**Tài Liệu Thiết Kế Hệ Thống**

1. Thành phần hệ thống

Gồm 2 thành phần chính là phần giao diện người dùng và phần máy chủ để xử lí và lưu trữ dữ liệu

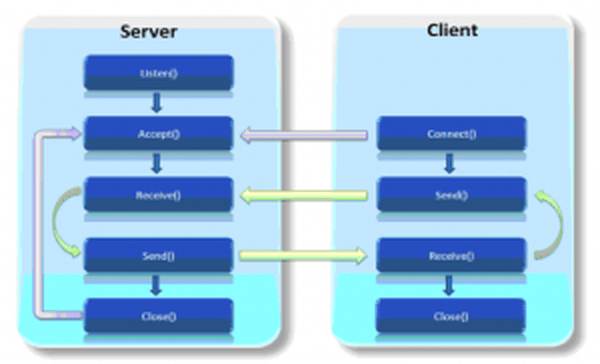
* Về giao diện người dùng:

Ngôn ngữ native mới có khả năng truy cập các phần cứng (như camera, GPS, thiết bị thu âm… ) và có thể truy cập đến các quyền như : sao chép, tạo, ghi, đọc tập tin (trên bộ nhớ của điện thoại), các thông tin danh bạ, thông tin cá nhân, thực hiện cuộc gọi, nhắn tin… Nói cách khác, khi sử dụng ngôn ngữ native thì ứng dụng có thể khai thác được sức mạnh phần cứng nhiều nhất của nền tảng.

React native và javascript chúng ta có thể dễ dàng xây dựng giao diện người dùng.

* Server: Gồm API(nhiệm vụ kết nối phía người dùng và hệ thống cơ sở dữ liệu) và cơ sở dữ liệu

1. Biểu diễn các thành phần bằng các hình vẽ



1. Mô tả về các chức năng

* Phần Server: đảm nhiệm lắng nghe ở một cổng nào đó và thực hiện các kết nối của các client với nhau hoặc giải phóng kết nối khi một client ngắt kết nối với server. Có chức năng để thực hiện kết nối và nói chuyện riêng giữa hai người với nhau hoăc nói chuyện công khai giữa các thành viên với nhau.
* Phần Client: kết nối tới Server, nếu kết nối thành công thì thông báo cho người sử dụng biết để họ có thể gửi message. Nhận message tử Server gửi về và hiển thị chat lên cửa sổ của Client
* Mô tả chức năng chi tiết:
* Nhóm chat:

+ Tạo nhóm

+ Xóa nhóm

+ Làm admin

+ Bắt đầu trò chuyện

+Mở cuộc trò chuyện

+Xóa cuộc trò chuyện

* Chat:

+ Lưu dữ liệu

+Đến phòng chat

+Chia sẻ thông tin

+Chia sẻ vị trí

+Chia sẻ ảnh hoặc video

* Người dùng:

+Bắt đầu trò chuyện

+Mở cuộc trò chuyện

+Xóa cuộc trò chuyện

* Người dùng:

+Tạo tài khoản

+Xóa tài khoản

* Đăng nhập:

+Đăng nhập

+Đăng kí tài khoản

* Thông tin cá nhân

+Thay đổi trạng thái

+Thay đổi hình ảnh đại diện

+Thay đổi tên

+Thay đổi số điện thoại

* Trợ giúp người dùng

+Tìm kiếm thông tin

* Phương thức kết nối

+Thêm kết nối bằng điện thoại

+Thêm kết nối bằng email

+Tìm kiếm liên hệ

+Xóa

+Cập nhật

+Mời người tham gia nhóm

1. Diagram

   Description automatically generatedClass diagram
2. Diagram, schematic

   Description automatically generatedSequence diagram
3. Mô tả cơ sở dữ liệu

* Cài đặt và thiết kế cơ sở dữ liệu cho chức năng gửi và nhận tin nhắn.

Với yêu cầu xây dựng một hệ thống cơ sở dữ liệu để máy khách và máy chủ có thể gửi và nhận dữ liệu liên tục với độ trễ thấp nhất thì lựa chọn của tôi ở đây là sử dụng Cloud FireStore – một dịch vụ của FireBase để xây dựng hệ thống cơ sở dữ liệu.

Trước tiên thì Cloud FireStore là cơ sở dữ liệu dựa trên đám mây NoQuery, đồng bộ hóa dữ liệu trên tất cả các máy khách trong thời gian thực và cung cấp chức năng ngoại tuyến. Dữ liệu được lưu trữ trong cơ sở dữ liệu thời gian thực dưới dạng các bộ sưu tập ( Collection ) và tất cả các máy khách được kết nối chia sẻ một phiên bản cơ sở dữ liệu, tự động nhận các bản cập nhật với dữ liệu mới nhất.

Sử dụng Cơ sở dữ liệu thời gian thực Firebase từ góc độ lập trình rất khác so với những gì ta có thể được sử dụng với các cơ sở dữ liệu quan hệ truyền thống như MySQL hoặc SQL Server. Về cơ bản, ta sẽ cấu trúc dữ liệu của mình theo một cách khác đó là sử dụng các bộ sưu tập (Collection) thay vì các bảng và phép nối truyền thống.

Ta cũng sẽ thấy rằng từ góc độ kiến trúc, cách tiếp cận sẽ khác biệt đáng kể. Thay vì cách xây dựng truyền thống, với cơ sở dữ liệu được bao quanh bởi một số lượng lớp truy cập dữ liệu mà ta chạy truy vấn hoặc thủ tục được lưu trữ. Và có một điều là tất cả mã khi viết cho cơ sở dữ liệu thời gian thực Firebase nằm trong ứng dụng máy khách. Nó giúp cho việc xây dựng và duy trì các ứng dụng đơn giản hơn nhiều.

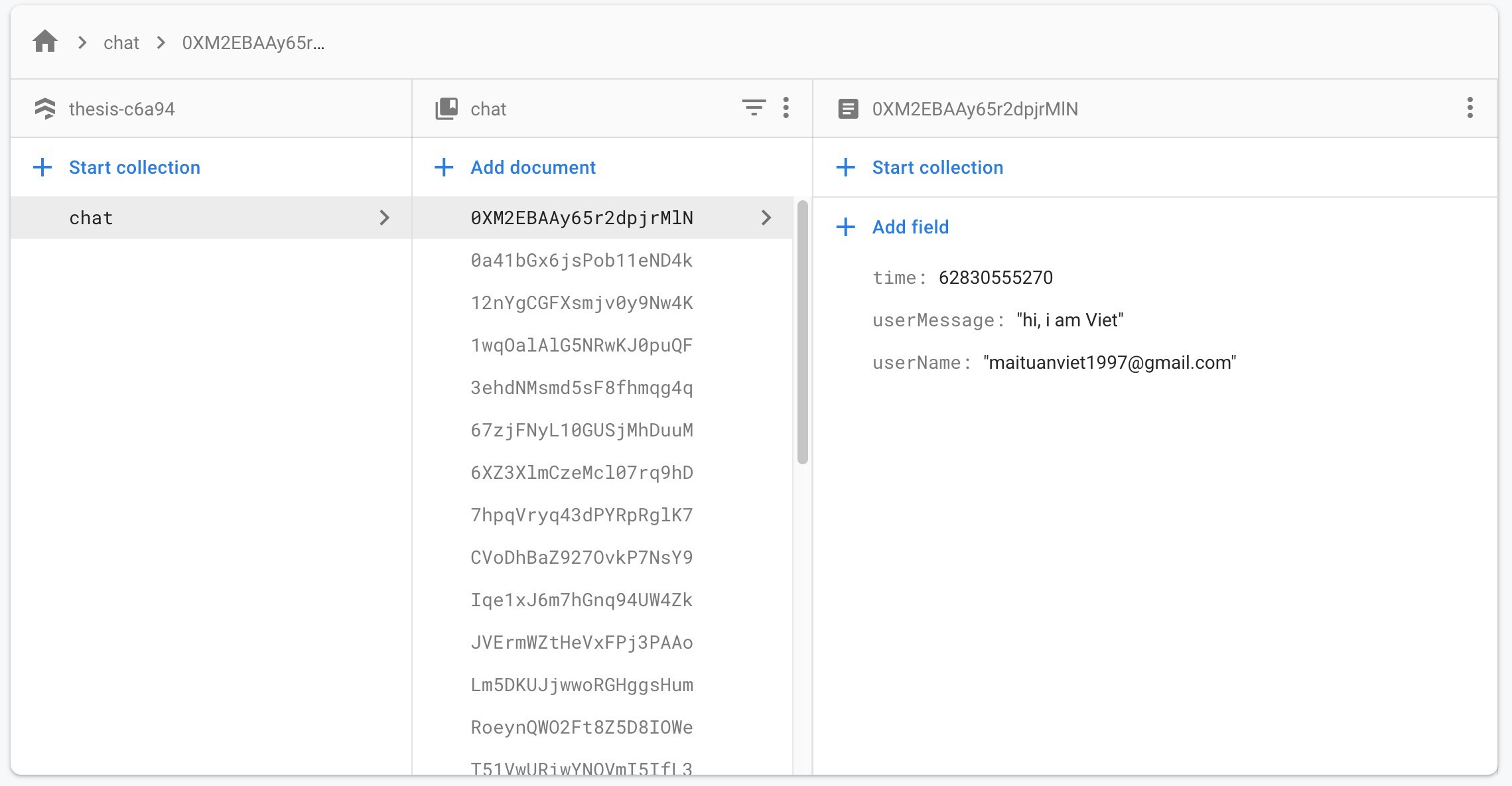
Trước tiên, ta cần truy cập vào Firebase console để bật trước khi thêm vào ứng dụng đoạn mã tham chiếu đến cơ sở dữ liệu mà ta đã đăng ký trên FireStore :

const ref = firestore().collection('chat');

Vậy là đã xong, giờ đây ứng dụng chat đã được tham chiếu đến cơ sở dữ liệu đã được đăng ký trên FireBase. Từ đây thì mỗi khi người dùng gửi tin nhắn thì

chúng ta sẽ cần sẽ dụng đoạn mã ref.add để đây đẩy lên cơ sở dữ liệu thông tin tin nhắn của người dùng. ở đây dữ liệu được gửi lên và lưu lại là Thời gian (Time), tin nhắn (userMessage), và tên người gửi ( UserName ) .

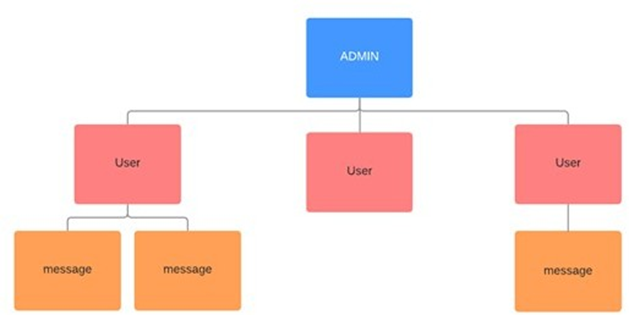
Trong hình dưới là bảng chứa các tài liệu và bộ sưu tập – cái mà được lưu trong Firestore của FireBase trong hệ thống chat.



*Bảng lưu trữ tin nhắn của người dung*

Và cùng với đó thì ta cần thực thi đoạn mã ref.onSnapshot để mỗi khi có sự thay đổi ở cơ sở dữ liệu thì dữ liệu mới sẽ được gửi trở lại cho tất cả máy khách. Và điều này chính là cơ sở để tạo nên hệ cơ sở dữ liệu thời gian thực

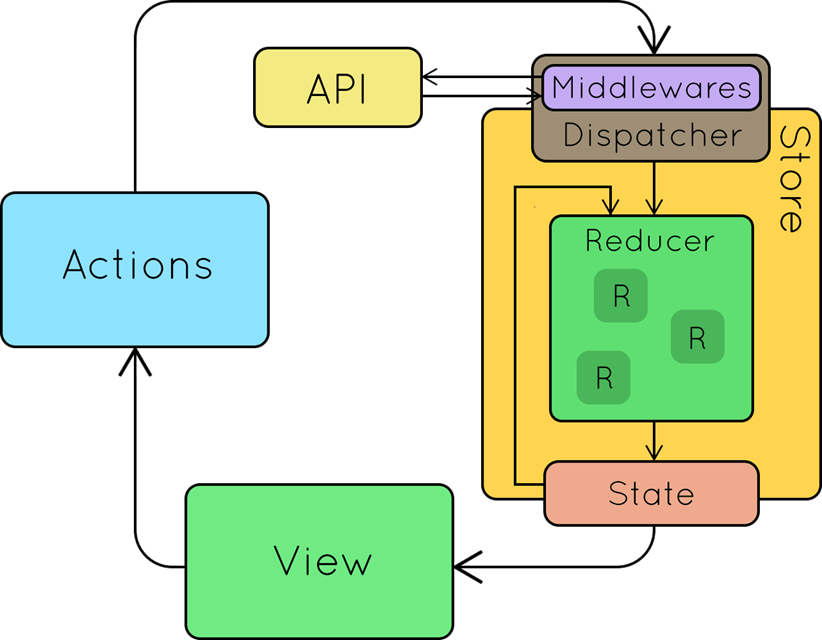
* Sơ đồ quản trị



*Sơ đồ vai trò của quản trị viên và người dùng*

* Kiến trúc quản lý dữ liệu

Trong ứng dụng Chat, để xử lý một lượng dữ liệu rất lớn chúng ta cần phải tuân thủ kiến trúc quản lý dữ liệu,và ở đây thì Redux Sau đây là sơ đồ đường đi của dữ liệu trong hệ thống :



*Sơ đồ kiến trúc dữ liệu Redux*

* Action

Ở đây action chính là hoạt động của người dùng, khi người dùng nhấn nút gửi tin nhắn hay chẳng hạn như đăng nhập , đăng ký tài khoản thì một action – hành động sẽ được gửi đến middlewares , và cấu trúc của action chính là một đối tượng (object ) có chứa tên của action đó và dữ liệu của người sử dụng.

export const setAvatarAction = createTypeAction('SET\_AVATAR');

* Reducer

Reducer là nơi duy nhất để chúng ta có thể sửa, tính toán, thay đổi dữ liệu. Ở đây chúng sẽ nhận dữ liệu từ action mà chúng ta vừa gửi lên rồi thay đổi nó trước khi lưu lại vào store . Và chúng ta sẽ dùng reducer để lưu lại thông tin cá nhân của người sử dụng như là Email và thông tin danh bạ.

* Middlewares

Middlewares là một tầng nằm giữa reducer và store nó được sinh ra là bởi vì reducer của chúng ta chỉ có thể xử lý dữ liệu theo bất đồng bộ. vì thế nó được sinh ra là để xử lý dữ liệu đồng bộ như là gọi api , gửi nhận request đến server.

* Store

Store là nơi lưu trữ toàn bộ dữ liệu của ứng dụng. Và store thì được nằm ở ngoài sơ đồ cây giữa các thành phần trong ứng dụng để mọi thành phần của ứng dụng đều có thể truy cập đến nó để lấy dữ liệu. Ví dụ:

export const InitialState = Record<IStore>({ profileState: createInitProfileState(),

})();

export const rootReducer = combineReducers(

{

profileState: profileReducer,

},

InitialState,

);

export const configureStore = (initialStateMap = InitialState) => { const initialState = Record<IStore>({

profileState: initialStateMap.profileState || createInitProfileState(),

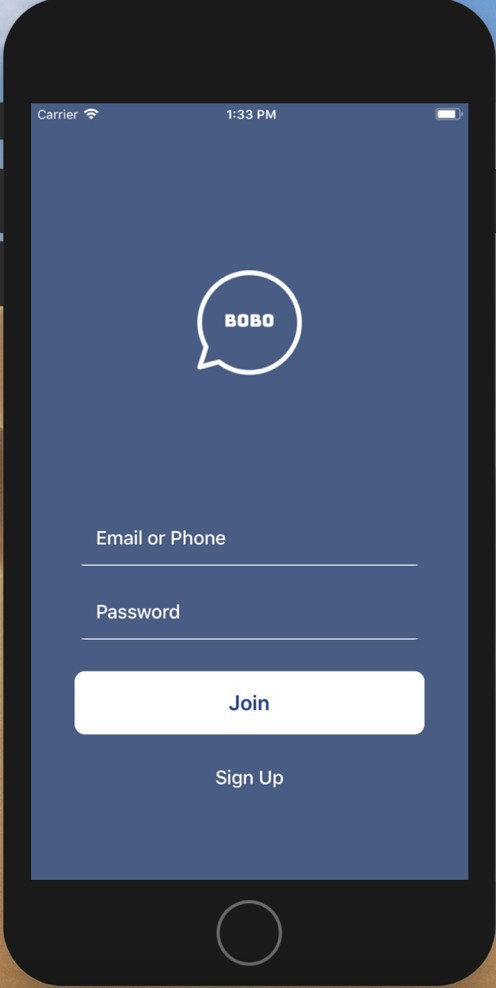
})();

return createStore(rootReducer, initialState);

};

export const store = configureStore();

1. Mô tả về giao diện người dùng

* Màn hình đăng nhập :gồm logo của ứng dụng và hai Text Input cùng một nút Join để người dùng đăng nhập vào ứng dụng và một nút sign Up để dẫn đến màn hình đăng kí .

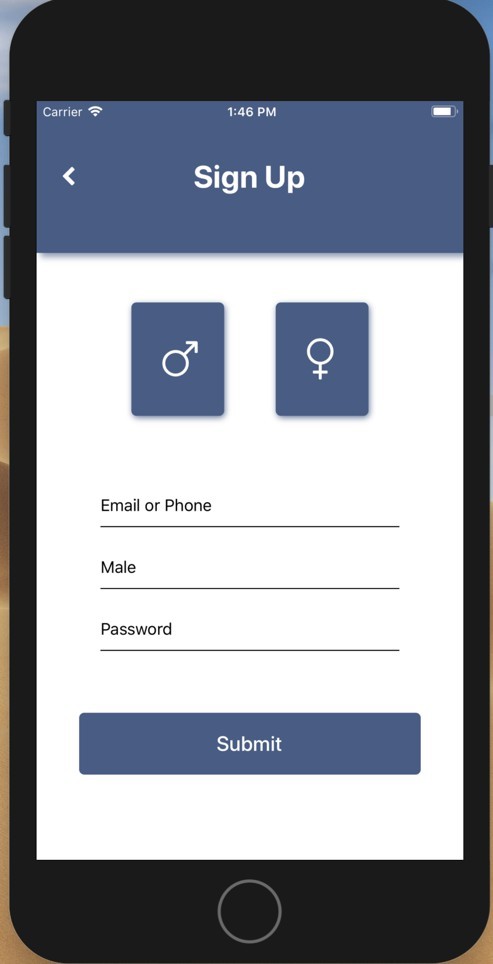
*Màn hình đăng nhập*

Màn hình Login được viết thành một Component và Render với các thành phần là Text Input , Touchable và View, mỗi khi người dùng thay đổi giá trị trong Text Input thì giá trị đó sẽ được lưu lại vào async Storage của thiết bị trước khi được gửi lên server để xác thực . Và ở đây , chúng ta sẽ sử dụng thư viện axios – một thư viện cung cấp các phương thức để phía client có thể giao tiếp được với server . một số phương thức có thể kể đến ở đây như là Post , Put , Get , Delete .. Và khi đó từ màn hình đăng nhập khi server trả về dữ liệu là True với status 200 thì người dùng sẽ được chuyển đến màn hình chứa nội dung chat . Nếu dữ liệu là false sẽ trả về cảnh báo sai mật khẩu hoặc email cho người dùng .

* Xây dựng màn hình đăng ký :

Màn hình chính sẽ là một Navigation Tab bao gồm 3 mục : danh mục người dùng trò chuyện gần đây , Danh bạ , Profile cá nhân .

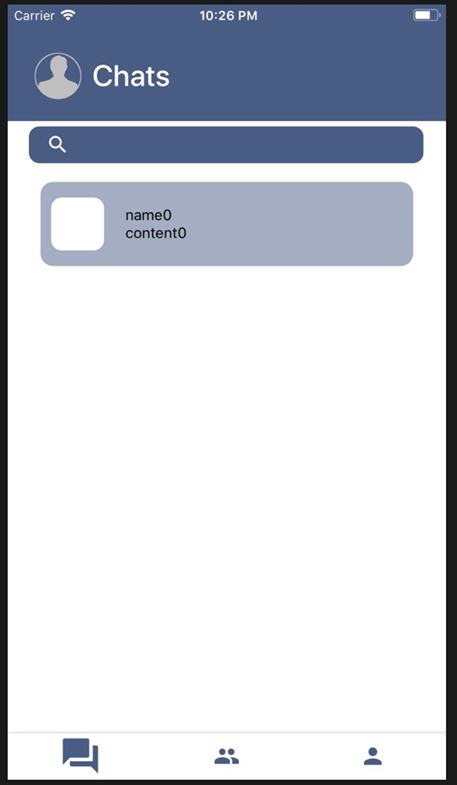
Trước hết là danh mục người dùng trò chuyện gần đây , màn hình được xây dựng với 1 Text Input với chức năng tìm kiếm và 1 FlatList để hiện thị các cuộc trò chuyện , dữ liệu truyền vào là dữ liệu được trả về từ máy chủ.



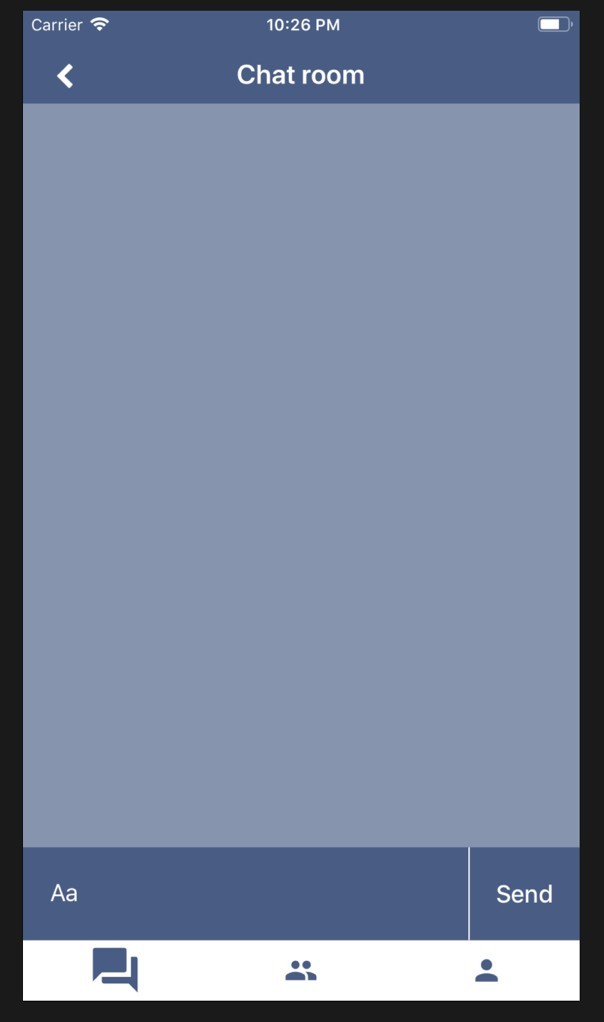
* Màn hình chính:

Màn hình chính sẽ là một Navigation Tab bao gồm 3 mục : danh mục người dùng trò chuyện gần đây , Danh bạ , Profile cá nhân .

Trước hết là danh mục người dùng trò chuyện gần đây , màn hình được xây dựng với 1 Text Input với chức năng tìm kiếm và 1 FlatList để hiện thị các cuộc trò chuyện , dữ liệu truyền vào là dữ liệu được trả về từ máy chủ.

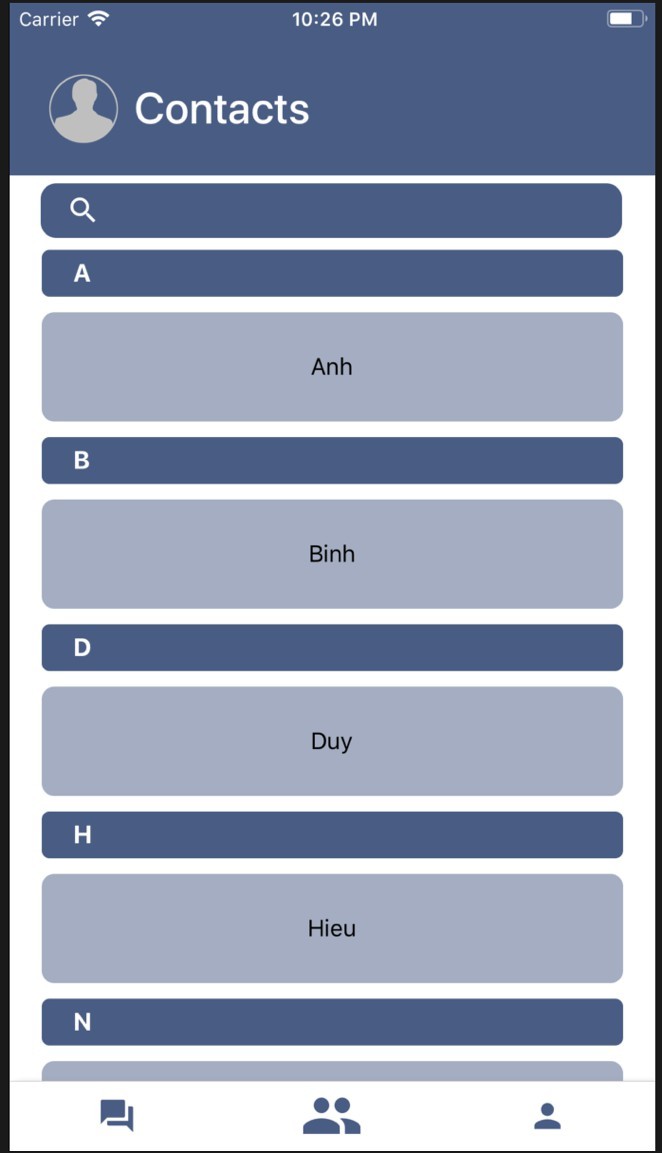


* Phòng chat

Người dùng sẽ được chuyển đến màn hình phòng chat riêng khi họ chạm vào tên một người dùng bất kỳ trong danh bạ hoặc trong màn hình người dùng gần đây. Ở màn hình này , sau khi người dùng nhập và chạm vào nút Send

* Màn hình danh bạ :

Bên cạnh là Tab màn hình danh bạ. dữ liệu khi truyền vào sẽ đi qua hàm xử lý với các vòng lặp và các hàm xử lý chuỗi để có thể sắp xếp các phần tử trong mảng sao cho đúng với cấu trúc của một danh bạ là sắp xếp từ A – Z .



*Màn hình danh bạ*

Và cuối cùng là màn hình Profile với chức năng cập nhật và thay đổi thông tin người dùng , thông tin người dùng sau khi thay đổi sẽ được gửi 1 action và chúng ta sẽ lắng nghe hành động này và lưu lại trong store theo kiến trúc Redux

1. Phân chia công viêc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tuần | Nguyễn Thị Thảo | Vũ Đức Long | Trần Thị Thu | Phạm Duy Phương |
| 1 | Phân chia công việc  Tìm tài liệu | Làm class diagram và sequence diagram | Tổng hợp và làm word | Tìm tài liệu và tổng hợp |
| **2** | Làm giao diện đăng kí, đăng nhập và phát triển chức năng | Làm giao diện màn hình chính và phát triển các chức năng | Làm giao diện phòng chat và phát triển các chức năng | Làm giao diện contact và phát triển các chức năng |
| 3 | Phát triển các chức năng và ghép hệ thống | Phát triển các chức năng | Phát triển các chức năng | Phát triển các chức năng |
| 4 | Tổng hợp và đánh giá | Hoàn thiện | Hoàn thiện | Hoàn thiện |