**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP CHUYÊN NGÀNH**

**BỘ MÔN : CÔNG NGHỆ THÔNG TIN**

***ĐỀ TÀI:***

***Tìm hiểu về lập trình di động, và viết ứng dụng quản lí thu chi cá nhân.***

*Giảng viên hướng dẫn: GV.MAI CƯỜNG THỌ*

**Nha Trang, tháng 6/2016**

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

|  |  |
| --- | --- |
| TÌM HIỂU VỀ LẬP TRÌNH DI ĐỘNG VÀ  VIẾT ỨNG DỤNG QUẢN LÝ THU CHI CÁ NHÂN | PHẦN 1  **Mở Đầu** |
|  | |

## Lý do chọn đề tài

Hiện nay, thiết bị di động – smart phone ngày càng phổ biến. Việc đưa một ứng dụng lên các kho lưu trữ cũng tương đối dễ dàng. Hơn nữa,người sử dụng cũng sẵn sàng chi trả một khoản tiền phù hợp cho những ứng dụng mà họ cảm thấy cần thiết hơn so với trước đây.

Theo số liệu thống kê của IDC Việt Nam, năm 2012 hệ điều hành adroid chiếm tới hơn 79% thị phần smart phone tại Việt Nam, số lượng người sử dụng nhiều cũng đồng nghĩa với việc ứng dụng sẽ dễ dàng tiếp cận người sử dụng hơn.

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

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

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

## Mục đích nghiên cứu.

Em hi vọng rằng qua thời gian nghiên cứu em có dịp tiếp cận với hệ điều hành android, một số ngôn ngữ mới.

## Đối tượng nghiên cứu.

Với đề tài này em sẽ tìm hiểu về hệ điều hành android, ngôn ngữ Java và viết được ứng dụng trên nền tản android, tìm hiểu về cách đánh giá ngôn ngữ Java và viết code trên nền tảng android. Hiểu được phương pháp để xây dựng một phần mềm như thế nào cho hợp lí với thị trường. Hiểu về vấn đề quản lý mã nguồn và làm việc nhóm với GIT

Sau đó em sẽ nghiên cứu cụ thể vào ngôn ngữ Java được xem là nền tản của cuộc cách mạng lập trình .

|  |  |
| --- | --- |
| TÌM HIỂU VỀ LẬP TRÌNH DI ĐỘNG VÀ  VIẾT ỨNG DỤNG QUẢN LÝ THU CHI CÁ NHÂN | PHẦN 2  **Tổng Quan** |
|  | |

## Chương 1: Tìm hiểu về lập trình cho thiết bị android

## Tổng quan về lập trình di đông.

Ngày nay, với sự bùng nổ dân số, cùng với sự phát triển công nghệ thông tin như vũ bão, máy tính hay laptop dần được con người thu nhỏ lại nhưng vẫn đảm bảo có đầy đủ các chức năng như một máy tính cá nhân. Vì vậy, các dòng điện thoại smartphone lần lượt ra đời, có thể kể đến một số thiết bị di động nổi tiếng như IPhone 7 xài hệ điều hành iOS của hãng Apple, Samsung Galaxy S8 xài hệ điều hành Android của hãng Samsung, Nokia Lumia 930 xài hệ điều hành Windows Phone của Microsoft, …. Và để tăng số lượng người sử dụng smartphone, download ứng dụng từ AppStore thì các hãng thiết bị di động nổi tiếng trên vô Công nghệ thông tin cùng khuyến khích các nhà lập trình viên lên ý tưởng và làm ra một ứng dụng dựa trên ý tưởng của mình. Từ đó, môn học lập trình trên thiết bị di động (TBDĐ) ra đời để các nhà lập trình viên tạo ra được nhiều ứng dụng đa dạng đáp ứng được nhu cầu của nhiều tầng lớp người sử dụng.

## Các đặc điểm của lập trình di động.

### Dễ tiếp cận, dễ tìm hiểu và dễ học.

### Giúp lập trình viên tạo ra ứng dụng cho người sử dụng.

### Giúp hiện thực hóa ý tưởng của lập trình viên.

### Giúp lập trình viên tạo ra ứng dụng là cầu nối giao tiếp với mọi người trên thế giới qua số lượng người download và sử dụng ứng dụng của mình.

## Vai trò của lập trình di động.

Bạn sẽ luôn được tiếp cận với những tri thức mới.Bạn có thể thấy những kiến thức, những công nghệ của vài năm trước đây đã hoàn toàn lỗi thời so với hiện tại.Làm việc trong ngành này, bạn sẽ luôn được nắm bắt những tri thức mới nhất, công nghệ hiện đại nhất của nhân loại. Nếu bạn là người say mê khám phá và ưa sự mới mẻ, bạn sẽ không bao giờ cảm thấy nhàm chán. LTDĐ là một lĩnh vực đầy năng động và sáng tạo. Phần lớn các nhân viên làm việc trong lĩnh vực LTDĐ đều còn rất trẻ, đầy tài năng, hoài bão và khát vọng.Làm việc trong một cộng đồng như thế, bạn có thể phát huy hết những tiềm năng và năng lực vốn có của bản thân. Đây sẽ là điều kiện thuận lợi giúp bạn thể hiện tối đa óc sáng tạo. Bạn có nhiều thách thức và cơ hội để khẳng định mình. LTDĐ là một trong những nghề có tính cạnh tranh gay gắt và tính đào thải khốc liệt. Bởi đây là lĩnh vực phát triển với tốc độ nhanh nhất và quy tụ nhiều nhất những trí tuệ siêu việt trên thế giới.Tuy nhiên, nếu bạn là người tài năng và có hoài bão, bạn có thể vượt qua tất cả. Hầu hết những nhân vật nổi tiếng trong ngành Công nghệ thông tin đều khởi đầu từ hai bàn tay trắng, nhưng ngày nay họ được cả thế giới ngưỡng mộ.

## Xu hướng công nghê tương lai.

Di dộng đang và sẽ trở thành xu hướng của tương lai. Có rất nhiều cách cho các lập trình viên có thể phát triển các ứng dụng trên di động, từ việc thiết kế các website tối ưu cho di động (web app), phát triển ứng dụng lai dựa trên HTML (hybrid app) cho đến viết các ứng dụng gốc cho nền tảng (native app). Mặc dù có nhiều phương pháp để phát triển một ứng dụng cho di động nhưng chúng đều có một điểm chung đó là chạy trên mã gốc của một nền tảng nhất định. Do đó, khi một tổ chức muốn phát triển ứng dụng chạy trên một nền tảng nào đó, họ sẽ thường tuyển dụng các ứng viên có kiến thức chuyên sâu về nền tảng được yêu cầu, hiểu rõ các lớp, các thành phần của nền tảng dù cho ứng dụng có được phát triển theo kỹ thuật nào đi chăng nữa. Nói về quy trình phát triển phần mềm, thuật ngữ này không hẳn chỉ dành cho các quản lý dự án như nhiều người vẫn nghĩ. Một lập trình viên cũng cần phải hiểu được quá trình phát triển của một phần mềm như thế nào, theo dõi các tác vụ, tiến độ ra sao, làm việc với các lập trình viên khác như thế nào thì hiệu quả… Kể cả khi một lập trình viên "chiến" một mình một dự án thì cũng cần hiểu rõ về vấn đề này. Agile là một process giúp cho việc phát triển phần mềm được nhanh gọn và linh hoạt hơn do đó, nếu như các developer nắm bắt được process này và áp dụng một cách hiệu quả, quá trình phát triển phần mềm sẽ được rút ngắn và tinh gọn đi rất nhiều. Quy trình phát triển phần mềm nhanh gọn (agile) có rất nhiều phương pháp khác nhau như Scrum, Kanban, XP…và các lập trình viên cần chọn cho dự án của mình một phương pháp phù hợp dựa trên các tiêu chí đánh giá về dự án. Các bộ công cụ được cung cấp để có thể phát triển theo hướng agile một cách hiệu quả cũng rất nhiều, có thể kể đến như Pivotal hay Trello, giúp cho việc phát triển phần mềm được rõ ràng, nhanh gọn hơn so với các phương pháp truyền thống.

## Cơ hội nghề nghiệp về lập trình di động.

Câu trả lời là bất kỳ nơi nào bạn muốn, trong các công ty, xí nghiệp, trường học, bệnh viện, các tổ chức, hiệp hội... Thậm chí, bạn có thể là một chuyên gia Công nghệ thông tin tự do, hình mẫu ngày càng được ưa thích, làm việc độc lập, không phụ thuộc vào bất kỳ cơ quan, tổ chức nào. Bạn cũng có thể cùng một số đồng nghiệp khác lập ra một nhóm hay một công ty của riêng mình. Tóm lại, công nghệ thông tin là một ngành nghề phát huy tối đa khả năng sáng tạo và tự chủ của bạn.

Nhìn chung, nếu muốn lựa chọn công tác ở những địa chỉ chuyên nghiệp về Công nghệ thông tin, bạn sẽ làm việc tại:

### Các công ty phần mềm: Các công ty này nghiên cứu, xây dựng, phát triển và cung cấp các phần mềm, các ứng dụng, xây dựng website, games... cho thị trường. Hiện nay, phát triển phần mềm là lĩnh vực Công nghệ thông tin mạnh nhất ở Việt Nam và thu hút lực lượng nhân lực tham gia đông đảo nhất. Vì vậy, bạn có thể dễ dàng tìm kiếm và phát triển khả năng của mình trong lĩnh vực phần mềm.

### Các công ty sản xuất, lắp ráp, sửa chữa trang thiết bị phần cứng: Tuy chưa đạt được ưu thế như những công ty phần mềm, nhưng sẽ là sai lầm nếu bạn hoàn toàn thờ ơ với các công ty sản xuất, lắp ráp và sửa chữa thiết bị phần cứng. Nó thực sự hứa hẹn cả một nền công nghiệp hùng mạnh trong tương lai. Ngay bây giờ, không ít chuyên gia Công nghệ thông tin trong chúng ta đang nỗ lực cho ra đời và ngày một hoàn thiện những chiếc máy tính mang thương hiệu Việt Nam.

### Các công ty cung cấp giải pháp tích hợp Các công ty này chuyên thiết kế, triển khai các giải pháp tích hợp về phần cứng và phần mềm. Đây tất nhiên cũng sẽ là một lĩnh vực mới hứa hẹn đầy thách thức và cả thành công trong tương lai. Cùng với sự phát triển mạnh mẽ của Công nghệ thông tin tại Việt Nam, chúng ta đang ngày càng làm chủ tốt hơn lĩnh vực này. Vì vậy có rất nhiều cơ sở để bạn ngay từ bây giờ bắt tay vào tìm hiểu về các công ty cung cấp giải pháp tích hợp.

### Các công ty cung cấp giải pháp về mạng và an ninh mạng: Với sự phát triển “thần tốc” của Internet tại Việt Nam và cả những vấn đề về an ninh mạng như virus, hacker... lĩnh vực này đầy tiềm năng phát triển và đang mở ra những cơ hội lớn cho sự nghiệp của bạn.

## Những nền tảng di động phổ biến hiện nay.

Hệ điều hành trên điện thoại thông minh hay máy tính bảng được xem là bí quyết quyết định thành công của các nhà sản xuất. Song dù thời gian có khác nhau , thì mỗi hệ điều hành đều có ưu và khuyết điểm riêng , ảnh hưởng trực tiếp đến việc lựa chọn của người dùng .

Trên thị trường ứng dụng di động hiện nay, 3 hệ điều hành chiếm thị phần cao nhất là : Android, iOS và Window phone, tiếp sau là 1 số hệ điều hành khác như BlackBerry…

|  |  |  |
| --- | --- | --- |
| Tên hệ điều hành trên điện thoại | Ưu điểm | Nhược điểm |
| Andorid ( Google) | * Tuỳ biến nhanh chóng * Dễ làm việc * Ứng dụng hổ trợ phong phú * Tương thích với đa cấu hình phần cứng liên tục được cập nhật biên phản mới. | * Bị phân mảnh (quá nhiều phiên bản ) khiến cho việc nâng cấp lên phiên bản mới khó được thực hiện đồng nhất trên nhiều thiết bị * Đối với từng hãng có điều chĩnh riêng, khiến người dùng bối rối khi thay đổi thiết bị, * Android có quá nhiều lổ hổng bảo mật . |
| ISO (Apple) | * Giao diện thiết kế đẹp , tinh tế * Bảo mật cao * Kho ứng dụng vô cùng phong phú * Sau khi nâng cấp , luôn khiến ngừoi dùng phải ngạc nhiên , thích thú với quà nhiều thay đổi | * Thiết lập đồng bộ với iTunes gây khó khăn với mới tiếp cận iOS * Nếu không am hiểu , bạn sẽ mất thời gian trong việc đồng bộ : danh bạ chép ứng dụng nhạc , phim từ máy tính vào điện thoại |
| Windows Phone (Microsoft) | * Tính đồng bộ với máy tính cao ( vì sử dụng các dịch vụ của Microsoft ) * Giao diện Metro độc đáo. * Hỗ trợ công việc với các tiện ích văn phòng mạng mẽ. * Cũng rất dễ tiếp cận và sử dụng. | * Quá cũ kĩ ( ảnh hưỡng từ thế hệ thiết bị Windows Mobile ) * Chỉ hấp đẫn doanh nghiệp chứ không hấp dẫn người sử dụng cá nhân * Giao diện Metro không đẹp * Chợ ứng dụng chưa thật sự đa dạng. * Chi phí cho ứng dụng qua cao. |

## Giới thiệu về lập trình di động android.

## Giới thiệu về android.

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. 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 cập đến 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).

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.

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.

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 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.

## Kiến trúc ứng dụng.

Các thành phần (component) cơ bản tạo nên một ứng dụng Android được chia thành 6 loại. Nắm bắt được các thành phần này rất cần thiết cho việc lập trình.

#### Activity.

Hiểu một cách nôm na thì Activity tương tự như khái niệm Window (cửa sổ hay Form) trong hệ điều hành Windows. Khi khởi động một ứng dụng, bao giờ cũng có một Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác.

#### Service.

Thành phần chạy ẩn của Android. Service sử dụng để update dữ liệu, đưa ra các cảnh báo (notification) và không bao giờ hiển thị cho người dùng thấy.

Ví dụ: khi bạn yêu cầu phát một bài nhạc, sẽ có một Service chạy bên dưới để phát bài nhạc. Khi bạn download một tập tin, sẽ có một Serive thực hiện tác vụ download.

#### Content Provider.

Kho dữ liệu chia sẻ, được dùng để quản lý và chia sẽ dữ liệu giữa các ứng dụng.

Ví dụ: thông tin người dùng lưu trong contact, dữ liệu lưu trữ trên SQL Lite, dữ liệu lưu trữ trong các tận tin.

#### Intent.

Được sử dụng để truyền các thông báo nhằm khởi tạo một Activity hoặc Service để thực hiện công việc bạn mong muốn.

Ví dụ: khi mở một trang web, bạn sẽ gửi một intent đi để tạo một Activity mới hiển thị trang web đó.

#### Broadcast Receiver.

Thành phần thu nhận các intent từ bên ngoài gửi tới.

Ví dụ: bạn viết một chương trình thay thế cho phần gọi điện thoại mặc định của Android, khi đó, bạn cần 1 Broadcast Receiver để nhận các intent là các cuộc gọi đến.

#### Notification.

Đưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động Activity, Service, Broadcast Receiver và Content Provider là những thành phần chính cấu thành nên ứng dụng Android, bắt buộc phải khai báo trong AndroidManifest.

## 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.

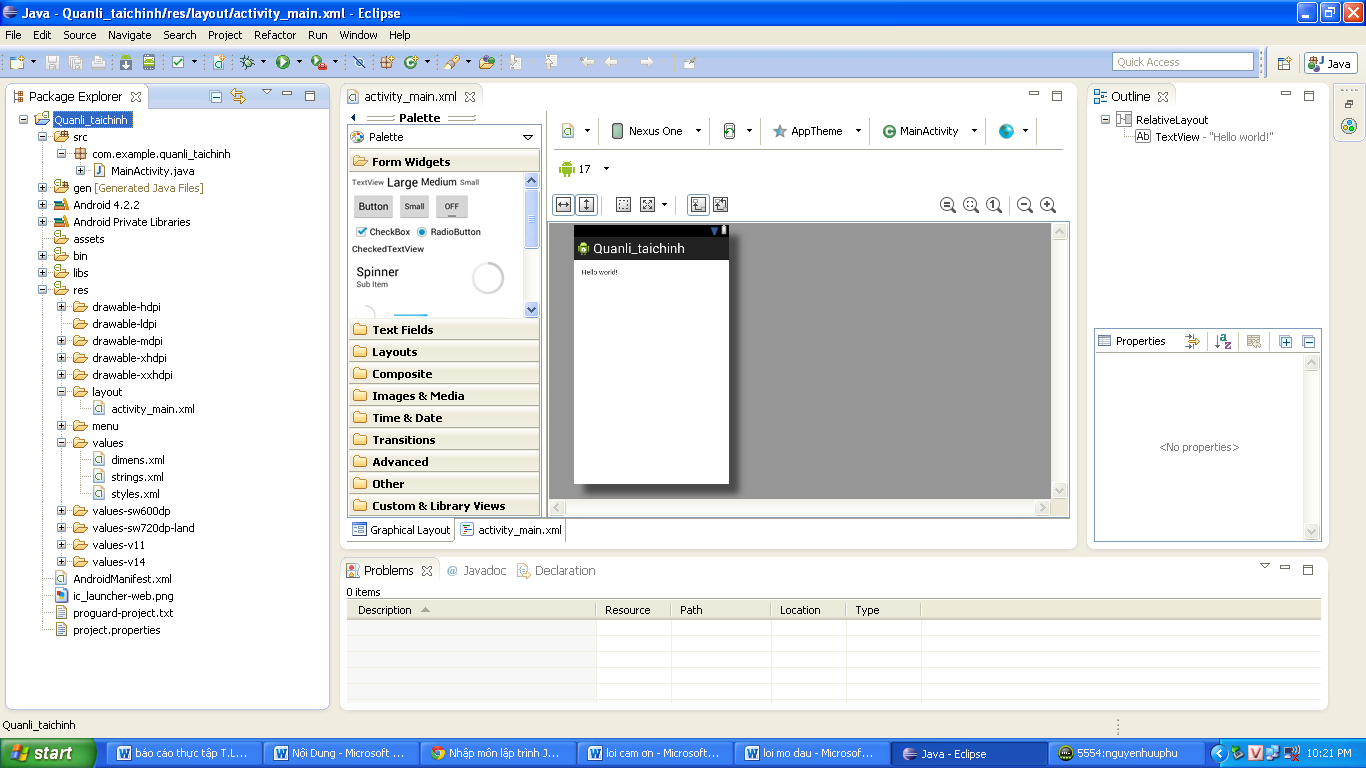
### Android.app: Gói API cấp cao, bao gồm các Activity và Service – hai lớp cơ sở cho mọi ứng dụng Android.

### 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…

## 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.

* ***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.

## Chương 2: Tìm hiểu về vấn đề quản lý mã nguồn và làm việc nhóm với GIT

## Tổng Quan Về Quản Lý Phiên Bản(mã nguồn).

Quản lý phiên bản (mã nguồn) là gì, tại sao bạn nên quan tâm? Quản lý phiên bản là một hệ thống lưu trữ các thay đổi của một tập tin (file) hoặc tập hợp các tập tin theo thời gian, do đó nó giúp bạn có thể quay lại một phiên bản xác định nào đó sau này. Mặc dù các ví dụ trong cuốn sách này sử dụng mã nguồn của phần mềm là đối tượng cho quản lý phiên bản, song trong thực thế bất kỳ loại file nào trên máy tính cũng có thể được sử dụng cho quản lý phiên bản.

Nếu bạn là một nhà thiết kế đồ hoạ hoặc thiết kế website, bạn muốn lưu trữ tất cả các phiên bản của một bức ảnh hoặc bố cục (cái mà chắc chắn bạn cần), thì sử dụng một Hệ Thống Quản Lý phiên bản (Version Control System - VCS) là một cách làm rất khôn ngoan. Một VCS cho phép bạn: khôi phục lại phiên bản cũ của các file, khôi phục lại phiên bản cũ của toàn bộ dự án, xem lại các thay đổi đã được thực hiện theo thời gian, xem ai là người thực hiện thay đổi cuối cùng có thể gây ra sự cố, hay xem ai là người đã gây ra sự cố đó và còn nhiều hơn thế nữa. Sử dụng VCS còn đồng nghĩa với việc khi bạn làm rối tung mọi thứ lên hay vô tình xoá mất các file đi, bạn có khôi phục lại chúng một cách dễ dàng. Hơn nữa, tất cả quá trình này có thể được thực hiện rất nhanh chóng và không hề tốn quá nhiều công sức.

## Hệ Thống Quản Lý Phiên Bản Cục Bộ

Nhiều người chọn phương pháp quản lý phiên bản bằng cách copy các file sang một thư mục khác (có thể là các thư mục được đặt tên theo thời gian, nếu họ thông minh). Đây là một phương pháp rất phổ biến bởi vì nó rất đơn giản, tuy nhiên nó cũng rất dễ gây ra lỗi. Bạn sẽ rất dễ quên rằng bạn đang ở trong thư mục nào hay vô tình sửa hoặc sao chép nhầm file mà bạn không muốn.

Để giải quyết vấn đề này, từ lâu các lập trình viên đã phát triển các phiên bản VCS cục bộ có chứa một database đơn giản lưu trữ tất cả các sự thay đổi của các files dưới sự kiểm soát thay đổi (xem Hình 1-1).



Hình 1-1. Mô hình quản lý phiên bản cục bộ.

Một trong những hệ thống quản lý phiên bản phổ biến hơn có tên là rcs vẫn còn được sử dụng ở nhiều máy tính cho tới bây giờ. Ngay cả hệ điều hành Mac OS X nổi tiếng cũng đưa vào các lệnh rcs khi bạn cài đặt Developer Tools (Các công cụ dành cho lập trình viên). Phần mềm này cơ bản hoạt động bằng cách lưu giữ các bản vá (những sự thay đổi giữa các file) từ phiên bản này qua phiên bản khác ở một định dạng đặc biệt được lưu trên ổ cứng; nó có thể tái tạo lại bất kỳ file nào ở bất kỳ thời điểm nào bằng cách gộp tất cả các bản vá lại với nhau.

## Hệ Thống Quản Lý Phiên Bản Tập Trung.

Vấn đề nghiêm trọng tiếp theo mà mọi người thường mắc phải là họ cần cộng tác với các lập trình viên khác trong hệ thống. Để vượt qua trở ngại này, Hệ Thống Quản Lý Phiên Bản Tập Trung (Centralized Version Control Systems - CVCSs) được phát triển. Các hệ thống này, ví dụ như CVS, Subversion, và Perforce, bao gồm một máy chủ có chứa tất cả các tập tin đã được "phiên bản hoá" (versioned), và danh sách các máy khách có quyền thay đổi các tập tin này trên máy chủ trung tâm đó. Trong vòng nhiều năm, mô hình này đã trở thành tiêu chuẩn cho việc quản lý phiên bản (xem Hình 1-2).



Hình 1-2. Mô hình quản lý phiên bản tập trung.

Mô hình này cung cấp rất nhiều lợi thế, đặc biết so với việc quản lý cục bộ. Ví dụ, tất cả người dùng đều biết một phần nào đó những việc mà những người khác trong dự án đang làm. Người quản lý có quyền quản lý ai có thể làm gì theo ý muốn; và việc này dễ dàng hơn nhiều so với việc phải quản lý ở từng cơ sở dử liệu ở từng máy riêng biệt.

Tuy nhiên, mô hình này cũng có những bất cập nghiêm trọng. Dễ nhận thấy nhất đó là "sự cố tập trung" mà máy chủ trung tâm mắc phải. Nếu máy chủ đó không hoạt động trong một giờ, nghĩa là trong khoảng thời gian đó không ai có thể cộng tác với những người còn lại hoặc lưu trữ các thay đổi đã được phiên bản hoá của bất kỳ tập tin nào mà người đó đang thao tác. Nếu ổ cứng lưu trữ cơ sở dữ liệu trung tâm bị hỏng, và các sao lưu dự phòng chưa được tạo ra tính đến thời điểm đó, bạn sẽ mất toàn bộ lịch sử của dự án đó, ngoại trừ những phiên bản cục bộ mà người dùng có được trên máy tính cá nhân. Các hệ thống quản lý phiên bản cục bộ phải đối diện với vấn đề tương tự như thế này mỗi khi toàn bộ lịch sử của dự án được lưu ở một nơi, bạn có nguy cơ mất tất cả.

## Hệ Thống Quản Lý Phiên Bản Phân Tán.

Đã tới lúc cần tới các Hệ Thống Quản Lý Phiên Bản Phân Tán - Distributed Version Control Systems (DVCSs). Trong các DVCS (ví dụ như Git, Mercurial, Bazaar hay Darcs), các máy khách không chỉ "check out" (sao chép về máy cục bộ) phiên bản mới nhất của các tập tin: chúng sao chép (mirror) toàn bộ kho chứa (repository). Chính vì vậy nếu như một máy chủ nào mà các hệ thống quản lý phiên bản này (mỗi máy khách là một hệ thống riêng biệt) đang cộng tác ngừng hoạt động, thì kho chứa từ bất kỳ máy khách nào cũng có thể dùng để sao chép ngược trở lại máy chủ để khôi phục lại toàn bộ hệ thống. Mỗi checkout thực sự là một bản sao đầy đủ của tất cả dữ liệu (xem Hình 1-3).



Hình 1-3. Mô hình quản lý phiên bản phân tán.

Ngoài ra, phần lớn các hệ thống này xử lý rất tốt việc quản lý nhiều kho chứa từ xa, vì thế bạn có thể cộng tác với nhiều nhóm người khác nhau theo những cách khác nhau trong cùng một dự án. Việc này cho phép bạn cài đặt nhiều loại "tiến trình công việc" (workflow) không thể thực hiện được với các hệ thống tập trung, ví dụ như các mô hình phân cấp.

## Tóm lại.

Trong lĩnh vực phát triển phần mềm, hệ thống quản lí mã nguồn (version control system) là vô cùng quan trọng bới các lý do:

1. Mã nguồn (code) là sản phẩm duy nhất và quan trọng nhất từ công sức của các kỹ sư phần mềm nên nó cần được quản lý chặt chẽ sự thay đổi, lưu trữ (backup) các phiên bản của phần mềm.

2. Trong hầu hết các trường hợp, phát triển phần mềm cần có sự tham gia của một nhóm lập trình viên làm việc tại các máy độc lập nên cơ chế làm việc chung trên cùng một mã nguồn là cực kì quan trọng.

Git là một hệ thống quản lý mã nguồn đáp ứng các yêu cầu trên với sự tập trung cho tốc độ, tính toàn vẹn dữ liệu và hỗ trợ các quy trình làm việc phi tuyến tính. Có một điều khá thú vị là người thiết kế và phát triển Git cũng chính là cha đẻ của hệ điều hành Linux, ông Linus Torvalds vào năm 2005. Phát minh mã nguồn mở này nhanh chóng được sử dụng rộng rãi và dần trở thành công cụ không thể thiếu với mọi lập trình viên.

## Cơ bản về Git.

Tóm lại thì, Git là gì? Đây là một phần quan trọng để tiếp thu, bởi vì nếu bạn hiểu được Git là gì và các nguyên tắc cơ bản của việc Git hoạt động như thế nào, thì sử dụng Git một cách hiệu quả sẽ trở nên dễ dàng hơn cho bạn rất nhiều. Khi học Git, hãy cố gắng gạt bỏ những kiến thức mà có thể bạn đã biết về các VCS khác, ví dụ như Subversion và Perforce; việc này sẽ giúp bạn tránh được sự hỗn độn, bối rối khi sử dụng nó. Git "nghĩ" về thông tin và lưu trữ nó khá khác biệt so với các hệ thống khác, mặc dù giao diện người dùng tương đối giống nhau; hiểu được những khác biệt đó sẽ giúp bạn tránh được rất nhiều bối rối.

## Ảnh chụp, không phải sự khác biệt.

Sự khác nhau cơ bản giữa Git với bất kỳ VCS nào khác (bao gồm Subversion và tương tự là cách Git "nghĩ" về dữ liệu. Về mặt lý thuyết mà nói, phần lớn hệ thống khác lưu trữ thông tin dưới dạng danh sách các tập tin được thay đổi. Các hệ thống này (CVS, Subversion, Perforce, Bazaar,...) coi thông tin được lưu trữ như là một tập hợp các tập tin và các thay đổi được thực hiện trên mỗi tập tin theo thời gian, được minh hoạ trong hình 1-4.



Hình 1-4. Các hệ thống khác hướng tới lưu trữ tập tin dưới dạng các thay đổi so với bản cơ sở của mỗi tập tin.

Git không nghĩ hoặc xử lý dữ liệu theo cách này. Mà thay vào đó Git coi dữ liệu của nó giống như một tập hợp các "ảnh" (snapshot) của một hệ thống tập tin nhỏ. Mỗi lần bạn "commit", hoặc lưu lại trạng thái hiện tại của dự án trong Git, về cơ bản Git "chụp một bức ảnh" ghi lại nội dung của tất cả các tập tin tại thời điểm đó và tạo ra một tham chiếu tới "ảnh" đó. Để hiệu quả hơn, nếu như tập tin không có sự thay đổi nào, Git không lưu trữ tập tin đó lại một lần nữa mà chỉ tạo một liên kết tới tập tin gốc đã tồn tại trước đó. Git thao tác với dữ liệu giống như Hình 1-5.



Hình 1-5. Git lưu trữ dữ liệu dưới dạng ảnh chụp của dự án theo thời gian.

Đây là sự khác biệt lớn nhất giữa Git và hầu hết các VCS khác. Nó khiến Git cân nhắc lại hầu hết các khía cạnh của quản lý phiên bản mà phần lớn các hệ thống khác chỉ áp dụng lại từ các thế hệ trước. Chính lý do này làm cho Git giống như một hệ thống quản lý tập tin thu nhỏ với các tính năng, công cụ vô cùng mạnh mẽ được xây dựng dựa trên nó, không chỉ là một hệ thống quản lý phiên bản đơn giản.

## Phần lớn thao tác diễn ra cục bộ.

Phần lớn các thao tác/hoạt động trong Git chỉ cần yêu cầu các tập tin hay tài nguyên cục bộ - thông thường nó sẽ không cần bất cứ thông tin từ máy tính nào khác trong mạng lưới của bạn. Nếu như bạn quen với việc sử dụng các hệ thống quản lý phiên bản tập trung nơi mà đa số hoạt động đều chịu sự ảnh hưởng bởi độ trễ của mạng, thì với Git đó lại là một thế mạnh. Bởi vì toàn bộ dự án hoàn toàn nằm trên ổ cứng của bạn, các thao tác được thực hiện gần như ngay lập tức.

Ví dụ, khi bạn muốn xem lịch sử của dự án, Git không cần phải lấy thông tin đó từ một máy chủ khác để hiển thị, mà đơn giản nó được đọc trực tiếp từ chính cơ sở dữ liệu cục bộ của bạn. Điều này có nghĩa là bạn có thể xem được lịch sử thay đổi của dự án gần như ngay lập tức. Nếu như bạn muốn so sánh sự thay đổi giữa phiên bản hiện tại của một tập tin với phiên bản của một tháng trước, Git có thể tìm kiếm tập tin cũ đó trên máy cục bộ rồi sau đó so sánh sự khác biệt cho bạn. Thay vì việc phải truy vấn từ xa hoặc "kéo về" (pull) phiên bản cũ của tập tin đó từ máy chủ trung tâm rồi mới thực hiện so sánh cục bộ.

Điều này còn đồng nghĩa với có rất ít việc mà bạn không thể làm được khi không có kết nối Internet hoặc VPN bị ngắt. Nếu bạn muốn làm việc ngay cả khi ở trên máy bay hoặc trên tầu, bạn vẫn có thể commit bình thường cho tới khi có kết nối Internet để đồng bộ hoá. Nếu bạn đang ở nhà mà VPN lại không thể kết nối được, bạn cũng vẫn có thể làm việc bình thường. Trong rất nhiều hệ thống khác, việc này gần như là không thể hoặc rất khó khăn. Ví dụ trong Perforce, bạn gần như không thể làm gì nếu như không kết nối được tới máy chủ; trong Subversion và CVS, bạn có thể sửa tập tin nhưng bạn không thể commit các thay đổi đó vào cơ sở dữ liệu (vì cơ sở dữ liệu của bạn không được kết nối). Đây có thể không phải là điều gì đó lớn lao, nhưng bạn sẽ ngạc nhiên về sự thay đổi lớn mà nó có thể làm được.

## Git mang tính toàn vẹn.

Mọi thứ trong Git được "băm" (checksum or hash) trước khi lưu trữ và được tham chiếu tới bằng mã băm đó. Có nghĩa là việc thay đổi nội dung của một tập tin hay một thư mục mà Git không biết tới là điều không thể. Chức năng này được xây dựng trong Git ở tầng thấp nhất và về mặt triết học được coi là toàn vẹn. Bạn không thể mất thông tin/dữ liệu trong khi truyền tải hoặc nhận về một tập tin bị hỏng mà Git không phát hiện ra.

Cơ chế mà Git sử dụng cho việc băm này được gọi là mã băm SHA-1. Đây là một chuỗi được tạo thành bởi 40 ký tự của hệ cơ số 16 (0-9 và a-f) và được tính toán dựa trên nội dung của tập tin hoặc cấu trúc thư mục trong Git. Một mã băm SHA-1 có định dạng như sau:

24b9da6552252987aa493b52f8696cd6d3b00373

Bạn sẽ thấy các mã băm được sử dụng ở mọi nơi trong Git. Thực tế, Git không sử dụng tên của các tập để lưu trữ mà bằng các mã băm từ nội dung của tập tin vào một cơ sở dữ liệu có thể truy vấn được.

## Git chỉ thêm mới dữ liệu.

Khi bạn thực hiện các hành động trong Git, phần lớn tất cả hành động đó đều được thêm vào cơ sở dữ liệu của Git. Rất khó để yêu cầu hệ thống thực hiện một hành động nào đó mà không thể khôi phục lại được hoặc xoá dữ liệu đi dưới mọi hình thức. Giống như trong các VCS khác, bạn có thể mất hoặc làm rối tung dữ liệu mà bạn chưa commit; nhưng khi bạn đã commit thì rất khó để mất các dữ liệu đó, đặc biệt là nếu bạn thường xuyên đẩy (push) cơ sở dữ liệu sang một kho chứa khác.

Điều này khiến việc sử dụng Git trở nên thích thú bởi vì chúng ta biết rằng chúng ta có thể thử nghiệm mà không lo sợ sẽ phá hỏng mọi thứ.

## Ba trạng thái.

Bây giờ, hãy chú ý. Đây là điều quan trọng cần ghi nhớ về Git nếu như bạn muốn hiểu được những phần tiếp theo một cách trôi chảy. Mỗi tập tin trong Git được quản lý dựa trên ba trạng thái: committed, modified, và staged. Committed có nghĩa là dữ liệu đã được lưu trữ một cách an toàn trong cơ sở dữ liệu. Modified có nghĩa là bạn đã thay đổi tập tin nhưng chưa commit vào cơ sở dữ liệu. Và staged là bạn đã đánh dấu sẽ commit phiên bản hiện tại của một tập tin đã chỉnh sửa trong lần commit sắp tới.

Điều này tạo ra ba phần riêng biệt của một dự án sử dụng Git: thư mục Git, thư mục làm việc, và khu vực tổ chức (staging area).



Hình 1-6. Thư mục làm việc, khu vực khán đài, và thư mục Git.

Thư mục Git là nơi Git lưu trữ các "siêu dữ kiện" (metadata) và cơ sở dữ liệu cho dự án của bạn. Đây là phần quan trọng nhất của Git, nó là phần được sao lưu về khi bạn tạo một bản sao (clone) của một kho chứa từ một máy tính khác.

Thư mục làm việc là bản sao một phiên bản của dự án. Những tập tin này được kéo về (pulled) từ cơ sở dữ liệu được nén lại trong thư mục Git và lưu trên ổ cứng cho bạn sử dụng hoặc chỉnh sửa.

Khu vực khán đài là một tập tin đơn giản được chứa trong thư mục Git, nó chứa thông tin về những gì sẽ được commit trong lần commit sắp tới. Nó còn được biết đến với cái tên "chỉ mục" (index), nhưng khu vực tổ chức (staging area) đang dần được coi là tên tiêu chuẩn.

Tiến trình công việc (workflow) cơ bản của Git:

1. Bạn thay đổi các tập tin trong thư mục làm việc.

2. Bạn tổ chức các tập tin, tạo mới ảnh của các tập tin đó vào khu vực tổ chức.

3. Bạn commit, ảnh của các tập tin trong khu vực tổ chức sẽ được lưu trữ vĩnh viễn vào thư mục Git.

Nếu một phiên bản nào đó của một tập tin ở trong thư mục Git, nó được coi là đã commit. Nếu như nó đã được sửa và thêm vào khu vực tổ chức, nghĩa là nó đã được staged. Và nếu nó được thay đổi từ khi checkout nhưng chưa được staged, nó được coi là đã thay đổi. Trong Chương 2, bạn sẽ được tìm hiểu kỹ hơn về những trạng thái này cũng như làm thế nào để tận dụng lợi thế của chúng hoặc bỏ qua hoàn toàn giai đoạn tổ chức (staged).

## Chương 3 : Tìm hiểu và phân tích bài toán quản lý thu chi.

## Mở Đầu.

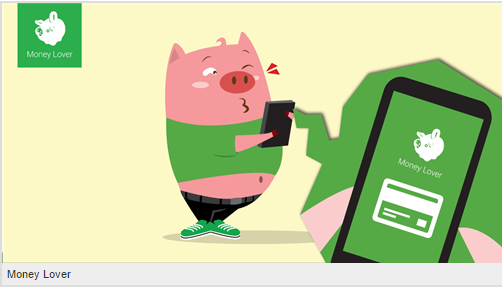
Việc quản lý tài chính có thể là vấn đề khiến khá nhiều người đau đầu. Thậm chí nếu không có giải pháp tốt thì có thể dẫn tới thâm hụt trong chi tiêu khá cao. Vì thế mà hiện nay những ứng dụng cho việc quản lý tài chính ra đời ngày càng nhiều để giúp theo dõi tiêu dùng cá nhân một cách hiệu quả nhất.

Hiện tại đã có rất nhiều app quản lý tài chính cá nhân được phát triển miễn phí trên iOS và Android. Bạn có thể quản lý các khoản tiền, các khoản chi tiêu cá nhân một cách nhanh chóng và thuận tiện ngay trên chiếc điện thoại di động của mình.

Sau đây là một số ứng dụng quản lý tài chính cá nhân miễn phí cho di động mà nhóm đã tìm hiểu.

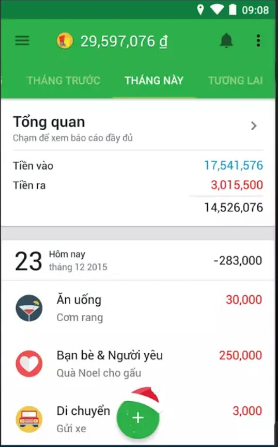
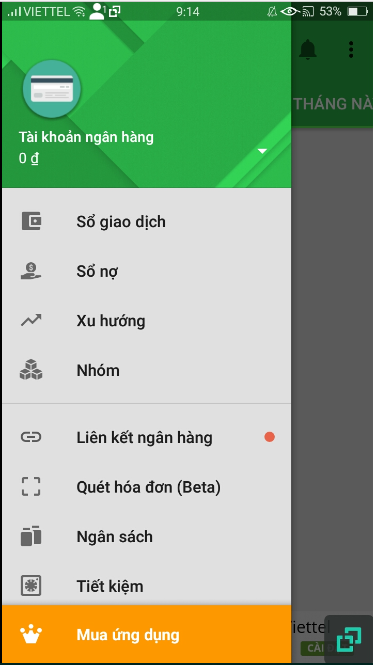
## Money Lover – Ứng dụng quản lý tài chính thuần Việt.

Đây là ứng dụng quản lý tài chính cá nhân được hàng triệu người dùng trên toàn thế giới yêu thích do đội ngũ lập trình người Việt tại ZooStudio thiết kế. Hiện tại, Money Lover đã có phiên bản dành cho nhiều quốc gia khác nhau với 34 ngôn ngữ. Money Lover từng đạt giải nhất cuộc thi Nhân tài đất Việt 2014 cho Ứng dụng di động xuất sắc nhất.



Ứng dụng này giúp việc ghi chép và lên kế hoạch chi tiêu cho bản thân hoặc cả gia đình trở nên đơn giản và thuận tiện trên thiết bị di động. Bên cạnh việc nhập giao dịch thủ công, Money Lover hỗ trợ người dùng kết nối với tài khoản ngân hàng của hầu hết ngân hàng ở Việt Nam và các dịch vụ khác, giúp việc theo dõi thu chi hoàn toàn tự động và tiết kiệm thời gian.

Chỉ việc nhập số tiền và chọn mục chi hay thu, công việc ghi chép và theo dõi chi tiêu chỉ trong vài phút. Với tính năng Ngân sách, người dùng có thể nhanh chóng thiết lập các hạn mức chi tiêu cho toàn bộ các khoản chi hay từng khoản cụ thể. Ứng dụng sẽ nhắc nhở bạn khi mức chi tiêu đã đạt gần mức giới hạn đề ra. Hệ thống báo cáo, thống kê thu chi sẽ giúp người dùng biết được tháng này tiêu hết bao nhiêu tiền, khoản nào chi nhiều nhất, thu nhập tuần này là bao nhiêu... Ngoài ra, với khái niệm ghi nhớ chủ động (note-taking concept), Money Lover sẽ tạo cho người dùng ý thức chủ động quản lý tài chính cá nhân và thay đổi hành vi tiêu dùng phù hợp với tình hình tài chính hiện tại.

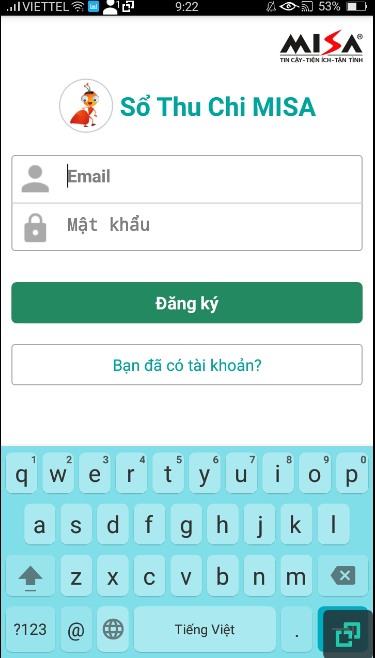


Ưu điểm khác của ứng dụng đó là sử dụng công nghệ điện toán đám mây – Money Cloud, giúp dự trữ và bảo vệ dữ liệu an toàn, loại bỏ được nỗi lo của người dùng khi mất dữ liệu trên thiết bị của họ. Ngoài ra, Money Lover đó đảm bảo tính bảo mật thông tin khi không yêu cầu thông tin thẻ tín dụng hay tài khoản của người dùng.

## Ứng dụng Sổ Thu Chi MISA.

Sổ Thu Chi giúp người dùng ghi chép lại các khoản thu chi hằng ngày của cá nhân hoặc liên quan đến gia đình, ngoài ra ứng dụng này còn cung cấp việc ghi chép chi tiêu liên quan tới 1 sự kiện nào đó như đi du lịch hay đám cưới.

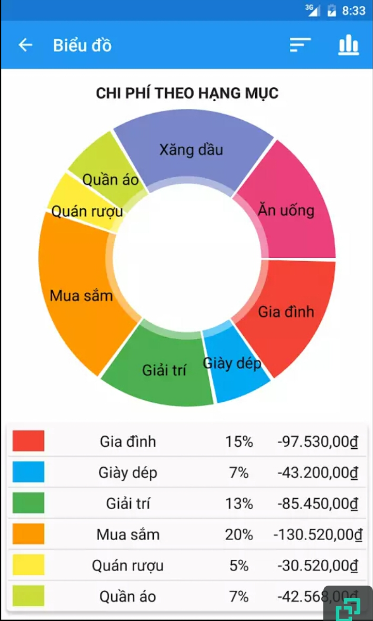
Sổ thu chi MISA được chia là 4 phần chính bao gồm Ghi chép, Tài Khoản, Báo cáo và Thiết lập rất trực quan. Ứng dụng cũng tích hợp sẵn bảng tính (calculator) để người dùng tiện sử dụng ngay trong ứng dụng mà không cần thoát ra ngoài.

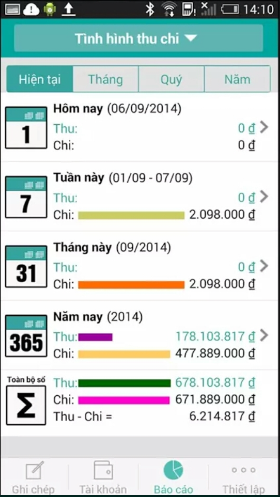
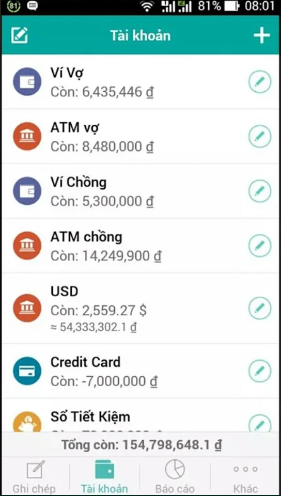


Người dùng có thể dễ dàng ghi lại các khoản thu, chi từ ví/tài khoản nào, vào thời điểm nào, với từng hạng mục như xăng xe, bảo hiểm, quần áo… trong mục Ghi Chép.

## Fast Budget - Expense Manager.

Ứng dụng giúp người dùng có thể kiểm tra thu nhập và chi tiêu và cố gắng cải thiện tiền tiết kiệm. Người dùng có thể hình dung được dòng tiền thông qua trang Tổng quan mới và hoàn toàn có thể tùy biến được. Trong mục Quản lý Ngân sách Chi tiêu, người dùng còn có rất nhiều công cụ để quản lý thu chi tốt hơn, như là những biểu đồ đa dạng hoặc một lịch trình hữu ích. Người dùng cũng có thể hoạch định ngân sách cho cả năm, cho tháng, thậm chí cho tuần tiếp theo

Ứng dụng quản lý chi tiêu cho Android này hỗ trợ sao lưu tự động hoặc thủ công vào thẻ SD, trên Dropbox hoặc gửi bản sao lưu vào mail.



Các thông tin này cho phép theo dõi nguồn tiền cũng như đánh giá thu chi, người dùng có thể lựa chọn tính toán theo từng sự kiện cụ thể. Việc kiểm tra xem mình còn bao nhiêu tiền trong tài khoản, bao nhiêu tiền tiết kiệm cũng thuận tiên hơn khi mục Tài khoản sẽ giúp người dùng theo dõi cũng như quản lý túi tiền của mình. Tại đây, người dùng có thể tạo thêm hay bớt nhiều loại tài khoản như tiền mặt, tài khoản ATM, sổ tiết kiệm

Ứng dụng này còn giúp người dùng cập nhật tỷ giá, thêm ngôn ngữ, thay đổi thông tin người dùng hay lập mã bảo vệ cho những khoản “quỹ đen” của mình.

Người dùng hoàn toàn có thể đồng bộ dữ liệu của mình lên đám mây hoặc xuất dữ liệu ra theo định dạng Excel hoặc PDF để xem trên máy tính cũng như các thiết bị khác.