|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU PHÁT TRIỂN ỨNG DỤNG CHAT TRÊN MOBILE** |
| Chuyên ngành: An toàn thông tin  Mã số: 7.48.02.02  *Học viên thực hiện:*  **Ngô Ngọc Đạt**  *Lớp:* AT12C  *Người hướng dẫn:*  **ThS Lê Bá Cường** |
| **HÀ NỘI, 2020** |

|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU PHÁT TRIỂN ỨNG DỤNG CHAT TRÊN MOBILE** |
| Chuyên ngành: An toàn thông tin  Mã số: 7.48.02.02  *Học viên thực hiện:*  **Ngô Ngọc Đạt**  *Lớp:* AT12C  *Người hướng dẫn:*  **ThS Lê Bá Cường** |
| **HÀ NỘI, 2020** |

Lời cảm ơn

Trước khi làm đề tài này, em đã nhiều băn khoăn chẳng hạn như: nó có khó không, với kiến thức hiện tại thì em có làm được đề tài này hay không,.., vì thế em đã gặp thầy, người hướng dẫn trực tiếp (Thầy Lê Bá Cường) để xin thầy ý kiên, Thầy luôn động viên khuyến khích em làm cho em có them động lực và sự tự tin để chọn đề tài này.

Trong quá trình làm đề tài thì thầy đã tận tình chỉ dẫn khi mà em gặp khó khan hay mắc phải những vướng mắc phức tạp. Cưới cùng thì đề tài em chọn đã được hoàn thành.

Qua đây, cho em xin gửi lời cảm ơn chân thành đến thầy. Nhờ thầy mà em học được rất nhiều điều mới bổ ích. Kiến thức của em được nâng cao hơn, nó rất cần thiếu cho em cũng như công việc của em sau này. Xin cảm ơn tất cả mọi người đã tạo những điều kiện tốt nhất để chúng tôi hoàn thành thực tập tốt nghiệp này!

**SINH VIÊN THỰC HIỆN**

Ngô Ngọc Đạt

Lời cam ĐOan

Tôi xin cam đoan bản đồ án này do tôi tự nghiên cứu dưới sự hướng dẫn của thầy giáo THS Lê Bá Cường.

Để hoàn thành đồ án này, tôi chỉ dự dụng những tài liệu đã ghi trong mục tài liệu tham khảo, ngoài ra không sử dụng bất cứ tài liệu nào khác mà không được ghi.

Nếu sai, tôi xin chịu mọi hình thưc kỷ luật theo quy định của Học viện.

**SINH VIÊN THỰC HIỆN**

Ngô Ngọc Đạt

**MỤC LỤC**

[Lời cảm ơn](#_Toc43660050)

[Lời cam ĐOan](#_Toc43660051)

[Danh mục hình ảnh](#_Toc43660052)

[Lời nói đầu](#_Toc43660053)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG CHAT AN TOÀN TRÊN MOBILE 1](#_Toc43660054)

[1.1 Mô hình 1](#_Toc43660055)

[1.2 Các kiến trúc Client/Server 3](#_Toc43660056)

[1.2.1 Client/Server hai tầng 3](#_Toc43660057)

[2.2.2 Client/Server ba tầng: 4](#_Toc43660058)

[1.2.3 Client/Server n tầng 5](#_Toc43660059)

[1.3 Tiền trình xử lý 5](#_Toc43660060)

[1.3.1 Nhánh máy trạm (Client side) 5](#_Toc43660061)

[1.3.2 Nhánh máy chủ (Server side) 6](#_Toc43660062)

[1.4 Ưu điểm của mô hình Client/ Server 6](#_Toc43660063)

[1.4.1 Phát triển để giảm chi phí 6](#_Toc43660064)

[1.4.2 Mô hình Thin/ Client 6](#_Toc43660065)

[1.4.3 Tôc độ nhanh 9](#_Toc43660066)

[1.4.4 Tính tương thích cao 9](#_Toc43660067)

[1.5 Mô hình truyền tin Socket 10](#_Toc43660068)

[1.5.1 Giới thiệu khái niệm 11](#_Toc43660069)

[1.5.2 Nguyên lý hoạt động 11](#_Toc43660070)

[1.5.3 Socket hỗ trợ TCP 12](#_Toc43660071)

[1.5.4 Một số hàm và thuộc tính trong Socket 13](#_Toc43660072)

[1.6 Các cổng hướng kết nối 14](#_Toc43660073)

[CHƯƠNG 2: Nghiên cứu và phát triển ứng dụng chat 16](#_Toc43660074)

[2.1. Mô tả bài toán 16](#_Toc43660075)

[2.2 Yêu cầu chức năng của ứng dụng 16](#_Toc43660076)

[2.3. Phân tích, thiết kế yêu cầu ứng dụng 17](#_Toc43660077)

[2.3.1. Sơ đồ phân cấp chức năng 17](#_Toc43660078)

[2.3.2. Biểu đồ use case 18](#_Toc43660079)

[2.3.3. Sơ đồ luồng dữ liệu 22](#_Toc43660080)

[2.4. Phân tích, thiết kế yêu cầu an toàn 25](#_Toc43660081)

[2.4.1. Mã hóa mật khẩu – Bcrypt 25](#_Toc43660082)

[2.4.2. Mã hóa dữ liệu đầu cuối( End to end encryption) 25](#_Toc43660083)

[2.5 Thiết kế cơ sở dữ liệu 29](#_Toc43660084)

[2.5.1. Cơ sở dữ liệu cho service quản lý tài khoản 29](#_Toc43660085)

[2.5.2. Cơ sở dữ liệu cho service trò chuyện 31](#_Toc43660086)

[2.5.3. Cơ sở dữ liệu cho service quản lý liên lạc 33](#_Toc43660087)

[Chương 3: triển khai thực tiễn 35](#_Toc43660088)

[3.1 Cài đặt môi trường phát triển 35](#_Toc43660089)

[3.1.1 Cài đặt IDE Visual Studio Code (VSC) 35](#_Toc43660090)

[3.1.2 Cài đặt React Native 36](#_Toc43660091)

[3.2 Thiết kế giao diện 39](#_Toc43660092)

[Kết luận 44](#_Toc43660093)

[Tài liệu tham khảo 45](#_Toc43660094)

Danh mục hình ảnh

[Hình 1: Mô hình truyền tin Client/Server 2](#_Toc40298366)

[Hình 2: Mô hình Client/Server hai tầng 4](#_Toc40298367)

[Hình 3: Mô hình Client/ Server ba tầng 4](#_Toc40298368)

[Hình 4: Mô hình truyền tin Socket 10](#_Toc40298369)

[Hình 5: Client gửi yêu cầu đến Server 13](#_Toc40298370)

[Hình 6: Server chấp nhận yêu cầu và tạo ra socket đẻ phục vụ Client 13](#_Toc40298371)

[Hình 7: Sơ đồ phân cấp chức năng 18](#_Toc40298372)

[Hình 8: Biểu đồ use-case Quản lý tài khoản 19](#_Toc40298373)

[Hình 9: Biểu đồ use-case hiển Quản lý danh sách bạn bè 20](#_Toc40298374)

[Hình 10: Biểu đồ use-case Quản lý trò chuyện 21](#_Toc40298375)

[Hình 11: Biểu đồ use-case Quản lý file 21](#_Toc40298376)

[Hình 12: Sờ đồ đăng ký tài khoản 22](#_Toc40298377)

[Hình 13: Sơ đồ đăng nhập 22](#_Toc40298378)

[Hình 14: Sơ đồ chỉnh sửa thông tin cá nhân 23](#_Toc40298379)

[Hình 15: Sơ đồ tìm kiếm bạn bè 23](#_Toc40298380)

[Hình 16: Sơ đồ thao tác kết bạn 23](#_Toc40298381)

[Hình 17: Sơ đồ hiển thị liên hệ 24](#_Toc40298382)

[Hình 18: Sơ đồ gửi tin nhắn 24](#_Toc40298383)

[Hình 19: Sơ đồ cuộc gọi video 24](#_Toc40298384)

[Hình 20: Sơ đồ hiển thị nội dung cuộc trò chuyện 25](#_Toc40298385)

Lời nói đầu

Ngày nay, ứng dụng công nghệ thông tin và việc tin học hóa được xem là một trong những yếu tố mang tính quyết định trong hoạt động của các tổ chức nhà nước, cũng như các công ty cổ phần. Nó đóng vai trò hết sức quan trọng và có thể tạo ra những bước phát triển mạnh mẽ. Nhu câu trao đổi thông tin của con người cũng rất lớn và thông qua nhiều hình thức khácnhau như trao đổi với nhau bằng thư từ, điện thoại, mạng máy tính. Với tốc độ phát triển của công nghệ thông tin hiện nay có nhiều phần mềm trao đổi thông tin trên mạng Internet đã ra đời và được sử dụng rộng rái như: Yahoo!, Zalo, Skype, Messenger, …

Đề tài “Nghiên cứu phát triển ứng dụng chat trên mobile” được đưa ra với mong muốn xây dựng cho người dùng một mạng xã hội chat dễ dàng vào bảo mật dễ dàng chia sẻ các thông tin về các lĩnh vực thông qua chiếc máy điện thoại cầm tay. Người dụng sử dụng điện thaoị của mình để chụp ảnh sau đó chia sẻ ảnh đó đên mọi người trong danh sách bạn bè. Đây cũng là đề tài tốt nghiệp em đã chọn với sự định hướng, hướng dẫn của thầy giáo Ths. Lê Bá Cường.

Sau một thời gian tìm hiểu và nghiên cứu vè công nghệ di động React Native được sự giúp đỡ, hướng dẫn nhiệt tình của thầy giáo Ths. Lê Bá Cường, em đã hoàn thành đề tài được giao. Em xin chân thành cmả ơn các thầy cô giáo trong trường Học Viện Kỹ Thuật Mật Mã, Khoa An toàn thông tin đã giúp đỡ em em có được kết quả như hôm nay.

Tuy nhiên, do thời gian có hạn, với khối lượng công việc nhiều nên dù đã rất cố gắng nhưng đồ án không tránh khỏi những hạn chế và sai sót. Em rất mong nhận được sự đóng góp và giúp đỡ của các thầy cô và bạn bè để đề tài được hoàn thiện hơn, có thể triển khai vận hành trong thực tế trong thời gian tương lai.

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG CHAT AN TOÀN TRÊN MOBILE

1.1 Mô hình

Mô hình được phổ biến nhất và được chấp nhận rộng rãi trong các hệ thông phân tán là mô hình Client/Server. Trong mô hình này sẽ có một tập các tiền trình mà mỗi tiền trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trướcvà một tập hợp các tiền trình client trong đó mỗi tiền trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hơcj phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiền trình vừa là tiến trình client vừa là tiến trình Server. Các tiền trình phát ra các yêu cầu tới các Server. Bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các Server. Nếu yêu cầu là đúng đắn thì Server sẽ thực hiện hành động được yêu cầu và gửi một đạp ứng trả lời tới Client.

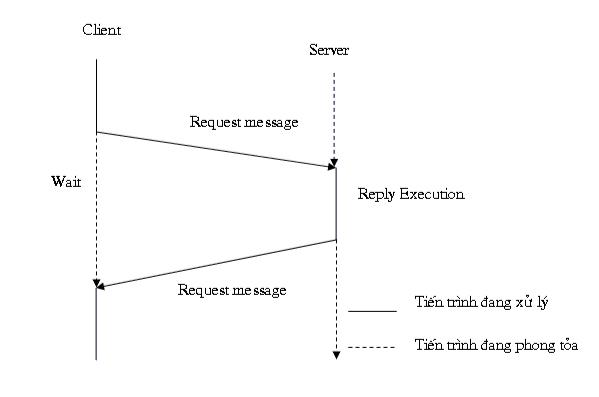
Mô hình Client/Server cung cấp một cách tiệp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán. Mô hình này có thể đọc và cài đặ bằng rất nhiều môi trường phần cứng và phần mềm các hác. Các máy tính được sử sụng đẻ chạy các tiến trình Client/Server có nhiều kiểu khác nhau và không cần thiếu phải phân việt giữ chúng, cả tiến trình Client và tiên trình Server đều có thể chạy trên cùng một máy tính. Một tiến trình Server có thể sử dụng dịch vụ của một Server khác.

Mô hình truyền tin Client/Server hướng tới việc cung cấp dịch vụ. Qúa trình trao đỏi dữ liệu bao gồm:

1. Truyền một yêu cầu tự tiến trình Client tới tiến trình Server
2. Yêu cầu được Server sử lý
3. Truyền đáp ứng cho Client

Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và một dạng đồng bộ hóa cụ thể giữa Client và Server. Tiến trình Server phải nhận thức được thông điệp được yêu cầu ở bước một ngay khi nó đến và hành động phát ra yêu cầu trong Client phải được tạm dừng (bị phong tỏa) và buộc tiến trình Client ở trạng thái chờ cho tới khi nó nhận dược đáp ứng do Server gửi về ở bước ba

Mô hình Client/Server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận (receive)



Hình 1: Mô hình truyền tin Client/Server

Quá trình giao tiếp Client và Server có thể diễ ra theo một trong h.ai chế độ: bị phong tỏa (blocked) và không bị phong tỏa (non-blocked).

Trong chế độ bị phong tỏa, khi tiến trình Client hoặc Server phát ra lệnh gửi dữ liệu (send), việc thực thi của tiến trình sẽ bị tạm ngừng cho tới khi tiến trình nhận phát ra kệnh nhân dữ liệu (receive).

Tương dự đối với tiến trình nhận dữ liệu, nếu tiền trình nào đó (Client hoặc Server) phát ra lệnh nhận dữ liệu, và tại thời điểm đó cha nó dữ liệu gửi ới thì việc thực thi của tiến trình cũng sẽ bị tạm ngừng cho tới khi có dữ liệu gửi tới.

+ Chế độ không bị phong tỏa (non-blocked)

Trong chế độ này, khi tiến trình Client hay Server phát ra lệnh gửi dữ liệu thực sự, việc thực thi của tiến trình vẫn được tiến hành và không quan tâm đến việc có tiến trình nào phát ra lệnh nhận dữ liệu đó hay không.

Tương tự cho trường hợp nhận dữ liệu, khi tiến trình ra lệnh nhận dữ liệum nó sẽ nhận dữ liệu hiên cóm và thực thi của tiến trình vẫn được tiến hành mà không quan tâm đến việc có tiến trình nào phát ra lệnh gửi dữ liệu tiếp theo hay không.

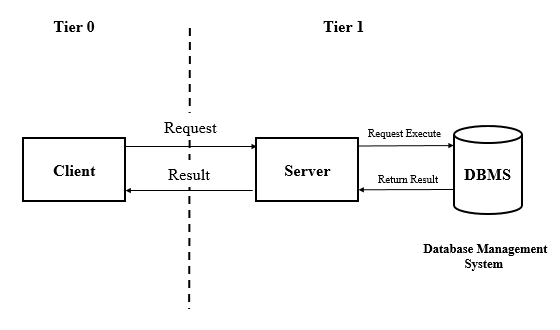
1.2 Các kiến trúc Client/Server

1.2.1 Client/Server hai tầng

Kiến trúc Client/Server đơn giản nhất là kiến trúc hai tầng. Trong thực tế hầu hết các kiến trúc Client/Server là khiến trúc hai tầng. Một ứng dụng hai tầng cung cấp nhiều trạm làm việc với một tầng trình diễn thống nhất, tầng này truyền tin với tầng lưu trữ dữ liệu tập trung. Tầng trình diễn thông thường là Client, và tầng lưu trữ dữ liệu là Server

Hầu hết các ứng dụng Internet nhưu là Email, Telnet, FPT thậm chí là cả Web là các ứng dụng hai tầng. Phần lớn các lập trình viên ứng dụng viết các ứng dụng Client/Server có xu thế sử dụng kiến trúc này. Trong bài tập của chúng ta cũng có thể sử dụng mô hình này trong quá trình truyền với một số chọn lọc phù hợp với ứng dụng của chúng ta

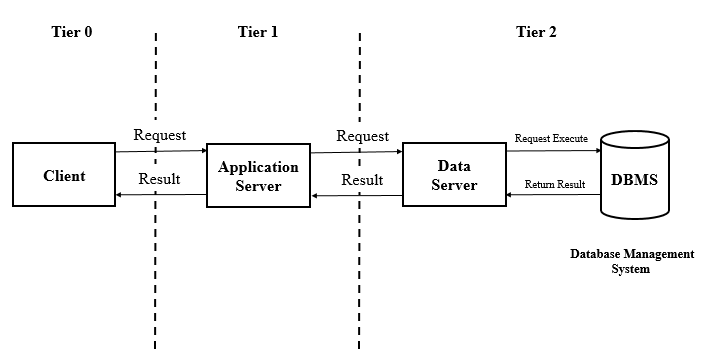
Trong ứng dụng hai tầng truyền thống, khối lượng công việc xử lý được dành cho phía Cient trong khi Server chỉ đơn giản đóng vai trò như là chương trình kiểm soát lường vào ra giữu ứng dụng và dữ liệu. Kết quả là không chỉ hiệu năng của ứng dụng bị giảm đi do tài nguyên hạn chế của PC, mà khối lượng dữ liệu truyền đi trên mạng cũng tang theo. Khi toàn bộ ứng dụng được xử lý trên một PC, ưng dụng bắt buộc phải yêu cầu nhiều dữ liệu trước khi đưa ra bất kỳ kết quả xử lý nào cho người dùng. Nhiều yêu cầu dữ liệu cũng làm giảm hiệu năng của mạng. Một vấn đề thường gặp khác đối với ứng dụng hai tầng là vẫn đề bảo trì. Chỉ cần một thay đổi nhỏ đói với ứng dụng cũng cần phải thay đổi lại toàn bộ ứng dụng Client và Server



Hình 2: Mô hình Client/Server hai tầng

2.2.2 Client/Server ba tầng:

Ta có thể tránh được các vấn đề của kiến trúc Client/ Server hai tầng bằng cách mở rộng các kiến trúc ba tầng. Một kiến trúc ba tầng có them một tầng mới tách biết xử lý dữ liệu ở vị trí trung tâm



Hình 3: Mô hình Client/ Server ba tầng

Theo kiến trúc ba tầng, một ứng dụng dược chia thành ba tầng tách biệt nhau về mặt logic. Tầng đầu tiên là tầng trình diễn thường bao gồm các giao diện đồ họa. Tầng thức haim còn được gọi là tầng trung gian hay tầng tác nghiệp. Tầng thưucs ba chứa dữ liệu cần cho ứng dụng. Tầng thứ ba về cơ bản là chương trình thực hiện các lời gọi hàm để tìm kiếm dữ liệu cần thiết. Tầng trình diễn nhận dữ liệu và định dạng nó để hiển thị. Sự tách biệt giữa chức năng xư rlý với giao diện đã tạo nên sự linh hoạt cho việc thiết kế ứng dụng. Nhiều giao diện người dùng được xây dựng và triển khai mà không làm thay đổi logic ứng dụng

Tầng thứ ba chưa dữ liệu cần thiết cho ứng dụng. Dữ liệu này có thể bao gồm bất ký nguồn thông tin nào,bao gồm cơ sở dữ liệu nhưu Oracale, SQL Server hoặc tài liệu XML.

1.2.3 Client/Server n tầng

Kiến trúc n-tầng được chia thành các tầng như sau:

* Tầng giao diện người dùng: quản lý tương tác của người dùng với ứng dụng.
* Tầng logic trình diễn: Xác định các thức hiển thị giao diện người dùng và các yêu cầu của người dùng được quản lý như thế nào
* Tầng logic tác nghiệp: Mô hình hóa các quy tắc tác nghiệp.
* Tầng các dịch vụ hạ tầng: Cung cấp một chức năng bổ trợ cần thiết cho ứng dụng các thành phần (truyền thông điệp, hỗ trợ giao tiêp)

1.3 Tiền trình xử lý

Trong mô hình khách chủ, ngoài hệ thống mạng máy tính phải có còn đòi hỏi việc tổ chức các xử lý bên dưới sao cho hiệu quả, đảm bảo các yêu cầu (request) tự các máy trạm phải được máy chủ đáp ứng (response) một cách nhanh chóng, không làm tắc nghẽn hệ thống. Khi thiết kế các ứng dụng theo mô hình khách chủ, người ta chia các xử lý ra làm hai nhánh: nhánh máy trạm và nhanh máy chủ.

1.3.1 Nhánh máy trạm (Client side)

Các ứng dụng sẽ thực hiện các công việc đọc và hiển thị dữ liệu hiện có bên trong cở sở dữ liệu, tính toàn dữ liệu đang hiển thị trên các màn hình ứng dụng, in dữ liệu ra

Các ngôn ngữ dùng để xây dựng ứng dụng là Delphi, Visual Basic, C++, C#, … Các ứng dụng này còn cho phép người dùng có thể thực hiện các thao tác xóa, them, sửa dữ liệu hiện có bên trong cở sở dữ liệu bên nhành máy chủ.

Các ứng dụng khi xây dựng nên tránh việc đọc toàn bộ dữ liệu của bằng (Table) khi truy xuất dữ liệu tự máy chủ mà chỉ nên lấy về đúng các thông tin cần thiết cho các xử lý. Việc này làm giảm đi lượng thông tin lưu thông trên mạng

1.3.2 Nhánh máy chủ (Server side)

Các xử lý được thực hiện trực tiếp trên máy chủ. Để đảm bảo việc bảo mật (security), những người dùng trên mạng phải được cấp phát quyền truy cập thì mới có thể truy xuất được các dữ liệu dùng chugn.

Việc cập nhật dữ liệu cho phép đồng thời cùng lúc giữu những người dùng hiện hành trên mạng, ví dụ như máy chủ cho phpé cùng lúc cả hau người dùng có thể cập nhật thông tin của khách hang trong bảng khách hang.

Việc sao lưu dữ liệu (backup data) được tự động để đảm bảo dữ liệu không bị mất trong các trường hợp xảy ra.

1.4 Ưu điểm của mô hình Client/ Server

1.4.1 Phát triển để giảm chi phí

Trong môi trường cạnh tranh mạnh mẽ như ngày nay, giảm chi phí đầu ra nhưng đảm bảo được hiệu quả công việc là một trong những yếu tố quan tâm hang đầu của các nhà kinh doanh. Mặt khác, dầu ra cho một trường công nghệ thông tin (cả phần mềm lẫ phần cứng) là không thể thiếu đối với mọi doanh nghiệp ngày nay, đặc biệt là các doanh nghiệp có quy mô tự trung bình đến lớn.

Về vấn đề này, các nhà quản lý doanh nghiệp luôn phải đối diện với một mâu thuẫn. Để đảm bảo hiệu quả, mọi nhân viên của công ty (hay phần lớn nhân viên) phải được trang bị một máy tính riêng của mình. Các máy tính này phải đủ mạnh và tương đối đồng bộ với nhau về phần cứng lẫn phần mềm. Tuy nhiên, việc trang bị cho mỗi nhân viên một máy tính với cấu hình (phần cứng lẫn phần mềm) đồng bộ đòi hỏi một chi phí đầu ra khá lớn. Đó là chưa kể đến vấn đề bảo trì. Bảo trì một hệ thống máy tính được nối mạng thường phức tạp và tốn rất nhiều thời gian và itền bạc (thông thường, các doanh nghiệp phải duy trì một đội ngũ bảo trì hoạt động thường xuyên)

1.4.2 Mô hình Thin/ Client

Ngày nay, các doanh nghiệp thường sử dụng rổng rãi mô hình Client/ Server. Trong mô hìnhnày, tất cả máy tính để bàn các nhân viên (gọi là máy Cient) sẽ được kết nối vào một máy tính trung tâm (máy Server). Máy Server sẽ đóng vai trò như một kho tài nguyên phcụ vụ cho tất cả các máy Client kết nối vào nó. Các tài nguyên này thường là dữ liệu, cá thiết bị phần cứng và một số dịch vụ mạng khác.

Bên cạnh khả năng xử lý của Server các máy Client đều có năng lực xử lý (thường là yếu hơn Server) là lưu trữ riêng của nó để phục vụ cho các tác vụ không liên quan đếnmạng như soạn thảo văn bản, bảng tính, hình vẽ, …

Tuy nhiên, trong một doanh nghiệp, các ứng dụng phải mang của các máy Client thường là giống nhau. Ngày nay, vào bất cứ doanh nghiệp nào, bạn cũng đều thấy mỗi máy tính của các nhân viên đều có cài đặt Microsoft Windows, Microsoft Word, Microsoft Excel, các chường trình nghe nhạc, xem phim, …

Để cài đặt toàn bộ hệ thống gồm có 100 máy tính của công ty. Những nhân viên phụ trách (ở Việt Nam thường gọi là admin) phải cài đặt 100 lần giống hệt nhau ở mỗi máy tính. Và để bảo trì, họ cũng phải tiến hành bảo trì từng máy tính một. Bạn thử nghĩ xem, nếu 100 máy tính này đặt ở nhiều phòng ban khác nhau, thậm chí ở những tầng khác nhau của một tòa nhà thì công tác cập nhật sẽ khó khăn đến mưucj nào? Đa số nhân viên đều không phải là người thành thạo máy tính, nên đối với họ, việc cài đặt hoặc xử lý các hỏng hóc của máy tính là việc rất khó khắn. Một trong những phản ứng tự nhiên của các nhân viên “xin lỗi, cho phép tôi gặp quản trị”. Và điều này đã làm cho công tác quản trị đốiv ới các doanh nghiệp và đặc bệt là các thành viên quản trị.

Để giải quyết vấn đề “phân tán” này. Người ta đã nghĩ ra một giải pháp tương đối đơn giản. Đó là đặt tất cả các tập tin chương trình lẫn dữ liệu của phần mềm lên máy Server. Khi muốn sử dụng phần mềm thì máy Client sẽ tải chương trình (và dữ liệu) xuống máy Client và sau đó, chương trình (hoặc dữ liệu) này sẽ được thực thi bằng CPU của máy Client.

Tuy nhiên, giải pháp này sẽ ảnh hưởng đến sở thích cá nhân của mỗi nhân vien. Các nhan viên thường muốn phần mềm có một bề ngoài riêng (như các Wall-Paper hoặc Screen Saver trên môi trường Windows, hoặc các skin của WinAmp, …)

Như vậy, làm sao có thể quản trị tập trung hệ thóng máy tính mà vẫn đảm bảo được tính riêng tư của từng người dùng?

Đến đây, người ta đã ra công nghệ Thin-Client. Thin-Client có nghĩ là “máy tính của người dùng có cấu hình tối thiêu”. Công nghệ Thin-Client tương tự công nghệ Client/Server, điểm khác duy nhất và cũng uqna trọng nhất là tất cả năng lực xử lý lẫn lưu trữ của toàn bộ mạng máy tính đều tập trung vào máy Server. Mọi máy Client đều chỉ còn một màn hinh, bàn phím và chuột. Mọi chương trình hay phần mềm của người dùng sẽ được lưu trữ trên Server và cũng sẽ được thi hành bời CPU của máy Server.

Ưu điểm của công nghệ này nằm ở các điểm sau:

**Đối với người dùng**

Thực hiện: cắm máy tính của mình vào mạng, mở màn hình lên, tiến hành một số thử tục đơn giản để đăng ký cấu hình cho “máy tính” của mình (thông thường là thao tác chọn một dòng trong một danh sach). Môi trường Windows cùng với các phần mềm Office quen thuộc sẽ ngay lập tức xuất hiện trên màn hình sẵn sang chờ lệnh chúng ta mà không cần phải trải qua hang giờ liền ngồi cài đặt máy tính.

Tính bảo mật cao hơn (vì toàn bộ thao tác đều được kiểm soát chặt chẽ bởi Server). Những kẻ xâm nhập trái phép sẽ hầu như không có cơ hội tìm kiếm các thông tin quan trọng được lưu trữ trên các máy người dùng như trước kia. Bây giừo, những kẻ xâm nhập muốn tìm kiếm thông tin đầu phải đối diện với một khó khăn cực lớn là xâm nhập vào các máy Server, vốn đã được bảo vệ cực kỳ nghiêm ngặ so với các máy tính người dùng. Nguy cơ bị nhiễm virus vào các “máy tính” Client gần như là không có

**Đối với chủ doanh nghiệp**

Chi phí thấp hơn: vì chỉ cần mua một máy chủ mà thay vì phải mua 100 máy con. Cho phép sử dụng một đội ngũ quản trị với ít người hơn. Hơn nữa, việc tiêu thụ điện sẽ thấp hơn nhiều. Người ta ước tính rằng côn gnghệ này có khả năng giúp doanh nghiệp tiết kiệm đến 80% chi phí đầu tư cho CNTT.

Dễ dàng quản trị: chỉ cần cài đặt phần mềm một lần, hơn nữa chi phí mua phần mềm cũng rẻ hơn (mua 100 phần mềm riêng lẻ sẽ mắc hơn rất nhiều khi mua một phần mềm cho 100 người dùng). Việc bảo trì thiết bị và xử lý sự cố cũng đơn giản hơn (vì chỉ cần bảo trì máy Server thay vì phải bảo trì 100 máy con).

Rất tiện lớik khi nâng cấp: chỉ cần nâng cấp một máy Server thay vì phải nâng cáp riêng từng máy.

Tiết kiệm được không gian lưu trữ: thay vì mỗi máy phải lưu trữ một bản sao của phầnmềm thì chỉ càn lưu một bản sao duy nhất trên Server. Bảo vệ trước sự xâm nhập của virus: mọi hoạt động của mạng đều có thẻ được theo dõi và giám sắt tập trung để ngăn chặn ngay từ đầu nguy cơ bị phá hoại bời các loại virus (thay vì trước kia phải khuyến cáo mỗi người dùng tự cài đặt các phần mềm chống virus và có trách nhiệm đối với việc bảo vệ máy tính của mình). Người dùng không có quyền tự ý download và cài đặt những phần mềm nguy hiềm.

Cho đến nay, vị trí của công nghệ Thin-Client gần như đã được khẳng định. Với một chi phí đầu vào thấp hởn rất nhiều, công nghệ Thin-Client có thể hoàn toàn thay thế được cho mô hình truyền thống mà vẫn đảm bảo được năng lực lẫn tính hiệu quả của hệ thống

**Tương lai của Thin-Client**

Vai trò của Thin-Client đối với môi trường doanh nghiệp đã rõ rang. Liẹu trong tương lai, Thin-Client có thể là công nghệ chủ lực với viẹc truy cập từ xa hay không?

Khi người dùng bắt đầu làm việc với công nghệ “Thin”, họ đã nhanh chóng cảm nhận được sự tiện lợi của nó. Cho dù người dùng đang ở bất kỳ đâu, trong văn phòng làm việc của công ty, trong một đại lý ở xa của công ty hoặc thậm chí ở cả phi trường, chỉ cần sử dụng phần mềm giả lập của Thin-Client chạy trênmột máy tính xách tay, toàn bộ môi trường quen thuộc và ổn định của mạng Lan sẽ xuất hiện ngay trước mắt người dùng cho phép họ truy xuất vào toàn bộ kho dữ liệu, sử dụng các ứng dụng, kết nối Internet tốc độ cao với đầy đủ dịch vụ (e-mail, www, …)

1.4.3 Tôc độ nhanh

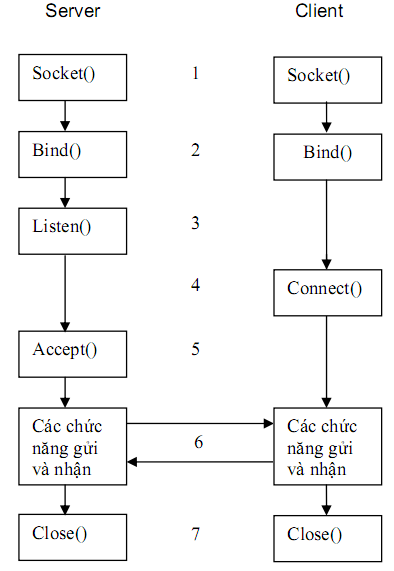
Giảm việc tắc nghẽn thông tin hệ thống mạng do phân chia các xử lý ra thành 2 nhánh. Các xử lý nào phức tạp tác động nhiều lên cơ sở dữ liệu sẽ được lưu trữ ngay trên máy chủ, các xử lý đơn giản sẽ được thực hiện nay trong ứng dụng trên máy trạm. Chính vì thếhệ thống vận hành sẽ hiệu quả hơn.

1.4.4 Tính tương thích cao

Việc chọn lựa các phần mềm để phát triển ứng dụng có thể hoàn toàn độc lập từ ngôn ngữ lập trình đến hệ cơ sở dữ liệu quan hệ và các thiết bị phần cứng. Bạn có thể chọn các thành phần tối ưu nhất khi xây dựn một hệ thống ứng dụng.

1.5 Mô hình truyền tin Socket

Đây là mô hình và cũng là các bước ta cần đặc biệt quan tâm trong quá trình tạo ứng dụng của chúng ta. Nó ucng cấp cho ta quá trình trao đổi thông tin và dữ liệu giữa các thànhphần trong mô hình Client/Server. Mô hình này cung cấp cho ta các thực thể và trình tự để ta có thể xây dựng ứng dụng. Khi lập trình ta có thẻ tuân theo trình tự sau đây đẻ tạo ứng dụng.



Hình 4: Mô hình truyền tin Socket

Khi lập trình người ta thường quan tâm đến chế độ bị phong tỏa vì nó có thể dẫn đến tình huống một tiến trình nào đó sẽ rơi vào vòng lặp vô hạn của quá trình gửi hoặc nhận.

1.5.1 Giới thiệu khái niệm

Socket là giao diện lập trình ứng dụng (API-Application Programming Interface), được giới thiệu lần đầu tiên trong ấn bản UNIX-BSD 4.2 dưới dạng các hàm hệ thống theo cú phép ngôn ngữ C (socket(), bind(), connect(), send(), receice(), read(), write(), close(), …)

Ngày nay, socket hỗ trợ hầu hết các hệ điều hành như MS Windows, Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau như: C, C#, C++, Java, Visual basic, … Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hau cổng (port), thông qua các cổng này một quá trình có thể nhận và gửi dữ liệu với các quá trình khác nhau.

1.5.2 Nguyên lý hoạt động

Trước hết chúng ta hãy xem làm thế nào các các socket có thể xác định được nhau.

Khi một chương trình tạo ra một socket, một định dạng số sẽ được gán cho socket, định dạng này chính là cổng mà chúng ta đã tìm hiểu ở trên. Việc gắn số cổng này cho socket có thể được thực hiện bởi chương trình hoặc bời hệ điều hành tùy theo các socket được sử dụng như thế nào. Trong mỗi gói tin mà socket gửi đi có chưa hai thông tin để xác minh đích đến của gói tin:

* Một địa chỉ mạng để xác đinh hệ thông sẽ nhận gói tin
* Một số định dạng cổng đẻ nói cho hệ thống đích đến socket nào trên nó sẽ nhạn dữ liệu

Nhờ hai thông tin này mà gói tin có thể đến được đúng máy tính chứa socket mà nó cần đến (nhờ địa chỉ mạng) và được phân phối đến đúng socket đích (nhờ địa chỉ cổng của socket đích).

Bời dưới góc độ lập trình các socket thường làm việc thoe cặp, một socket đóng vai trò làm Client còn các socket khác đóng vai trò như một Server. Socket phía Server xác định một cổng cho giao tiếp mạng, sau đó chờ nghe dữ liệu mà Client gửi tới nó bằng Client socket. Do đó các cổng cho Server socket phải được viết bởi các chương trình Client. Ví dụ Server FTP sử dụng một socket để nghe tại cổng 21 do đó nếu một chương trình Client muốn giao tiếp với Server FTP nó cần phỉa kêt snối đến socket đang nghe tại cổng 21.

Như vậy cổng của socket phía Server được xác minh bời chương trình, ngược lại cổng cho Client socket được xác địch bởi hệ điều hành. Khi một socket phía client gửi một gói tin tói socket phía Server thì trong ogí tin đã có chứa thông tin về địa chỉ của hệ thống Client và cổng của socket phía Client nên Server hoàn toàn có thể gửi thông tin phản hồi cho Client.

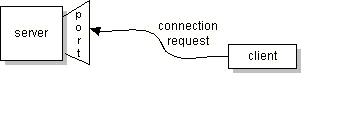
Chúng ta có thể khái quát quá trình trao đổi dữ liệu thông qua các socket như sau:

1. Chương trình phía Server tạo ra 1 socket, socket này được chương trình gắn với một cổng trên Server. Sauk hi được tạo ra socket này (mà từ nay ta sẽ gọi là socket phía Server) sẽ chờ nghe yêu cầu từ phía Client.
2. Khi chương trình phía Client cần kết nối với một Server, nó cũng tảo ra một socket, socket này cũng được hệ điều hành gắn với một cổng. Chương trình Client sẽ cung cấp cho socket của nó (mà từ nay ta sẽ gọi là socket phía Client) địa chỉ mạng và cổng của socket phía Server và yêu cầu thực hiện kết nối (nếu chương trình định sử dụng giao thức hướng kết nối) hoặc truyền dữliệu (nếu chương trình sử dụng giao thức hướng kết nối).
3. Chương t rình phía Server và chương trình phía Client trao đổi dữ liệu với nhau bằng cách đọc từ socket hoặc ghi vào socket của mình. Các socket ở hai phía nhận dữ liệu tự ứng dụng và đóng gói để gửi đi hoặc nhận các dữ liệu được gửi đến và chuyển cho chương trình ứng dụng bởi socket ở cả hai phía đều biết được địa chỉ mạng và địa chỉ cổng của nhau

Bước thứ hai chúng ta thấy chương trình ứng dụng phải lựa chọn giao thức mà nó định sử dụng trao đổi dữ liệu. Tùy theo việc sử dụng giao thức nào (TCP hay UDP) mà cách thưucs xử lý trước yêu cầu của Client có thể khác. Sau đâu chúng ta sẽ xem xét chi tiết cách thức đổi dữ liệu của Socket ứng với từng laoị giao thức.

1.5.3 Socket hỗ trợ TCP

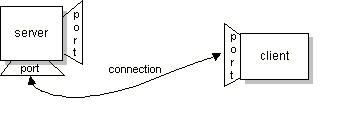
**Phía Server:** Khi một ứng dụng trên Server hoạt động nó sẽ tạo một socket và đăng ký với Server một cổng ứng dụng và chờ đợi yêu cầu kết nối từ phí Client qua cổng này.



Hình 5: Client gửi yêu cầu đến Server

**Phía client:** nó biết địa chỉ của máy trên đó Server đang chạy vào cổng và Server đang chờ nghe yêu cầu. Do đó khi muốn kết nối đến Server, nó cũng tạo ra một socket chưa địa chỉ máy Client và cổng của ứng dụng trên máy Client đồng thời Client sẽ cung cấp cho socket của nó địa chỉ và cổng của Server mà nó cần kết nối và yêu cầu socket thực hiện kêt nối.

Khi server nhận được yêu cầu kết nối từ Clietn, nếu có chấp nhận thì Server sẽ sinh ra một socket mới được gắn với một cổng khác với cổng mà nó đang nghe yêu cầu. Sở dĩ Server làm như vậy bời nó cần cổng cũ để tiếp tục nghe yêu cầu từ phía Client trong khí vẫn cần một kết nối với Client



Hình 6: Server chấp nhận yêu cầu và tạo ra socket đẻ phục vụ Client

Sau đó chương trình ứng dụng phía Server sẽ gửi thông báo chấp nhậ kết nối cho Client cùng thông tin vè địa chỉ cổng mới của socket mà nó dành cho Client.

Quay lại phía Client, nếu kết nối được chấp nhận nghĩa là socket của nó đã được tạo ra thành công và nó có thể sử dụng socket để giao tiếp với Server bằng cách viết và ghi tới socket theo cách giao tiếp với một tài nguyên trên máy tính thông thuơgf

1.5.4 Một số hàm và thuộc tính trong Socket

* Accept(): Chấp nhận một kết nối vào socket
* Bind(): Gán tên định danh cho socket
* Close(): Đóng một kết nối socket
* Connect(): Khởi tạo một kết nối tới một socket
* Listen(): Mở một socket trên cổng để đợi một kết nối tới
* Create(): Tạo một socket (đầu cuối cho giao tiếp)
* GetSocketoption(): Đa cách tùy chọn cho socket
* Receive(): Nhạ dữ liệu từ một kết nối socket
* Recvfrom(): Nhận dữ liệu tự một socket có định hướng không
* Send(): Gửi dữ liệu tự một kết nối socket
* SendTo(): Gửi tin nhắn tới một socket thông báo nó đã kết nối được hay chưa

1.6 Các cổng hướng kết nối

Port có thể gọi là một cổng kết nối cho mạng thông tin, mạng Internet

Cổng kết nối có thể là một cổng thực, vật lý, ví dụ như là trong máy tính có cổng kết nối dây chuyển (serial port) COM1, cổng kết nối song song (parallel port) LPT1 cho máy in, cổng USB, cổng qua card mạng, … Tuy nhiên trong lý thuyết mạng, cổng kết nối có thể là khái niệm trừu tượng, đánh số từ 0 đến 65535, được dùng trong các phần mềm (không vật lý), ví dụ cổng kết nối 8080 cho Web Brower, cổng kết nối 80 cho Web Server, … Lý do người ta phát minh ra cái khái niệm trừu tượngnày là vì có thể có nhiều ứng dụng cùng dùng chung một cổng vật lý, ví dụ có nhiều ứng dụng cổng card mạng: email, browerr, … Để tránh nhầm lẫn nhau, mỗi ứng dụng phải dùng riêng ổ kết nối (socket), dùng riêng giao thức (protocol)

Một Client có thể chạy nhiều dịch vụ trên mạng với giao thức TCP/IP sẽ liên kết với các port (cổng) ở lớp vận chuyển với một ứng dụng tương ứng. Số thứ tự của port được sử dụng để phân biệt giữa nhiều cuộc kết nối vơi snhau tại cùng một thời điển trên mạng. Chỉ số port đặc biệt cần thiết khi host được thực hiện liên lạc với một Server chạy nhiều dịch vụ đồng thời.

Số port được phân ra thành 3 phần như sau:

* Chỉ số port dưỡi 255 dành cho các ứng dụng công cộng
* Chỉ số port tự 255 đến 1023 cung cấp cho các ứng dụng được mua bán trên thị trường của các công ty
* Chỉ số port trên 1023 chưa được sắp xếp

Hệ thống đầu cuối sẽ sử dụng chỉ số port để xác đinh ứng dụng tương ứng. Số port nguồn được host lựa chọn tự động và thường là số port lớn hớn 1023. Số port từ 0-1023 được kiếm soát bởi tổ chức cung cấp số Internet (Internet Assigned Numbers Authority).

Khi Client kết nối vào một dịch vụ Server thì phải xác định được port nguồn và port đích. Trong gói TCP và UDP có phần dành cho số port đích và nguồn. Port đích là port dành cho dịch vụ, nên thông thường là các port nổi tiếng. Còn port nguồn thì do Client tự động lựa chọn.

Để có thể dễ dàng hiểu được công việc của Socket và Port trong kết nối giữu Client và Server thì ta lấy ví dụ sau:

*Ví dụ:* Để dễ hiểu ta có thể liên tưởng hai máy tính được kết nối vơi snhau như hai thí sinh trong phòng thi, A và B đang thi lại môn chế tạo bom chẳng hạn. Tất nhiên là B không biết khoản này nên tìm cách kết nối với A đẻ hỏi bài. Khi đó có thể B như là chương trình phía Client và A chạy bên phía Server. Chủ thể B (Client) khi bí câu nào sẽ gửi thông điệp (bằng cahcs đưa cho mảnh giấy chẳng hạn) đến A (Server). A sau khi nghiên cứu sẽ chép câu trả lời vào igấy và chuyển sang cho B.

Trong mạng máy tính khi hai máy tính kết nối vơi snhau theo giao thức TCP, chúng sẽ giao tiếp với nhau bằng cách gửi và nhận thông tin qua socket. Như trong ví dụ trên thì socket chính là mảnh giấy, Client và Server giao tiếp với nhau bằng cách ghi và đọc với mảnh giấy.

Tiếp tục với ví dụ trên, trước khi chuyển bài chắc chắn B và A sẽ phải quy ước với nhau là sẽ chuyển giấy xuống dưỡi chân cho khỏi bị phát hiện. Và A sẽ luôn phải chờ B chuyển giấy (giả sử là A làm xong bài rồi). Sau khi đọc yêu cầu, A sẽ phải tìm tờ giấy ghi bài làm rồi chuyển cho B.

Trong mạng máy tính cũng vậy, thông thường mỗi socket gắn vỡi một cổng (port) nhất định và phía Server luôn lắng nghe (listen) là đợi đến khi socket từ phía Client gửi đến. Sauk hi nhận, Server sẽ tạo mới socket và gửi về Client. Và dĩ nhiên là thông tin được đọc và ghi lên socket.

CHƯƠNG 2: Nghiên cứu và phát triển ứng dụng chat

2.1. Mô tả bài toán

Các ứng dụng trò truyện trực tuyến hiện nay đang rất phổ biến và đa dạng, đáp ứng nhu cầu về giải trí, liên lạc cũng như trao đổi công việc. Các ứng dụng nổi bật hiên nay có thể kể đến như: Messenger, Zalo, Discord, Skype, Chatwork,… Hệ thống liên lạc trực tuyến đã giải quyết các vấn đề về không gian và thời gian, đem con người lại gần nhau hơn

Ứng dụng chat an toàn trực tuyến dựa trên cũng mang nhữn chức năng chính của những ứng dụng nổi bật hiện nay như trò truyện, gọi điện trực tuyến, kết bạn, chia sẻ hình ảnh tài liệu,… Hệ thống trò truyện phải ưu tiên cho việc đảm bảo tính bảo mật và riêng tư. Việc này là hết sức quan trọng vì nột dung các cuộc trò truyện dù là ở bất cứ nơi đâu vẫn luôn cần được giữ bí mật. Hệ thống trò tuyện an toàn sẽ mang lại sự thoải mái và tin tưởng cho người dung. Ngoài ra, vì là một ứng dụng thời gian thực nên phải đảm bảo tính sẵn sàng, tốc độ phản hồi tức thì, giảm nhiều sự chậm trễ

Ứng dụng trò truyện trực tuyến đáp ứng được nhu cầu truy cập của nhiều người dùng truy cập lớn từ các thiết bị khác nhau cũng như phải thích nghi với sự thay đổi của công nghệ. Hơn nữa, một ứng dụng trò chuyện xây dựng cần sự tham gia của nhiều người, nên hệ thống cần phải có kiến trúc dễ hiểu và đáp ứng khả năng có thể chia nhỏ để nhiều người cùng phát triển mà không ảnh hưởng đến nhau và nó phải dễ bảo trì tròn quá trình hoạt động. Việc bảo trì bất cứ chức năng trong hệ thống phải độc lập và không ảnh hướng đến hoạt động của cả hệ thống. Chính vì những yêu cầu này tôi quyết định chọn làm kiến trúc để xây dựng ứng dụng trò truyện trực tuyến

2.2 Yêu cầu chức năng của ứng dụng

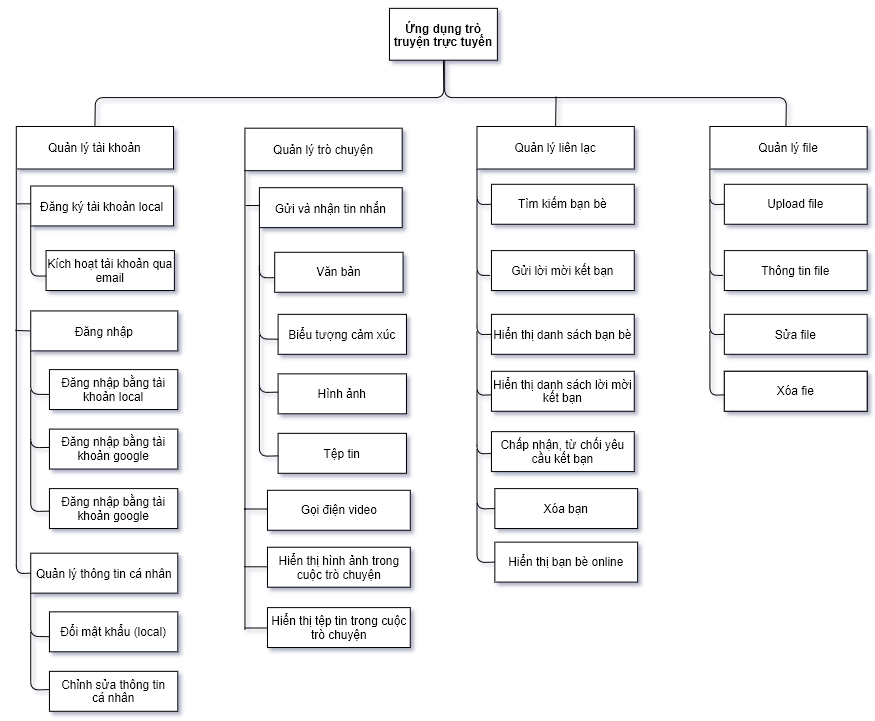
* Đăng ký, đăng nhập, đăng xuất với tài khoản local hoặc mạng xã hội
* Xác thực qua Email khi người dùng đăng ký tài khoản Local.
* Chức năng chỉnh sửa thông tin cá nhân: tên người dùng, ảnh đại diện, số điện thoại, địa chỉ, mật khẩu.
* Tìm kiếm người dùng, kết bạn.
* Hiển thị danh sách bạn bè, lời mời kết bạn.
* Danh sách yêu cầu kết bạn đã gửi, danh sách bị chặn.
* Hiển thị bạn bè online, offline
* Chat riêng giữa bạn bè với nhau.
* Chat nhóm.
* Call stream video.
* Call sream voice.
* Gửi nhận tin nhắn/hình ảnh/tệp tin/biểu tượng cảm xúc.

+ Trong box chat, có phần chuyển sang xem những hình ảnh trong box.

+ Trong box chat, có phần chuyển sang xem những tệp tin trong box.

2.3. Phân tích, thiết kế yêu cầu ứng dụng

2.3.1. Sơ đồ phân cấp chức năng

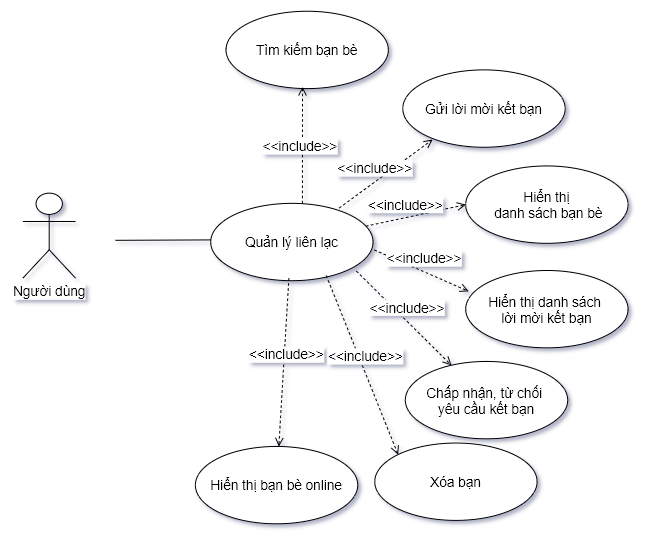


Hình 7: Sơ đồ phân cấp chức năng

2.3.2. Biểu đồ use case



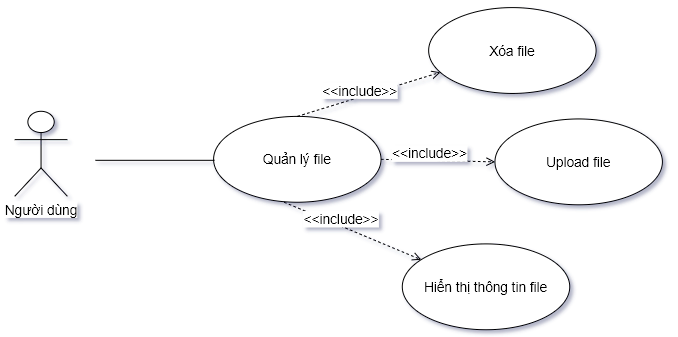
Hình 8: Biểu đồ use-case Quản lý tài khoản



Hình 9: Biểu đồ use-case hiển Quản lý danh sách bạn bè

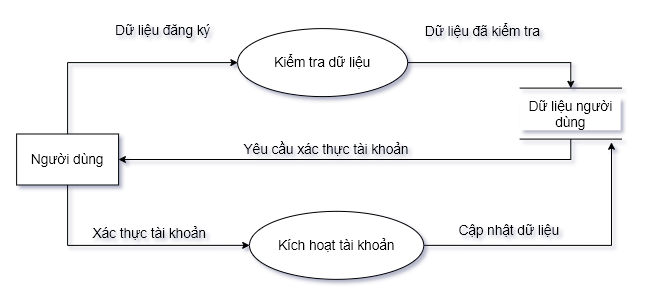


Hình 10: Biểu đồ use-case Quản lý trò chuyện

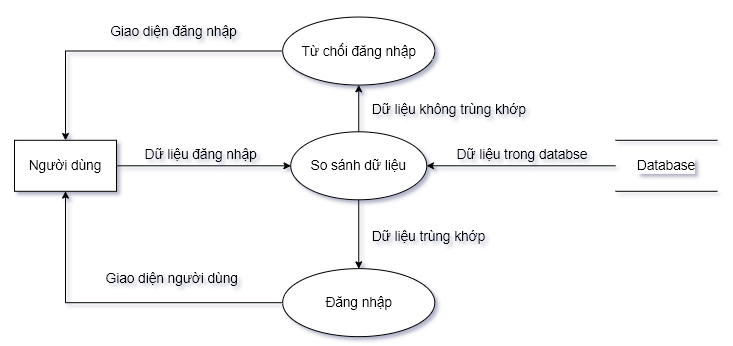


Hình 11: Biểu đồ use-case Quản lý file

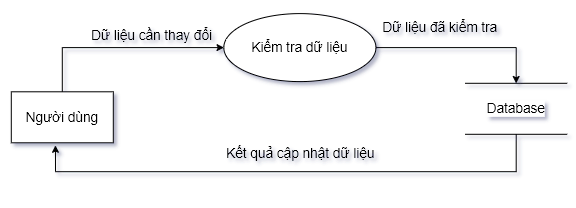
2.3.3. Sơ đồ luồng dữ liệu



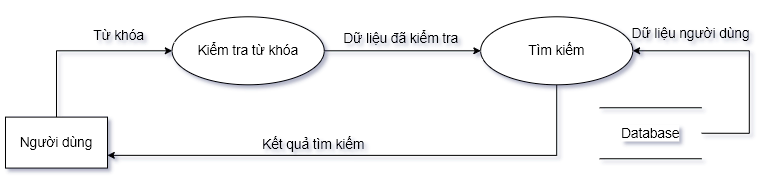
Hình 12: Sờ đồ đăng ký tài khoản



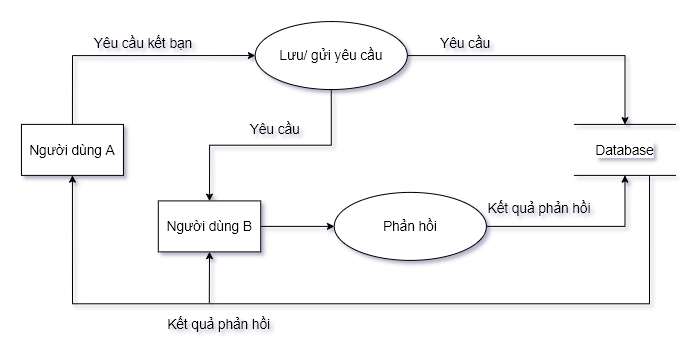
Hình 13: Sơ đồ đăng nhập



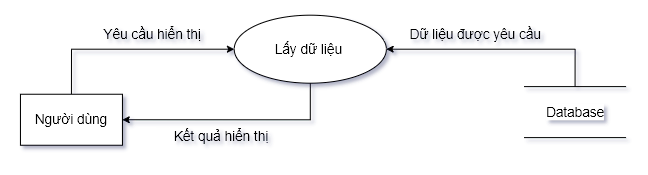
Hình 14: Sơ đồ chỉnh sửa thông tin cá nhân



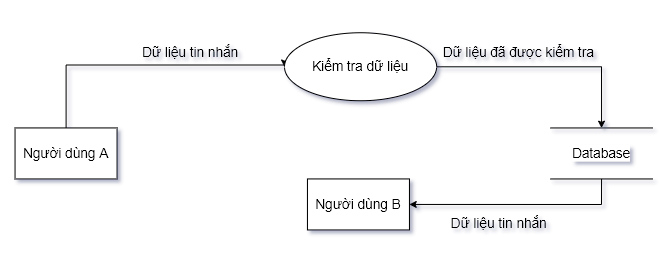
Hình 15: Sơ đồ tìm kiếm bạn bè



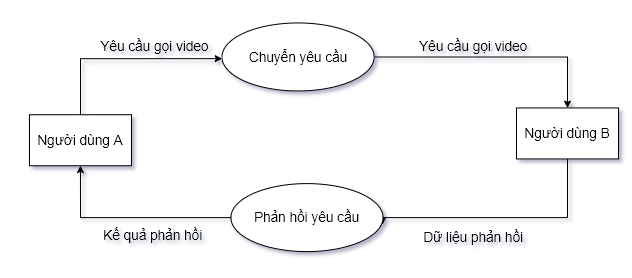
Hình 16: Sơ đồ thao tác kết bạn



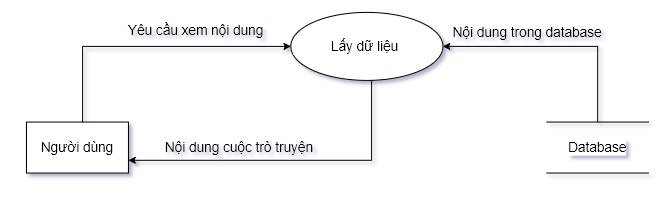
Hình 17: Sơ đồ hiển thị liên hệ



Hình 18: Sơ đồ gửi tin nhắn



Hình 19: Sơ đồ cuộc gọi video



Hình 20: Sơ đồ hiển thị nội dung cuộc trò chuyện

2.4. Phân tích, thiết kế yêu cầu an toàn

2.4.1. Mã hóa mật khẩu – Bcrypt

Bcrypt chính là thuật toán được khuyên dùng nhất khi lưu trữ mật khẩu để bảo vệ người dùng. Bcrypt là một thuật toán mã hóa 1 chiều.

Bcrypt được đánh giá là bảo mật và an toàn hơn so với MD5 và SHA bởi mỗi lần thực hiện băm nó lại cho một giá trị khác nhau, việc này khiến cho việc dò tìm mật khẩu trở nên khó khăn hơn.

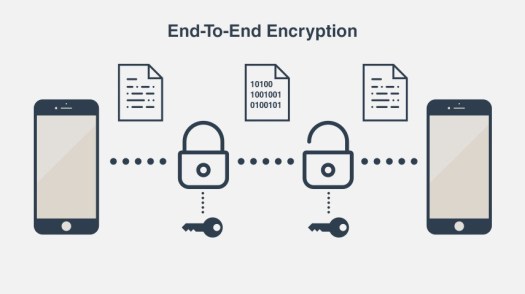
Dữ liệu mà ở đây là mật khẩu sẽ được mã hóa bằng Bcrypt rồi mới được lưu vào database để đảm bảo bất cứ ai tấn công vào cơ sở dữ liệu cũng không thể đánh cắp được mật khẩu.

2.4.2. Mã hóa dữ liệu đầu cuối( End to end encryption)

End-To-End (E2E) là một dạng mã hóa dữ liệu đầu cuối đối với những tin nhắn chat, thư email được gửi và trao đổi trên Internet.

Tất cả những trao đổi trực tiếp trên ứng dụng đó đều được mã hóa bảo vệ như:

“Tin nhắn, hình ảnh, cuộc gọi, video, âm thanh, và một vài tệp tin đính kèm khác”.



Mục đích của việc mã hóa dữ liệu E2E là bảo vệ mọi dữ liệu đang được lưu trữ trên hệ thống máy tính và những dữ liệu được truyền qua Internet hay giữa các máy với nhau. Các thuật toán mã hóa đều có tính xác thực nguồn gốc của dữ liệu gửi, đảm bảo tính toàn vẹn rằng nội dung dữ liệu đó sẽ không bị thay đổi, hủy bỏ hay thu hồi lại khi gửi dữ liệu đi.

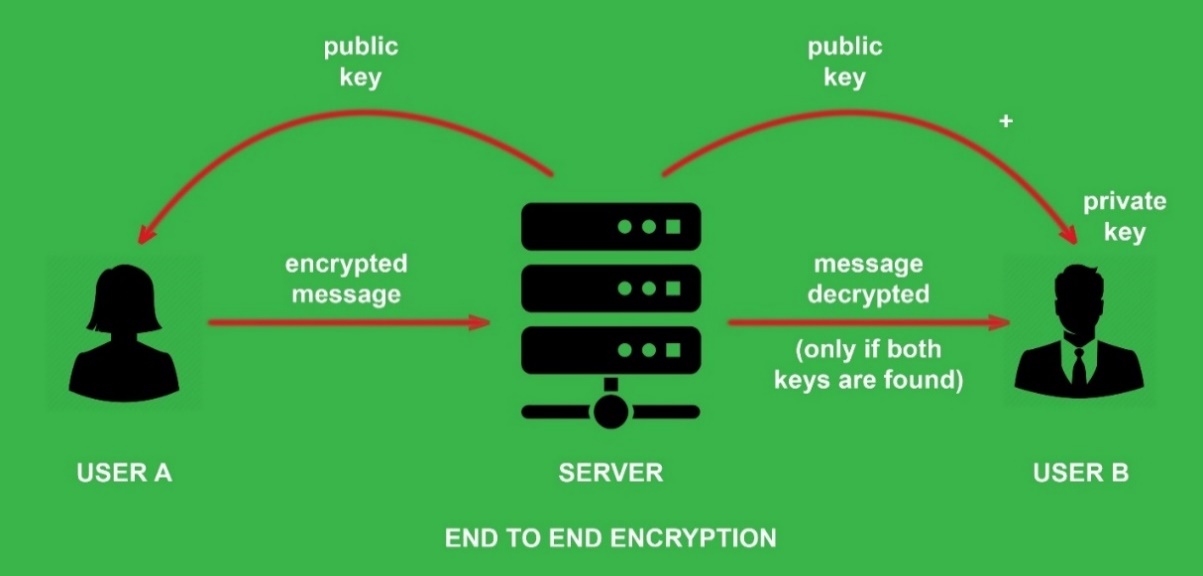
Điểm nổi trội trong mã hóa dữ liệu End – To - End Security là cho dù dữ liệu không may bị đánh cắp thì phải mất nhiều thời gian và nguồn lực cho việc giải mã hóa dữ liệu. Điều này nâng cao tính an toàn, bảo mật cho dữ liệu, tránh những trường hợp thất thoát thông tin dữ liệu quan trọng.

Quy trình mã hóa end to end encryption.

Kiến trúc E2EE thường sử dụng thuật toán mã hóa đối xứng, nhưng mã hóa Rivest-Sharmir-Adleman (RSA) được sử dụng rộng rãi để bảo vệ dữ liệu nhạy cảm, đặc biệt là khi nó được gửi qua một mạng không an toàn như Internet.

Trong đó, có hai loại khóa được sử dụng là khóa công khai(public key) và khóa riêng tư (private key).

Sự phổ biến của thuật toán này là do cả khóa công khai và khóa bí mật của nó đều có thể mã hóa dữ liệu và đảm bảo tính bí mật, toàn vẹn, xác thực và không thu hồi của dữ liệu và truyền thông kỹ thuật số thông qua việc sử dụng chữ ký số.



Khóa công khai được chia sẻ, bất kỳ người dùng nào cũng có thể lấy được khóa công khai của người khác, khóa công khai được sử dụng để mã hóa dữ liệu trước khi gửi đi.

Ngược lại, khóa riêng tư không được chia sẻ, được dùng để giả mã dữ liệu được mã hóa bỏ khóa công khai của cặp khóa này.

Như vậy, người A khi gửi tin nhắn cho người B sẽ mã hóa dữ liệu bằng một khóa công khai của người B, và người B khi nhận được tin nhắn, sẽ giải mã nó bằng mã bí mật của người B và ngược lại tương tự như vậy.

* **Mô tả sơ lược**

[RSA](http://people.csail.mit.edu/rivest/Rsapaper.pdf) là một hệ mã hóa bất đối xứng được phát triển bởi Ron Rivest, Adi Shamir và Leonard Adleman (tên của nó cũng chính là tên viết tắt của 3 tác giả này) và được sử dụng rộng rãi trong công tác mã hoá và công nghệ chữ ký điện tử. Trong hệ mã hóa này, public key có thể chia sẻ công khai cho tất cả mọi người. Hoạt động của RSA dựa trên 4 bước chính: Key generation (sinh khóa) , Key distribution (chia sẻ key), Encription (mã hóa) và Decryption (giải mã).

**Bước 1: Key generation**

Chọn ngẫu nhiên hai số nguyên tố p,q được giữ bí mật, n = pq

* n được dùng như một mô đun cho cả khóa công khai và khóa bí mật. Độ dài của nó thường được biểu diễn dạng bít là độ dài của khóa
* Tính giá trị hàm số ơ le Ø(n) = (p-1)(q-1)
* Chọn một số tự nhiên e sao cho 1< e < Ø(n) và là một số nguyên tố cùng nhau với Ø(n)
* Tính d sao cho d.e ≡ 1 (mod Ø(n) ).

Lưu ý:

* Các số nguyên tố được chọn bằng phương phát thử xác suất
* Các bước 4 và 5 có thể được thực hiện bằng giải thuật Euclid mở rộng

Khóa công khia bao gồm:

* Mô đun
* e số mũ công khai (số mũ mã hóa)

Khóa bí mật bao gồm:

* Mô đun n xuất hiện trong cả khóa công khai và khóa bị mật
* d số mũ bí mật (số mũ giải mã)

**Bước 2: Key distribution**

Nếu A muốn gửi thông tin cho B và sử dụng mã hóa RSA thì A phải biết được public key để mã hóa tin nhắn của B và B phải dùng khóa bí mật để giải mã.

Để A có thể gửi tin nhắn đã mã hóa tới B. Thì B phải gửi khóa công khai (n , e) đến A thông qua một cách đáng tin cậy nhưng không cần thiết bí mật. Khóa bí mật của B là (d) không bao giờ được chia sẻ.

**Bước 3: Encryption**

Sau khi A nhận được khóa công khai của B thì A có thể gửi tin M tới B

Để làm được điều đó trước tiên A chuyển đổi M( tin nhắn văn bản, hình ảnh giọng nói…) thành một số nguyên m sao cho 0 ≤ m < n theo một hàm có thể đảo ngược ( tức là từ m có thể chuyển ngược lại M ) được thỏa thuận trước.

Quá trình này sử dụng mã hóa AES

Lúc này A có m và biết n, e do B gửi tới. A sẽ tính c là bản mã hóa của m theo công thức c = m­e mod n sau đó A gửi c đến B

**Bước 4: Decryption**

B nhận được c từ A và biết khóa bí mật. B có thể tìm được m từ c theo công thức m= cd mod n

Quá trình giải mã hoạt động vì ta có: cd≡ (m­e)d ≡ med (mod n)

Do ed ≡ 1 nên med = m (mod p) và med ≡ m (mod p) và med ≡ m (mod q)

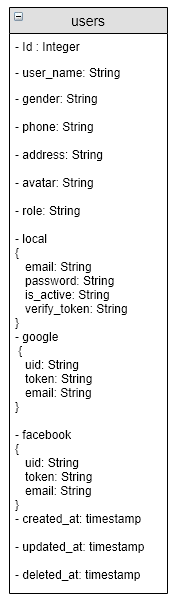
Do p và q là hai số nguyên tố cùng nhau nên áp dụng định lý số dư Trung Quốc ta có med ≡ m ( mod pq) hay cd ≡ m (mod n) biết m, B tìm lại M theo phương phái mã hóa đã thỏa thuận trước

* **Tính an toàn của RSA**

Tính an toàn của RSA chủ yếu dựa vào bộ tạo số ngẫu nhiên sinh ra 2 số nguyên tố p và q ban đầu. Việc tính ngược lại p và q từ n là chuyện hầu như không thể với hai số nguyên tố 2048 bits như đã đề cập ở trên. Nhưng việc tính ra d từ từ p và q là việc rất dễ dàng. Do đó nếu như một bên nào đó đoán ra được hoặc tìm ra lỗ hổng của bộ sinh số ngẫu nhiên đó thì coi RSA bị hoá giải.

2.5 Thiết kế cơ sở dữ liệu

2.5.1. Cơ sở dữ liệu cho service quản lý tài khoản

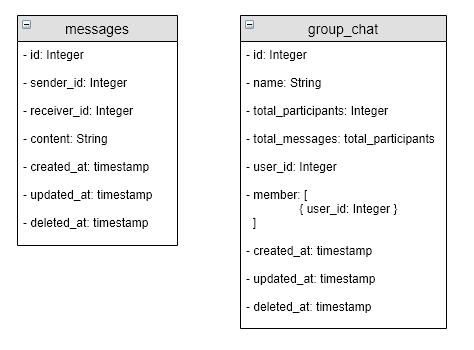


**Bảng 2.1: Lược đồ CSDL Users.**

|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
|  |  |
| id | - id của user |
|  |  |
| username | - Tên hiển thị của người dùng trong ứng dụng. |
|  |  |
| gender | - Giới tính. |
|  |  |
| phone | - Số điện thoại. |
|  |  |
| address | - Địa chỉ. |
|  |  |

|  |  |
| --- | --- |
| avatar | - Ảnh đại diện. |
|  |  |
| role | - Phân quyền người dùng, mặc định người dùng bình thường là |
|  | "customer" và quản trị viên là "admin" |
|  |  |
| local | - Loại tài khoản Local, bao gồm các document con là: |
|  | **email**: email của người dùng. |
|  | **password**: mật khẩu của người dùng. |
|  | **is\_active**: tài khoản đã được xác nhận hay chưa. |
|  | **verify\_token**: lưu token mà sẽ gửi mail về cho người dùng sử |
|  | dụng token này để xác nhận tài khoản. |
|  |  |
| facebook | - Loại tài khoản Facebook, khi người dùng đăng nhập bằng |
|  | facebook. chúng ta sẽ lưu lại: |
|  | **uid**: id của người dùng trên facebook. |
|  | **token**: token mà facebook gửi về khi xác thực. |
|  | **email**: email của người dùng. |
|  |  |
| google | - Loại tài khoản Google, khi người dùng đăng nhập bằng google. |
|  | chúng ta sẽ lưu lại: |
|  | **uid**: id của người dùng trên facebook. |
|  | **token**: token mà google gửi về khi xác thực. |
|  | **email**: email của người dùng. |
|  |  |
| created\_at | Nhãn thời gian người dùng đăng ký mới tài khoản trên hệ thống. |
|  |  |
| updated\_at | Nhãn thời gian người dùng chỉnh sửa thông tin tài khoản trên hệ |
|  | thống. |
|  |  |
| deleted\_at | Nhãn thời gian người dùng xóa bỏ tài khoản của họ trên hệ |
|  | thống. |
|  |  |

2.5.2. Cơ sở dữ liệu cho service trò chuyện



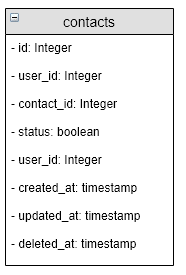
**Bảng 2.2: Lược đồ CSDL Messages.**

|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
|  |  |
| id | - id của message. |
|  |  |
| sender\_id | - id của người gửi tin nhắn. |
|  |  |
| receiver\_id | - id của người nhận tin nhắn, có thể là id của một người cụ thể |
|  | khi chat riêng hoặc là id của nhóm trò chuyện khi chat nhóm. |
|  |  |
| content | - Nội dung tin nhắn. |
|  |  |
| file | - Tin nhắn kiểu hình ảnh hoặc tệp tin. |
|  |  |
| created\_at | - Nhãn thời gian khi tin nhắn được tạo. |
|  |  |
| updated\_at | - Nhãn thời gian khi tin nhắn được chỉnh sửa. |
|  |  |
| deleted\_at | - Nhãn thời gian khi tin nhắn bị xóa |
|  |  |

**Bảng 2.3: Lược đồ CSDL Group\_chat.**

|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
|  |  |
| \_id | - id mặc định ban đầu do MongoDB tạo ra. |
|  |  |
| name | - Tên của cuộc trò chuyện nhóm. |
|  |  |
| total\_participants | - Tổng số lượng người trong nhóm trò chuyện. |
|  |  |
| total\_messages | - Tổng số tin nhắn đã chat trong nhóm trò chuyện. |
|  |  |
| user\_id | - id của người tạo nhóm |
|  |  |
| members | - lưu trữ một mảng id của các thành viên trong nhóm chat. |
|  |  |
| created\_at | - Nhãn thời gian bắt đầu cuộc trò chuyện. |
|  |  |
| updated\_at | - Nhãn thời gian cập nhật cuộc trò chuyện, khi có tin nhắn |
|  | mới. |
| deleted\_at | - Nhãn thời gian khi xóa cuộc trò chuyện. |
|  |  |
|  |  |

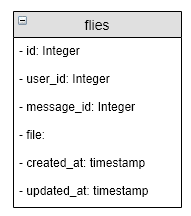
2.5.3. Cơ sở dữ liệu cho service quản lý liên lạc



**Bảng 2.4: Lược đồ CSDL Contacts*.***

|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
|  |  |
| \_id | - id của contact |
|  |  |
| user\_id | - id của người gửi lời mời kết bạn. |
|  |  |
| contact\_id | - id của người nhận lời mời kết bạn. |
|  |  |
| status | - Trạng thái của quan hệ bạn bè, true hoặc false. |
|  | - **true** có nghĩa là cả 2 bên người dùng đều đồng ý thành bạn bè. |
|  | - **false** nghĩa là có một trong 2 bên không đồng ý. |
|  |  |
| created\_at | - Nhãn thời gian 2 người thành bạn bè |
|  |  |
| updated\_at | - Nhãn thời gian người dùng chỉnh sửa quan hệ bạn bè. |
|  |  |
| deleted\_at | - Nhãn thời gian người dùng xóa quan hệ bạn bè. |
|  |  |

**2.5.4. Cơ sở dữ liệu cho service quản lý file**



**Bảng 2.5: Lược đồ CSDL Contacts.**

|  |  |
| --- | --- |
|  |  |
| **Tên trường** | **Mô tả** |
|  |  |
| id | - id của file |
|  |  |
| user\_id | - id của người gửi file. |
|  |  |
| message\_id | - id của cuộc trò truyện mà file được gửi. |
|  |  |
| created\_at | - Nhãn thời gian tạo file |
|  |  |
| updated\_at | - Nhãn thời gian update file |
|  |  |
| deleted\_at | - Nhãn thời gian xóa file |
|  |  |
|  |  |

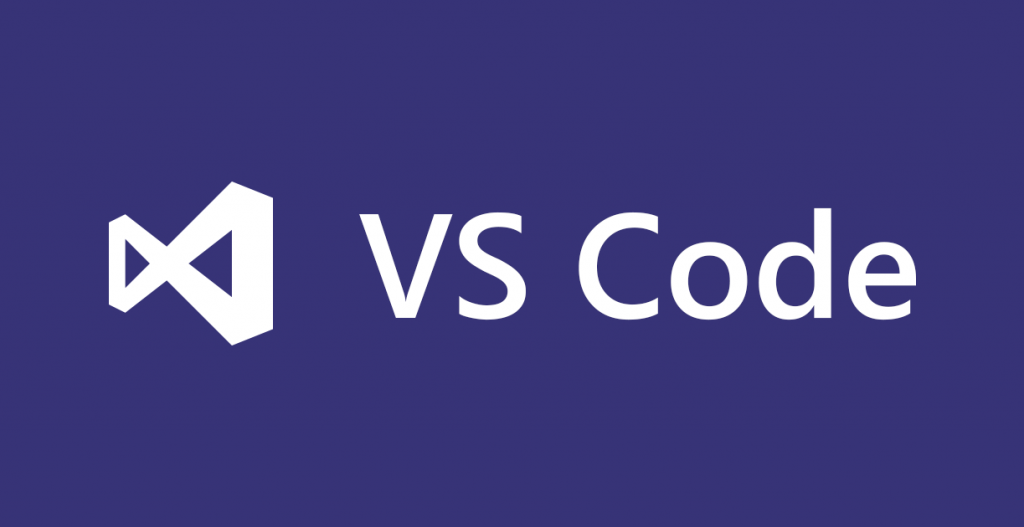
Chương 3: triển khai thực tiễn

3.1 Cài đặt môi trường phát triển

Trong dự án này, các môi trường phát triển cũng như công cụ sẽ được cài đặt trên hệ điều hành Mac

3.1.1 Cài đặt IDE Visual Studio Code (VSC)

VSC là một trình biên tập mã được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Nó hỗ trợ chức năng debug, đi kèm với Git, có syntax highlighting, tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn. Nó cũng cho phép tùy chỉnh, do đó, người dùng có thể thay đổi theme, phím tắt, và cá tùy chọn khác. Nó miễn phí và là phần mềm mã nguồn mở.



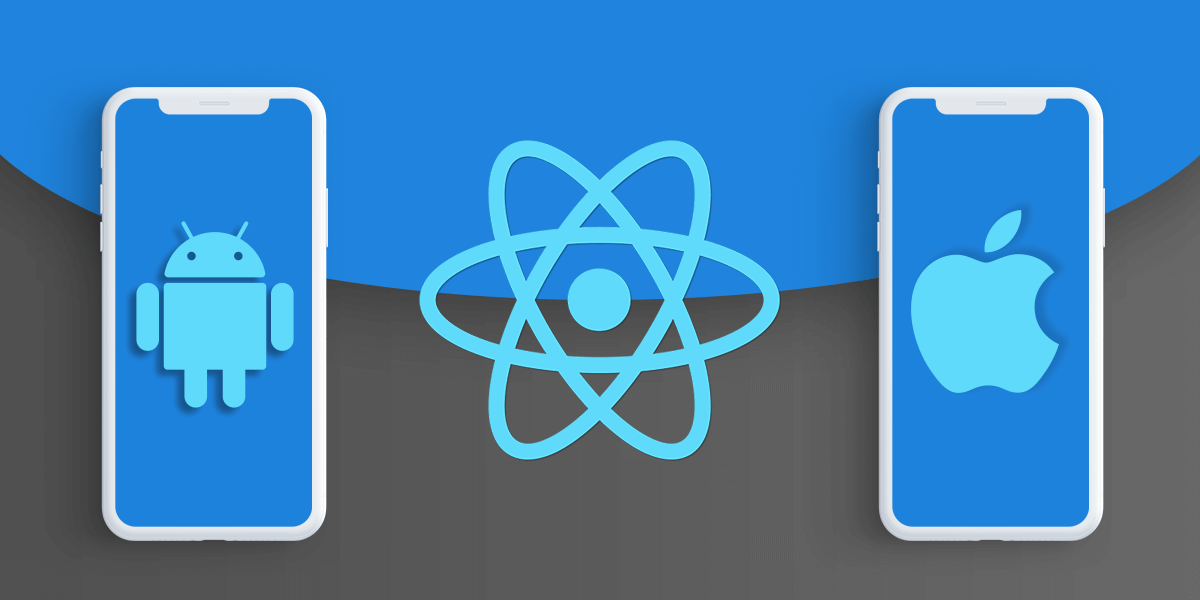
Cài đặt VSC cực kỳ đơn giản,

Bước 1: Truy cập vào trang web <https://code.visualstudio.com/download> , chọn file cài đặt cho win 10 và tải về.

Bước 2: Chạy file cài đặt vừa tải về.

Có thể cài đặt thêm một số plugin cần thiết cho VSC dễ dàng tại phần Extension để xây dựng các dự án một cách dễ dàng.

3.1.2 Cài đặt React Native



React Native là một framework do công ty công nghệ nổi tiếng Facebook phát triển nhằm mục đích giải quyết bài toán hiệu năng của Hybrid và bài toán chi phí khi mà phải viết nhiều loại ngôn ngữ native cho từng nền tảng di động.

Chúng ta sẽ build được ứng dụng Native, và chúng ta cũng có thể build ứng dụng đó một cách đa nền tảng (multi-platform) chứ không phải là một “mobile web app”, không phải là “HTML5 app”, và cũng không phải là một “hybrid app” hay cũng không chỉ build trên iOS hay Android mà chúng ta build và chạy được cả hai hệ sinh thái.

Giảm chi phí recompile của Native bằng cách sử dụng Hot-Loading tức là bạn không cần phải build lại ứng dụng từ đầu nên việc chỉnh sửa diễn ra rất nhanh chóng. Giúp cho lập trình viên có thể thấy được những chỉnh sửa của họ một cách nhanh chóng trực quan, không còn phải bỏ quá nhiều thời gian trong việc build và run ứng dụng nữa.

Và điểm lợi hại kế tiếp của React Native đó chính là chúng ta chỉ cần sử dụng JS để phát triển được một ứng dụng di động hoàn chỉnh, đồng thời giải quyết được các vấn đề mà Native App gặp phải mà mình đã nêu ở trên. Và rồi còn cả kết hợp với code native như Swift, Java, v.v…

**Native App và Hybird App**

Ngày nay chúng ta có hai nền tảng cạnh tranh chính là iOS và Android, mỗi kẻ một thế lực làm khuynh đảo giới lập trình. Từ đó sinh ra các công cụ hỗ trợ nhà phát triển ứng dụng có thể phát triển mỗi ứng dụng một cách dễ dàng chẳng hạn như Apple thì có XCode, Android thì có Android Studio.

Nhưng cũng chính vì lẽ mỗi kẻ một thế lực nên không ai chơi với ai, mỗi bên xưng đế một “phương” cuối cùng con dân lập trình viên là những người phải chịu trận. Và cũng từ lẽ đó mà giang hồ sinh ra hai môn vang danh thiên hạ là Native và Hybrid.

Trước khi học lập trình React Native thì chúng ta nên biết Hybrid và Native là gì và những mặt lợi và hại giữa hai platform này. Thông qua đó chúng ta sẽ biết được tại sao mà **React/React Native** lại được khai sinh ra đời để giải quyết vấn đề nan giải của con dân lập trình viên di động.

**Hướng dẫn cài đặt**

Làm theo hướng dẫn trên trang chủ của React-Native:

(<https://reactnative.dev/docs/environment-setup>)

MACOS:

**OS**

Bước 1: Cài đặt HomeBrew



/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

Chạy câu lệnh này trên terminal

Bước 2: Cài đặt Node, Watchman, Cocoapods

brew install node

brew install watchman

brew install cocoapods

Bước 3: Cài đặt Xcode trên App Store của Mac

Bước 4: Cài đặt React-Native-CLI

npm install –g react-native-cli

**ANDROID**

Làm tương tự bước 1 và 2 ở IOS

Bước 3: Cài đặt Java Development Kit

Brew cask install adoptopenjdk/openjdk/adoptopenjdk8

Bước 4: Cài đặt Android Studio

Link download (<https://developer.android.com/studio/index.html>)

Sau khi cài đặt chúng ta cần export thư mục android home

export ANDROID\_HOME=$HOME/Library/Android/sdk

export PATH=$PATH:$ANDROID\_HOME/emulator

export PATH=$PATH:$ANDROID\_HOME/tools

export PATH=$PATH:$ANDROID\_HOME/tools/bin

export PATH=$PATH:$ANDROID\_HOME/platform-tools

Bước 5: Cài đặt React-Native-CLI

npm install –g react-native-cli

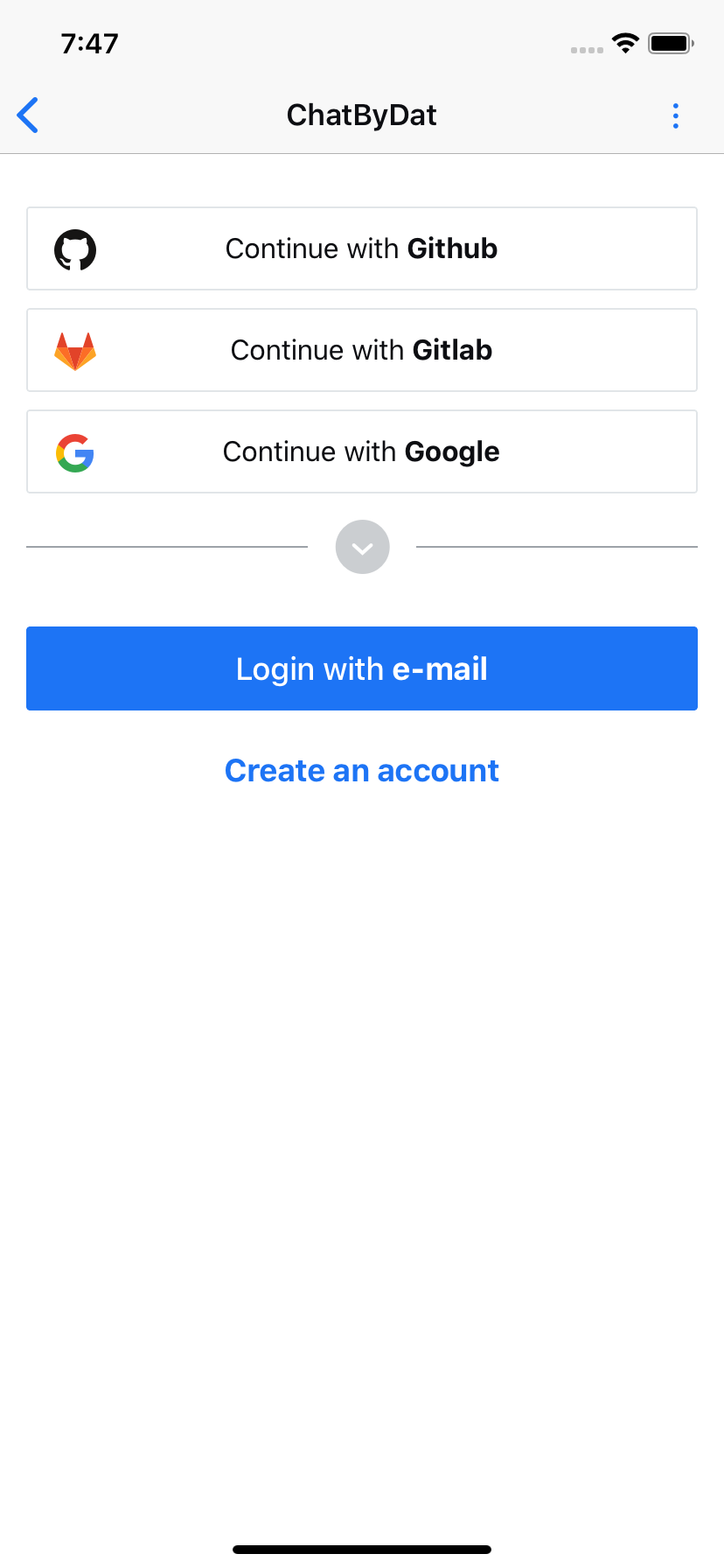
Chạy ứng dụng

Clone project bằng đường dẫn (<https://github.com/datfc97pro/AppChat.git>)

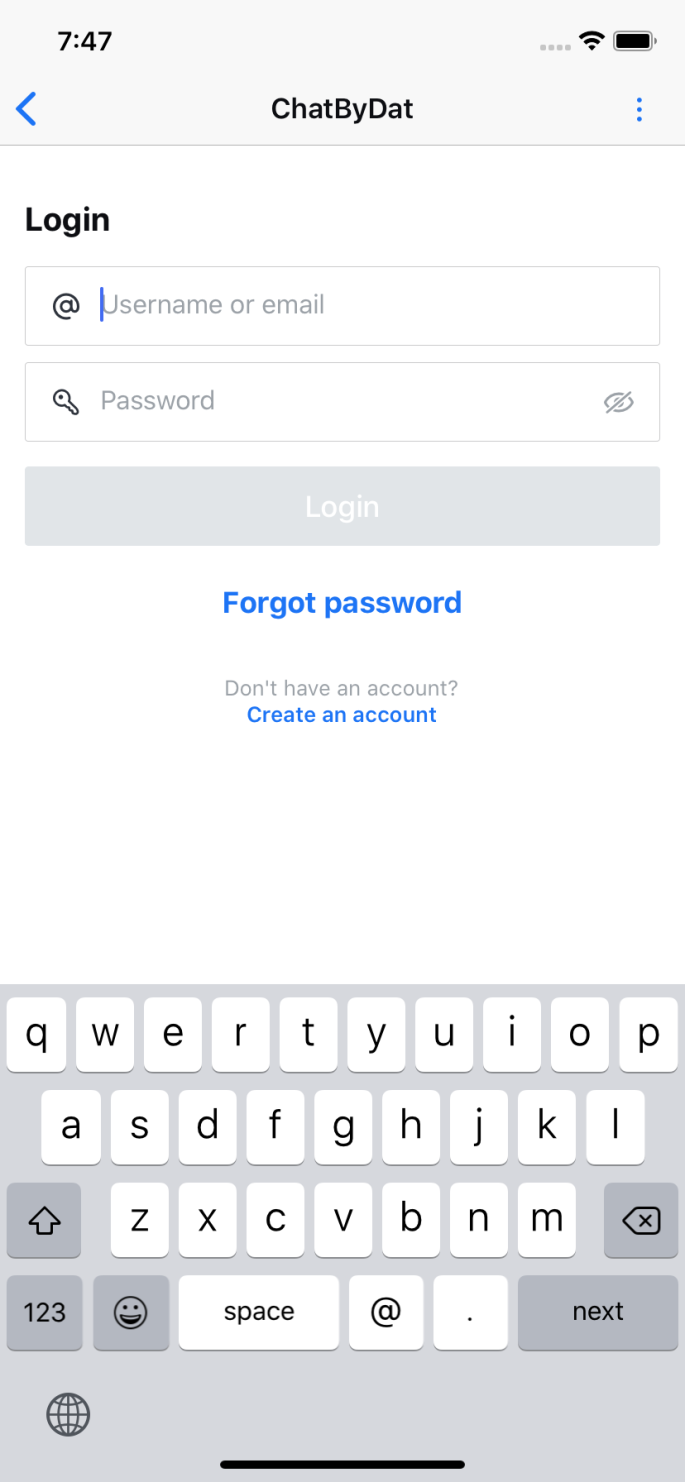
Cài đặt các thư viện cần thiết bằng câu lệnh: npm install

Chạy project bằng câu lệnh: npm run ios/ npm run android

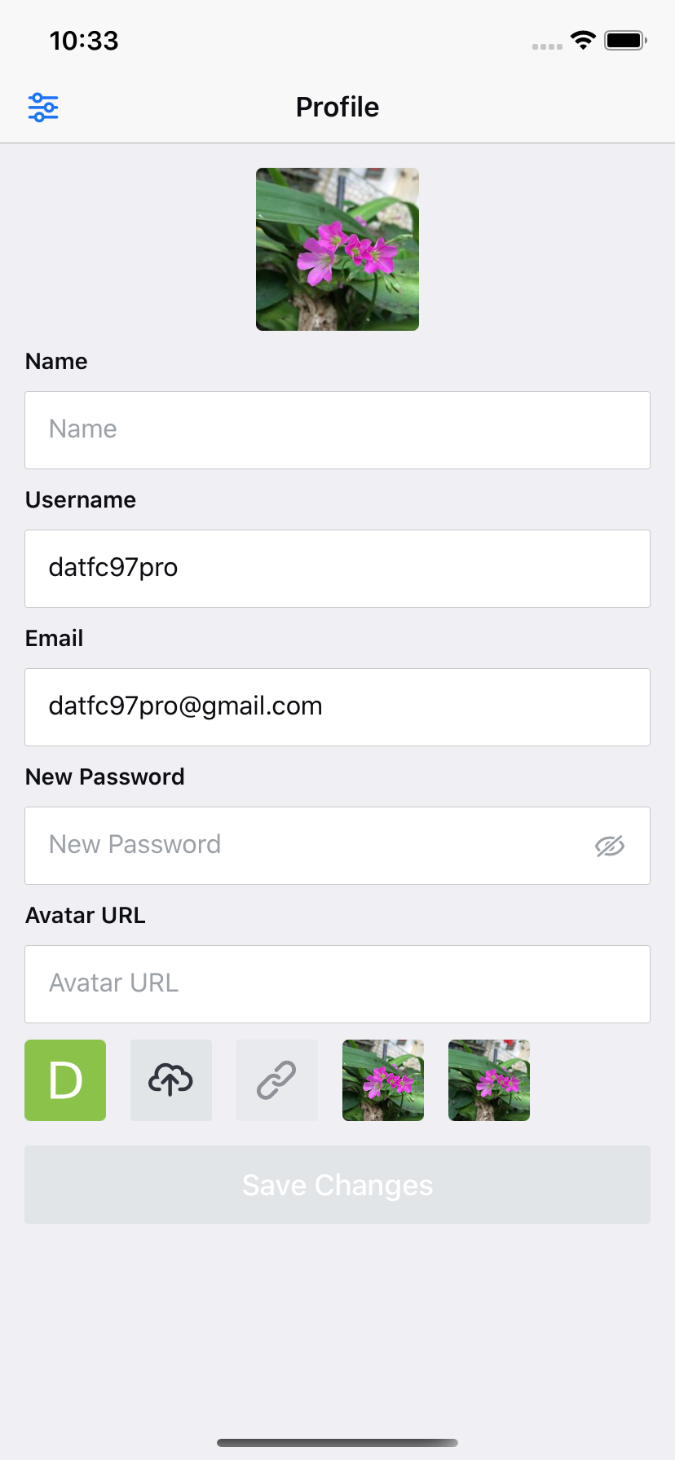
3.2 Thiết kế giao diện



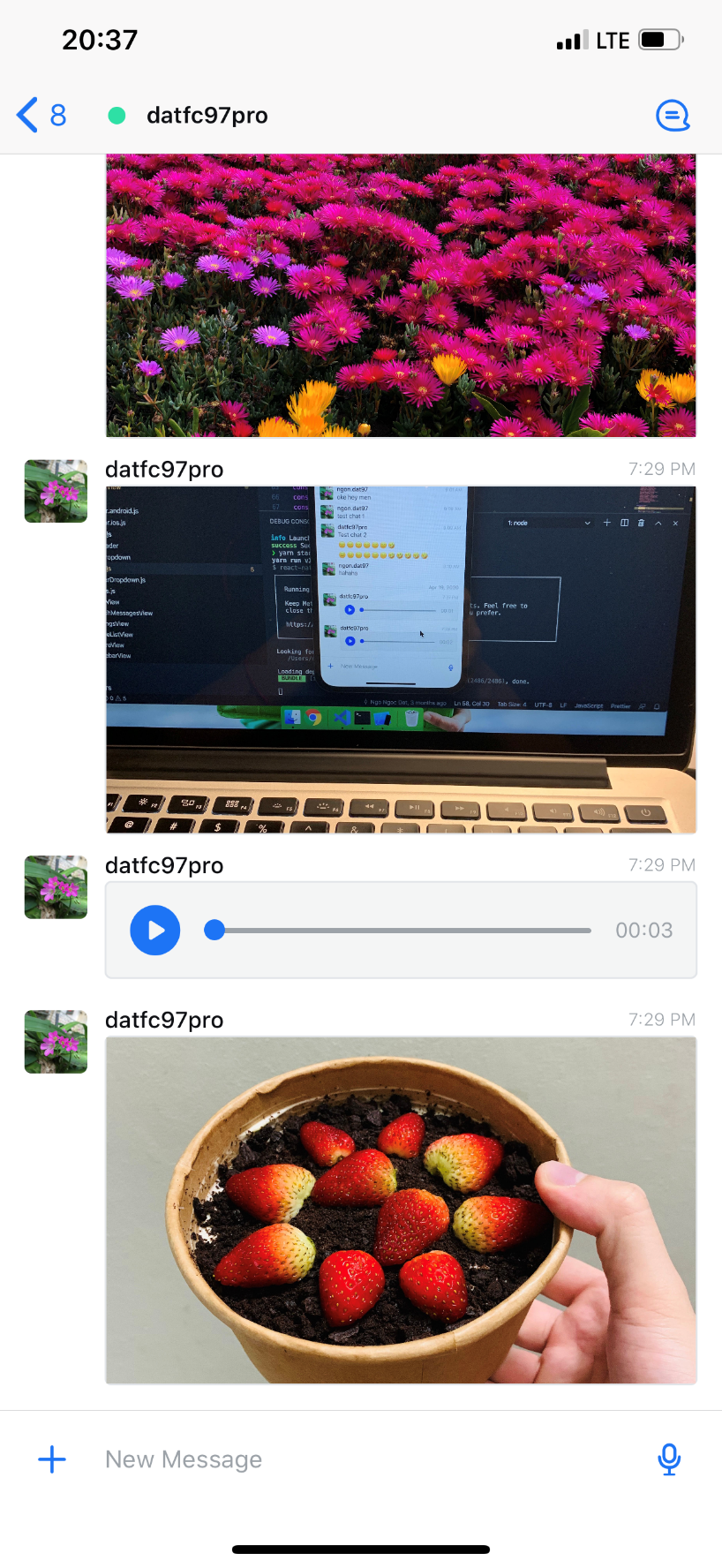
Màn hình đăng nhập với mạng xã hội



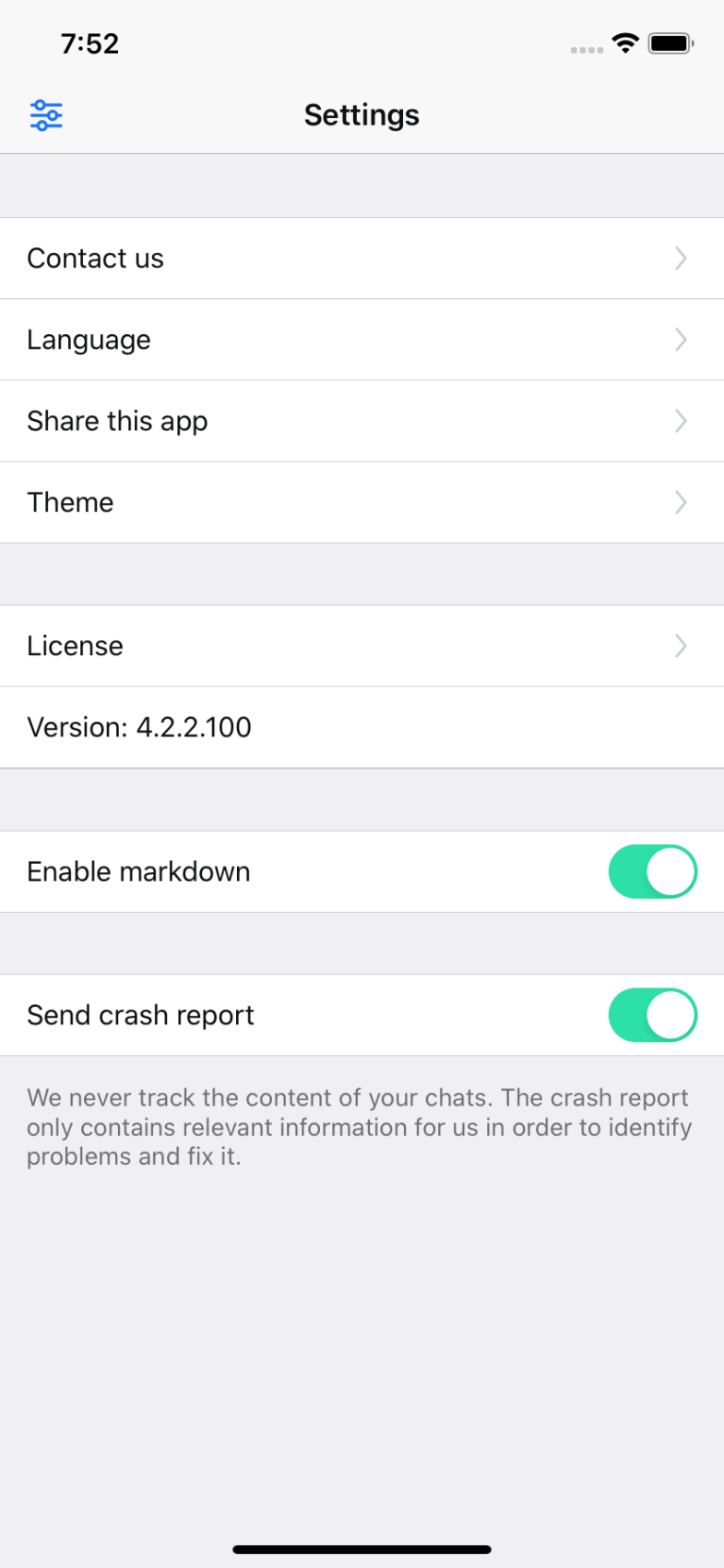
Màn hình đăng nhập



Thông tin tài khoản



Màn hình trò chuyện



Màn hình cài đặt

Kết luận

Sau khi trình bày các kiến thức cơ bản về việc xây dựng app mobile và nêu ra những giải pháp an toàn khi cho ứng dụng, đồ án có những kết quả và hạn chế sau:

Kết quả:

- Giới thiệu React-Native

- Tìm kiểu kiến trúc Client/Server

- Mô hình truyền tin Socket

- Mã hóa mật khẩu – Bcrypt

Hướng phát triển trong thời gian tới:

Trong quá trình thực hiện đồ án, do hàn chế về kiến thức và kinh nghiệm thực tế nên đồ án của em còn nhiều thiếu sót, em rất mong được sự đánh giá và góp ý của thầy cô và các bạn. Một lần nữa em xin cảm ơn thầy Lê Bá Cường đã hướng dẫn tận tình và các thầy cô, bạn bè đã giúp đỡ em hoàn thành đồ án này.

*Em xin chân thành cảm ơn!*

**SINH VIÊN THỰC HIỆN**

**Ngô Ngọc Đạt**

Tài liệu tham khảo

[1] Sách Learning React Native Building Navtive Mobile Apps with JavaScript Tác giả: Bonnie Eisenman xuất bản 3/12/2015

[2] Tài liệu tại trang chủ của React-Native: (<https://facebook.github.io/react-native/>*)*

[3] Deep Linking React-Native: (<https://medium.com/react-native-training/deep-linking-your-react-native-app-d87c39a1ad5e>)

[4] Authentication với SAML: (<https://auth0.com/blog/how-saml-authentication-works/>)

[5]  Đa ngôn ngữ : (<https://github.com/AlexanderZaytsev/react-native-i18n>)

[6] Nghiên cứ về audio trong chat (<https://github.com/jsierles/react-native-audio>)

[7] Nghiên cứu về video trong chat (<https://github.com/react-native-community/react-native-video>)

[8] React-Native-Firebase: (<https://rnfirebase.io/>)

[9] Cách build ứng dụng android: (<https://www.instamobile.io/android-development/generate-react-native-release-build-android/>)

[10] Cách build ứng dụng ios: (<https://medium.com/better-programming/create-ipa-and-apk-from-react-native-72fe53c6a8db>)