LÊ THỊ HỒNG QUYÊN MSSV: N14CCCN059 Tên Đề tài:Xây dựng ứng dụng Chat trên Android Lớp: C14CQCP01-N 2016 - 2017

**Gáy**

**TẬP ĐOÀN BƯU CHÍNH VIỄN THÔNG VIỆT NAM**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**BÁO CÁO THỰC TẬP**

**TỐT NGHIỆP CAO ĐẲNG**

***Đề tài*: “** **Xây dựng ứng dụng Chat trên Android”**

**Người hướng dẫn : HỒ ĐẮC HƯNG**

**Sinh viên thực hiện : LÊ THỊ HỒNG QUYÊN**

**Mã số sinh viên : N14CCCN059**

**Lớp : C14CQCP01-N**

**Khoá** **: 2014-2017**

**Hệ** **: CAO ĐẲNG CHÍNH QUY**

**TP.HCM, tháng 3 /2017**

TP. HCM

2017

**TẬP ĐOÀN BƯU CHÍNH VIỄN THÔNG VIỆT NAM**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**BÁO CÁO THỰC TẬP**

**TỐT NGHIỆP CAO ĐẲNG**

***Đề tài*: “ Xây dựng ứng dụng Chat trên Android”**

**Người hướng dẫn : HỒ ĐẮC HƯNG**

**Sinh viên thực hiện : LÊ THỊ HỒNG QUYÊN**

**Mã số sinh viên : N14CCCN059**

**Lớp : C14CQCP01-N**

**Khoá** **: 2014-2017**

**Hệ** **: CAO ĐẲNG CHÍNH QUY**

**TP.HCM, tháng 3 /2017**

**MỞ ĐẦU**

Các thiết bị công nghệ di động như smartphone, máy tính bảng, thiết bị chơi game cầm tay,… ngày càng phát triển mạnh mẽ vì thế đã tạo nên một cuộc cạnh tranh giữa các hãng sản xuất thiết bị như Samsung, Apple, HTC, Sony,… để trở thành hang sản xuất lớn mạnh và đứng đầu thị trường. Nó không chỉ cạnh tranh giữa các hãng mà còn tạo ra cuộc cạnh tranh giữa các hãng phát triển các hệ điều hanh chạy trên các thiết bị đó ví dụ như Google, Apple, RIM, Microsoft,… Và trong các hãng đó thì Google với hệ điều hành Android của mình đang là hệ điều hành phổ biến nhất trên các thiết bị di động. Và với thời buổi công nghệ thông tin phát triển như hiện nay, nhắn tin qua hệ thống mạng đang là xu hướng, thói quen phổ biến của người dùng với các ưu điểm như tiết kiệm, thuận tiện và dễ dàng. Em làm báo cáo này để xây dựng một ứng dụng chat trên android đơn giản, demo xem cách xây dựng nó, hoạt động của nó như thế nào…

**LỜI CẢM ƠN**

Em xin gởi lời cám ơn chân thành tới các thầy cô trong khoa công nghệ thông tin Học viện Công nghệ Bưu chính Viễn Thông đã cùng với tri thức và tâm huyết của mình để dạy dỗ và truyền đạt cho em những kiến thức quý giá trong suốt thời gian học tập tại trường. Cám ơn quý công ty PINE ENTERTAINMENT đã cho phép và tạo điều kiện thuận lợi để em thực tập tại công ty. Và đặc biệt, em xin tỏ lòng biết ơn sâu sắc đến thầy Hồ Đắc Hưng đã tận tình hướng dẫn trong suốt quá trình thực tập tốt nghiệp này. Do thời gian và kiến thức có hạn nên đồ án thực tập không tránh khỏi có điểm thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô, các anh chị cũng như các bạn học cùng lớp để kiến thức của em trong lĩnh vực này được hoàn thiện hơn. Sau cùng, em xin kính chúc các thầy cô trong khoa, các anh chị công ty PINE ENTERTAINMENT thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Sinh viên thực hiện

Lê Thị Hồng Quyên

**MỤC LỤC**

**Mở đầu**:

Trang

**CHƯƠNG 1: WEB API**

* 1. **Giới thiệu về API** 1
  2. **Giới thiệu Web API** 2
  3. **Ưu điểm của Web API so với WCF REST Service** 3

**CHƯƠNG 2: LẬP TRÌNH ANDROID**

**2.1. Tổng quan về hệ điều hành Android** 4

**2.2. Giao diện** 7

**2.3. Ưu và nhược điểm** 8

**2.4. Thiết lập môi trường** 9

**2.5. Fragmentation trong Android** 9

**2.6. Các thành phần cơ bản của ứng dụng Android** 10

**2.6.1. Activity** 10

**2.6.2. Service** 13

**2.6.3. Broadcast Receivers** 14

**2.6.4. Content Provider** 15

**2.6.5. Intent** 17

**2.6.6.Notification** 18

**CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ CƠ SỞ DỮ LIỆU**

20

**CHƯƠNG 4: XÂY DỰNG WEB SERVICE**

**4.** **1. Giới thiệu về Web service**

**4.** **2. Đặc điểm của Web service**

**4.** **3. Kiến trúc của Web service**

**4.** **4. Các thành phần của Web service**

**4.** **5. An toàn cho Web service**

**4.** **6. Xây dựng một dịch vụ Web**

**4.** **7. Tích hợp dịch vụ Web theo chuẩn**

**CHƯƠNG 5: XÂY DỰNG ỨNG DỤNG ANDROID VÀ CÀI ĐẶT CƠ SỞ DỮ LIỆU**

**5.1. Phân tích yêu cầu**

**5.2. Lược đồ Use case**

**5.3. Sơ đồ lớp diagram**

**5.4. Mô hình thực thể ERD**

**CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC**

**6.1. Client Android**

**6.1.1. Giao diện đăng nhập**

**6.1.2. Màn hình chính**

**6.1.3. Khung chat**

**6.1.4. Thông tin bản thân**

**6.1.5. Thêm bạn**

**6.1.6. Xóa bạn**

**6.1.7. Đổi mật khẩu và thông tin cá nhân**

**6.1.8 Đăng xuất**

**6.2. Server NodeJS**

**KẾT LUẬN 35**

**PHỤ LỤC 40**

**DANH MỤC TÀI LIỆU THAM KHẢO 50**

**DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| STT | Chữ viết tắt, ký hiệu | Chữ viết đầy đủ, giải thích |
| 1 | API | Application Programming Interface |
| 2 | server | Hệ thống |
| 3 | Remote server | Máy chủ từ xa |
| 4 | Requests | Yêu cầu |
| 5 | local server | Máy chủ cục bộ |
| 6 | JSON | JavaScript Object Noattion |
| 7 | HTTP | HyperText Transfer Protocol |
| 8 | SDK | Software Development Kit |
| 9 | JDK | Java Development Kit |
| 10 | ADV | Android Virtual Device |
| 11 | UNIQUE | Ngăn cản việc hai bản ghi có các giá trị đồng nhất trong một cột cụ thể |
| 12 | component | Thành phần |
| 13 | folder | Thư mục |
| 3 | internet | Mạng |

**DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH**

HÌNH 2.1 Các tầng kiến trúc của hệ điều hành Android

HÌNH 2.2 Giao diện Android 5.0 trên điện thoại

HÌNH 2.3 Sơ đồ vòng đời cùa Activity

HÌNH 2.4 Sơ đồ minh họa vòng đời service

HÌNH 3.1 Bảng các cột trong table User

HÌNH 3.2 Ràng buộc UNIQUE của username trong table User

HÌNH 3.4 Bảng các cột trong table Friends

HÌNH 3.4 Bảng các cột trong table Offline

HÌNH 4.1 Chồng giao thức của Web service

HÌNH 4.2 Tầng giao thức tương tác dịch vụ

HÌNH 5.1 Use case của User

HÌNH 5.2 Sơ đồ lớp diagram

HÌNH 5.3 Mô hình thực thể ERD

HÌNH 6.1 Giao diện đăng nhập trên máy ảo

HÌNH 6.2 Tab Friends (danh sách bạn bè) - Màn hình chính đã đăng nhập của ứng dụng với 2 máy ảo AVD và Genymotion

HÌNH 6.3 Tab More tại giao diện màn hình chính

HÌNH 6.4 Tab List online tại giao diện màn hình chính

HÌNH 6.5 Khung chat của ứng dụng

HÌNH 6.6 Trang Your profile – Thông tin cá nhân người dùng

HÌNH 6.7 Giao diện thêm một người bạn

HÌNH 6.8 Cách xóa một người bạn trong danh sách bạn bè

HÌNH 6.9 Các thông tin cá nhân của người dùng khi chưa thay đổi

HÌNH 6.10 Màn hình khi nhấp vào thay đổi mật khẩu

HÌNH 6.11 Màn hình khi nhấp nút Sign out – đăng xuất

HÌNH 6.12 Các gói của server NodeJS

**CHƯƠNG 1: WEB API**

**1.1 Giới thiệu về API**

- Về “máy móc”, API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng). Hầu như công ty nào cũng đã từng xây dựng qua API cho khách hàng, hoặc cho mục đích nội bộ.

- API là các hàm , phương thức để cho các ứng dụng bên ngoài có thể gọi , tương tác để trao đổi thông tin , tính toán.

- Việc trao đổi này giúp các nhà lập trình tạo ra các service  hỗ trợ những lập trình viên khác có thể tương tác với ứng dụng của chính mình.

- “Application” có thể ám chỉ nhiều thứ. Trong phạm vi API, Application có thể là:

+ Một phần mềm với chức năng riêng riêng biệt.

+ Cả server, cả ứng dụng, hoặc một phần nhỏ của ứng dụng.

- Về cơ bản, bất cứ phần mềm nào cũng có thể được tách biệt rạch ròi khỏi moi trường của nó, có thể trở thành “A” trong API, và có thể sẽ chứa một API nào đó.

- Mỗi trang trên internet được lưu trữ đâu đó trong một remote server. Remote server cũng không quá “huyền ảo”, mà chỉ là một máy tính được đặt tại một vị trí nhất định, được tối ưu để xử lý requests.

- Nói cách khác, bạn có thể hoàn toàn biến chiếc laptop của mình thành một server có khả năng đưa website lên Web (trong thực tế, local server là phương tiện để các kỹ sư có thể phát triển website trước khi tung ra công khai).

- Ví dụ, khi bạn gõ [www.facebook.com](https://www.facebook.com/) trên trình duyệt, một request sẽ được gửi đến remote server của Facebook. Khi trình duyệt đã nhận được phản hồi, code sẽ được dịch và page được hiển thị.

- Với trình duyệt (hay còn gọi là client), server của Facebook là một API. Đồng nghĩa với việc mỗi khi bạn truy cập một page trên Web, bạn sẽ tương tác với API của remote server tương ứng.

- Lưu ý, API không phải là remote server – mà là một bộ phận của server, chịu trách nhiện nhận requests và gửi phản hồi.

- Hiện nay, hầu như website nào cũng có sử dụng API của bên thứ ba.

- Trên web các dịch vụ của google, facebook cung cấp rất nhiều API để lập trình viên có thể xây dựng tương tác giữa website của họ với google, facebook.

- Mỗi phần mềm, ứng dụng có các cung cấp các API để các ứng dụng khác có thể tương tác với nó. Và việc xây dựng lên các API để cho các ứng dụng bên ngoài cũng cần tuân thủ các chuẩn công nghệ để nhiều nền tảng công nghệ có thể sử dụng được API mà ứng dụng cung cấp.

**1.2 Giới thiệu Web API**

**-** Web API là công nghệ mới nhất của Microsoft để xây dựng các dịch vụ thành phần phân tán.

**-** Web API hỗ trợ mô hình MVC: routing, controller, action result, filter, model binder, IoCcontainer, dependency injection, unit test.

**-** Web API hỗ trợ RESTfull đầy đủ phươngthức: GET/ POST/ PUT/ DELETE dữ liệu.

**-** REST (Representational State Transfer) là kiến trúc được sử dụng trong việc giao tiếp giữa các máy khách chủ, trong việc quản lý các tài nguyên trên internet. REST được sử dụng rất nhiều trong việc phát triển các ứng dụng Web Services sử dụng giao thức HTTP trong giao tiếp thông qua mạng internet. Các ứng dụng sử dụng kiến trúc REST này được gọi là ứng dụng phát triển theo kiểu RESTful)

- Web API là một phiên bản kế tiếp của WCF, giúp chúng ta xây dựng lên các Service cung cấp dịch vụ cho các ứng dụng web, window…

- Trước web API chúng ta để có các service API chúng ta phải cấu hình, xây dựng các ứng dụng wcf, web service khá phức tạp.

- Một ứng dụng đơn giản như là: Chúng ta có 1 Web API cung cấp các dịch vụ lưu trữ dữu liệu, cung cấp các chỉ số chứng khoán, kết quả bóng đá, xổ số…

- Các ứng dụng Client như website, ứng dụng winform, wpf có thể kết nối vào Web API để lấy các dữ liệu về xử lý, cũng như cập nhật thông tin lại Web API.

- Web API dùng phương thức trao đổi dữ liệu là HTTP, kiểu dữ liệu trao đổi là JSON, một chuẩn dữ liệu hướng đối tượng được dùng khá nhiều trong việc lưu chuyển thông tin trên Internet.

- Do dùng JSon là kiểu dữ liệu chuyển đổi nên tốc độ các trang web sử dụng web API tương tác dữ liệu có tốc độ khá cao. Ngoài ra do Web API dùng giao thức HTTP nên hầu như tất cả các ứng dụng trên các công nghệ đều có thể kết nối tới để lấy cũng như tương tác với web API cụ thể như chúng ta có thể dùng các công nghệ web như: Asp.net ( MVC, Web Page, Web form), PHP,jsp hay các ứng dụng desktop như: winform, wpf đều có thể dễ dàng kết nối tới web API.

- Với Web API chúng ta có thể xây dựng và phân tách các ứng dụng web lớn. Cấu hình từng thành phần riêng biệt của website. Đâu là tầng data, đâu là tầng xử lý, đâu là tầng dịch vụ… Nền tảng của các ứng dụng lớn luôn là các service để các website thành viên có thể kết nối tương tác dữ liệu.

- Do đó với Web API chúng ta có thể ứng dụng vào các dự án Web ( cũng như window) lớn để phát triển trên nhiều tầng xử lý khác nhau.

- Dùng web API chúng ta dễ dàng xây dựng các ứng dụng window kiểu điện toán ( dữ liệu ở server) còn client chỉ cài giao diện .

- Hay có thể xây dựng các website Single Page Application (SPA) (tất cả web chỉ gói gọn trong 1 trang). Ứng dụng này tương tác khá cao với người dùng, tốc độ nhanh (do dùng ajax) thường được dùng làm các website tương tác với các thiết bị di động (các thiết bị di động thường có kết nối Internet chậm).

**1.3.Ưu điểm của Web API so với WCF REST Service**

- Có độ hoàn thiện cao, có thể host trong ứng dụng hoặc trênIIS, là kiến trúc lý tưởng cho các thiết bị có băng thông giới hạn như smartphone, tablet. Web API service sử dụng được ở hầu hết client như ứng dụng desktop, ứng dụng web, ứng dụng mobile.

- Web API trả về client định dạng dữ liệu có thể là JSON, XML hoặc định dạng khác

- Xây dựng các HTTP service rất đơn giản và nhanh chóng

- Mã nguồn mở (Open Source) và có thể được sử dụng bởi bất kỳ client nào hỗ trợ XML, JSON

- Hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content formats…

**CHƯƠNG 2: LẬP TRÌNH ANDROID**

**2.1. Tổng quan về hệ điều hành Android**

- Android là hệ điều hành mã nguồn mở, dựa trên Linux Kernel, dành cho các thiết bị di động nói chung (điện thoại, máy tính bảng, đồng hồ thông minh, máy nghe nhạc,…).

- Có nghĩa là Android không chỉ giới hạn trong phạm vi một hệ điều hành cho điện thoại! Nó có thể được nhà sản xuất cài đặt lên đồng hồ, máy nghe nhạc, thiết bị định vị GPS, thậm chí là ô tô (các thiết bị Android Auto).

- Android cũng không phải là một thiết bị hay sản phẩm cụ thể, nó là một hệ điều hành dựa trên Linux, nguồn mở, linh hoạt. Có khả năng tùy biến rất cao và có thể chạy trên nhiều thiết bị, nhiều kiến trúc vi xử lý (ARM / x86).

- Thiết bị Android được phân phối đang có tốc độ tăng trưởng nhanh hơn bất kỳ nền tảng di động khác, điều đó giúp Android trở thành một sự lựa chọn tuyệt vời trong hành trình phát triển ứng dụng trên điện thoại di động, đặc biệt là nếu bạn là một nhà phát triển Java. Một nhược điểm lớn của ứng dụng Android là sự phân mảnh, trong đó trình bày những thách thức đáng kể cho các ứng dụng thử nghiệm. Thực tế cho thấy là hầu hết các nhà phát triển ứng dụng gần như không thể kiểm tra một ứng dụng cho mọi thiết bị Android cho đến các phiên bản hệ điều hành. (Trong thực tế, làm như vậy theo một cách hiệu quả kinh tế chính là sự phát triển của ngành công nghiệp dịch vụ.)

- Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ Google, sau này được chính Google mua lại vào năm 2005 và hệ điều hành Android đã ra mắt vào năm 2007. Chiếc điện thoại đầu tiên chạy Android là HTC Dream được bán vào ngày 22 tháng 10 năm 2008.

- Chính mã nguồn mở cùng với giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên được điều chỉnh và phân phối Android một cách tự do. Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới. 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. Tháng 10 năm 2012, đã 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. Hiện nay con số này đã giảm xuống do sự ảnh hưởng lớn của iOS từ Apple và một phần nhỏ của Windows Phone, tuy nhiên Android vẫn dẫn đầu thị phần.

- Tính đến nay, Android đã có các phiên bản (kèm tên mã) lần lượt là:

* Android 1.5 Cupcake
* Android 1.6 Donut
* Android 2.1 Eclair
* Android 2.2 Froyo
* Android 2.3 Gingerbread
* Android 3.2 Honeycomb – Phiên bản Android đầu tiên được thiết kế cho máy tính bảng.
* Android 4.0 Ice Cream Sandwich
* Android 4.1 Jelly Bean
* Android 4.2 Jelly Bean
* Android 4.3 Jelly Bean
* Android 4.4 KitKat
* Android 5.0 Lollipop
* Android 6.0 Marshmallow
* Android 7.0 Nougat

- Về kiến trúc của hệ điều hành Android, chúng ta có thể xem qua sơ đồ sau:

Hình 2.1 Các tầng kiến trúc của hệ điều hành Android

- Các tầng của kiến trúc này như sau (từ trên xuống):

* Tầng Applications: Là tầng chứa các ứng dụng Danh bạ, Gọi điện, Trình duyệt, Nghe nhạc,… các ứng dụng này thường mua máy về chúng ta đã có sẵn rồi. Ứng dụng được viết sẽ được cài đặt vào lớp này.
* Tầng Framework: Là tầng chứa các API để làm việc với hệ điều hành như lấy thông tin danh bạ, quản lý các Activity, quản lý địa điểm, quản lý các View. Lớp Application Framework cung cấp nhiều dịch vụ cấp cao hơn cho các ứng dụng trong các lớp Java. Các lập trình viên cũng được phép sử dụng các dịch vụ này trong các ứng dụng của họ.
* Tầng Libraries: Chứa các thư viện, API gần như là cốt lõi của Android, bao gồm bộ quản lý bề mặt cảm ứng (Surface Manager), OpenGL (phục vụ cho việc dựng đồ họa phức tạp),… Libraries trong Android tập hợp các bộ thư viện mã nguồn mở WebKit, bộ thư viện nổi tiếng libc, cơ sở dữ liệu SQLite hữu ích cho việc lưu trữ và chia sẻ dữ liệu, bộ thư viện thể phát, ghi âm về âm thanh, hoặc video. Thư viện SSL chịu trách nhiệm cho bảo mật Internet.
* Tầng Android Runtime: Chứa các thư viện lõi của Android và máy ảo Dalvik Virtual Machine (từ Android 4 trở lên chúng ta có thêm máy ảo ART) , được thiết kế tối ưu cho Android. Máy ảo Dalvik VM sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ, đa luồng, mà thực chất là bên trong ngôn ngữ Java. Máy ảo Dalvik cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó. Android Runtime cũng cung cấp bộ thư viện cốt lõi, cho phép các lập trình viên Android sử dụng để viết các ứng dụng Android.
* Tầng Kernel: Là nhân lõi của hệ điều hành, chứa các tập lệnh, driver giao tiếp giữa phần cứng và phần mềm của Android. Linux Kernel là lớp thấp nhất. Nó cung cấp các chức năng cơ bản như quản lý tiến trình, quản lý bộ nhớ, quản lý thiết bị như: Camera, bàn phím, màn hình, … Ngoài ra, nó còn quản lý mạng, driver của các thiết bị, điều này gỡ bỏ sự khó khăn về giao tiếp với các thiết bị ngoại vi.

- Trong quá trình làm việc, chúng ta sẽ gần như chỉ làm việc với tầng xanh lam (Applications và Application Framework) và xanh lá (Libraries). Chương trình Android được viết bằng ngôn ngữ Java và được máy ảo DVM / ART trong mỗi thiết bị Android biên dịch ra mã máy.

**2.2. Giao diện**

**-** 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 dãn và thu lại để xử lý các đối tượng trên màn hình.

****

Hình 2.2 Giao diện Android 5.0 trên điện thoại

- 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 trên máy tính để bàn. Màn hính chính Android thường gồm nhiều biểu tượng (icon) và tiện ích (widget). 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 hay iOS. 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.

**2.3. Ưu và nhược điểm**

- Ưu điểm:

+ Điều làm cho Android được chú ý nhất chính vì nó là mã nguồn mở. Android đã thu hút nhiều nhà phát triển và từ tất cả các tầng lớp xã hội khi Goole phân phối miễn phí Android và các đoạn mã có bản quyền khác cho bất cứ ai muốn tham gia phát triển phần mềm cho Goole. Vì thế, chỉ trong một thời gian ngắn, các lập trình viên cũng như nhà phát triển có thể nhanh chóng đưa ra phiên bản Android được tùy chỉnh của riêng mình.

+ Đa dạng sản phẩm, rất nhiều hãng điện thoại, thiết bị công nghệ đã ưu ái chọn Android cho thiết bị của họ, giá cả thì hợp lý từ bình dân đến cao cấp.

+ Kho ứng dụng Google Play Store đồ sộ.

+ Thân thiện và dễ sử dụng.

+ Khả năng đa nhiệm, chạy cùng lúc nhiều ứng dụng cao.

- Nhược điểm:

+ Dễ nhiễm phần mềm độc hại và virus. Do tính chất mã nguồn mở, nhiều phần mềm không được kiểm soát có chất lượng không tốt hoặc lỗi bảo mật vẫn được sử dụng.

+ Kho ứng dụng quá nhiều dẫn đến khó kiểm soát chất lượng, thiếu các ứng dụng thật sự tốt.

+ Sự phân mảnh lớn. Trong khi một số thiết bị Android xuất sắc đã trình làng như Galaxy S8, Galaxy Note 7, Xperia XZs…, vẫn còn rất nhiều sản phẩm giá rẻ bình thường khác. Độ phân mảnh cũng gây khó khăn cho nhà phát triển ứng dụng. Họ phải xây dựng phiên bản cho từng kích cỡ màn hình cũng như phù hợp với phần cứng. Đôi khi, người ta phải viết lại ứng dụng để tương thích với một dòng điện thoại nhất định.

+ Từ khi Android chạy được trên hầu hết các điện thoại smartphone, nhiều mẫu điện thoại không hỗ trợ hết tiềm năng của nó. Đối với một số điện thoại bị giới hạn bởi phần cứng, máy có thể chạy chậm, bị treo và tự khởi động lại để giải phóng bộ nhớ.

+ Cập nhật không tự động với tất cả thiết bị. Khi một phiên bản hệ điều hành mới ra mắt, không phải tất cả sản phẩm đều được cập nhật, thậm chí nếu muốn trải nghiệm bạn thường xuyên phải mua mới thiết bị.

**2.4. Thiết lập môi trường**

- Hai thành phần cơ bản nhất mà chúng ta cần phải có để lập trình Android là bộ JDK (Java Development Kit) và Android SDK (Software Development Kit).

- JDK dùng để tạo ra môi trường thực thi máy ảo cho hệ điều hành mà chúng ta đang sử dụng.

- Android SDK chứa các phiên bản Android, các hàm API cần thiết, source code minh họa cũng như các công cụ hỗ trợ lập trình khác. Mỗi khi Google ra phiên bản Android mới thì Android SDK cũng được cập nhật tương ứng.

- Cài đặt Android Studio để tạo ứng dụng.

- Lưu ý khi chọn API:

+ Khi chọn Minium SDK, không nên chọn API quá mới vì sẽ giới hạn số lượng máy có thể chạy được ứng dụng của mình. Ví dụ như nếu bạn chọn Minimum SDK là API 23, thì những máy có API <23 sẽ không thể chạy được.

+ Không nên chọn API thấp quá vì sẽ thiếu thư viện hỗ trợ những API cao hơn. Ví dụ như nếu chọn Minimum SDK là 15 thì các máy có API >15 đều có thể chạy được, nhưng không khai thác được các thư viện hỗ trợ mà chỉ trong API >15 mới có.

+ Do đó, thì tốt nhất là nên chọn Minimum SDK theo API phổ biến, tức là có tỷ lệ % phù hợp mà Google khuyến cáo.

- Cài máy ảo: có thể dùng ADV Manager của Android Studio hoặc các máy ảo riêng biệt có thể hỗ trợ như Genymotion, Bluestack, …

**2.5. Fragmentation trong Android**

- Android là một thị trường bị phân mảnh với các phiên bản hệ điều hành khác nhau và các thiết bị. Nhiều phiên bản và các thiết bị ứng dụng hỗ trợ, các thử nghiệm nhiều hơn và bảo trì (và các chi phí liên quan).

- Các Gingerbread (được giới thiệu trong năm 2011) phiên bản của Android vẫn lh 20% của phân phối thiết bị Android. Thị trường, tuy nhiên, được củng cố trên Android 4.x và mới hơn (điều này đại diện cho ba phiên bản Android lớn cuối cùng). Điện thoại thông minh, máy tính bảng, và hybrid “phablets” mỗi người đều có kích thước và độ phân giải khác nhau. Cần có các tài nguyên thích hợp, phông chữ và bố trí đảm bảo rằng các kinh nghiệm tốt nhất có thể trên các đặc điểm như màn hình khác nhau.

- Cần phải xem xét các mảng của Android hỗ trợ cảm biến và / hoặc các cơ sở UI. Ví dụ, không phải tất cả các thiết bị Android có thể hỗ trợ cùng một loại máy ảnh, và họ có thể không hỗ trợ Bluetooth. Một ví dụ điển hình là Kindle, mà là dựa trên Android 2.3, nhưng nó không cung cấp hỗ trợ cho rất nhiều các cảm biến phần cứng hoặc các thiết bị giao diện người dùng tìm thấy trên các thiết bị Android khác.

**2.6. Các thành phần cơ bản của ứng dụng Android**

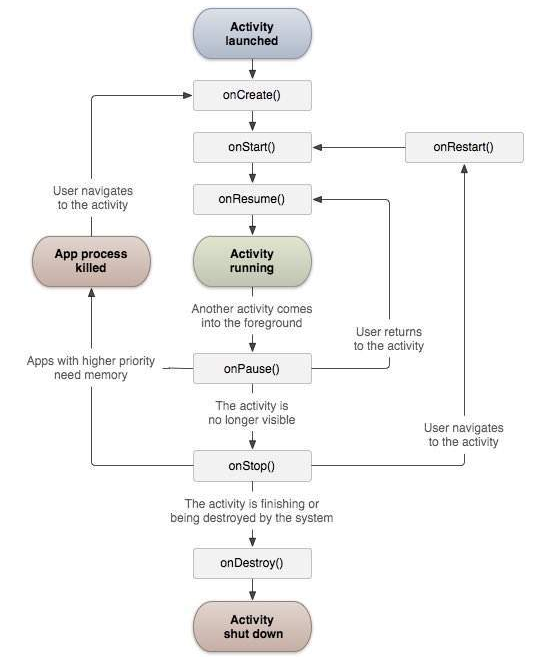
Các thành phần (component) cơ bản tạo nên một ứng dụng Android được chia thành 6 loại:

* Activity
* Service
* Broadcast Receivers
* Content Provider
* Intent
* Notification

**2.6.1. Activity**

- Hệ điều hành Android khởi tạo chương trình của nó bên trong một Activity bắt đầu với một lời gọi trên phương thức callback là onCreate() tương tự như hàm main() trong ngôn ngữ Java hoặc C/C++. Có một dãy các phương thức callback mà khởi động một Activity và một dãy phương thức callback khác sẽ hủy một Activity như sau trong sơ đồ vòng đời của Activity.

- Lớp Activity định nghĩa các callback như events, … Bạn không cần triển khai tất cả phương thức này. Tuy nhiên, điều quan trọng là bạn hiểu từng phương thức và triển khai chúng để sao cho ứng dụng của bạn vận hành theo cách người dùng mong đợi.



Hình 2.3 Sơ đồ vòng đời cùa Activity

- Một ứng dụng có thể có một hoặc nhiều Activity mà không có bất kỳ giới hạn nào. Mỗi Activity bạn định nghĩa cho ứng dụng của bạn phải được khai báo trong AndroidManifest.xml file và main activity cho ứng dụng phải được khai báo trong manifest với một <intent-filter> mà bao MAIN action và LAUNCHER category, như sau:

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

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

package="com.hq.demoviewpagerandtabhost">

<uses-permission android:name="android.permission.INTERNET"></uses-permission>

<application

android:allowBackup="true"

android:icon="@mipmap/clogo"

android:label="Chit Chat"

android:supportsRtl="true"

android:theme="@style/AppTheme"

android:name=".MSocket">

<activity android:name="com.hq.demoviewpagerandtabhost.SigninActivity">

<intent-filter>

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

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

</intent-filter>

</activity>

<activity android:name="com.hq.demoviewpagerandtabhost.MainActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.SignupActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.AboutmeActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.FramechatActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.AddActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.ChangeprofileActivity"></activity>

<activity android:name="com.hq.demoviewpagerandtabhost.ChangepassActivity"></activity>

</application>

</manifest>

- Nếu hoặc MAIN hoặc LAUNCHER không được khai báo, thì biểu tượng ứng dụng sẽ không xuất hiện trong danh sách ứng dụng của màn hình Home.

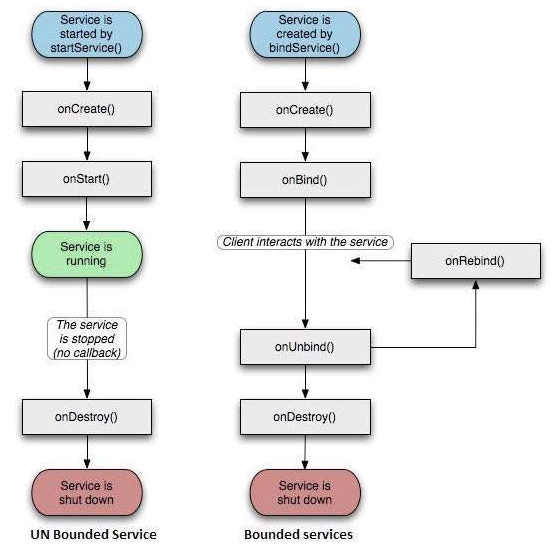
**2.6.2. Service**

**-** Một Service là một thành phần mà chạy trong Background để thực hiện các hoạt động mà không cần tương tác với người dùng và nó làm việc ngay cả khi nếu ứng dụng bị hủy. Về cơ bản, một Service nhận hai trạng thái:

* Started: Một Service được bắt đầu (started) khi một Component, chẳng hạn như activity, bắt đầu bằng việc gọi startService(). Khi đã được bắt đầu, một Service có thể chạy ngầm định trong Background, ngay cả khi thành phần mà bắt đầu nó đã bị hủy.
* Bound: Một Service được gắn kết (bound) khi một Component kết nối tới nó bởi việc gọi bindService(). Một bound service cung cấp một giao diện Client-Server cho phép các thành phần để tương tác với Service đó, gửi yêu cầu, nhận kết quả, và có thể thực hiện một số tiên trình khác với Interprocess Communication (IPC)

- Một Service có các phương thức callback mà bạn có thể triển khai để giám sát các thay đổi trong trạng thái của Service và bạn có thể thực hiện công việc trên các giai đoạn cụ thể. Hình 2.4 phía dưới cho ta thấy sơ đồ minh họa vòng đời của Service: bên trái là khi Service được tạo với phương thức startService() và bên phải là khi Service được tạo với bindService().

- Để tạo một Service, bạn tạo một lớp Java mà kế thừa lớp cơ sở Service hoặc một trong các lớp con đang tồn tại của nó. Lớp cơ sở Service định nghĩa các phương thức callback đa dạng và các phương thức quan trọng nhất được liệt kê dưới đây. Bạn không cần triển khai toàn bộ chúng. Điều quan trọng là bạn hiểu từng phương thức và cách triển khai chúng để đảm bảo ứng dụng của bạn vận hành như người dùng mong đợi.



Hình 2.4 Sơ đồ minh họa vòng đời service

**2.6.3. Broadcast Receivers**

- Broadcast Receiver phản hồi các thông báo phát ra từ các ứng dụng khác hoặc từ chính hệ thống. Những thông báo này đôi khi được gọi là các event hoặc intent. Ví dụ, các ứng dụng cũng có thể khởi tạo các tín hiệu broadcast để thông báo cho ứng dụng khác biết rằng một số dữ liệu đã được về tới thiết vị và là có sẵn cho chúng để sử dụng, vì thế Broadcast Receiver thông dịch thông tin đó và khởi tạo hành động thích hợp.

- Sau đây là hai bước quan trọng để làm Broadcast Receiver làm việc cho các Intent:

* Tạo Broadcast Receiver.
* Đăng ký Broadcast Receiver.

- Tạo Broadcast Receiver trong Android: Một Broadcast Receiver được triển khai như là một lớp con của lớp BroadcastReceiver và ghi đè phương thức onReceive(), nơi mà mỗi thông báo được nhận như là một tham số của đối tượng Intent.

public class MyReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

Toast.makeText(context, "Intent Detected.", Toast.LENGTH\_LONG).show();

}

}

- Đăng ký Broadcast Receiver trong Android: Một ứng dụng nghe các Intent được phát ra cụ thể bằng cách đăng ký một Broadcast Receiver trong AndroidManifest.xml file. Giả sử chúng ta đang đăng ký MyReceiver cho system event (sự kiện được tạo từ hệ thống) đã được tạo là ACTION\_BOOT\_COMPLETED, mà được kích hoạt bởi hệ thống một khi hệ điều hành Android đã hoàn thành tiến trình boot.

<application

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

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

android:theme="@style/AppTheme" >

<receiver android:name="MyReceiver">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED">

</action>

</intent-filter>

</receiver>

</application>

- Bây giờ, bất cứ khi nào thiết bị Android của bạn được boot, thì thông báo sẽ được nhận bởi MyReceiver và trình triển khai logic bên trong phương thức onReceive() sẽ được thực thi.

**2.6.4. Content Provider**

- Một thành phần Content Provider cung cấp dữ liệu từ một ứng dụng tới các ứng dụng khác theo yêu cầu. Những yêu cầu này được xử lý bởi các phương thức của lớp ContentResolver. Một Content Provider có thể sử dụng các cách khác nhau để lưu giữ dữ liệu và dữ liệu có thể được lưu giữ trong một cơ sở dữ liệu, một file, hoặc có thể thông qua một mạng.

- Đôi khi, việc chia sẽ dữ liệu qua các ứng dụng là thực sự cần thiết, khi đó Content Provider trở nên rất hữu ích.

- Content Provider giúp bạn tập trung nội dung trong một vị trí và có nhiều các ứng dụng khác nhau có thể truy cập nó khi cần thiết. Một Content Provider vận hành khá giống một cơ sở dữ liệu, nơi mà bạn có thể truy vấn nó, sửa đổi nội dung, cũng như thêm hoặc xóa nội dung bởi sử dụng các phương thức insert(), update(), delete() và query(). Trong hầu hết tình huống, dữ liệu này được lưu trữ trong một SQlite database.

-Một Content Provider được triển khai như là một lớp con của lớp ContentProvider và phải triển khai một tập APIs chuẩn để cho phép các ứng dụng thực hiện các transaction.

public class My Application extends ContentProvider {

}

-Content URI trong Android:

+Để truy vấn một Content Provider, bạn xác định chuỗi truy vấn dạng một URI có định dạng sau: <prefix>://<authority>/<data\_type>/<id>

|  |  |
| --- | --- |
| Phần | Mô tả |
| prefix | Luôn luôn được thiết lập là content:// |
| authority | Xác định tên của Content Provider, ví dụ contacts, browser ... Đối với Content Provider bên thứ ba, nó có thể là tên đầy đủ, ví như com.tutorialspoint.statusprovider |
| data\_type | Chỉ kiểu dữ liệu mà Provider cụ thể cung cấp. Ví dụ, nếu bạn đang lấy tất cả contact từ Content Provider có tên là Contacts, thì dữ liệu sẽ là people và URI sẽ như thế này content://contacts/people |
| id | Xác định bản ghi cụ thể được yêu cầu. Ví dụ, nếu bạn đang tìm số số liên hệ thứ 5 trong Coctacts thì URI sẽ như thế này content://contacts/people/5. |

- Để tạo một Content Provider, có 5 bước sau:

* Đầu tiên, bạn cần tạo một lớp Content Provider mà kế thừa từ lớp cơ sở ContentProviderbaseclass.
* Thứ hai, bạn cần định nghĩa địa chỉ URI của Content Provider, sẽ được sử dụng để truy cập nội dung đó.
* Tiếp theo, bạn sẽ cần tạo Database cho riêng bạn để giữ nội dung này. Thường thì, Android sử dụng SQLite Database và framework cần thiết để ghi đè phương thức onCreate() sẻ sử dụng phương thức SQLite Open Helper để tạo và mở Database của Provider này. Khi ứng dụng của bạn chạy, phương thức onCreate() được gọi trên mỗi luồng ứng dụng chính.
* Kế tiếp, bạn sẽ phải triển khai các truy vấn Content Provider để thực hiện các hoạt động cơ sở dữ liệu cụ thể.
* Cuối cùng, đăng ký Content Provider của bạn trong activity file bởi sử dụng thẻ <provider> .

- Sau đây là danh sách bạn cần ghi đè trong lớp Content Provider:

* onCreate() Phương thức này được gọi khi Provider được bắt đầu.
* query() Phương thức này nhận một yêu cầu từ người dùng. Kết quả được trả về dưới dạng một đối tượng Cursor.
* insert()Phương thức này chèn một bản ghi mới vào trong Content Provider.
* delete() Phương thức này xóa một bản ghi đang tồn tại từ Content Provider.
* update() Phương thức này cập nhật một bản ghi đang tồn tại trong Content Provider.
* getType() Phương thức này trả về kiểu MIME của dữ liệu tại URI đã cho.

**2.6.5. Intent**

-Một Intent trong Android là một miêu tả trừu tượng của một hoạt động để được thực hiện. Nó có thể được sử dụng với startActivity để chạy một Activity, broadcastIntent để gửi nó tới bất kỳ thành phần BroadcastReceiver nào quan tâm đến, và với startService(Intent)hoặc bindService(Intent, ServiceConnection, int) để giao tiếp với một Service ở Background.

-Intent chính nó, một đối tượng Intent, là một cấu trúc dữ liệu thụ động giữ một miêu tả trừu tượng của một hành động để được thực hiện.

-Ví dụ về mở một Activity khác và truyền dữ liệu qua nó:

Intent intent = new Intent(context, NextActivity.class);

intent.putExtra(key, value);

+Nếu đang ở trong 1 activity thì truyền vào this hoặc đầy đủ hơn thì là ThisActivity.this; còn nếu ở trong 1 Fragment thì truyền vào getActivity().

+key là 1 String để nhận diện giá trị này, và value nhận nhiều kiểu dữ liệu khác nhau như String, số nguyên, số thực…

**2.6.6.Notification**

-Một notification là một thông báo bạn có thể hiển thị tới người dùng. Thông báo này được hiển thị ở bên người giao diện UI thông thường của ứng dụng. Khi bạn nói cho hệ thống để thông báo một Notification, đầu tiên nó xuất hiện như một icon trong khu vực thông báo. Để thấy chi tiết về thông báo, người dùng mở Notification Drawer. Cả khu vực thông báo và drawer là các khu vực được điều khiển bởi hệ thống mà người dùng có thể quan sát bất cứ lúc nào.

-Tạo và gửi thông báo trong Android:

* Bước 1: Tạo Notification Builder:

+Trong bước đầu tiên, bạn cần tạo một Notification Builder bởi sử dụng phương thức NotificationCompat.Builder.build(). Bạn sẽ sử dụng Notification Builder để thiết lập các thuộc tính thông báo đa dạng như icon lớn, icon nhỏ, title, độ ưu tiên, …

NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)

* Bước 2: Thiết lập thuộc tính cho thông báo:

+Khi bạn có đối tượng Builder, bạn có thể thiết lập các thuộc tính thông báo của nó bởi sử dụng đối tượng Builder tùy theo yêu cầu của bạn. Nhưng bắt buộc ít nhất phải thiết lập như sau:

* Một icon nhỏ, được thiết lập bởi setSmallIcon()
* Một title, được thiết lập bởi setContentTitle()
* Text chi tiết, được thiết lập bởi setContentText()

mBuilder.setSmallIcon(R.drawable.notification\_icon);

mBuilder.setContentTitle("Notification Alert, Click Me!");

mBuilder.setContentText("Hi, This is Android Notification Detail!");

* Bước 3: Đính kèm các action:

+Đây là một bước tùy ý, và được thực hiện nếu bạn muốn gắn một action với thông báo. Một action cho phép người dùng trực tiếp từ một thông báo đi tới một Activity trong ứng dụng, ở đây họ có thể thấy một hoặc nhiều sự kiện hoặc thực hiện công việc khác.

+Action được định nghĩa bởi một PendingIntent chứa một Intent mà bắt đầu một Activity trong ứng dụng của bạn. Để liên kết PendingIntent đó với một động tác, gọi phương thức thích hợp của NotificationCompat.Builder. Ví dụ, nếu bạn muốn bắt đầu Activity khi người dùng click vào phần text của Notification Drawer, bạn thêm PendingIntent bằng việc gọi phương thức setContentIntent().

+Một đối tượng PendingIntent giúp bạn thực hiện một action trên ứng dụng, không quan tâm có hay không ứng dụng là đang chạy.

+Chúng ta sử dụng đối tượng StackBuilder mà chứa một backstack cho Activity đã bắt đầu. Điều này đảm bảo rằng điều hướng từ Activity về màn hình chính.

Intent resultIntent = new Intent(this, ResultActivity.class);

TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);

stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the stack

stackBuilder.addNextIntent(resultIntent);

PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0,PendingIntent.FLAG\_UPDATE\_CURRENT);

mBuilder.setContentIntent(resultPendingIntent);

* Bước 4: Thông báo Notification:

Cuối cùng, bạn truyền đối tượng Notification tới hệ thống bằng việc gọi NotificationManager.notify() để gửi thông báo. Đảm bảo rằng bạn đã gọi phương thức NotificationCompat.Builder.build() trên đối tượng Builder trước khi thông báo nó. Phương thức này kết hợp tất cả tùy chọn mà đã được thiết lập và trả về một đối tượng Notification mới .

NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

// notificationID allows you to update the notification later on.

mNotificationManager.notify(notificationID, mBuilder.build());

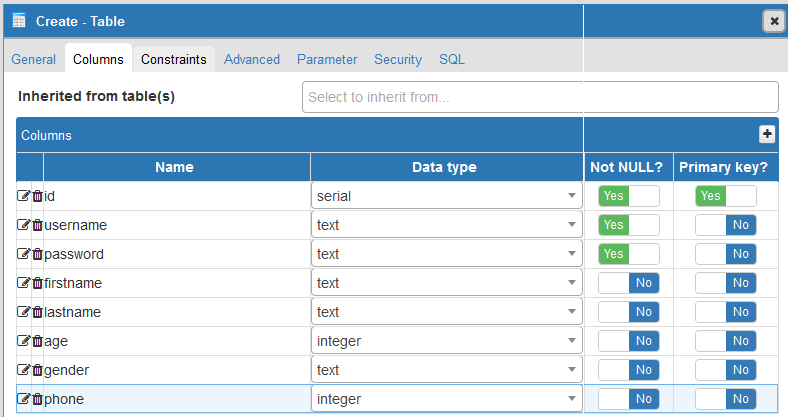
**CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ CƠ SỞ DỮ LIỆU**

Cơ sở dữ liệu: sử dụng chương trình pgAdmin4 tạo các bảng cơ sở dữ liệu như sau:

* Bảng User: chứa thông tin người dùng đăng kí gồm có username, password, firstname, lastname, age, gender, phone trong đó username và id là duy nhất trong toàn bộ chương trình và cơ sở dữ liệu.

User (id, username, password, firstname, lastname, age, gender, phone)

* id(serial): số id của người dùng(tự tăng) – khóa chính, không được trống.
* username(text): tên đăng nhập của người dùng – ràng buộc UNIQUE, không được trống.
* password(text): mật khẩu của người dùng, không được trống.
* firstname(text): họ của người dùng
* lastname(text): tên của người dùng
* age(integer): tuổi của người dùng
* gender(text): giới tính của người dùng
* phone(integer): số điện thoại của người dùng



Hình 3.1 Bảng các cột trong table User

CREATE TABLE public."User"

(

id serial NOT NULL,

username text NOT NULL,

password text NOT NULL,

firstname text,

lastname text,

age integer,

gender text,

phone integer,

PRIMARY KEY (id),

CONSTRAINT unique\_username UNIQUE (username)

)

WITH (

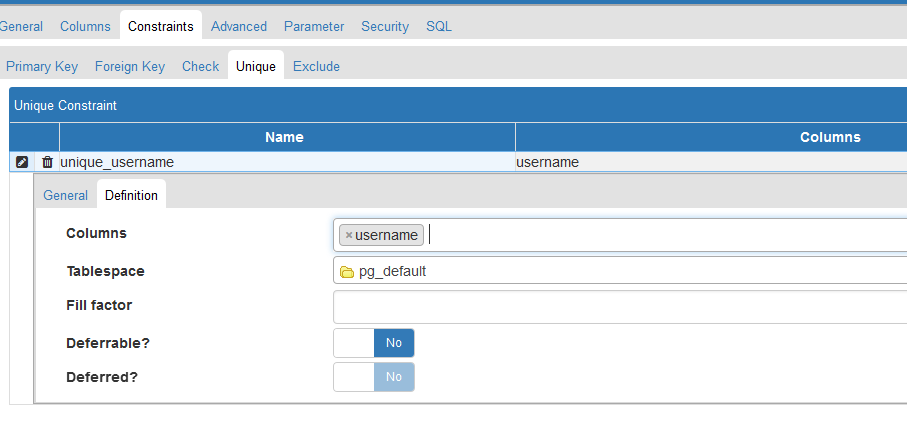
OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE public."User"

OWNER to postgres;

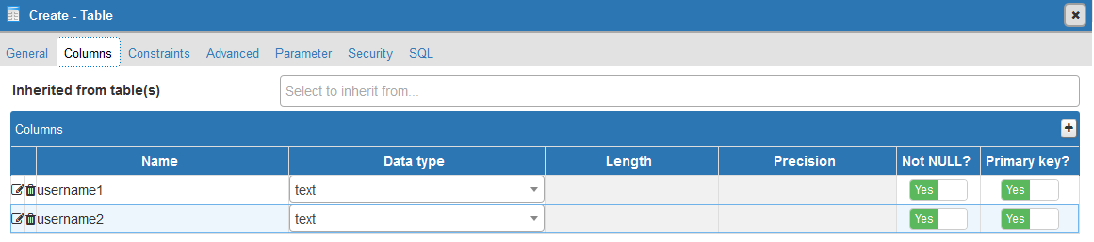


Hình 3.2 Ràng buộc UNIQUE của username trong table User

* Bảng Friends: thể hiện những username nào đang là bạn bè của nhau nếu chúng nằm cùng hàng trong đó username1 và username2 cùng là khóa chính

Friends (username1, username2)

* username1(text): username của người thứ nhất – khóa chính, không được trống.
* username2(text): username của người thứ hai – khóa chính, không được trống.



Hình 3.3 Bảng các cột trong table Friends

CREATE TABLE public."Friends"

(

username1 text NOT NULL,

username2 text NOT NULL,

PRIMARY KEY (username1, username2)

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE public."Friends"

OWNER to postgres;

* Bảng Offline chứa username người gửi và người nhận:

Offline (username1, username2, message)

* username1(text): username của người thứ nhất – khóa chính, không được trống.
* username2(text): username của người thứ hai – khóa chính, không được trống.
* message(text): tin nhắn của username1 gửi cho username2 hoặc username2 gửi cho username1 khi người kia không trực tuyến.

CREATE TABLE public."Offline"

(

username1 text NOT NULL,

username2 text NOT NULL,

message text,

CONSTRAINT unique\_username UNIQUE (username1)

)

WITH (

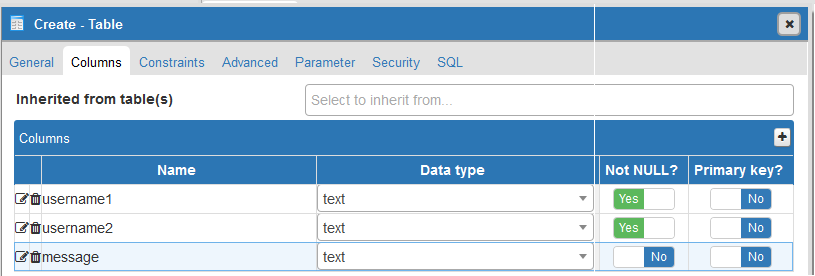
OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE public."Offline"

OWNER to postgres;



Hình 3.4 Bảng các cột trong table Offline

**CHƯƠNG 4: XÂY DỰNG WEB SERVICE**

Web service (Web Service) được coi là một công nghệ mang đến cuộc cách mạng trong cách thức hoạt động của các dịch vụ B2B (Business to Business) và B2C (Business to Customer). Giá trị cơ bản của Web service dựa trên việc cung cấp các phương thức theo chuẩn trong việc truy nhập đối với hệ thống đóng gói và hệ thống kế thừa.

Các phần mềm được viết bởi những ngôn ngữ lập trình khác nhau và chạy trên những nền tảng khác nhau có thể sử dụng Web service để chuyển đổi dữ liệu thông qua mạng Internet theo cách giao tiếp tương tự bên trong một máy tính. Tuy nhiên, công nghệ xây dựng Web service không nhất thiết phải là các công nghệ mới, nó có thể kết hợp với các công nghệ đã có như XML, SOAP, WSDL, UDDI… Với sự phát triển và lớn mạnh của Internet, Web service thật sự là một công nghệ đáng được quan tâm để giảm chi phí và độ phức tạp trong tích hợp và phát triển hệ thống. Chúng ta sẽ xem xét các Web service từ mức khái niệm đến cách thức xây dựng.

**4. 1. Giới thiệu về Web service**

Theo định nghĩa của W3C (World Wide Web Consortium), Web service là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng Internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML. Web service là tài nguyên phần mềm có thể xác định bằng địa chỉ URL, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu. Một Web service được tạo nên bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng nhìn thấy và có thể truy cập đến những dịch vụ mà nó thực hiện, đồng thời có thể yêu cầu thông tin từ Web service khác. Nó bao gồm các mô đun độc lập cho hoạt động của khách hàng và doanh nghiệp và bản thân nó được thực thi trên server.

Trước hết, có thể nói rằng ứng dụng cơ bản của Web service là tích hợp các hệ thống và là một trong những hoạt động chính khi phát triển hệ thống. Trong hệ thống này, các ứng dụng cần được tích hợp với cơ sở dữ liệu (CSDL) và các ứng dụng khác, người sử dụng sẽ giao tiếp với CSDL để tiến hành phân tích và lấy dữ liệu. Trong thời gian gần đây, việc phát triển mạnh mẽ của thương mại điện tử và B2B cũng đòi hỏi các hệ thống phải có khả năng tích hợp với CSDL của các đối tác kinh doanh (nghĩa là tương tác với hệ thống bên ngoài – bên cạnh tương tác với các thành phần bên trong của hệ thống trong doanh nghiệp).

Dưới đây, chúng ta sẽ xem qua những khái niệm và cách thức cơ bản nhất để xây dựng một Web service trong tích hợp và phát triển hệ thống.

**4. 2. Đặc điểm của Web service**

a. Đặc điểm

- Web service cho phép client và server tương tác được với nhau ngay cả trong những môi trường khác nhau. Ví dụ, đặt Web server cho ứng dụng trên một máy chủ chạy hệ điều hành Linux trong khi người dùng sử dụng máy tính chạy hệ điều hành Windows, ứng dụng vẫn có thể chạy và xử lý bình thường mà không cần thêm yêu cầu đặc biệt để tương thích giữa hai hệ điều hành này.

- Phần lớn kĩ thuật của Web service được xây dựng dựa trên mã nguồn mở và được phát triển từ các chuẩn đã được công nhận, ví dụ như XML.

- Một Web service bao gồm có nhiều mô-đun và có thể công bố lên mạng Internet.

- Là sự kết hợp của việc phát triển theo hướng từng thành phần với những lĩnh vực cụ thể và cơ sở hạ tầng Web, đưa ra những lợi ích cho cả doanh nghiệp, khách hàng, những nhà cung cấp khác và cả những cá nhân thông qua mạng Internet.

- Một ứng dụng khi được triển khai sẽ hoạt động theo mô hình client-server. Nó có thể được triển khai bởi một phần mềm ứng dụng phía server ví dụ như PHP, Oracle Application server hay Microsoft.Net…

- Ngày nay Web service đang rất phát triển, những lĩnh vực trong cuộc sống có thể áp dụng và tích hợp Web service là khá rộng lớn như dịch vụ chọn lọc và phân loại tin tức (hệ thống thư viện có kết nối đến web portal để tìm kiếm các thông tin cần thiết); ứng dụng cho các dịch vụ du lịch (cung cấp giá vé, thông tin về địa điểm…), các đại lý bán hàng qua mạng, thông tin thương mại như giá cả, tỷ giá hối đoái, đấu giá qua mạng…hay dịch vụ giao dịch trực tuyến (cho cả B2B và B2C) như đặt vé máy bay, thông tin thuê xe…

- Các ứng dụng có tích hợp Web service đã không còn là xa lạ, đặc biệt trong điều kiện thương mại điện tử đang bùng nổ và phát triển không ngừng cùng với sự lớn mạnh của Internet. Bất kì một lĩnh vực nào trong cuộc sống cũng có thể tích hợp với Web service, đây là cách thức kinh doanh và làm việc có hiệu quả bởi thời đại ngày nay là thời đại của truyền thông và trao đổi thông tin qua mạng. Do vậy, việc phát triển và tích hợp các ứng dụng với Web service đang được quan tâm phát triển là điều hoàn toàn dễ hiểu.

b. Ưu và nhược điểm

Ưu điểm:

+ Web service cung cấp khả năng hoạt động rộng lớn với các ứng dụng phần mềm khác nhau chạy trên những nền tảng khác nhau.

+ Sử dụng các giao thức và chuẩn mở. Giao thức và định dạng dữ liệu dựa trên văn bản (text), giúp các lập trình viên dễ dàng hiểu được.

+ Nâng cao khả năng tái sử dụng.

+ Thúc đẩy đầu tư các hệ thống phần mềm đã tồn tại bằng cách cho phép các tiến trình/chức năng nghiệp vụ đóng gói trong giao diện Web service.

+ Tạo mối quan hệ tương tác lẫn nhau và mềm dẻo giữa các thành phần trong hệ thống, dễ dàng cho việc phát triển các ứng dụng phân tán.

+ Thúc đẩy hệ thống tích hợp, giảm sự phức tạp của hệ thống, hạ giá thành hoạt động, phát triển hệ thống nhanh và tương tác hiệu quả với hệ thống của các doanh nghiệp khác.

Nhược điểm:

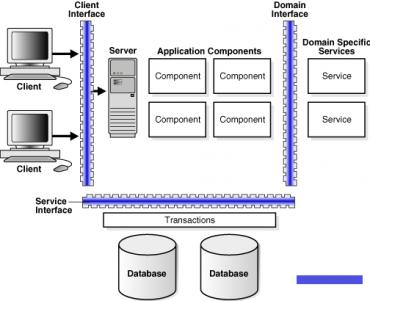
+ Những thiệt hại lớn sẽ xảy ra vào khoảng thời gian chết của Web service, giao diện không thay đổi, có thể lỗi nếu một máy khách không được nâng cấp, thiếu các giao thức cho việc vận hành.

+ Có quá nhiều chuẩn cho Web service khiến người dùng khó nắm bắt.

+ Phải quan tâm nhiều hơn đến vấn đề an toàn và bảo mật.

**4. 3. Kiến trúc của Web service**

Web service gồm có 3 chuẩn chính: SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) và UDDI (Universal Description, Discovery, and Integration). Hình 4.1 mô tả chồng giao thức của Web service, trong đó UDDI được sử dụng để đăng ký và khám phá Web service đã được miêu tả cụ thể trong WSDL. Giao tác UDDI sử dụng SOAP để nói chuyện với UDDI server, sau đó các ứng dụng SOAP yêu cầu một Web service. Các thông điệp SOAP được gửi đi chính xác bởi HTTP và TCP/IP.



Hình 4.1 Chồng giao thức của Web service

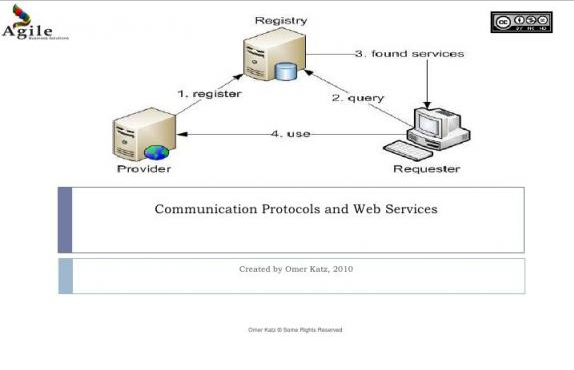
Chồng giao thức Web service là tập hợp các giao thức mạng máy tính được sử dụng để định nghĩa, xác định vị trí, thi hành và tạo nên Web service tương tác với những ứng dụng hay dịch vụ khác. Chồng giao thức này có 4 thành phần chính:

- Dịch vụ vận chuyển (Service Transport): có nhiệm vụ truyền thông điệp giữa các ứng dụng mạng, bao gồm những giao thức như HTTP, SMTP, FTP, JSM và gần đây nhất là giao thức thay đổi khổi mở rộng (Blocks Extensible Exchange Protocol- BEEP).

- Thông điệp XML: có nhiệm vụ giải mã các thông điệp theo định dạng XML để có thể hiểu được ở mức ứng dụng tương tác với người dùng. Hiện tại, những giao thức thực hiện nhiệm vụ này là XML-RPC, SOAP và REST.

- Mô tả dịch vụ: được sử dụng để miêu tả các giao diện chung cho một Web service cụ thể. WSDL thường được sử dụng cho mục đích này, nó là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. Web service sẽ sử dụng ngôn ngữ này để truyền tham số và các loại dữ liệu cho các thao tác và chức năng mà Web service cung cấp.

- Khám phá dịch vụ: tập trung dịch vụ vào trong một nơi được đăng ký, từ đó giúp một Web service có thể dễ dàng khám phá ra những dịch vụ nào đã có trên mạng, tốt hơn trong việc tìm kiếm những dịch vụ khác để tương tác. Một Web service cũng phải tiến hành đăng ký để các dịch vụ khác có thể truy cập và giao tiếp. Hiện tại, UDDI API thường được sử dụng để thực hiện công việc này.



Hình 4.2 Tầng giao thức tương tác dịch vụ

Trong đó, tầng giao thức tương tác dịch vụ (Service Communication Protocol) với công nghệ chuẩn là SOAP. SOAP là giao thức nằm giữa tầng vận chuyển và tầng mô tả thông tin về dịch vụ, cho phép người dùng triệu gọi một dịch vụ từ xa thông qua một thông điệp XML. Ngoài ra, để các dịch vụ có tính an toàn, toàn vẹn và bảo mật thông tin, trong kiến trúc Web service, chúng ta có thêm các tầng Policy, Security, Transaction, Management.

**4. 4. Các thành phần của Web service**

a. XML – eXtensible Markup Language

Là một chuẩn mở do W3C đưa ra cho cách thức mô tả dữ liệu, nó được sử dụng để định nghĩa các thành phần dữ liệu trên trang web và cho những tài liệu B2B. Về hình thức, XML hoàn toàn có cấu trúc thẻ giống như ngôn ngữ HTML nhưng HTML định nghĩa thành phần được hiển thị như thế nào thì XML lại định nghĩa những thành phần đó chứa cái gì. Với XML, các thẻ có thể được lập trình viên tự tạo ra trên mỗi trang web và được chọn là định dạng thông điệp chuẩn bởi tính phổ biến và hiệu quả mã nguồn mở.

Do Web service là sự kết hợp của nhiều thành phần khác nhau nên nó sử dụng các tính năng và đặc trưng của các thành phần đó để giao tiếp. XML là công cụ chính để giải quyết vấn đề này và là kiến trúc nền tảng cho việc xây dựng một Web service, tất cả dữ liệu sẽ được chuyển sang định dạng thẻ XML. Khi đó, các thông tin mã hóa sẽ hoàn toàn phù hợp với các thông tin theo chuẩn của SOAP hoặc XML-RPC và có thể tương tác với nhau trong một thể thống nhất.

b. WSDL - Web Service Description Language

WSDL định nghĩa cách mô tả Web service theo cú pháp tổng quát của XML, bao gồm các thông tin:

- Tên dịch vụ

- Giao thức và kiểu mã hóa sẽ được sử dụng khi gọi các hàm của Web service

- Loại thông tin: thao tác, tham số, những kiểu dữ liệu (có thể là giao diện của Web service cộng với tên cho giao diện này).

Một WSDL hợp lệ gồm hai phần: phần giao diện (mô tả giao diện và phương thức kết nối) và phần thi hành mô tả thông tin truy xuất CSDL. Cả hai phần này sẽ được lưu trong 2 tập tin XML tương ứng là tập tin giao diện dịch vụ và tập tin thi hành dịch vụ. Giao diện của một Web service được miêu tả trong phần này đưa ra cách thức làm thế nào để giao tiếp qua Web service. Tên, giao thức liên kết và định dạng thông điệp yêu cầu để tương tác với Web service được đưa vào thư mục của WSDL.

WSDL thường được sử dụng kết hợp với XML schema và SOAP để cung cấp Web service qua Internet. Một client khi kết nối tới Web service có thể đọc WSDL để xác định những chức năng sẵn có trên server. Sau đó, client có thể sử dụng SOAP để lấy ra chức năng chính xác có trong WSDL.

c. Universal Description, Discovery, and Integration (UDDI)

Để có thể sử dụng các dịch vụ, trước tiên client phải tìm dịch vụ, ghi nhận thông tin về cách sử dụng và biết được đối tượng nào cung cấp dịch vụ. UDDI định nghĩa một số thành phần cho biết các thông tin này, cho phép các client truy tìm và nhận những thông tin được yêu cầu khi sử dụng Web service.

- Cấu trúc UDDI :

+ Trang trắng – White pages: chứa thông tin liên hệ và các định dạng chính yếu của Web service, chẳng hạn tên giao dịch, địa chỉ, thông tin nhận dạng… Những thông tin này cho phép các đối tượng khác xác định được dịch vụ.

+ Trang vàng – Yellow pages: chứa thông tin mô tả Web service theo những loại khác nhau. Những thông tin này cho phép các đối tượng thấy được Web service theo từng loại với nó.

+ Trang xanh – Green pages: chứa thông tin kỹ thuật mô tả các hành vi và các chức năng của Web service.

+ Loại dịch vụ – tModel: chứa các thông tin về loại dịch vụ được sử dụng.

Những thông tin về Web service được sử dụng và công bố lên mạng sử dụng giao thức này. Nó sẽ kích hoạt các ứng dụng để tìm kiếm thông tin của Web service khác nhằm xác định xem dịch vụ nào sẽ cần đến nó.

d. SOAP – Simple Object Access Protocol

Chúng ta đã hiểu cơ bản Web service như thế nào nhưng vẫn còn một vấn đề khá quan trọng. Đó là làm thế nào để truy xuất dịch vụ khi đã tìm thấy? Câu trả lời là các Web service có thể truy xuất bằng một giao thức là Simple Object Access Protocol – SOAP. Nói cách khác chúng ta có thể truy xuất đến UDDI registry bằng các lệnh gọi hoàn toàn theo định dạng của SOAP.

SOAP là một giao thức giao tiếp có cấu trúc như XML. Nó được xem là cấu trúc xương sống của các ứng dụng phân tán được xây dựng từ nhiều ngôn ngữ và các hệ điều hành khác nhau. SOAP là giao thức thay đổi các thông điệp dựa trên XML qua mạng máy tính, thông thường sử dụng giao thức HTTP.

Một client sẽ gửi thông điệp yêu cầu tới server và ngay lập tức server sẽ gửi những thông điệp trả lời tới client. Cả SMTP và HTTP đều là những giao thức ở lớp ứng dụng của SOAP nhưng HTTP được sử dụng và chấp nhận rộng rãi hơn bởi ngày nay nó có thể làm việc rất tốt với cơ sở hạ tầng Internet.

Cấu trúc một thông điệp theo dạng SOAP

Thông điệp theo định dạng SOAP là một văn bản XML bình thường bao gồm các phần tử sau:

- Phần tử gốc – envelop: phần tử bao trùm nội dung thông điệp, khai báo văn bản XML như là một thông điệp SOAP.

- Phần tử đầu trang – header: chứa các thông tin tiêu đề cho trang, phần tử này không bắt buộc khai báo trong văn bản. Header còn có thể mang những dữ liệu chứng thực, những chứ ký số, thông tin mã hóa hay cài đặt cho các giao dịch khác.

- Phần tử khai báo nội dung chính trong thông điệp – body, chứa các thông tin yêu cầu và thông tin được phản hồi.

- Phần tử đưa ra các thông tin về lỗi -fault, cung cấp thông tin lỗi xảy ra trong qúa trình xử lý thông điệp.

Một SOAP đơn giản trong body sẽ lưu các thông tin về tên thông điệp, tham chiếu tới một thể hiện của dịch vụ, một hoặc nhiều tham số. Có 3 kiểu thông báo sẽ được đưa ra khi truyền thông tin: request message(tham số gọi thực thi một thông điệp), respond message (các tham số trả về, được sử dụng khi yêu cầu được đáp ứng) và cuối cùng là fault message (thông báo tình trạng lỗi).

Kiểu truyền thông: Có 2 kiểu truyền thông

- Remote procedure call (RPC): cho phép gọi hàm hoặc thủ tục qua mạng. Kiểu này được khai thác bởi nhiều Web service.

- Document: được biết đến như kiểu hướng thông điệp, nó cung cấp giao tiếp ở mức trừu tượng thấp, khó hiểu và yêu cầu lập trình viên mất công sức hơn.

Hai kiểu truyền thông này cung cấp các định dạng thông điệp, tham số, lời gọi đến các API khác nhau nên việc sử dụng chúng tùy thuộc vào thời gian và sự phù hợp với Web service cần xây dựng.

Cấu trúc dữ liệu: Cung cấp những định dạng và khái niệm cơ bản giống như trong các ngôn ngữ lập trình khác như kiểu dữ liệu (int, string, date…) hay những kiều phức tạp hơn như struct, array, vector… Định nghĩa cấu trúc dữ liệu SOAP được đặt trong namespace SOAP-ENC.

Mã hóa: Giả sử service rquester và service provider được phát triển trong Java, khi đó mã hóa SOAP là làm thế nào chuyển đổi từ cấu trúc dữ liệu Java sang SOAP XML và ngược lại, bởi vì định dạng cho Web Service chính là XML. Bất kỳ một môi trường thực thi SOAP nào cũng phải có một bảng chứa thông tin ánh xạ nhằm chuyển đổi từ ngôn ngữ Java sang XML và từ XML sang Java – bảng đó được gọi là SOAPMappingRegistry. Nếu một kiểu dữ liệu được sử dụng dưới một dạng mã hóa thì sẽ có một ánh xạ tồn tại trong bộ đăng ký của môi trường thực thi SOAP đó.

**4. 5. An toàn cho Web service**

Web service liên kết và tương tác với các ứng dụng qua Internet, chính vì vậy bảo mật là một vấn đề được quan tâm khi các công ty tiến tới kết hợp ứng dụng với một Web service. Việc đảm bảo an toàn cho Web service là một vấn đề quan trọng, đặc biệt đối với những dịch vụ liên quan đến trao đổi tiền tệ, thông tin từ thị trường chứng khoán hay dịch vụ bán hàng qua mạng (liên quan đến trả tiền bằng tài khoản và có yêu cầu thông tin cá nhân của người dùng).

Trước khi có WS-Security (bảo mật cho Web service) thì ý nghĩa thông thường của an toàn Web service là bảo mật kênh truyền dữ liệu. Hiện nay, nó được thực hiện cho những SOAP/HTTP dựa trên cơ chế truyền thông điệp bằng cách sử dụng giao thức HTTPS. Không chỉ là an toàn ở mức truyền thông điệp, HTTPS còn cung cấp sự an toàn tới toàn bộ gói dữ liệu HTTP.

Mặc dù HTTPS không bao gồm tất cả các khía cạnh trong chuẩn an toàn chung cho Web service nhưng nó đã cung cấp một lớp bảo mật khá đầy đủ với định danh, chứng thực, tính toàn vẹn thông điệp hay độ tin cậy.

- Đảm bảo an toàn cho Web service:

Khái niệm về WS-Security: đây là một chuẩn an toàn bao trùm cho SOAP, nó được dùng khi muốn xây dựng những Web service toàn vẹn và tin cậy. Toàn vẹn có nghĩa là khi có một giao dịch hay khi truyền thông tin, hệ thống và thông tin sẽ không bị chặn, giao dịch sẽ không bị mất cũng như không thể có người lấy cắp được dữ liệu trên đường truyền. WS-security được thiết kế mang tính mở nhằm hướng tới những mô hình an toàn khác bao gồm PKI, Kerberos và SSL. Nó cũng đưa ra nhiều hỗ trợ cho các cơ chế an toàn khác, nhiều khuôn dạng chữ ký và công nghệ mã hóa, đảm bảo sự an toàn, toàn vẹn thông điệp và tính tin cậy của thông điệp. Tuy nhiên, WS-security cũng chưa thể đảm bảo được tất cả yêu cầu về bảo mật và an toàn thông tin, nó chỉ là một trong những lớp của giải pháp an toàn cho Web service.

Tính toàn vẹn tạo ra một chữ ký số hóa XML dựa trên nội dung của thông điệp. Nếu dữ liệu bị thay đổi bất hợp pháp, nó sẽ không còn thích hợp với chữ ký số hóa XML đó. Chữ ký này được tạo ra dựa trên khóa mà người gửi thông điệp tạo ra, do đó người nhận chỉ nhận thông điệp khi có chữ ký sử dụng và nội dung phù hợp. Ngược lại sẽ có một thông báo lỗi. Việc chứng thực được thực hiện giữa client và server là cách chứng thực rất cơ bản (sử dụng định danh người dùng và mật khẩu).

WS-security chỉ là một trong những lớp an toàn và bảo mật cho Web service, vì vậy cần một mô hình an toàn chung lớn hơn để có thể bao quát được các khía cạnh khác. Các thành phần được thêm có thể là WS-Secure Conversation Describes,WS-Authentication Describes,WS-Policy Describes hay WS-Trust Describes. Chúng sẽ thực hiện việc đảm bảo an toàn hơn cho hệ thống khi trao đổi dữ liệu, mở và đóng các phiên làm việc cũng như quản lý dữ liệu cần chứng thực và chính sách chứng thực.

**4. 6. Xây dựng một dịch vụ Web**

ó 4 giai đoạn chính để xây dựng một dịch vụ Web là xây dựng, triển khai, tiến hành và quản lý, trong đó:

- Giai đoạn xây dựng bao gồm phát triển và chạy thử ứng dụng dịch vụ Web, xây dựng các chức năng và định nghĩa dịch vụ. Có hai cách khác nhau để tiến hành trong giai đoạn này, đó là Red-path- solod và Blue-path-dashed. Với Red- path-solod, chúng ta sẽ xây dựng một dịch vụ Web mới từ trạng thái ban đầu hoặc với một dịch vụ đã có sẵn. Từ đó, xây dựng định nghĩa service (WSDL) với các đối tượng, hàm chức năng mà chúng ta mong muốn. Nếu theo cách Blue-path-dashed, dịch vụ Web sẽ được xây dựng từ đầu hoặc từ một định nghĩa dịch vụ WSDL. Sử dụng WSDL này, xây dựng hoặc sửa đổi lại mã để thực hiện các yêu cầu mong muốn trong dịch vụ Web.

- Giai đoạn triển khai: công bố định nghĩa dịch vụ, xây dựng WSDL và triển khai mã thực thi của dịch vụ Web. Triển khai dịch vụ Web tới một ứng dụng phía server, sau đó sẽ công bố dịch vụ Web trên mạng Internet để các client có thể nhìn thấy. Sử dụng UDDI registry để công bố lên mạng.

- Giai đoạn tiến hành: tìm kiếm và gọi thực thi dịch vụ Web bởi những người dùng muốn sử dụng dịch vụ.

- Quản lý: Quản lý và quản trị dịch vụ, duy trì sự ổn định của dịch vụ, cập nhật thông tin mới, sửa lỗi khi nó xảy ra…

Để xây dựng một dịch vụ Web, chúng ta cần hiểu được những việc phải làm và nên bắt đầu từ đâu. Có 3 cách tiếp cận chủ yếu để xây dựng nên một dịch vụ Web, có thể từ một ứng dụng đã có (bottom-up); từ một định nghĩa dịch vụ, WSDL để phát sinh một ứng dụng mới (top-down) hoặc có thể từ một nhóm các dịch vụ Web hiện có, kết hợp lại với nhau để tạo nên các chức năng mới hoặc mở rộng thêm chức năng. Những hướng tiếp cận này dựa trên những gì mà chúng ta đã có, tùy thuộc vào yêu cầu của hệ thống, trong đó tối đa việc sử dụng lại các chức năng, các thành phần, môđun đã được xây dựng.

Qui trình xây dựng một dịch vụ Web bao gồm các bước sau:

1. Định nghĩa và xây dựng các chức năng, các dịch vụ mà dịch vụ sẽ cung cấp (sử dụng ngôn ngữ Java chẳng hạn).

2. Tạo WSDL cho dịch vụ

3. Xây dựng SOAP server

4. Đăng ký WSDL với UDDI registry để cho phép các client có thể tìm thấy và truy xuất.

5. Client nhận file WSDL và từ đó xây dựng SOAP client để có thể kết nối với SOAP server

6. Xây dựng ứng dụng phía client (chẳng hạn sử dụng Java) và sau đó gọi thực hiện dịch vụ thông qua việc kết nối tới SOAP server.

Lựa chọn một ngôn ngữ, xây dựng các tiến trình nghiệp vụ và chúng ta bắt đầu tạo nên một dịch vụ Web như ý muốn. Sau đó là cung cấp dịch vụ Web này trên Internet.

**4. 7. Tích hợp dịch vụ Web theo chuẩn**

Để có thể thành công với dịch vụ Web chúng ta phải quan tâm đến khá nhiều vấn đề, bao gồm việc triển khai, giám sát và tích hợp hệ thống. Doanh nghiệp không những phải phát triển một ứng dụng dịch vụ Web mới mà còn phải tích hợp các ứng dụng nghiệp vụ phụ trợ của họ trong kiến trúc Dịch vụ Web. Cùng với việc triển khai và tích hợp, những nhà kinh doanh và những người sử dụng kỹ thuật cũng cần có khả năng giám sát, triển khai toàn diện để đảm bảo hoạt động kinh doanh hiệu quả và tin cậy.

- Giám sát (monitoring): Cần hỗ trợ ở cả mức công cụ và cơ sở hạ tầng để giám sát các dịch vụ Web chạy như thế nào qua toàn bộ mạng, từ một chi nhánh con của một công ty trên mạng tới các chi nhánh khác trong công ty hay giao tiếp với doanh nghiệp khác. Kết hợp thông báo theo sự kiện với các lỗi trong luồng nghiệp vụ cho những người dùng không có kinh nghiệm giám sát dịch vụ Web và các dịch vụ kế thừa khác.

- Xác định đường đi dữ liệu (Data routing): Việc thiết lập đường đi của dữ liệu giữa những thành phần của dịch vụ Web hướng tới tối đa hóa khả năng sử dụng lại. Nếu coi một thành phần (component) là một đối tượng thì mỗi thể hiện (instance) của nó sẽ không quan tâm đến các thể hiện khác của cùng thành phần đó. Những thể hiện của cùng một thành phần có thể dễ dàng được sử dụng lại trong các ứng dụng phân tán khác bởi vì chúng hoàn toàn độc lập và không phụ thuộc lẫn nhau.

- Triển khai (Deployment): Triển khai các dịch vụ Web có khả năng nâng cấp, điều khiển và cấu hình các thành phần từ xa thông qua mạng phân tán.

- Quản lý (Management): Có thể xây dựng theo kiến trúc P2P (Peer-to-Peer). Các hoạt động chính như thực thi các thành phần, định tuyến dữ liệu, xử lý luồng công việc và chuyển đổi dữ liệu được thực hiện tại các điểm cuối của mạng. Server sẽ tập trung giải quyết các hoạt động khác như quản lý, điều khiển sự kiện, chứng thực bảo mật và quản trị.

- Cấu hình và quản lý phiên bản (Configuration and version management): Sử dụng các công cụ linh hoạt để quản lý các phiên bản khác nhau của dịch vụ Web, cho phép các phiên bản được nâng cấp và điều khiển từ một công cụ quản lý tập trung. Kết hợp giữa ứng dụng và mạng giúp các kỹ sư triển khai có thể điều khiển các thành phần chạy trên nền tảng hệ thống phần cứng cụ thể bên trong mạng.

- Bảo mật (Security): các chuẩn mở như HTTP, XML, SOAP, WSDL và chuẩn bảo mật JSM được sử dụng rộng rãi khiến chúng trở thành lý tưởng để xây dựng các ứng dụng web. Đầu tiên, dịch vụ Web sử dụng những công nghệ này giống như firewall, SSL và các chứng nhận số. Dịch vụ Web thế hệ sau này sẽ kết hợp với những công nghệ có khả năng bảo mật cao hơn, giống như mã hóa XML và chứng nhận số XML.

Như vậy, với một dịch vụ Web, việc giao tiếp và truyền nhận dữ liệu trở nên dễ dàng và hiệu quả hơn, đồng thời đem lại chi phí thấp hơn và tăng cường những khả năng giao tiếp thời gian thực, kết nối với mọi người trên khắp thế giới. Bản chất của nền tảng công nghệ này là kiến trúc hướng dịch vụ và sự phát triển của dịch vụ Web có tương lai rất khả quan.

**CHƯƠNG 5: XÂY DỰNG ỨNG DỤNG ANDROID VÀ CÀI ĐẶT CƠ SỞ DỮ LIỆU**

**5.1. Phân tích yêu cầu**

Tên đề tài: Xây dựng ứng dụng Chat trên Android.

Chức năng của ứng dụng:

- Tạo tài khoản mới

- Đăng nhập

- Xem thông tin cá nhân gồm username, firstname, lastname, age, gender, phone

- Thêm bạn

- Nhắn tin online cũng như offline với một tài khoản nào đó

- Xóa bạn

- Xem danh sách bạn đang online

**5.2. Lược đồ Use case**

Người dùng có các chức năng chính như sau:

**-** Log in: đăng nhập

- Chat: nhắn tin bằng tin nhắn văn bản

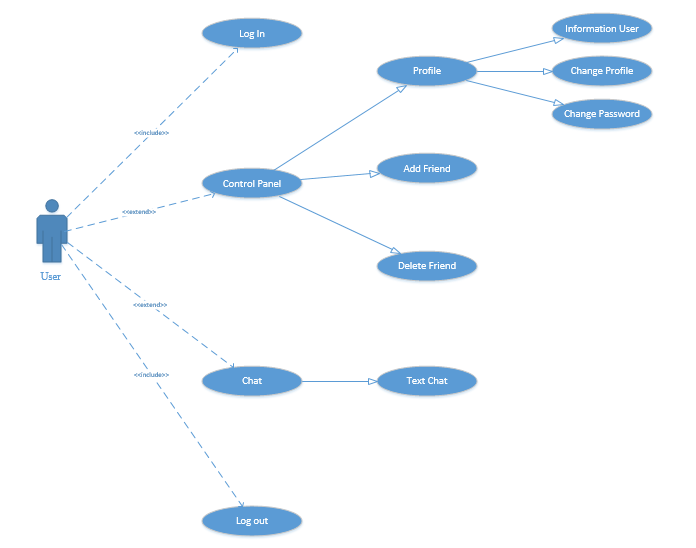
- Log out: đăng xuất

- Control Panel: các tương tác như sau:

+ Profile: bao gồm xem thông tin cá nhân, thay đổi mật khẩu, thay đổi các thông tin khác.

+ Add friend: thêm bạn mới.

+ Delete friend: xóa 1 bạn trong danh bạ



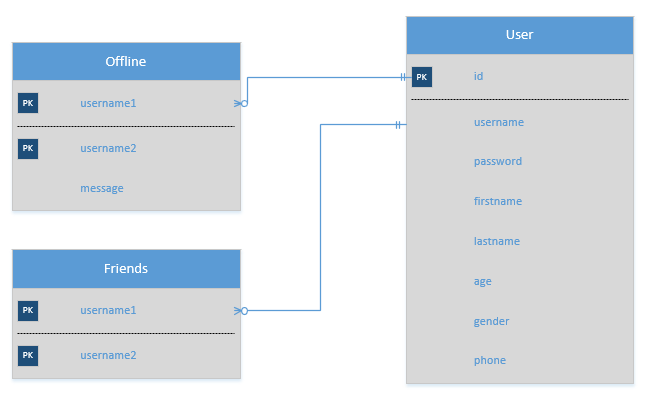
Hình 5.1 Use case của User

**5.3. Sơ đồ lớp diagram**

- Bảng User gồm: id, username, password, firstname, lastname, age, gender, phone.

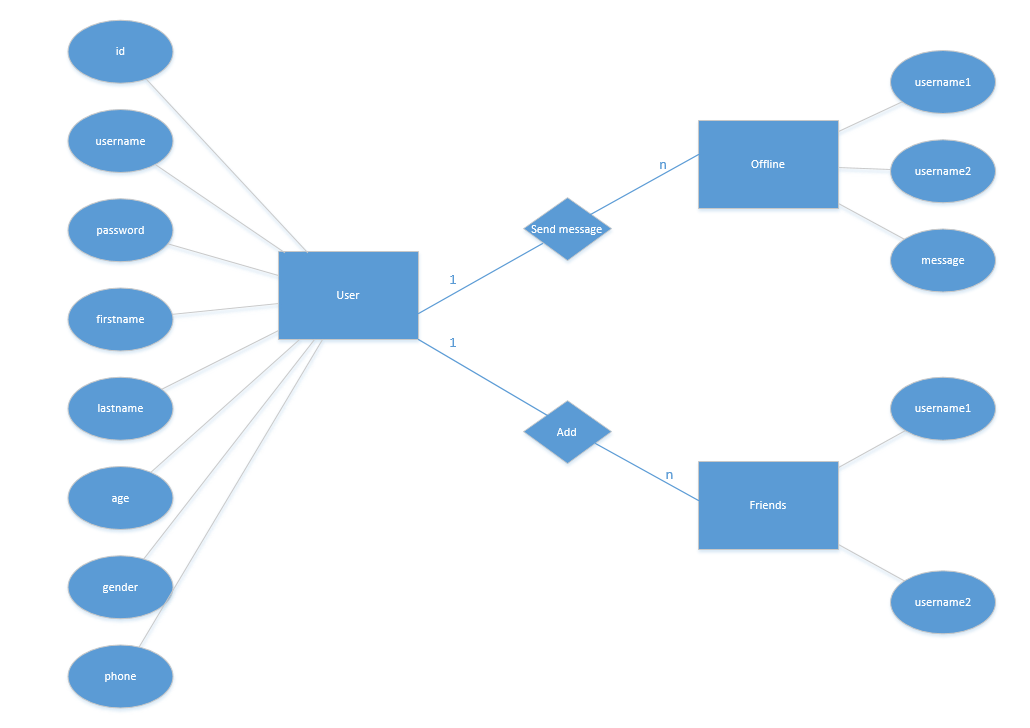
- Bảng Friends gồm username1 và username2 lấy từ username đã có trong bảng User.

- Bảng Offline gồm username1, username2, message trong đó username1 và username2 lấy từ username đã có trong bảng User.



Hình 5.2 Sơ đồ lớp diagram

**5.4. Mô hình thực thể ERD**



Hình 5.3 Mô hình thực thể ERD

- User gồm: id, username, password, firstname, lastname, age, gender, phone.

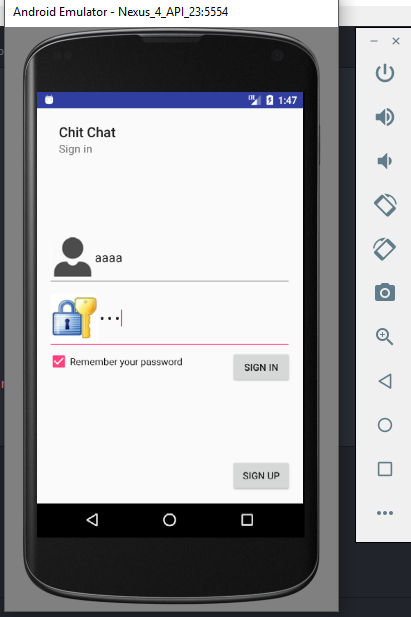
- Một user có thể thêm nhiều bạn bè

- Một user có thể gửi nhiều tin nhắn offline

**CHƯƠNG 6: KẾT QUẢ ĐẠT ĐƯỢC**

**6.1. Client Android**

**6.1.1. Giao diện đăng nhập**



Hình 6.1 Giao diện đăng nhập trên máy ảo

Gồm 2 edittext, 1 checkbox và 1 button:

- Edittext thứ nhất: điền username (tên đăng nhập) đã đăng kí

- Edittext thứ hai: điền password (mật khẩu) đã đăng kí

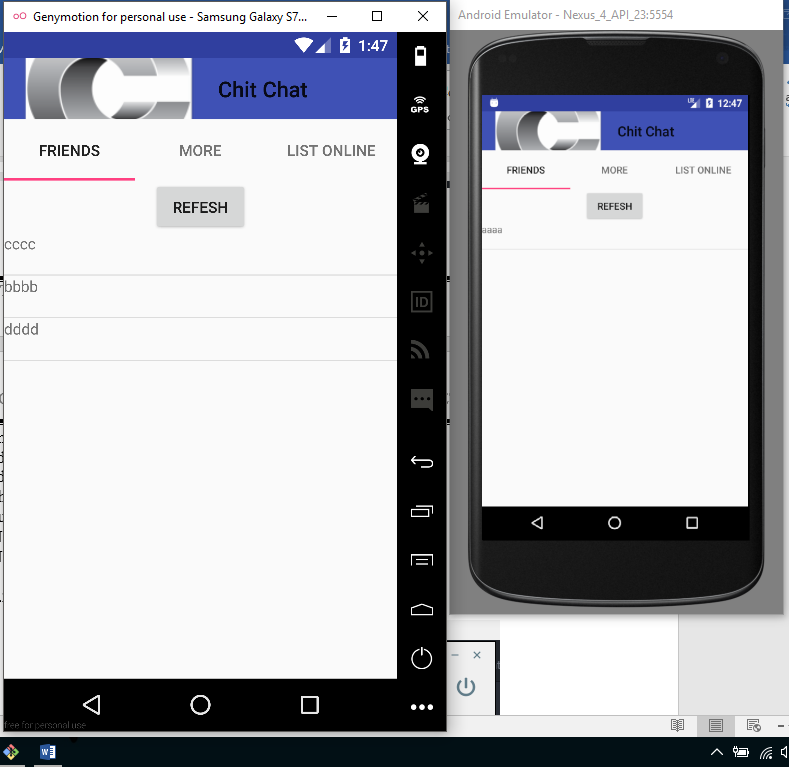
- Checkbox: đánh dấu ghi nhớ mật khẩu

- Button Sign in: nhấn vào để đăng nhập.

+ Nếu sai sẽ hiện ra lỗi: Error: Check your username or password

+ Nếu đúng sẽ vào trang chủ của ứng dụng chat thông qua intent.

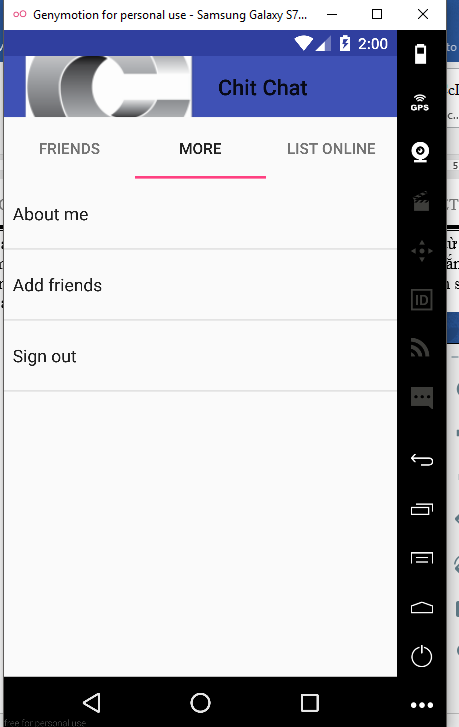
**6.1.2. Màn hình chính**

****

Hình 6.2 Tab Friends (danh sách bạn bè) - Màn hình chính đã đăng nhập của ứng dụng với 2 máy ảo AVD và Genymotion

- Màn hình chính gầm 3 tab được xây dựng bằng tabhost và viewpager:

+ Tab Friends: danh sách bạn bè mà cá nhân đã thêm vào được tạo từ list các DBUser gồm các thuộc tính : username (tên của người bạn), message (tin nhắn chờ), countmess (đếm số tin nhắn chờ) và một nút button Refesh để cập nhật lại danh sách bạn bè.

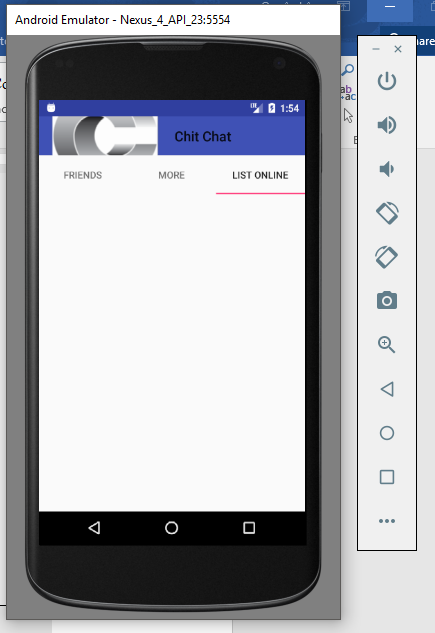
****

Hình 6.3 Tab More tại giao diện màn hình chính

+ Tab More: gồm About me, Add friends, Sign out click vào sẽ có các chức năng khác nhau như:

* About me: thông tin cá nhân và chỉnh sửa
* Add friends: thêm 1 người bạn nếu biết username của họ
* Sign out: đăng xuất khỏi ứng dụng.

+ Tab List online: hiện tất cả những người đang online lên.

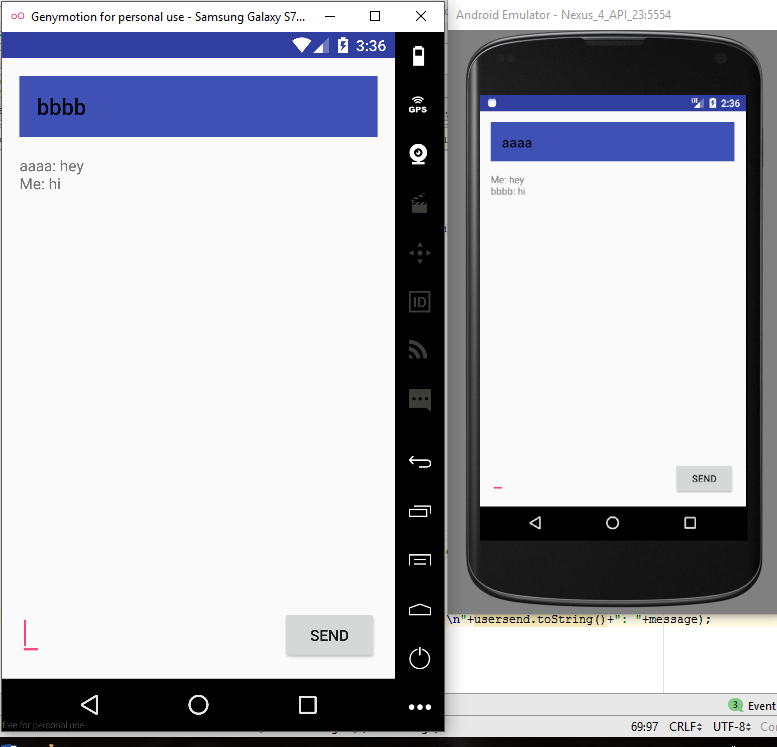
****

Hình 6.4 Tab List online tại giao diện màn hình chính

**6.1.3. Khung chat**

**-** Nhấp vào 1 người bạn và khung chat sẽ hiện lên đồng thời truyền username của người bạn chat vào.

- Chat một văn bản bất kỳ và nhấn nút send để gửi, khung chat của bạn và người bạn đang chat sẽ hiện lên đoạn văn bản đó.

****

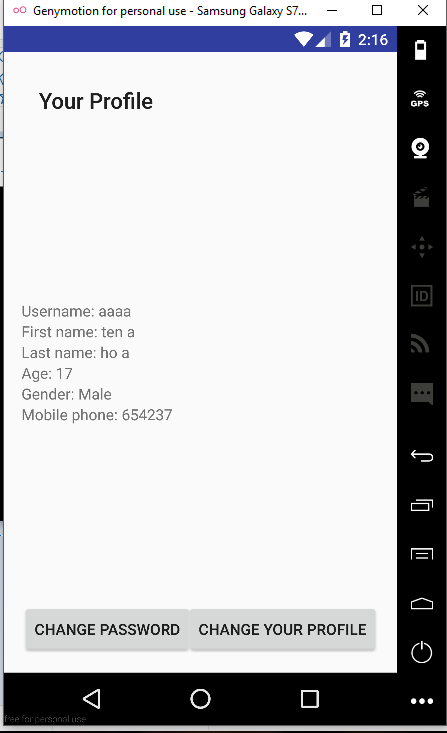
Hình 6.5 Khung chat của ứng dụng

**6.1.4. Thông tin bản thân**

**-** Click vào About me tại tab More để xem thông tin bản thân

- Các thông tin gồm username, firstname, lastname, age, gender sẽ hiện lên thông qua textview, password không được phép hiện để xem nhằm bảo mật.

- Ngoài thông tin bản thân còn có 2 button gồm Change password (đổi mật khẩu) và Change profile (đổi thông tin cá nhân còn lại).

****

Hình 6.6 Trang Your profile – Thông tin cá nhân người dùng

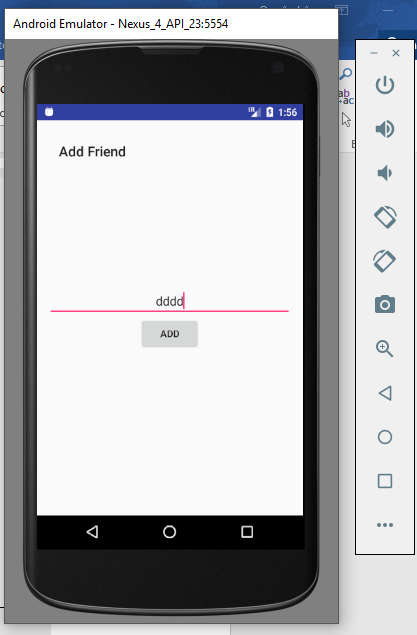
**6.1.5. Thêm bạn**

**-** Tại tab More chọn nhấn vào Add friends để them bạn mới

- Cần phải biết username của người bạn đó để nhập vào khung edittext duy nhất ở đó

- Nhấn nút Add để thêm bạn:

* Nếu username có tồn tại và add thành công
* Nếu bạn add một người đã có trong danh sách bạn, nó sẽ thông báo lỗi thông qua toast: Error: You and …(tên nhập trùng) have been friend
* Nếu username không tồn tại nó sẽ thông báo lỗi: Error: Username does not exists.

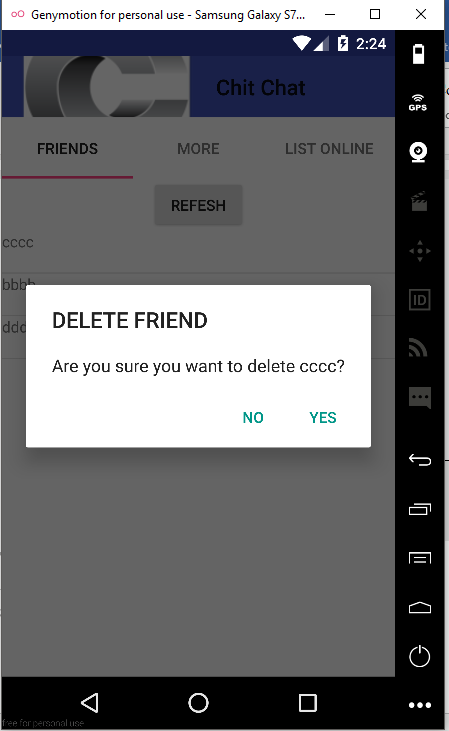
****

Hình 6.7 Giao diện thêm một người bạn

**6.1.6. Xóa bạn**

- Để xóa một người bạn, bạn vào danh bạ Friends ở màn hình chính ứng dụng, nhấn và giữ vào người bạn muốn xóa, nó sẽ hiện ra một thông báo alert như hình 6.8

- Màn hình sẽ hiện thông báo hỏi bạn có chắc muốn xóa không, nhấn YES để xóa và NO để trở về.

****

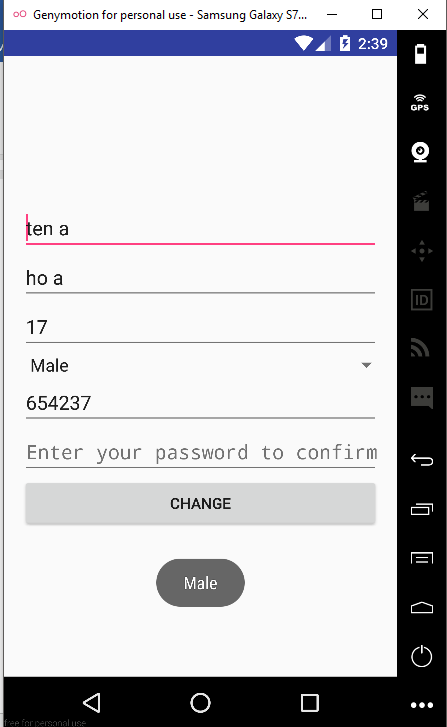
Hình 6.8 Cách xóa một người bạn trong danh sách bạn bè

**6.1.7. Đổi mật khẩu và thông tin cá nhân**

- Khi bấm vào thay đổi mật khẩu, màn hình sẽ mặc định các thuộc tính trước khi thay đổi, bạn cần xóa chúng và điền thông tin mới vào.

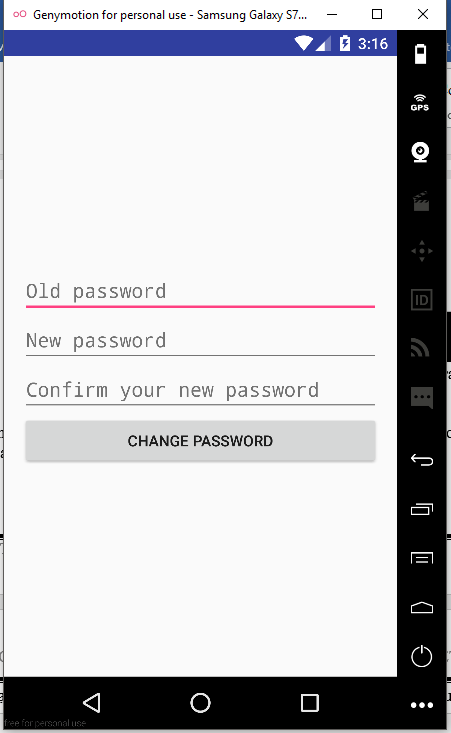
- Bạn cần phải nhập đúng mật khẩu để xác nhận bản thân khi muốn thay đổi thông tin, nếu bạn không nhập đúng mật khẩu, màn hình sẽ hiện ra thông báo lỗi và thông tin sẽ không được chỉnh sửa.

- Nhấp nút Change để thay đổi



Hình 6.9 Các thông tin cá nhân của người dùng khi chưa thay đổi

- Thay đổi mật khẩu sẽ nằm ở một trang khác khi bạn nhấp vào button Change password ở trang About me (hình 6.10)



Hình 6.10 Màn hình khi nhấp vào thay đổi mật khẩu

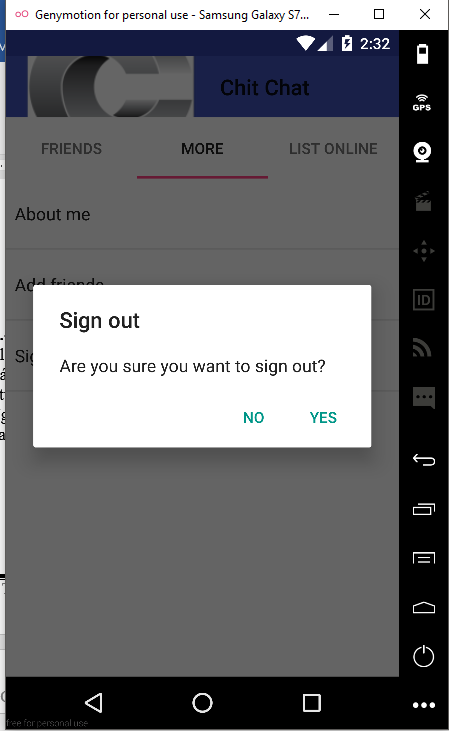
- Bạn nhập vào mật khẩu cũ để xác nhận người dùng, nhập mật khẩu mới và nhập lại mật khẩu mới 1 lần nữa và edittext tiếp theo để xác nhận mật khẩu mới của bạn

- Nhấp nút Change password để đổi mật khẩu, nếu mật khẩu cũ không đúng, nó sẽ hiện ra thông báo và mật khẩu sẽ không được thay đổi.

**6.1.8 Đăng xuất**

**-** Để đăng xuất, chỉ cần vào tab More của màn hình chính, nhấp vào Sign out.

- Màn hình sẽ thông báo như hình 6.9 hỏi rằng bạn có chắc muốn đăng xuất không, nhấp YES để thoát hoặc NO để ở lại màn hình chính.



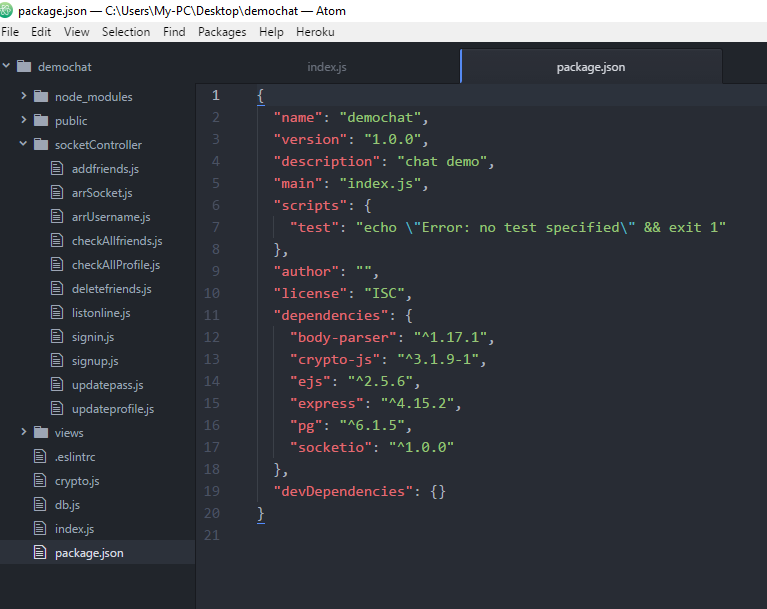
Hình 6.11 Màn hình khi nhấp nút Sign out – đăng xuất

**6.2. Server NodeJS**

- Node.js là một nền tảng chạy trên môi trường V8 JavaScript runtime - một trình thông dịch JavaScript cực nhanh chạy trên trình duyệt Chrome. Bình thường thì bạn cũng có thể tải bộ V8 và nhúng nó vào bất cứ thứ gì; Node.js làm điều đó đối với các web server. JavaScript suy cho cùng cũng chỉ là một ngôn ngữ - vậy thì không có lý do gì để nói nó không thể sử dụng trên môi trường server tốt như là trong trình duyệt của người dùng được.

-Trong bài này, server NodeJS cầm cài đặt các gói thư viện gồm body-parser, crypto-js, ejs, pg, socketio và express và sau khi cài nó sẽ nằm trong folder node\_modules.

- File index.js sẽ là file chính của server, để chạy server ta sẽ mở terminal hoặc command prompt và cd vào đúng đường dẫn của thư mục chứa nó (demochat), nhập lệnh “node index” và nhấn enter.



Hình 6.12 Các gói của server NodeJS

- File crypto.js là file chứa các dòng code cần để mã hóa mật khẩu nhằm đảm bảo hơn về tính bảo mật.

- File db.js sẽ chứa các câu lệnh truy vấn cơ sở dữ liệu

- Các file trong socketController là các lệnh code đã được tách ra từ file index.js nhằm dễ dàng quản lý code.

**DANH MỤC TÀI LIỆU THAM KHẢO**

**Danh mục các Website tham khảo:**

1. <https://www.ibm.com/developerworks/vn/library/mobile/2013Q3/j-mobileforthemasses1/>
2. <http://vietjack.com/android/>
3. <https://developer.mozilla.org/vi/docs/Web/Reference/API>
4. <https://tungnt.net/su-khac-nhau-giua-web-service-wcf-wcf-rest-web-api/>
5. <https://techmaster.vn>

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

**Độc lập- Tự do- Hạnh phúc**

TP. Hồ Chí Minh, ngày tháng năm 20……

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

**THỰC TẬP TỐT NGHIỆP CAO ĐẲNG**

1. **Tên đề tài: Xây dựng ứng dụng Chat trên Android**
2. **Sinh viên: LÊ THỊ HỒNG QUYÊN Lớp: C14CQCP01-N**
3. **Giáo viên hướng dẫn: HỒ ĐẮC HƯNG**
4. **Nơi công tác:**

**NỘI DUNG NHẬN XÉT**

1. Đánh giá chung:

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

1. Đánh giá chi tiết:

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

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

1. Nhận xét về tinh thần, thái độ làm việc:

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

1. Kết luận:

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

1. Điểm hướng dẫn ():

**GIẢNG VIÊN HƯỚNG DẪN**

(Ký, ghi rõ họ tên)