữ liệu**

Android có nhiều cách để lưu trữ dữ liệu. Một cách đơn giản nhất là sử dụng hệ thống preferences. Android cho phép các activity và ứng dụng lưu giữ preferences theo dạng cặp từ khóa và giá trị, nằm giữa các lần gọi một activity. Mục đích chính là để lưu trữ thông tin cấu hình chi tiết mà người dùng hiệu chỉnh, chẳng hạn trang web người dùng xem gần đây nhất. Bạn có thể lưu trữ bất kì thông tin nào trong preferences bằng từ khóa có kiểu String và kiểu dữ liệu nguyên thủy (boolean, String…)

Preferences có thể dùng cho activity độc lập hay được chia sẻ giữa các activity của một ứng dụng. Và tương lai, Preference có thể được chia sẻ giữa các ứng dụng với nhau.

Bên cạnh việc sử dụng Preferences để lưu trữ dữ liệu thì cũng còn một cách đơn giản khác để thực hiện công việc lưu trữ dữ liệu là sử dụng tập tin. Android cung cấp 2 mô hình để truy cập tập tin:

Một cho các tập tin được đóng gói sẵn cho ứng dụng.

Một dành cho tập tin được tạo trên thiết bị thực thi ứng dụng đó.

Đối với dữ liệu tĩnh, chẳng hạn danh sách các từ dùng để kiểm tra chính tả. Cách đơn giản nhất để thực hiện điều đó là lưu vào một tập tin, và tập tin này phải được đặt trong thư mục res/raw của project. Nhờ vậy, tập tin sẽ được đặt trong tập tin APK của ứng dụng như một nguồn tài nguên cơ bản.

Ngoài 2 phương pháp trên,  Android còn hỗ trợ hệ quản trị cơ sở dữ liệu quan hệ SQLite để quản lý dữ liệu lớn và có quan hệ phức tạp. Nhờ vậy, người lập trình có thể thực hiện quản lý dữ liệu một cách dễ dàng, linh hoạt hơn.

* **SQLite**

Khi lập trình trên di động hay các thiết bị có dung lượng bộ nhớ hạn chế, người ta thường dùng SQLite. SQLite là một hệ quản trị cơ sở dữ liệu nhúng được hiện thực từ chuẩn SQL-92. Giống với cái tên của nó, SQLite chiếm dung lượng nhỏ (khoảng 275KB) nên việc truy xuất dữ liệu được nhanh chóng, không chiếm dụng quá nhiều tài nguyên hệ thống. Do SQLIte là phần mềm mã nguồn mở nên nó không bị giới hạn tác quyền. Vì lý do đó mà SQLite được nhiều hãng sử dụng (Adobe, Apple, Google, Sun, Symbian)  và các dự án mã nguồn mở (Mozila, PHP, Python).

Đặc biệt, đối với Android, SQLite rất thích hợp để tạo cơ sở dữ liệu cho các ứng dụng trên nền Android. Từ khi SQLite có giao diện SQL thì nó càng được sử dụng dễ dàng và thông dụng hơn.

Sqlite tuy nhẹ hơn so với các hệ cơ sở dữ liệu khác nhưng cũng không khác biệt nhiều. SQLite cũng sử dụng ngôn ngữ truy vấn SQL (SELECT, INSERT, DELETE…), thích hợp cho các ứng dụng trên điện thoại, đặc biệt là ứng dụng trên hệ điều hành Android vì Android Runtime có thể chứa cả SQLite mà không cần phải chia nhỏ ra.

SQLite có một hệ thống câulệnhSQL đầy đủ với các triggers, transactions... Các câu truy vấn cũng như các hệ cơ sở dữ liệu khác. SQLite như bản thu nhỏ của so với các hệ CSDL khác, vì vậy nó không thể có đầy đủ các chứ năng trên chiếc điện thoại di động của bạn.

SQLite là một lựa chọn thích hợp dành cho ứng dụng trên hệ điều hành Android. Ngoài dung lượng lưu trữ nhỏ gọn, SQLite còn cho phép sử dụng Unicode, kiểu dữ liệu không được cài đặt trong một số phiên bản Android.

* **Đặc trưng của SQLite**

SQLite được hiện thực từ tiêu chuẩn SQL-92 của một ngôn ngữ SQL nhưng vẫn còn chứa một số khiếm khuyết.

Tuy SQLite hỗ trợ triggers nhưng bạn không thể viết trigger cho view. Hoặc SQLite không hỗ trợ lệnh ALTER TABLE, do đó, bạn không thể thực hiện chỉnh sửa hoặc xóa cột trong bảng.

SQLite không hỗ trợ ràng buộc khóa ngoại , các transactions lồng nhau, phép kết right outer join, full outer join.

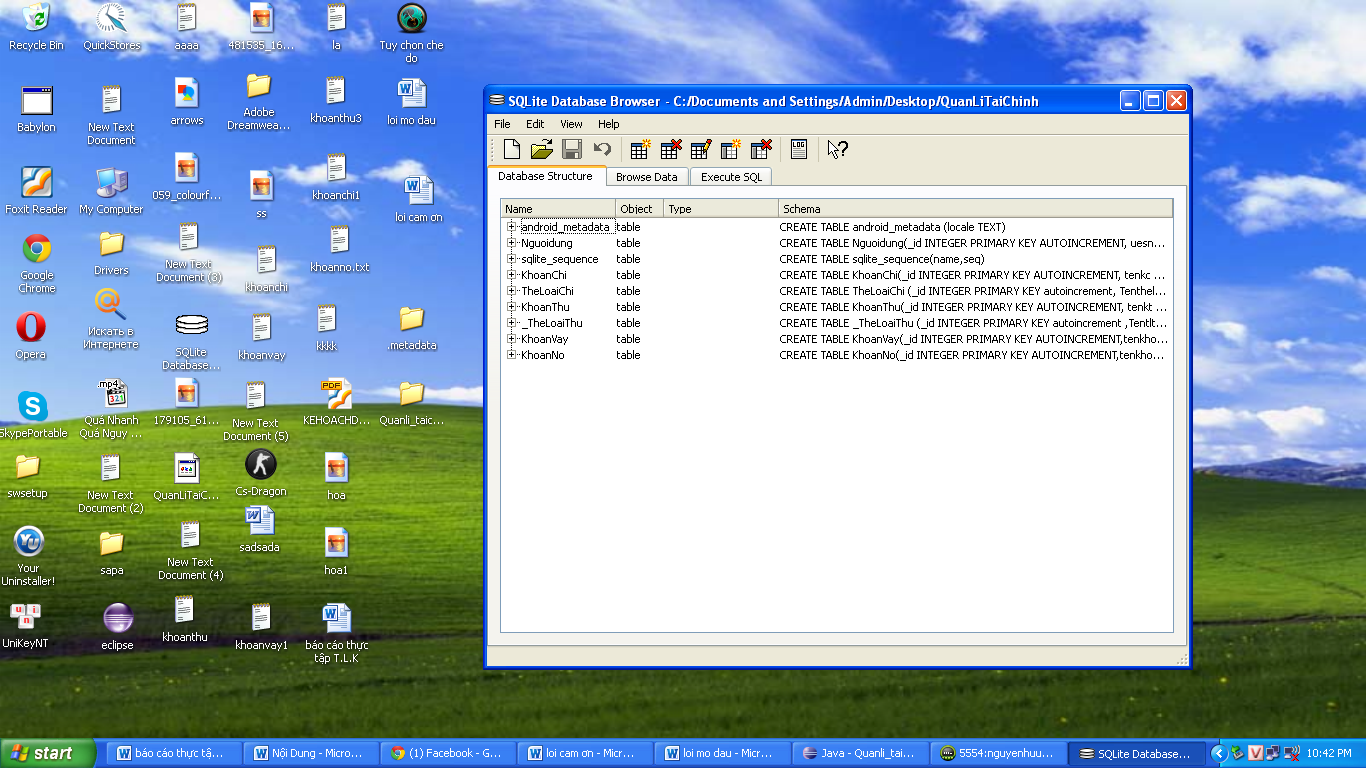
SQLite sử dụng kiểu dữ liệu khác biệt so với hệ quản trị cơ sở dữ liệu tương ứng. Bạn có thể insert dữ liệu kiểu string vào cột kiểu integer mà không gặp phải bất kỳ lỗi nào.

Vài tiến trình hoặc luồng có thể truy cập tới cùng một cơ sở dữ liệu. Việc đọc dữ liệu có thể chạy song song, còn việc ghi dữ liệu thì không được phép chạy đồng thời.

Ngoài các khiếm khuyết trên thì Sqlite cung cấp cho người dùng gần như đầy đủ các chưng năng mà một hệ cơ sở dữ liệu cần có như tạo database; tạo bảng; thêm, xóa, sửa dữ liệu.

* ***SQLite Manager***

SQLite Manager là công cụ hiệu quả hỗ trợ cho việc xây dựng cơ sở dữ liệu cho ứng dụng Android đó là SQLite Manager, một add-on miễn phí của Firefox (<https://addons.mozilla.org/en-US/firefox/addon/5817>)



Hình 1.5: Màn hình làm việc của addon SQLite Browser

Sử dụng SQLite Manager để quản lí cơ sở dữ liệu Android

Tạo database mới :

Vào File Database/ New Database

Tạo bảng mới:

Tại Database vừa tạo, nhấp chuột phải chọn Create table. Thiết lập các giá trị thích hợp.

Xử lý dữ liệu:

Chọn bảng cần nhập và chọn tab Browse & Search

Nút Add: Thêm dữ liệu

Nút Duplicate: Nhân đôi dữ liệu

Nút Edit: Chỉnh sửa dữ liệu

Nút Delete: xóa dữ liệu

Thực thi câu truy vấn:

Chọn tab Execute SQL, Nhập câu lệnh SQL vào khung Enter SQL và chọn Run SQL.

Kết quả sẽ được trình bày ở khung phía dưới. Nếu câu SQL có lỗi sẽ được hiển thị tại ô Last Error.

Tùy chỉnh Database: Chọn tab DB Settings và thiết lập các giá trị như mong muốn

* **Import Database vào ứng dụng Android**

Sau khi tạo CSDL bằng SQLite Manager,chọn menu File/Export Database. Lưu file .sql vừa tạo vào máy.

Để import database vào ứng dụng, người lập trình cần :

Build and Run chương trình Android.

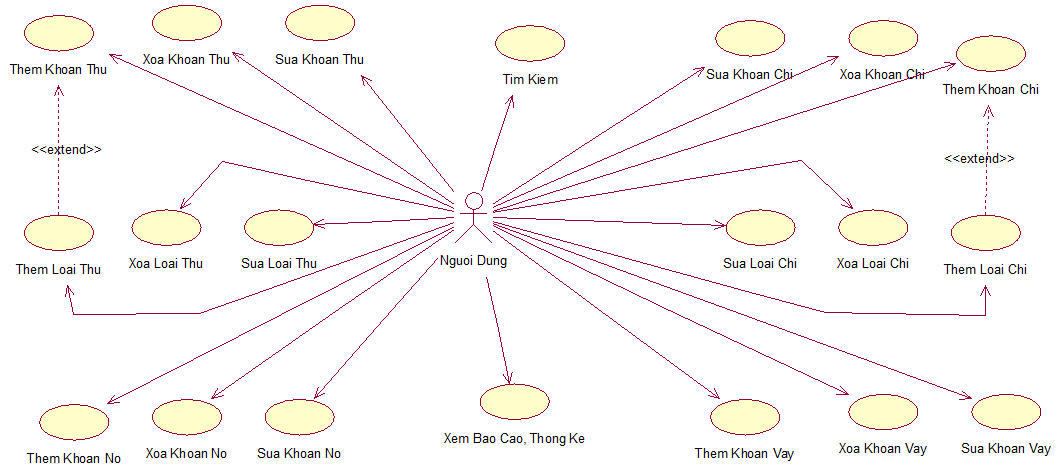
Khởi động máy ảo xong, chọn nút DDMS ở góc trên bên phải chương trình Eclipse để xem cấu trúc file.

Vào thư mục chứa database có dường dẫn như sau :  
/data/data/your.app.package/databases/your-db-name

Sau đó chọn nút import và chọn tập tin database.

## CHƯƠNG 2: MÔ TẢ SƠ ĐỒ USECASE

## 2.1. Sơ đồ Use-case

**

*Hình 2.1 Sơ đồ Use-case*

**2.2. Danh sách các Actor**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Actor** | **Ý nghĩa / Ghi chú** |
| 1 | **Người dùng** |  |

## 2.3. Danh sách các Use-case

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Use-case** | **Ý nghĩa / Ghi chú** |
| 1 | **Thêm khoản thu** | Thêm khoản thu cần nhập |
| 2 | **Xóa khoản thu** | Xóa khoản thu |
| 3 | **Sửa khoản thu** | Cập nhập khoản thu đã có |
| 4 | **Thêm khoản chi** | Thêm khoản chi |
| 5 | **Xóa khoản chi** | Xóa khoản chi |
| 6 | **Sửa khoản chi** | Cập nhập khoản chi đã có |
| 7 | **Thêm khoản nợ** | Thêm khoản nợ mà người dùng cho nợ |
| 8 | **Xóa khoản nợ** | Xóa khoản nợ |
| 9 | **Sửa khoản nợ** | Cập nhật khoản nợ |
| 10 | **Tìm kiếm** | Tìm kiếm khoản thu, khoản chi |
| 11 | **Thêm loại chi** | Thêm loại chi |
| 12 | **Xóa loại chi** | Xóa loại chi |
| 13 | **Sửa loại chi** | Cập nhật loại chi đã có |
| 14 | **Thêm loại thu** | Thêm loại thu |
| 15 | **Xóa loại thu** | Xóa loại thu |
| 16 | **Sửa loại thu** | Cập nhật loại thu đã có |
| 17 | **Thêm khoản vay** | Thêm khoản vay mà người dùng vay |
| 18 | **Xóa khoản vay** | Xóa khoản vay |
| 19 | **Sửa khoản vay** | Cập nhật khoản vay đã có |
| 20 | **Xem báo cáo, thống kê** | Xem báo cáo, thống kê khoản thu, chi theo ngày, tuần, tháng, năm |

## 

## 2.4. Đặc tả Use-case

### 2.4.1. Use-case Thêm khoản thu

* **Tóm tắt :**

Usecase là usecase để người dùng nhập vào các khoản thu nhập.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản thu.
2. Giao diện sẽ chuyển sang khoản thu. Người dùng nhập tên khoản thu, số tiền, ngày thu, loại thu, ghi chú (nếu có) rồi nhấn nút thêm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi thêm khoản thu, số tiền sẽ cộng thêm vào thu nhập và cập nhật tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.2. Use-case Xóa khoản thu

* **Tóm tắt :**

Usecase Xóa khoản thu là usecase để người dùng xóa các khoản thu nhập đã có.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản thu.
2. Giao diện sẽ chuyển sang bảng thu. Người dùng chọn khoản thu và nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi xóa khoản thu, số tiền sẽ được xóa và cập nhật tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.3. Use-case Sửa khoản thu

* **Tóm tắt :**

Usecase Sửa khoản thu là usecase để người dùng sửa các khoản thu nhập đã có.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản thu.
2. Giao diện sẽ chuyển sang bảng thu. Người dùng chọn vào khoản thu để sửa lại và nhấn sửa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi sửa khoản thu, khoản thu sẽ được sửa lại và cập nhật tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.4. Use-case Thêm khoản chi

* **Tóm tắt :**

Usecase Thêm khoản chi là usecase để người dùng nhập vào các khoản chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản chi.
2. Giao diện sẽ chuyển sang bảng chi. Người dùng nhập tên khoản chi, số tiền, ngày chi, loại chi, ghi chú (nếu có) rồi nhấn thêm .

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi thêm khoản chi, số tiền sẽ trừ vào tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.5. Use-case Xóa khoản chi

* **Tóm tắt :**

Usecase Xóa khoản chi là usecase để người dùng xóa các khoản chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản chi.
2. Giao diện sẽ chuyển sang bảng chi. Người dùng chọn khoản chi và nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi thêm khoản chi, số tiền sẽ cộng thêm vào thu nhập và cập nhật tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.6. Use-case Sửa khoản chi

* **Tóm tắt :**

Usecase Sửa khoản chi là usecase để người dùng sửa các khoản chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản chi.
2. Giao diện sẽ chuyển sang bảng chi. Người dùng chọn khoản chi đã nhập, sửa lại khoản chi đó rồi nhấn sửa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi sửa khoản chi, khoản chi sẽ được sửa lại và cập nhật tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.7. Use-case Thêm khoản nợ

* **Tóm tắt :**

Usecase Thêm khoản nợ là usecase để người dùng nhập vào các khoản nợ.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản nợ.
2. Giao diện sẽ chuyển sang bảng nợ. Người dùng nhập tên khoản nợ, số tiền, lãi suất, ngày cho nợ, ngày thu nợ rồi nhấn thêm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi thêm khoản nợ, số tiền sẽ trừ vào tổng tiền.

* **Điểm mở rộng :**

Không có.

### 2.4.8. Use-case Xóa khoản nợ

* **Tóm tắt :**

Usecase Xóa khoản nợ là usecase để người dùng xóa các khoản nợ.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản nợ.
2. Giao diện sẽ chuyển sang bảng nợ. Người dùng chọn khoản nợ và nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi xóa nợ, số tiền nợ được chọn xóa.

* **Điểm mở rộng :**

Không có.

### 2.4.9. Use-case Sửa khoản nợ

* **Tóm tắt :**

Usecase Sửa khoản nợ là usecase để người dùng sửa các khoản nợ.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Khoản nợ.
2. Giao diện sẽ chuyển sang bảng nợ. Người dùng chọn khoản nợ sau đó sửa lại khoản nợ rồi nhấn sửa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi sửa khoản nợ, số tiền sẽ sửa lại.

* **Điểm mở rộng :**

Không có.

### 2.4.10. Use-case Tìm kiếm

* **Tóm tắt :**

Usecase Tìm kiếm là usecase để người dùng tìm kiếm cái khoản thu, chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào tìm kiếm.
2. Giao diện sẽ chuyển sang bảng tìm kiếm. Người dùng nhập tên khoản thu, chi muốn tìm kiếm rồi nhấn tìm kiếm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Hệ thống hiện những thông người dùng muốn tìm kiếm.

* **Điểm mở rộng :**

Không có.

### 2.4.11. Use-case Thêm loại chi

* **Tóm tắt :**

Usecase Thêm loại chi là usecase để người dùng thêm vào các loại chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm loại chi.
2. Giao diện sẽ chuyển sang bảng loại chi. Người dùng nhập loại chi muốn thêm rồi nhấn thêm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi nhập loại chi thì khoản chi sẽ xuất hiện loại mới.

* **Điểm mở rộng :**

Không có.

### 2.4.12. Use-case Sửa loại chi

* **Tóm tắt :**

Usecase Sửa loại chi là usecase để người dùng sửa các loại chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào loại chi.
2. Giao diện sẽ chuyển sang bảng loại chi. Người dùng chọn vào loại chi, sửa lại rồi nhấn sửa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Loại chi sẽ được sửa sau khi thực hiện usecase.

* **Điểm mở rộng :**

Không có.

### 2.4.13. Use-case Xóa loại chi

* **Tóm tắt :**

Usecase Xóa loại chi là usecase để người dùng xóa các loại chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào loại chi.
2. Giao diện sẽ chuyển sang bảng loại chi. Người dùng chọn vào loại chi, rồi nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Loại chi sẽ được xóa sau khi thực hiện usecase.

* **Điểm mở rộng :**

Không có.

### 2.4.14. Use-case Thêm loại thu

* **Tóm tắt :**

Usecase Thêm loại thu là usecase để người dùng thêm các loại thu.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào loại thu.
2. Giao diện sẽ chuyển sang bảng loại thu. Người dùng nhập thêm loại chi, rồi nhấn thêm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Loại thu sẽ được thêm sau khi thực hiện usecase.

* **Điểm mở rộng :**

### 2.4.15. Use-case Xóa loại thu

* **Tóm tắt :**

Usecase Xóa loại thu là usecase để người dùng thêm các loại thu.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào loại thu.
2. Giao diện sẽ chuyển sang bảng loại thu. Người dùng chọn loại thu muốn xóa rồi nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Loại thu sẽ được xóa sau khi thực hiện usecase.

* **Điểm mở rộng :**

Không có.

### 2.4.16. Use-case Sửa loại thu

* **Tóm tắt :**

Usecase Sửa loại thu là usecase để người dùng thêm các loại thu.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào loại thu.
2. Giao diện sẽ chuyển sang bảng loại thu. Người dùng chọn loại thu, nhấn sửa rồi sửa lại.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Loại thu sẽ được sửa sau khi thực hiện usecase.

* **Điểm mở rộng :**

Không có.

### 2.4.17. Use-case Thêm khoản vay

* **Tóm tắt :**

Usecase Thêm khoản vay là usecase để người dùng nhập vào các khoản vay.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản vay.
2. Giao diện sẽ chuyển sang bảng vay. Người dùng nhập tên khoản vay, số tiền, lãi suất, ngày vay, ngày trả rồi nhấn thêm.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi thêm khoản vay, số tiền sẽ cộng vào thu nhập.

* **Điểm mở rộng :**

Không có.

### 2.4.18. Use-case Sửa khoản vay

* **Tóm tắt :**

Usecase Sửa khoản vay là usecase để người dùng sửa các khoản vay.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào thêm khoản vay.
2. Giao diện sẽ chuyển sang bảng vay. Người dùng chọn khoản vay sau đó sửa lại khoản vay đã chọn rồi nhấn sửa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi sửa khoản vay, số tiền sẽ sửa lại.

* **Điểm mở rộng :**

Không có.

### 2.4.19. Use-case Xóa khoản vay

* **Tóm tắt :**

Usecase Xóa khoản vay là usecase để người dùng xóa các khoản vay.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào Thêm khoản vay.
2. Giao diện sẽ chuyển sang bảng vay. Người dùng chọn khoản vay và nhấn xóa.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

* **Trạng thái hệ thống sau khi thực hiện Use-case :**

Sau khi xóa vay, số tiền vay được chọn xóa sẽ mất.

* **Điểm mở rộng :**
* Không có.

### 2.4.20. Use-case Thống kê

* **Tóm tắt :**

Usecase Thống kê là usecase để người dùng thông kê cái khoản thu, chi.

* **Các dòng sự kiện chính :**

1. Ở giao diện chính người dùng nhấp vào thống kê.
2. Giao diện sẽ chuyển sang bảng thống kê. Người dùng có thể chọn cách hiển thị thống kê theo dữ liệu hay theo biểu đồ.

* **Các dòng sự kiện khác :**

Không có.

* **Các yêu cầu đặc biệt :**

Không có.

* **Trạng thái hệ thống khi bắt đầu thực hiện Use-case :**

Không có.

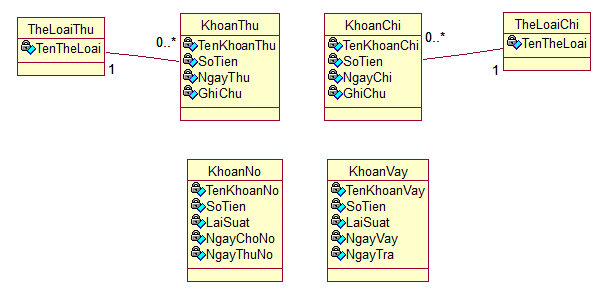
* **Điểm mở rộng :**

Không có.

## CHƯƠNG 3: SƠ ĐỒ PHÂN TÍCH

## 3.1. Sơ đồ lớp

### 3.1.1. Sơ đồ lớp

**

*Hình 3.1 Sơ đồ lớp*

### 3.1.2. Các lớp đối tượng và quan hệ

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên lớp / quan hệ** | **Loại** | **Ý nghĩa / Ghi chú** |
| 1 | KhoanThu | Bình thường | Lớp khoản thu |
| 2 | KhoanChi | Bình thường | Lớp khoản chi |
| 3 | TheLoaiThu | Bình thường | Lớp thể loại thu |
| 4 | TheLoaiChi | Bình thường | Lớp thể loại chi |
| 5 | KhoanNo | Bình thường | Lớp khoản nợ |
| 6 | KhoanVay | Bình thường | Lớp khoản vay |

*Bảng 3.1 Sơ đồ các lớp đối tượng và quan hệ*

### 3.1.3. Mô tả chi tiết các lớp đối tượng

* Lớp KhoanThu (Khoản Thu)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenKhoanThu | private |  | Tên khoản thu |
| 2 | SoTien | private |  | Số tiền |
| 3 | NgayThu | private |  | Ngày thu |
| 4 | GhiChu | private |  | Ghi chú |

*Bảng 3.2 Lớp khoản thu*

* Lớp KhoanChi (Khoản Chi)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenKhoanChi | private |  | Tên khoản chi |
| 2 | SoTien | private |  | Số tiền |
| 3 | NgayChi | private |  | Ngày chi |
| 4 | GhiChu | private |  | Ghi chú |

*Bảng 3.3 Lớp khoản chi*

* Lớp TheLoaiThu (Thể Loại Thu)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenTheLoai | private |  | Tên thể loại |

*Bảng 3.4 Lớp thể loại thu*

* Lớp TheLoaiChi (Thể Loại Chi)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenTheLoai | private |  | Tên thể loại |

*Bảng 3.5 Lớp thể loại chi*

* Lớp KhoanNo (Khoản Nợ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenKhoanNo | private |  | Tên khoản nợ |
| 2 | SoTien | private |  | Số tiền |
| 3 | LaiSuat | private |  | Lãi suất |
| 4 | NgayChoNo | private |  | Ngày cho nợ |
| 5 | NgayThuNo | private |  | Ngày thu nợ |

*Bảng 3.6 Lớp khoản nợ*

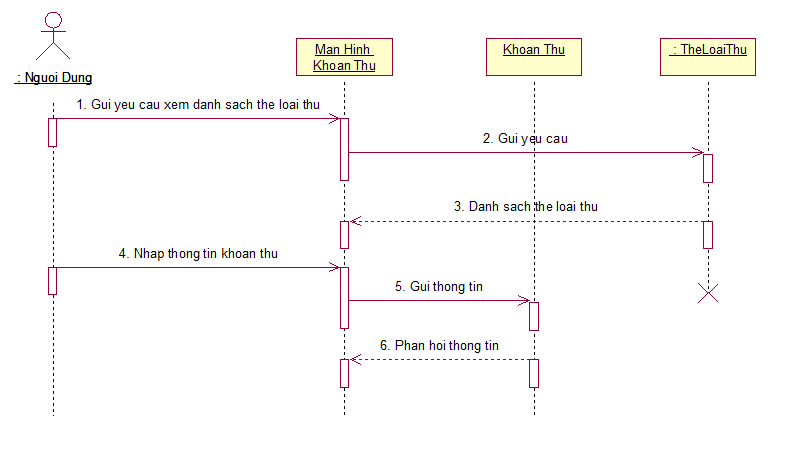
* Lớp KhoanVay (Khoản Vay)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Loại** | **Ràng buộc** | **Ý nghĩa / Ghi chú** |
| 1 | TenKhoanVay | private |  | Tên khoản vay |
| 2 | SoTien | private |  | Số tiền |
| 3 | LaiSuat | private |  | Lãi suất |
| 4 | NgayVay | private |  | Ngày vay |
| 5 | NgayTra | private |  | Ngày trả |

*Bảng 3.7 Lớp khoản vay*

## . Sơ đồ tuần tự

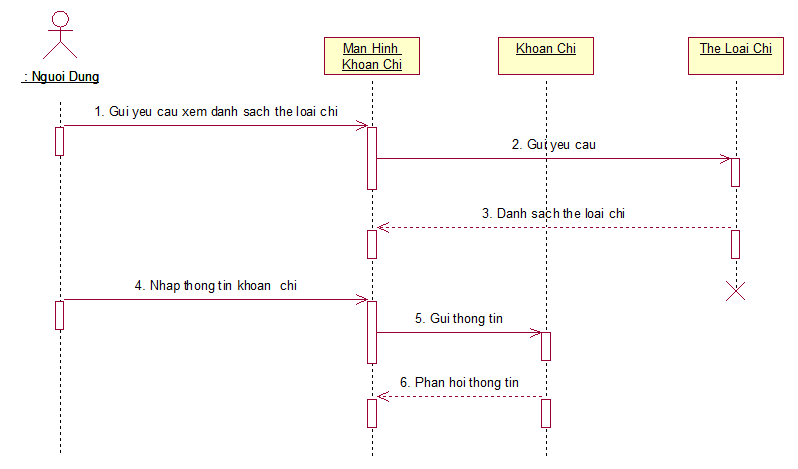
* + 1. **Thêm khoản thu**



*Hình 3.2 Sơ đồ tuần tự thêm khoản thu*

Mô tả các bước:

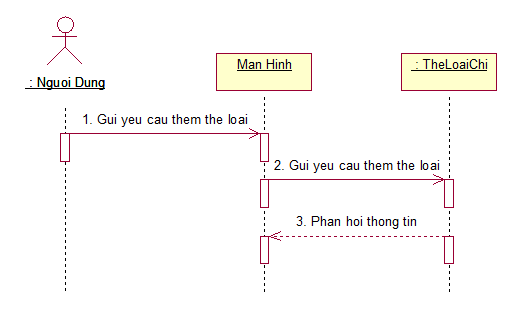
1. Gửi yêu cầu xem danh sách thể loại thu.
2. Gửi yêu cầu xem thông tin.
3. Trả về danh sách thể loại thu.
4. Nhập thông tin khoản thu.
5. Gửi thông tin khoản thu.
6. Phản hồi kết quả.
   * 1. **Thêm khoản chi**



*Hình 3.3 Sơ đồ tuần tự thêm khoản chi*

Mô tả các bước:

1. Gửi yêu cầu xem danh sách thể loại chi.
2. Gửi yêu cầu xem thông tin.
3. Trả về danh sách thể loại chi.
4. Nhập thông tin khoản chi.
5. Gửi thông tin khoản chi.
6. Phản hồi kết quả.
   * 1. **Thêm thể loại chi**



*Hình 3.4 Sơ đồ tuần tự thêm thể loại chi*

Mô tả các bước:

1. Gửi yêu cầu thêm thể loại chi.
2. Gửi yêu cầu.

Phản hồi kết quả.

## CHƯƠNG 4: THIẾT KẾ GIAO DIỆN

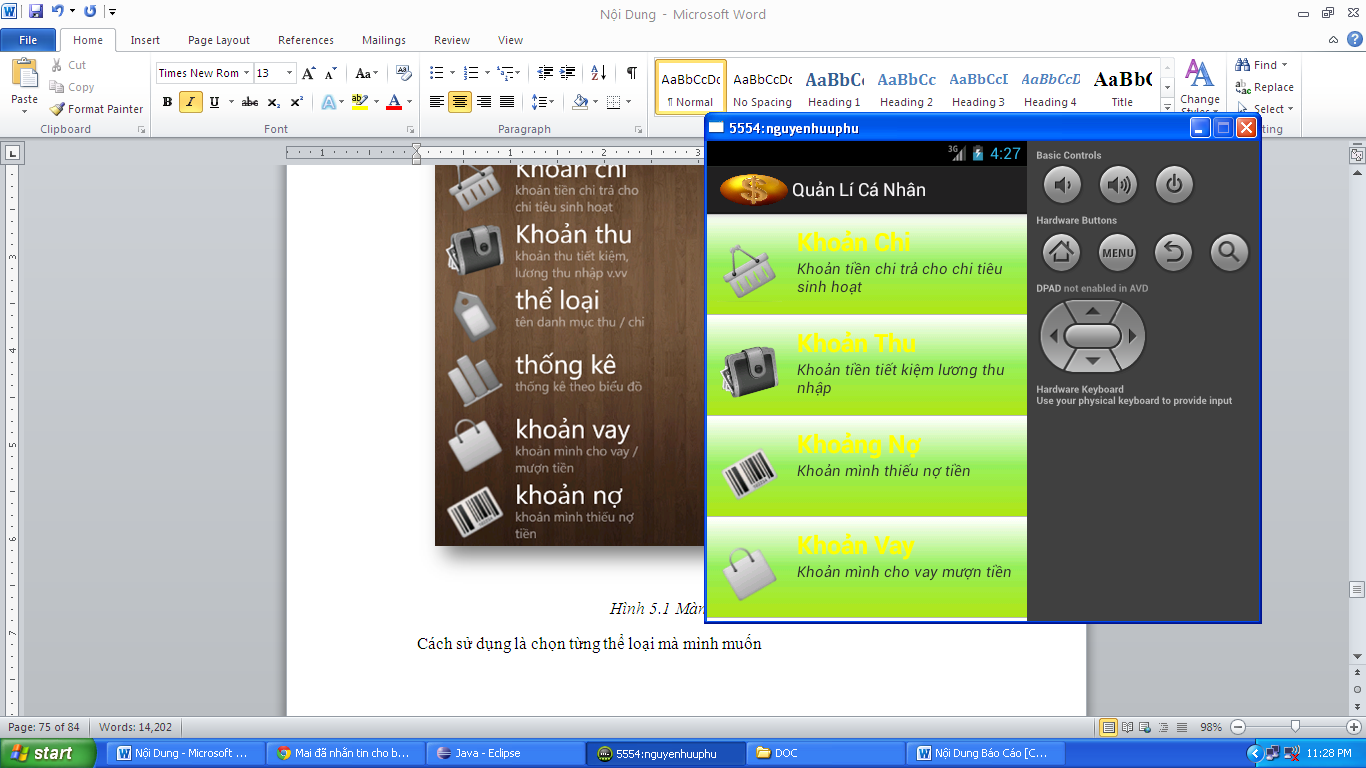
## 4.1. Danh sách các màn hình

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên màn hình** | **Ý nghĩa / Ghi chú** |
| 1 | **Màn hình chính** | **menu** |
| 2 | **Thêm khoản chi** | **Thêm khoản chi tiêu** |
| 3 | **Danh sách khoản chi** | **Danh sách khoản chi** |
| 4 | **Sửa, xóa khoản chi** | **Sửa, xóa trong danh sách khoản chi** |
| 5 | **Thêm khoản thu** | **Thêm khoản thu** |
| 6 | **Danh sách khoản thu** | **Danh sách khoản thu** |
| 7 | **Sửa, xóa khoản thu** | **Cập nhật khoản thu, xóa trong danh sách.** |
| 8 | **Thể loại** | **Liệt kê lựa chọn thể loại thu / chi** |
| 9 | **Thêm thể loại chi** | **Thêm tên thể loại chi** |
| 10 | **Sửa, xóa thể loại chi** | **Cập nhật, xóa thể loại chi** |
| 11 | **Thêm thể loại thu** | **Thêm tên thể loại thu** |
| 12 | **Sửa, xóa thể loại thu** | **Cập nhật, xóa thể loại thu** |
| 13 | **Thêm khoản vay** | **Thêm khoản cho vay / mượn** |
| 14 | **Thong tin khoản vay** | **Danh sách khoản cho vay** |
| 15 | **Sửa, xóa khoản vay** | **Sửa, xóa khoản vay trong danh sách** |
| 16 | **Thêm khoản nợ** | **Thêm khoản nợ** |
| 17 | **Thông tin khoản nợ** | **Danh sách khoản nợ tiền** |
| 18 | **Sửa, xóa khoản nợ** | **Sửa, xóa khoản nợ trong danh sách** |
| 19 | **Thống kê thu/chi** | **Thống kê khoản thu chi theo biểu đồ.** |
| 20 | **Thống kê vay/nợ** | **Thống kê khoản vay nợ theo biểu đồ.** |
| 21 |  |  |
| 22 |  |  |
|  |  |  |

*Bảng 4.1 Danh sách các màn hình*

## 4.2. Mô tả chi tiết mỗi màn hình

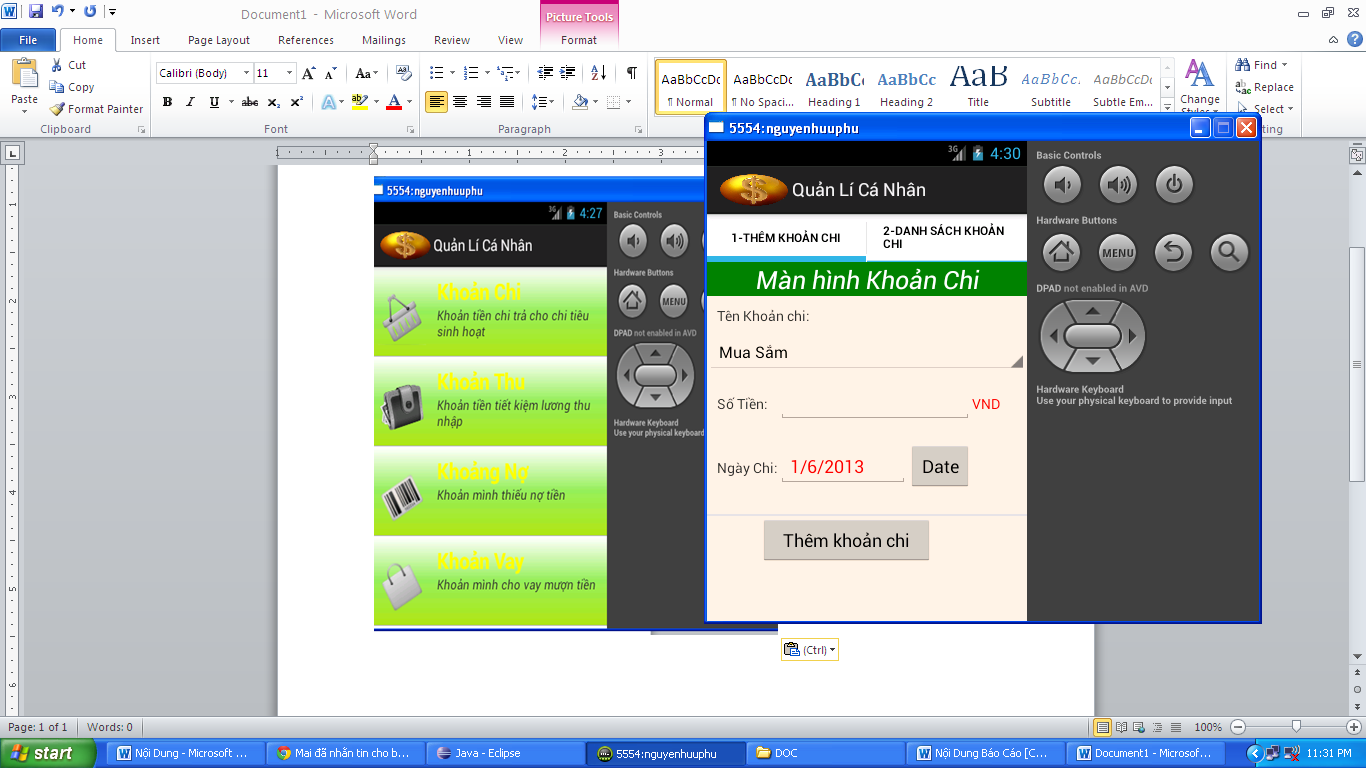
### 4.2.1. Màn hình “Màn hình khởi động”



*Hình 4.1 Màn hình khởi động*

Cách sử dụng là chọn từng thể loại mà mình muốn

4.2.2. Màn hình “Thêm khoản chi”



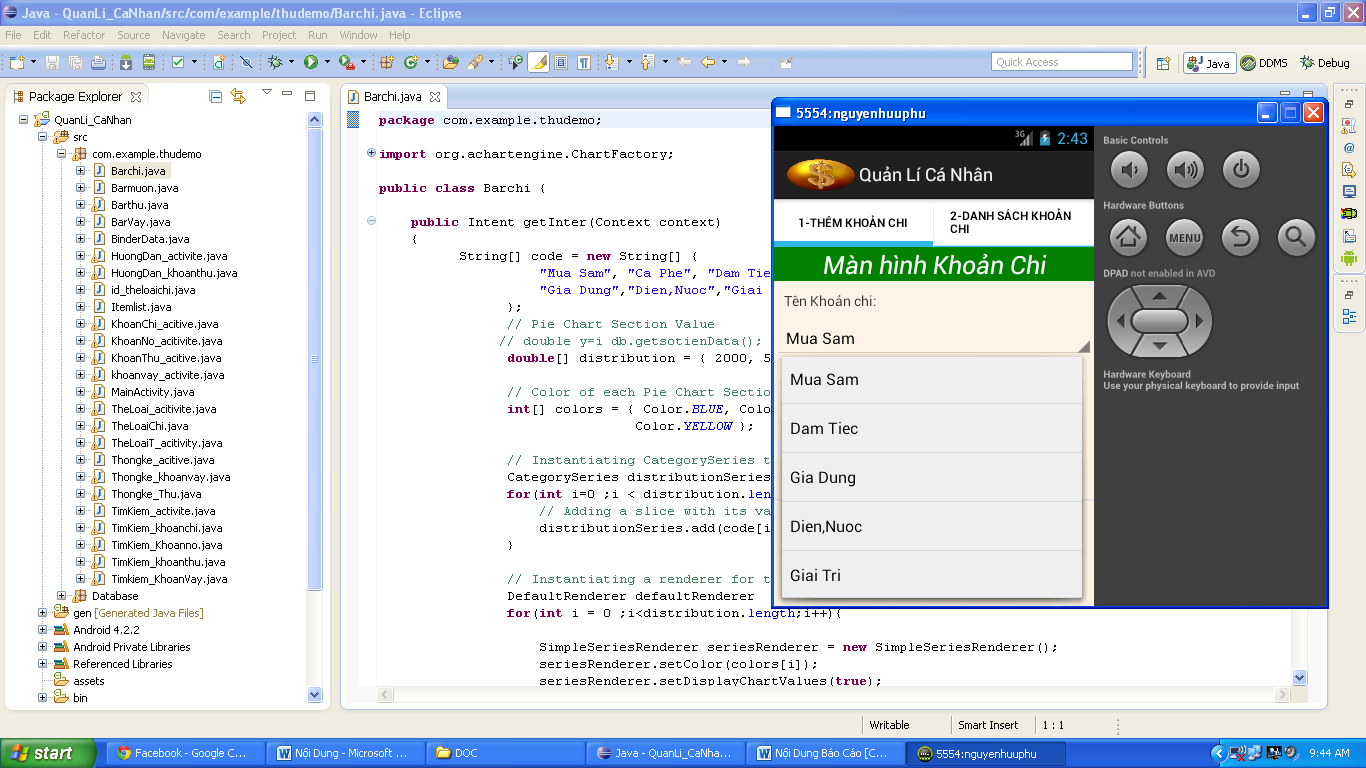
*Hình 4.2 Màn hình thêm khoản chi*

Các thành phần của giao diện:

* Spinner Tên khoản chi: hiển thị tên thể loại chi.
* Edit Text Số tiền: nhập số tiền của khoản chi.
* TextView ngày chi: Hiển thị ngày được chọn.
* Button Date: chọn ngày chi.
* Button Thêm khoản chi thêm khoản chi vào CSDL.

Cách sử dụng:

* Chọn tên thể loại chi từ Spinner Tên khoản chi.

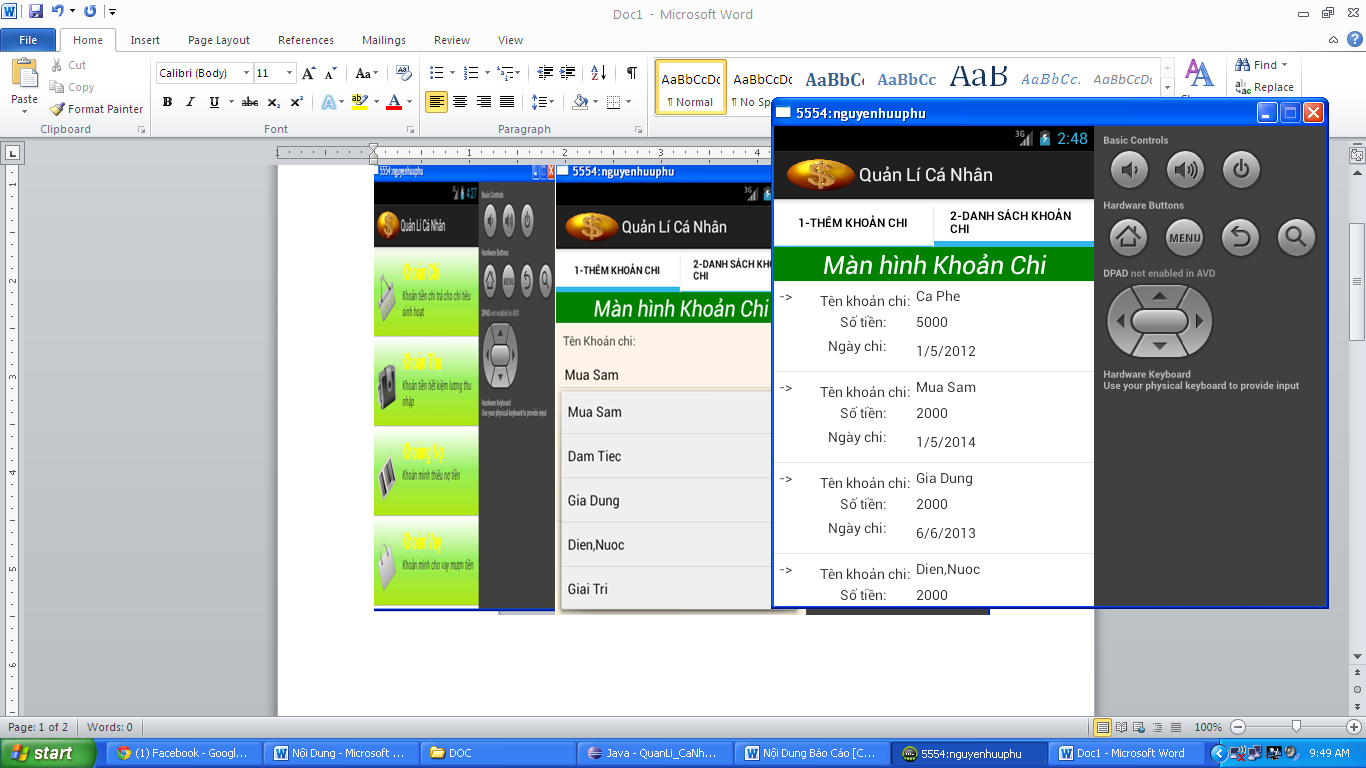


*Hình 4.3 Màn hình chọn tên thể loại chi.*

* Nhập vào số tiền.
* Chọn ngày chi.
* Nhấn nút “Thêm”.

Sau khi thêm thành công . Màn hình giao diện sẽ thông báo “Bạn thêm khoản chi thành công”.Bạn sẽ qua xem bên Danh Sách Khoản Chi, tên khoản chi mà bạn vừa thêm được.

### 4.2.3. Màn hình “Danh sách khoản chi”



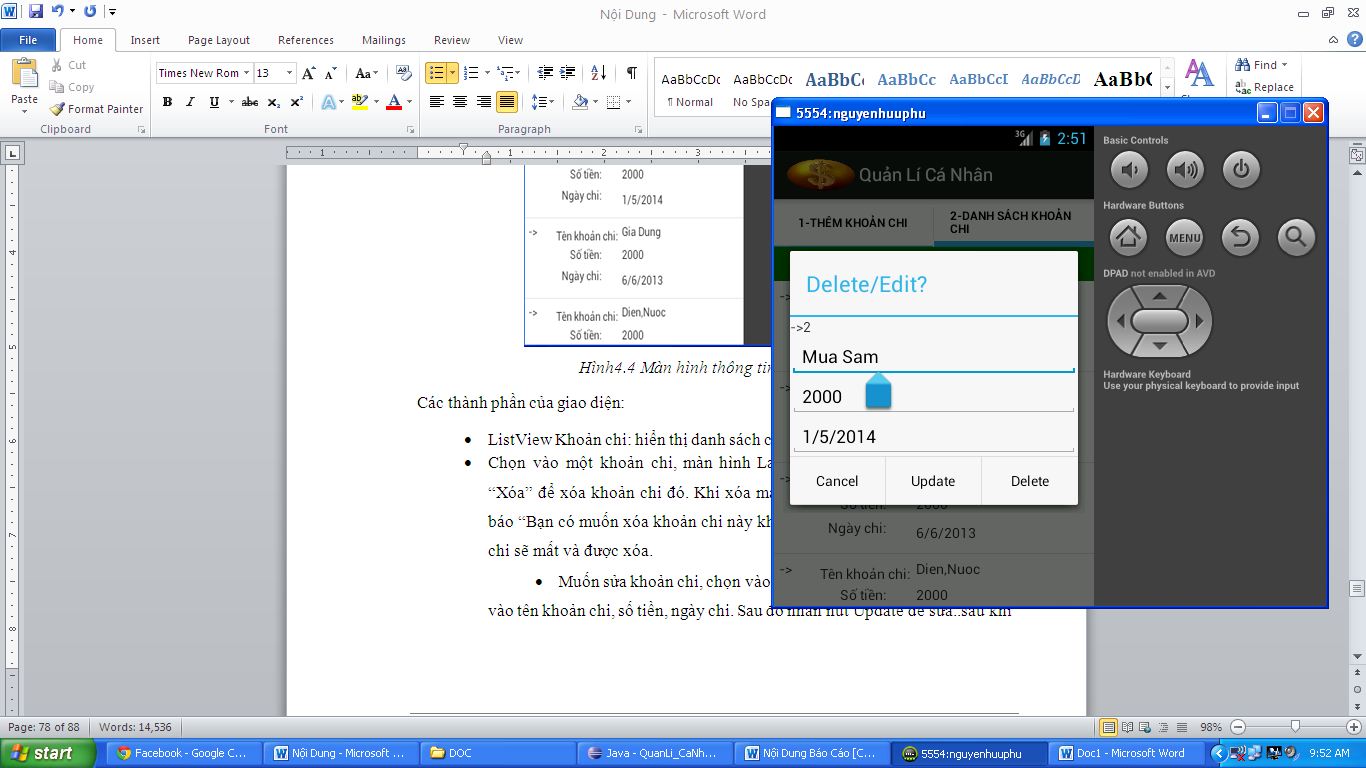
*Hình4.4 Màn hình thông tin khoản chi*

Các thành phần của giao diện:

* ListView Khoản chi: hiển thị danh sách các khoản chi.

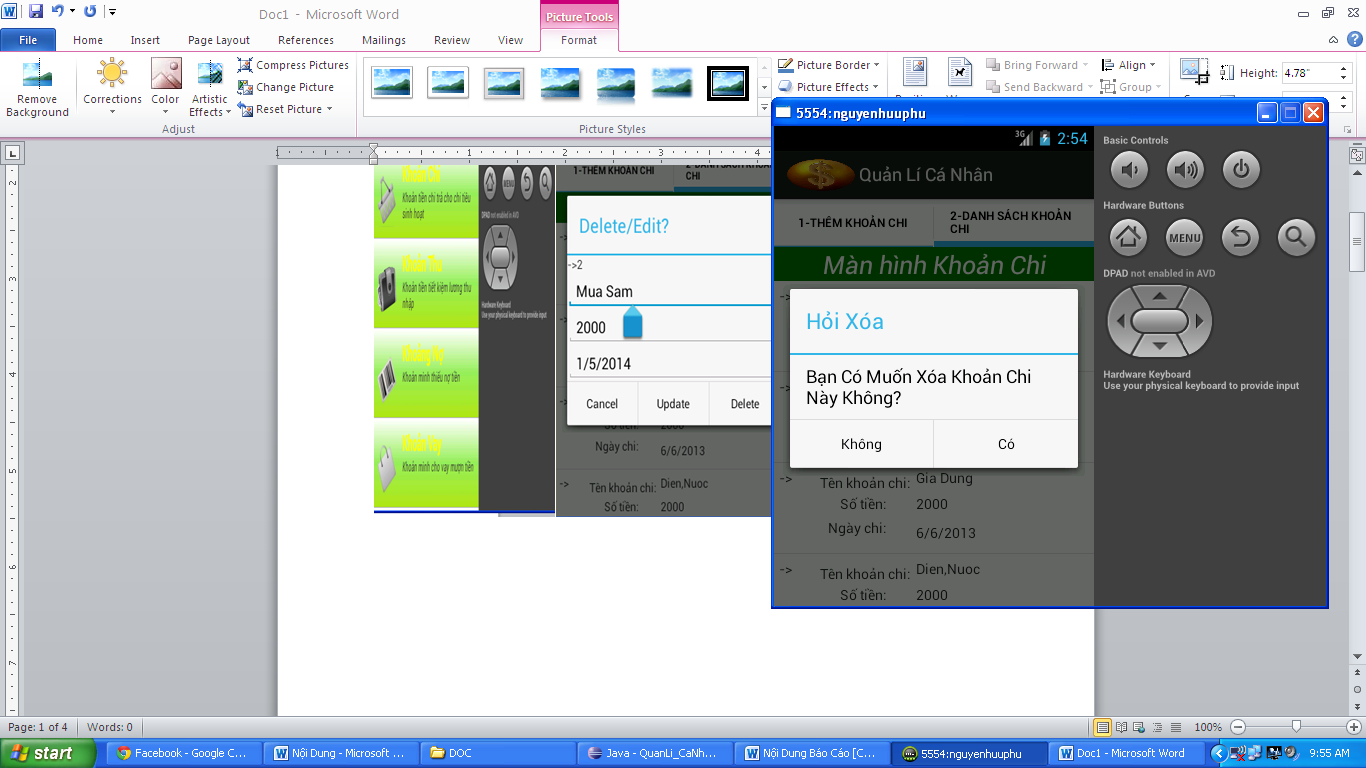
Cách sử dụng:

* Chọn vào một khoản chi, màn hình Layout Params sẽ hiện ra .
* Muốn sửa khoản chi, người dùng chọn vào khoản chi cần sửa và nhập vào tên khoản chi, số tiền, ngày chi. Sau đó nhấn nút Update để sửa.sau khi sửa thành công màn hình sẽ hiển thị lên thông báo “Bạn sửa tên khoản chi đó thành công”.



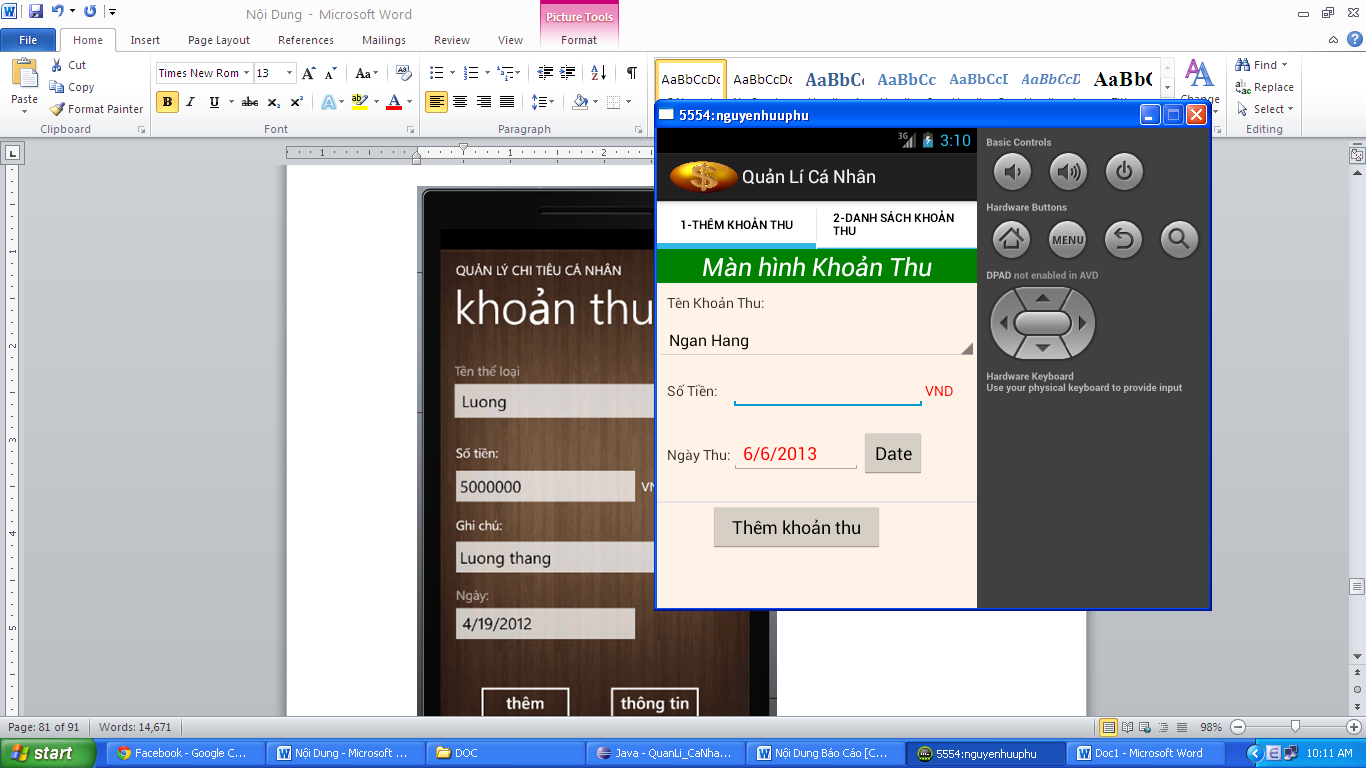
Hình 4.5 Màn hình Xóa khoản chi

* Bạn muốn xóa khoản chi thì hãy nhấn nút “Delete” để xóa khoản chi đó.Khi xóa màn hình giao diện sẽ hiển thị thông báo “Bạn có muốn xóa khoản chi này không?”, nếu bạn chọn có thì khoản chi sẽ được xóa.



*Hình 4.6 Màn hình hỏi xóa khoản chi.*

**4.2.4. Màn hình “Thêm khoản thu”**



*Hình4 .7 Màn hình thêm khoản thu*

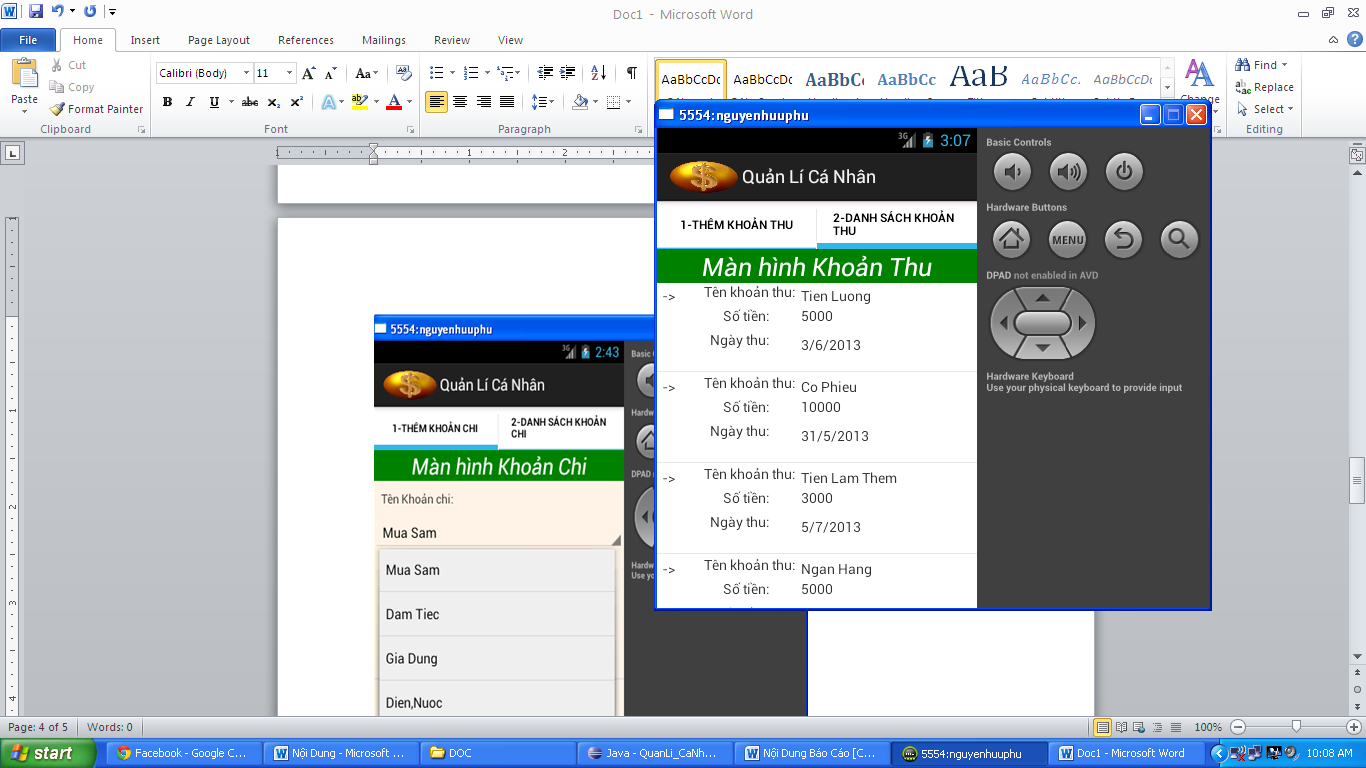
Các thành phần của giao diện:

* Spinner Tên khoản thu: hiển thị tên thể loại thu.
* Edit Text Số tiền: nhập số tiền của khoản thu.
* TextView ngày thu: Hiển thị ngày được chọn.
* Button Date: chọn ngày thu.
* Button Thêm khoản thu thêm khoản thu vào CSDL.

Cách sử dụng:

* Chọn tên thể loại chi từ Spinner Tên khoản thu.
* Nhập vào số tiền.
* Chọn ngày chi.
* Nhấn nút “Thêm”.

**4.2.5. Màn hình “Danh sách khoản thu”**



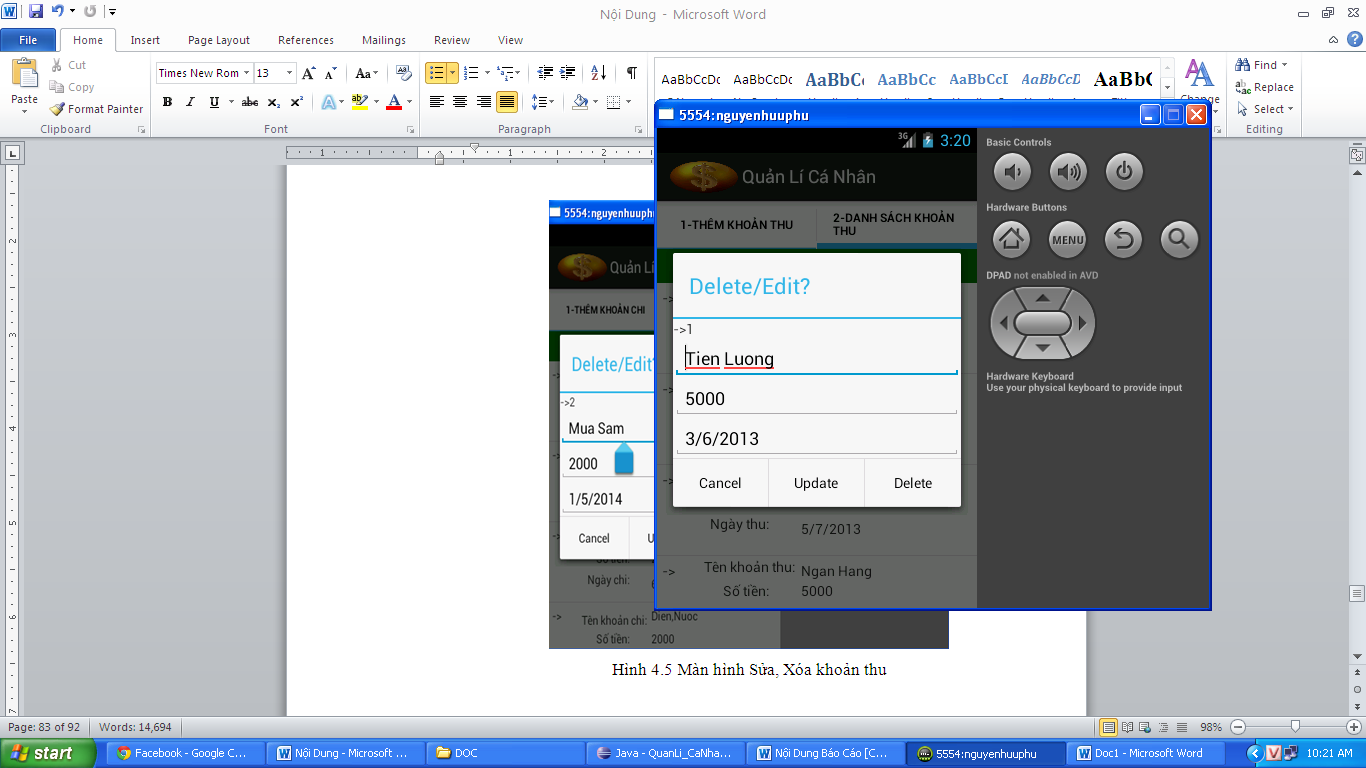
*Hình 4.8 Màn hình thông tin khoản thu*

Các thành phần của giao diện:

* TextView Khoản thu: hiển thị danh sách các khoản thu.

Cách sử dụng:

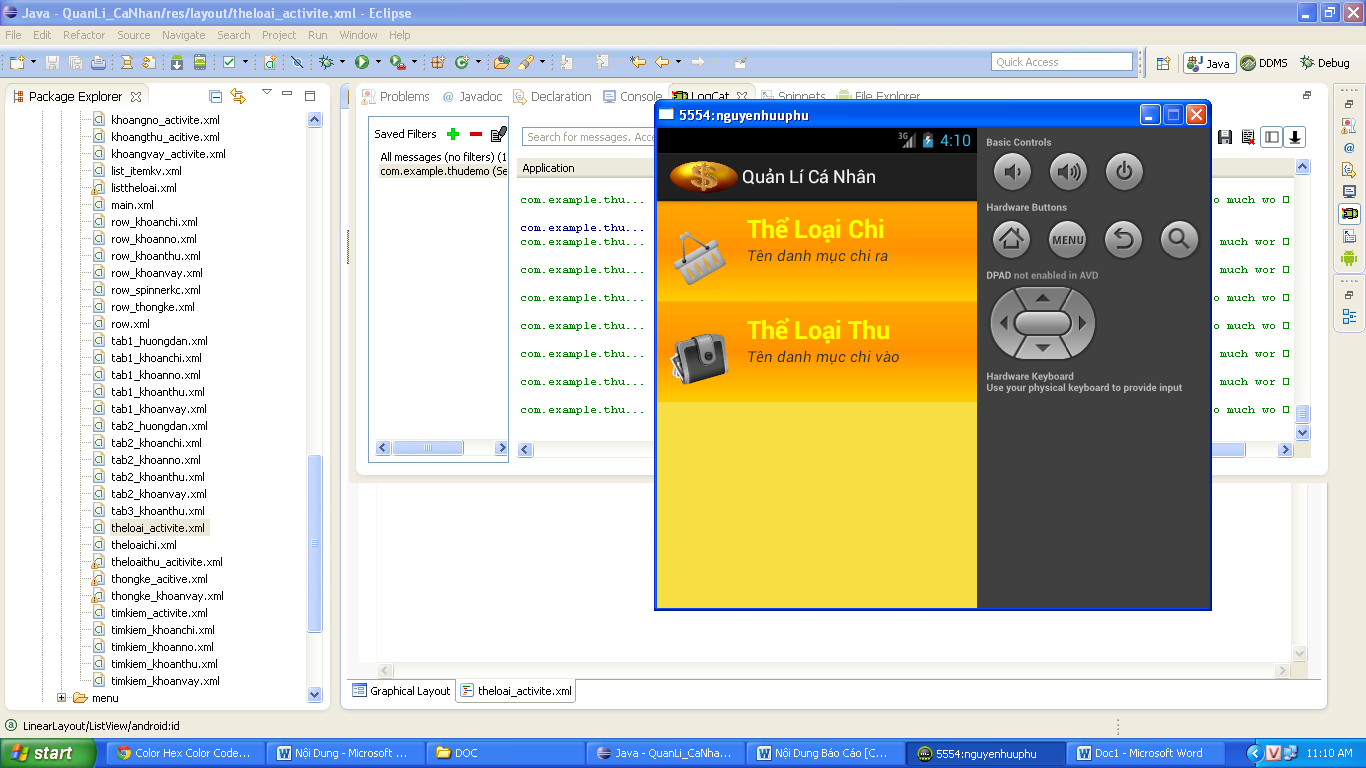
* Chọn vào một khoản thu, màn hình Layout Params khoản thu sẽ hiện ra .
* Muốn sửa khoản thu, người dùng chọn vào khoản thu cần sửa và nhập vào tên khoản thu, số tiền, ngày thu. Sau đó nhấn nút Update để sửa.sau khi sửa thành công màn hình sẽ hiển thị lên thông báo “Bạn sửa tên khoản thu đó thành công”.



*Hình 4.9 Màn hình Sửa, Xóa khoản thu*

* Bạn muốn xóa một khoản thu thì hãy Nhấn nút “Delete” để xóa khoản thu . Khi xóa màn hình giao diện sẽ hiển thị thông báo “Bạn có muốn xóa khoản thunày không?”, nếu bạn chọn có thì khoản thu sẽ được xóa.

**4.2.6 Màn hình “Thể loại”:**



*Hình 4.10 Màn hình thể loại*

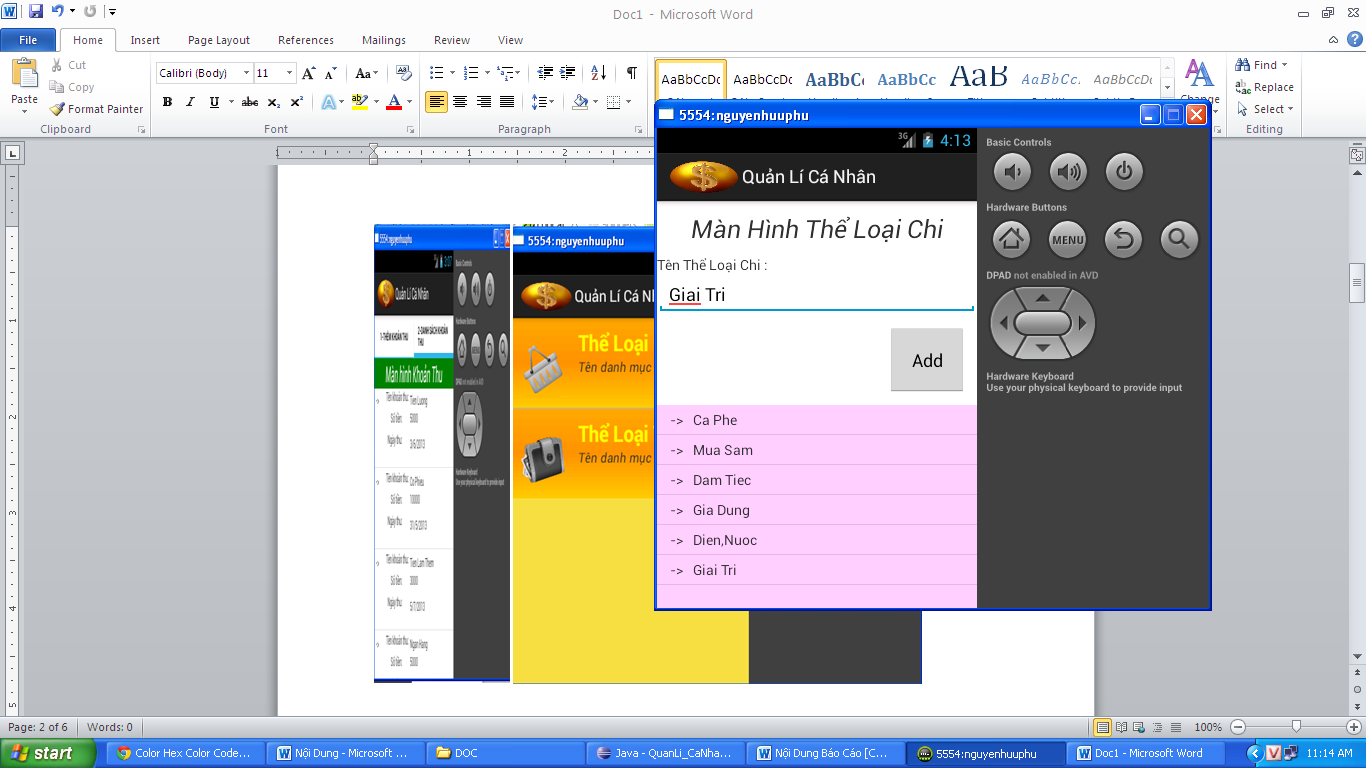
**Thành phần giao diện:**

* ListView thể loại: Hiển thị thể loại thu và thể loại chi.

**Xử lý:**

Nhấn chọn thể loại thu hoặc thể loại chi để đi đến màn hình thể loại thu hoặc thể loại chi.

**4.2.7 Màn hình “Thể loại chi”:**



*Hình 4.11 Màn hình thêm thể loại chi*

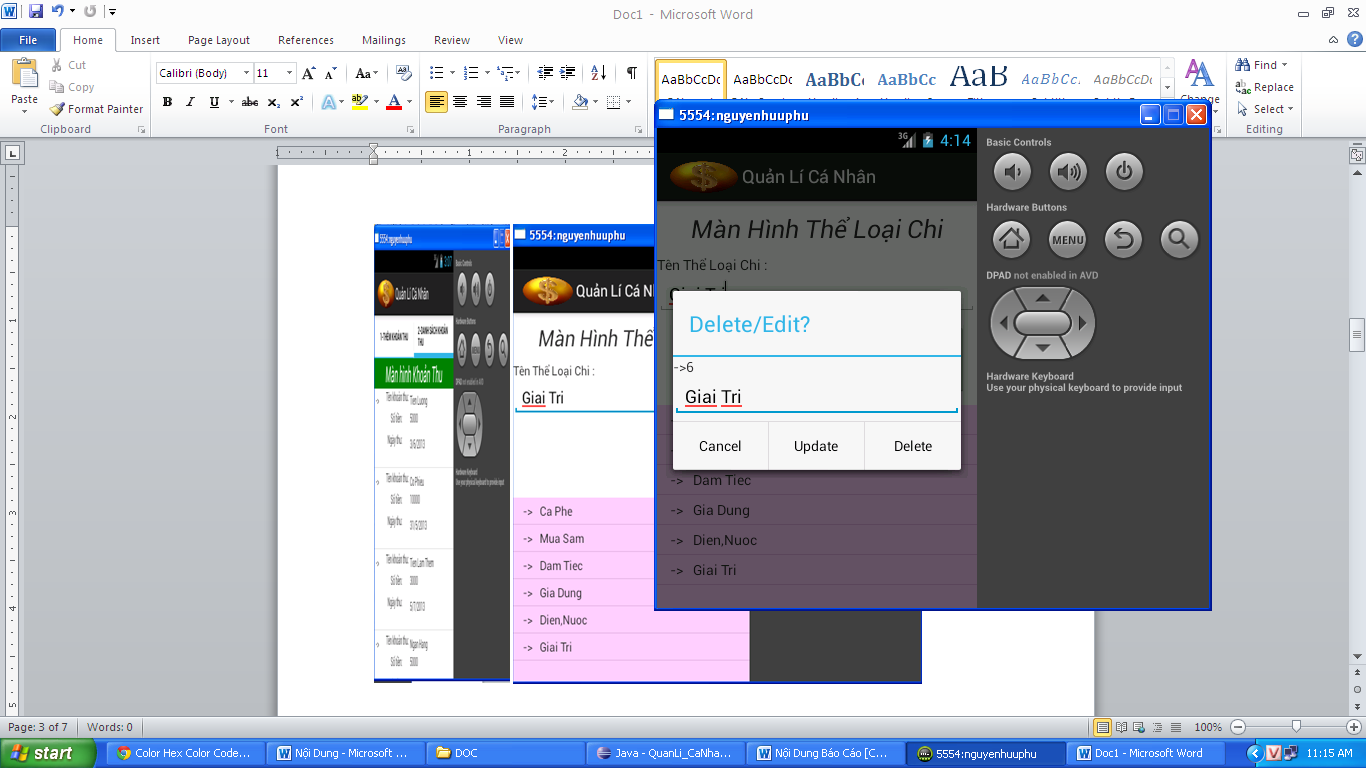
**Thành phần giao diện:**

* Edit Text tên thể loại: nhập tên thể loại chi
* Button ADD: thêm mới thể loại chi vào CSDL
* ListView hiển thị thông tin: Load các thể loại chi đã có trong CSDL xuống cơ sở dữ liệu.

**Xử lý:**

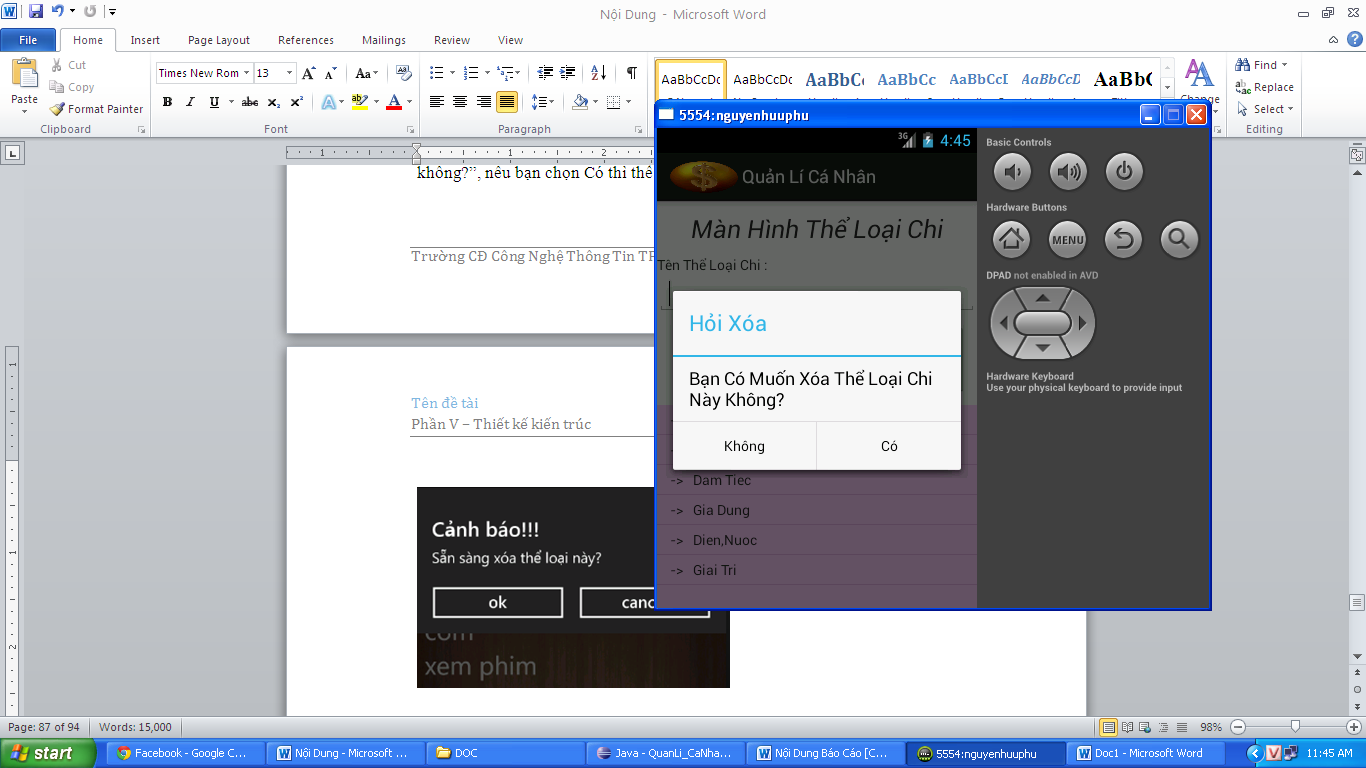
Nhập tên thể loại chi vào EditText “Tên thể loại”. Sau khi nhập xong nhấn button “Add” để thêm mới thể loại chi. Khi đó tên khoản chi được load xuống listview hiển thị “Thông tin” để xem và xóa,sửa các thể loại chi đã nhập vào. Bằng cách chọn vào một thể loại chi, màn hình Layout Params thể loại chi sẽ hiện ra .

Muốn sửa thể loại chi, người dùng chọn vào tên thể loại chi cần sửa và nhập vào tên thể loại chi. Sau đó nhấn nút Update để sửa.sau khi sửa thành công màn hình sẽ hiển thị lên thông báo “Bạn sửa tên thể loại chi thành công”.



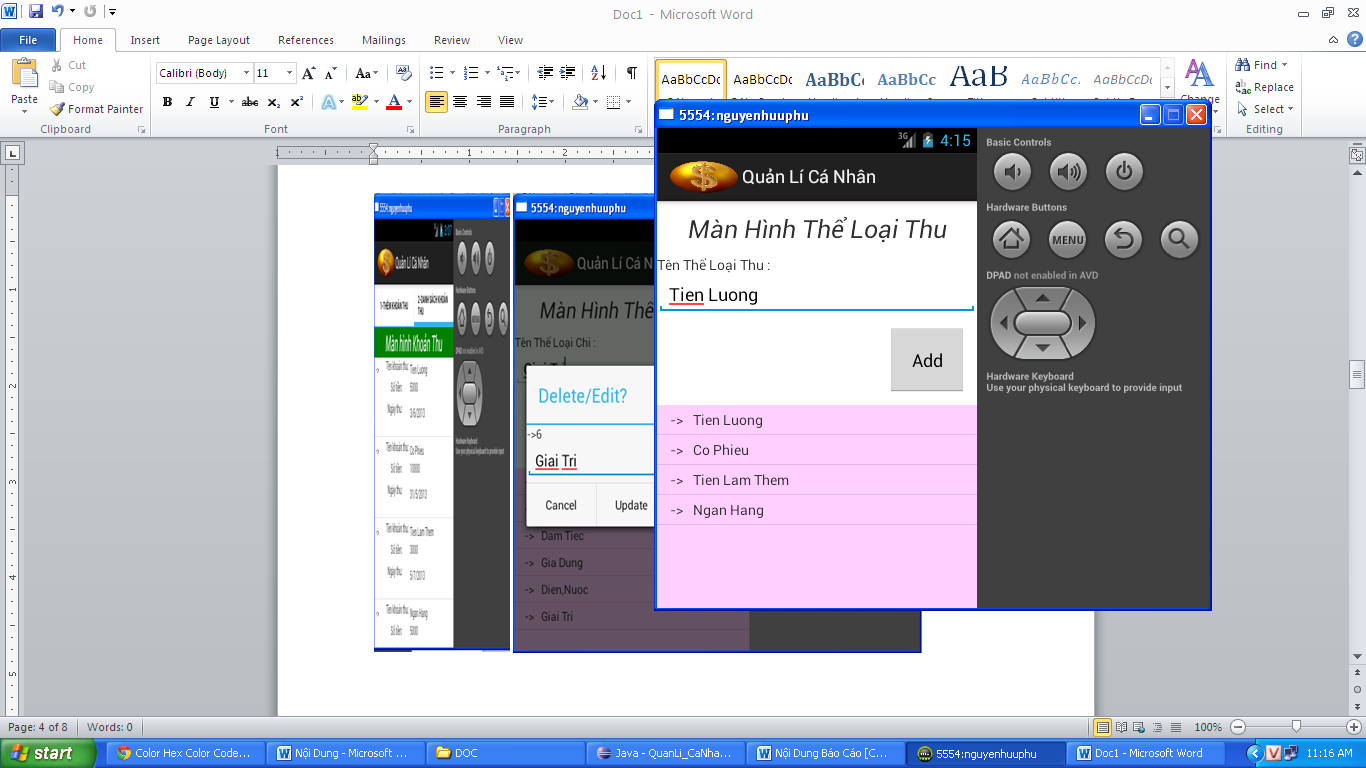
*Hình 4.12 Màn hình Sửa, Xóa thể loại chi*

Bạn muốn xóa một thể loại thì hãy Nhấn nút “Delete” để xóa thể loại chi. Khi xóa màn hình giao diện sẽ hiển thị thông báo “Bạn có muốn xóa thể loại chi này không?”, nếu bạn chọn Có thì thể loại chi sẽ được xóa.



*Hình 4.13 Màn hình hỏi xóa thể loại chi.*

**4.2.8 Màn hình “Thể loại thu”:**



*Hình 4.14 Màn hình thêm thể loại thu*

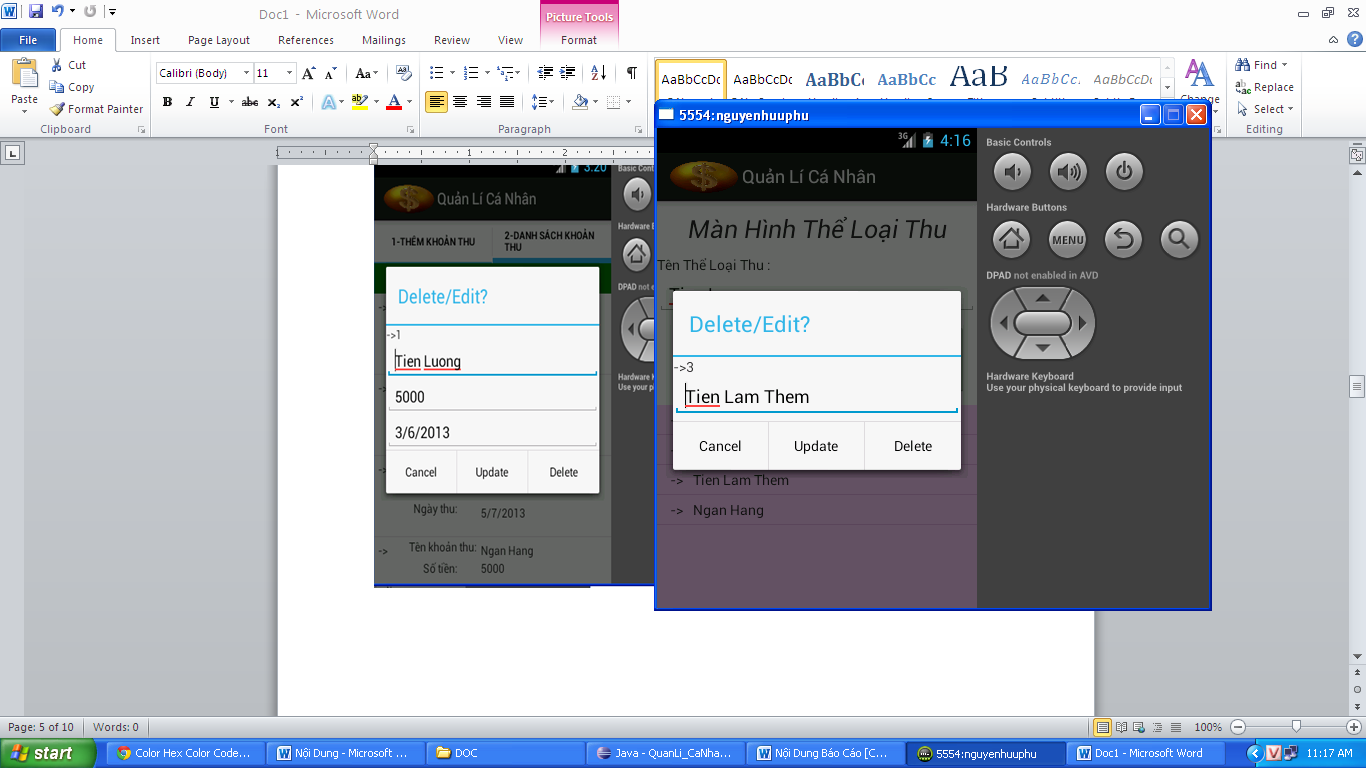
**Thành phần giao diện:**

* Edit Text tên thể loại: nhập tên thể loại thu
* Button ADD: thêm mới thể loại thu vào CSDL
* ListView hiển thị thông tin: Load các thể loại thu đã có trong CSDL xuống cơ sở dữ liệu.

**Xử lý:**

Nhập tên thể loại thu vào EditText “Tên thể loại”. Sau khi nhập xong nhấn button “Add” để thêm mới thể loại thu. Khi đó tên thể loại thu được load xuống listview hiển thị “Thông tin” để xem và xóa, sửa các thể loại chi đã nhập vào. Bằng cách chọn vào một thể loại thu, màn hình Layout Params thể loại chi sẽ hiện ra .

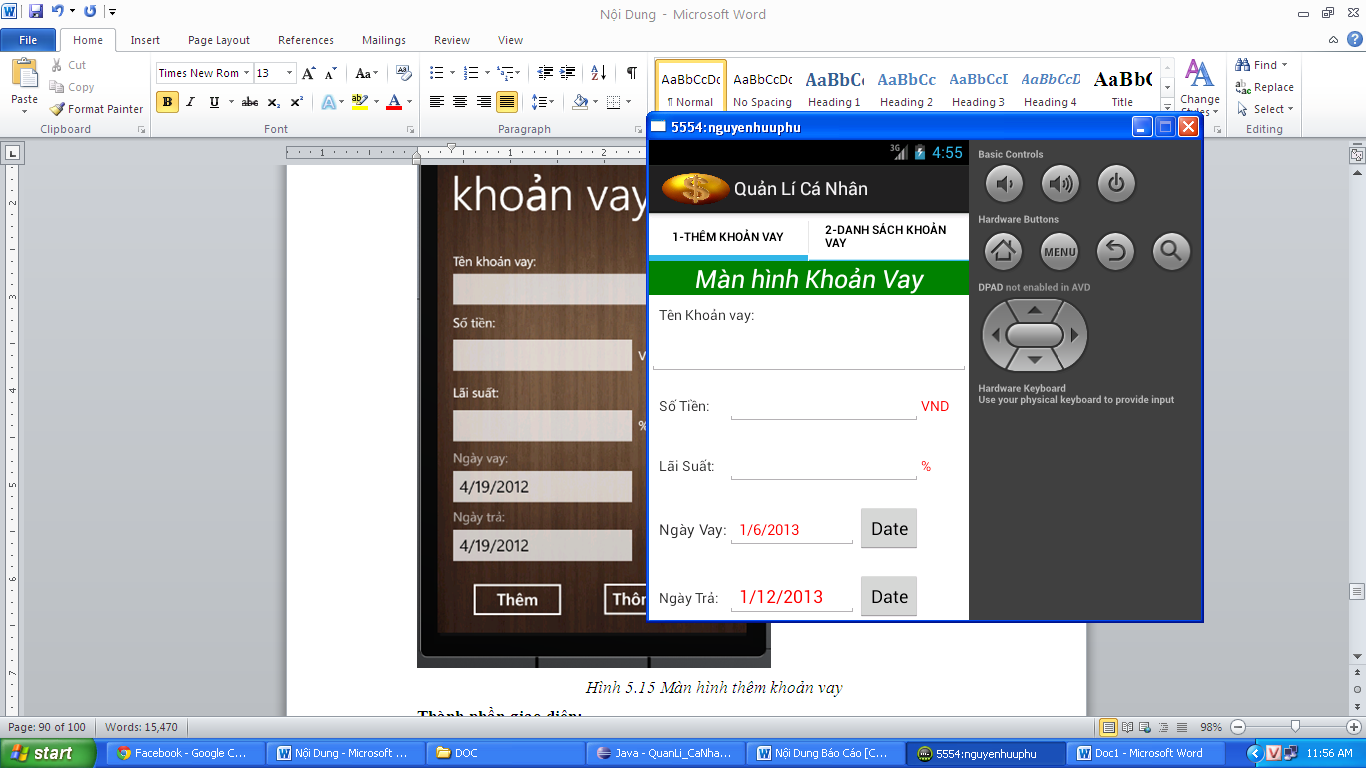
Muốn sửa thể loại thu, người dùng chọn vào tên thể loại thu cần sửa và nhập vào tên thể loại thu. Sau đó nhấn nút Update để sửa khi sửa thành công màn hình sẽ hiển thị lên thông báo “Bạn sửa tên thể loại thu thành công”.



*Hình 4.15 Màn hình Sửa, Xóa thể loại thu*

Bạn muốn xóa một thể loại thì hãy Nhấn nút “Delete” để xóa thể loại thu. Khi xóa màn hình giao diện sẽ hiển thị thông báo “Bạn có muốn xóa thể loại thu này không?”, nếu bạn chọn Có thì thể loại thu sẽ được xóa.

**4.2.9. Màn hình “Thêm khoản vay”**



*Hình 4.16 Màn hình thêm khoản vay*

**Thành phần giao diện:**

EditText Tên khoản vay: nhập tên khoản vay.

EditText Số tiền: nhập số tiền cho vay.

EditText Lãi suất: Nhập lãi suất.

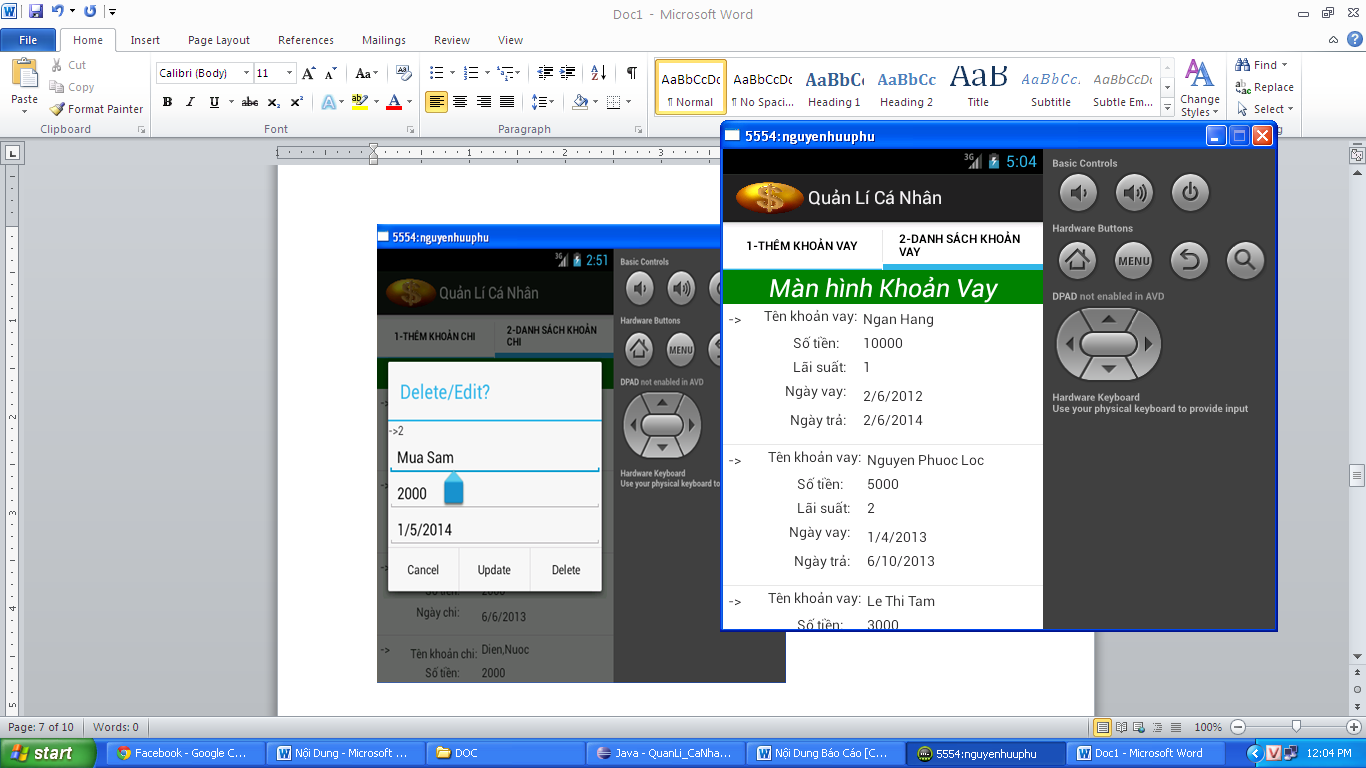
Button Ngày vay: nhập ngày cho vay.

Button Ngày trả: nhập ngày trả.

Button Thêm KV: thêm khoản vay mới vào CSDL.

**Xử lý:**

Sau khi người dùng nhập đủ thông tin thì bấm nút thêm để thêm khoản vay mới vào CSDL. Nếu muốn xem các khoản vay trong CSDL thì chuyển qua trang Danh sách khoản vay.



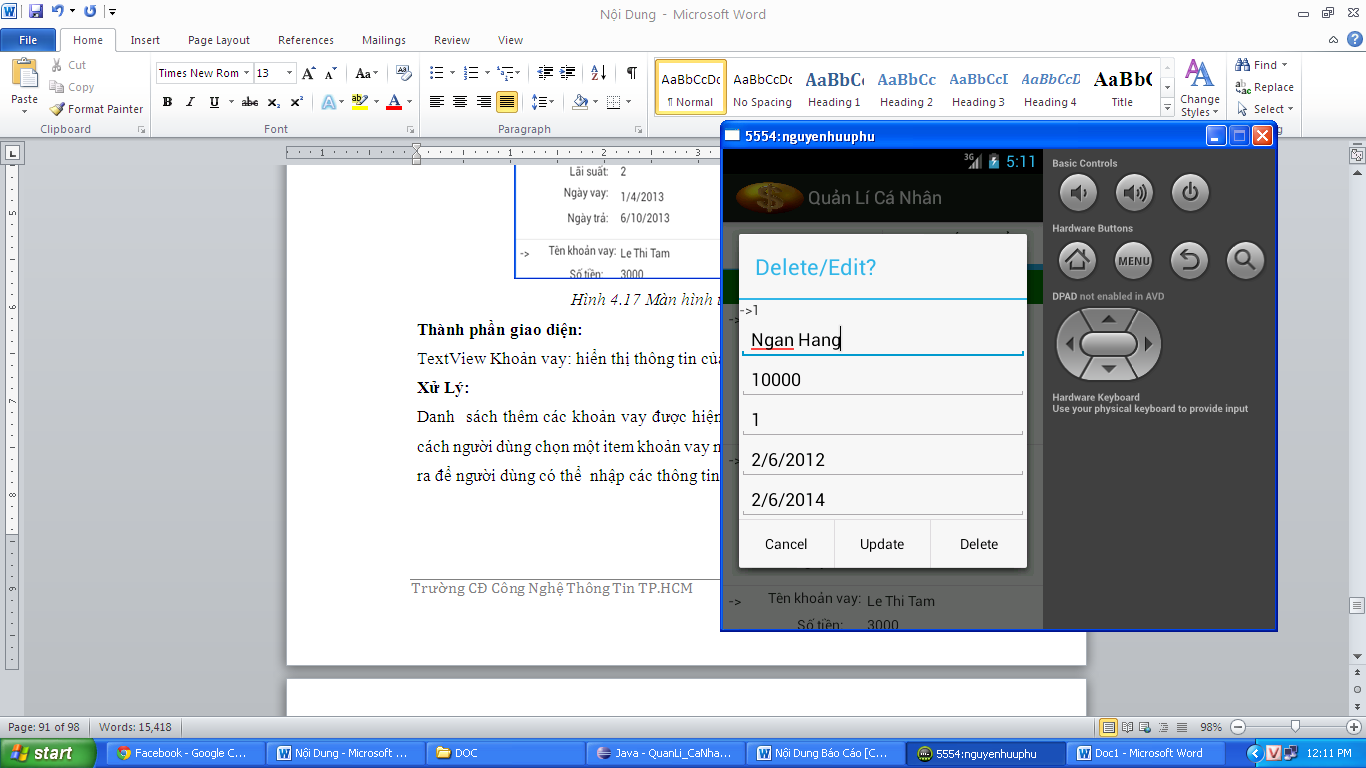
*Hình 4.17 Màn hình thông tin khoản vay*

**Thành phần giao diện:**

TextView Khoản vay: hiển thị thông tin của các khoản vay.

**Xử Lý:**

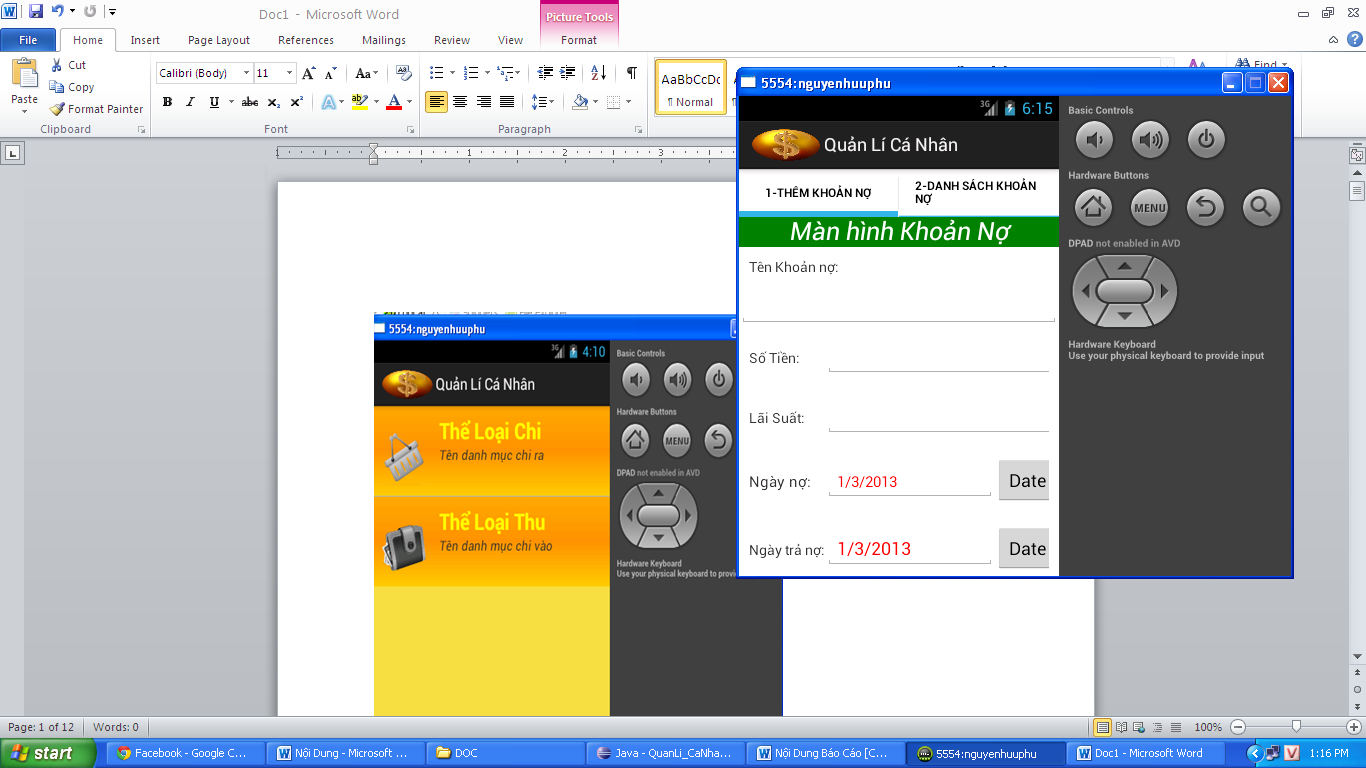
* Danh sách thêm các khoản vay được hiện ra cho người dùng muốn xóa, sửa .Bằng cách người dùng chọn một item khoản vay màn hình Layout Params khoản vay sẽ hiện ra để người dùng có thể nhập các thông tin cần sửa rồi nhấn nút Update để lưu lại .



*Hình 4.18 Màn hình xóa, sửa khoản vay.*

* Người dùng chọn một Khoản vay cần Xóa sau đó nhấn nút Delete. Khi xóa một khoản vay nào thì chương trình sẽ hỏi xác nhận ta có muốn xóa khoản vay đó không.

**4.2.10. Màn hình “Thêm khoản nợ”**



*Hình 4.19 Màn hình thêm khoản nợ*

**Thành phần giao diện:**

EditText Tên khoản nợ: nhập tên khoản nợ.

EditText Số tiền: nhập số tiền cho nợ.

EditText Lãi suất: Nhập lãi suất.

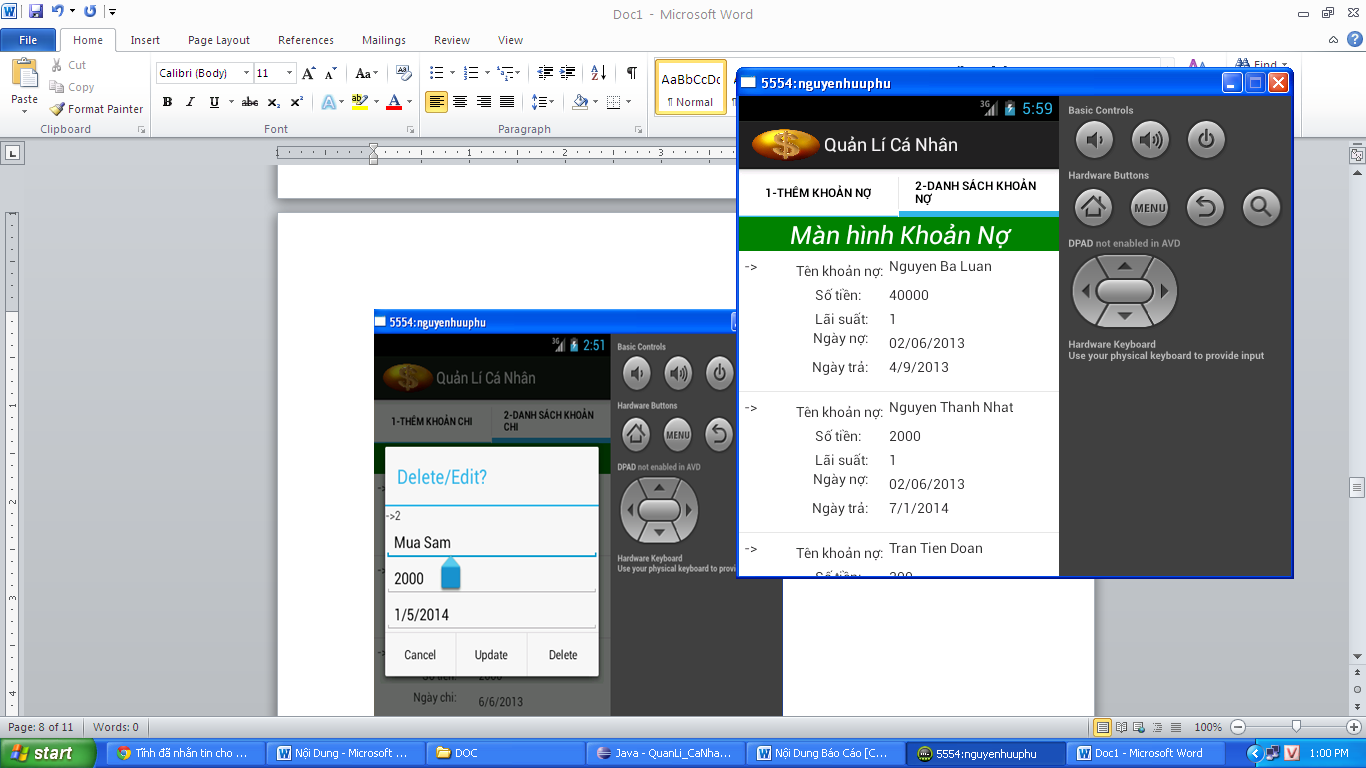
Button Ngày nợ: nhập ngày cho nợ.

Button Ngày trả nợ: nhập ngày trả nợ.

Button Thêm KN: thêm khoản nợ mới vào CSDL.

**Xử lý:**

Sau khi người dùng nhập đủ thông tin thì bấm nút thêm để thêm khoản nợ mới vào CSDL. Nếu muốn xem các khoản nợ trong CSDL thì chuyển qua trang Danh sách khoản nợ.



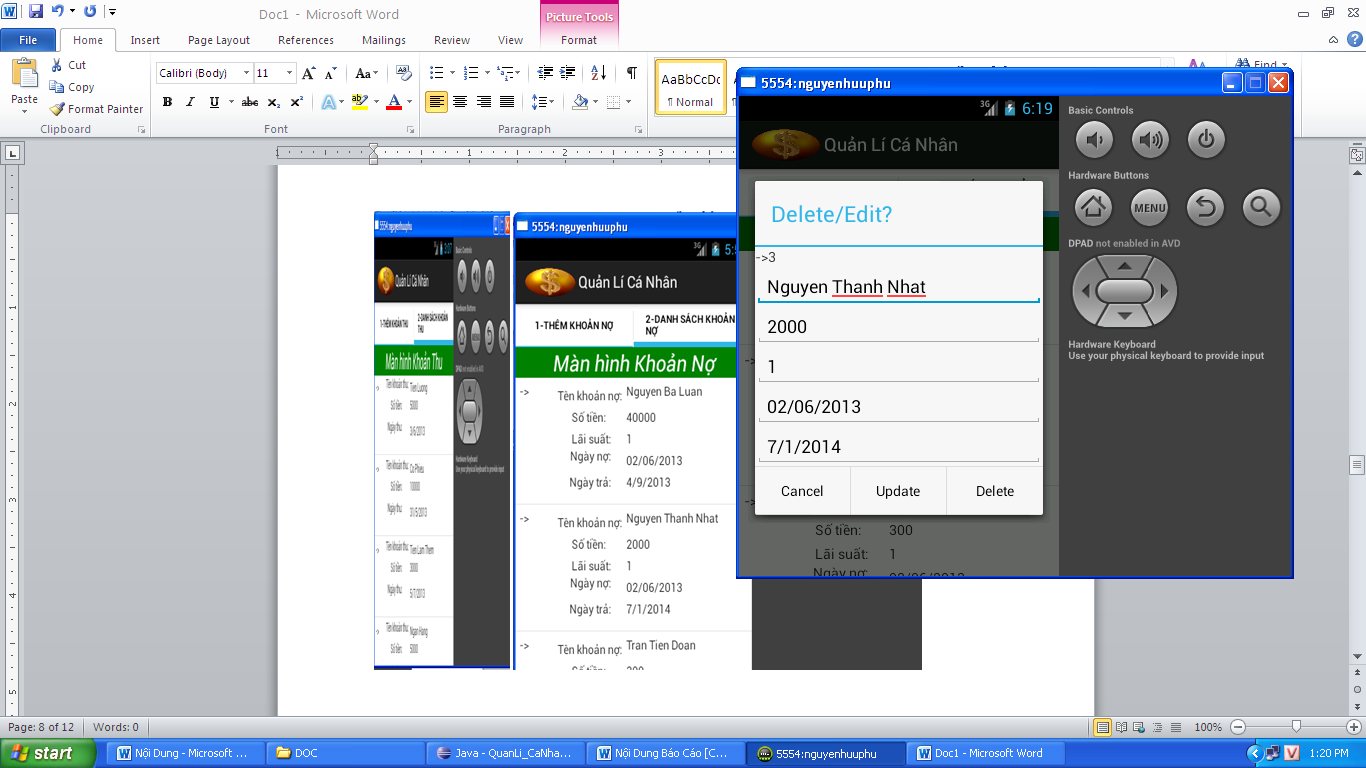
*Hình 4.20 Màn hình thông tin khoản nợ*

**Thành phần giao diện:**

TextView Khoản nợ: hiển thị thông tin của các khoản nợ.

**Xử Lý:**

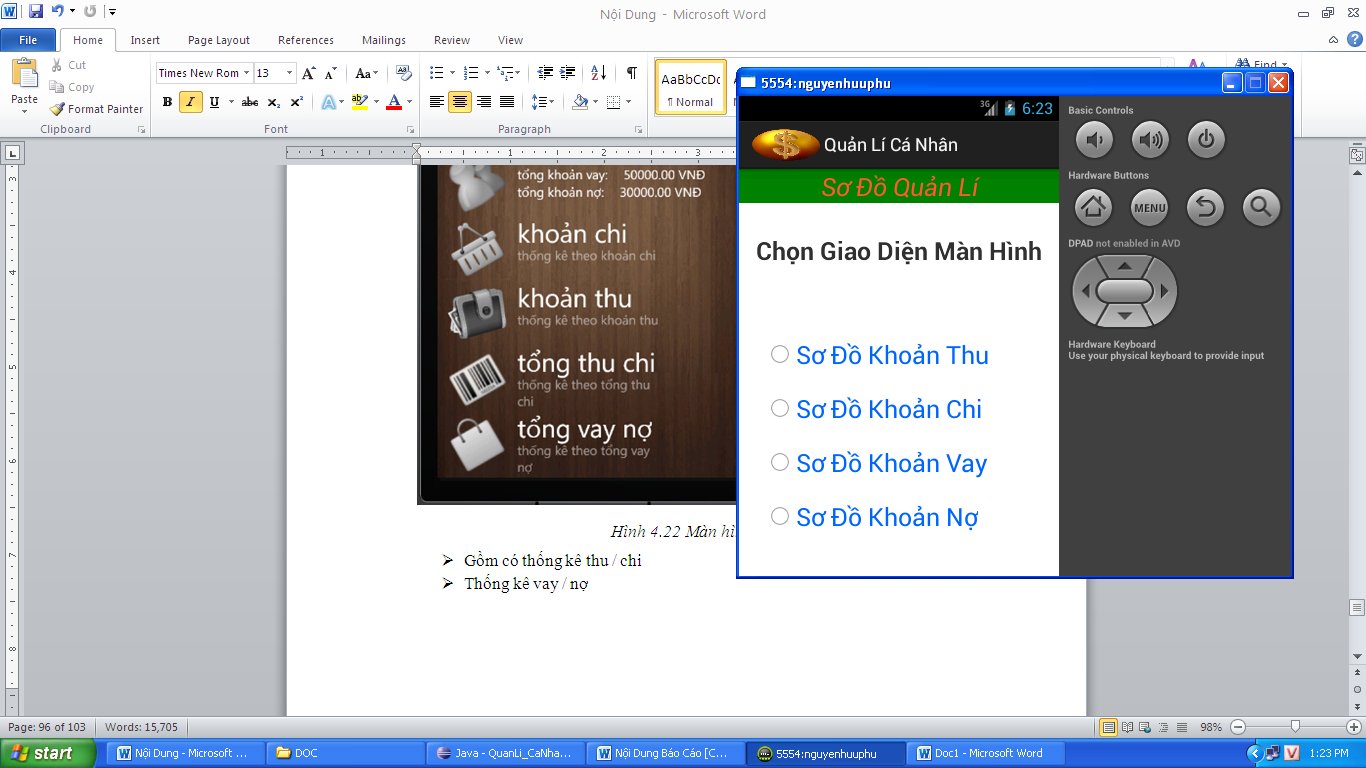
* Danh sách thêm các khoản nợ được hiện ra cho người dùng muốn xóa, sửa . Bằng cách người dùng chọn một item khoản nợ màn hình Layout Params khoản nợ sẽ hiện ra để người dùng có thể nhập các thông tin cần sửa rồi nhấn nút Update để lưu lại .



*Hình 4.21 Màn hình xóa, sửa khoản nợ.*

* Người dùng chọn một Khoản nợ cần Xóa sau đó nhấn nút Delete. Khi xóa một khoản nợ nào thì chương trình sẽ hỏi xác nhận ta có muốn xóa khoản nợ đó không.

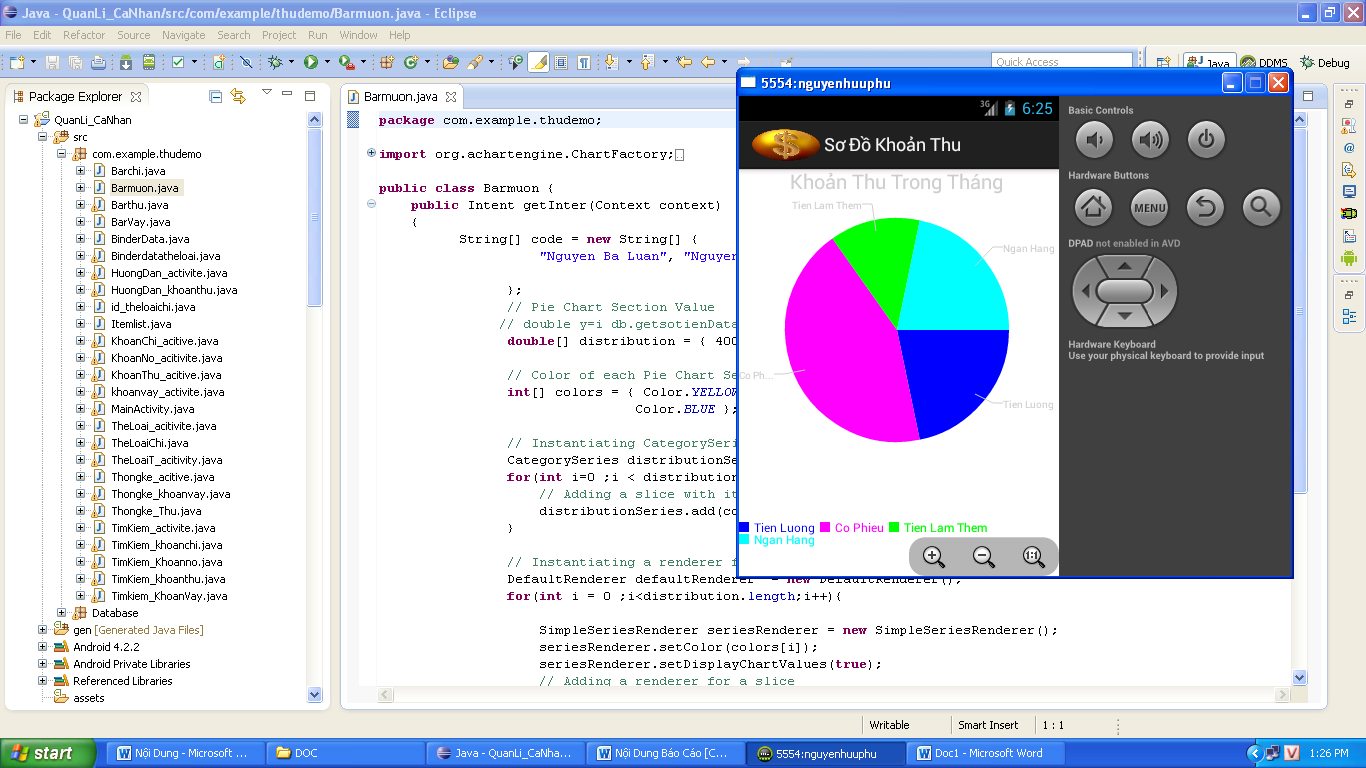
### 4.2.11. Màn hình “thống kê”



*Hình 4.22 Màn hình thống kê*

* Gồm có thống kê thu / chi
* Thống kê vay / nợ

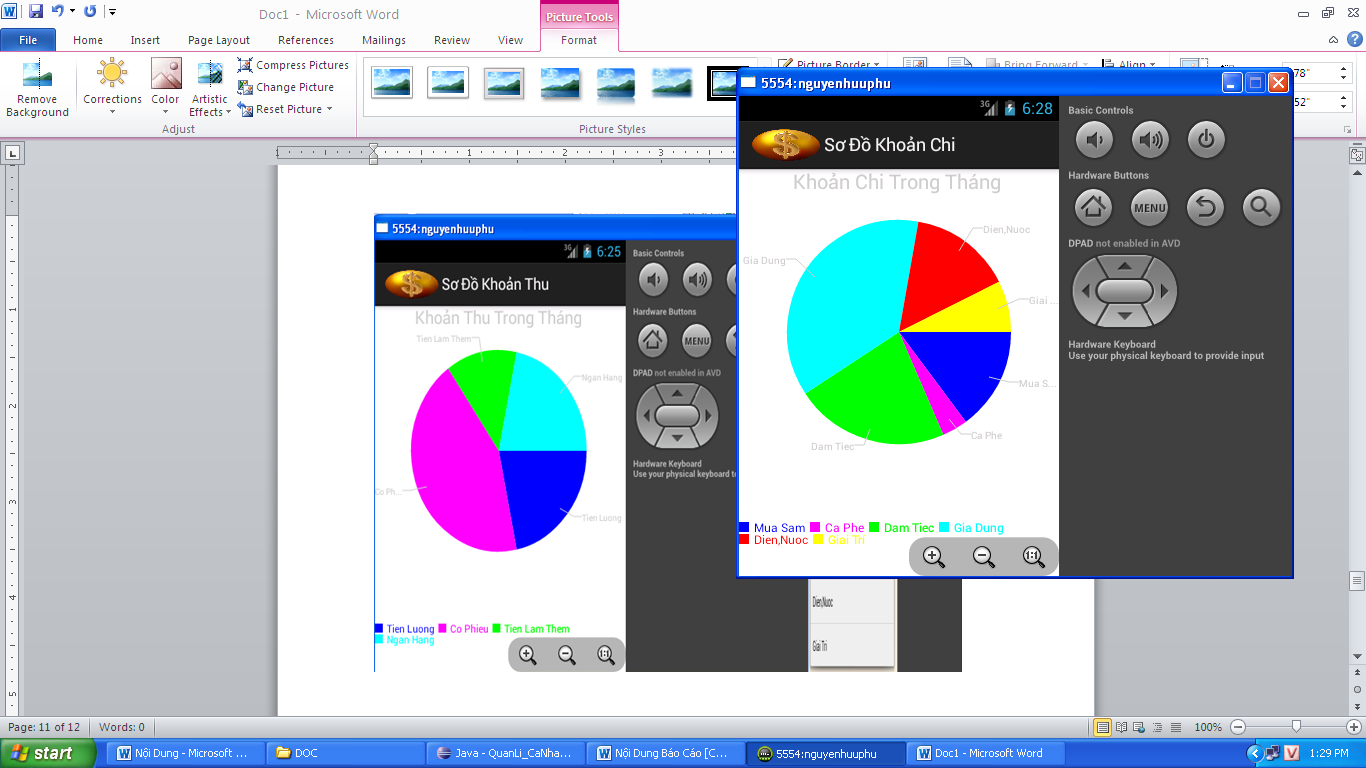
### 4.2.12. Màn hình “thống kê khoản thu”



*Hình 4.23 Màn hình thống kê khoản thu*

* Thống kê khoản thu theo biểu đò tròn.

### 4.2.13. Màn hình “thống kê khoản chi”



*Hình 4.24 Màn hình thống kê khoản chi*

* Thống kê khoản chi theo biểu đò tròn.

|  |  |
| --- | --- |
| Quản Lý  Tài Chính | PHẦN 4 **Kết Quả Thực Hiện** |
|  | |

## 1. Môi trường phát triển và triển khai

* ***Môi trường phát triển ứng dụng:***
  + Hệ điều hành: Android 4.2.2
  + Hệ quản trị cơ sở dữ liệu: SQLITE.
  + Công cụ phân tích thiết kế: IBM Rational Rose Enterprse Edition, sqlitebrowser.
  + Công cụ xây dựng ứng dụng: SDK Manager,Java, Eclipse.
  + Các phần mềm đã dùng:
* achartengine-1.0.0 : phần mềm giúp vẽ biểu đồ.
* ***Môi trường triển khai ứng dụng:***
  + Hệ điều hành: Android 4.2.2

## 2. Kết quả đạt được

* Những chức năng đã phân tích, thiết kế chi tiết:
  + - * Thêm, xóa, sửa thể loại thu / chi.
      * Quản lý, cập nhật các khoản thu, khoản chi dựa trên các thể loại thu/chi đã thêm trong CSDL.
      * Quản lý, cập nhật các khoản cho vay, các khoản mượn nợ.
      * Thống kê các khoản thu/chi, các khoản vay/nợ. Thống kê theo dạng biểu đồ hình tròn.
* Những chức năng đã cài đặt hoàn chỉnh:
  + - * Thêm, xóa, sửa thể loại thu / chi.
      * Quản lý, cập nhật các khoản thu, khoản chi dựa trên các thể loại thu/chi đã thêm trong CSDL.
      * Thống kê các khoản thu/chi, các khoản vay/nợ. Thống kê theo các dạng biểu đồ.
* Những chức năng đã cài đặt nhưng chưa hoàn chỉnh:
  + - * Tổng thống kê khoản thu, khoản chi, khoản vay / nợ theo ngày.
* Những chức năng chỉ có giao diện ,chưa được xử lý:
  + - * Thiết lập thay đổi giao diện (hình nền) và ngôn ngữ cho ứng dụng.

## 3. Hướng phát triển

* Hoàn chỉnh các chức năng tìm kiếm.
* Xử lý thiết lập thay đổi ngôn ngữ, thay đổi hình nền.
* Xây dựng thêm các dạng biểu đồ thống kê.

# TÀI LIỆU THAM KHẢO

1. Sử dụng biểu đồ trong Android

<http://www.youtube.com/watch?v=E9fozQ5NlSo&list=PL2603F3CABBF5EEB0> .

1. Học Android qua web

<http://duythanhcse.wordpress.com/android/>.

1. Hướng dẫn Local Database Sqlite

<http://stackoverflow.com/questions/5901536/read-database-android-sqlite?rq=1>,

<http://www.codeproject.com/Articles/119293/Using-SQLite-Database-with-Android>.

1. Hướng dẫn Local Database Sqlite lên Spinner

<http://www.androidhive.info/2012/06/android-populating-spinner-data-from-sqlite-database/>.

1. Hướng dẫn tìm kiếm trong danh sách Database

<http://androidcocktail.blogspot.com/2012/04/search-custom-listview-in-android.html>.

<http://android-helper.blogspot.com/2011/03/android-search-in-listview-example.html>.

<http://stackoverflow.com/questions/11901836/how-to-handle-a-search-suggestion-clicked-item>.

1. Hướng dẫn đọc file text lên giao diện

<http://stackoverflow.com/questions/3800001/how-to-draw-arabic-text-on-android-after-reading-from-a-file?rq=1>.

1. Copyright © 2009 by Mark L. Murphy. Beginning Android. Appress
2. Copyright © 2009 by Sayed Y. Hashimi and Satya Komatineni. Pro Android. Appress
3. Tài liệu học android

[http://sinhvienit.net/forum/tai-lieu-lap-trinh-di-dong-tren-android-tieng- viet.205952.html](http://sinhvienit.net/forum/tai-lieu-lap-trinh-di-dong-tren-android-tieng-%20%20%20%20%20%20%20%20%20%20%20%20viet.205952.html).