***ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH***

***TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN***

******

***BÁO CÁO ĐỒ ÁN CUỐI KỲ***

---------oOo---------

Chuyên đề Mobile   
and Pervasive Computing

**Khoa: *CÔNG NGHỆ PHẦN MỀM***

**Đề tài: Quản lý thu chi**

**Giáo viên hướng dẫn: Thầy Nguyễn Trác Thức**

**Thực hiện: 12520069 Đào Thị Kiều Diễm**

**12520128 Lê Văn Hạnh**

**12520130 Vưu Chí Hào**

**12520181 Bùi Thị Lệ Huyền**

**TP.HCM 12/2016**

**Nhận xét của giáo viên**:

…………………………………………………………………………………………..

…………………………………………………………………………………………..

…………………………………………………………………………………………..

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

…………………………………………………………………………………………..

…………………………………………………………………………………………..

…………………………………………………………………………………………..

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

…………………………………………………………………………………………..

…………………………………………………………………………………………..

…………………………………………………………………………………………..

……………………………………………………………………………………………

……………………………………………………………………………………………

Mục lục:

[CHƯƠNG 1: GIỚI THIỆU: 6](#_Toc470909222)

[1. Tổng quan về Mobile & Pervasive Computing: 6](#_Toc470909223)

[1.1. Mobile Computing (Điện toán di động): 6](#_Toc470909224)

[1.2. Pervasive Computing (Ubiquitous Computing – Điện toán phân tán rộng khắp): 22](#_Toc470909225)

[2. Giới thiệu về đồ án: 25](#_Toc470909226)

[a. Môi trường phát triển và triển khai 25](#_Toc470909227)

[b. Đối tượng và mục tiêu sử dụng: 25](#_Toc470909228)

[CHƯƠNG 2: SƠ LƯỢC VỀ HỆ ĐIỀU HÀNH 26](#_Toc470909229)

[1. Sơ lược về hệ điều hành: 26](#_Toc470909230)

[2. Giới thiệu Android: 26](#_Toc470909231)

[a. Lịch sử phát triển của Android: 27](#_Toc470909232)

[b. Các ứng dụng có sẵn trên Android: 29](#_Toc470909233)

[c. Android software development kit (SDK): 32](#_Toc470909234)

[d. Kiến trúc ứng dụng: 32](#_Toc470909235)

[e. Các thư viện của Android: 33](#_Toc470909236)

[3. Kiến trúc hệ điều hành Android 35](#_Toc470909237)

[a. Tầng ứng dụng (applications): 36](#_Toc470909238)

[b. Application Framework: 36](#_Toc470909239)

[c. Library: 37](#_Toc470909240)

[d. Android runtime: 37](#_Toc470909241)

[4. Các thành phần trong ứng dụng Android 38](#_Toc470909242)

[CHƯƠNG 3: MÔ TẢ SƠ ĐỒ USECASE: 64](#_Toc470909243)

[1. Sơ đồ Use-case: 64](#_Toc470909244)

[2. Danh sách các Actor: 65](#_Toc470909245)

[3. Danh sách các Use-case: 65](#_Toc470909246)

[4. Đặc tả Use-case 66](#_Toc470909247)

[a) Use-case Thêm khoản thu 66](#_Toc470909248)

[b) Use-case Xóa khoản thu 66](#_Toc470909249)

[c) Use-case Sửa khoản thu 67](#_Toc470909250)

[d) Use-case Thêm khoản chi: 67](#_Toc470909251)

[e) Use-case Xóa khoản chi: 68](#_Toc470909252)

[f) Use-case Sửa khoản chi: 68](#_Toc470909253)

[g) Use-case Thêm khoản nợ: 69](#_Toc470909254)

[h) Use-case Xóa khoản nợ: 69](#_Toc470909255)

[i) Use-case Sửa khoản nợ: 69](#_Toc470909256)

[j) Use-case Tìm kiếm: 70](#_Toc470909257)

[k) Use-case Thêm loại chi: 71](#_Toc470909258)

[l) Use-case Sửa loại chi: 71](#_Toc470909259)

[m) Use-case Xóa loại chi: 72](#_Toc470909260)

[n) Use-case Thêm loại thu: 72](#_Toc470909261)

[o) Use-case Xóa loại thu 72](#_Toc470909262)

[p) Use-case Sửa loại thu: 73](#_Toc470909263)

[q) Use-case Thêm khoản vay: 73](#_Toc470909264)

[r) Use-case Sửa khoản vay: 74](#_Toc470909265)

[s) Use-case Xóa khoản vay: 74](#_Toc470909266)

[q) Use-case Thống kê: 75](#_Toc470909267)

[CHƯƠNG 4: SƠ ĐỒ PHÂN TÍCH: 76](#_Toc470909268)

[1. Sơ đồ lớp: 76](#_Toc470909269)

[a. Sơ đồ lớp: 76](#_Toc470909270)

[b. Các lớp đối tượng và quan hệ: 76](#_Toc470909271)

[c. Mô tả chi tiết các lớp đối tượng: 77](#_Toc470909272)

[2. Sơ đồ tuần tự 79](#_Toc470909273)

[a. Thêm khoản thu: 79](#_Toc470909274)

[b. Thêm khoản chi 81](#_Toc470909275)

[c. Thêm thể loại chi 81](#_Toc470909276)

[CHƯƠNG 5: THIẾT KẾ GIAO DIỆN 82](#_Toc470909277)

[1. Danh sách các màn hình 82](#_Toc470909278)

[2. Mô tả chi tiết mỗi màn hình: 84](#_Toc470909279)

[a. Màn hình “Màn hình khởi động”: 84](#_Toc470909280)

[b) Màn hình “Thêm khoản chi”: 85](#_Toc470909281)

[c) Màn hình “Thêm khoản thu” 87](#_Toc470909282)

[d) Màn hình “Danh sách khoản thu” 89](#_Toc470909283)

[e) Màn hình “Thể loại”: 91](#_Toc470909284)

[f) Màn hình “Thể loại chi”: 92](#_Toc470909285)

[g) Màn hình “Thể loại thu”: 94](#_Toc470909286)

[h) Màn hình “Thêm khoản vay” 96](#_Toc470909287)

[i) Màn hình “Thêm khoản nợ” 98](#_Toc470909288)

[j) Màn hình “thống kê”: 100](#_Toc470909289)

[k) Màn hình “thống kê khoản thu” 101](#_Toc470909290)

[l) Màn hình “thống kê khoản chi” 102](#_Toc470909291)

[CHƯƠNG 6: TỔNG KẾT 103](#_Toc470909292)

[1. Kết quả đạt được 103](#_Toc470909293)

[2. Hướng phát triển: 104](#_Toc470909294)

[3. Tài liệu tham khảo: 105](#_Toc470909295)

**Lời cảm ơn:**

Nhóm chúng em xin gởi lời cảm ơn chân thành tới các Thầy, nhờ sự hướng dẫn và chỉ bảo nhiệt tình từ Thầy đã cho chúng em những kiến thức cũng như hướng đi cho quá trình tìm hiểu và thực hiện đồ án của mình, đồng thời qua quá trình đã tích lũy thêm những kinh nghiệm quý báo và tiếp thêm động lực để chúng em có thể hoàn thành đồ án này một cách tốt hơn.

# CHƯƠNG 1: GIỚI THIỆU:

1. **Tổng quan về Mobile & Pervasive Computing:**

### Mobile Computing (Điện toán di động):

* + - 1. **Giới thiệu tổng quan:**

Điện toán di động là một công nghệ cho phép truyền tải dữ liệu, thoại và video thông qua một máy tính hoặc bất kỳ kích hoạt thiết bị không dây khác mà không cần phải được kết nối với một liên kết vật lý cố định. Các khái niệm chính liên quan đến: thông tin di động, phần cứng di động, và phần mềm di động

* **Thông tin di động:**

Các thông tin di động trong trường hợp này, đề cập đến cơ sở hạ tầng đưa ra để đảm bảo rằng thông tin liên lạc thông suốt và đáng tin cậy đi về. Đây sẽ bao gồm các thiết bị như giao thức, dịch vụ, băng thông, và Portals cần thiết để tạo điều kiện và hỗ trợ của các dịch vụ đã nêu. Định dạng dữ liệu cũng được xác định ở giai đoạn này. Điều này đảm bảo rằng không có va chạm với các hệ thống khác hiện có trong đó cung cấp các dịch vụ tương tự.



Kể từ khi các phương tiện truyền thông là không điều khiển / vô biên, cơ sở hạ tầng bao phủ là chi tiết của sóng vô tuyến theo định hướng. Đó là, các tín hiệu được thực hiện qua mạng đến các thiết bị dự định có khả năng tiếp nhận và gửi các loại tương tự của các tín hiệu.

* **Phần cứng di động**

Phần cứng điện thoại di động bao gồm các thiết bị di động hoặc các thành phần thiết bị nhận hoặc truy cập vào các dịch vụ di động. Họ sẽ dao động từ máy tính xách tay di động, điện thoại thông minh,, Personal Digital Assistants Tablet PC của.



Các thiết bị này sẽ có trung thụ thể có khả năng cảm nhận và nhận được tín hiệu. Các thiết bị này được cấu hình để hoạt động song công toàn thời, nhờ đó mà họ có khả năng gửi và nhận tín hiệu cùng một lúc. Họ không cần phải chờ đợi cho đến khi một thiết bị đã hoàn tất giao tiếp cho các thiết bị khác để bắt đầu truyền thông.

Các thiết bị nói trên sử dụng một mạng lưới hiện có và thành lập để hoạt động trên. Trong hầu hết các trường hợp, nó sẽ là một mạng không dây.

* **Phần mềm di động**

Phần mềm điện thoại di động là chương trình thực tế chạy trên các phần cứng di động. Nó đề với đặc điểm và yêu cầu của các ứng dụng di động. Đây là động cơ của thiết bị di động. Nói cách khác, nó là hệ điều hành của thiết bị đó. Nó là thành phần thiết yếu mà làm cho các thiết bị di động hoạt động.



Kể từ tính di động là yếu tố chính, loại máy tính đảm bảo rằng người dùng không bị ràng buộc hoặc gắn với một địa điểm duy nhất, nhưng có thể hoạt động ở mọi nơi. Nó sẽ kết hợp tất cả các khía cạnh của truyền thông không dây.

Hướng dẫn này sẽ xem xét các khái niệm về điện toán di động, xu hướng hiện tại và tương lai, lợi thế, các ứng dụng, và mối quan tâm an ninh lớn.

* + - 1. **Xu hướng hiện tại:**

Trong thế giới điện toán hiện nay, các công nghệ khác nhau đã đưa ra. Những đã phát triển để hỗ trợ các mạng máy tính hiện có trên toàn thế giới. Với điện thoại di động, chúng ta thấy rằng cần phải được giới hạn trong phạm vi một địa điểm vật lý đã được loại trừ. Chúng ta nghe nói về thuật ngữ như tele đi lại. Đây là việc có thể làm việc tại nhà hoặc trường nhưng đồng thời truy cập vào các nguồn tài nguyên như nếu có trong văn phòng.

Sự xuất hiện của máy tính xách tay và máy tính xách tay, Trợ lý kỹ thuật số cá nhân (PDA), máy tính máy tính bảng và điện thoại thông minh, đã lần lượt làm điện thoại di động tính toán rất thuận tiện. Tính di động của các thiết bị đảm bảo và cho phép người sử dụng để truy cập vào tất cả các dịch vụ như thể họ đang ở trong mạng nội bộ của công ty họ. Ví dụ, việc sử dụng Tablet PC và iPad. Công nghệ mới này cho phép người dùng cập nhật các văn bản, lướt Internet, gửi và nhận e-mail, truyền các file video trực tiếp, chụp ảnh và cũng hỗ trợ video và hội nghị thoại.

Nhu cầu ngày càng tăng liên tục và bao giờ cho các thiết bị thông minh cao cấp và mạnh mẽ đã được như là một chất xúc tác cho thị trường cổ phiếu. Mỗi nhà sản xuất đang cố gắng để đường cong một niche của mình trên thị trường. Các thiết bị này được phát minh và cải tiến để cung cấp hàng đầu của các ứng dụng và dịch vụ tầm. Với điện thoại di động, các nhà sản xuất khác nhau đã đến với điện thoại thông minh duy nhất có khả năng thực hiện nhiệm vụ tương tự như máy tính và với tốc độ xử lý tương tự. Thị trường cổ phiếu cho đối thủ cạnh tranh khác nhau là liên tục được chiến đấu cho. Ví dụ như các nhà sản xuất của Iphone OS của Apple, Microsoft Windows Mobile của Google Android, Blackberry OS của Research In Motion, đang liên tục cạnh tranh để cung cấp sản phẩm tốt hơn với mỗi bản phát hành.

Sự cần thiết phải tốt hơn, xách tay, giá cả phải chăng, và mạnh mẽ cũng đã khiến những nhà cung cấp phải liên tục được cải tiến. hình thị trường và thống kê cho thấy một nhu cầu ngày càng phát triển nhanh chóng để mua và sử dụng các thiết bị như vậy cho cả hai chuyên nghiệp hoặc sử dụng nhà. Kể từ khi công nghệ được thúc đẩy bởi nhu cầu của thị trường, đó là trong ánh sáng này mà các dịch vụ phù hợp cho việc thực hiện dài hạn được phát triển hoặc cải tiến. Điều này cũng đã đẩy các nhà cung cấp ngành công nghiệp khác thông qua dịch vụ sẽ cung cấp giao hàng dịch vụ tốt hơn. Ví dụ, các nhà cung cấp dịch vụ di động buộc phải cải thiện và đổi mới để chụp thuê bao nhiều hơn. Điều này có thể về dịch vụ như internet tốc độ cao và truy cập dữ liệu, thoại và video, vv do đó việc áp dụng các thế hệ khác nhau của các mạng như các dịch vụ của mạng 2G, 2,5G, 3G, 4G.

Bản chất của điện toán di động là để làm việc từ bất kỳ vị trí. Việc sử dụng iPad, Máy tính bảng, điện thoại thông minh, và ghi chú sách, có sẽ đẩy nhu cầu đối với các thiết bị này. công nhân hiện đại ngày nay có những thiết bị đó cho phép họ thực hiện công việc của mình từ sự hạn chế hoặc thoải mái của vị trí hiện tại của họ. Các thiết bị này được cấu hình để truy cập và lưu trữ một lượng lớn dữ liệu quan trọng. quản lý điều hành và đầu có thể đóng vai trò quyết định dựa trên các thông tin sẵn sàng mà không đi đến văn phòng. Ví dụ, báo cáo bán hàng và dự báo thị trường có thể được truy cập thông qua các thiết bị này hoặc cuộc họp thực hiện qua video hoặc âm thanh hội nghị qua điện thoại. Với tính năng như vậy là cao trong nhu cầu, nhà sản xuất luôn và liên tục đến với các ứng dụng hướng đến việc hỗ trợ cung cấp dịch vụ khác nhau về điện toán di động.

* + - 1. **Phân loại:**

Điện toán di động không giới hạn chỉ có điện thoại di động, nhưng cũng có những tiện ích khác nhau có sẵn trên thị trường giúp điện toán di động. Chúng thường được phân loại trong các loại sau: Trợ lý kỹ thuật số cá nhân (PDA), Điện thoại thông minh (Smartphone), Tablet PC và I-Pads.

* Trợ lý kỹ thuật số cá nhân (PDA):

Mục đích chính của thiết bị này là để hoạt động như một tổ chức điện tử hoặc ngày kế hoạch đó là xách tay, dễ sử dụng và. có khả năng chia sẻ thông tin với bạn với một hệ thống máy tính.

PDA là một phần mở rộng của máy tính, không phải là một thay thế. Những hệ thống này có khả năng chia sẻ thông tin với một hệ thống máy tính thông qua một quá trình hoặc dịch vụ được gọi là đồng bộ hóa. Trường hợp cả hai thiết bị sẽ truy cập mỗi khác để kiểm tra các thay đổi hoặc cập nhật trong các thiết bị cá nhân. Việc sử dụng các kết nối hồng ngoại và Bluetooth cho phép các thiết bị này để luôn luôn được đồng bộ.



Với các thiết bị PDA, một người sử dụng có thể; trình duyệt internet, nghe nhạc, xem video clip, chỉnh sửa và sửa các tài liệu văn phòng, và nhiều dịch vụ hơn. Họ đã có một bút stylus và một màn hình cảm ứng cho các mục đích đầu vào và đầu ra.

* Điện thoại thông minh (Smartphone)

Đây là loại điện thoại kết hợp các tính năng của một PDA với một điện thoại hoặc máy ảnh điện thoại di động. Nó có một cạnh vượt trội hơn các loại điện thoại di động.

Điện thoại thông minh có khả năng chạy nhiều chương trình cùng một lúc. Các điện thoại này bao gồm độ phân giải cao hình cảm ứng kích hoạt màn hình, các trình duyệt web có thể truy cập và đúng cách hiển thị các trang web tiêu chuẩn chứ không phải là các trang web chỉ điện thoại di động được tối ưu hóa, và truy cập dữ liệu tốc độ cao thông qua Wi-Fi và tốc độ cao băng thông rộng di động.

Các hệ thống phổ biến nhất điện thoại di động điều hành (OS) được sử dụng bởi điện thoại thông minh hiện đại bao gồm Android của Google, iOS của Apple, Symbian của Nokia, BlackBerry OS của RIM, Bada của Samsung, Windows Phone của Microsoft, và nhúng các bản phân phối Linux như Maemo và MeeGo. hệ điều hành này có thể được cài đặt trên nhiều mẫu điện thoại khác nhau, và thường mỗi thiết bị có thể nhận được nhiều thông tin cập nhật phần mềm hệ điều hành trong suốt cuộc đời của mình.



* Tablet PC và I-Pads

thiết bị di động này là lớn hơn so với một chiếc điện thoại di động hoặc một trợ lý kỹ thuật số cá nhân và tích hợp vào màn hình cảm ứng và hoạt động sử dụng chuyển động cảm ứng trên màn hình. Họ thường được kiểm soát bởi một cây bút hoặc liên lạc của một ngón tay. Chúng thường ở dạng đá và là ánh sáng trong trọng lượng. Ví dụ sẽ bao gồm; IPad, Galaxy Tab, Playbooks Blackberry vv



Họ cung cấp các chức năng tương tự như máy tính xách tay. Chúng hỗ trợ điện toán di động một cách cao hơn nhiều và có rất lớn sức mạnh xử lý ngựa. Người dùng có thể chỉnh sửa và sửa đổi các file tài liệu, truy cập internet tốc độ cao, các dòng video và dữ liệu âm thanh, nhận và gửi e-mail, thực hiện các bài giảng và thuyết trình trong số rất nhiều các chức năng khác. Họ có độ phân giải màn hình tuyệt vời và rõ ràng.

* + - 1. **Ưu điểm chính:**

Điện toán di động đã làm thay đổi cảnh quan hoàn toàn của cuộc sống con người. Sau đây là những lợi thế rõ ràng của điện toán di động: địa linh hoạt, tiết kiệm thời gian, năng suất tăng cường, dễ nghiên cứu, sự giải trí, tinh giản các quy trình kinh doanh.

* **Linh hoạt**

Điều này đã cho phép người sử dụng để làm việc từ bất cứ đâu miễn là có một kết nối được thiết lập. Một người sử dụng có thể làm việc mà không được ở một vị trí cố định. tính di động của chúng đảm bảo rằng họ có thể thực hiện nhiều nhiệm vụ cùng một lúc thực hiện công việc đã đề của họ.

* **Tiết kiệm thời gian**

Hiện tiêu thụ hoặc lãng phí bằng cách đi du lịch từ các địa điểm khác nhau hoặc đến văn phòng và trở lại, đã bị mất giá. Một bây giờ có thể truy cập vào tất cả các tài liệu quan trọng và các tập tin trên một kênh an toàn hoặc cổng thông tin và làm việc như là đã được trên máy tính của họ. Nó đã tăng cường việc từ xa ở nhiều công ty. Điều này cũng làm giảm chi phí không cần thiết mà có thể phát sinh.

* **Năng suất tăng cường**

Tính chất sản xuất đã được thúc đẩy bởi một thực tế là một công nhân chỉ có thể làm việc hiệu quả và hiệu quả từ đó bao giờ vị trí mà họ thấy thoải mái và phù hợp. Người dùng có thể làm việc với môi trường thoải mái.

* **Dễ nghiên cứu**

Nghiên cứu đã được thực hiện dễ dàng hơn, vì người dùng sẽ đi vào lĩnh vực này và tìm kiếm các sự kiện và cho chúng ăn trở lại hệ thống. Nó cũng đã làm cho nó dễ dàng hơn cho cán bộ thực địa và nghiên cứu để thu thập dữ liệu và thức ăn chăn nuôi từ bất cứ nơi nào họ mà không làm cho chuyến đi không cần thiết đến và đi từ văn phòng đến lĩnh vực này.

* **Sự giải trí**

Video và âm thanh ghi âm có thể được xem trực tiếp trên di chuyển bằng máy tính di động. Thật dễ dàng để truy cập vào một loạt các phim ảnh, tài liệu giáo dục và thông tin. Với sự cải tiến và tính sẵn sàng của các kết nối dữ liệu tốc độ cao với chi phí đáng kể, người ta có thể nhận được tất cả các giải trí mà họ muốn khi họ trình duyệt internet cho dữ liệu stream. Người ta có thể có thể xem tin tức, phim ảnh, và phim tài liệu trong Mời giải trí khác trên internet. Đây không phải là như vậy trước khi tính toán di động chợt nhận ra thế giới máy tính.

* **Tinh giản các quy trình kinh doanh**

Các quy trình kinh doanh hiện nay là dễ dàng có sẵn thông qua các kết nối an toàn. Căn cứ vào các yếu tố an toàn, biện pháp thích hợp đã được đưa ra để đảm bảo xác thực và ủy quyền của người sử dụng truy cập các dịch vụ.

Một số chức năng kinh doanh có thể được chạy trên các liên kết an toàn và cũng chia sẻ thông tin giữa các đối tác kinh doanh. Ngoài ra nó có giá trị lưu ý rằng đi du lịch dài đã được giảm, kể từ khi có việc sử dụng thoại và hội nghị video.

Các cuộc họp, hội thảo và các dịch vụ thông tin khác có thể được thực hiện bằng cách sử dụng hội nghị truyền hình và giọng nói. Điều này cắt giảm thời gian đi lại và chi tiêu.

* + - 1. **Vấn đề an ninh:**

Điện toán di động đã chia sẻ công bằng của các mối quan tâm an ninh như các công nghệ khác. Do tính chất du mục của họ, nó không phải dễ dàng để theo dõi việc sử dụng thích hợp. Người sử dụng có thể có ý định khác nhau về cách sử dụng đặc quyền này. thực hành không đúng cách và phi đạo đức như hacking, gián điệp công nghiệp, ăn cắp, lừa đảo trực tuyến và hủy diệt độc hại là một số nhưng ít người trong những vấn đề mà máy tính di động.



Một vấn đề lớn gây rắc rối cho máy tính di động là xác minh chứng. Nó không thể để cho người sử dụng người đó là rào cản thực sự. Các thành viên khác chia sẻ tên đăng nhập và mật khẩu. Đây cũng là một mối đe dọa lớn đối với an ninh. Đây là một vấn đề rất nhạy cảm, nhất công ty là rất miễn cưỡng để thực hiện tính toán di động với sự nguy hiểm của trình bày sai.

Các vấn đề về hành vi trộm cắp danh tính là rất khó khăn để chứa hoặc tiêu diệt. Các vấn đề với truy cập trái phép vào dữ liệu và thông tin của tin tặc, cũng là một vấn đề gây rắc rối. Họ truy cập để đánh cắp dữ liệu quan trọng từ các công ty. Vấn đề này đã được một nhức đầu lớn và trở ngại trong việc phủ sóng dịch vụ điện thoại di động.

Không có công ty muốn nằm mở bí mật của họ để hacker và những kẻ xâm nhập khác, người về sẽ bán cho các đối thủ cạnh tranh của họ. Nó cũng quan trọng để có những biện pháp phòng ngừa cần thiết để giảm thiểu các mối đe dọa từ diễn ra. Một số các biện pháp đó bao gồm:

* Thuê nhân viên có trình độ.
* Cài đặt phần cứng bảo mật và phần mềm.
* Giáo dục người dùng về đạo đức tính toán di động thích hợp.
* Kiểm toán và âm thanh phát triển, chính sách hiệu quả để quản lý máy tính di động.
* Thực thi các quyền và quyền truy cập thích hợp.

Đây chỉ là nhưng một số cách để giúp ngăn chặn các mối đe dọa có thể đến bất kỳ kế hoạch công ty để cung cấp điện toán di động. Vì thông tin là rất quan trọng, tất cả các biện pháp có thể cần được đánh giá và triển khai cho mục đích tự vệ.

Trong trường hợp không có các biện pháp như vậy, nó có thể cho khai thác và các mối đe dọa chưa biết khác để xâm nhập và gây hại không thể chối cãi rằng sẽ có chi phí rất lớn của các thiệt hại. Những có lẽ về mặt danh tiếng hoặc các hình phạt tài chính. Trong trường hợp như vậy, nó rất dễ bị lợi dụng trong các hoạt động phi đạo đức khác nhau.

Các vấn đề khác sẽ được bảo mật trực tuyến. Nếu yếu tố này là không đúng đã làm việc trên, nó có thể là một con đường để đe dọa thường trực. Trộm cắp và gián điệp cũng có thể là một thực tế hạn chế sử dụng đầy đủ của nó. các mối đe dọa khác nhau để bảo mật vẫn tồn tại trong việc thực hiện loại công nghệ này.

* + - 1. **Xu hướng tương lai:**

Phần này sẽ liệt kê các công nghệ di động hiện tại và tương lai bắt đầu từ công nghệ 3G là công nghệ di động hotest có sẵn trên thị trường.

* **3G:**



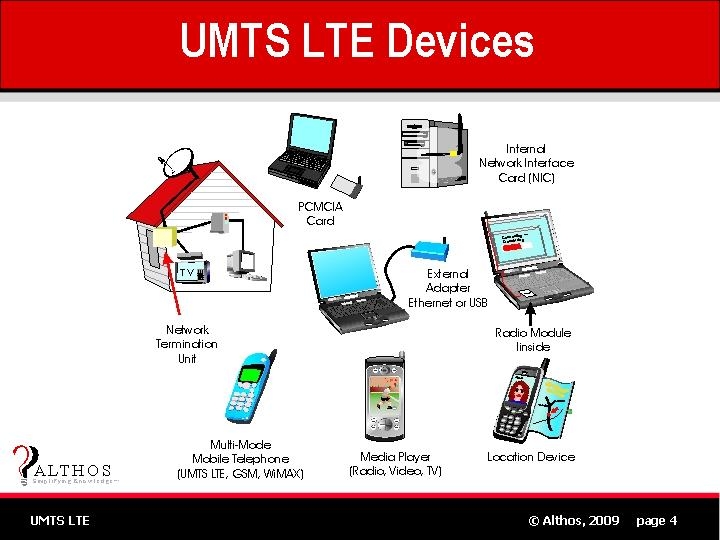
3G hoặc viễn thông di động thế hệ thứ 3 là một thế hệ các tiêu chuẩn dành cho điện thoại di động và các dịch vụ viễn thông di động thực hiện các di động quốc tế Viễn thông-2000 (IMT-2000) thông số kỹ thuật của Liên minh Viễn thông Quốc tế. dịch vụ ứng dụng bao gồm rộng vùng điện thoại bằng giọng nói không dây, truy cập Internet di động, cuộc gọi video và truyền hình di động, tất cả trong một môi trường di động.

* **GPS (hệ thống định vị toàn cầu):**



Hệ thống định vị toàn cầu (GPS) là hệ thống định vị vệ tinh trên vũ trụ cung cấp địa điểm và thời gian thông tin trong mọi thời tiết, mọi nơi trên hoặc gần Trái đất, nơi có một dòng không bị cản trở tầm nhìn đến bốn hoặc nhiều vệ tinh GPS. Các chương trình GPS cung cấp khả năng quan trọng đối với người sử dụng quân sự, dân sự, thương mại trên khắp thế giới. Ngoài ra, GPS là xương sống cho hiện đại hóa hệ thống dịch vụ không lưu toàn cầu, thời tiết, địa điểm.

* **Term Evolution Long (LTE):**



LTE là một tiêu chuẩn cho truyền thông không dây dữ liệu tốc độ cao cho điện thoại di động và các thiết bị đầu cuối dữ liệu. Nó được dựa trên GSM / EDGE và công nghệ UMTS / HSPA mạng, tăng cường năng lực và tốc độ sử dụng các kỹ thuật điều chế mới. Nó liên quan với việc thực hiện các quy định thế hệ công nghệ (4G).

* **WiMax:**



WiMAX (Worldwide Interoperability cho Microwave Access) là một tiêu chuẩn truyền thông không dây được thiết kế để cung cấp 30-40 tốc độ dữ liệu megabit mỗi giây, với những cập nhật mới nhất cung cấp lên tới 1 Gbit/s cho các trạm cố định. Nó là một phần của một thế hệ thứ tư, hoặc 4G, công nghệ không dây truyền thông. WiMax vượt xa phạm vi không dây 30 mét của một mạng Wi-Fi nội bộ thông thường (LAN), cung cấp một mạng lưới khu vực đô thị có bán kính tín hiệu trong khoảng 50 km. WiMax cung cấp tốc độ truyền dữ liệu có thể được cấp trên để cable- thông thường modem và kết nối DSL, tuy nhiên, băng thông phải được chia sẻ giữa nhiều người dùng và do đó mang lại tốc độ thấp hơn trong thực tế.

* [**NFC - Giao dịch cận truyền thông**](http://www.pcworld.com.vn/articles/cong-nghe/cong-nghe/2011/03/1223866/nfc-giao-dich-can-truyen-thong/)**:**

NFC (Near-Field Communications) là công nghệ kết nối không dây phạm vi tầm ngắn trong khoảng cách 4cm, sử dụng cảm ứng từ trường để thực hiện kết nối giữa các thiết bị khi có sự tiếp xúc trực tiếp hay để gần nhau. NFC hoạt động ở dải băng tần 13,56MHz và tốc độ truyền tải dữ liệu tối đa 424Kbps. Do khoảng cách truyền dữ liệu khá ngắn nên giao dịch qua công nghệ NFC được xem là an toàn.

NFC đã được phê chuẩn ISO/IEC vào cuối năm 2003. Tháng 3/2004, Nokia, Sony và NXP đã thành lập diễn đàn NFC để phát triển công nghệ này.

Công nghệ NFC ra đời từ sự kết hợp công nghệ nhận dạng không tiếp xúc và những công nghệ kết nối truy cập mới. NFC có 4 định dạng thẻ dựa trên các chuẩn ISO 14443 Type A, 14443 Type B và ISO 18092.

Công nghệ NFC có 2 chế độ truyền dữ liệu: chủ động (active) và thụ động (passive). Trong chế độ thụ động: thiết bị nguồn phát sẽ phát ra từ trường đến nguồn đích. Trong chế độ này, nguồn đích ở trạng thái bị động và chỉ trả lời khi nhận tín hiệu từ nguồn phát. Trong chế độ chủ động: cả thiết bị nguồn phát và thiết bị đích truyền dữ liệu bằng cách tạo ra từ trường riêng.

Hầu hết các ứng dụng hiện nay đều kết hợp cả 2 chế độ chủ động và thụ động, vì sẽ hữu ích cho các thiết bị trong việc truyền dữ liệu giữa các thiết bị không có nguồn điện, ví dụ các thẻ không tiếp xúc.

Một giao dịch diễn ra trên NFC tuần tự theo các bước: phát hiện (Discovery), xác thực (Authentication), trao đổi (Negotiation), truyền dữ liệu (Transfer) và xác nhận từ phía nhận dữ liệu (Acknowledgment).

Đối với các ứng dụng mang tính nhạy cảm cao, chẳng hạn trong trường hợp muốn tăng cường tính bảo mật trong lúc giao dịch, công nghệ NFC có thêm 2 chuẩn mã hóa: chuẩn mã hóa tiên tiến (Advanced Encryption Standard – AES) là thuật toán mã hóa khối được Mỹ áp dụng làm tiêu chuẩn mã hóa; và chuẩn mã hóa dữ liệu Triple DES (Data Encryption Standard) nghĩa là một thông tin được mã hóa DES 3 lần với 3 khóa khác nhau, do đó chiều dài mã hóa sẽ lớn hơn và an toàn hơn.

|  |
| --- |
| http://www.pcworld.com.vn/files/articles/2011/1223866/a1101-82a.jpg |

Các ứng dụng sử dụng NFC.

Công ty nghiên cứu Jupiter cho biết, trên toàn cầu sẽ có khoảng 700 triệu chiếc ĐTDĐ hỗ trợ NFC được tiêu thụ vào năm 2013. Theo công ty nghiên cứu thị trường Strategy Analytics cho biết loại hình thanh toán qua ĐTDĐ tích hợp NFC sẽ đạt 36 tỷ USD trên toàn cầu vào năm nay.

Mục đích chính dùng NFC:

- Kết nối với các thiết bị điện tử.

- Truy cập nội dung số, người dùng chỉ cần áp ĐTDĐ lên áp phích quảng cáo (có gắn thẻ từ tính – RF tag), lập tức người dùng sẽ nhận được các thông tin liên quan.

- Giao dịch không tiếp xúc, ví dụ thanh toán, mua vé.

Công nghệ NFC sẽ làm thay đổi thói quen sử dụng thiết bị của mọi người. Theo lý thuyết, có khá nhiều ứng dụng dành cho NFC, chẳng hạn như NFC được sử dụng trong điện thoại di động, camera số, TV, hay hệ thống định vị lái xe để thanh toán hóa đơn những món hàng muốn mua. Khi muốn mua sắm, người dùng có thể thanh toán bằng cách để “sát” thẻ tín dụng vào màn hình máy tính có gắn NFC. Khi chip NFC chứa thông tin thẻ tín dụng (credit card) trong điện thoại di động, ĐTDĐ của bạn sẽ "biến" thành chìa khóa, vé xem ca nhạc, thẻ lưu thông tin cá nhân... Ngoài ra, người dùng có thể biết thông tin của các chương trình giảm giá, khuyến mãi khi dùng ĐTDĐ có NFC chạm gần các áp phích quảng cáo thông minh hay chụp hình và gửi ảnh đến màn hình, máy tính hay trao đổi danh thiếp qua ĐTDĐ có hỗ trợ NFC. Với ĐTDĐ có tích hợp NFC, bạn có thể mua vé và nhận vé và quét thẻ. Sau đó, bạn có thể kiểm tra số tài khoản của mình ngay trên ĐTDĐ.

NFC mở ra một xu hướng trao đổi dữ liệu theo dạng mạng ngang hàng (P2P). Do NFC hoạt động trên tần số 13,56MHz nên không ảnh hưởng đến các công nghệ kết nối không dây khác. Khi mạng NFC được kích hoạt, người dùng vẫn có thể sử dụng các công nghệ kết nối khác như Bluetooth, Wi-Fi nếu muốn kết nối tầm xa hơn hay khi cần truyền lượng dữ liệu lớn hơn.

Công nghệ NFC hứa hẹn sẽ tiến xa và có nhiều tiềm năng hơn thẻ tần số vô tuyến (RFID tag), là nhân tố quan trọng trong việc khai thác các dịch vụ tài chính trên thiết bị di động của các nhà khai thác mạng di động (MNOs-Mobile Network Operators), các nhà thiết kế ĐTDĐ.

Mặc dù các giao thức của NFC khác nhau, nhưng vẫn được xem là linh động hơn, bảo mật hơn so với RFID hay thẻ thông minh. Sự khác biệt chủ yếu là NFC được tích hợp trong các thiết bị di động để liên kết đến các dịch vụ bán lẻ hay các giao dịch tài chính…

Thêm vào đó, tính năng bảo mật khi truyền dữ liệu là vô cùng quan trọng, do đó NFC trên các thiết bị di động không chỉ đơn giản là một công nghệ hay ứng dụng mà là một “hệ sinh thái”. NFC trên thiết bị di động được xác định là nhân tố quan trọng cho các giải pháp thanh toán của Amex, Mastercard và VISA. Việc triển khai NFC đang thực hiện ở châu Âu, tương tự công nghệ không tiếp xúc đang được triển khai rộng rãi trong hệ thống phương tiện di chuyển công cộng trên khắp thế giới. Một ĐTDĐ tích hợp một thiết bị bảo mật điện tử tạo mật khẩu ngẫu nhiên dựa trên phần cứng, đó có thể là thẻ mạch tích hợp toàn cầu (Universal Integrated Circuit Card-UICC) – là nền tảng lý tưởng cho các ứng dụng NFC. UICC chạy ứng dụng SIM trong mạng GSM và ứng dụng USIM trong mạng 3G/UMTS. Mỗi UICC có thể bao gồm nhiều ứng dụng, cho phép truy cập cả 2 mạng 2G và 3G, lưu danh bạ và chạy những ứng dụng khác.

Đến nay có khoảng 14 nhà khai thác mạng di động đã cùng nhau phát triển các ứng dụng NFC mà thị phần của 14 nhà khai thác này chiếm đến 40% thị trường di động toàn cầu. Đó là Bouygues Tlcom, China Mobile, Cingular Wireless, KPN, Mobilkom Austria, Orange, SFR, SK Telecom, Telefonica Mviles Espaa, Telenor, TeliaSonera, Telecom Italia Mobile (TIM), Vodafone và 3.

Sự khác biệt:

|  |
| --- |
| **NFC có khác biệt gì so với các công nghệ không dây khác?** |
| **▪ Bluetooth**là công nghệ không dây được thiết kế để truyền dữ liệu đến các thiết bị như ĐTDĐ, máy tính xách tay và các thiết bị khác trong phạm vi 10 mét.  **▪ Công nghệ Wi-Fi** dành cho mạng nội bộ (LAN) cho phép mở rộng hệ thống mạng hay thay thế cho mạng có dây trong phạm vi khoảng 100 mét.   **▪ ZigBee**là công nghệ không dây có khả năng kiểm soát và giám sát các ứng dụng trong ngành công nghiệp và khu nhà ở trong phạm vi hơn 100 mét.   **▪ IrDA**là chuẩn liên lạc không dây tầm ngắn (<1 mét), truyền dữ liệu qua tia hồng ngoại. Giao diện IrDA thường được dùng ở các máy tính và ĐTDĐ.   **▪ RFID**là phương thức nhận dạng tự động, cho phép lưu trữ và lấy dữ liệu từ xa dựa vào thẻ nhãn tần số vô tuyến. Thẻ RFID được gắn kèm vào sản phẩm. Hệ thống RFID gồm hai thành phần: thẻ nhãn có gắn chip silicon cùng ăng-ten và phần thứ 2 là bộ đọc giao tiếp với thẻ nhãn và truyền dữ liệu tới hệ thống máy tính trung tâm.   **▪ Thẻ thông minh**không tiếp xúc (contactless smart card) tích hợp trong chip để truyền dữ liệu đến đầu đọc nhờ công nghệ RFID. Ví dụ, thẻ thông minh truyền dữ liệu qua chuẩn ISO/IEC 14443 và FeliCa trong phạm vi khoảng 10cm. |

### *Pervasive Computing (Ubiquitous Computing – Điện toán phân tán rộng khắp):*

**Khái niệm:**

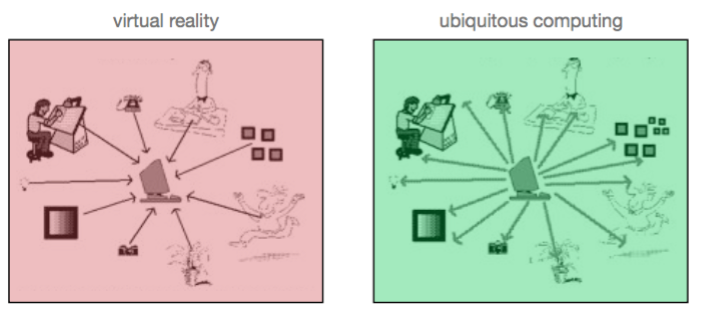
Ubicomp là một khái niệm trong công nghệ phần mềm và khoa học máy tính, ở đó máy tính được làm ra để xuất hiện mọi lúc mọi nơi. Ngược với máy tính để bàn, Ubicomp có thể sử dụng ở thiết bị, vị trí, định dạng bất kỳ. Người dùng có thể tương tác với máy tính, mà có thể tồn tại ở nhiều dạng khác nhau, bao gồm máy tính xách tay, máy tính bảng và thiết bị đầu cuối trong các đối tượng hàng ngày như tủ lạnh hay một cặp kính. Các công nghệ cơ bản để hỗ trợ Ubicomp bao gồm Internet, advance middleware, hệ điều hành, mobile code, cảm biến, vi xử lý, I/O và UI, và nhiều thứ khác nữa.

Ubicomp liên quan đến rất nhiều đề tài nguyên cứu, bao gồm điện toán phân tán (distributed computing), điện toán di dộng (mobile computing), điện toán vị trí (location computing), mạng di động (mobile networking), máy tính nhận biết ngữ cảnh (context-aware computing), mạng cảm biến (sensor computing), giao tiếp người máy (HCI) và trí tuệ nhân tạo (artificial intelligence).

**Ubiquitous Computing và Virtual Reality:**

Ubiquitous computing ngược lại với Virtual Reality

|  |  |
| --- | --- |
| Ubiquitous Computing | Virtual Reality |
| Tập trung vào việc đưa máy tính mà môi trường thực tế với con người | Tập trung vào việc đưa con nguòi vào môi trường ảo với máy tính |
| Rất khó để tích hợp con người, khoa học máy tính, kỹ thuật máy tính và khoa học xã hội lại với nhau. |  |



**Lịch sử:**

* Mark Weiser đặt ra cụm từ "Ubiquitous Computing" khoảng năm 1988, ở Trung tâm nghiên cứu Xerox Palo Alto (PARC).
* John Seely Brown và Weiser đã viết một số bài báo đầu tiên về đề tài này, chủ yếu là xác định nó và phác thảo ra mối quan tâm lớn của nó.
* Andy Hopper từ Đại học Cambridge Vương quốc Anh đề xuất và chứng minh khái niệm "Teleporting - Dịch chuyển" - nơi các ứng dụng theo dõi người sử dụng bất cứ nơi nào họ di chuyển.
* Bill Schilit (tại Google) cũng đã làm việc trước đó trên chủ đề này, và tham gia các hội thảo Computing Mobile lần đầu tổ chức tại Santa Cruz vào năm 1996.
* Tiến sĩ Ken Sakamura của Đại học Tokyo, Nhật Bản đừng đầu phòng thí nghiệm Ubiquitous Networking (UNL), Tokyo cũng như T-Engine Forum. Mục tiêu chung của Ubiquitous Networking Sakamura và diễn đàn T-Engine, là cho phép bất kỳ thiết bị hàng ngày có thể phát sóng và nhận được thông tin.
* Roy Want, một nhà nghiên cứu và sinh viên làm việc cho Andy Hopper tại Đại học Cambridge, làm việc trên " hệ thống Active Badge", là một hệ thống tính toán vị trí tiên tiến, nơi mà cá nhân được sáp nhập với máy tính.
* MIT cũng đã góp phần nghiên cứu quan trọng trong lĩnh vực này, đáng chú ý là Things That Think tại Media Lab.

**Các khái niệm chính:**

Ubicomp có mặt ở khắp mọi nơi, bao gồm nhiều lớp, mỗi lớp có vai trò riêng, kết hợp với nhau thành một hệ thống hoàn chỉnh.

* Lớp 1: Task management layer
  + Giám sát công việc, nội dung, chỉ số của người dùng.
  + Kết nối công việc của người dùng với các dịch vụ trong môi trường.
  + Quản lý các phụ thuộc phức tạp.
* Lớp 2: Environment management layer
  + Giám sát tài nguyên và khả năng của nó.
  + Kết nối các dịch vụ cần thiết với tình trạng xác định của khả năng người dùng.
* Lớp 3: Environment layer
  + Giám sát các tài nguyên liên quan.
  + Quản lý độ tin cậy của tài nguyên.

**Các vấn đề của Ubicomp:**

* Làm thế nào để tích hợp truyền thông di động với cơ sở hạ tầng thông tin phức tạp?
* Lợi ích mà chúng mang lại trong công việc và giải trí?
* Tính riêng tư.
* Làm thế nào để phát triển và quản lý hệ thống nhận biết ngữ cảnh?
* Những sự hỗ trợ nào là cần thiết trong mạng?
* Nguồn cung cấp.
  1. **Giới thiệu về đồ án:**
     + 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

* + - 1. ***Đối tượng và mục tiêu sử dụng:***

Áp dụng những kiến thức đã học trong môn học, nhóm sẽ thực hiện đồ án là App quản lý chi tiêu cho cá nhân.

Đối tượng sử dụng: Các cá nhân có nhu cầu muốn quản lý tài chính của bản thân trong một thời gian nhất đinh, đồng thời có thể xem các biểu đồ chức năng về thu chi.

Mục tiêu: Tạo ra một ứng dụng giúp mọi người có thể quản lý thu nhập và các chi phí của mình một cách thuận tiện và dễ dàng sử dụng.

# CHƯƠNG 2: SƠ LƯỢC VỀ HỆ ĐIỀU HÀNH

* + 1. **Sơ lược về 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. **Giới thiệu 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.

1. **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.



Các thành viên của liên minh di động mở.

1. **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.

1. **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.

1. **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…

1. **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…

## 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

### *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. ***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. ***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. ***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.

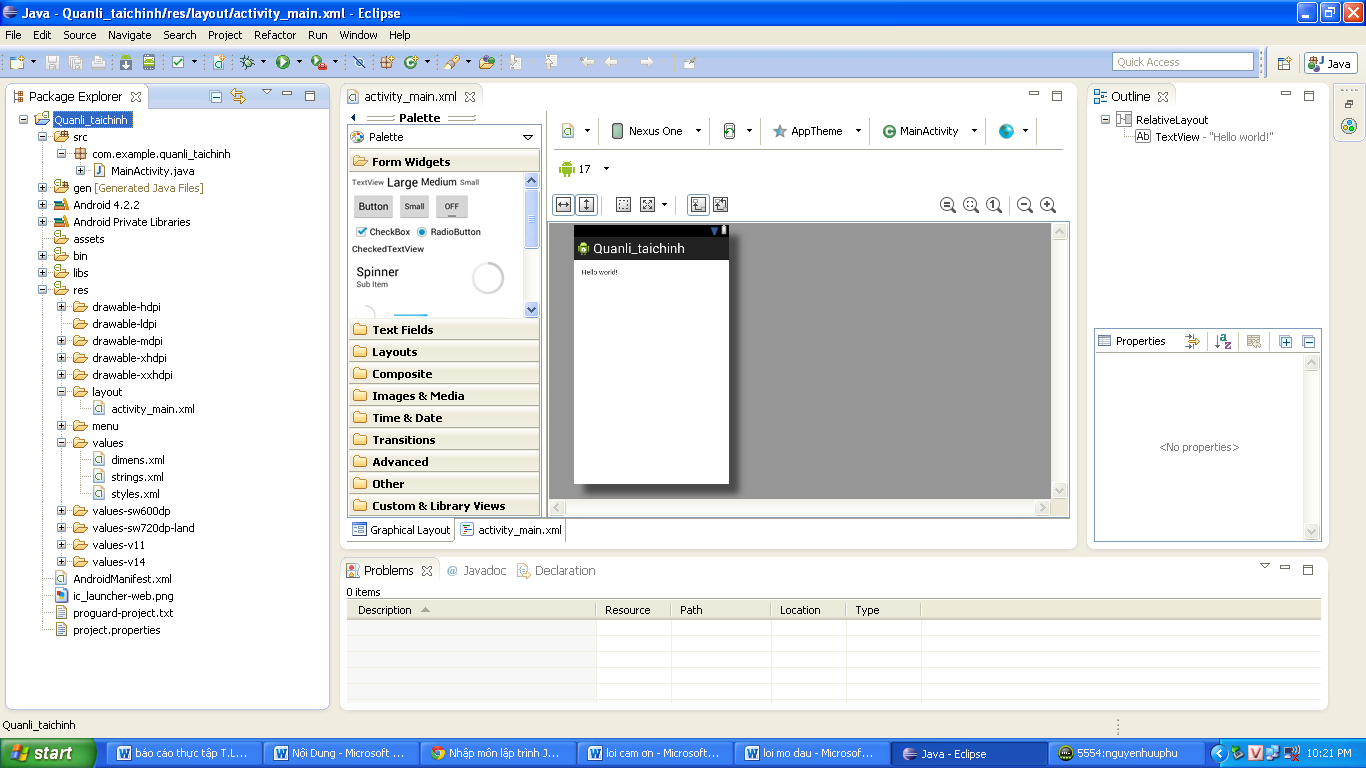
*2.4.5. Linux kernel*

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.

## 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.



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)

1. **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.

1. **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:

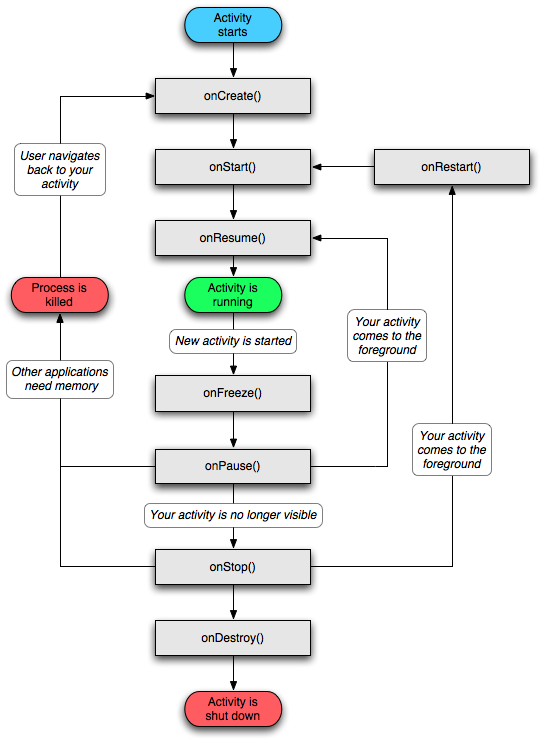
* **onCreate(Bundle)** - Nơi khởi tạo activity. Quan trọng hơn, đây chính người lập trình gọi setContentView(int) kèm theo layout để thể hiện UI của riêng mình. Đồng thời còn có findViewById(int) giúp gọi các widget (buttons, text boxes, labels,..) để dùng trong UI.
* **onPause()** - Nơi giải quyết sự 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 nếu bị ẩn.*

**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.*

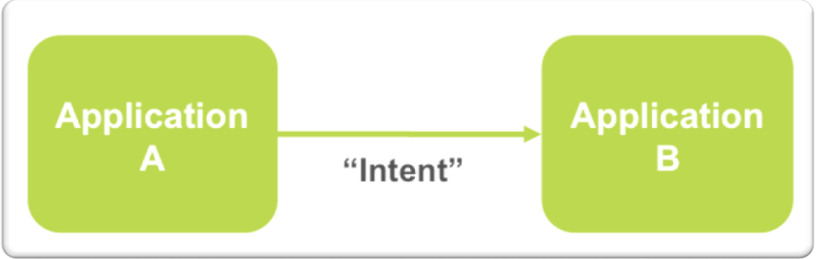
**}**

* **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:

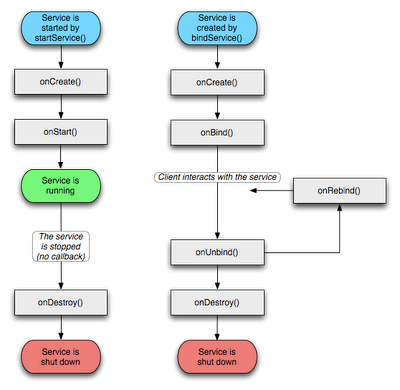
Là một chức năng để ứng dụng thông báo với hệ thống về việc mà nó muốn thực hiện ở phía dưới nền (ngay cả khi người dùng không tương tác trực tiếp tới ứng dụng). Điều này tương ứng với việc gọi phương thức Context.startservice(), nó sẽ yêu cầu hệ thống lên lịch cho service để chạy cho đến khi bị chính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

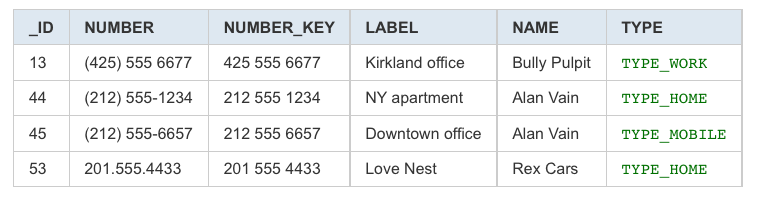
Cách thức hoạt động của Content Provider:

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.

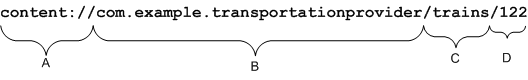
URI (Uniform Resource Identifier)

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>)

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.

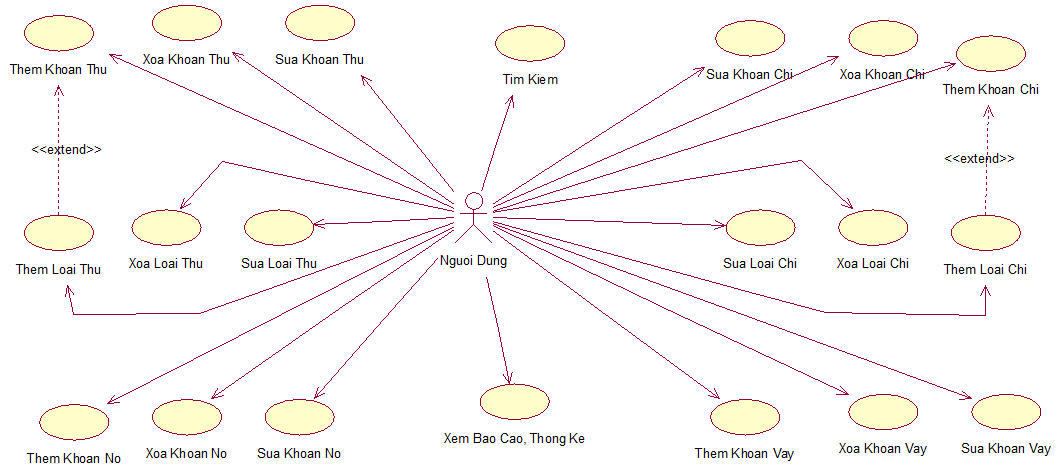
+ Khởi động máy ảo xong, chọn nút DDMS ở góc trên bên phải chương trình Eclipse để xem cấu trúc file.

+ 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 3: MÔ TẢ SƠ ĐỒ USECASE:

## Sơ đồ Use-case:

**

*Hình 2.1 Sơ đồ Use-case*

## Danh sách các Actor:

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

## 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 |

## 4. Đặc tả Use-case

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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.

### *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 :**

### *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.

### *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.

### *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.

### *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.

### *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.

### *q) 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 :**

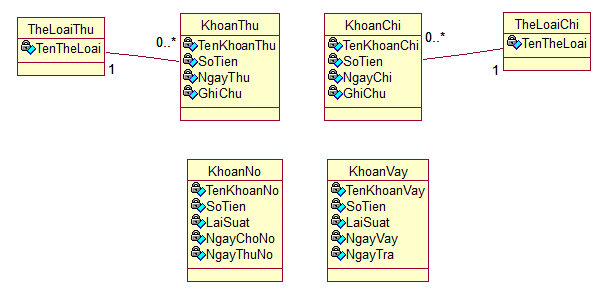
1. Ở giao diện chính người dùng nhấp vào thống kê.
2. Giao diện sẽ chuyển sang bảng thống kê. Người dùng có thể chọn cách hiển thị thống kê theo dữ liệu hay theo biểu đồ.

* **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 4: SƠ ĐỒ PHÂN TÍCH:

## Sơ đồ lớp:

### *Sơ đồ lớp:*

**

*Hình 3.1 Sơ đồ lớp*

### *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*

### *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*

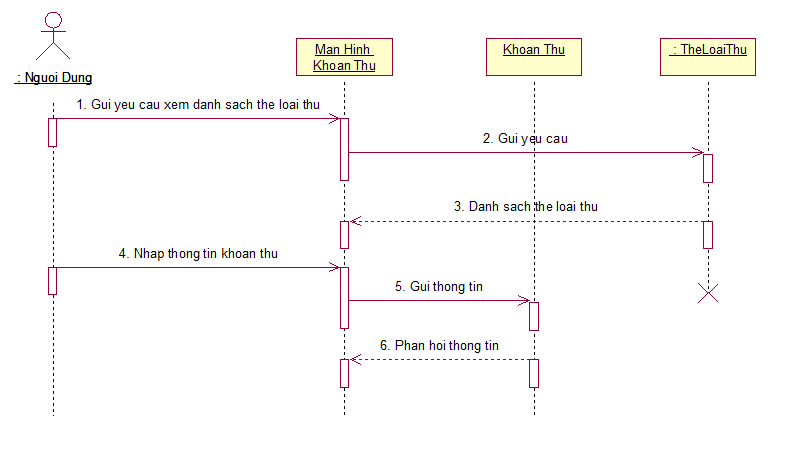
* Lớp KhoanVay (Khoản Vay)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

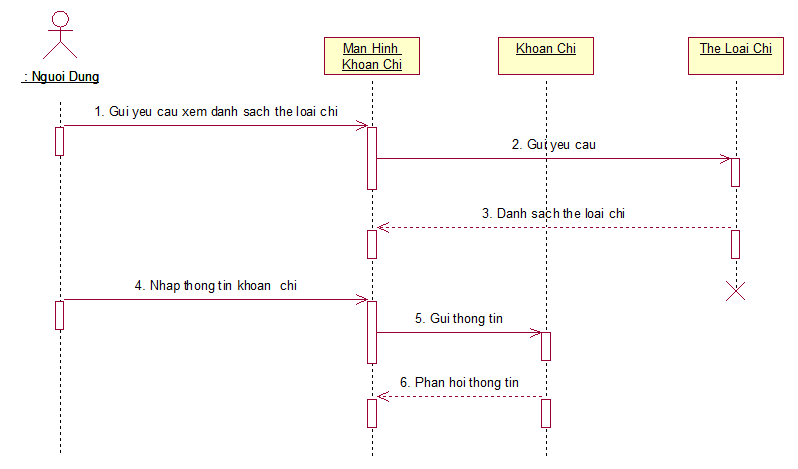
* + 1. ***Thêm khoản thu:***



*Hình 3.2 Sơ đồ tuần tự thêm khoản thu*

Mô tả các bước:

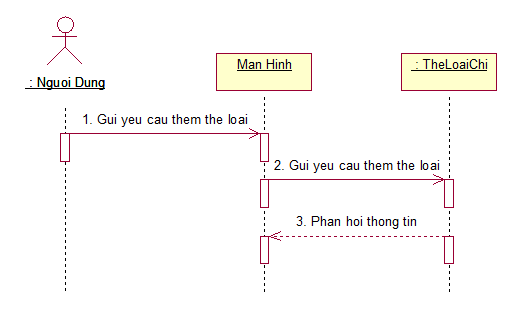
1. Gửi yêu cầu xem danh sách thể loại thu.
2. Gửi yêu cầu xem thông tin.
3. Trả về danh sách thể loại thu.
4. Nhập thông tin khoản thu.
5. Gửi thông tin khoản thu.
6. Phản hồi kết quả.
   * 1. *Thêm khoản chi*



*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 5: THIẾT KẾ GIAO DIỆN

* 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*

## Mô tả chi tiết mỗi màn hình:

### *Màn hình “Màn hình khởi động”:*



*Màn hình khởi động*

Cách sử dụng là chọn từng thể loại mà mình muốn

1. *Màn hình “Thêm khoản chi”:*



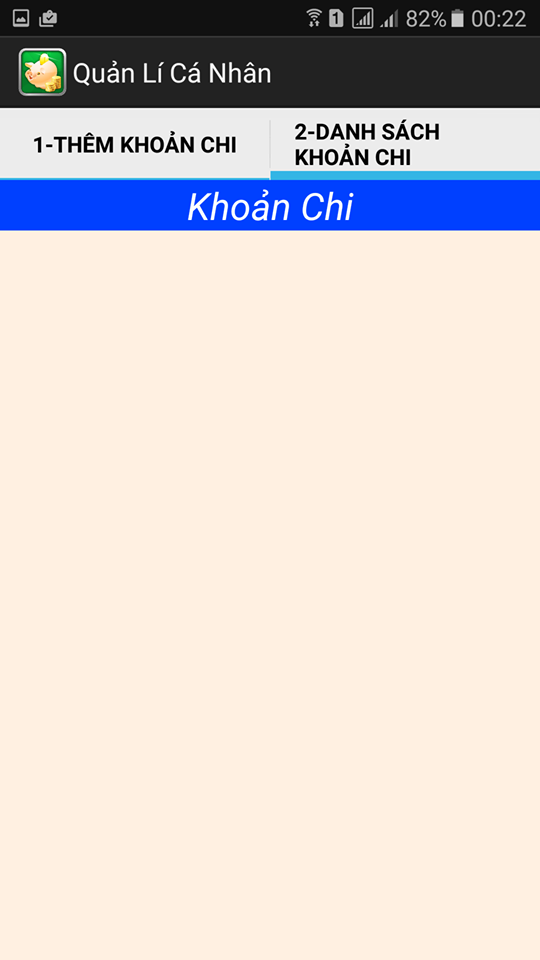
*Màn hình thêm khoản chi*

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.

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”.
* 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.

1. ***Màn hình “Thêm khoản thu”***



*Màn hình thêm khoản thu*

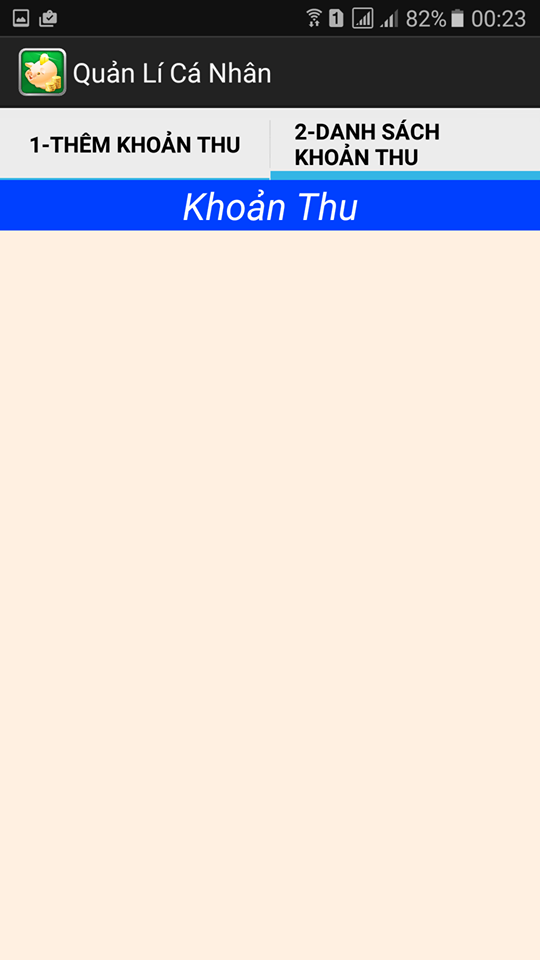
Các thành phần của giao diện:

* 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”.

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



*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”.
* 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.

1. ***Màn hình “Thể loại”:***



*Màn hình thể loại*

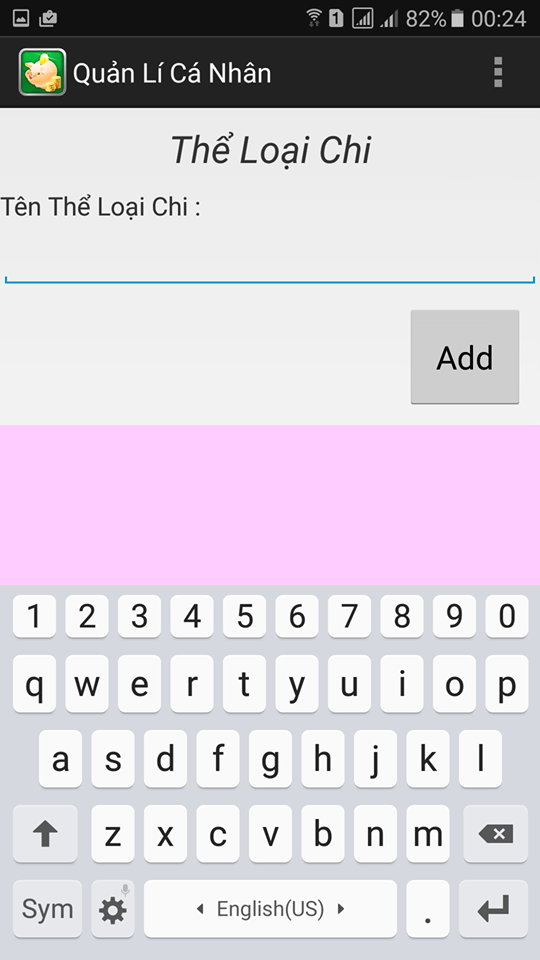
**Thành phần giao diện:**

* 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.

1. **Màn hình “Thể loại chi”:**



*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.

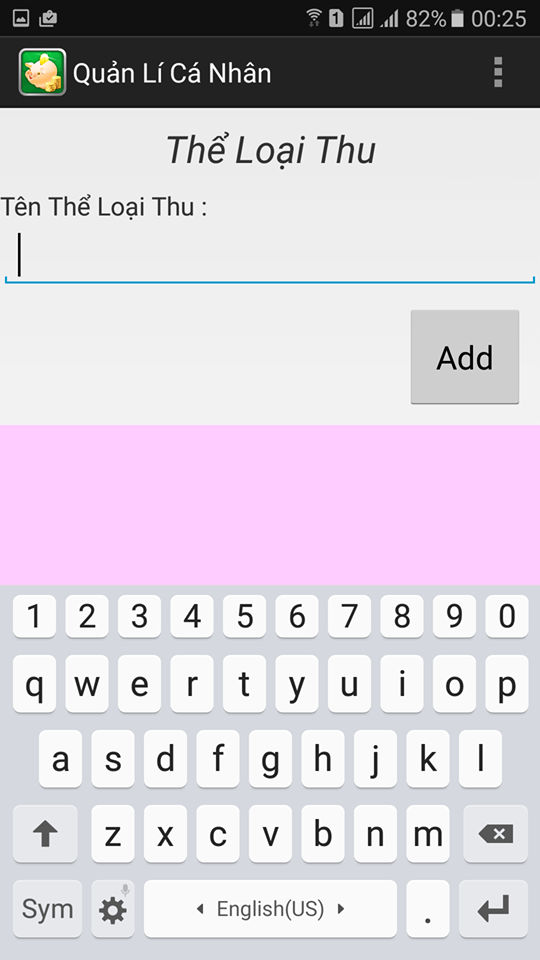
**Xử lý:**

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”.

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.

1. ***Màn hình “Thể loại thu”:***



*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”.

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.

1. ***Màn hình “Thêm khoản vay”***



*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.

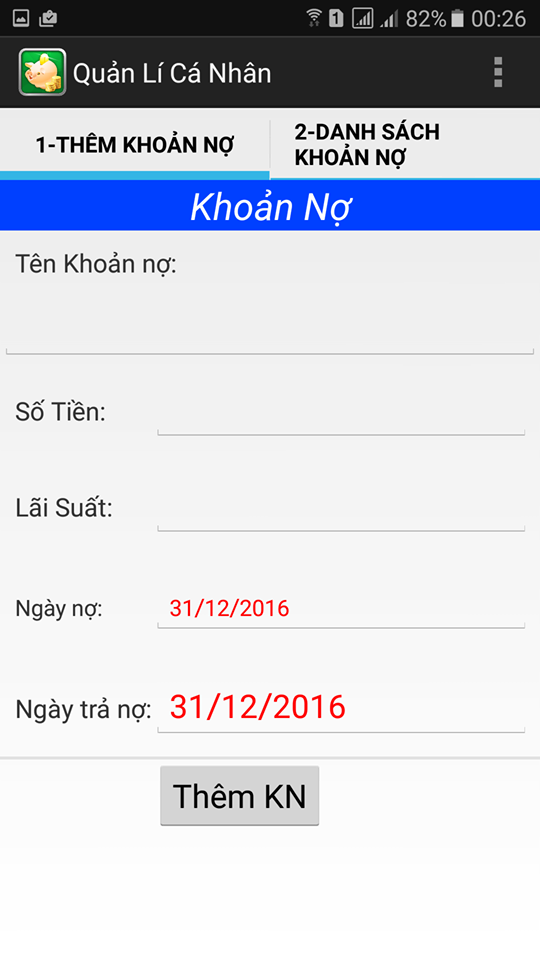
**Thành phần giao diện:**

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 .
* 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.

1. ***Màn hình “Thêm khoản nợ”***



*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.

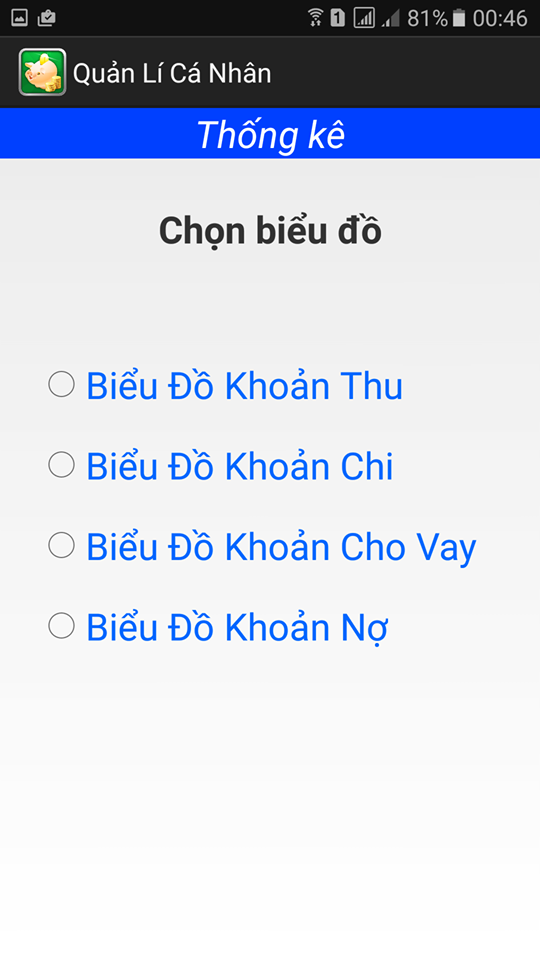
**Thành phần giao diện:**

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 .
* 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.

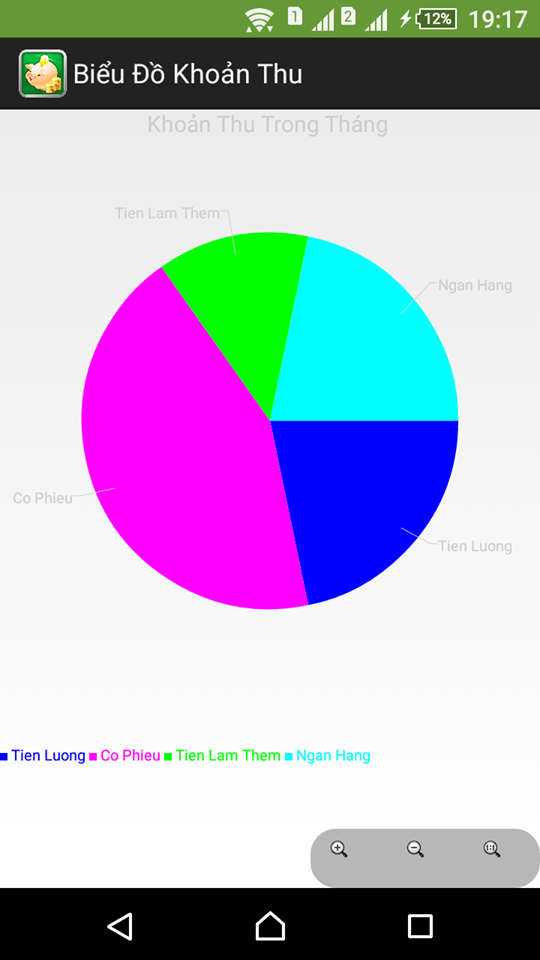
### *Màn hình “thống kê”:*



*Màn hình thống kê*

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

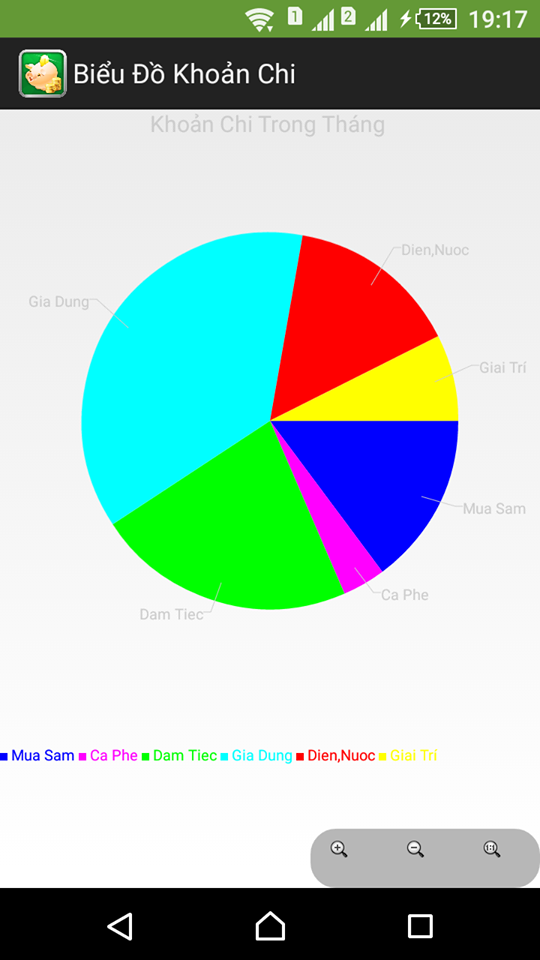
### *Màn hình “thống kê khoản thu”*



*Màn hình thống kê khoản thu*

* Thống kê khoản thu theo biểu đò tròn.

### *Màn hình “thống kê khoản chi”*



*Màn hình thống kê khoản chi*

* Thống kê khoản chi theo biểu đò tròn.

# CHƯƠNG 6: TỔNG KẾT

* 1. **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.

## Hướng phát triển:

* Hoàn chỉnh các chức năng tìm kiếm.
* Xử lý thiết lập thay đổi ngôn ngữ, thay đổi hình nền.
* Thống kê hoàn thiện và có thể so sánh các mục thu chi.
* 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