BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

ISO 9001:2000

ISO 9001 : 2008

ĐỒ ÁN TỐT NGHIỆP

Ngành: Công nghệ thông tin

Hải Phòng 2017

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

TÌM HIỂU SQLITE VÀ XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG

ĐỒ AN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-------o0o-------

TÌM HIỂU SQLITE VÀ XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

Sinh viên thực hiện: Nguyễn Hoàng Long

Giáo viên hướng dẫn: Ths. Phùng Anh Tuấn

Mã số sinh viên: 101432

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

Cộng hoà xã hội chủ nghĩa Việt Nam

Độc lập - Tự do - Hạnh phúc

-------o0o-------

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Nguyễn Hoàng Long Mã số: 101432

Lớp: CT1601 Ngành: Công nghệ Thông tin

Tên đề tài: Tìm hiểu SQLite và xây dựng chương trình ứng dụng

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Nội dung:

- Tìm hiểu về hệ điều hành android
- Tìm hiểu về môi trường lập trình Android Studio
- Tìm hiểu về cụng cụ tạo lập thiết bị di động ảo Genymotion
- Tìm hiểu về hệ quản trị cơ sở dữ liệu cho thiết bị di động SQLite
- Xây dựng chương trình ứng dụng thử nghiệm quản lý chi tiêu cá nhân

b. Các yêu cầu cần giải quyết:

- Nắm được một số khái niệm cơ bản của hệ điều hành Android.
- Cài đặt, cấu hình môi trường lập trình ứng dụng cho thiết bị di động Android Studio.
- Cài đặt, cấu hình thiết bị di động ảo cho chạy thử nghiệm chương trình.
- Thực hiện được các thao tác cơ bản với dữ liệu trên hệ quản trị cơ sở dữ liệu SQLite.
- Xây dựng được chương trình ứng dụng, đóng gói ứng dụng thành file *.apk cho phép cài và chạy trên thiết bị di động Android thật.

2. Các số liệu cần thiết để thiết kế, tính toán

- Số liệu giả lập

3. Địa điểm thực tập

Trường Đại học Dân lập Hải Phòng.

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Người hướng dẫn thứ nhất:

Họ và tên: Phùng Anh Tuấn

Học hàm, học vị: Thạc sỹ

Cơ quan công tác: Trường Đại học Dân lập Hải Phòng

Nội dung hướng dẫn:

- Tìm hiểu về hệ điều hành android
- Tìm hiểu về môi trường lập trình Android Studio
- Tìm hiểu về cụng cụ tạo lập thiết bị di động ảo Genymotion
- Tìm hiểu về hệ quản trị cơ sở dữ liệu cho thiết bị di động SQLite
- Xây dựng chương trình ứng dụng thử nghiệm quản lý chi tiêu cá nhân, cài đặt và chạy trên thiết bị android thật.

Người hướng dẫn thứ hai:

Họ và tên: …….....

Học hàm, học vị……..

Cơ quan công tác: ……

Nội dung hướng dẫn: …….....

……

Đề tài tốt nghiệp được giao ngày ..... tháng ..... năm 2017

Yêu cầu phải hoàn thành trước ngày ..... tháng ..... năm 2017

Đã nhận nhiệm vụ: Đ.T.T.N	Đã nhận nhiệm vụ: Đ.T.T.N
Sinh viên	Cán bộ hướng dẫn Đ.T.T.N

Hải Phòng, ngày .....tháng.....năm 2017

Hiệu trưởng

GS.TS.NGƯT Trần Hữu Nghị

PHẦN NHẬN XÉT TÓM TẮT CỦA CÁN BỘ HƯỚNG DẪN

1 . Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp:

- Có nhiều cố gắng trong việc tìm kiếm, nghiên cứu tài liệu phục vụ cho nội dung của đề tài tốt nghiệp.

- Chấp hành tương đối tốt các yêu cầu của cán bộ hướng dẫn.

- Khả năng làm việc độc lập còn nhiều hạn chế trong thuật toán và kỹ năng lập trình.

2 . Đánh giá chất lượng của đề tài tốt nghiệp (so với nội dung yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp)

- Hoàn thành các yêu cầu đã đề ra trong nhiệm vụ tốt nghiệp.

- Xây dựng thành công chương trình ứng dụng thử nghiệm, cài đặt chạy trên thiết bị di động android thật.

- Nội dung đề tài là tài liệu tham khảo hữu ích cho các đối tượng quan tâm tới chủ đề lập trình với cơ sở dữ liệu trên thiết bị di động android.

- Đề nghị cho phép sinh viên được bảo vệ trước hội đồng chấm bảo vệ đồ án tốt nghiệp.

3 . Cho điểm của cán bộ hướng dẫn:

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Ngày.....tháng.....năm 2017

Cán bộ hướng dẫn chính

(Ký, ghi rõ họ tên)

PHẦN NHẬN XET DANH GIA CỦA CAN BỘ CHẤM PHẢN BIỆN DỀ TAI TỐT NGHIỆP

1. **Đánh giá chất lượng đề tài tốt nghiệp (về các mặt như cơ sở lý luận, thuyết minh chương trình, giá trị thực tế, ...)**

2. Cho điểm của cán bộ phản biện

(*Điểm ghi bằng số và chữ*)

.....

.....

Ngày.....tháng.....năm 2017

Cán bộ chấm phản biện

(Ký, ghi rõ họ tên)

Mục Lục

LỜI CẢM ƠN

Để đồ án này đạt kết quả tốt đẹp, em đã nhận được sự hỗ trợ, giúp đỡ của nhiều cơ quan, tổ chức, cá nhân. Với tình cảm sâu sắc, chân thành, cho phép em được bày tỏ lòng biết ơn sâu sắc đến tất cả các cá nhân và cơ quan đã tạo điều kiện giúp đỡ trong quá trình học tập và nghiên cứu làm đồ án. Trước hết em xin gởi tới các thầy cô khoa Công nghệ - Thông tin trường Đại học Dân Lập Hải Phòng lời chào trân trọng, lời chúc sức khỏe và lời cảm ơn sâu sắc. Với sự quan tâm, dạy dỗ, chỉ bảo tận tình chu đáo của thầy cô, đến nay em đã có thể hoàn thành đồ án: "Tìm hiểu SQLite và xây dựng chương trình ứng dụng ". Đặc biệt em xin gửi lời cảm ơn chân thành nhất tới thầy giáo – Ths. Phùng Anh Tuấn đã quan tâm giúp đỡ, hướng dẫn em hoàn thành tốt đồ án này trong thời gian qua. Em xin bày tỏ lòng biết ơn đến lãnh đạo Trường Đại học Dân Lập Hải Phòng, Phòng Đào Tạo, các Khoa Phòng ban chức năng đã trực tiếp và gián tiếp giúp đỡ em trong suốt quá trình học tập tại trường. Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của một sinh viên, đồ án này không thể tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công việc thực tế sau này.

Xin chân thành cảm ơn!

Sinh viên

Nguyễn Hoàng Long

Chương 1: HỆ ĐIỀU HÀNH ANDROID

1.1 Giới thiệu HĐH Android

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008.

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình Java có sửa đổi. Vào tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian vào quý 4 năm 2010, và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu. Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game và các thiết bị điện tử khác. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 75% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 3 năm 2012, với tổng cộng 500 triệu thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Sự thành công của hệ điều hành cũng khiến

nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "cuộc chiến điện thoại thông minh" giữa các công ty công nghệ.

1.2 Lịch sử phát triển

Tổng công ty Android (Android, Inc.) được thành lập tại Palo Alto, California vào tháng 10 năm 2003 bởi Andy Rubin (đồng sáng lập công ty Danger),[20] Rich Miner (đồng sáng lập Tổng công ty Viễn thông Wildfire), Nick Sears (từng là Phó giám đốc T-Mobile), và Chris White (trưởng thiết kế và giao diện tại WebTV) để phát triển, theo lời của Rubin, "các thiết bị di động thông minh hơn có thể biết được vị trí và sở thích của người dùng". Dù những người thành lập và nhân viên đều là những người có tiếng tăm, Tổng công ty Android hoạt động một cách âm thầm, chỉ tiết lộ rằng họ đang làm phần mềm dành cho điện thoại di động. Trong năm đó, Rubin hết kinh phí. Steve Perlman, một người bạn thân của Rubin, mang cho ông 10.000 USD tiền mặt nhưng từ chối tham gia vào công ty.

Google mua lại Tổng công ty Android vào ngày 17 tháng 8 năm 2005, biến nó thành một bộ phận trực thuộc Google. Những nhân viên của chủ chốt của Tổng công ty Android, gồm Rubin, Miner và White, vẫn tiếp tục ở lại công ty làm việc sau thương vụ này. Vào thời điểm đó không có nhiều thông tin về Tổng công ty, nhưng nhiều người đồn đoán rằng Google dự tính tham gia thị trường điện thoại di động sau bước đi này. Tại Google, nhóm do Rubin đứng đầu đã phát triển một nền tảng thiết bị di động phát triển trên nền nhân Linux. Google quảng bá nền tảng này cho các nhà sản xuất điện thoại và các nhà mạng với lời hứa sẽ cung cấp một hệ thống uyển chuyển và có khả năng nâng cấp. Google đã liên hệ với hàng loạt hãng phần cứng cũng như đối tác phần mềm, bắn tin cho các nhà mạng rằng họ sẵn sàng hợp tác với các cấp độ khác nhau.

Ngày càng nhiều suy đoán rằng Google sẽ tham gia thị trường điện thoại di động xuất hiện trong tháng 12 năm 2006. Tin tức của BBC và Nhật báo phố Wall chú thích rằng Google muốn đưa công nghệ tìm kiếm và các ứng dụng của họ vào điện thoại di động và họ đang nỗ lực làm việc để thực hiện điều này. Các phương tiện truyền thông truyền thống lẫn online cũng viết về tin đồn rằng Google đang phát triển một thiết bị cầm tay mang thương hiệu Google. Một vài tờ báo còn nói rằng trong khi Google vẫn đang thực hiện những bản mô tả kỹ thuật chi tiết, họ đã trình diễn sản phẩm mẫu cho các nhà sản xuất điện thoại di

động và nhà mạng. Tháng 9 năm 2007, InformationWeek đăng tải một nghiên cứu của Evalueserve cho biết Google đã nộp một số đơn xin cấp bằng sáng chế trong lĩnh vực điện thoại di động.

Ngày 5 tháng 11 năm 2007, Liên minh thiết bị cầm tay mở (Open Handset Alliance), một hiệp hội bao gồm nhiều công ty trong đó có Texas Instruments, Tập đoàn Broadcom, Google, HTC, Intel, LG, Tập đoàn Marvell Technology, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel và T-Mobile được thành lập với mục đích phát triển các tiêu chuẩn mở cho thiết bị di động. Cùng ngày, Android cũng được ra mắt với vai trò là sản phẩm đầu tiên của Liên minh, một nền tảng thiết bị di động được xây dựng trên nhân Linux phiên bản 2.6. Chiếc điện thoại chạy Android đầu tiên được bán ra là HTC Dream, phát hành ngày 22 tháng 10 năm 2008. Biểu trưng của hệ điều hành Android mới là một con rôbốt màu xanh lá cây do hãng thiết kế Irina Blok tại California vẽ.

Từ năm 2008, Android đã trải qua nhiều lần cập nhật để dần dần cải tiến hệ điều hành, bổ sung các tính năng mới và sửa các lỗi trong những lần phát hành trước. Mỗi bản nâng cấp được đặt tên lần lượt theo thứ tự bảng chữ cái, theo tên của một món ăn tráng miệng; ví dụ như phiên bản 1.5 Cupcake (bánh bông lan nhỏ có kem) tiếp nối bằng phiên bản 1.6 Donut (bánh vòng). Phiên bản mới nhất hiện nay là 5.0 Lollipop. Vào năm 2010, Google ra mắt loạt thiết bị Nexus—một dòng sản phẩm bao gồm điện thoại thông minh và máy tính bảng chạy hệ điều hành Android, do các đối tác phần cứng sản xuất. HTC đã hợp tác với Google trong chiếc điện thoại thông minh Nexus đầu tiên, Nexus One. Kể từ đó nhiều thiết bị mới hơn đã gia nhập vào dòng sản phẩm này, như điện thoại Nexus 4 và máy tính bảng Nexus 10, lần lượt do LG và Samsung sản xuất. Google xem điện thoại và máy tính bảng Nexus là những thiết bị Android chủ lực của mình, với những tính năng phần cứng và phần mềm mới nhất của Android.Năm 2014, Google công báo Android Wear, hệ điều hành dành cho các thiết bị đeo được.

1.3 Ứng dụng Android

Android có lượng ứng dụng của bên thứ ba ngày càng nhiều, được chọn lọc và đặt trên một cửa hàng ứng dụng như Google Play hay Amazon Appstore để người dùng lấy về, hoặc bằng cách tải xuống rồi cài đặt tập tin APK từ trang web khác. Các ứng dụng trên Cửa hàng Play cho phép người dùng duyệt, tải về

và cập nhật các ứng dụng do Google và các nhà phát triển thứ ba phát hành. Cửa hàng Play được cài đặt sẵn trên các thiết bị thỏa mãn điều kiện tương thích của Google. Ứng dụng sẽ tự động lọc ra một danh sách các ứng dụng tương thích với thiết bị của người dùng, và nhà phát triển có thể giới hạn ứng dụng của họ chỉ dành cho những nhà mạng cố định hoặc những quốc gia cố định vì lý do kinh doanh. Nếu người dùng mua một ứng dụng mà họ cảm thấy không thích, họ được hoàn trả tiền sau 15 phút kể từ lúc tải về, và một vài nhà mạng còn có khả năng mua giúp các ứng dụng trên Google Play, sau đó tính tiền vào trong hóa đơn sử dụng hàng tháng của người dùng. Đến tháng 9 năm 2012, có hơn 675.000 ứng dụng dành cho Android, và số lượng ứng dụng tải về từ Cửa hàng Play ước tính đạt 25 tỷ.

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java sử dụng Bộ phát triển phần mềm Android (SDK). SDK bao gồm một bộ đầy đủ các công cụ dùng để phát triển, gồm có công cụ gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại dựa trên QEMU, tài liệu hướng dẫn, mã nguồn mẫu, và hướng dẫn từng bước. Môi trường phát triển tích hợp (IDE) được hỗ trợ chính thức là Eclipse sử dụng phần bổ sung Android Development Tools (ADT). Các công cụ phát triển khác cũng có sẵn, gồm có Bộ phát triển gốc dành cho các ứng dụng hoặc phần mở rộng viết bằng C hoặc C++, Google App Inventor, một môi trường đồ họa cho những nhà lập trình mới bắt đầu, và nhiều nền tảng ứng dụng web di động đa nền tảng phong phú.

1.4 Giao diện Android

Giao diện người dùng của Android dựa trên nguyên tắc tác động trực tiếp, sử dụng cảm ứng chạm tương tự như những động tác ngoài đời thực như vuốt, chạm, kéo giãn và thu lại để xử lý các đối tượng trên màn hình. Sự phản ứng với tác động của người dùng diễn ra gần như ngay lập tức, nhằm tạo ra giao diện cảm ứng mượt mà, thường dùng tính năng rung của thiết bị để tạo phản hồi rung cho người dùng. Những thiết bị phần cứng bên trong như gia tốc kế, con quay hồi chuyển và cảm biến khoảng cách được một số ứng dụng sử dụng để phản hồi một số hành động khác của người dùng, ví dụ như điều chỉnh màn hình từ chế độ hiển thị dọc sang chế độ hiển thị ngang tùy theo vị trí của thiết bị, hoặc cho phép người dùng lái xe đua bằng xoay thiết bị, giống như đang điều khiển vô-lăng.

Các thiết bị Android sau khi khởi động sẽ hiển thị màn hình chính, điểm khởi đầu với các thông tin chính trên thiết bị, tương tự như khái niệm desktop (bàn làm việc) trên máy tính để bàn. Màn hình chính Android thường gồm nhiều biểu tượng (icon) và tiện ích (widget); biểu tượng ứng dụng sẽ mở ứng dụng tương ứng, còn tiện ích hiển thị những nội dung sống động, cập nhật tự động như dự báo thời tiết, hộp thư của người dùng, hoặc những mẩu tin thời sự ngay trên màn hình chính. Màn hình chính có thể gồm nhiều trang xem được bằng cách vuốt ra trước hoặc sau, mặc dù giao diện màn hình chính của Android có thể tùy chỉnh ở mức cao, cho phép người dùng tự do sắp đặt hình dáng cũng như hành vi của thiết bị theo sở thích. Những ứng dụng do các hãng thứ ba có trên Google Play và các kho ứng dụng khác còn cho phép người dùng thay đổi "chủ đề" của màn hình chính, thậm chí bắt chước hình dáng của hệ điều hành khác như Windows Phone chẳng hạn. Phần lớn những nhà sản xuất, và một số nhà mạng, thực hiện thay đổi hình dáng và hành vi của các thiết bị Android của họ để phân biệt với các hãng cạnh tranh.

Ở phía trên cùng màn hình là thanh trạng thái, hiển thị thông tin về thiết bị và tình trạng kết nối. Thanh trạng thái này có thể "kéo" xuống để xem màn hình thông báo gồm thông tin quan trọng hoặc cập nhật của các ứng dụng, như email hay tin nhắn SMS mới nhận, mà không làm gián đoạn hoặc khiến người dùng cảm thấy bất tiện. Trong các phiên bản đời đầu, người dùng có thể nhấn vào thông báo để mở ra ứng dụng tương ứng, về sau này các thông tin cập nhật được bổ sung theoe tính năng, như có khả năng lập tức gọi ngược lại khi có cuộc gọi nhỡ mà không cần phải mở ứng dụng gọi điện ra. Thông báo sẽ luôn nằm đó cho đến khi người dùng đã đọc hoặc xóa nó đi.

1.5 Nhân Linux

Android có một hạt nhân dựa trên nhân Linux phiên bản 2.6, kể từ Android 4.0 Ice Cream Sandwich (bánh ngọt kẹp kem) trở về sau, là phiên bản 3.x, với middleware, thư viện và API viết bằng C, còn phần mềm ứng dụng chạy trên một nền tảng ứng dụng gồm các thư viện tương thích với Java dựa trên Apache Harmony. Android sử dụng máy ảo Dalvik với một trình biên dịch động để chạy 'mã dex' (Dalvik Executable) của Dalvik, thường được biên dịch sang Java bytecode. Nền tảng phần cứng chính của Android là kiến trúc ARM. Người ta

cũng hỗ trợ x86 thông qua dự án Android x86, và Google TV cũng sử dụng một phiên bản x86 đặc biệt của Android.

Một số tính năng cũng được Google đóng góp ngược vào nhân Linux, đáng chú ý là tính năng quản lý nguồn điện có tên wakelock, nhưng bị những người lập trình chính cho nhân từ chối vì họ cảm thấy Google không có định sẽ tiếp tục bảo trì đoạn mã do họ viết. Google thông báo vào tháng 4 năm 2010 rằng họ sẽ thuê hai nhận viên để làm việc với cộng đồng nhân Linux, nhưng Greg Kroah-Hartman, người bảo trì nhân Linux hiện tại của nhánh ổn định, đã nói vào tháng 12 năm 2010 rằng ông ta lo ngại rằng Google không còn muốn đưa những thay đổi của mình vào Linux dòng chính nữa. Một số lập trình viên Android của Google tỏ ý rằng "nhóm Android thấy chán với quy trình đó," vì nhóm họ không có nhiều người và có nhiều việc khẩn cấp cần làm với Android hơn.

Vào tháng 8 năm 2011, Linus Torvalds rằng "rốt cuộc thì Android và Linux cũng sẽ trở lại với một bộ nhân chung, nhưng điều đó có thể sẽ không xảy ra trong 4 hoặc 5 năm nữa". Vào tháng 12 năm 2011, Greg Kroah-Hartman thông báo kích hoạt Dự án Dòng chính Android, nhắm tới việc đưa một số driver, bản vá và tính năng của Android ngược vào nhân Linux, bắt đầu từ Linux 3.3. Linux cũng đưa tính năng autosleep (tự nghỉ hoạt động) và wakelocks vào nhân 3.5, sau nhiều nỗ lực phối trộn trước đó. Tương tác thì vẫn vậy nhưng bản hiện thực trên Linux dòng chính cho phép hai chế độ nghỉ: bộ nhớ (dạng nghỉ truyền thống mà Android sử dụng), và đĩa (là ngủ đông trên máy tính để bàn). Việc trộn sẽ hoàn tất kể từ nhân 3.8, Google đã công khai kho mã nguồn trong đó có những đoạn thử nghiệm đưa Android về lại nhân 3.8.

Bộ lưu trữ flash trên các thiết bị Android được chia thành nhiều phân vùng, như "system" dành cho hệ điều hành và "/data" dành cho dữ liệu người dùng và cài đặt ứng dụng. Khác với các bản phân phối Linux cho máy tính để bàn, người sở hữu thiết bị Android không được trao quyền truy cập root vào hệ điều hành và các phân vùng nhạy cảm như /system được thiết lập chỉ đọc. Tuy nhiên, quyền truy cập root có thể chiếm được bằng cách tận dụng những lỗ hổng bảo mật trong Android, điều mà cộng đồng mã nguồn mở thường xuyên sử dụng để nâng cao tính năng thiết bị của họ, kể cả bị những người ác ý sử dụng để cài virus và phần mềm ác ý.

Việc Android có được xem là một bản phân phối Linux hay không vẫn còn là vấn đề gây tranh cãi, tuy được Linux Foundation và Chris DiBona, trưởng nhóm mã nguồn mở Google, ủng hộ. Một số khác, như linux-magazine.com thì không đồng ý, do Android không không hỗ trợ nhiều công cụ GNU, trong đó có glibc.

1.6 Bộ nhớ Android

Vì các thiết bị Android được thiết kế để quản lý bộ nhớ (RAM) để giảm tối đa mức tiêu thụ điện năng, trái với hệ điều hành máy tính để bàn luôn cho rằng máy tính sẽ có nguồn điện không giới hạn. Khi một ứng dụng Android không còn được sử dụng, hệ thống sẽ tự động ngưng nó trong bộ nhớ - trong khi ứng dụng về mặt kỹ thuật vẫn "mở", những ứng dụng này sẽ không tiêu thụ bất cứ tài nguyên nào (như năng lượng pin hay năng lượng xử lý) và nằm đó cho đến khi nó được cần đến. Cách làm như vậy có lợi kép là vừa làm tăng khả năng phản hồi nói chung của thiết bị Android, vì ứng dụng không nhất phải đóng rồi mở lại từ đầu, vừa đảm bảo các ứng dụng nền không làm tiêu hao năng lượng một cách không cần thiết.

Android quản lý các ứng dụng trong bộ nhớ một cách tự động: khi bộ nhớ thấp, hệ thống sẽ bắt đầu diệt ứng dụng và tiến trình không hoạt động được một thời gian, sắp theo thời điểm cuối mà chúng được sử dụng (tức là cũ nhất sẽ bị tắt trước). Tiến trình này được thiết kế ẩn đi với người dùng, để người dùng không cần phải quản lý bộ nhớ hoặc tự tay tắt các ứng dụng. Tuy nhiên, sự che giấu này của hệ thống quản lý bộ nhớ Android đã dẫn đến sự thịnh hành của các ứng dụng tắt chương trình của bên thứ ba trên cửa hàng Google Play; những ứng dụng kiểu như vậy được cho là có hại nhiều hơn có lợi.

1.7 Bảo mật của Android

Các ứng dụng Android chạy trong một "hộp cát", là một khu vực riêng rẽ với hệ thống và không được tiếp cận đến phần còn lại của tài nguyên hệ thống, trừ khi nó được người dùng trao quyền truy cập một cách công khai khi cài đặt. Trước khi cài đặt ứng dụng, Cửa hàng Play sẽ hiển thị tất cả các quyền mà ứng dụng đòi hỏi: ví dụ như một trò chơi cần phải kích hoạt bộ rung hoặc lưu dữ liệu vào thẻ nhớ SD, nhưng nó không nên cần quyền đọc tin nhắn SMS hoặc tiếp cận danh bạ điện thoại. Sau khi xem xét các quyền này, người dùng có thể chọn đồng ý hoặc từ chối chúng, ứng dụng chỉ được cài đặt khi người dùng đồng ý.

Hệ thống hộp cát và hỏi quyền làm giảm bớt ảnh hưởng của lỗi bảo mật hoặc lỗi chương trình có trong ứng dụng, nhưng sự bối rối của lập trình viên và tài liệu hướng dẫn còn hạn chế đã dẫn tới những ứng dụng hay đòi hỏi những quyền không cần thiết, do đó làm giảm đi hiệu quả của hệ thống này. Một số công ty bảo mật, như Lookout Mobile Security, AVG Technologies, và McAfee, đã phát hành những phần mềm diệt virus cho các thiết bị Android. Phần mềm này không có hiệu quả vì cơ chế hộp cát vẫn áp dụng vào các ứng dụng này, do vậy làm hạn chế khả năng quét sâu vào hệ thống để tìm nguy cơ.

Một nghiên cứu của công ty bảo mật Trend Micro đã liệt kê tình trạng lạm dụng dịch vụ trả tiền là hình thức phần mềm ác ý phổ biến nhất trên Android, trong đó tin nhắn SMS sẽ bị gửi đi từ điện thoại bị nhiễm đến một số điện thoại trả tiền mà người dùng không hề hay biết. Loại phần mềm ác ý khác hiển thị những quảng cáo không mong muốn và gây khó chịu trên thiết bị, hoặc gửi thông tin cá nhân đến bên thứ ba khi chưa được phép. Đe dọa bảo mật trên Android được cho là tăng rất nhanh theo cấp số mũ; tuy nhiên, các kỹ sư Google phản bác rằng hiểm họa từ phần mềm ác ý và virus đã bị thổi phồng bởi các công ty bảo mật nhằm mục đích thương mại, và buộc tội ngành công nghiệp bảo mật đang lợi dụng sự sợ hãi để bán phần mềm diệt virus cho người dùng. Google vẫn giữ quan điểm rằng phần mềm ác ý thật sự nguy hiểm là cực kỳ hiếm, và một cuộc điều tra do F-Secure thực hiện cho thấy chỉ có 0,5% số phần mềm ác ý Android là len vào được cửa hàng Google Play.

Google hiện đang sử dụng bộ quét phần mềm ác ý Google Bouncer để theo dõi và quét các ứng dụng trên Cửa hàng Google Play. Nó sẽ đánh dấu các phần mềm bị nghi ngờ và cảnh báo người dùng về những vấn đề có thể xảy ra trước khi họ tải nó về máy. Android phiên bản 4.2 Jelly Bean được phát hành vào năm 2012 cùng với các tính năng bảo mật được cải thiện, bao gồm một bộ quét phần mềm ác ý được cài sẵn trong hệ thống, hoạt động cùng với Google Play nhưng cũng có thể quét các ứng dụng được cài đặt từ nguồn thứ ba, và một hệ thống cảnh báo sẽ thông báo cho người dùng khi một ứng dụng cố gắng gửi một tin nhắn vào số tính tiền, chặn tin nhắn đó lại trừ khi người dùng công khai cho phép nó.

Điện thoại thông minh Android có khả năng báo cáo vị trí của điểm truy cập Wi-Fi, phát hiện ra việc di chuyển của người dùng điện thoại, để xây dựng

những cơ sở dữ liệu có chứa vị trí của hàng trăm triệu điểm truy cập. Những cơ sở dữ liệu này tạo nên một bản đồ điện tử để tìm vị trí điện thoại thông minh, cho phép chúng chạy các ứng dụng như Foursquare, Google Latitude, Facebook Places, và gửi những đoạn quảng cáo dựa trên vị trí. Phần mềm theo dõi của bên thứ ba như TaintDroid, một dự án nghiên cứu trong trường đại học, đôi khi có thể biết được khi nào thông tin cá nhân bị gửi đi từ ứng dụng đến các máy chủ đặt ở xa.

Bản chất mã nguồn mở của Android cho phép những nhà thầu bảo mật lấy những thiết bị sẵn có rồi điều chỉnh để sử dụng ở mức độ bảo mật cao hơn.

Chương 2: MÔI TRƯỜNG LẬP TRÌNH ANDROID STUDIO

2.1 Sơ lược về Android Studio

Google cung cấp một công cụ phát triển ứng dụng Android trên Website chính thức dựa trên nền tảng IntelliJ IDEA gọi là Android Studio. Android studio dựa vào IntelliJ IDEA, là một IDE tốt cho nhất Java hiện nay. Do đó Android Studio sẽ là môi trường phát triển ứng dụng tốt nhất cho Android.

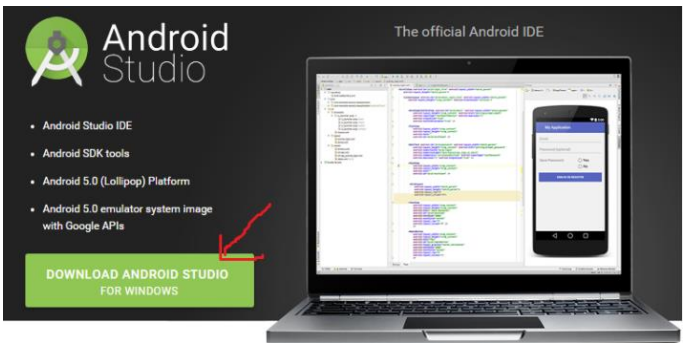
2.2 Cài đặt android studio

2.2.1 Yêu cầu phần cứng máy tính

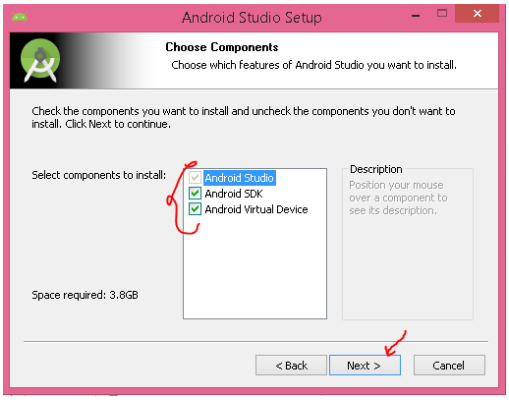
- Microsoft® Windows® 8/7 (32 or 64-bit)
- 4 GB RAM. (Tốt nhất là 8GB)
- 400 MB hard disk space + ít nhất 1GB cho Android SDK, emulator system images và caches
- Độ phân giải tối thiểu 1280 x 800
- Java Development Kit ()

2.2.2 Phần mềm Android Studio

- Vào đường dẫn: "http://developer.android.com/sdk/index.html"
- Để download bản mới nhất và tiến hành cài đặt click như hình:



- Khi cài đặt chú ý chọn cả SDK và trình giả lập thiết bị android như hình:

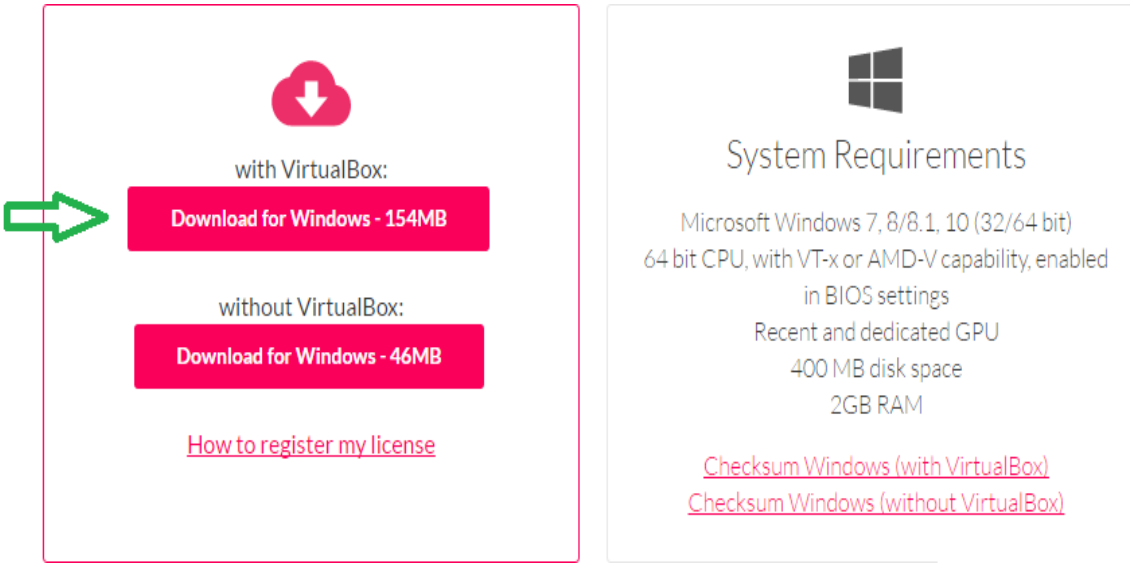


- Tiếp tục chọn next và agree cho đến khi hoàn tất.
- Đây là màn hình khởi động.

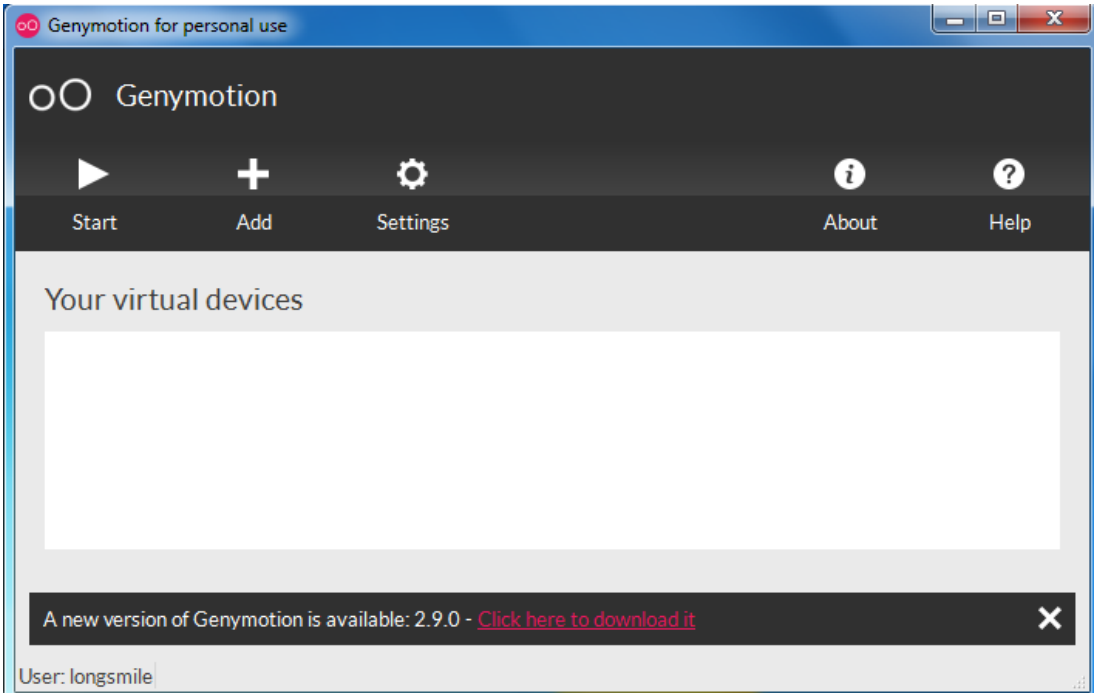
2.2.3 Thiết bị ảo trong android studio

- Máy ảo Android là một phần không thể thiếu khi chúng ta lập trình ứng dụng cho hệ điều hành Android, nó giúp chúng ta chạy thử ứng dụng ngay trên máy tính. Và công cụ máy ảo tiện dụng hiện giờ là Genymotion.
- Để cài đặt máy ảo **Genymotion** ta truy cập vào đường dẫn :

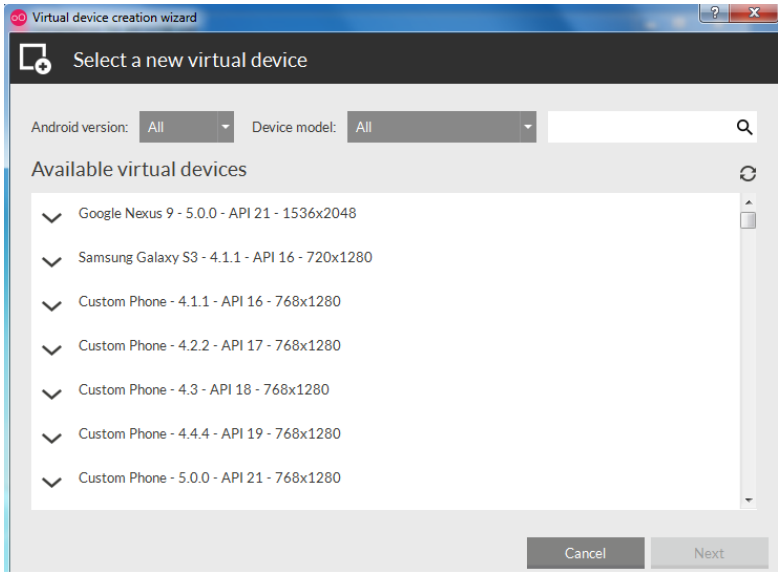
https://www.genymotion.com/download/



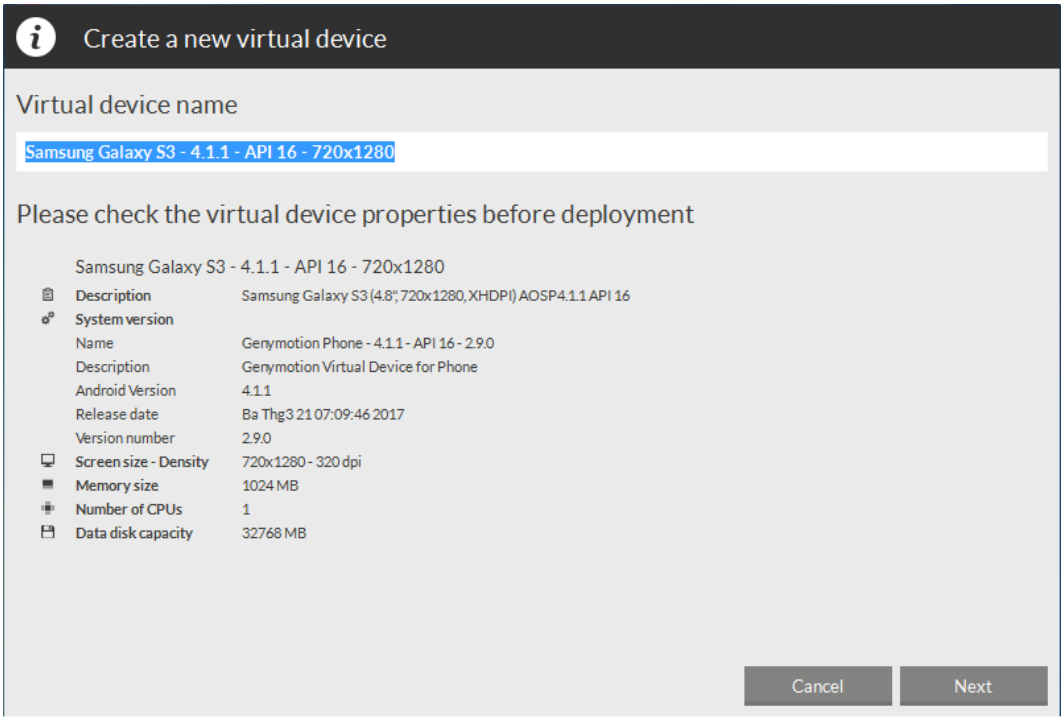
- Ta phải tạo một tài khoản rồi đăng nhập vào mới thấy được mục này.
- Ở đây ta nên tải phiên bản with VirtuaBox , vì chương trình tích hợp sẵn VirtuaBox cho chúng ta vì máy ảo này phải có VirtuaBox mới chạy được.
- Sau khi cài xong sẽ có giao diện như sau :



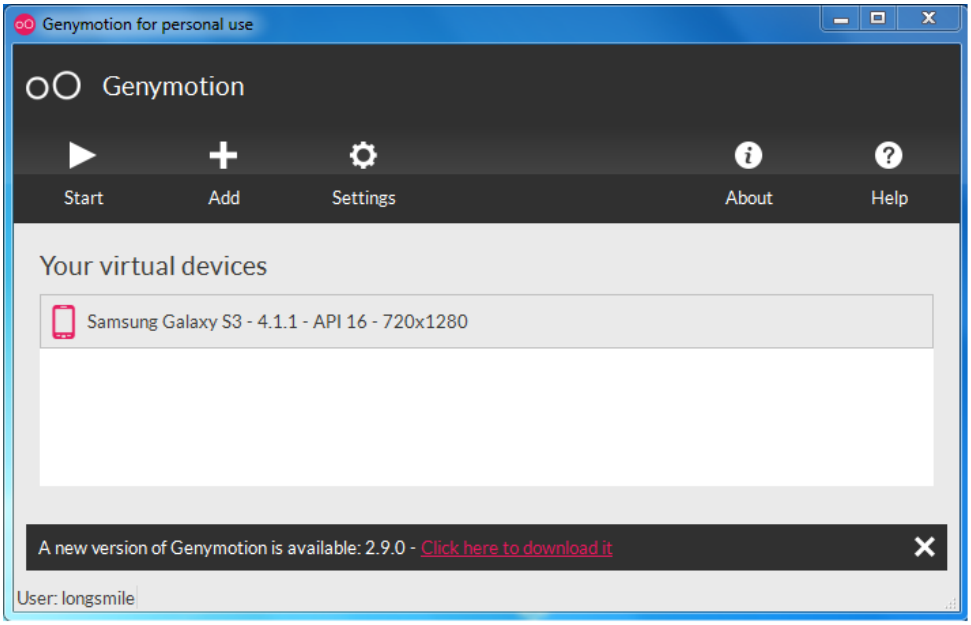
- Chúng ta cần đăng nhập vào tạo một cái máy ảo. Như ở đây tôi đã đăng nhập với tài khoản User :longsmile
- Các phím chức năng của máy ảo Genymotion
    - **Start** : Bắt đầu khởi động máy ảo
    - **Add :** Tạo máy ảo
    - **Setting :** Cài đặt
- Nhấn vào Add để tạo máy ảo.



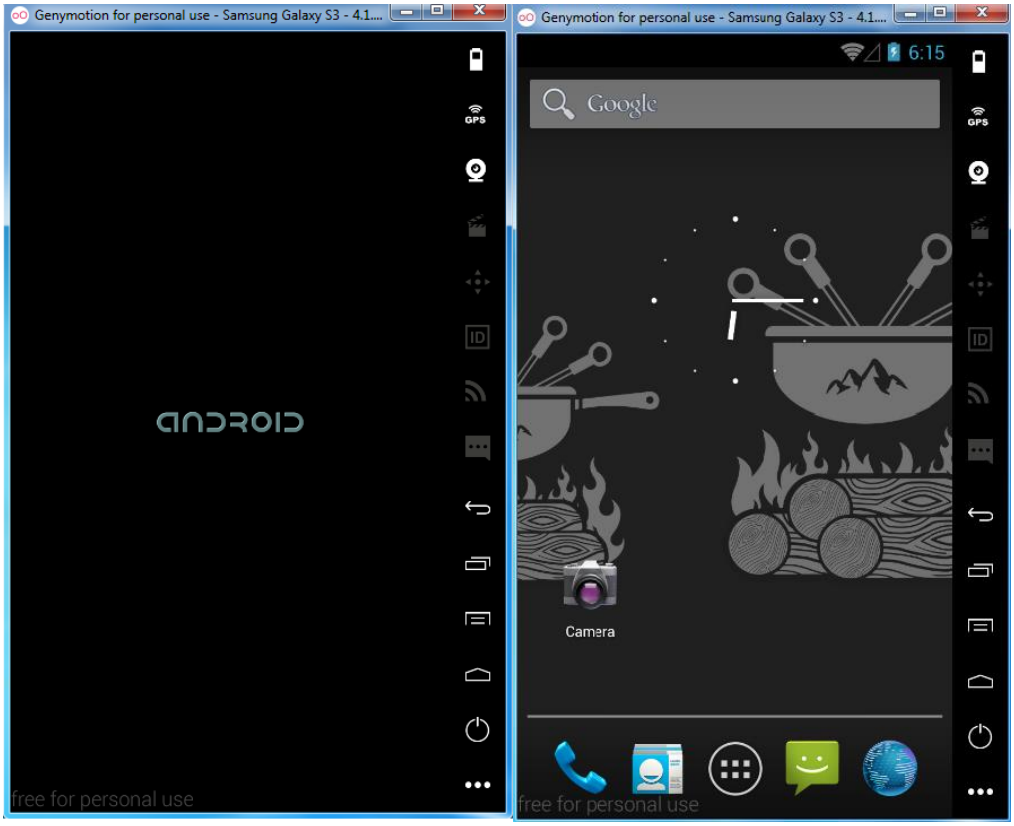
- Sau đó chọn một hoặc nhiều thiết bị theo ý muốn (như ở đây tôi chọn Samsung Galaxy S3 -4.1.1 - API 16 - 720x1280) :
    - Samsung Galaxy S3 : là tên thiết bị máy ảo
    - 4.1.1 : là phiên bản hệ điều hành Android
    - API :Application Programming Interface (giao diện lập trình ứng dụng)
    - 720x1280 : Độ phân giải màn hình
- Bạn chọn thiết bị sau đó nhấn Next sẽ có giao diện thông tin thiết bị :



- Sau đó tiếp tục nhấn Next và Finish để hoàn thành cài máy ảo

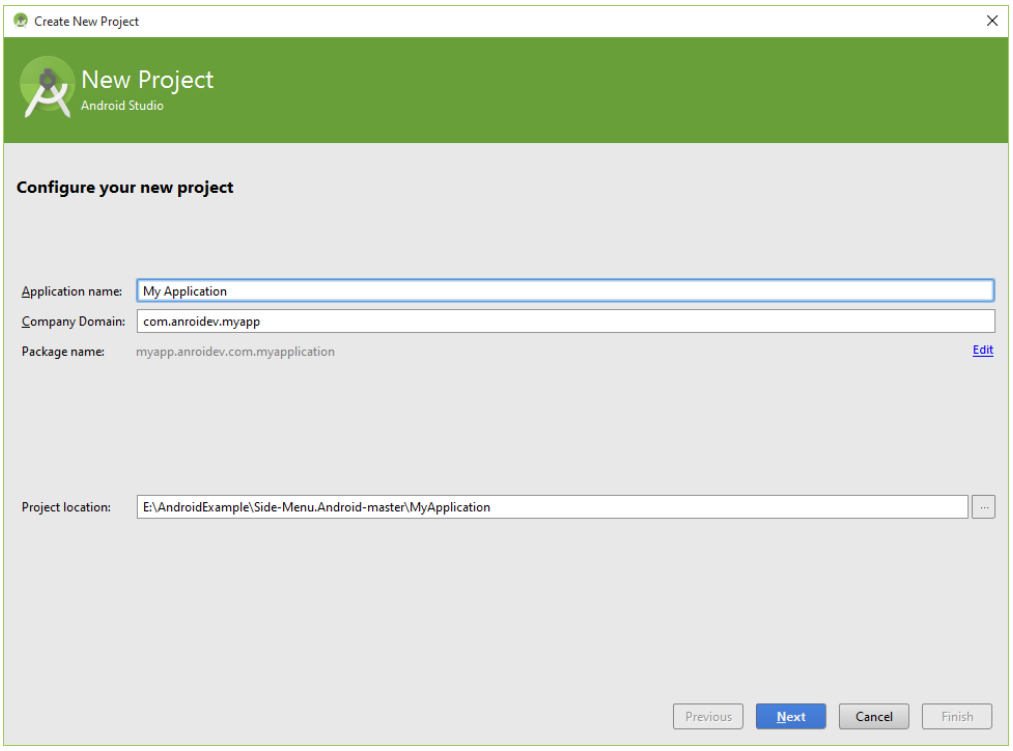


- Ta tiếp tục nhấn vào thiết bị và nhấn Start để khởi động máy ảo



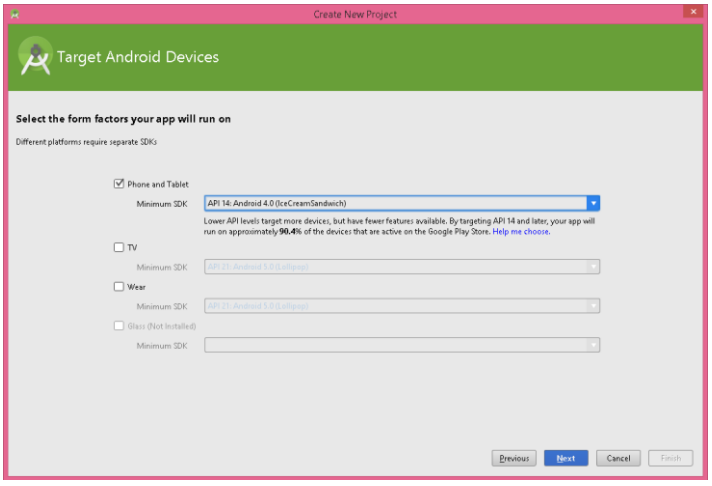
2.3 Cấu trúc dự án android studio

2.3.1 Tạo mới một project



- **Application Name:** Tên ứng dụng muốn đặt
- **Company Domain:** Tên domain công ty, thường được dùng để kết hợp với tên Application để tạo thành Package (chú ý viết thường hết và có ít nhất 1 dấu chấm).
- **Package name:** Nó sẽ tự động nối ngược Company Domain với Application name.
- **Project location:** Là nơi lưu trữ ứng dụng.
- Sau đó nhấp Next để tiếp tục.

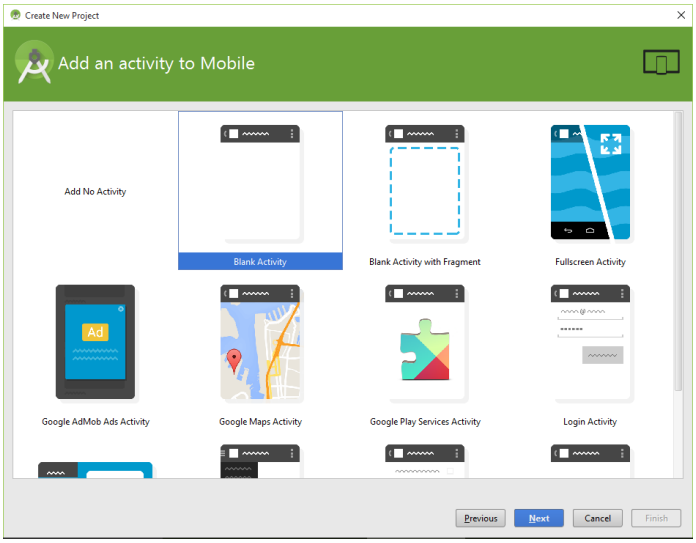
2.3.2 Cài đặt một project.



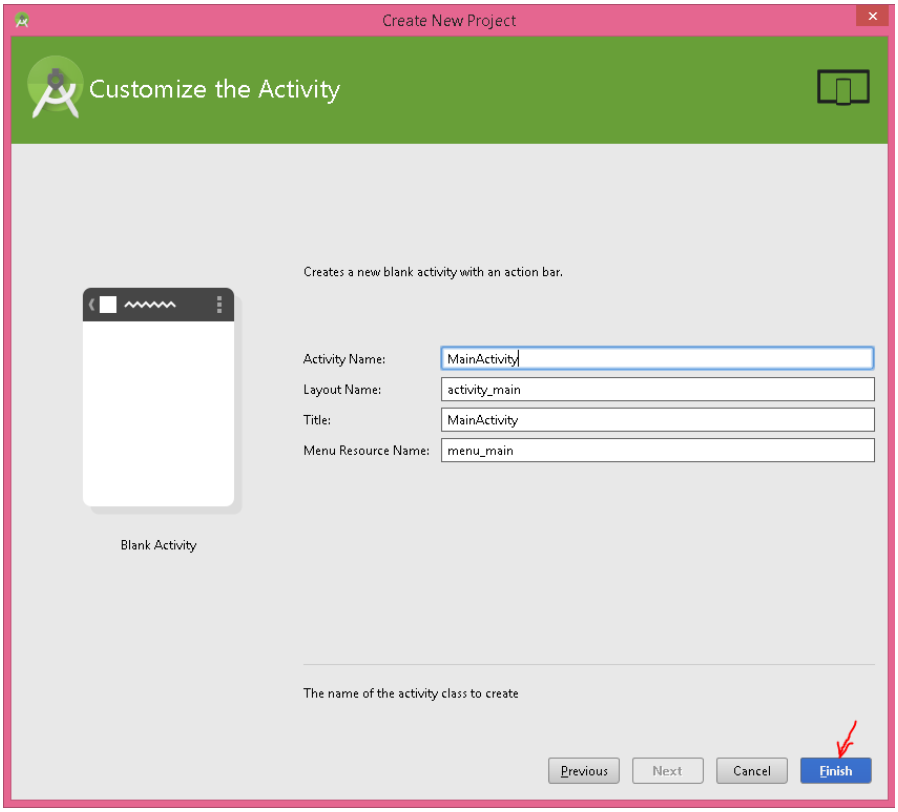
Ở hộp thoại trên cho phép ta lựa chọn là ứng dụng sẽ được viết cho những thiết bị nào (Phone and Tablet, TV, Wear).

Ở mục Minium SDK, quy định phiên bản android tối thiểu để chạy ứng dụng.

Hiện nay bản **API14 Android 4.0** (**IceCreamSandwich**) vẫn đứng đầu về số lượng thiết bị sử dụng chiếm tới hơn 90%) nên thường lựa chọn. Màn hình này hiển thị cho phép chọn loại Activity mặc định.



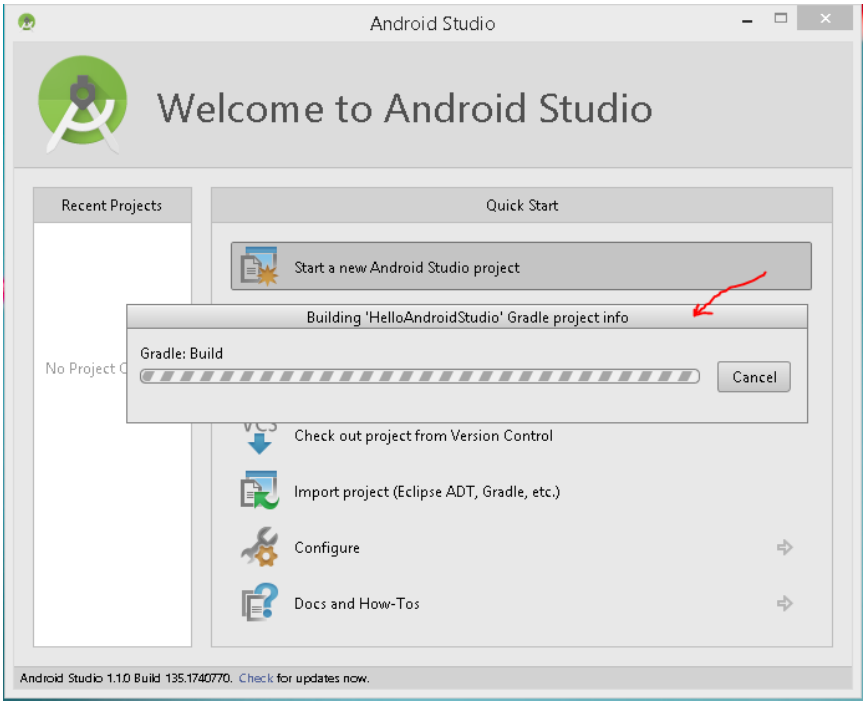
Chọn **Blank Activity** rồi bấm Next:



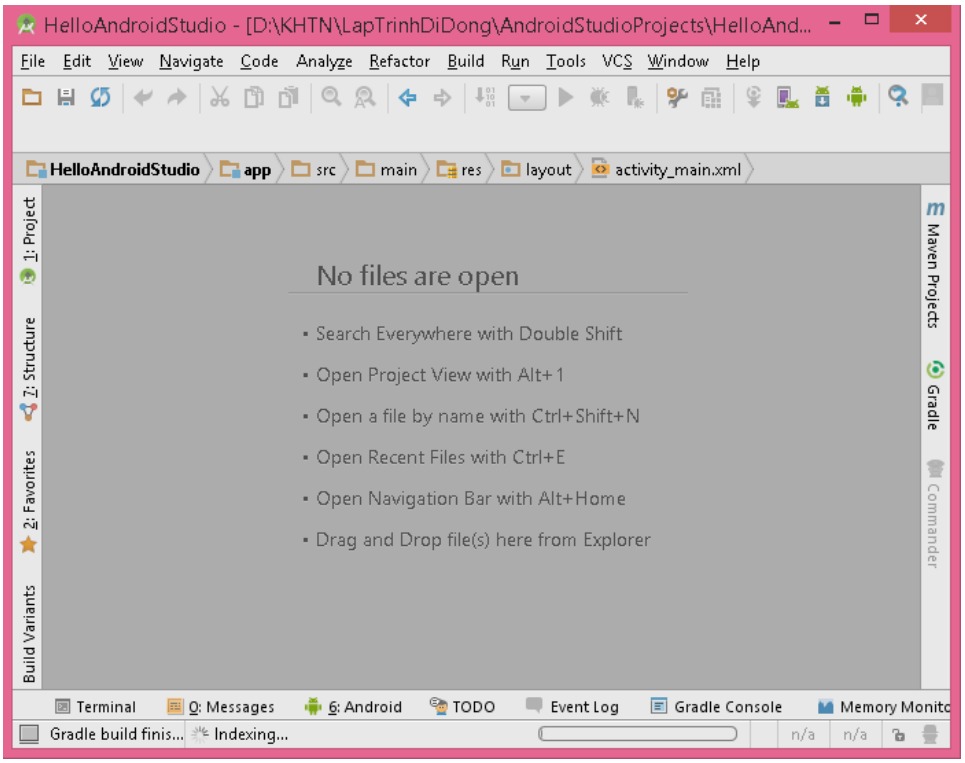
- **Activity Name:** Tên class Activity (java) để ta viết mã lệnh

- **Layout Name:** Tên file XML làm giao diện cho Activity Name.
- **Title:** Tiêu đề hiển thị khi kích hoạt Activity trên thiết bị.
- **Menu Resource Name:** Tên file xml để tạo menu cho phần mềm.

Sau khi cấu hình xong, bấm Finish, Màn hình Build Gradle project hiển thị:



Khi build xong mặc định có màn hình dưới đây:

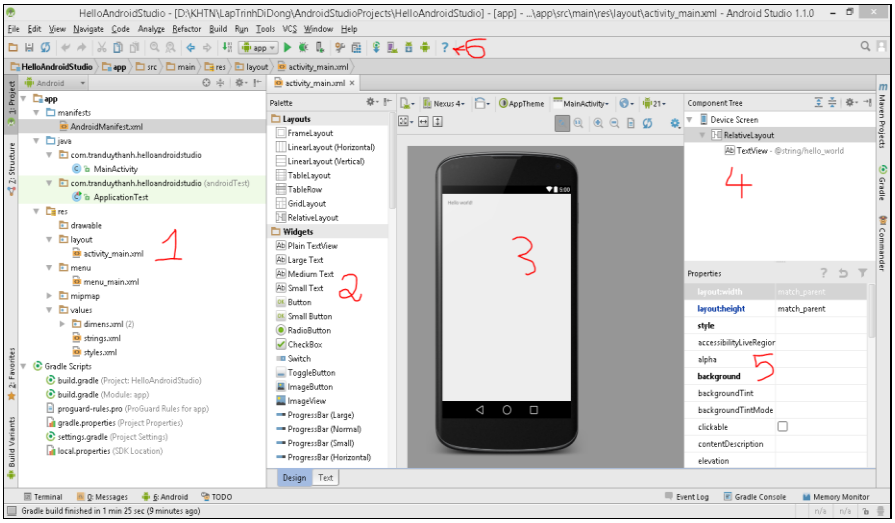


2.3.3 Màn hình làm việc của dự án android studio

Theo mặc định Android Studio hiển thị các files trong project theo góc nhìn Android. Góc nhìn này Android Studio sẽ tổ chức các files theo 3 module:

- manifests: chứa file AndroidManifest.xml.
- java: chứa các file mã nguồn Java.
- res: chứa tất cả các file layout, xml, giao diện người dùng(UI), ảnh

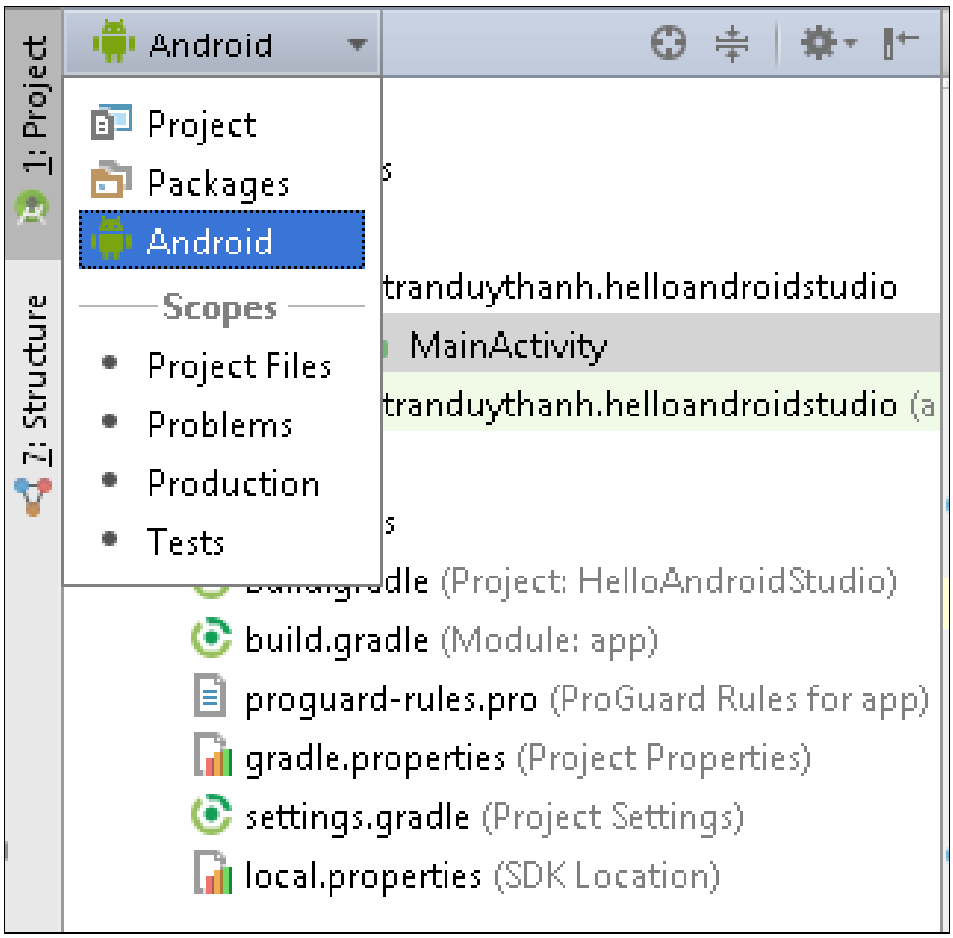
Mở Project mặc định **activity_main.xml** sẽ được chọn ta có màn hình như sau:



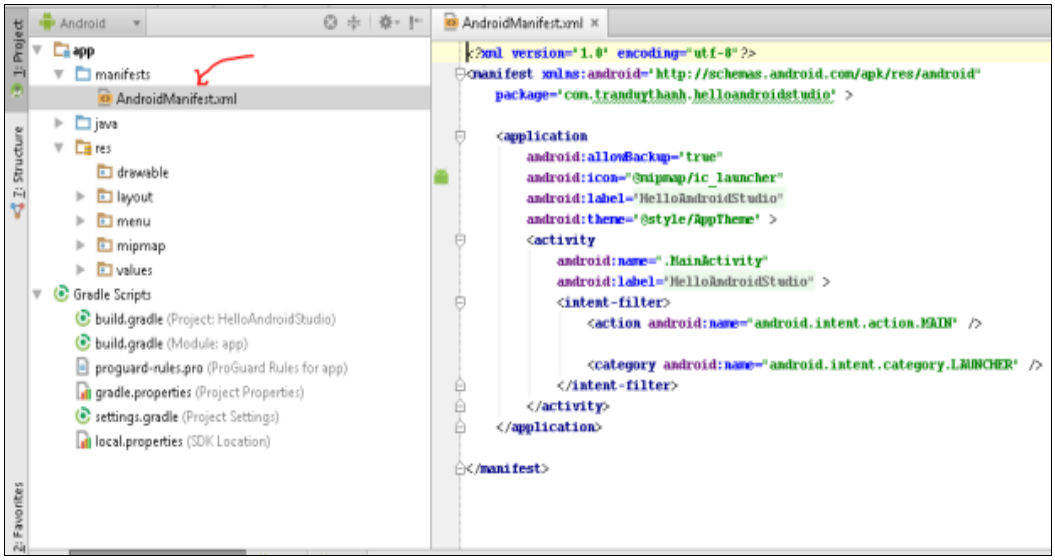
Ở trên tạm thời chia làm 6 vùng làm việc mà lập trình viên thường tương tác.

- **Vùng 1.**

Là nơi cấu trúc hệ thống thông tin của Ứng dụng, Ta có thể thay đổi cấu trúc hiển thị (thường để mặc định là **Android**).



Màn hình ở chế độ Android:



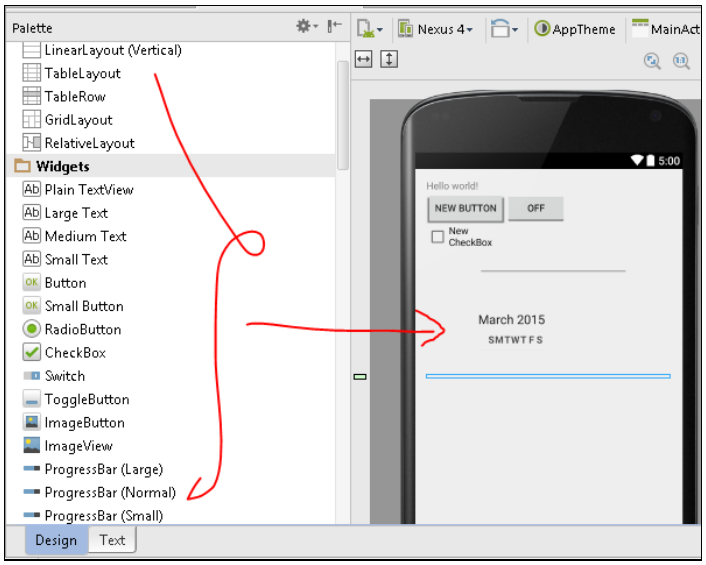
Ta có thể thấy **AndroidManifest.xml** nằm ở đây. File này vô cùng quan trọng trong việc cấu hình ứng dụng.

Các thư mục:

- **drawable**: chứa các file hình ảnh và xml trong ứng dụng.
- **layout**: chứa các giao diện màn hình được thiết kế dưới dạng xml.
- **values**: chứa các file lưu giá trị màu sắc, kích thước, chuỗi,....

- **Vùng 2.**

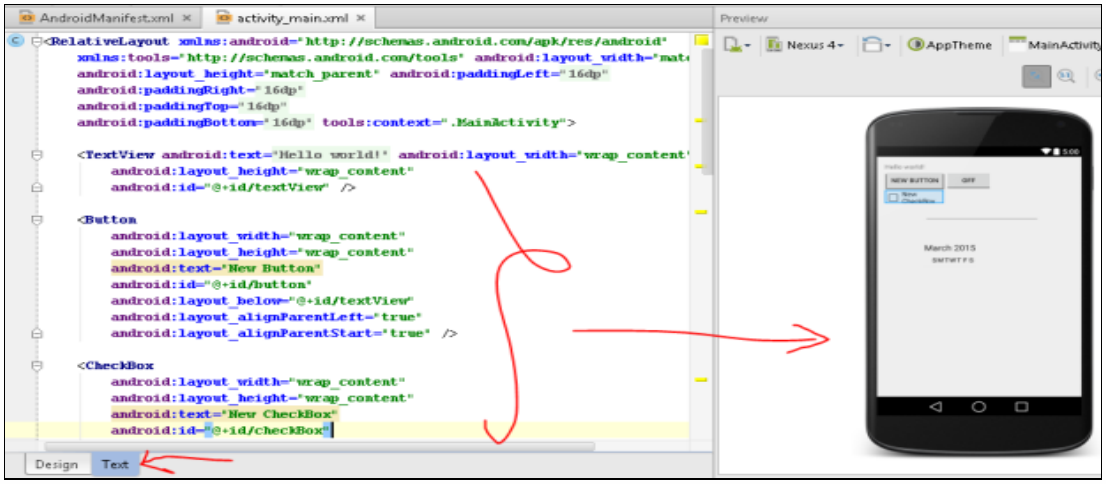
Là vùng khá quan trọng cho những bạn mới bắt đầu lập trình, nó là nơi hiển thị các Control mà Android hỗ trợ, cho phép bạn kéo thả trực tiếp vào vùng 3 (Giao Diện Thiết Bị) để thiết kế.



Ở vùng số 2 này nó có 2 tab: **Design** và **Text** ở góc trái dưới cùng.

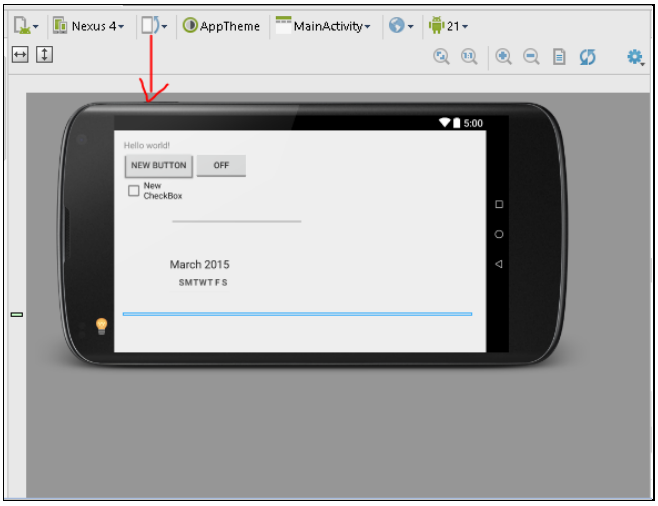
- Tab **Design** là tab mà ta đang nhìn và thao tác với nó (cho phép thiết kế giao diện bằng cách kéo thả).

- Tab **Text** là tab cho phép ta thiết kế giao diện bằng viết Tag XML:



- **Vùng 3.**

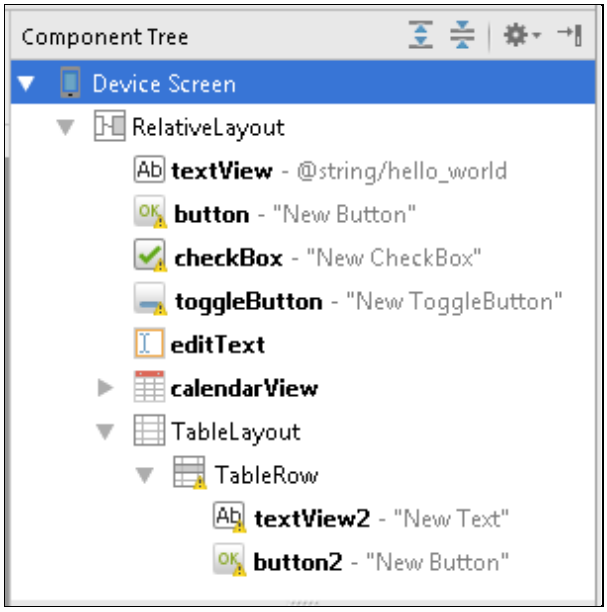
Là vùng giao diện thiết bị, cho phép các Control kéo thả vào đây và đồng thời cho ta hiểu chính control.



Vùng 3 ta có thể chọn cách hiển thị theo nằm ngang nằm đứng, phóng to thu nhỏ, căn chỉ control, lựa chọn loại thiết bị hiển thị….

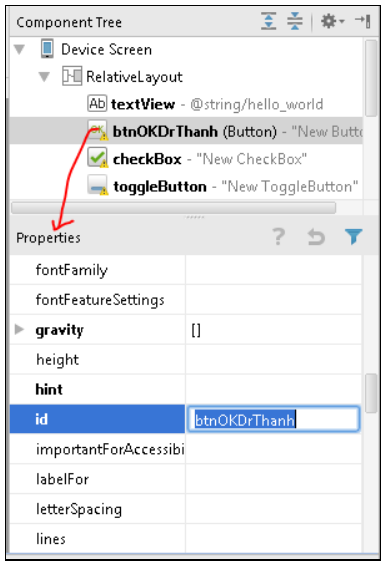
- **Vùng 4.**

Khi màn hình có nhiều control thì vùng 4 này trở lên hữu ích, nó cho phép hiển thị giao diện theo dạng cấu trúc cây, nên ta dễ dàng quan sát và lựa chọn control khi chúng bị chồng lập trên giao diện (vùng 3).



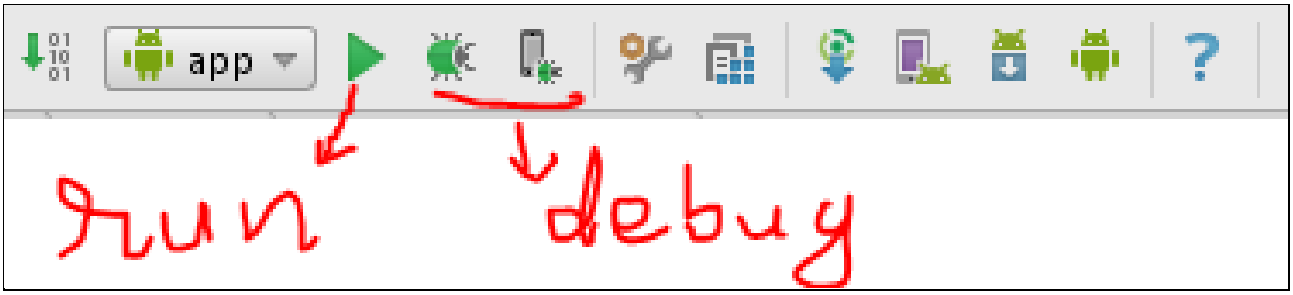
- **Vùng 5.**

Vùng này rất quan trọng, đây là vùng cho phép thiết lập trạng thái hay thuộc tính cho các Control trên giao diện.

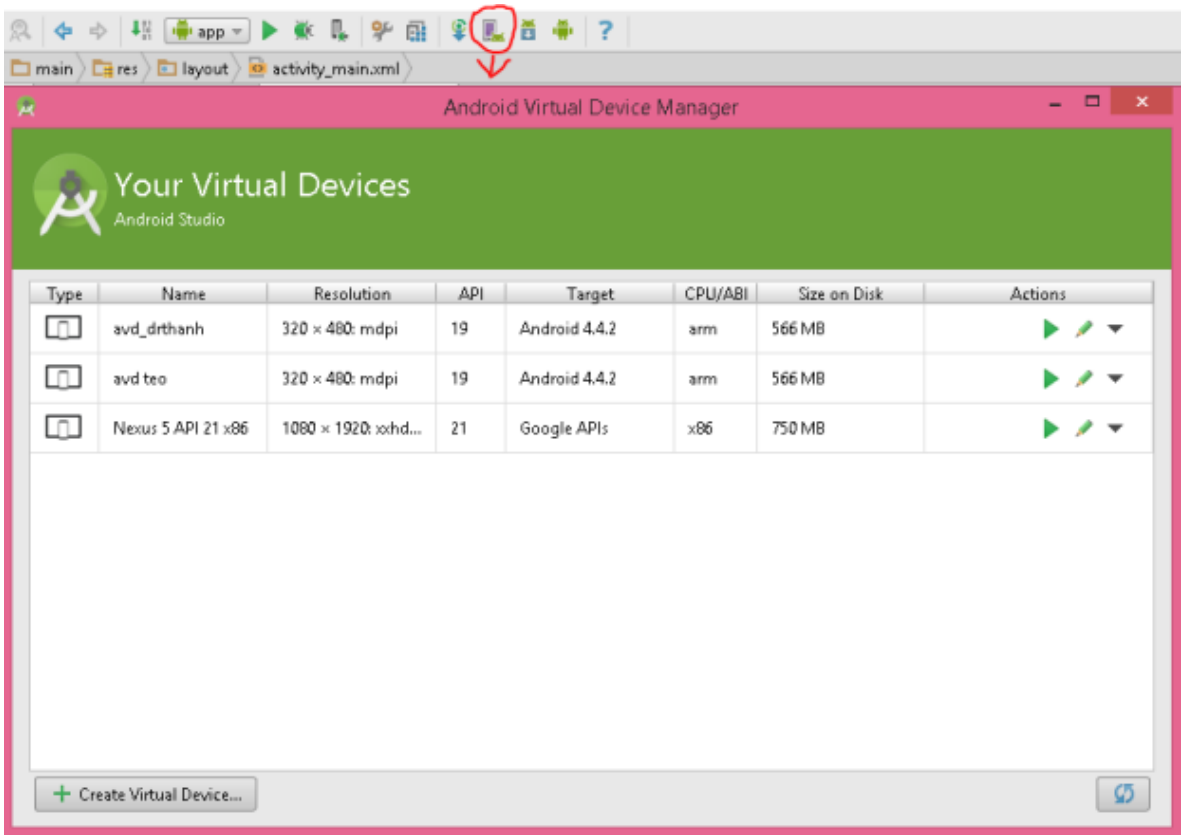


- **Vùng 6**

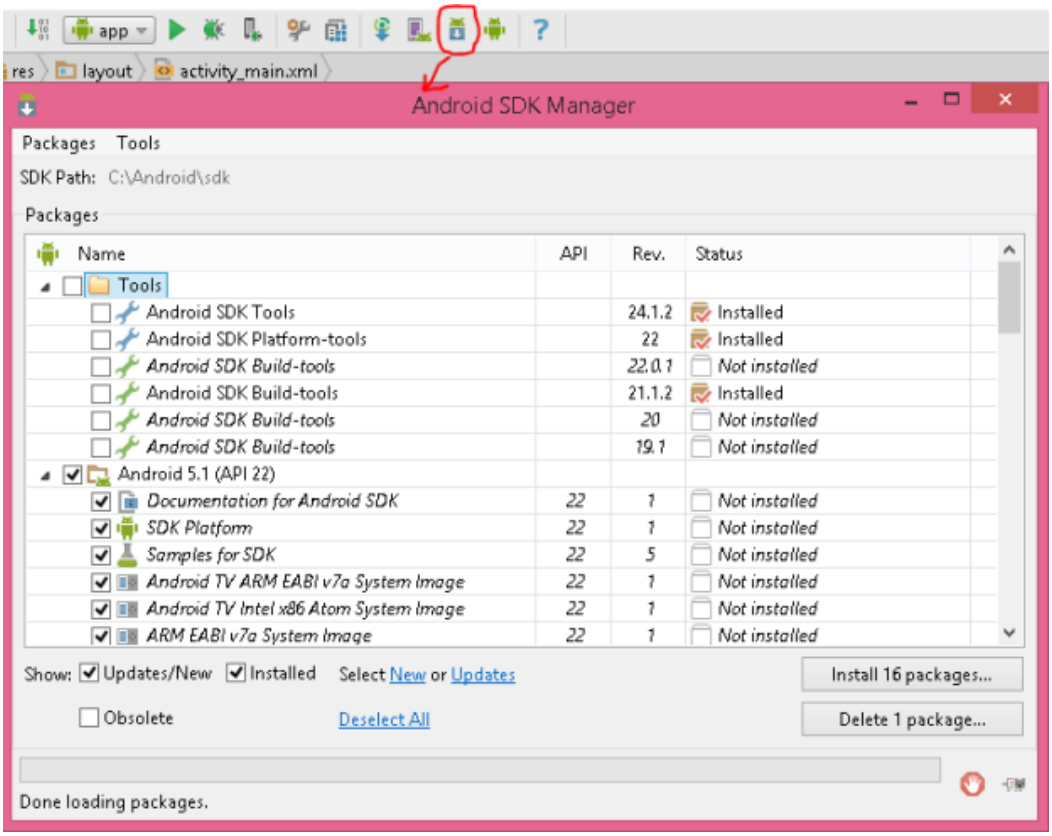
Là vùng các chức năng quan trọng thường dùng trong Android Studio.



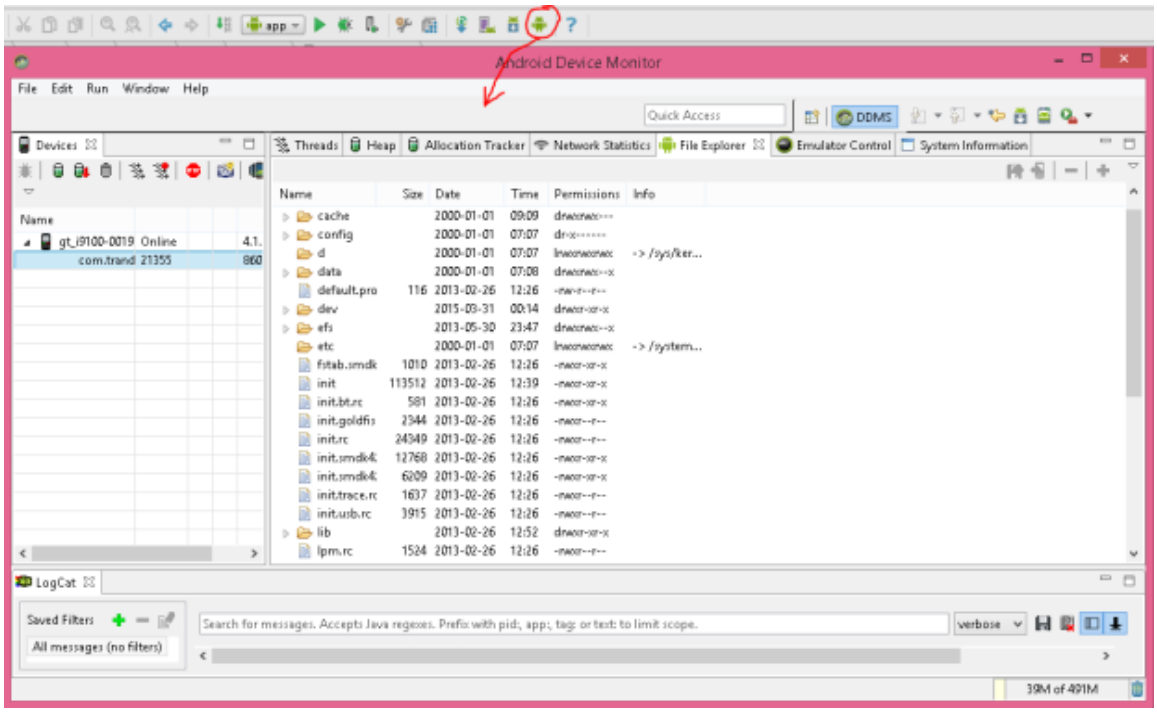
Quản lý máy ảo (AVD Manager)



Quản lý Android SDK Manager (thường dùng để cập nhật).



Quản lý Android Device Manager



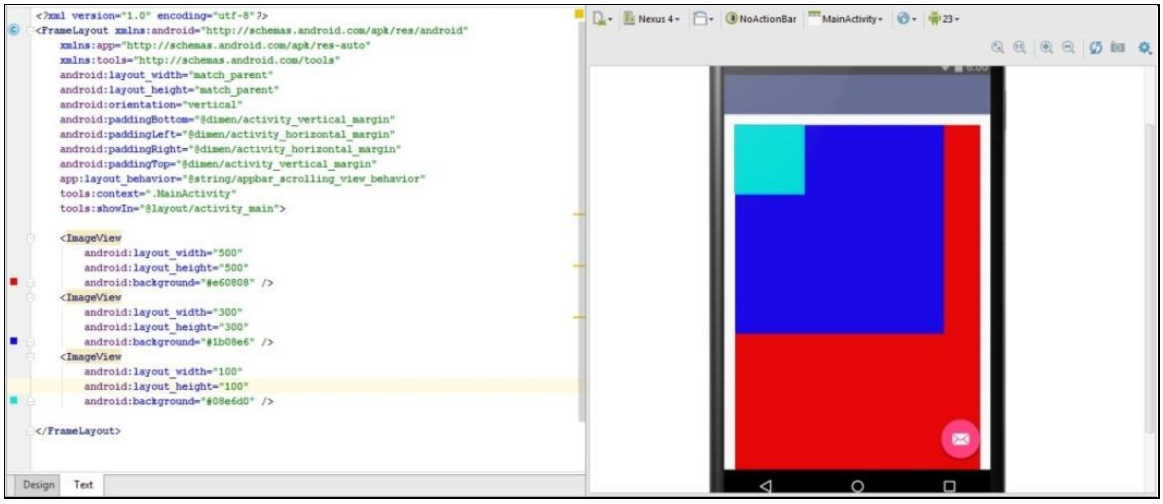
2.4 Tạo giao diện chương trình trong android studio

2.4.1 Giới thiệu android Layout

Layout là nơi chứa các control lên giao diện và mỗi layout có một cách sắp xếp các control khác nhau, vì vậy với mỗi cấu trúc giao diện khác nhau ta nên chọn layout cho phù hợp. Sau đây là một số layout cơ bản cho để ta thiết kế giao diện.

- FrameLayout.

Là loại layout cơ bản nhất, nó sẽ được dùng nhiều khi ta sử dụng vẽ giao diện nâng cao sau này. Khi ta kéo các control vào thì mặc định các control sẽ nằm ở vị trí trên cùng bên trái. Các control khi được kéo vào framelayout sẽ bị đè lên nhau, control sau sẽ đè lên control trước. Cách duy nhất để căn các control vào giữa là sử dụng thuộc tính android:layout_gravity="center". Ta có thể tham khảo đoạn XML sau để hiểu thêm về framelayout.



- LinearLayout.

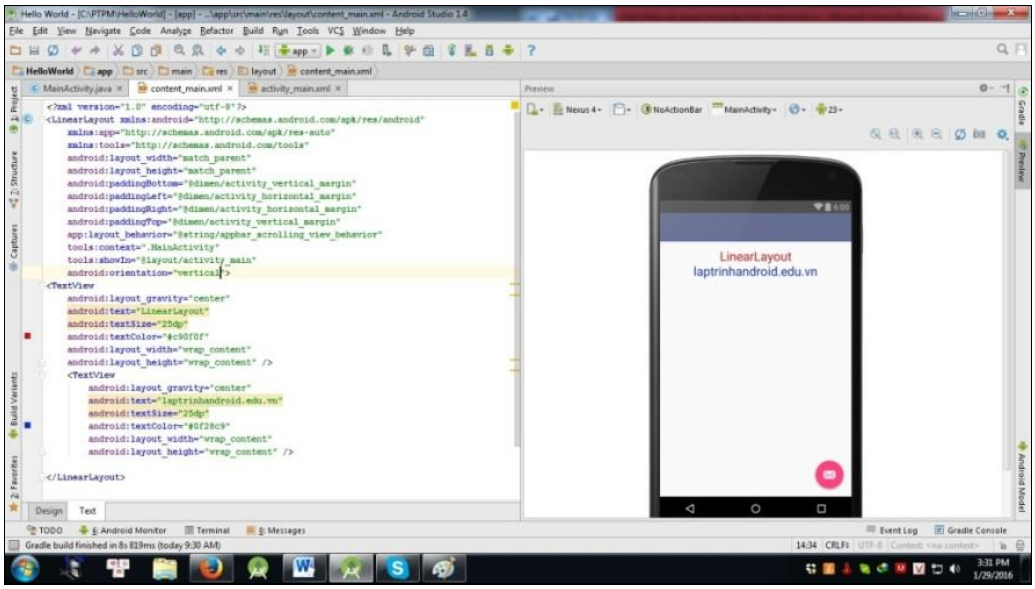
Layout này cho phép ta vẽ giao diện theo 2 hướng, từ trái qua phải hoặc từ trên xuống dưới.Để xét chiều cho các control trong layout ta sử dụng thuộc tính **orientation.**

- Android:orientation="horizontal" : Xếp các control từ trái sang phải (theo cột).
- Android:orientation="vertical" : Xếp các control từ trên xuống dưới (theo hàng).

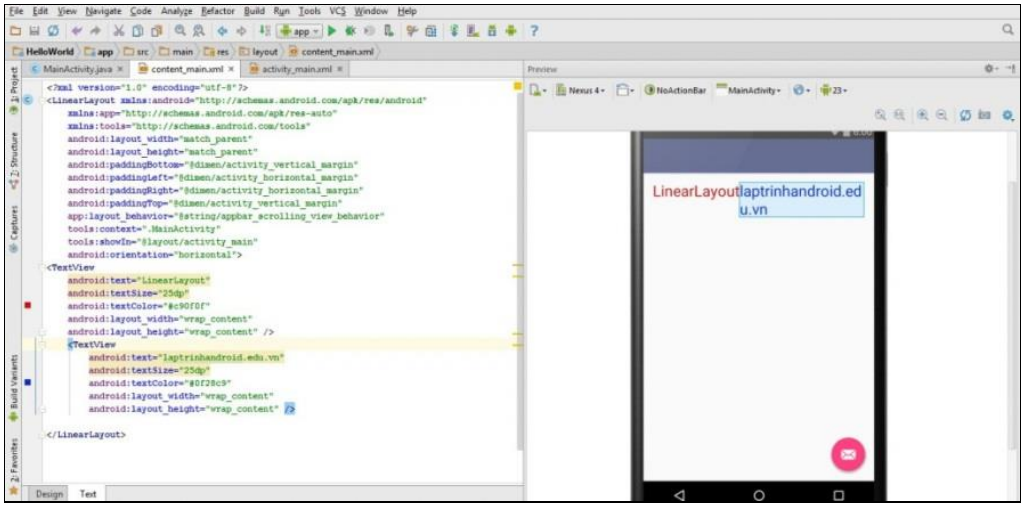
Với những giao diện có độ phức tạp vừa phải thì dùng LinearLayout là rất hiệu quả, rất thuận tiện trong thiết kế và đi bảo trì ứng dụng sau này.

Sau đây là đoạn XML demo cách sử dụng layout này:

Theo hàng



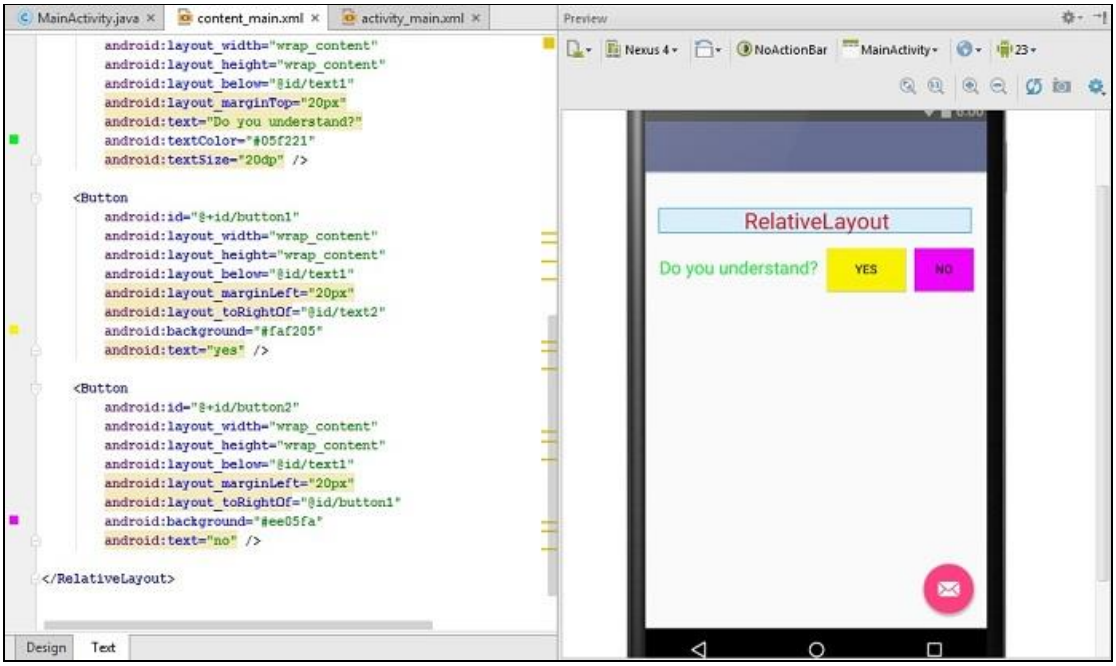
Theo cột



- RelativeLayout.

Layout này cho phép ta sắp xếp các control theo vị trí tương đối giữa các control khác kể cả control chứa nó. Khi gặp những layout có độ phức tạp cao, có nhiều giao diện nhỏ thì sử dụng RelativeLayout là lựa chọn tốt nhất. Một vài chú ý khi sử dụng layout này:

- Các control đều có id riêng, việc đặt tên id phải rõ rang dễ hiểu.
- Các control được sắp xếp dựa vào id của các control khác.
- Các control có sự ràng buộc và tương tác với nhau nên khi thay đổi một control sẽ làm thay đổi vị trí của mọi control khác. Vì vậy rất khó trong việc bảo trì nếu giao diện quá phức tạp.
- Ta có thể tham khảo đoạn XML demo sau để hình dung dễ hơn:



2.4.2 Giới thiệu một số android View cơ bản

- **TextView**: là view sử dụng để hiển thị text màn hình. TextView được định nghĩa bởi thẻ <TextView> trong xml.
- **EditText**: là view dùng để lấy giá trị từ người dùng nhập vào. EditText được định nghĩa bởi thẻ <EditText>trong xml.
- **ImageView**: là một view sử dụng rất nhiều trong ứng dụng android, ImageView sử dụng để hiển thị hình ảnh.
- **Button**: là view được sử dụng khá nhiều trong android, hầu như sử dụng ở mọi nơi cùng với EditText, TextView. Button có chức năng là làm nhiệm vụ nào đó khi mà người dùng click trong phương thức onClick.
- **ListView**: được tạo từ một danh sách các ListItem. ListItem là một dòng (row) riêng lẻ trong listview nơi mà dữ liệu sẽ được hiển thị. Bất kỳ dữ liệu nào trong listview chỉ được hiển thị thông qua listItem. Có thể coi listview như là một nhóm cuộn của các ListItem.

2.4.3 Bắt và xử lý sự kiện trên giao diện.

Sự kiện là một cách hữu ích để thu thập dữ liệu về sự tương tác của người dùng với các thành phần tương tác của ứng dụng. Giống như bấm vào một nút hoặc chạm vào màn hình cảm ứng, vv. Ta có thể nắm bắt những sự kiện trong chương trình và có những xử lý thích hợp theo yêu cầu.

Có hai khái niệm liên quan đến quản lý sự kiện Android:

- **Event Listeners** là một interface. Event Listeners được sử dụng để đăng ký sự kiện cho các thành phần trong UI. (Đăng ký sự kiện). Trong các giao tiếp event listener có những phương thức sau đây:
  - **onClick()**: Thuộc View.OnClickListener. Nó được gọi khi người dùng hoặc chạm vào item (khi ở chế độ cảm ứng), hoặc lựa chọn vào item với các phím điều hướng và nhấn nút "enter" phù hợp.
  - **onLongClick():** Thuộc **View.OnLongClickListener.** Nó được gọi khi người dùng chạm và giữ item (khi ở chế độ cảm ứng), hoặc lựa chọn vào item với các phím điều hướng sau đó nhấn và giữ phím "enter".
  - **onFocusChange():** Thuộc **View.OnFocusChangeListener.** Nó được gọi khi người dùng điều hướng ra khỏi item, bằng cách sử dụng phím điều hướng.
  - **onKey():** Thuộc **View.OnKeyListener**. Nó được gọi khi người dùng lựa chọn và nhấn lên item.

- **onTouch():** Thuộc **View.OnTouchListener.** Nó được gọi khi người dùng thực hiện một hành động xác định đủ điều kiện như là một sự kiện cảm ứng, bao gồm việc nhấn, thoát ra, hoặc bất kỳ cử chỉ chuyển động vẽ trên màn hình (bên trong phạm vi của item).
- **onCreateContextMenu()**: Thuộc **View.OnCreateContextMenuListener.** Nó được gọi khi một menu ngữ cảnh (Context Menu) đang được xây dựng (là kết quả của một "long click"). Xem thêm thông tin về context menus trong hướng dẫn phát triển Menus**.**
- Ví dụ dưới đây cho thấy làm thế nào để đăng ký một bộ bắt sự kiện khi nhấp chuột vào một Button.

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

Ta cũng có thể tìm thấy cách thuận tiện hơn để bổ sung OnClickListener như một phần Activity. Ví dụ:

```
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    // Implement the OnClickListener callback
    public void onClick(View v) {
      // do something when the button is clicked
    }
    ...
}
```

Chú ý rằng lời gọi **onClick()** trong ví dụ trên không trả về giá trị, nhưng các phương thức của bộ nghe sự kiện khác phải trả lại một biến kiểu boolean. Lý do phụ thuộc vào sự kiện này. Đây là một vài lý do:

- **onLongClick()** - Trả về một giá trị kiểu boolean để cho biết ta đã dùng sự kiện này và nó không cần thực hiện "long click") thêm nữa. Trả về giá trị TRUE để chỉ ra rằng ta đã xử lý sự kiện này và nó nên

dừng lại ở đây; trả về FALSE nếu ta không xử lý nó và / hoặc sự kiện nên chuyển tới bất kỳ bộ nghe sự kiện on-click nào khác.

- **onKey()** - Trả về một giá trị kiểu boolean để cho biết ta đã dùng sự kiện này và nó không cần được thực hiện thêm. Trả về giá trị TRUE để chỉ ra rằng ta đã xử lý sự kiện này và nó nên dừng lại ở đây; trả về FALSE nếu ta không xử lý nó và / hoặc sự kiện nên chuyển tới bất kỳ bộ nghe sự kiện on-key nào khác.
- **onTouch()** - Trả về một giá trị kiểu boolean để cho biết: liệu bộ nghe của ta đã dùng sự kiện này hay chưa. Điều quan trọng là sự kiện này có thể có nhiều hành động nối tiếp nhau. Vì vậy, nếu trả về FALSE, ta biết rằng ta đã không sử dụng và cũng không quan tâm đến hành động tiếp theo từ sự kiện này. Như vậy, ta không được gọi tới bất kỳ thao tác nào khác bên trong sự kiện này.
- **Event Handlers** – Là phương thức xử lý khi phát sinh sự kiện. (Xử lý sự kiện)
- Nếu ta đang xây dựng một thành phần tùy chỉnh từ View, ta sẽ phải định nghĩa một số phương thức sử dụng như của xử lý sự kiện mặc định. Trong tài liệu về Custom Components, ta sẽ tìm hiểu một số callbacks thường được sử dụng để xử lý sự kiện, bao gồm:
- **onKeyDown(int, KeyEvent)** – Được gọi khi một sự kiện nhấn phìm mới xảy ra.
- **onKeyUp(int, KeyEvent)** – Được gọi khi mọt sự kiện thả phím xảy ra.
- **onTrackballEvent(MotionEvent)** – Được gọi khi một sự kiện chuyển động trackball xảy ra.
- **onTouchEvent(MotionEvent)** – Được gọi khi một sự kiện chuyển động màn hình cảm ứng xảy ra.
- **onFocusChanged(boolean, int, Rect)** – Được gọi khi view được chọn (focus) hoặc bỏ chọn.
- Có một số phương thức khác mà ta nên biết, chúng không phải là một phần của lớp View, nhưng có thể trực tiếp tác động đến cách bạn có thể xử lý các sự kiện. Vì vậy, khi quản lý sự kiện phức tạp hơn bên trong một layout, ta nên xem xét các phương pháp sau:
- **Activity.dispatchTouchEvent(MotionEvent)** – Điều này cho phép Activity bắt tất cả các sự kiện chạm màn hình trước khi chúng được gửi đến cửa sổ.

CHƯƠNG 3 : SQLITE

3.1 Giới thiệu

SQLite là 1 hệ quản trị cơ sở dữ liệu thu nhỏ, không có server, thao tác trên file và có thể sử dụng trong hầu hết các hệ điều hành. SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C. Phiên bản mới nhất là 3.8.6 (15/8/2014)

3.1.1 Nhận định chung về SQLite

a. Ưu điểm

- Tin cậy: các transaction trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng.
- Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ)
- Không cần cài đặt, cấu hình
- Kích thước chương trình gọn nhẹ (khoảng 300 kB)
- Thực hiện các thao tác đơn giản nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ khác.
- Không cần phần mềm phụ trợ
- Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng.
- Trong Android, chúng ta không cần cài đặt nhiều, chỉ cần cung cấp các hàm để thao tác và chương trình sẽ quản lí phần còn lại.

b. Một số hạn chế

- Không hỗ trợ một số tính năng của SQLite 92

Tính năng	Diễn giải
RIGHT OUTER JOIN	Không hỗ trợ. Chỉ dùng được đối với LEFT OUTER JOIN
FULL OUTER JOIN	Không hỗ trợ. Chỉ dùng được đối với LEFT OUTER JOIN

ALTER TABLE	Chỉ hỗ trợ các biến thể về RENAME TABLE và ADD COLUMN của lệnh ALTER TABLE. Không hỗ trợ về DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT
Trigger support	Chỉ hỗ trợ phát biểu FOR EACH ROW nhưng hỗ trợ phát biểu FOR EACH STATEMENT
VIEWs	VIEWs trong SQLite có dạng là read-only. Bạn không thể thực thi các phát biểu như DELETE, INSERT hoặc UPDATE trong view

- Cơ sở dữ liệu do SQLite tạo ra sẽ private, tức là chỉ sử dụng cho bản thân ứng dụng.
- 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 JOINT.
- 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

c . SQLite phù hợp trong các tình huống sau

- Ứng dụng sử dụng dạng flat file để lưu trữ dữ liệu: như từ điển, các ứng dụng nhỏ, lưu cấu hình ứng dụng... Nó mạnh mẽ hơn nhiều kỹ thuật lưu trữ file thông thường nhờ kỹ thuật hiện, dễ dùng và tin cậy hơn.
- Sử dụng trong các thiết bị nhúng: smart phone, PDA và các thiết bị di động.
- Các website có khoảng 100.000 lượt truy cập/ngày (về mặt lý thuyết thì SQLite có thể đáp ứng cao hơn nhiều).
- Sử dụng làm database tạm để lưu trữ dữ liệu lấy về từ các database trên SQL server, Oracle...
- Làm database demo cho các ứng dụng lớn, dùng để làm mô hình khái niệm cho ứng dụng.

- Sử dụng trong giảng dạy cho những người mới làm quen với ngôn ngữ truy vấn SQL. Ngoài các trường hợp trên, tốt hơn hết là bạn sử dụng SQL Server, Oracle, DB2, …

3.1.2 Phân loại các lệnh tương tác với CSDL trong SQLite

DDL - Data Definition Language

Lệnh	**Diễn giải**
CREATE	Tạo mới table, view, hoặc các đối tượng khác trong cơ sở dữ liệu (CSDL)
ALTER	Hiệu chỉnh các đối tượng đã tồn tại trong CSDL (như table)
DROP	Xóa table, view hoặc các đối tượng khác đã tồn tại trong (CSDL)

DML- Data Manipulation Language

Lệnh	**Diễn giải**
INSERT	Thêm mới dòng (record) vào table
UPDATE	Hiệu chỉnh dữ liệu của các records
DELETE	Xóa records

DQL - Data Query Language

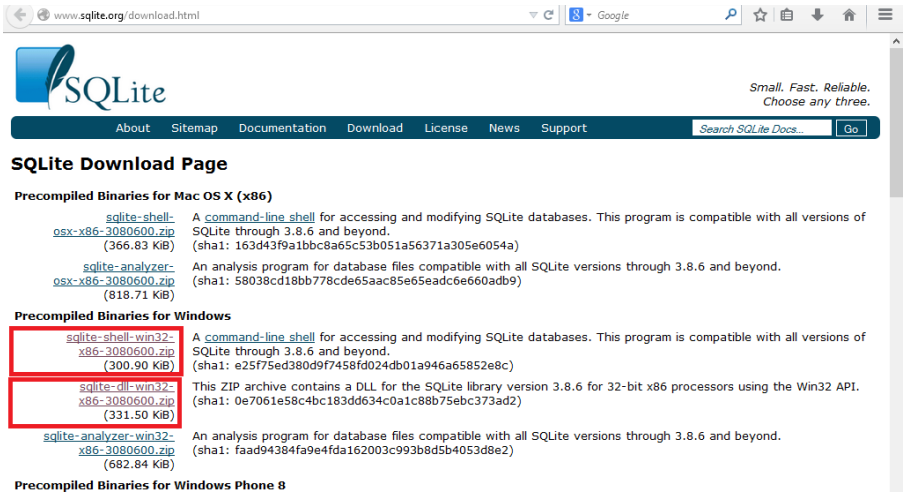
Lệnh	**Diễn giải**
SELECT	Lấy dữ liệu từ một hoặc nhiều table

3.2 Cài đặt SQLite trên Windows

3.2.1 Cài đặt SQLite

B1 : Truy cập trang *http://www.sqlite.org/download.html*

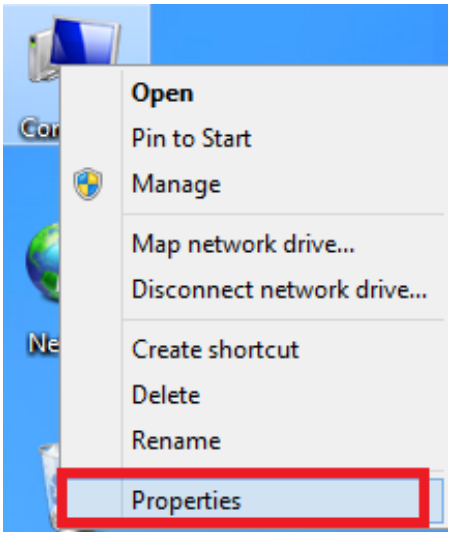
B2 : Tìm và download 2 file nén: **sqlite-shell-win32-*.zip** và **sqlite-dll-win32-*.zip**



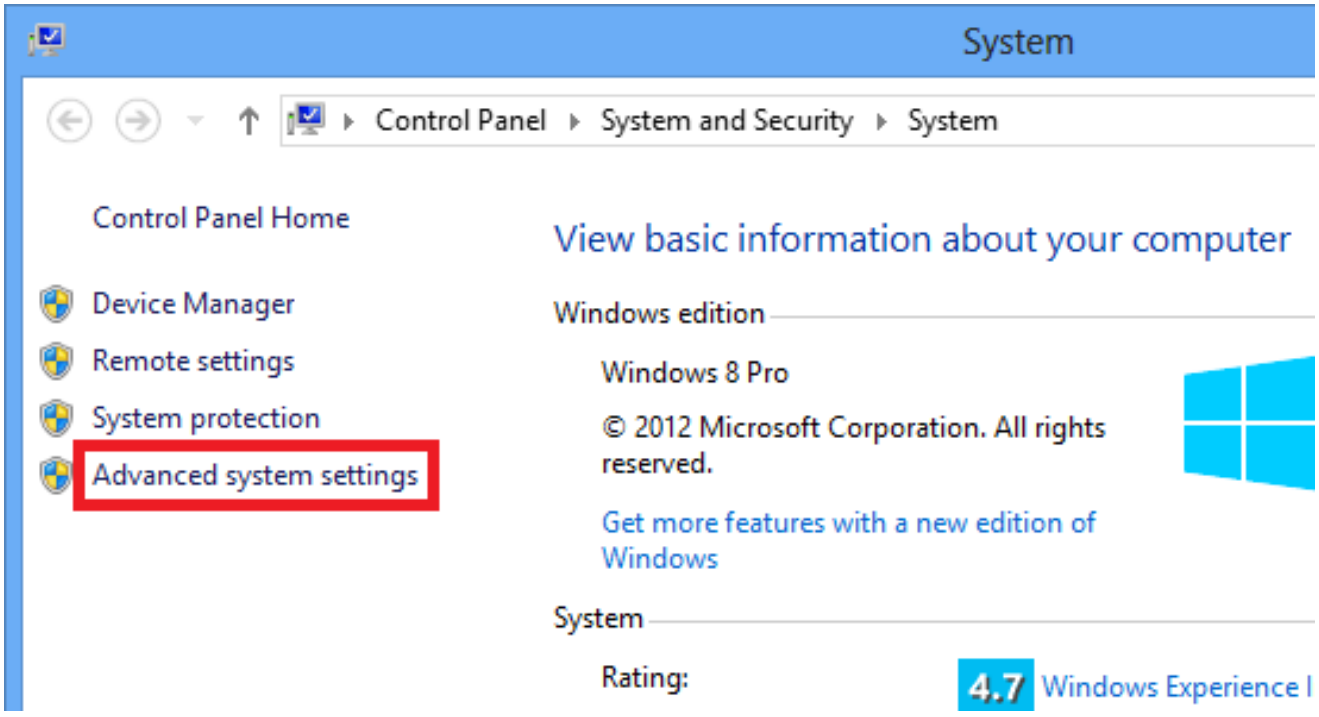
B3 : Giải nén 2 file vừa download được: Để dễ quản lý (không bắt buộc), bạn nên tạo folder với tên sqlite trong cùng folder với eclipse và SDK, rồi giải nén 2 file vào đó. Kết quả sẽ thu được 3 file sqlite3.def, sqlite3.dll và sqlite3.exe. Copy đường dẫn để sử dụng cho bước kế tiếp

B4: Thêm đường dẫn vào biến môi trường PATH của Windows:

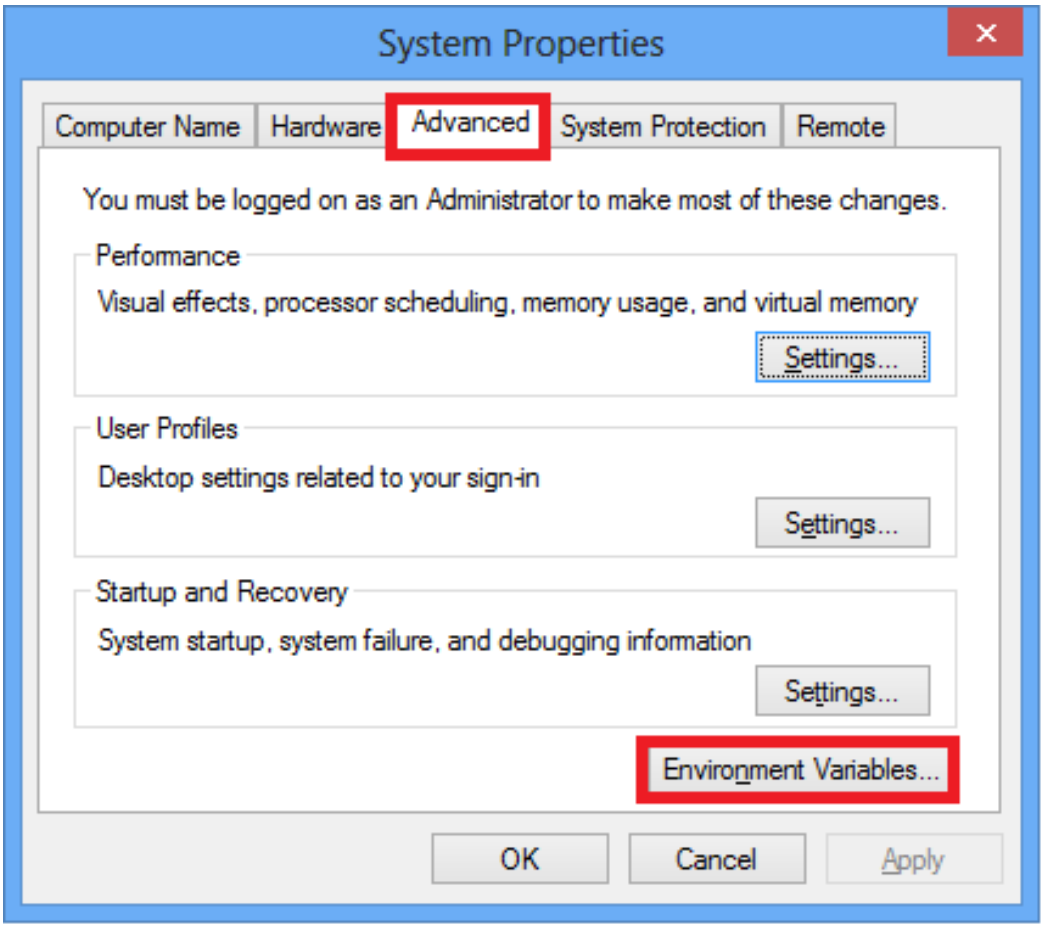
- **B4.1:** Right Click vào icon của Computer trên desktop, chọn Properties.



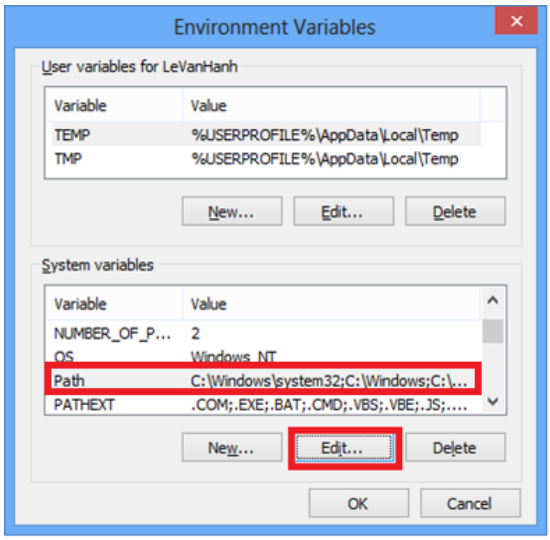
- **B4.2: Trong hộp thoại System, chọn Advanced system settings.**

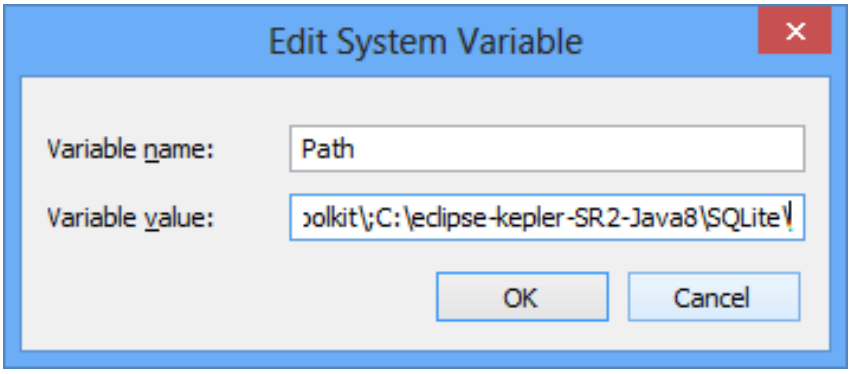


- **B4.3:** Trong hộp thoại System Properties, chọn tab Advanced, chọn button EnvironmentVariables … để mở tiếp hộp thoại Environment Variables.



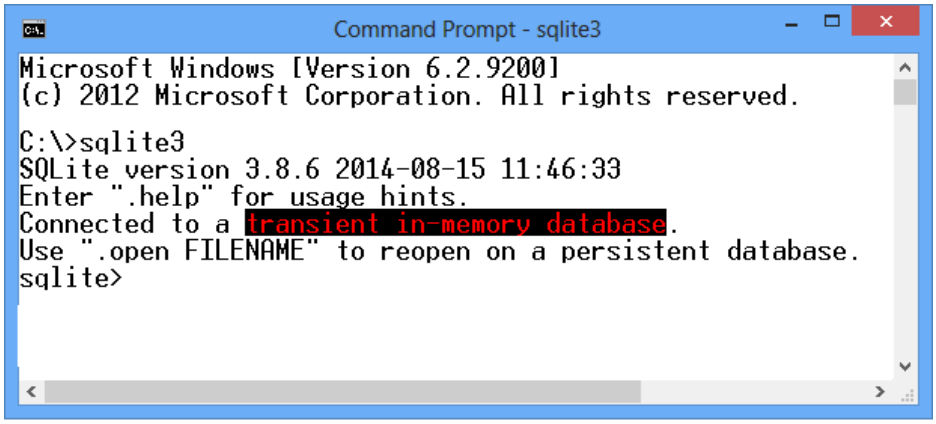
- **B4.4:** Trong hộp thoại Environment Variables, trong vùng System variables, tìm và chọn biến Path, xong click button Edit để mở hộp thoại Edit Environment Variables.





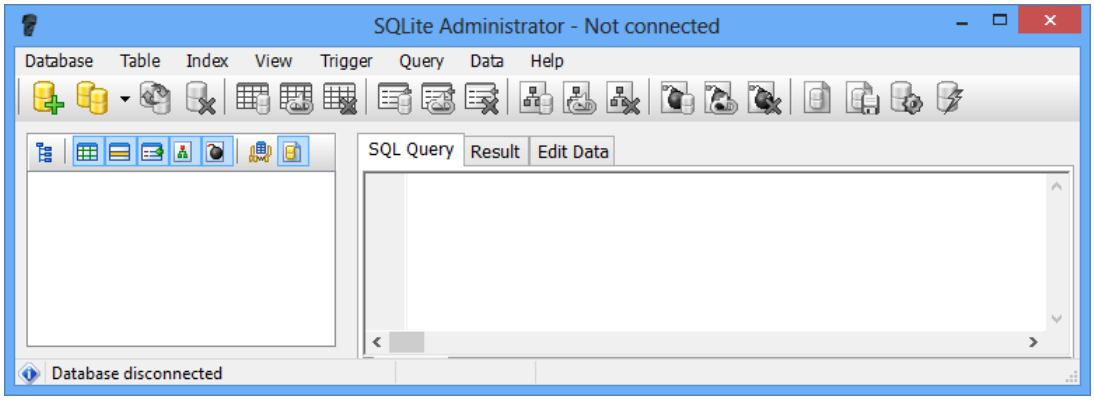
- **B4.5:** Click mouse vào textbox. Di chuyển về cuối, bổ sung dấu chấm phẩy (;) rồi paste đường dẫn đã copy ở B3 vào đây.
- **B4.6**: Lần lượt click chọn OK trong từng hộp thoại để hoàn tất việc bổ sung biến môi trường

B5: Kiểm tra kết quả thực hiện: mở cửa sổ cmd của Windows, gõ lệnh sqlite3 sẽ xuất hiện kết quả như hình minh họa



3.2.2 Sử dụng SQLite Administrator

Có thể sử dụng lệnh SQLite trong cửa sổ command của Windows. Tuy nhiên, bạn nên dùng công cụ SQLite Administrator (download http://download.orbmu2k.de/files/sqliteadmin.zip). Sau khi download hoàn tất, giải nén vào 1 folder nào đó. Tìm và chạy (double click) file sqliteadmin.exe để thấy cửa sổ sau:



Tuy rất tiện ích trong việc quản lý CSDL, nhưng trong tài liệu này không hướng dẫn cách sử dụng ứng dụng SQLite Administrator. Bạn có thể tự tìm hiểu vì cách sử dụng ứng dụng này tương đối giống như trong SQL Server của Microsoft.

3.3 Kiểu dữ liệu trong SQLite

Mỗi cột, biến và biểu thức có liên quan đến dữ liệu trong SQLite đều phải có thuộc tính để xác định kiểu dữ liệu

3.3.1 Các Classes lưu trữ kiểu dữ liệu

Mỗi giá trị được lưu trữ trong CSDL của SQLite đều phải thuộc 1 trong những class lưu trữ sau:

Class lưu trữ	**Diễn giải**
NULL	Chứa giá trị NULL
INTEGER	Là giá trị nguyên, có dấu được lưu trữ trong 1, 2, 3, 4, 6, hoặc 8 bytes tùy thuộc vào độ lớn của dữ liệu
REAL	Là giá trị số thực, được lưu trữ trong 8-byte
TEXT	Là chuỗi ký tự, lưu trữ bằng cách mã hóa CSDL (UTF-8, UTF-16BE hoặc UTF16LE)
BLOB	Giá trị là 1 cả 1 khối (blob) của dữ liệu, sẽ được lưu đầy đủ theo cả khối

Tên của các kiểu dữ liệu trong SQLite :

Class lưu trữ	**Các kiểu dữ liệu**
INTEGER	INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT,INTEGER

	UNSIGNED BIG INT, INT2, INT8
REAL	REAL, DOUBLE, DOUBLE PRECISION, FLOAT
TEXT	CHARACTER(20), VARCHAR(255), VARYING CHARACTER(255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100), TEXT CLOB
BLOB	(không chỉ ra kiểu dữ liệu cụ thể)
NUMERIC	NUMERIC, DECIMAL(10,5), BOOLEAN, DATE, DATETIME

3.3.2 Kiểu lý luận(Boolean Datatype)

SQLite không có class lưu trữ riêng cho kiểu dữ liệu luận lý. Thay vào đó, giá trị luận lý được lưu trữ dưới dạng số nguyên với 0 là *false* và 1 là *true*.

3.3.3 Kiểu ngày và giờ (Date and Time Datatype)

SQLite cũng không có class lưu trữ riêng cho kiểu dữ liệu ngày/giờ, nhưng SQLite có khảnăng lưu trữ ngày/giờ như là TEXT, REAL hoặc INTEGER.

Class lưu trữ	**Định dạng**
TEXT	Ngày được định dạng theo dạng thức "YYYY-MM-DD HH:MM:SS.SSS"
REAL	Số ngày tính từ 24 November 24 năm 4714 trước Công nguyên
INTEGER	Số lượng thời gian tính từ 1970-01-01 00:00:00 UTC

Bạn có thể chọn kiểu ngày/giờ bất kỳ trong các định dạng trên. Khi sử dụng SQLite sẽ cung cấp 1 số hàm giúp chuyển đổi giữa các định dạng.

3.4 Lệnh SQLite

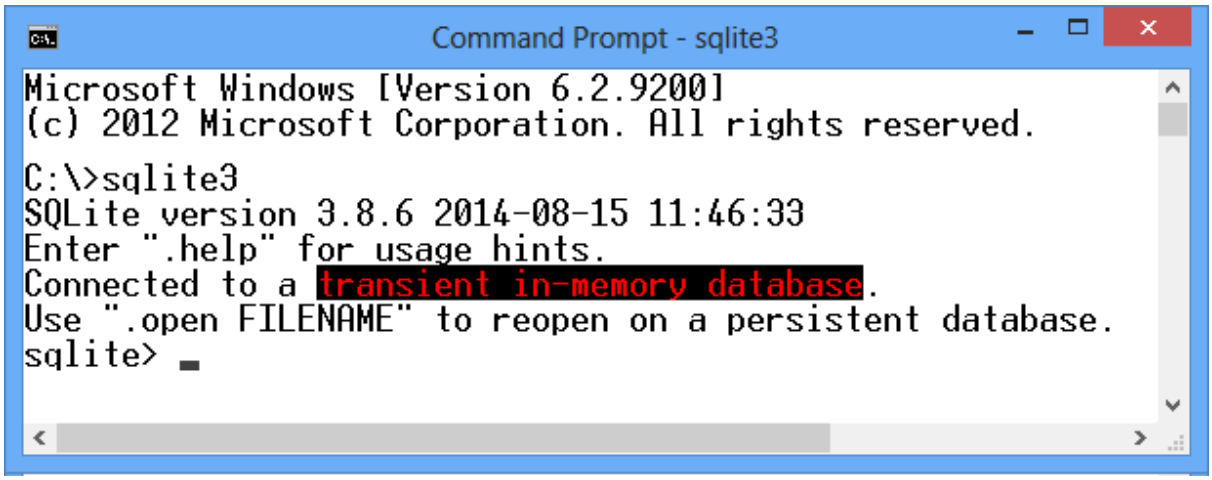
3.4.1 Một số lưu ý

- *Case Sensitivity :* SQLite có phân biệt chữ hoa/thường(case insensitive). Mặt khác, SQLite có trường hợp cùng 1 tên nhưng khi viết hoa và khi viết thường là 2 lệnh có ý nghĩa khác nhau như lệnh GLOB và lệnh glob.
- *Sử dụng ghi chú trong SQLite :*
  - Giúp giải thích hoặc ghi chú 1 vấn đề nào đó trong mã lệnh.
  - Nội dung ghi chú có thể chứa khoảng trắng hoặc các kí tự khác nhau nhưng ko thể đặt các ghi chú lồng vào nhau.
  - Sử dụng :
    - Ghi chú theo dòng : trong SQLite bắt đầu bằng hai liên tiếp "-" kí tự(ASCII 0X2D).
    - Ghi chú theo đoạn (Theo phong cách ngôn ngữ C): các ghi chú được đặt trong cặp kí hiệu /* và */ .
- *SQLite Statements (Phát Biểu):*
  - Tất cả các phát biểu trong SQLite đều bắt đầu với 1 từ khóa như SELECT,INSERT,CREATE,UPDATE,DELETE,ALTER,DROP,... và đều phải kết thúc bằng dấu (,).
  - Mỗi phát biểu có thể được trình bày trên nhiều dòng.

3.4.2 Các lệnh SQLite thường dùng :

Những lệnh này được gọi là dấu chấm lệnh SQLite và ngoại lệ với các lệnh này là chúng không được kết thúc bằng 1 dấu (;) .

Để tiện theo dõi các bạn hãy khởi động cmd trong Windows. Sau đó gõ lệnh sqlite3 tại dấu nhắc để có kết quả :



Các lệnh thường dùng :

Lệnh	Diễn giải
.backup ?DB? FILE	Backup DB (default “main”) thành FILE
.bail ON|OFF	Dừng lại sau khi chạm một lỗi. Mặc định là OFF
.databases	Liệt kê tên và các file CSDL đính kèm (attached)
.dump ?TABLE?	Dump the database in an SQL text format. If TABLE specified, only dump tables matching LIKE pattern TABLE.Đổ cơ sở dữ liện sang đinh dạng SQL. Nếu TABLE được chỉ ra, lệnh chỉ thực hiện với bảng đó
.echo ON|OFF	Bật | tắt lệnh echo
.exit	Thoát khỏi dấu nhắc lệnh của SQLite
.explain ON|OFF	Bật | tắt EXPLAIN mode. Nếu không có đối số, mặc định sẽ là ON
.header(s) ON|OFF	Bật | tắt việc hiển thị tên tiêu đề cột của table
.help	Liệt kê các lệnh trong SQLite
.import FILE TABLE	Import dữ liệu từ FILE vào TABLE
.indices ?TABLE?	Hiển thị tên của tất cả các chỉ mục. Nếu tên TABLE được chỉ ra, chỉ liệt kê chỉ mục có trong TABLE
.load FILE ?ENTRY?	Nạp thư viện mở rộng
.log FILE|off	Bật tắt login
.mode MODE	Thiết lập output mode, trong đó MODE thuộc một trong giá trị sau: - **csv** các giá trị ngăn cách nhau bởi dấu phẩy - **column** canh lề trái cho các cột - **html** HTML<table>code - **insert** SQLite insert statements for TABLE - **line** One value per line - **list** Values delimited by .separator string

	- **tabs** Tab-separated values - **tcl** TCL list elements
.nullvalue STRING	Print STRING in place of NULL values
.output FILENAME	Send output to FILENAME
.output stdout	Send output to the screen
.print STRING...	Print literal STRING
.prompt MAIN CONTINUE	Thay thế dấu nhắc lệnh của SQLite
.quit	Thoát khỏi dấu nhắc lệnh của SQLite
.read FILENAME	Excute SQL in FILENAME
.schema ?TABLE?	Hiển thị nội dung của phát biểu CREATE. Nếu tên của TABLE được chỉ ra, sẽ hiển thị phát biểu CREATE của TABLE có dạng tương tự như TABLE có tên được chỉ ra
.separator STRING	Thay đổi dấu ngăn cách sẽ sử dụng cho output mode và ***.import***
.show	Hiển thị giá trị hiện tại của các thiết lập
.stats ON | OFF	Bật / tắt thiết lập stats
.table ?PATTERN	Liệt kê các table có dạng tương tự như mẫu PATTERN
.timeout MS	Try opening locked tables for MS milliseconds
.width NUM NUM	Thiết lập độ rộng cột cho mode “column”
.timer ON | OFF	Bật / tắt thiết lập timer của CPU

Sử dụng lệnh ***.show*** để xem các thiết lập mặc định :

```
sqlite>.show
```

```
echo: off
explain: off
headers: off
mode: column
nullvalue: ""
output: stdout
separator: "|"
width:
sqlite>
```

Không được có khoảng trắng giữa các ngắt lệnh ***sqlite>*** và lệnh cần thực hiện

3.4.3 Định dạng dữ liệu xuất :

Có thể sử dụng các lệnh của SQLite để định dạng dữ liệu cận xuất. Ví dụ :

```
sqlite>.header on
sqlite>.mode column
sqlite>.timer on
sqlite>
```

Với định dạng vừa có, kết quả hiển thị sẽ có dạng như sau :

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	Califonia	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	20000.0
4	Mark	25	Rich-Mond	65000.0
5	David	27	Texas	85000.0
6	Jim	22	South-Hall	45000.0
7	Jimmy	24	Houston	10000.0
CPU Time: user 0.000000 sys 0.000000				

3.4.4 Table sqlite_master

Table chính lưu giữ tất cả các thông tin về CSDL của bạn tên là sqlite_master.Bạn có thể xem các thành phần của table này bằng lệnh .schema như sau:

```
sqlite>.schema sqlite  master
```

Kết quả hiển thị có dạng:

```
CREATE TABLE sqlite_master (       type text,
name text,
tbl_name text,
rootpage integer,
sql text
);
```

3.5 Toán tử trong SQLite

3.5.1 Toán tử số học (Arithmetic Operators)

Toán tử	**Diễn giải**
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Lấy phần dư trong phép chia giữa 2 số nguyên

3.5.2 Toán tử so sánh (Comparison Operators)

Toán tử	**Diễn giải**
== or =	So sánh bằng
!= or <>	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hay bằng
<=	So sánh nhỏ hơn hay bằng
!<	So sánh nếu không nhỏ hơn
!>	So sánh nếu không lớn hơn

3.5.3 Toán tử luận lý (Logical Operators)

Toán tử	**Diễn giải**
AND	Và
BETWEEN *min* AND	Chọn nhừng giá trị nằm trong khoảng từ min đến max

max	
EXISTS	Chọn những dòng có giá trị hiện diện trong một table được chỉ ra với các tiêu chí nhất định
IN	Chọn những dòng có giá trị xuất hiện trong danh sách được liên kết sau với toán tử IN
NOT IN	Chọn những dòng không có giá trị xuất hiện trong danh sách được liên kết sau với toán tử IN
LIKE	So sánh giá trị theo một dạng thức cho trước. Like không phân biệt chữ hoa/thường
GLOB	So sánh giá trị theo một dạng thức cho trước. Glob có phân biệt chữ hoa/thường
NOT	Là phép phủ định. Thường dùng đi liền trước các toán tử luận lý khác nhau như : not exists, not between, not in, ...
OR	Hoặc
IS NULL	So sánh 1 giá trị với giá trị NULL
IS	Thực hiện tương tự như toán tử bằng (=)
IS NOT	Thực hiện tương tự như toán tử khác (!=)
||	Tạo một chuỗi mới bằng cách cộng hai chuỗi tham gia vào toán tử
UNIQUE	Tìm nhưng dòng chỉ xuất hiện 1 lần duy nhất trong table

3.5.4 Toán tử trên Bit (Bitwise Operators)

Toán tử hoạt đônhj trên Bit và thực hiện so sánh từng cặp bit với nhau :

P	Q	p&q	p|q
0	0	0	0
0	1	0	1
1	1	1	1
1	0	0	1

SQLite hỗ trợ các toán tử bit như sau :

Toán tử	**Diễn giải**
&	Và giữa 2 bit giá trị

|	Hoặc giữa 2 bit giá trị
~	Phủ định các bit có trong giá trị đi sau toán tử
<<	Dịch trái các bit theo số lần được chỉ ra ngay sau toán tử. Mỗi lần dịch trái tương đương với việc nhân đôi giá trị của số đặt bên trái của toán tử
>>	Dịch phải các bit theo số lần được chỉ ra ngay sau toán tử. Mỗi lần dịch phải tương đương với việc chia lấy phần nguyên của số đặt bên trái toán tử

3.6 Biểu thức trong SQLite

Một biểu thức là sự kết hợp của một hoặc nhiều giá trị, toán tử hoặc các hàm trong SQLite để đánh giá 1 giá trị là đúng hay sa

3.6.1 Cú pháp :

```
SELECT column1, column2, columnN
FROM table_name
WHERE [CONTION | EXPRESSION] ;
```

3.6.2 Phân loại biểu thức :

- ***Biểu thức luận lý (Boolean expressions)***

Tìm những dòng dữ liệu phù hợp với điều kiện đưa ra. Cú pháp có dạng:

```
SELECT column1, column2, columnN
FROM table_name
WHERE SINGLE VALUE MATCHING
 EXPRESSION
```

- ***Biểu thức số học***

Là biểu thức thực hiện bất kỳ những phép tính toán có trong vị trí bất kỳ nào của câu truy vấn:

```
SELECT numerical_expression as
 OPERATION_NAME
[FROM table_name WHERE CONDITION];
```

- ***Biểu thức ngày***

Là những biểu thức trả về ngày/giờ của hệ thống và kết quả đó có thể tham gia vào các thao tác về dữ liệu khác:

Ví dụ:

```
sqlite> SELECT CURRENT_TIMESTAMP;
CURRENT_TIMESTAMP = 2017-10-5  03:37:36
```

3.7 Các lệnh liên quan đến CSDL

3.7.1 Tạo mới CSDL:

Mặc định CSDL được tạo ra sẽ lưu:

DATA/data/APP_NAME/databases/DATABASE_NAME

Trong đó :

- ***DATA:*** *là đường dẫn Enviroiment.getDataDirectory()*
- ***APP_NAME :*** *tên của ứng dụng*
- ***DATABASE_NAME :*** *tên của CSDLđược tạo ra*

a.Sử dụng tại dấu nhắc lệnh của SQLite:

Người dùng không cần có bất kỳ quyền đặc biệt để tạo ra một CSDL.

Cú pháp :

```
sqlite3  DatabaseName . db
```

Tên cơ sở dữ liệu luôn pahir là duy nhất trong RDBMS (không được trùng tên giữa cac CSDL)

Lệnh .database : giúp liệt kê các CSDL hiện có

```
sqlite>.database
seq   name            file
----  --------------- ------------------------
0     main            C:\sqlite\testDB.db
```

Lệnh .dump : giúp chuyển đổi 1 CSDL hoàn chỉnh trong 1 file dạng văn bản. Lệnh phải thực hiện trên dấu nhắc lệnh của hệ điều hành chứ không thể thực hiện trong dấu nhắc lệnh của sqlite

```
C:\sqlite>sqlite3  testDB.db  .dump  >  testDB.sql
```

Lệnh trên sẽ chuyển đổi toàn bộ nội dung của CSDL SQLite testDB.db vào file văn bản ASCII testDB.sql . Nhờ vậy bạn có thể làm phục hồi CSDL testDB.db từ file testDB.sql đã được tạo ra như sau:

```
C:\sqlite>sqlite3  testDB.db  <  testDB.sql
```

b. Sử dụng bằng mã lệnh:

Để tạo ra một CSDL, bạn chỉ cần gọi phương thức *openOrCreateDatabase* này với tham số là tên CSDL của bạn và mode tạo lập. Phương thức này trả về một thể hiện của CSDL SQLite.

Cú pháp sử dụng như sau :

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",
                                                 MODE_PRIVATE, null);
```

Ngoài ra, có các chức năng khác trong gói database cùng làm công việc này gồm:

openDatabase (String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)

Phương thức này chỉ mở ra CSDL hiện có với chế độ cờ thích hợp. Chế độ cờ phổ biến có thể là OPEN_READWRITE OPEN_READONLY

openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)

Phương thức này không chỉ mở mà còn có khả năng tạo ra CSDL nếu nó không tồn tại.

openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)

Cũng tương tự như phương thức trên nhưng thay vì dùng đối số có kiểu là String, phương thức này sử dụng File. Có thể dùng *file.getPath()* thay cho đối số File.

3.7.2 Attach Database

Gắn kèm 1 CSDL vào SQLite

Cú Pháp

```
ATTACH DATABASE 'DatabaseName' As 'Alias-name';
```

Lệnh trên gắn CSDL *DatabaseName* vào SQLite với tên là Alias-name

Nếu không tìm thấy CSDL *DatabaseName*, SQLite sẽ tạo ra 1 CSDL mới

3.7.3 Detach Database

- Sử dụng để tách 1 CSDL đã được gắn kèm trước đó bằng lệnh *ATTACH DATABASE* .
- Nếu cùng 1 CSDL nhưng được gắn kèm với nhiều bí danh, lệnh DETACH sẽ thực hiện ngắt kết nối với tên bí danh được chỉ ra, các bí danh con lại vẫn được sử dụng bình thường.
- Không thê tách các CDSL main hoặc TEMP.
- Nếu thực hiện DETACH với CSDL được lưu trong bộ nhớ hoặc CSDL tạm, CSDL sẽ bị phá hủy và các nội dung sẽ bị mất.

- Cú Pháp:

```
DETACH DATABASE 'Alias-name';
```

Trong đó ***Alias-Name*** là tên bí danh mà bạn sử dụng khi gắn CSDL bằng phát biểu ATTACH.

3.8 Các lệnh liên quan đến cấu trúc của TABLE

3.8.1 Create Table

Công Dụng:

Tạo mới 1 Table với tên và kiểu dữ liệu của các cột được chỉ ra.

Cú Pháp :

```
CREATE TABLE database_name.table_name( column1 datatype
                                       PRIMARY KEY(one or more columns),
                                       column2 datatype,
                                       column3 datatype,
                                       .....
                                       columnN datatype, );
```

Trong đó *database_name* được dùng khi tồn tại nhiều CSDL.

AutoIncrement :

AUTOINCREMENT là từ khóa được sử dụng để tự động tăng 1 giá trị của một thuộc tính (field) trong table. AUTOINCREMENT chỉ được dùng với thuộc tính có kiểu là số nguyên và thường sử dụng kèm với lệnh tạo lập table.

Cú Pháp:

```
CREATE TABLE table_name(  column1 INTEGER AUTOINCREMENT,
                          column2 datatype,
                          .....
                          columnN datatype );
```

Các lệnh khác liên quan đến Table:

.tables : Xem danh sách các table đang có

```
sqlite>.table
COMPANY     DEPARTMENT
```

.schema : xem lại nội dung lệnh đã tạo ra cấu trúc của table đang có

```
sqlite>.schema COMPANY
CREATE TABLE COMPANY( ID INT PRIMARY KEY NOT NULL,
                      NAME      TEXT          NOT NULL,
                      AGE       INT           NOT NULL,
                      ADDRESS CHAR(50),
                      SALARY REAL);
```

3.8.2 Drop Table

Xóa table cùng tất cả các kết hợp đã có (data, indexes,triggers, constraints và quyền hạn được cấp trên table). Vì vậy bạn cần cân nhắc trước khi xóa 1 table.

Không thể sử dụng DROP TABLE để xóa các rằng buộc (constraints).

Cú Pháp:

```
DROP TABLE  database_name . table_name;
```

3.8.3 Alter Table

Dùng để đổi tên table, thêm mới cột vào table hoặc có thể dùng để thay đổi tên cột.

Không thể sử dụng ALTER TABLE để đổi tên các rằng buộc (constraints)

Cú Pháp:

Thêm cột vào table:

```
ALTER TABLE table_name ADD COLUMN column_def.....;
```

Đổi tên cột :

```
ALTER TABLE table_name RENAME TO new_table_name;
```

3.8.4 Rằng buộc kiểu dữ liệu trong SQLite(Constraints)

Constraints là các quy tắc thực thi trên dữ liệu của table, giúp đảm bảo tính chính xác và độ tin cậy của dữ liệu trong CSDL.

- PRIMARYKEY Constraints:
  - Rằng buộc PRIMARYKEY xác định duy nhất mỗi record trong 1 table của CSDL
  - Khi cài đặt, một table bắt buộc phải có duy nhất 1 khóa chính.
  - Khóa chính có thể là sự kết hợp của 1 hay nhiều cột.
  - Trong SQLite khóa chính có thể được chứa giá trị null

- NOT NULL Constraints:

  Theo mặc định, 1 cột có thể chứa giá trị null. Để rằng buộc 1 cột không được phép sử dụng giá trị null,cần sự dụng rằng buộc loại này cho cột đó. Null ko được xem là không có dữ liệu, thay vào đó nó diễn tả dữ liệu chưa được biết.

- DEFAULT Constraints:

  Rằng buộc DEFAULT cung cấp giá trị mặc định cho cột khi lệnh INSERT INTO không cung cấp giá trị cho cột đó.

- UNIQUE Constraints:
  - Dùng để ngăn cản 2 record giá trị giống hệt nhau trong 1 cột cụ thể.
  - Có thể xuất hiện nhiều rằng buộc UNIQUE trên cùng 1 table(mỗi rằng buộc được gắn cho 1 cột).
  - UNIQUE được chứa giá trị NULL.
- CHECK Constraints:

  Ràng buộc CHECK cho phép kiểm tra giá trị được nhập vào record phải tuân thủ theo 1 điều kiện nào đó. Nếu dữ liệu nhập vào vi phạm ràng buộc CHECK , record đó sẽ không được nhập vào table.

3.9 Lệnh Insert Into

- Thêm 1 dòng dữ liệu mới vào table
- Cú pháp:

- Dạng 1:

```
INSERT INTO TABLE NAME ( column1,column2,column3....columnN)
VALUES   ( value1,value2,value3,.....valueN);
```

Trong đó column1,column2,column3,...columnN là tên các cột

- Dạng 2 :

```
INSERT INTO TABLE NAME ( column1,column2,column3....columnN)
```

Sử dụng khi thứ tự dữ liệu đưa vào đúng với thứ tự khi tạo lập table

- Insert với nguồn dữ liệu được lấy từ table khác

Cú pháp:

```
INSERT INTO   destination_table_name [( column1,column2,.....
                                                    columnN)]
   SELECT  column1,column2,.............columnN
   FROM   source_table_name
      [ WHERE   condition ];
```

3.10 Lệnh truy vấn dữ liệu

Lệnh SELECT được sử dụng để lấy dữ liệu từ 1 table trong SQLite và trả về dứ liệu dưới dạng bảng kết quả. Các bảng kết quả còn được gọi là tập kết quả

3.10.1 Cú pháp đơn giản

```
SELECT  [DISTINCT]  column1,column2,column3,.........columnN
```

Trong đó column1,column2,column3,...columnN là tên các cột cần lấy và những tên cột này phải có trong table. Khi bạn cần lấy tất cả các cột, sử dụng cú pháp sau:

```
SELECT   *   FROM   table_name;
```

3.10.2 Mệnh đề LIMIT

- Dùng để giới hạn số lượng dòng được trả về trong tập kết quả
- OFFSET ***n*** :nếu có sử dụng sẽ lấy từ record có số thứ tự ***n***(dòng đầu tính từ 0)
- Cú pháp:

```
SELECT  column1,column2,column3,.........columnN
FROM   table_name
LIMIT   [no  of  row ]   OFFSET  [ row  num ]
```

3.10.3 Mệnh đề FROM

Mệnh đề FROM giúp chỉ ra tên các table cần lấy dữ liệu phục vụ cho câu truy vấn

Khi có nhiều table cùng tham gia trong câu truy vấn, các table này cần thực hiện phép kết (JOIN). Trong SQLite hỗ trợ các phép sau :

- CROSS JOIN

  - CROSS JOIN thực hiện kết trên mọi record của table thứ nhất với tất cả các record của bảng thứ hai
  - Bảng kết quả sẽ có:
    - Số lượng cột = tổng số lượng cột của 2 table thành phần

- Số lượng dòng = tích giữa số lượng các dòng của 2 table thành phần

- Do CROSS JOIN có khả năng tạo ra các bảng rất lớn, vì vậy chỉ nên sử dụng khi cần thiết

- Cú pháp:

```
SELECT ...  FROM  table1  CROSS JOIN  table2  ....
```

- INNER JOIN
  - INNER JOIN tạo ra một bảng kết quả mới bằng các dò tìm và kết hợp các giá trị cột của 2 bảng(table1 và table2) dựa trên điều kiện kết hợp
  - INNER JOIN được đùng khá phổ biến nhờ không gây ra hiện tượng "bùng nổ tổ hợp" như các loại JOIN khác.
  - Cú pháp:

```
SELECT .... FROM  table1  [INNER]  JOIN  table2  ON
                                    conditional_expression....
```

- OUTER JOIN
  - OUTER JOIN là một phần mở rộng của INNER JOIN.Mặc dù tiêu chuẩn SQL định nghĩa 3 loại OUTER JOIN là : LEFT, RIGHT và FULL nhưng SQLite chỉ hỗ trợ LEFT OUTER JOIN
  - Tương tự như INNER JOIN, OUTER JOIN sử dụng từ khoá ON để chỉ ra điều kiện kết.
  - Cú pháp:

```
SELECT  .... FROM   table1   LEFT OUTER JOIN  table2  ON
                                    conditional_expression .....
```

3.10.4 Mệnh đề WHERE

Mệnh đề WHERE được dùng đẻ lọc hoặc lấy các record cần thiết theo một điều kiện nào đó từ dữ liệu có trong một hoặc nhiều bảng. Nếu điều kiện được đáp ứng thì kết quả sẽ trả về giá trị cụ thể từ table.

Mệnh đề WHERE không chỉ được sử dụng trong câu lệnh SELECT mà còn được sử dụng trong các lệnh UPDATE, DELETE,....

a) Cú pháp:

```
SELECT   column1,column2,....columnN
FROM   table_name
WHERE    [condition] ;
```

b) Toán tử LIKE

- Sử dụng để chọn những record có giá trị của thuộc tính phù hợp với mẫu được đưa ra
- Hai ký tự đại điện được dùng với LIKE là:
  - Ký tự phần trăm (%): đại diện cho không, một, hoặc nhiều số (hay ký tự)
  - Ký tự gạch dưới (_) : đại diện cho một số (hay ký tự)

c) Toán tử GLOB

- Tương tự như toán tử LIKE, được sử dụng để chọn những record có giá trị của thuộc tính phù hợp với mẫu được đưa ra.
- Khác biệt
  - Có phân biệt chữ hoa/thường (LIKE không phân biệt)
  - Hai ký tự đại diện được dùng kèm với GLOB là:
  - Ký tự hoa thị (*): đại diện cho không, một, hoặc nhiều số (hay ký tự)
  - Ký tự hỏi chấm (?): đại diện cho một số (hay ký tự)

d) Sử dụng NULL trong mệnh đề WHERE

NULL là một thuật ngữ được sử dụng để đại diện cho 1 giá trị thiếu và khác với giá trị số không (zero) hay khoảng trống(‘’)

3.10.5 GROUP BY

- Sử dụng khi câu truy vấn có sử dụng các hàm thuộc nhóm Aggregate (MIN,MAX,.....). Mục đích để sắp xếp dữ liệu thành các nhóm.
- Trong câu lệnh SELECT, mệnh để GROUP BY đi sau mệnh đề WHERE và đi trước mệnh đề ORDER BY.
- Cú pháp:

```
SELECT   column - list
FROM   table _ name
WHERE   [ condition  ]
GROUP BY   column1,  column2  ...  columnN
ORDER BY   column1,  column2  ...  columnN
```

3.10.6 HAVING

- Sử dụng khi câu truy vấn có sử dụng các hàm thuộc nhóm Aggregate (MIN,MAX,.....) trong biểu thức điều kiện. HAVING luôn đi chung và đi sau mệnh đề GROUP BY.
- Cú pháp:

```
SELECT   column1,  column2
FROM   table _ name
WHERE    CONDITION
GROUP BY   column1,  column2   HAVING  [ condition  ]
ORDER BY   column1,  column2
```

3.10.7 ORDER BY

- Sử dụng khi câu truy vấn có yêu cầu sắp xếp dựa trên giá trị của 1 hoặc nhiều cột.
- Cú pháp:

```
SELECT   column1,  column2
FROM   table _ name
WHERE    CONDITION
ORDER BY   column_ name   { ASC | DESC };
```

3.10.8 Sử dụng ALIAS trong câu lệnh SELECT

- Alias được sử dụng để đổi tên 1 table hoặc một cột tạm thời (trong câu lệnh đang được thực hiện) bằng cách đưa ra một cái tên khác, được gọi là bí danh.
- Cú pháp:

- Sử dụng đối với table

```
SELECT   column1,  column2
FROM  table name AS alias name  WHERE [  condition  ];
```

- Sử dụng đối với cột

```
SELECT   column _ name  AS  alias _name
FROM   table _ name   WHERE   [ condition  ];
```

3.10.9 Thiết lập độ rộng cột cho kết quả

Sử dụng lệnh **.width** *num,num,...*

Ví dụ : sqlite>.width 10,20,10

3.10.10 Thông tin về lược đồ (Schema)

Do các lệnh dẫn trước bởi dấu chấm chỉ có tác dụng trong dấu nhắclệnh của SQLite, vì vậy khi lập trình với SQLite để hiển thị danh sách các table có trong CSDL của mình, bạn cần sử dụng lệnh sau :

```
sqlite> SELECT tb1_name FROM sqlite_master WHERE type = 'table'
```

3.10.11 Mệnh đề/toán tử UNIONS trong SQLite

Mệnh đề/toán tử UNION được dùng để kết nối kết quả của 2 hay nhiều câu lệnh SELECT lại với nhau mà kết quả sau này sẽ *không* chứa những dòng trùng nhau

Để sử dụng UNION, mỗi lệnh SELECT phải có cùng số lượng cột, cùng kiểu dữ liệu(có thể không có cùng kích thước) và cùng được sắp xếp thứ tự trong lênh SELECT

- Cú pháp:

```
SELECT column1 [ , column2 ]
FROM table1 [ , table2 ]
[ WHERE condition ]
        UNION
SELECT column1 [ , column2 ]
FROM table1 [ , table2 ]
[ WHERE condition ]
```

Trong đó điều kiện condition có thể khác nhau tùy vào nhu cầu sử dụng

- The UNION ALL Clause:

Mệnh đề/toán tử UNION được dùng để kết nối kết quả của 2 hay nhiều câu lệnh SELECT lại với nhau mà kết quả sau này sẽ *có* chứa những dòng trùng nhau

Các quy tắc sử dụng cho UNION và UNION ALL là tương tự nhau

Cú pháp

```
SELECT column1 [ , column2 ]
FROM table1 [ , table2 ]
[ WHERE condition ]
        UNION ALL
SELECT column1 [ , column2 ]
```

```
FROM table1 [ , table2 ] [ WHERE condition ]
```

3.11 Update

- Dùng để hiệu chỉnh dữ liệu của các record trong table
- Sử dụng mệnh đề WHERE:
  - Không sử dụng: việc hiệu chỉnh sẽ thực hiện trên tất cả các record
  - Có sử dụng: chỉ những record nào thỏa điều kiện mới được hiệu chỉnh
- Cú pháp :

```
UPDATE table _ name
SET column1 = value1, column2 = value2, ...columnN = valueN
WHERE [ condition ];
```

3.12 Delete

- Dùng để xóa các record trong table
- Sử dụng mệnh đề WHERE:
  - Không sử dụng: xóa tất cả các record
  - Có sử dụng: chỉ xóa những record nào thỏa điều kiện
- Trong SQLite sau khi sử dụng lệnh DELETE, bạn nên sử dụng thêm lệnh VACUUM để xóa đi không gian không còn sử dụng
- Cú pháp:

```
DELETE FROM table_name WHERE [ condition ];
VACUUM
```

3.13 Lệnh VACUUM

Lệnh VACUUM làm sạch CSDL chính bằng cách sao chép nội dung của nó vào 1 file csdl tạm thời và tải lại các file csdl ban đầu từ các bản sao. Điều này giúp sắp xếp các dữ liệu table sát nhau. Cũng có thể sử dụng lệnh VACUUM để sửa đổi các thông số cấu hình CSDL cụ thể

3.13.1 Sử dụng lệnh VACUUM thủ công

Sử dụng đơn giản:

```
sqlite> VACUUM;
```

Sử dụng cho Database có tên được chỉ ra

```
$sqlite3 database_name "VACUUM;"
```

Sử dụng cho riêng table có tên được chỉ ra

```
sqlite> VACUUM table_name;
```

3.13.2 Sử dụng lệnh VACUUM tự động

- Lệnh *auto_vacuum* không thực hiện tương tự như lệnh VACUUM mà có chức năng giúp CSDL chống lại việc phân mảnh, nhờ vậy CSDL nhỏ gọn.
- Bạn có thể kích hoạt (Enable)/vô hiệu hóa (Disable) lệnh auto_vacuum tại dấu nhắc lệnh của SQLite như sau:

```
sqlite> PRAGMA auto_vacuum = NONE; hay = 0  =>disable
sqlite> PRAGMA auto_vacuum = INCREMENTAL; hay = 1
                                => enable incremental vacuum
sqlite> PRAGMA auto_vacuum = FULL; hay = 2
                                => enable full auto vacuum
```

- Sử dụng lệnh sau để kiểm tra thiết lập về auto_vacuum

```
$sqlite3   database_name  "PRAGMA  auto_vacuum;"
```

3.14 Sub Queries

- Một Subquery hay một Inner query hoặc Nested query là 1 lệnh truy vấn dữ liệu . Lệnh này được nhúng vào bên trong mệnh đề WHERE của một lệnh SQLite khác
- Mục đích của subquery là trả về dữ liệu và dữ liệu này sẽ được dùng để hạn chế số record thu được (hoặc bị ảnh hưởng) bởi lệnh SQLite chính chứa subquery

  Subquery có thể được nhúng vào bên trong mệnh đề WHERE của các lệnh SELECT, UPDATE, INSERT, DELETE và đi sau các toán tử như =,>,<,=>,=<, IN,BETWEEN,...
- Một số quy định khi dùng subqueries:
  - Subquery phải được đặt trong cặp ngoặc đơn
  - Subquery chỉ được có duy nhất 1 cột trong mênh đề SELECT của mình
  - Mệnh đề ORDER BY: không được sử dụng trong Subquery, mặc dù lệnh truy vấn chính vẫn có thể được dùng
  - Mệnh đề GROUP BY:có thể được dùng trong cả Subquery và Mainquery
  - Khi kết quả trả về từ Subquery gồm nhiều dòng , bạn chỉ được dùng những toán tử có khả năng so sánh với nhiều giá trị như toán tử IN
  - Toán tử BETWEEN không được dùng với Subquery, tuy nhiên toán tử BETWEEN vẫn có thể được dùng bên trong Subquery.

3.14.1 Sử dung Subqueries với lệnh SELECT

Cú pháp:

```
SELECT   column_name  [, column_name  ]
FROM     table1  [, table2  ]
WHERE   column_name   OPERATOR ( SELECT     column_name
                                              [, column_name  ]
                                 FROM      table1  [, table2  ]
                                 [  WHERE  ])
```

3.14.2 Sử dung Subqueries với lệnh INSERT

- Lệnh INSERT sử dụng dữ liệu trả về của subquery để thêm vào table khác. Có thể sử dụng các hàm về số, chuỗi, ngày/giờ trong subquery để tạo dữ liệu phù hợp với table được thêm dữ liệu vào.
- Cú pháp:

```
INSERT  INTO  table_name  [ (  column1  [,  column2])   ]
          SELECT  [  *| column1  [,  column2]   ]
          FROM  table1 [,  table2]
          [  WHERE  VALUE  OPERATOR  ]
```

3.14.3 Sử dung Subqueries với lệnh UPDATE

- Giúp việc cập nhật dữ liệu được thực hiện theo điều kiện cho trước, hay nói cách khác chỉ những record thỏa điều kiện có trong WHERE mới thúc hiện cập nhật lại dữ liệu
- Cú pháp:

```
UPDATE    table
SET    column_name  =  new_value
 [ WHERE  OPERATOR   [ VALUE ](SELECT  COLUMN_NAME
                                  FROM     TABLE_NAME
                                  [  WHERE ] )
```

3.14.4 Sử dung Subqueries với lệnh DELETE

- Giúp xóa những record thỏa điều kiện cho trước
- Cú pháp:

```
DELETE   FROM  TABLE_NAME
```

```
[ WHERE  OPERATOR  [ VALUE ] (SELECT  COLUMN_NAME
                                FROM    TABLE_NAME
                                [  WHERE ] )
```

3.15 Views

- View được cấu thành từ 1 lệnh truy vấn dữ liệu trong SQLite, do đó View sẽ chưa thành phần của nhiều hoặc 1 table trong CSDL
- Một View có thể được xem là một table ảo và có nhưng đặc điểm sau:
  - Cấu trúc dữ liệu theo cách người sử dụng mong muốn
  - Có thể là nguồi dữ liệu cung cấp cho lệnh SELECT khác
  - Thay vì hiển thị đầy đủ thông tin của table, View giúp hạn chế truy cập vào các dữ liệu mà người dùng không được phép xem hoặc sửa
  - Do dữ liệu của View là READONLY nên không thể thực hiện các thao tác DELETE, UPDATE hoặc INSERT trên 1 View

3.15.1 Tạo View

- Cú pháp:

```
CREATE   [ TEMP | TEMPORARY  ] VIEW  view_name  AS
SELECT   column1, column2  .....
FROM   table_name
WHERE    [ condition  ]
```

Trong đó : Nếu 1 trong 2 tùy chọn TEMP | TEMPORARY , View sẽ được tạo ra trong CSDL tạm thời

3.15.2 Dropping View

- Cú pháp:

```
sqlite>   DROP  VIEW    view_name;
```

3.16 Trigger

Trigger là chức năng do SQLite cung cấp cho người lập trình tự cài đặt. Sau khi cài đặt xong, Trigger tự động thực hiện khi 1 sự kiện xảy ra trên CSDL Một số điểm quan trọng về Trigger:

- Khi cài đặt Trigger cần chỉ định rõ:
  - Trigger áp dụng cho table nào?
  - Trigger được gắn với những thao tác nào trong các thao tác INSERT, DELETE hay UPDATE

- Hiện tại SQLite chỉ hỗ trợ Trigger cho FOR EACH ROW, chứ không hỗ trợ cho FOR EACH STATEMENT. Do đó khi tạo lập Trigger, việc ghi FOR EACH ROW trong lệnh là tùy chọn.
- Cả mệnh đề WHEN và Trigger actions có thể truy cập các phần tử của record được insert, delete hoặc update bằng cách sử dụng tham chiếu của mẫu NEW. column_name và OLD. column_name trong đó column_name là tên của 1 cột trong table mà Trigger được liên kết tới
- Nếu mệnh đề WHEN được chỉ ra các câu lệnh SQLite chỉ được thực hiện trên các record mà điều kiện trong WHEN là đúng. Nếu mệnh đề WHEN không được chỉ ra các câu lệnh SQLite được thực hiện trên mọi record.
- Từ khóa BEFORE hoặc AFTER cho biết Trigger sẽ được thực hiện trước (BEFORE) hay sau (AFTER) khi thực hiện các lệnh về INSERT, DELETE hay UPDATE trên các record.Trigger sẽ được tự động xóa khi table có liên quan đến nó bị xóa
- Cả table hoặc view được đính kèm Trigger và table được sủa đổi phải tồn tại trong CSDL do đó chỉ được dùng dạng table_name mà không database_name.table_name.
- Hàm RAISE() có thể được sử dụng khi lập trình với Trigger để nâng cao khả năng xử lý các ngoại lệ (exception).

3.16.1 Cú pháp

- Tạo Trigger trên table

```
CREATE  TRIGGER  trigger _ name [BEFORE | AFTER ] event_name
ON  table_name  [  FOR EACH ROW  ]
BEGIN
---  Trigger logic goes here .....   END;
```

Trong đó :

- **event_name** phải là các giá trị INSERT, DELETE hay UPDATE
- **table_name** chỉ ra tên table mà Trigger được gắn vào

- Tạo Trigger đối với thao tác UPDATE trên 1 hoặc nhiều cột của table

```
CREATE  TRIGGER  trigger _ name [BEFORE | AFTER ]
UPDATE  OF  column_name  ON  table_name
BEGIN
```

```
--- Trigger logic goes here .....
END;
```

3.16.2 Liệt kê các Trigger

- Liệt kê tên của các Trigger có trong CSDL bằng cách truy vấn dữ liệu trong table sqlite_master như sau:

```
sqlite> SELECT name FROM sqlite_master WHERE type='trigger';
```

- Liệt kê tên của các Trigger có trong một table

```
sqlite> SELECT name FROM sqlite_master
WHERE type='trigger' AND table_name = 'COMPANY';
```

3.16.3 Xóa Trigger

Sử dụng lệnh DROP TRIGGER và tên Trigger cần xóa theo cú pháp:

```
sqlite> DROP TRIGGER trigger_name
```

3.17 Transactions

Transactions là 1 hoặc các chuỗi công việc (lệnh) thực hiện theo thứ tự hợp lý

3.17.1 Các đặc tính tiêu chuẩn của Transactions

Các Transactions có bốn đặc tính sau đây :

- **Atomicity**(tính nguyên tử): đảm bảo rằng tất cả các hoạt động trong đơn vị công việc được hoàn thành, nếu không các Transactions bị hủy bỏ tại thời điểm thất bại và các hoạt động trước đó được cuộn trở lại tình trạng bắt đầu
- **Consistenci**(Tính nhất quán): đảm bảo rằng CSDL thay đổi trạng thái theo đúng những gì đã thực hiện trong Transactions
- **Isolation**(Tính cách ly): cho phép các Transactions hoạt động độc lập và minh bạch với nhau
- **Durability**(Độ bền): đảm bảo rằng các kết quả hoặc hiệu quả của 1 Transactions vẫn tồn tại trong trường hợp lỗi hệ thống.

3.17.2 Các lệnh sử dụng để điều khiển Transactions

Các lệnh điều khiển về Transactions chỉ được dùng với các lệnh INSERT, DELETE hay UPDATE mà không thể dùng với các lệnh CREATE TABLE hoặc DROP TABLE vì các lệnh này được thực hiện tự động trong CSDL

- **BEGIN TRANSACTIONS**
  - Transactions có thể được khởi tạo bằng cách dùng lệnh BEGIN TRANSACTIONS hoặc đơn giản hơn là BEGIN . Có thể xem các lệnh sử

dụng trong Transactions được xử lý trong bộ nhớ tạm cho đến khi gặp lệnh COMMIT hoặc ROLLBACK

- Cú pháp:

```
BEGIN;
or
BEGIN TRANSACTIONS;
```

- **COMMIT hoặc END TRANSACTIONS**

- Lệnh COMMIT dùng để lưu lại tất cả những thay đổi trên CSDL kể từ lệnh COMMIT hay ROLLBACK gần nhất đến lệnh COMMIT đang xét.

- Cú pháp :

```
COMMIT
or
END TRANSACTIONS
```

- Công dụng:

  - Lệnh ROLLBACK được dùng để bỏ qua các giao tác vừa thực hiện trên CSDL kể từ lệnh COMMIT hay ROLLBACK gần nhất đến lệnh ROLLBACK đang xét.
  - Một Transactions cũng sẽ tự động ROLLBACK khi CSDL đang làm việc bị đóng hoặc có lỗi xảy ra

- Cú pháp:

```
ROLLBACK;
```

3.18 Indexes

Thường trong những cuốn sách về kĩ thuật , phần cuối của cuốn sách sẽ liệt kê các từ khóa xuất hiện trong cuốn sách theo thứ tự alphabet và số trang nơi từ khóa đó xuất hiện. Khi cần tra cứu từ khóa nào, bạn tra cứu trong phần đó, dựa vào số trang đi kèm từ đó bạn sẽ tra ra nội dung của từ 1 cách dễ dàng.

Trong SQLite Indexes được tổ chức tương tự như vậy. Khi đó Indexes được tổ chức thành các table giúp tăng tốc đọ tìm kiếm, truy xuất dữ liệu.

Chỉ số cũng có thể là duy nhất, trong đó chỉ số ngăn cản mục trùng lặp trong cột hoặc kết hợp các cột mà trên đó có một chỉ số

3.18.1 Tạo Indexes

- Indexes dựa trên duy nhất 1 cột

```
CREATE INDEX Index_name ON table_name (column_name);
```

- Indexes dựa trên nhiều cột

```
CREATE INDEX Index_name ON table_name (column1, column2.....);
```

Thông thường người ta chit tạo Index dựa trên khóa chính của table. Vì vậy khi tạo Index mà các thuộc tính tham gia tạo Index không phải là khóa chính có thể làm ảnh hưởng đến hiệu suất của Index

- Implicit Indexes

Index tiềm ẩn (Implicit Indexes) là các Index có thể được tự động tạo bởi các database server. Khi đó các Index được tự động tạo ra dựa trên các rằng buộc khóa chính và rằng buộc unique

3.18.2 Liệt kê các Index

Sử dụng lệnh **.indices** đi kèm tên các table để xem Index của table :

```
sqlite> .indices  COMPANY;
```

Kết quả hiển thị có 2 Index, trong đó *sqlite_autoindex_COMPANY_1* là 1 Implicit Indexes , Index này được tạo ra ngay sau khi lệnh tạo table được thực hiện

```
id_index  sqlite_autoindex_COMPANY_1
```

3.18.2 Xóa Index

Sử dụng lệnh DROP với cú pháp như sau:

```
DROP INDEX index_name;
```

Ví dụ:

```
sqlite> DROP INDEX id_index;
```

3.18.3 Một số trường hợp nên tránh dùng indexes

Mặc dù indexes nhằm nâng cao hiệu suất của một CSDL, tuy nhiên có những lúc bạn cần cân nhắc khi sử dụng indexes trong các trường hợp sau:

- Kích thước table nhỏ.
- Table thường xuyên thực hiện hàng loạt thao tác cập nhật hoặc chèn dữ liệu.
- Cột dự định sử dụng làm indexes có chứa một số lượng lớn các giá trị NULL.

- Lập chỉ mục trên những cột thường xuyên phải thao tác (thêm, xóa, sửa).

3.18.4 Indexed By

Mệnh đề "*INDEXED BY index-name*" đi kèm trong câu lệnh cho biết việc dò tìm dữ liệu phải được ưu tiên thực hiện trên index trước.

Mệnh đề "*NOT INDEXED*" cho biết không có index được sử dụng khi truy cập vào table. Tuy nhiên, INTEGER PRIMARY KEY vẫn có thể được sử dụng để tìm ngay cả khi đã chỉ rõ mệnh đề "*NOT INDEXED*" trong câu lệnh.

Cú pháp :

Sử dụng được cho các lệnh DELETE, UPDATE hoặc SELECT:

```
SELECT|DELETE|UPDATE column1, column2...
INDEXED BY (index_name) table_name  WHERE
(CONDITION);
```

3.19 Các hàm thường dùng trong SQLite

3.19.1 Hàm về date và time:

Function	***Ví dụ***
`date(timestring,modifiers)`	Trả về ngày theo định dạng: YYYY-MM-DD
`time(timestring,modifiers)`	Trả về thời gian theo định dạng HH:MM:SS
`datetime(timestring,modifiers...)`	Trả về ngày giờ theo định dạng YYYY-MM-DD HH:MM:SS
`julianday(timestring,modifiers...)`	Trả về số ngày tính từ November 24, 4714 B.C.
`strftime(timestring, modifiers..)`	Trả về ngày được định dạng theo chuỗi định dạng được chỉ ra trong đối số đầu tiên. Tham khảo cách sử dụng chuỗi cho đối số đầu tiên trong các phần tiếp sau

Cả 5 hàm về ngày và thời gian ở trên đều có tham số *timestring*. *Timestring* được theo sau bằng số không (zero) hoặc các định dạng bổ sung khác (modifiers).

a. Time Strings

Timestring gồm các kiểu định dạng như sau:

Time String	*Ví du*
YYYY-MM-DD	2010-12-30
YYYY-MM-DD HH:MM	2010-12-30 12:10
YYYY-MM-DD HH:MM:SS.SSS	2010-12-30 12:10:04.100
MM-DD-YYYY HH:MM	30-12-2010 12:10
HH:MM	12:10
YYYY-MM-DDTHH:MM	2010-12-30 12:10
HH:MM:SS	12:10:01
YYYYMMDD HHMMSS	20101230 121001
Now	2013-05-07

Bạn có thể sử dụng "T" làm ký tự ngăn cách giữa date và time.

b. Modifiers

Timestring được theo sau bằng số không (zero) hoặc các định dạng bổ sung khác (modifiers), những định dạng này sẽ thay thế date và/hoặc time trả về bởi bất kỳ hàm nào trong 5 hàm ở trên.

- NNN days
- NNN hours
- start of month
- weekday N
- NNN months
- NNN minutes
- start of year
- unixepoch
- NNN years
- NNN.NNNN
- start of day
- localtime

c. Formatters

SQLite cung cấp hàm *strftime()* rất tiện dụng để định dạng về date và time. Bạn có thể sử dụng các định dạng thay thế sau đây để định dạng lại date và time của mình:

Định dạng thay thế	Diễn giải
%d	**Day of month, 01-31**
%f	**Fractional seconds, SS.SSS**
%H	**Hour, 00-23**
%j	**Day of year, 001-366**
%J	**Julian day number, DDDD.DDDD**
%m	**Month, 00-12**
%M	**Minute, 00-59**
%s	**Seconds since 1970-01-01**
%S	**Seconds, 00-59**
%w	**Day of week, 0-6 (0 is Sunday)**
%W	**Week of year, 01-53**
%Y	**Year, YYYY**
%%	**% symbol**

3.19.2 Hàm về số

- Có thể sử dụng tên hàm dưới dạng chữ thường, chữ hoa hay vừa thường vừa hoa đều được.
- Cho dữ liệu mẫu của table COMPANY như sau:

```
ID              NAME            AGE         ADDRESS     SALARY
----------      ----------      ----------  ----------  ----------
1               Paul            32          California  20000.0
2               Allen           25          Texas       15000.0
3               Teddy           23          Norway      20000.0
4               Mark            25          Rich-Mond   65000.0
5               David           27          Texas       85000.0
6               Kim             22          South-Hall  45000.0
7               James           24          Houston     10000.0
```

a. **COUNT**

– Đếm số dòng có trong table.

– Ví dụ:

```
sqlite> SELECT count(*) FROM COMPANY;
```

– Kết quả

```
count(*)---- 7
```

b. **MAX**

– Chọn giá trị lớn nhất trên cột có tên được chỉ ra.

– Ví dụ:

```
sqlite> SELECT max(salary) FROM COMPANY;
```

– Kết quả

```
max(salary)---------- 85000.0
```

c. **MIN**

– Chọn giá trị nhỏ nhất trên cột có tên được chỉ ra.

– Ví dụ:

```
sqlite> SELECT min(salary) FROM COMPANY;
```

– Kết quả

```
min(salary)---------- 10000.0
```

d. **AVG**

– Lấy giá trị trung bình trên cột có tên được chỉ ra.

– Ví dụ:

```
sqlite> SELECT avg(salary) FROM COMPANY;
```

– Kết quả

```
avg(salary)-------- 37142.8571428572
```

e. **SQLite SUM Function**

- Lấy tổng giá trị trên cột có tên được chỉ ra.
- Ví dụ:

```
sqlite> SELECT sum(salary) FROM COMPANY;
```

- Kết quả

```
sum(salary)---------- 260000.0
```

f. **RANDOM**

- Chọn giá trị nguyên ngẫu nhiên trong khoảng từ -9223372036854775808 đến +9223372036854775807.
- Ví dụ:

```
sqlite> SELECT random() AS Random;
```

- Kết quả

```
Random------------------ 5876796417670984050
```

g. **ABS**

- Trả về trị tuyệt đối của đối số.
- Ví dụ:

```
sqlite> SELECT abs(5), abs(-15), abs(NULL),
abs(0), abs("ABC");
```

- Kết quả

```
abs(5)      abs(-15)     abs(NULL)    abs(0)    abs("ABC")
----------  -------      ---------    ------    ---------
5           15                        0         0.0
```

3.19.3 Hàm xử lý chuỗi

a. **UPPER**

- Chuyển chuỗi tham số của hàm thành chữ hoa.
- Ví dụ:

```
sqlite> SELECT upper(name) FROM COMPANY;
```

Kết quả

```
upper(name)

-----------
PAUL ALLEN
TEDDY MARK
DAVID KIM
JAMES
```

b. **LOWER**

- Chuyển chuỗi tham số của hàm thành chữ thường.
- Ví dụ:

```
sqlite> SELECT lower(name) FROM COMPANY;
```

Kết quả

```
lower(name)

-----------

paul
allen
teddy
mark
david
kim
james
```

c. **LENGTH**

- Lấy chiều dài của tham số chuỗi.
- Ví dụ:

```
sqlite> SELECT name, length(name) FROM COMPANY;
```

- Kết quả

```
NAME                        length(name)
----------                  ------------
Paul                        4
Allen                       5
Teddy                       5
Mark                        4
David                       5
Kim                         3
James                       5
```

3.19.4 Hàm khác

sqlite_version

- Trả về version của SQLite library đang dùng.
- Ví dụ:

```
sqlite> SELECT sqlite_version() AS 'SQLite Version';
```

- Kết quả

```
SQLite Version
----------
3.6.20
```

Chương 4: CHƯƠNG TRÌNH THỰC NGHIỆM

4.1 Bài Toán

Điện thoại thông minh ngày càng mạnh mẽ có nhiều chức năng hỗ trợ cho con người trong học tập và lao động. Nó còn cho phép cài đặt các trương trình được phát triển bởi người dùng. Chương trình quản lý chi tiêu cá nhân sử dụng cơ sở dữ liệu SQlite sẽ giúp chủ nhân ghi lại mình đã mua gì trong tháng , hết bao nhiêu tiền ... Đặc biệt , không còn phải đau đầu khi thấy thiếu tiền mà không nhớ nổi đã chi cho cái gì.

4.2 Phân tích và thiết kế :

Để thực hiện chương trình quản lý chi tiêu cần sử dụng :

4.2.1 Môi trường lập trình Android Studio

Xây dụng chương trình quản lý chi tiêu với các chức năng và giao diện gồm có :

- Xem các khoản chi
- Thêm khoản chi
- Sủa khoản chi
- Xóa khoản chi

4.2.2 Sử dụng CSDL : SQLite

Sử dụng các lệnh cơ bản của SQLite để thực hiện chương trình:

- Tạo CSDL
- Tạo bảng trong CSDL
- Lấy dữ liệu trong bảng(SELECT)
- Nhập vào bảng(INSERT)
- Cập nhật thông tin cho bảng(UPDATE)
- Xóa các trường trong bảng(DELETE)

4.2.3 Tối ưu chương trình bằng chức năng bảo mật

Chức năng bảo mật giúp người sử dụng có thể giữ được thông tin các khoản chi của mình mà không sợ người khác xem được

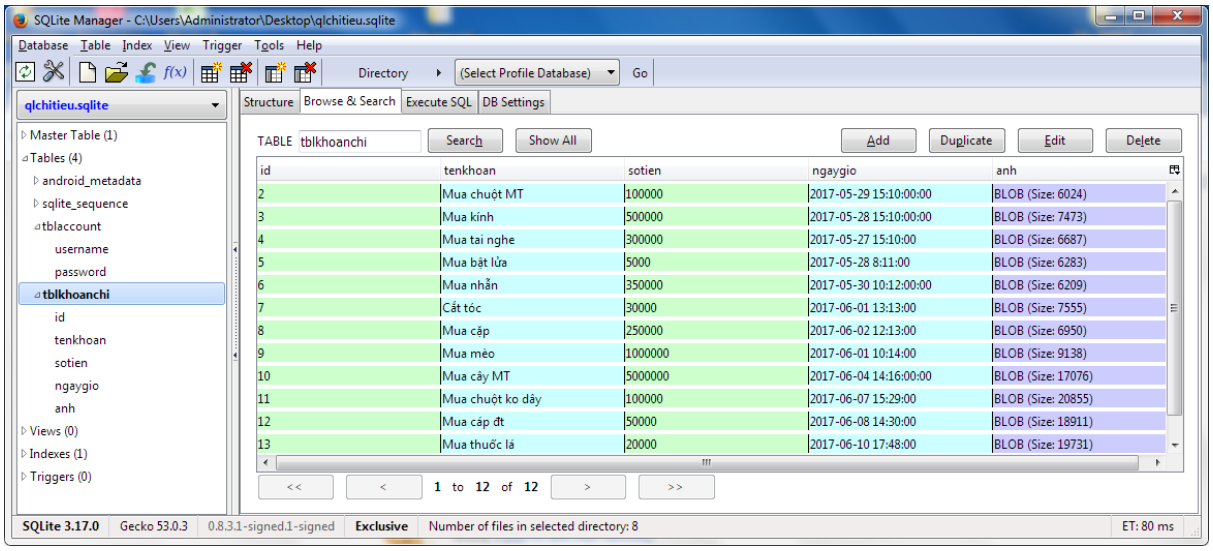
Chức năng gồm có :

- Đăng nhập

- Tạo tài khoản(account) bao gồm mật khẩu(password)
- Thay đổi mật khẩu
- Xóa tài khoản

4.3 CSDL SQLite

Cơ sở dữ liệu có bảng "tblkhoanchi" gồm các cột như hình ảnh:

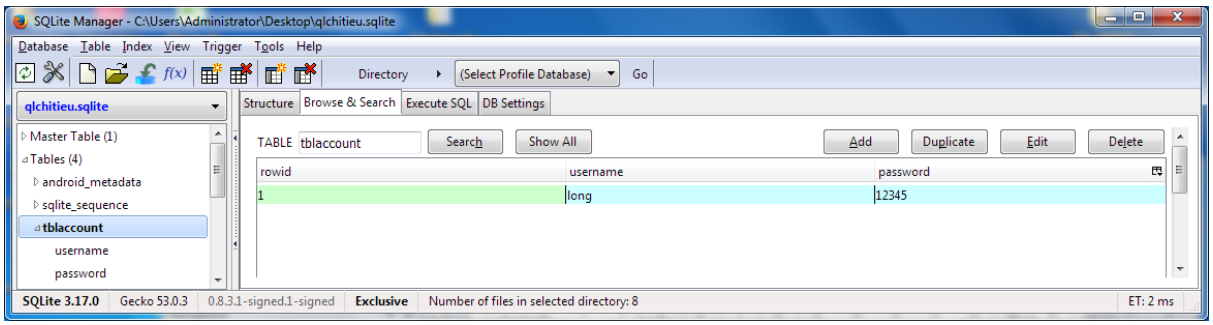


Trong bảng bao gồm các trường:

- id : id khoản chi
- tenkhoan : Tên khoản chi
- sotien : Số tiền chi từng khoản
- ngaygio : Ngày giờ chi
- anh : Ảnh khoản chi

Bên cạnh đó :

Cơ sở dữ liệu có bảng "tblaccount" giúp quản lý tài khoản người dùng gồm các cột như hình ảnh:



Trong đó bao gồm các trường:

- rowid : id tài khoản

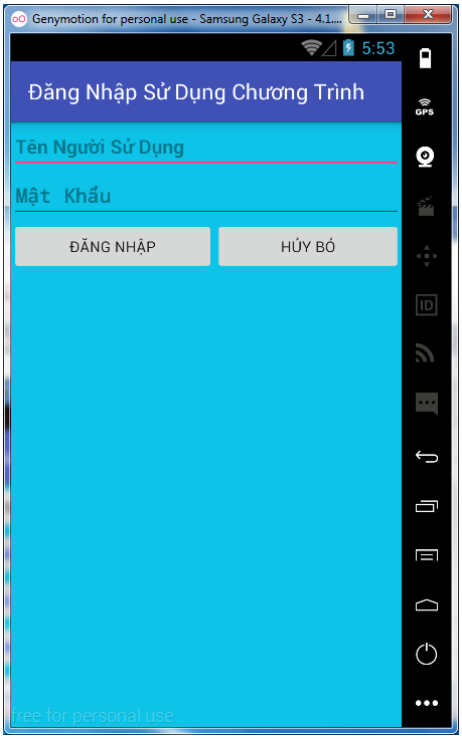
- username : tên người dùng
- password : mật khẩu

4.4 Giao diện chương trình

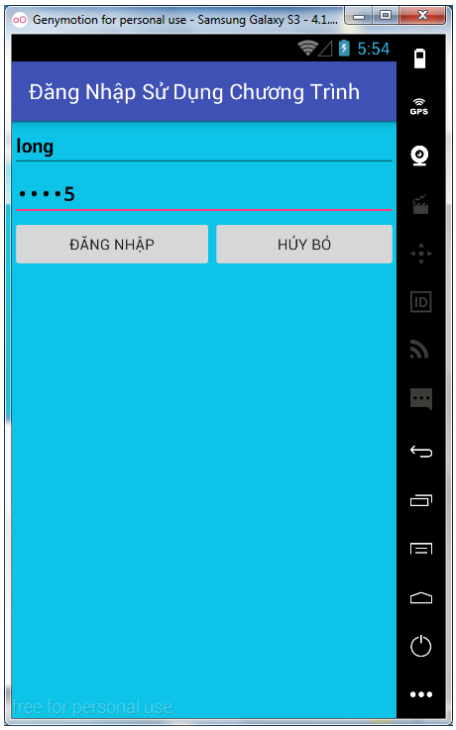
Là chương trình xây dựng ứng dụng truy xuất CSDL đến SQLite

Giao diện chính:

Giao diện đăng nhập :

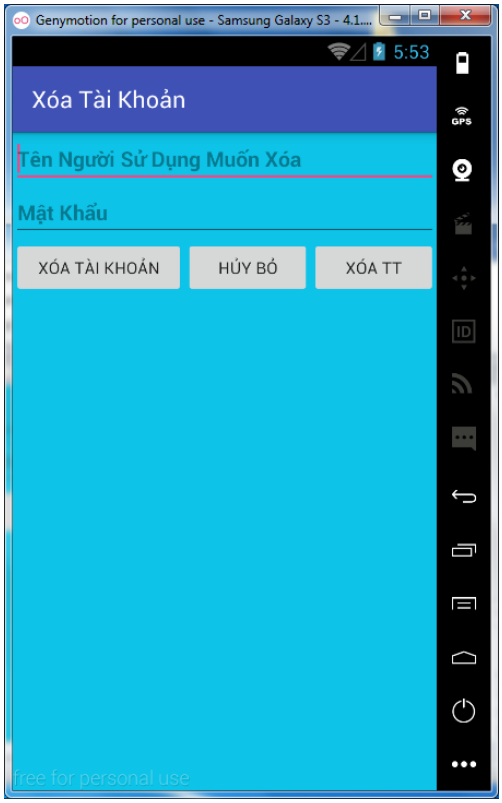


Giao diện tạo tài khoản :



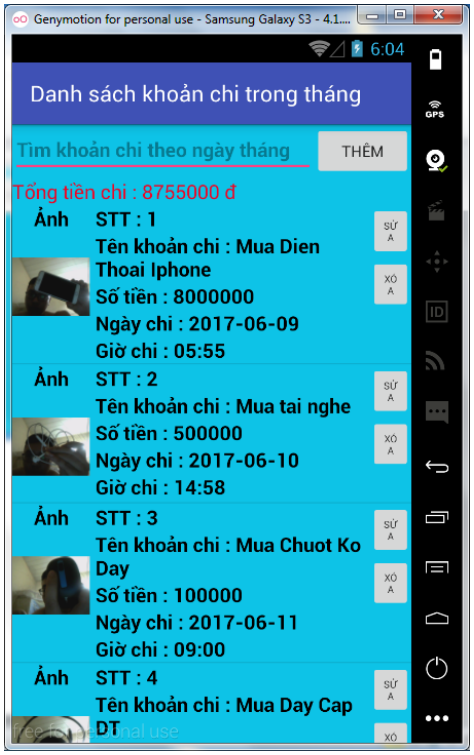
Giao diện đổi mật khẩu:

Giao diện xóa tài khoản:



Giao diện Xóa Dữ Liệu :

Sau khi đăng nhập thành công. Ta sẽ vào đc giao diện chính của chương trình:



Trên giao diện chính có thể thấy các chức năng của trương trình :

- Search
- Thêm
- Sửa
- Xóa
- Hiển thị các khoản chi trên ListView ngay màn hình chính

Giao diện Tìm Kiếm khoản chi (search) :

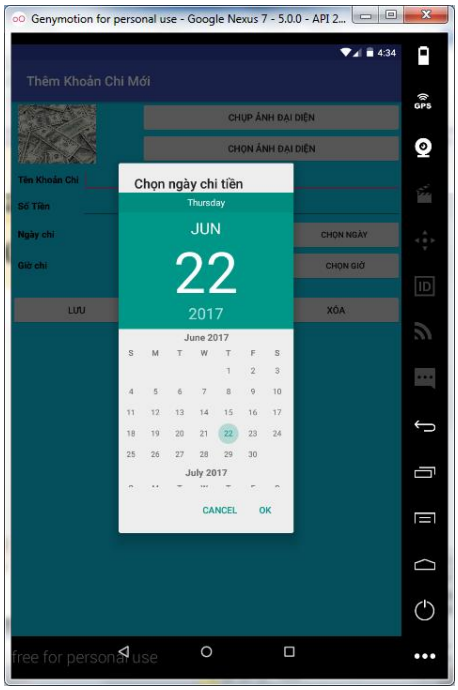
Giao diện Thêm khoản chi:

Chưa thêm thông tin

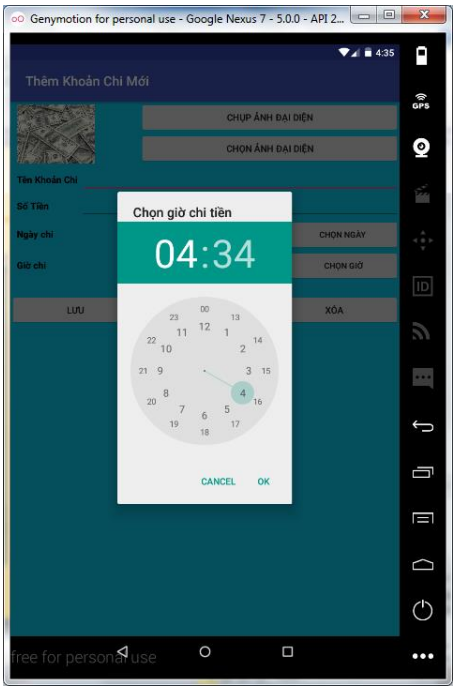
Đã thêm thông tin

Ở giao diện thêm khoản chi. Nếu ta chưa thêm thông tin thì "Ngày" và " Giờ"đã được lấy tự động từ giờ hệ thống.

Nếu muốn chỉnh " Ngày " và " Giò " ta nhấn vào chọn ngày và chọn giờ sẽ xuất hiện giao diện như sau:



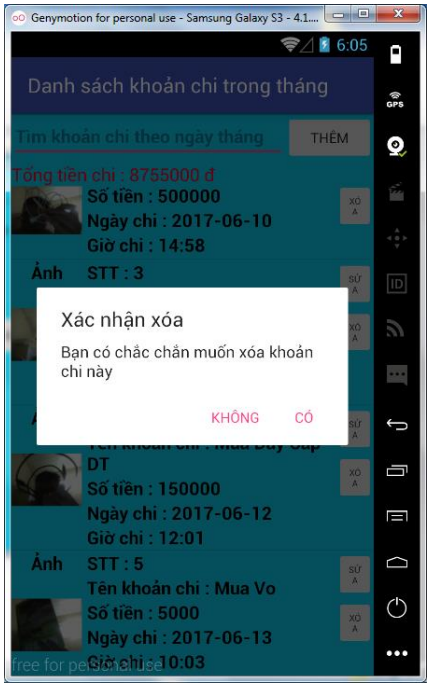
Chọn Ngày



Chọn Giờ

Giao diện Sửa khoản chi và xóa khoản chi:

Sửa



Xóa

4.5. Kết quả đạt được

Bước đầu đồ án đã đạt được kết quả như sau

- Tạo lập thành công cơ sở dữ liệu SQLite
- Xây dựng thành công ứng dụng android đã làm có thể thao tác được trên CSDL SQLite như: thêm mới, sửa, xóa.
- Bước đầu xây dựng được chức năng bảo mật hệ thống.
- Tuy nhiên vẫn còn tồn tại các hạn chế sau:
- Ứng dụng chưa có chức năng phân quyền về chức năng bảo mật và riêng rẽ về CSDL của từng tài khoản
- Ứng dụng chưa có chức năng tự động sắp xếp lại trên listview .

KẾT LUẬN

Trên đây em đã khảo sát trên mặt lý thuyết đối với xây dựng ứng dụng Android. Đồ án hướng tới mục tiêu xây dựng ứng dụng Android truy xuất cơ sở dữ liệu với SQLite. Trong khoảng thời gian nhất định dành cho việc thực hiện đề tài, nên một số vấn đề vẫn chưa được hoàn chỉnh. Tuy nhiên, đồ án đã đạt được một số kết quả:

- Về lý thuyết: Tìm hiểu, nghiên cứu được cách tạo cơ sở dữ liệu trên SQLite, các kỹ thuật lập trình với cơ sở dữ liệu để xây dựng ứng dụng Android truy xuất cơ sở dữ liệu như: tìm kiếm dữ liệu, thêm, sửa, xóa dữ liệu từ Android lên CSDL SQLite
- Về thực nghiệm: Sử dụng các kỹ thuật lập trình với cơ sở dữ liệu SQLite để xây dựng được ứng dụng truy xuất cơ sở dữ liệu với các thao tác với dữ liệu như: xem, sửa, thêm, xóa.

Do thời gian hạn chế, nên hiện tại đồ án mới chỉ dừng lại ở thao tác với cơ sở dữ liệu. Chưa làm sâu về bảo mật chương trình và chương trình vẫn còn một số hạn chế . Trong tương lai em sẽ tiếp tục phát triển ứng dụng hoàn chỉnh hơn.

TÀI LIỆU THAM KHẢO

Tài liệu tham khảo trực tuyến

http://duythanhcse.wordpress.com

https://thangcoder.com

http://khoapham.vn/KhoaPhamTraining/android