

BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP HCM
KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM



BÁO CÁO ĐỒ ÁN MÔN HỌC

**ĐỀ TÀI: ỨNG DỤNG FLUTTER XÂY DỰNG ỨNG DỤNG
CHATAPP TRÊN THIẾT BỊ DI ĐỘNG**

Giảng viên hướng dẫn: **ThS. Nguyễn Thanh Truyền**

Sinh viên thực hiện: **Nguyễn Thị Ngọc Huyền**

MSSV: 0850070020

Nguyễn Hưng Quốc

MSSV: 0850070072

Nguyễn Phước Triều

MSSV: 0850070059

Võ Thái Tài

MSSV: 0850070047

Trần Thanh Tuyên

MSSV: 0850070065

Phạm Thị Minh Xuân

MSSV: 0850070069

Lớp:

08_ĐH_TTMT

TP. Hồ Chí Minh, tháng 8 năm 2023

BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP HCM
KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THĂM



BÁO CÁO ĐỒ ÁN MÔN HỌC

**ĐỀ TÀI: ỨNG DỤNG FLUTTER XÂY DỰNG ỨNG DỤNG
CHATAPP TRÊN THIẾT BỊ DI ĐỘNG**

Giảng viên hướng dẫn: **ThS. Nguyễn Thanh Truyền**

Sinh viên thực hiện: **Nguyễn Thị Ngọc Huyền**

MSSV: 0850070020

Nguyễn Hưng Quốc

MSSV: 0850070072

Nguyễn Phước Triều

MSSV: 0850070059

Võ Thái Tài

MSSV: 0850070047

Trần Thanh Tuyền

MSSV: 0850070065

Phạm Thị Minh Xuân

MSSV: 0850070069

Lớp: **08_ĐH_TTMT**

TP. Hồ Chí Minh, tháng 8 năm 2023

BẢNG PHÂN CÔNG NHIỆM VỤ

Họ tên thành viên	Công việc	Đóng góp % (100%)
Nguyễn Thị Ngọc Huyền (Nhóm trưởng)	<ul style="list-style-type: none">- Lấy dữ liệu từ Server bằng cách Call API sử dụng Networking & HTTP, JSON. (Chương 1)- Chương 3,4.- Tổng hợp bài nhóm.- Viết báo cáo Word.	20%
Trần Thanh Tuyền	<ul style="list-style-type: none">- Xử lý kết nối, dữ liệu realtime giữa Client – Server.(Chương 1)	15%
Phạm Thị Minh Xuân	<ul style="list-style-type: none">- Giao thức Websocket sử dụng thư viện Socket IO. (Chương 1)	15%
Nguyễn Hưng Quốc	<ul style="list-style-type: none">- Kết nối mạng xử lý lỗi.- Xử lý phản hồi từ Server.- Asynchronous.	20%
Võ Thái Tài	<ul style="list-style-type: none">- Quản lý kết nối.- Chạy thử.- Xử lý lỗi.	15%
Nguyễn Phước Triều	<ul style="list-style-type: none">- Kết nối websocket từ client.- Xử lý Server.	15%

	<ul style="list-style-type: none">- Gửi dữ liệu từ Client tới Server.- Chương 2.	
--	---	--

LỜI MỞ ĐẦU

Thế giới càng phát triển công nghệ cũng vì thế mà ngày càng hiện đại và tân tiến hơn bao giờ hết. Bên cạnh những thay đổi làm cho công nghệ phát triển hơn là cả một sự cố gắng và nỗ lực của đội ngũ hoạt động trong lĩnh vực công nghệ nói chung và lĩnh vực IT nói riêng. Việc lập trình không còn đơn thuần chỉ lập trình trên các nền tảng đã có sẵn mà giờ đây còn phát triển thêm lập trình đa nền tảng trên ứng dụng di động. Sự phát triển này giúp cho những người lập trình có thể sáng tạo hơn trong việc tạo ra các sản phẩm cũng như tận dụng được công nghệ hiện đại để phát triển hơn trước kia.

Với sự hỗ trợ của công nghệ hiện đại thì hệ thống sẽ được phát triển một cách nhanh chóng và vận hành theo đúng những quy định được đặt ra. Người sử dụng hệ thống sẽ có cái nhìn tổng thể và khách quan hơn cho những sự lựa chọn của mình. Từ đó sẽ giúp họ tìm được lựa chọn phù hợp với năng lực của bản thân nhất. Đó cũng chính là lý do nhóm chúng em thực hiện đề tài này. Bên cạnh đó khi thực hiện đề tài này nhóm chúng em còn hiểu rõ hơn về cách lấy dữ liệu từ Server bằng cách gọi API sử dụng Networking và HTTP, JSON và SocketIO. Ứng dụng các công nghệ hiện đại để lập trình ra app minh họa đơn giản.

Vì lập trình đa nền tảng trên ứng dụng di động có tính ứng dụng rất cao tuy nhiên lại chưa thực sự phổ biến tại Việt Nam nên nhóm chúng em mong rằng với đề tài này sẽ có thể áp dụng được cho nhiều lĩnh vực khác để mở rộng phạm vi ứng dụng của đề tài.

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến Trường **Đại Học Tài Nguyên và Môi Trường TP HCM** đã đưa môn Công nghệ lập trình đa nền tảng cho ứng dụng di động vào chương trình giảng dạy. Đặc biệt, chúng em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn – THS. Nguyễn Thanh Truyền đã dạy dỗ và tâm huyết truyền đạt những kiến thức quý giá cho chúng em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học của thầy, chúng em đã trau dồi cho bản thân nhiều kiến thức bổ ích, tinh thần học tập nghiêm túc và hiệu quả. Bên cạnh những giờ lý thuyết kết hợp thực hành, đây chắc chắn sẽ là những kiến thức có giá trị sâu sắc, là hành trang để em vững bước sau này. Trên con đường đưa chúng em tới với thời kì công nghiệp hoá hiện đại hoá đất nước ngày càng phát triển hơn nữa.

Bộ môn Công nghệ lập trình đa nền tảng cho ứng dụng di động là môn học thú vị, bổ ích và có tính thực tế cao. Đảm bảo cung cấp đầy đủ kiến thức, kỹ năng, giúp sinh viên có thể ứng dụng vào thực tế. Tuy nhiên, do khả năng tiếp thu thực tế còn nhiều hạn hẹp, kiến thức chưa sâu rộng. Do khả năng tiếp nhận kiến thức mới còn nhiều khó khăn nên mặc dù bản thân đã cố gắng nhưng chắc chắn bài tiểu luận khó tránh khỏi những thiếu sót, kính mong thầy xem xét và góp ý để đề án môn học của nhóm chúng em được hoàn thiện và sẽ là một dự án có triển vọng trong tương lai.

Em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	2
1.1. Tìm hiểu về Flutter.....	2
1.1.1. Flutter là gì?	2
1.1.2. Flutter giúp phát triển ứng dụng	2
1.1.3. Ngôn ngữ lập trình Flutter sử dụng	3
1.1.4. Các loại Widget trong Flutter.....	4
1.2. Tìm hiểu về API, Networking & HTTP và JSON	7
1.2.1. API	7
1.2.2. Sử dụng thư viện Android Networking để lấy dữ liệu.....	9
1.3 Realtime Database	18
1.3.1. Khái niệm	18
1.3.2. Các khả năng chính của Realtime Database.....	19
1.3.3. Về tính mở rộng của Firebase Realtime Database	20
1.3.4. Client - Server	21
1.4. Websocket và thư viện Socket.io	26
1.4.1. Tìm hiểu chung	26
1.4.2. Giao thức WebSocket.....	27
1.4.3. Hoạt động	28
1.4.4. Ưu điểm và nhược điểm của Websocket	29
1.4.5. Thư viện Socket.io	31
1.5. Ưu điểm của socket io.....	32
1.6. Nhược điểm của shocket io	33

1.6.1. Ví dụ đơn giản.....	34
CHƯƠNG 2: PHƯƠNG PHÁP THỰC HIỆN	36
2.1. Add Firebase _ Android Platform.....	36
2.2. Result.....	36
2.3. Download File JSON	37
2.4. Result file JSON	37
2.5. Result file JSON	38
2.6. Paste vào file build.gradle	39
2.7. Add Firebase to webapp.....	39
2.8. Result.....	40
2.9. Copy api key, projectId, appId, messagingSenderId.....	40
2.10. Paste vào file constants.dart.....	41
CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM	42
3.1. Mục Profile Page.....	42
3.2. Màn hình đăng ký	43
3.3. Màn hình đăng nhập	43
3.4. Màn hình group chat	44
3.5. Nhóm chat đã tạo	44
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	45
DANH MỤC TÀI LIỆU THAM KHẢO	46

DANH MỤC HÌNH ẢNH

<i>Hình 1: Giới thiệu Flutter</i>	2
<i>Hình 2: Flutter và Dart</i>	4
<i>Hình 3: Web APIs là gì?</i>	7
<i>Hình 4: Sharding database</i>	20
<i>Hình 5: Client – Server Model</i>	21
<i>Hình 6: Mô hình Client-Server</i>	21
<i>Hình 7: Nguyên tắc hoạt động Client-Server</i>	22
<i>Hình 8: Khả năng truy cập</i>	24
<i>Hình 9: WebSockets là gì?</i>	27
<i>Hình 10: Hoạt động của Client-Server</i>	28
<i>Hình 11: Code minh họa kết nối bị ngắt</i>	30
<i>Hình 12: Tin nhắn từ máy khách</i>	31
<i>Hình 13: Thư viện Socket.io</i>	31
<i>Hình 14: Cài đặt Socket.io</i>	34
<i>Hình 15: Tiếp tục cài đặt</i>	34
<i>Hình 16: Gửi tin nhắn " Hello, Server!"</i>	35
<i>Hình 17: Xử lý sự kiện message</i>	35
<i>Hình 18: Add Firebase</i>	36
<i>Hình 19: Result</i>	37
<i>Hình 20: Download file JSON</i>	37
<i>Hình 21: Download file JSON</i>	38
<i>Hình 22: Result file JSON</i>	38

<i>Hình 23: Paste vào file build.gradle</i>	<i>39</i>
<i>Hình 24: Add Firebase to your web app.....</i>	<i>39</i>
<i>Hình 25: Kết quả</i>	<i>40</i>
<i>Hình 26: Copy API Key.....</i>	<i>41</i>
<i>Hình 27: Paste vào file constants.dart.....</i>	<i>41</i>
<i>Hình 28: Source code trang Profile</i>	<i>42</i>
<i>Hình 29: Kết quả Profile Page.....</i>	<i>42</i>
<i>Hình 30: Màn hình đăng ký</i>	<i>43</i>
<i>Hình 31: Màn hình đăng nhập</i>	<i>43</i>
<i>Hình 32: Màn hình Group chat.....</i>	<i>44</i>
<i>Hình 33: Màn hình nhóm chat vừa tạo</i>	<i>44</i>

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

UI	User Interface
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
XML	Extensible Markup Language
PoP	Points of Presence
SSE	SERVER SENT EVENTS

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1. Tìm hiểu về Flutter

1.1.1. Flutter là gì?

- Flutter là một khung nguồn mở do Google phát triển và hỗ trợ. Các nhà phát triển frontend và fullstack sử dụng Flutter để xây dựng giao diện người dùng (UI) của ứng dụng cho nhiều nền tảng chỉ với một nền mã duy nhất.

- Tại thời điểm ra mắt vào năm 2018, Flutter chủ yếu hỗ trợ phát triển ứng dụng di động. Hiện nay, Flutter hỗ trợ phát triển ứng dụng trên sáu nền tảng: iOS, Android, web, Windows, MacOS và Linux.



Hình 1: Giới thiệu Flutter

1.1.2. Flutter giúp phát triển ứng dụng

- Flutter đơn giản hóa quá trình tạo UI hấp dẫn, nhất quán cho một ứng dụng trên sáu nền tảng mà nó hỗ trợ.

- Vì Flutter là một khung phát triển đa nền tảng, nên trước tiên, chúng ta sẽ so sánh phát triển đa nền tảng với phát triển gốc. Sau đó, chúng ta có thể làm nổi bật các tính năng chỉ có ở Flutter.

- Phát triển ứng dụng gốc so với phát triển ứng dụng đa nền tảng

Viết mã một ứng dụng cho một nền tảng cụ thể, chẳng hạn như iOS, được gọi là phát triển ứng dụng gốc. Ngược lại, phát triển ứng dụng đa nền tảng sẽ xây dựng một ứng dụng cho nhiều nền tảng với một nền mã duy nhất.

- Phát triển ứng dụng gốc

Vì các nhà phát triển viết mã cho một nền tảng cụ thể trong phát triển ứng dụng gốc, họ có toàn quyền truy cập vào chức năng của thiết bị gốc. Điều này thường mang lại hiệu suất và tốc độ cao hơn so với phát triển ứng dụng đa nền tảng.

Tuy nhiên, nếu bạn muốn khởi chạy một ứng dụng trên nhiều nền tảng, phát triển ứng dụng gốc cần nhiều mã hơn và nhiều nhà phát triển hơn. Ngoài những chi phí này, phát triển ứng dụng gốc khiến việc khởi chạy trên các nền tảng khác nhau cùng một lúc với trải nghiệm người dùng nhất quán trở nên khó khăn hơn. Đây là nơi mà các khung phát triển ứng dụng đa nền tảng như Flutter có thể hữu ích.

- Phát triển ứng dụng đa nền tảng

Phát triển ứng dụng đa nền tảng cho phép các nhà phát triển sử dụng một ngôn ngữ lập trình và một nền mã để xây dựng một ứng dụng cho nhiều nền tảng. Nếu bạn chuẩn bị phát hành một ứng dụng cho nhiều nền tảng, phát triển ứng dụng đa nền tảng sẽ ít tốn kém và đỡ mất thời gian hơn so với phát triển ứng dụng gốc.

Quá trình này cũng cho phép các nhà phát triển tạo ra trải nghiệm nhất quán hơn cho người dùng trên các nền tảng.

Cách tiếp cận này có thể có những hạn chế so với phát triển ứng dụng gốc, đó là quyền truy cập hạn chế vào chức năng của thiết bị gốc. Tuy nhiên, Flutter có các tính năng giúp phát triển ứng dụng đa nền tảng mượt mà hơn và đạt hiệu suất cao.

1.1.3. Ngôn ngữ lập trình Flutter sử dụng

- Flutter sử dụng ngôn ngữ lập trình nguồn mở Dart, ngôn ngữ này cũng do Google phát triển. Dart được tối ưu hóa để xây dựng UI và nhiều điểm mạnh của Dart được sử dụng trong Flutter.

Ví dụ: một tính năng của Dart được sử dụng trong Flutter là sound null safety. Tính năng sound null safety của Dart giúp bạn dễ dàng phát hiện ra các lỗi phổ biến được gọi là lỗi null. Tính năng này giúp các nhà phát triển giảm thời gian bảo trì mã và có nhiều thời gian hơn để tập trung vào việc xây dựng các ứng dụng.



Hình 2: Flutter và Dart

1.1.4. Các loại Widget trong Flutter

- Trong Flutter, các nhà phát triển sử dụng các widget để xây dựng bố cục UI. Điều này có nghĩa là mọi thứ mà người dùng nhìn thấy trên màn hình, từ cửa sổ và bảng điều khiển đến các nút và văn bản, đều được tạo ra từ các widget.

- Các widget Flutter được thiết kế để các nhà phát triển có thể dễ dàng tùy chỉnh chúng. Flutter đạt được điều này thông qua cách tiếp cận thành phần. Điều này có nghĩa là hầu hết các widget được tạo thành từ các widget nhỏ hơn và các widget cơ bản nhất đều có những mục đích cụ thể. Điều này cho phép các nhà phát triển kết hợp hoặc chỉnh sửa các widget để tạo ra những widget mới.

- Flutter kết xuất các widget bằng công cụ đồ họa của riêng mình thay vì dựa vào các widget tích hợp sẵn của nền tảng. Theo cách này, người dùng sẽ trải nghiệm giao diện tương tự trong ứng dụng Flutter trên các nền tảng. Cách tiếp cận này cũng mang lại sự linh hoạt cho các nhà phát triển vì một số widget Flutter có thể thực hiện các chức năng mà những widget theo nền tảng không thể thực hiện được.

- Flutter cũng giúp việc sử dụng các widget do cộng đồng phát triển trở nên dễ dàng. Kiến trúc của Flutter hỗ trợ tạo ra nhiều thư viện widget và Flutter khuyến khích cộng đồng xây dựng và duy trì các thư viện widget mới.

- Các loại widget Flutter:

+ Flutter đi kèm với một danh mục widget mở rộng ngay từ khi bạn tải xuống. Danh mục có 14 hạng mục, bao gồm định kiểu, Cupertino (widget kiểu iOS) và Thành phần tư liệu (widget tuân theo hướng dẫn Thiết kế tư liệu của Google).

+ Flutter cũng có các bộ cục và chủ đề đi kèm, giúp các nhà phát triển có thể xây dựng ngay lập tức.

1.1.5. Ưu điểm và nhược điểm của Flutter

• Ưu điểm: Mỗi giải pháp đều có những thế mạnh và yếu điểm riêng, Flutter cũng không ngoại lệ. Nhưng trước tiên phải kể đến một số lợi ích vượt trội khi sử dụng Flutter:

- Flutter chạy nhanh giúp tiết kiệm thời gian, công sức và tiền bạc. Giống như bất kỳ công nghệ đa nền tảng nào khác, Flutter cho phép bạn sử dụng cùng một cơ sở mã để xây dựng các ứng dụng iOS và Android riêng biệt. Điều này sẽ đẩy nhanh toàn bộ quá trình phát triển mà không cần phân tích hai cơ sở mã khác nhau cho cùng một nền tảng.

- "Hot reload" (tải nóng/ nhanh) của Flutter giúp bạn thực hiện các thay đổi đối với mã code và xem được kết quả ngay lập tức trong bản xem trước ứng dụng mà không cần đọc lại mã. Bằng cách này, bạn có thể dễ dàng sửa lỗi và thử nghiệm với các phần tử và tính năng UI khác nhau.

- Tùy chỉnh toàn bộ & kết xuất nhanh nhờ cấu trúc phân lớp của Flutter. Ứng dụng này cung cấp quyền kiểm soát mọi pixel trên màn hình cũng như không giới hạn người dùng thêm và tạo hoạt ảnh trong thiết kế đồ họa, video, văn bản và điều khiển.

- Flutter cũng áp dụng cho web và cung cấp tài liệu thích hợp cho phép bạn kiểm tra cách các điều khiển gốc hoạt động.

- Flutter tách UI khỏi các điều khiển gốc giúp loại bỏ lỗi không tương thích (dù ít xảy ra) từ phía nhà sản xuất. UI riêng biệt cũng tự động đem đến một sự đồng nhất trên tất cả các phiên bản hệ thống.

- Nhược điểm: Mặc dù có nhiều ưu điểm, song qua thử nghiệm, Flutter cũng có những yếu điểm nhất định.

- Flutter vẫn chưa thật hoàn thiện. Vì là một ứng dụng mới nên Flutter vẫn chưa đạt mức hoàn hảo. Thực tế, nhiều tính năng nâng cao của Flutter vẫn chưa được hỗ trợ; nhiều thư viện chưa được thử nghiệm chính thức còn tồn tại hạn chế khi so sánh với các bản sao gốc (như là Google Maps).

- Dart khá "non nớt". Về cơ bản Dart khá giống với Swift và Kotlin, nhưng có ít tính năng hơn, hoặc những tính năng hiện có chưa được toàn diện.

- Các ứng dụng Flutter khá "nặng". Chúng chiếm nhiều dung lượng và mất nhiều thời gian để tải xuống hoặc cập nhật.

- Giao diện không giống 100% so với phiên bản gốc. Về cơ bản, Flutter không tạo ra các thành phần gốc mà sao chép không hoàn toàn các thiết kế Material Design của Android và các thành phần riêng của iOS bằng thư viện Cupertino. Thư viện này sẽ hiển thị, đặc biệt với các phiên bản hệ thống chứa các trường văn bản hoặc các nút - những thành phần biến đổi bên ngoài nhưng không thay đổi bên trong Flutter.

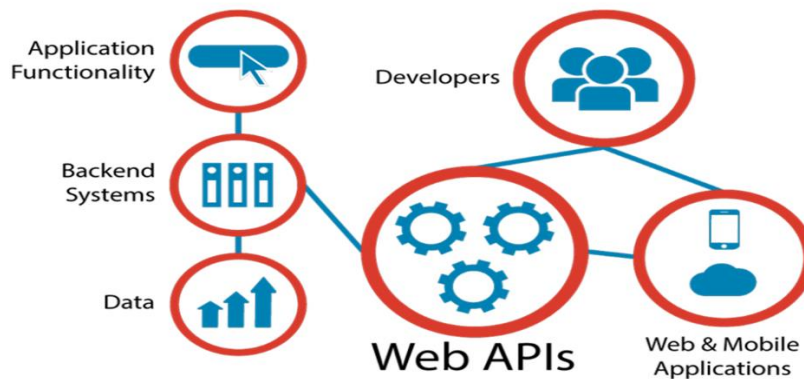
- Hướng dẫn phát triển ứng dụng Flutter chưa được đồng nhất, điều này có thể gây nhiều khó khăn khi xây dựng các phần mềm mang tính phức tạp.

- Framework thay đổi nhanh chóng gây khó khăn cho việc duy trì mã. Thêm vào đó, Flutter chưa chắc sẽ được ứng dụng trong tương lai khi Google liên tục loại bỏ các dự án của ứng dụng này.

1.2. Tìm hiểu về API, Networking & HTTP và JSON

1.2.1. API

API là viết tắt của Application Programming Interface, là một tập các chức năng cho phép các ứng dụng truy cập dữ liệu và tương tác với các thành phần phần mềm bên ngoài, hệ điều hành hoặc một dịch vụ web service.



Hình 3: Web APIs là gì?

- Đặc điểm của API:

+ API sử dụng mã nguồn mở, dùng được với mọi phần mềm hỗ trợ JSON/XML.

+ API có khả năng hỗ trợ đầy đủ các thành phần của HTTP như: URI, header, caching...

+ API là kiểu kiến trúc hỗ trợ tốt các thiết bị hạn chế băng thông như điện thoại, IoT...

- Ứng dụng của API: API có rất nhiều ứng dụng cụ thể, mình có thể tóm gọn 3 ứng dụng phổ biến nhất của API:

+ Web API: Là hệ thống các API được sử dụng cho các webservice như Facebook, Zalo, Twitter... Hầu hết các hệ thống web service đều cung cấp API cho các bên thứ 3 phát triển

ứng dụng dựa trên tài nguyên của họ. Ngày nay, đa số API được thiết kế theo chuẩn RESTFul.

+ API trên hệ điều hành: Các hệ điều hành đều phải xây dựng các API để bên thứ 3 có thể phát triển ứng dụng cho HĐH đó. Nhà phát hành hệ điều hành đều cung cấp tài liệu mô tả chi tiết từng API, đặc tả các hàm, phương thức cũng như các giao thức kết nối. Điều này giúp lập trình viên dễ dàng phát triển phần mềm tương tác với hệ điều hành.

+ API của thư viện: API là cách duy nhất để các thư viện của cung cấp các tính năng cho phần mềm sử dụng nó. API còn giúp cho một chương trình viết bằng ngôn ngữ lập trình này có thể sử dụng thư viện viết bằng ngôn ngữ lập trình khác.

- Ưu điểm:

- Web API được sử dụng hầu hết trên các ứng dụng desktop, ứng dụng mobile và ứng dụng website.

- Linh hoạt với các định dạng dữ liệu khi trả về client: Json, XML hay định dạng khác.

- Nhanh chóng xây dựng HTTP service: URI, request/response headers, caching, versioning, content formats và có thể host trong ứng dụng hoặc trên IIS.

- Mã nguồn mở, hỗ trợ chức năng RESTful đầy đủ, sử dụng bởi bất kì client nào hỗ trợ XML, Json.

- Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test.

- Giao tiếp hai chiều được xác nhận trong các giao dịch, đảm bảo độ tin cậy cao.

- Nhược điểm: Do web API còn khá mới nên chưa thể đánh giá nhiều về nhược điểm của mô hình này. Tuy nhiên, có hai nhược điểm dễ dàng nhận thấy:

- Web API chưa hoàn toàn phải là RESTful service, mới chỉ hỗ trợ mặc định GET, POST.

- Để sử dụng hiệu quả cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt.

Xây dựng ứng dụng Chatapp trên thiết bị di động GVHD: THS. Nguyễn Thanh Truyền

- Tốn thời gian và chi phí cho việc phát triển, nâng cấp và vận hành.
- Có thể gặp vấn đề về bảo mật khi hệ thống bị tấn công nếu không giới hạn điều kiện kỹ.

1.2.2. Sử dụng thư viện Android Networking để lấy dữ liệu

Hiện nay, JSON là một kiểu mô tả dữ liệu được sử dụng phổ biến, nhiều công ty sử dụng JSON để gửi dữ liệu cho các ứng dụng Android, iOS hay WindowPhone. Để có thể đọc được dữ liệu dạng JSON cho ứng dụng Android có rất nhiều cách, bạn có thể sử dụng apache (hiện nay không còn phổ biến nữa) hoặc sử dụng các thư viện như: Volley (google), Retrofit (Square), ... Hôm nay mình xin giới thiệu một thư viện mà mình hay sử dụng vì sự đơn giản, dễ hiểu, dễ sử dụng của nó. Đó là thư viện Android Networking.

Bước 1: Tạo project Android

Bước 2: Thêm quyền INTERNET trong manifest

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

Bước 3: Add thư viện vào project

Đặt vào dependencies của file app/build.gradle:

```
compile 'com.amitshekhar.android:android-networking:0.3.0'
```

Bước 4: Sử dụng thư viện

Mình có chuẩn bị một link dữ liệu dạng JSON sau <http://android-samsung.herokuapp.com/api/category>. Chúng ta sẽ đọc dữ liệu này với thư viện trên.

Mở file MainActivity, thêm vào hàm onCreate():

```
AndroidNetworking.initialize(getApplicationContext());
```

```
AndroidNetworking.get("http://android-samsung.herokuapp.com/api/category")  
    .build()
```

```
.getString(new StringRequestListener() {  
    @Override  
    public void onResponse(String response) {  
        Toast.makeText(getApplicationContext(),response,  
Toast.LENGTH_LONG).show();  
    }  
    @Override  
    public void onError(ANError anError) {  
        Toast.makeText(getApplicationContext(),          anError.toString(),  
Toast.LENGTH_LONG).show();  
    }  
});
```

Bước 5: Chạy thử project và xem kết quả

Ở đây mình sử dụng method là get(), dữ liệu callback là String, khi run ứng dụng thì sẽ Toast ra một dòng kết quả là dữ liệu JSON nếu thành công còn khi thất bại sẽ Toast ra lỗi.

- **Bật truy cập Internet**

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

- **Tạo các tiến trình nền**

Nền tảng Android không cho phép bạn chạy các các tác vụ kết nối mạng trên tiến trình chính của ứng dụng. Vì vậy, tất cả các code kết nối mạng của bạn phải thuộc về một tiến trình nền.

```
AsyncTask.execute(new Runnable() {  
    @Override
```

```
public void run() {  
    // All your networking logic  
    // should be here  
}  
});
```

Nếu bạn muốn tìm hiểu thêm về chạy các tác vụ trong nền, tôi đề nghị bạn đọc hướng dẫn về các tác vụ nền này từ loạt bài Android từ đầu.

- **Tạo một kết nối HTTP**

Bằng cách sử dụng phương thức `openConnection()` của lớp `URL`, bạn có thể nhanh chóng thiết lập một kết nối đến bất kỳ endpoint của REST. Giá trị trả về của `openConnection()` phải được chuyển cho một đối tượng của `URLConnection` hoặc `HttpsURLConnection`, tùy thuộc vào endpoint được truy cập thông qua HTTP hay HTTPS. `URLConnection` và `HttpsURLConnection` đều cho phép bạn thực hiện các thao tác như thêm thông tin header và đọc các phản hồi.

Đoạn code sau đây cho bạn thấy cách thiết lập kết nối với root endpoint của GitHub API:

```
// Create URL  
  
URL githubEndpoint = new URL("https://api.github.com/");  
  
// Create connection  
  
HttpsURLConnection myConnection =  
    (HttpsURLConnection) githubEndpoint.openConnection();
```

- **Thêm Header cho yêu cầu**

Hầu hết các trang web cung cấp REST API muốn có khả năng xác định ứng dụng của bạn là duy nhất. Cách dễ nhất để giúp họ làm như vậy là thêm header User-Agent duy nhất trong tất cả các yêu cầu của bạn.

Để thêm một header User-Agent vào yêu cầu của bạn, bạn phải sử dụng phương thức `setRequestProperty()` của đối tượng `URLConnection`. Ví dụ, ở đây là cách bạn thiết lập header User-Agent vào **my-rest-app-v0.1**:

```
myConnection.setRequestProperty("User-Agent", "my-rest-app-v0.1");
```

Bạn có thể thêm nhiều header vào yêu cầu của bạn bằng cách gọi phương thức `setRequestProperty()` nhiều lần. Ví dụ, đoạn code sau đây thêm một header `Accept` và một header `Contact-Me` tùy biến:

```
myConnection.setRequestProperty("Accept",  
    "application/vnd.github.v3+json");  
myConnection.setRequestProperty("Contact-Me",  
    "hathibelagal@example.com");
```

- **Đọc các phản hồi**

Một khi bạn đã truyền vào tất cả các header cho yêu cầu, bạn có thể kiểm tra xem bạn có một phản hồi hợp lệ hay không bằng cách sử dụng phương thức `getResponseCode()` của đối tượng `URLConnection`.

Nếu lớp `URLConnection` lấy được một mã số phản hồi, chẳng hạn như **301**, nó xử lý mã số một cách tự động và chuyển hướng sau đó. Vì vậy, thông thường, bạn sẽ không cần phải viết thêm bất kỳ code nào để kiểm tra chuyển hướng.

Trong trường hợp không có lỗi, bạn có thể gọi phương thức `getInputStream()` để có được một tham chiếu đến Input Stream của kết nối.

```
InputStream responseBody = myConnection.getInputStream();
```

Hầu hết các REST API ngày nay trả về dữ liệu dưới định dạng JSON hợp lệ. Vì vậy, thay vì đọc trực tiếp từ đối tượng `InputStream`, tôi khuyên bạn tạo ra một `InputStreamReader` cho nó.

```
InputStreamReader responseBodyReader =  
    new InputStreamReader(responseBody, "UTF-8");
```

- **Phân tích JSON trả về**

Android SDK có một lớp gọi là **JsonReader**, nó giúp bạn dễ dàng có thể phân tích tài liệu JSON. Bạn có thể tạo ra một đối tượng mới của lớp `JsonReader` bằng truyền đối tượng `InputStreamReader` vào hàm xây dựng của nó.

```
JsonReader jsonReader = new JsonReader(responseBodyReader);
```

Cách mà bạn trích xuất một phần cụ thể của thông tin từ tài liệu JSON phụ thuộc vào cấu trúc của nó. Ví dụ, tài liệu JSON trả về bởi root endpoint từ REST API của GitHub trông như thế này.

```
{  
    "current_user_url": "https://api.github.com/user",  
    "current_user_authorizations_html_url":  
    "https://github.com/settings/connections/applications{/client_id}",  
    "authorizations_url": "https://api.github.com/authorizations",  
    "code_search_url":  
    "https://api.github.com/search/code?q=1 {&page,per_page,sort,order}",
```

```
"emails_url": "https://api.github.com/user/emails",  
"emojis_url": "https://api.github.com/emojis",  
"events_url": "https://api.github.com/events",  
"feeds_url": "https://api.github.com/feeds",  
"followers_url": "https://api.github.com/user/followers",  
"following_url": "https://api.github.com/user/following{/target}",  
"gists_url": "https://api.github.com/gists{/gist_id}",  
"hub_url": "https://api.github.com/hub",  
"issue_search_url":  
"https://api.github.com/search/issues?q=1 { &page,per_page,sort,order }",  
"issues_url": "https://api.github.com/issues",  
"keys_url": "https://api.github.com/user/keys",  
"notifications_url": "https://api.github.com/notifications",  
"organization_repositories_url":  
"https://api.github.com/orgs/{org}/repos{?type,page,per_page,sort}",  
"organization_url": "https://api.github.com/orgs/{org}",  
"public_gists_url": "https://api.github.com/gists/public",  
"rate_limit_url": "https://api.github.com/rate_limit",  
"repository_url": "https://api.github.com/repos/{owner}/{repo}",  
"repository_search_url":  
"https://api.github.com/search/repositories?q=1 { &page,per_page,sort,order }",  
"current_user_repositories_url":  
"https://api.github.com/user/repos{?type,page,per_page,sort}",
```



```
"starred_url": "https://api.github.com/user/starred{/owner}/{repo}",  
"starred_gists_url": "https://api.github.com/gists/starred",  
"team_url": "https://api.github.com/teams",  
"user_url": "https://api.github.com/users/{user}",  
"user_organizations_url": "https://api.github.com/user/orgs",  
"user_repositories_url":  
"https://api.github.com/users/{user}/repos{?type,page,per_page,sort}",  
"user_search_url":  
"https://api.github.com/search/users?q=1{&page,per_page,sort,order}"  
}
```

Như bạn có thể thấy, các phản hồi chỉ là một đối tượng JSON lớn có chứa một số khoá. Để trích xuất giá trị của khoá được gọi là **organization_url** từ nó, bạn sẽ phải viết code sau đây:

```
jsonReader.beginObject(); // Start processing the JSON object  
  
while (jsonReader.hasNext()) { // Loop through all keys  
  
    String key = jsonReader.nextName(); // Fetch the next key  
  
    if (key.equals("organization_url")) { // Check if desired key
```

Code ở trên xử lý phản hồi JSON như là một stream của các token. Vì vậy, nó chiếm rất ít bộ nhớ. Tuy nhiên, bởi vì nó phải xử lý mỗi lần một token duy nhất, nên nó có thể bị chậm khi xử lý các phản hồi lớn.

Sau khi bạn đã trích xuất tất cả các thông tin cần thiết, bạn phải luôn luôn gọi phương thức `close()` của đối tượng `JsonReader` để nó giải phóng mọi tài nguyên mà nó nắm giữ.

```
jsonReader.close();
```

Bạn cũng phải đóng kết nối bằng cách gọi phương thức `disconnect()` của đối tượng `URLConnection`.

```
myConnection.disconnect();
```

- **Sử dụng các phương thức HTTP khác nhau**

Giao diện REST dựa trên HTTP sử dụng các phương thức HTTP để xác định kiểu hoạt động cần được thực hiện trên một nguồn tài nguyên. Trong các bước trước đó, chúng ta sử dụng phương thức GET của HTTP để thực hiện một thao tác đọc. Bởi vì lớp `URLConnection` sử dụng phương thức GET theo mặc định, nên chúng ta không cần phải chỉ định một cách rõ ràng.

Để thay đổi phương thức HTTP của đối tượng `URLConnection`, bạn phải sử dụng phương thức `setRequestMethod()` của nó. Ví dụ như đoạn code sau đây sẽ mở một kết nối đến một endpoint thuộc về [httpbin.org](https://httpbin.org/post) và thiết lập phương thức HTTP thành POST:

```
URL httpbinEndpoint = new URL("https://httpbin.org/post");
```

```
HttpsURLConnection myConnection
```

```
= (HttpsURLConnection) httpbinEndpoint.openConnection();
```

```
myConnection.setRequestMethod("POST");
```

Như bạn đã biết, yêu cầu POST được sử dụng để gửi dữ liệu đến máy chủ. Bằng cách ghi vào Output Stream của kết nối, bạn có thể dễ dàng thêm bất kỳ dữ liệu nào vào body của yêu cầu POST. Tuy nhiên, trước khi bạn làm vậy, bạn phải chắc chắn rằng bạn gọi phương thức `setDoOutput()` của đối tượng `URLConnection` và truyền true vào nó.

Đoạn code sau đây cho bạn thấy cách để gửi một cặp khóa-giá trị đơn giản lên máy chủ:

```
// Create the data
```

```
String myData = "message=Hello";
```

```
// Enable writing
```

```
myConnection.setDoOutput(true);
```

```
// Write the data
```

```
myConnection.getOutputStream().write(myData.getBytes());
```

- **Lưu các phản hồi vào bộ nhớ đệm**

Luôn là một ý tưởng tốt khi lưu vào bộ nhớ bộ nhớ đệm các phản hồi HTTP. Bằng cách đó, bạn có thể không những giảm mức tiêu thụ băng thông của ứng dụng, mà còn làm cho nó đáp ứng tốt hơn. Từ API Level 13 trở đi, Android SDK cung cấp một lớp gọi là `HttpResponseBodyCache`, cho phép bạn dễ dàng lưu bộ nhớ đệm mà không cần thực hiện bất kỳ thay đổi nào đối với logic của mạng.

Để cài đặt một bộ nhớ đệm cho ứng dụng của bạn, bạn phải gọi phương thức `install()` của lớp `HttpResponseBodyCache`. Phương thức cần một đường dẫn tuyệt đối xác định nơi mà bộ nhớ đệm phải được cài đặt, và một con số chỉ định kích thước bộ nhớ đệm. Bạn có thể sử dụng phương thức `getCacheDir()` nếu bạn không muốn chỉ định đường dẫn tuyệt đối theo cách thủ công.

Đoạn code sau sẽ cài đặt một bộ nhớ đệm có kích thước là 100.000 byte:

```
HttpResponseBodyCache myCache = HttpResponseCache.install(  
    getCacheDir(), 100000L);
```

Khi bộ nhớ đệm đã được cài đặt, thì lớp `HttpURLConnection` sẽ bắt đầu sử dụng nó một cách tự động. Để kiểm tra xem bộ nhớ đệm của bạn có hoạt động hay không, bạn có thể sử dụng phương thức `getHitCount()`, phương thức này trả về số lượng các phản hồi HTTP mà được phục vụ từ bộ nhớ đệm.

```
if (myCache.getHitCount() > 0) {
```

```
    // The cache is working
```

}

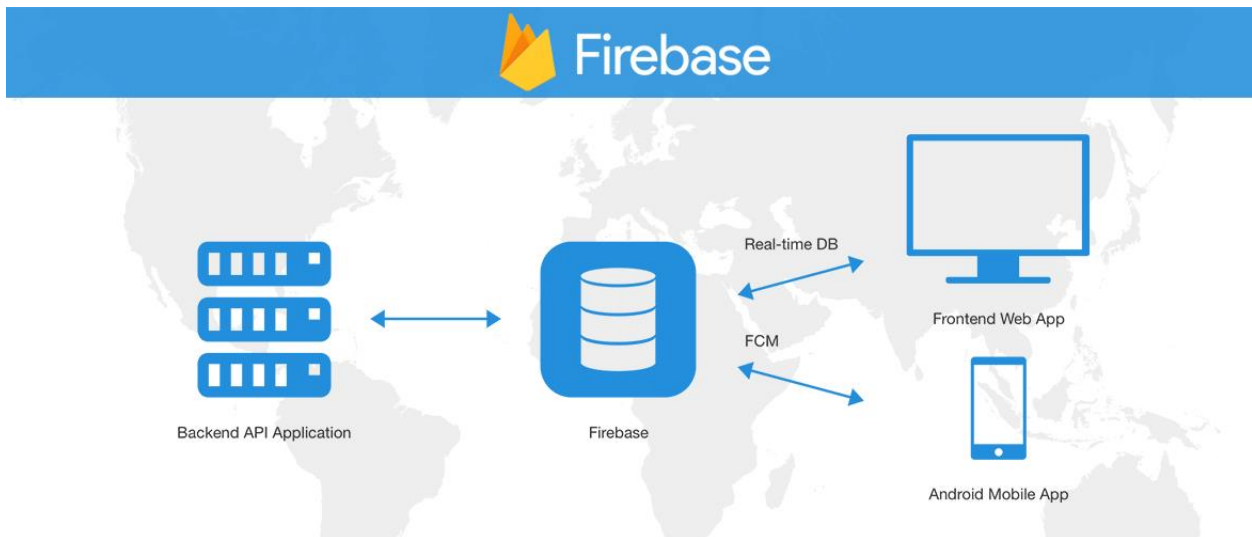
Tổng kết

Có hàng ngàn REST API để bạn có thể tự do sử dụng trong các ứng dụng Android. Bằng cách sử dụng chúng, bạn có thể làm cho ứng dụng của bạn thêm nhiều thông tin phong phú, thú vị và giàu tính năng. Trong hướng dẫn này, bạn đã học được cách sử dụng lớp `HttpURLConnection` để tiêu thụ những REST API như thế. Bạn cũng học được cách để tạo ra một bộ nhớ đệm cho các phản hồi HTTP để giữ cho ứng dụng của bạn sử dụng băng thông thấp.

1.3 Realtime Database

1.3.1. Khái niệm

Realtime Database một service của Firebase. Một cơ sở dữ liệu NoSQL lưu và đồng bộ dữ liệu trên mây. Dữ liệu được đồng bộ trên tất cả clients trong thời gian thực, và vẫn khả dụng khi ứng dụng offline.



Realtime Database và Firebase

Firebase Realtime Database là cơ sở dữ liệu lưu trữ trên mây. Dữ liệu được lưu trữ và đồng bộ hóa theo thời gian thực với mỗi client được kết nối. Khi bạn xây dựng ứng dụng

đa nền tảng với iOS, Android, và javascript SDK, tất cả các client của bạn chia sẻ một thể hiện Realtime Database và tự động tiếp nhận các thay đổi với dữ liệu mới nhất.

Firestore Realtime Database

- Là sản phẩm của Google (chính xác là được Google mua lại)
- Là cơ sở dữ liệu NoSQL được lưu trữ và quản lý trên đám mây.
- Dữ liệu được lưu dưới dạng JSON.
- Client có thể subscribe dữ liệu 1 cách realtime.

Ngoài tính năng của 1 database ra thì nó còn có 1 số tính năng khác nữa như:

- Tính năng xác thực thông qua Facebook, Google, Twitter ...
- Có thể cài đặt điều kiện để phân quyền xem ai được đọc, viết đến nguồn của database.

Ví dụ như ai login mới có quyền read, write chẳng hạn.

• Bởi vì Firestore được dùng ở phía client (web, mobile) nên việc giải mã để lấy ra khóa của firestore là điều hoàn toàn dễ dàng. Do đó nếu không cài đặt điều kiện 1 cách cẩn thận thì có thể sẽ bị bên thứ 3 đọc, viết tài nguyên 1 cách dễ dàng.

1.3.2. Các khả năng chính của Realtime Database

Realtime:

Firestore Realtime Database sử dụng đồng bộ dữ liệu mỗi khi dữ liệu có thay đổi, mọi thiết bị được kết nối sẽ nhận được thay đổi trong vài mili giây.

Offline:

Khi người dùng ngoại tuyến, dữ liệu sẽ được lưu trên bộ nhớ cache của thiết bị và tự động đồng bộ khi bạn trực tuyến. Tất cả là tự động

Accessible from Client Devices

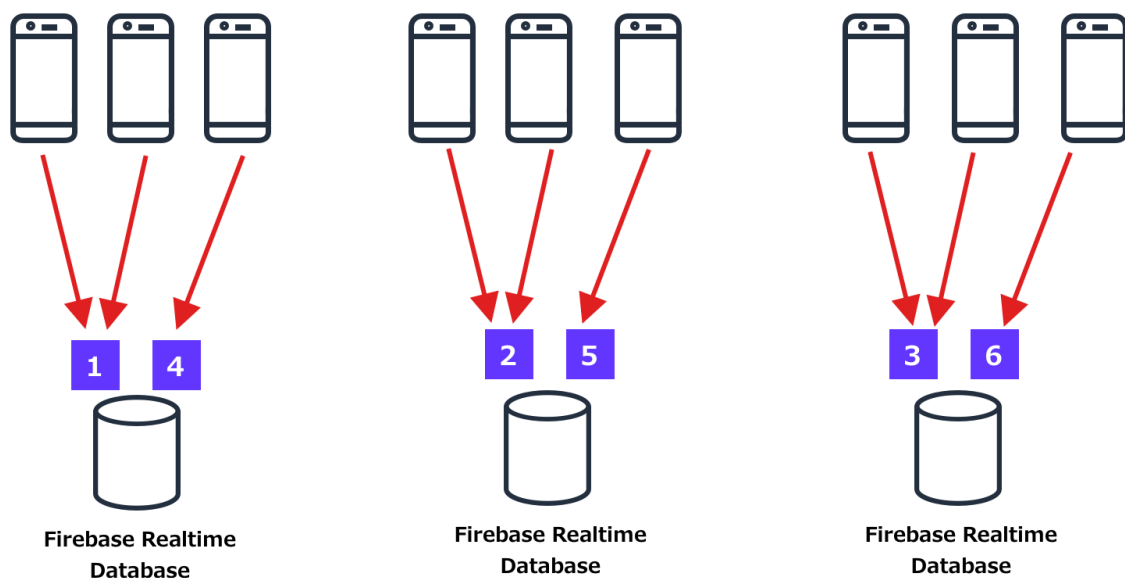
Firebase Realtime Database có thể truy cập từ một thiết bị mobile hoặc trình duyệt web. Nó không cần một ứng dụng server nào cả. Bảo mật và xác thực dữ liệu có thể thông qua các Rule bảo mật của Firebase Realtime Database, các rule được thực thi khi dữ liệu được đọc hoặc ghi.

1.3.3. Về tính mở rộng của Firebase Realtime Database

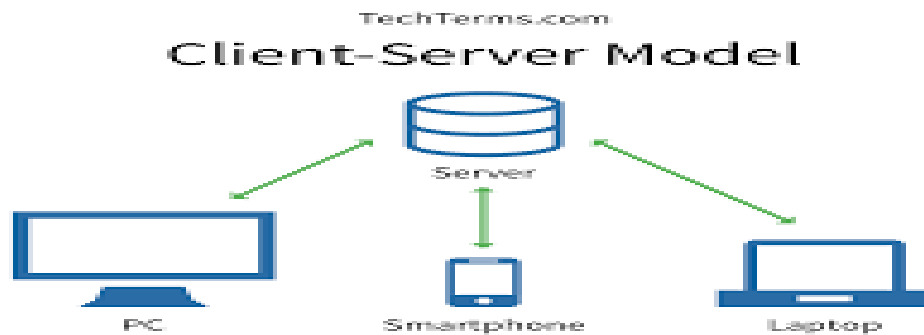
Điều cần chú ý là Firebase Realtime Database không thể scale up với scale out.

Do đó trước khi đưa nó vào hệ thống thì chúng ta cần test tải kỹ càng trước xem cần bao nhiêu con database thì hợp lý, dựa vào tùy tình hình mà sẽ tiến hành sharding để đáp ứng được số lượng lời yêu cầu đề ra.

Firebase realtime database hiện tại chỉ cho phép 1 triệu request đồng thời và 10k write/s. Do đó để không vượt quá ngưỡng này thì sharding là điều hoàn toàn cần thiết.



Hình 4: Sharding database

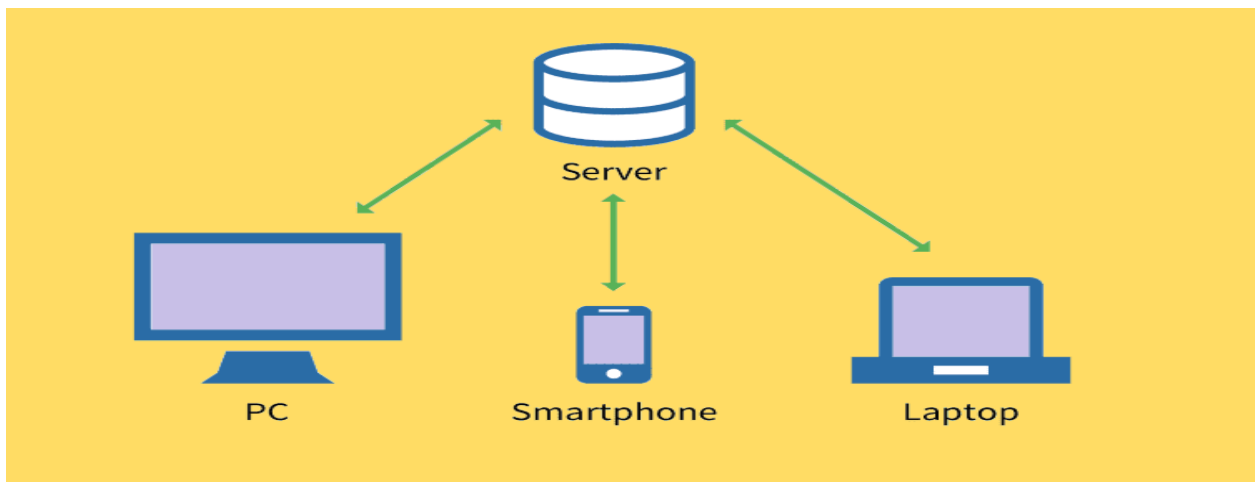


Hình 5: Client – Server Model

Chúng ta sẽ sharding database theo liveID. Như trong ví dụ trên thì với live_id=1 và 4 sẽ cho trong db1, live_id = 2 và 5 sẽ cho trong db2, và live_id=3 và 6 sẽ cho trong db3. Nếu chia như thế thì chúng ta có thể xử lý được 3 triệu request đồng thời và 30k write/s.

1.3.4. Client - Server

- Khái niệm: Mô hình client server mô hình mạng máy tính gồm có 2 thành phần chính là client và server (tức là máy khách – máy chủ). Client sẽ là bên yêu cầu dịch vụ cài đặt cũng như lưu trữ tài nguyên từ phía server.



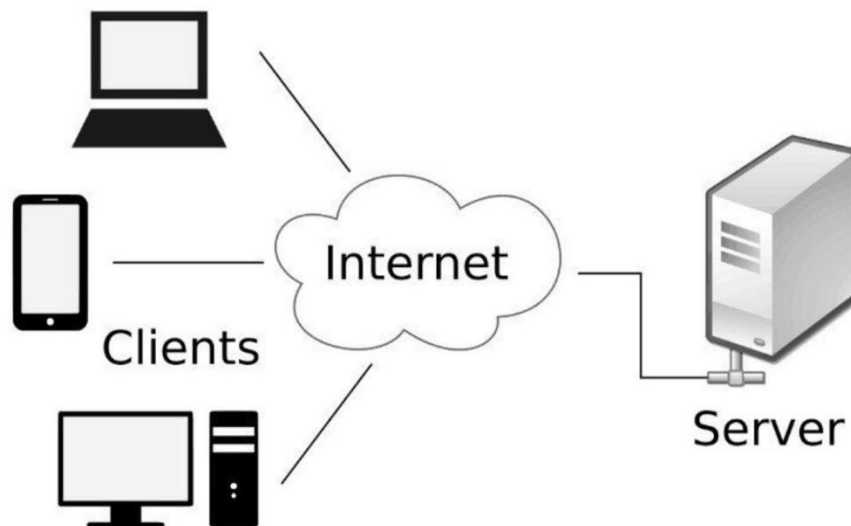
Hình 6: Mô hình Client-Server

Khi máy client gửi yêu cầu dữ liệu đến server thông qua Internet, server sẽ chấp nhận quy trình được yêu cầu. Sau đó gửi các gói dữ liệu được yêu cầu trở lại client. Client không

chia sẻ bất kỳ tài nguyên nào của họ. Đây là một cấu trúc ứng dụng phân tán, nó phân vùng các nhiệm vụ hay workload giữa các nhà cung cấp tài nguyên hoặc dịch vụ, gọi là server, và người yêu cầu dịch vụ (Client).

- Nguyên tắc hoạt động của Client – Server:

- Khi nói đến Client (khách hàng), thì nó có nghĩa là một người hay một tổ chức sử dụng một dịch vụ cụ thể nào đó. Trong thế giới kỹ thuật số cũng tương tự như vậy. Client là một máy tính (Host), tức là có khả năng nhận thông tin hoặc sử dụng một dịch vụ cụ thể từ các nhà cung cấp dịch vụ (Server).



Hình 7: Nguyên tắc hoạt động Client-Server

- Server

Tương tự như vậy, khi nói đến Server thì nó có nghĩa là một máy chủ hay một phương tiện phục vụ các dịch vụ nào đó. Trong lĩnh vực công nghệ thì Server là một máy tính từ xa. Nó cung cấp các thông tin (dữ liệu) hoặc quyền truy cập vào các dịch vụ cụ thể.

Vì vậy, về cơ bản thì trong mô hình Client và Server, Client là đối tượng yêu cầu một thứ gì đó. Server thì phục vụ nó, miễn là nó có mặt trong cơ sở dữ liệu.

Xây dựng ứng dụng Chatapp trên thiết bị di động GVHD: THS. Nguyễn Thanh Truyền

Ngoài ra, nếu bạn đang có nhu cầu sử dụng Server có thể tham khảo Vietnix – một trong những đơn vị cung cấp Server chất lượng cao, uy tín được đông đảo người dùng lựa chọn.

Dịch vụ máy chủ của Vietnix luôn đặt ra những tiêu chuẩn khắt khe về chất lượng phục vụ, Vietnix không ngừng nâng cấp hệ thống cơ sở vật chất, hạ tầng hiện đại:

Đảm bảo tốc độ truy cập nhanh nhất cho website: cũng như mang lại sự ổn định vượt trội cho máy chủ, Vietnix đã lựa chọn trung tâm dữ liệu VNPT IDC để triển khai hệ thống. Đây là trung tâm dữ liệu hiện đại số 1 tại Việt Nam hiện nay với băng thông lớn nhất cả nước.

Hệ thống ổn định: Để tiến độ công việc của khách hàng không bị ảnh hưởng Vietnix đã đầu tư hệ thống server chuyên nghiệp cùng đội ngũ chuyên viên bảo trì và điều hành. Ngoài ra, Vietnix cũng đầu tư hệ thống Router có năng lực phục vụ băng thông lên đến 400Gbps, mang lại tốc độ truy cập nhanh nhất cho tất cả các dịch vụ.

Phản hồi nhanh, hỗ trợ mọi thời điểm: Các khó khăn của khách hàng trong quá trình sử dụng tránh làm gián đoạn đến công việc, hoạt động kinh doanh sẽ được đội ngũ kỹ thuật chuyên môn cao, luôn sẵn sàng phản hồi và hỗ trợ nhanh chóng thông qua đa kênh như: Livechat, Zalo, Facebook, Telegram, Hotline, Ticket.

1.3.4. Ưu điểm và nhược điểm của mô hình mạng Client – Server

- Ưu điểm:

- Tập trung:

Ưu điểm chính của mô hình mạng khách chủ là khả năng kiểm soát tập trung (Centralization) được tích hợp sẵn. Với mô hình này, tất cả thông tin cần thiết đều được đặt ở một vị trí duy nhất. Việc này rất hữu ích cho những quản trị viên mạng. Vì họ có được toàn quyền quản lý và điều hành.

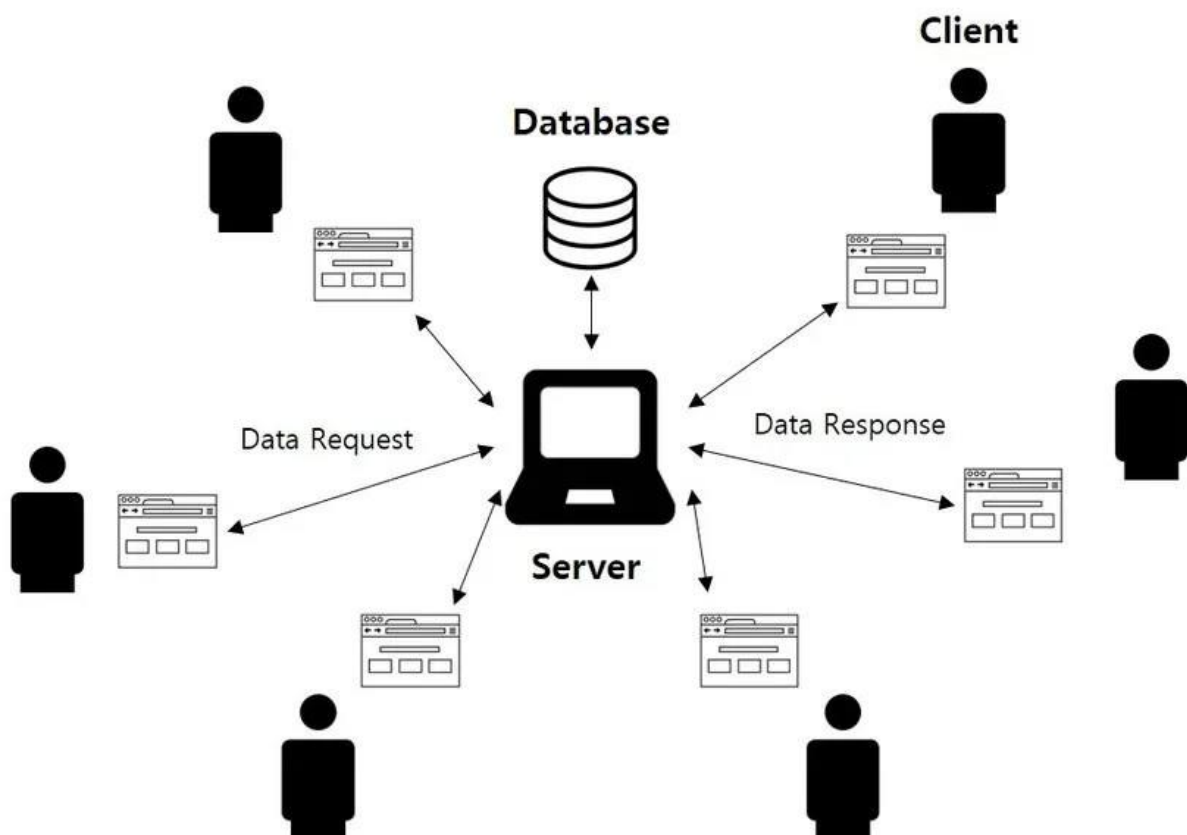
Bằng tính năng này, mọi sự cố trong mạng đều có thể được giải quyết ở một nơi duy nhất. Và do đó, việc cập nhật tài nguyên, dữ liệu cũng trở nên dễ dàng hơn.

- Bảo mật:

Trong mạng Client Server, dữ liệu được bảo vệ tốt do kiến trúc tập trung của mạng. Nó có thể được áp dụng các biện pháp kiểm soát truy cập, sao cho chỉ những người dùng được ủy quyền mới có thể truy cập.

Một trong những cách để làm vậy là áp đặt thông tin đăng nhập, chẳng hạn như Username hay Password. Hơn nữa, nếu dữ liệu bị mất, các file có thể được khôi phục dễ dàng từ chỉ một bản sao lưu duy nhất.

- Khả năng mở rộng: Mô hình mạng Client server có khả năng mở rộng tốt. Bất cứ khi nào người dùng cần, họ có thể tăng số lượng tài nguyên, chẳng hạn như số Client hay Server. Do đó có thể dễ dàng tăng kích thước của Server mà không bị gián đoạn nhiều.



Hình 8: Khả năng truy cập

- Khả năng truy cập:

Không có sự phân biệt giữa các vị trí hay nền tảng khác nhau, mọi client đều có thể đăng nhập vào hệ thống. Bằng cách này, tất cả nhân viên đều có thể truy cập thông tin của công ty của họ, không cần phải dùng một terminal mode hay một bộ xử lý nào.

- Nhược điểm:

- Tắc nghẽn lưu lượng:

Nhược điểm chính của mô hình mạng Client Server là tắc nghẽn lưu lượng. Nếu có quá nhiều Client tạo request từ cùng một Server, nó có thể làm chậm kết nối. Hoặc tệ hơn là dẫn đến hiện tượng crash. Một server bị quá tải có thể tạo ra nhiều vấn đề trong việc truy cập thông tin.

- Độ bền:

Như ta đã biết, mạng Client Server là mạng tập trung. Nên nếu Server chính xảy ra sự cố hay bị nhiễu, toàn bộ hệ thống mạng sẽ bị gián đoạn. Do đó, các mạng client server sẽ thiếu tính ổn định và độ bền.

- Chi phí:

Chi phí thiết lập và bảo trì server trong các mạng client server thường khá cao. Vì các hệ thống mạng có sức mạnh lớn có thể có giá rất đắt. Do đó, không phải tất cả người dùng đều có thể chi trả được.

- Bảo trì:

Khi các Server được triển khai, nó sẽ hoạt động không ngừng nghỉ. Có nghĩa là nó cần được quan tâm đúng mức nếu có bất kỳ vấn đề gì thì phải giải quyết ngay. Vì vậy, cần có một nhà quản lý mạng chuyên biệt để duy trì hoạt động của Server.

- Tài nguyên:

Không phải tất cả tài nguyên hiện có ở trên Server đều có thể sử dụng được. Ví dụ như bạn không thể in trực tiếp tài liệu trên web, hoặc chỉnh sửa bất kỳ thông tin nào trên ổ cứng của Client.

1.4. Websocket và thư viện Socket.io

1.4.1. Tìm hiểu chung

- Một server giao thức WebSocket là một ứng dụng máy chủ được thiết kế để lắng nghe các kết nối WebSocket đến từ các máy khách. Nhiệm vụ chính của server là xử lý các yêu cầu kết nối mới, thiết lập các kết nối WebSocket và truyền tải dữ liệu giữa các máy khách và máy chủ. Dưới đây là một số cách để xây dựng một server giao thức WebSocket:

Sử dụng thư viện WebSocket: Có nhiều thư viện WebSocket phổ biến có thể sử dụng để xây dựng một server giao thức WebSocket, bao gồm Socket.IO, ws, và uWebSockets. Các thư viện này cung cấp API để tạo và quản lý các kết nối WebSocket, xử lý các yêu cầu mới và truyền tải dữ liệu giữa các máy khách và máy chủ.

Sử dụng Node.js: Nếu bạn đang sử dụng Node.js, bạn có thể sử dụng giao thức WebSocket tích hợp sẵn của Node.js để xây dựng server giao thức WebSocket. Giao thức này cung cấp một API cho phép lắng nghe các kết nối đến từ các máy khách và xử lý các yêu cầu mới.

1. Sử dụng các dịch vụ cloud: Các dịch vụ đám mây như Amazon Web Services (AWS) và Microsoft Azure cung cấp các dịch vụ WebSocket tích hợp sẵn, cho phép bạn tạo và quản lý các kết nối WebSocket một cách dễ dàng và hiệu quả.

Hiện nay ứng dụng web đã phát triển khác xa so với ngày đầu nó xuất hiện, kèm theo đó là vô số các kỹ thuật mới được áp dụng để phục vụ cho quá trình này nhằm đem lại trải nghiệm mới mẻ, đầy hứng thú và cũng không kém phần tiện dụng cho người dùng. Công nghệ web thời gian thực(realtime) ngày càng trở nên phổ biến. Có nhiều công nghệ, phương pháp giúp xây dựng ứng dụng thời gian thực

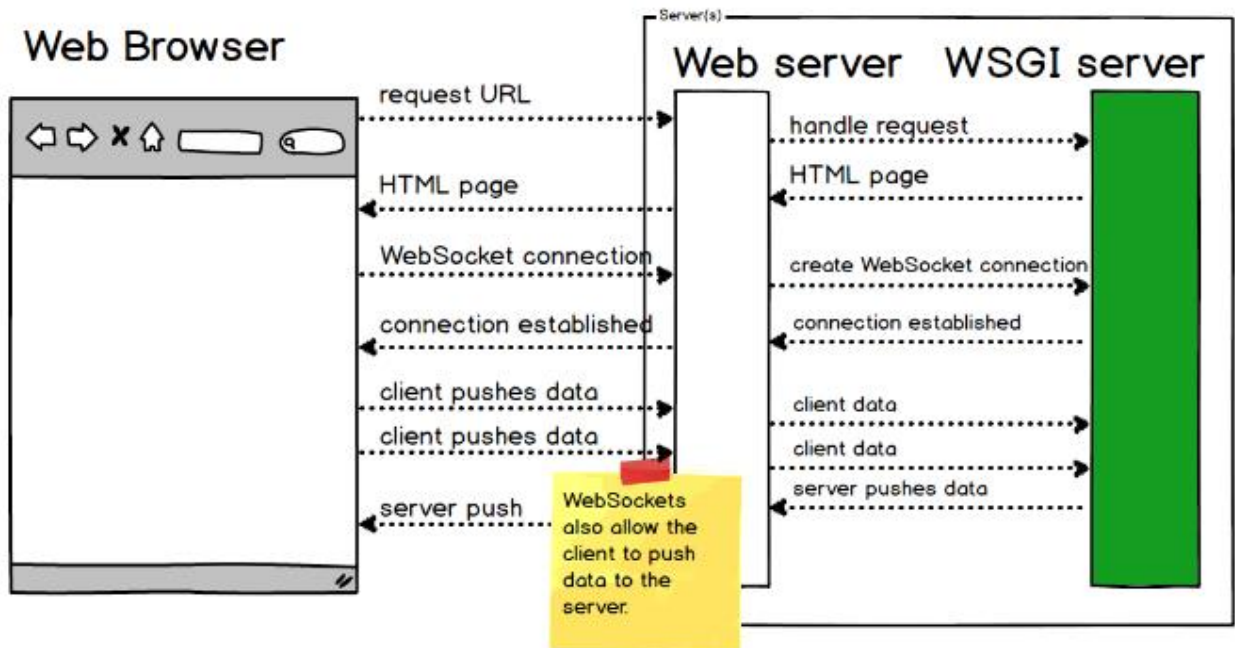
- AJAX LONG-POLLING:
- SERVER SENT EVENTS (SSE)
- COMET

- WEBSOCKET

Trong đó WEBSOCKET với sự hỗ trợ của HTML 5 đang trở lên chiếm ưu thế tuyệt đối.

1.4.2. Giao thức WebSocket

WebSockets



Hình 9: WebSockets là gì?

WebSocket là một giao thức giúp truyền dữ liệu hai chiều giữa server-client qua một kết nối TCP duy nhất. Hơn nữa, websocket là một giao thức được thiết kế để truyền dữ liệu bằng cách sử dụng cổng 80 và cổng 443 và nó là một phần của HTML5. Vì vậy, webSockets có thể hoạt động trên các cổng web tiêu chuẩn, nên không có rắc rối về việc mở cổng cho các ứng dụng, lo lắng về việc bị chặn bởi các tường lửa hay proxy server

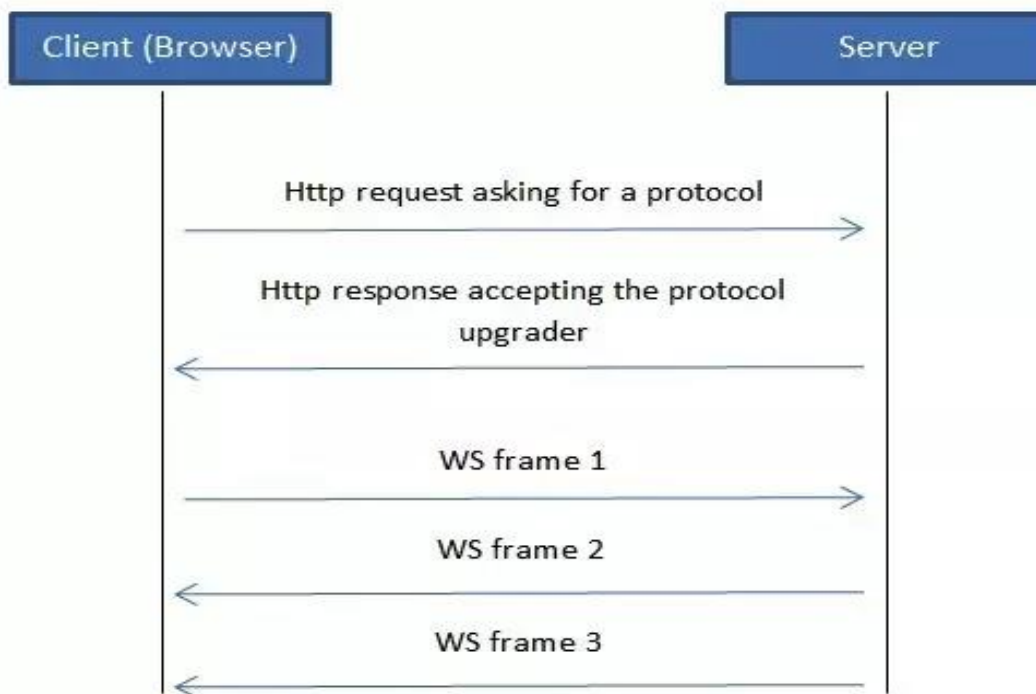
Không giống với giao thức HTTP là cần client chủ động gửi yêu cầu cho server, client sẽ chờ đợi để nhận được dữ liệu từ máy chủ. Hay nói cách khác với giao thức Websocket thì server có thể chủ động gửi thông tin đến client mà không cần phải có yêu cầu từ client.

Tất cả dữ liệu giao tiếp giữa client-server sẽ được gửi trực tiếp qua một kết nối cố định làm cho thông tin được gửi đi nhanh chóng và liên tục khi cần thiết. WebSocket làm giảm độ trễ bởi vì một khi kết nối WebSocket được thành lập, server không cần phải chờ đợi cho một yêu cầu từ client.

Tương tự như vậy, client có thể gửi tin nhắn đến server bất cứ lúc nào. Yêu cầu duy nhất này giúp làm giảm đáng kể độ trễ, mà sẽ gửi một yêu cầu trong khoảng thời gian, cho dù thông điệp có sẵn.

Để có thể sử dụng được Websocket thì không phải chỉ cần trình duyệt hỗ trợ mà còn phải có server Websocket, server Websocket có thể được tạo ra bằng bất kỳ ngôn ngữ server-side nào, nhưng Node.js được sử dụng rộng rãi hơn cả vì nó viết bằng Javascript nên mang nhiều ưu điểm so với các ngôn ngữ server-side truyền thống khác.

1.4.3. Hoạt động



Hình 10: Hoạt động của Client-Server

Giao thức có hai phần: Bắt tay và truyền dữ liệu Ban đầu client sẽ gửi yêu cầu khởi tạo kết nối websocket đến server, server kiểm tra và gửi trả kết quả chấp nhận kết nối, sau đó kết nối được tạo và quá trình gửi dữ liệu có thể được thực hiện, dữ liệu chính là các Ws frame.

1.4.4. Ưu điểm và nhược điểm của WebSocket

- Về ưu điểm:

WebSockets cung cấp giao tiếp hai chiều mạnh mẽ với độ trễ thấp và tỷ lệ lỗi thấp. Nó không yêu cầu nhiều kết nối như phương pháp bỏ phiếu dài Comet và không có nhược điểm của phát trực tuyến Comet.

API cũng là một thư viện tuyệt vời để xử lý các kết nối lại, hết thời gian chờ, yêu cầu ajax (yêu cầu Ajax), xác nhận và các phương tiện truyền tải tùy chọn khác nhau (Ajax long polling và JSONP polling) trực tiếp mà không cần các lớp bổ sung. Điều này rất đơn giản so với Comet thường yêu cầu.

- Về nhược điểm:

Đây là một thông số kỹ thuật mới cho HTML5, vì vậy không phải tất cả các trình duyệt đều hỗ trợ nó. Không có khu vực bắt buộc. Sockets là cổng TCP, không phải là yêu cầu HTTP, vì vậy việc sử dụng các dịch vụ nhận biết yêu cầu như SessionInViewFilter của Hibernate không phải là chuyện nhỏ.

Hibernate là một khung công tác cổ điển cung cấp các bộ lọc xung quanh các yêu cầu HTTP. Bắt đầu một yêu cầu đặt ra các xung đột (bao gồm các giao dịch và ràng buộc JDBC) được liên kết với luồng yêu cầu. Khi yêu cầu này kết thúc, bộ lọc sẽ ngừng cạnh tranh.

- Ví dụ: ví dụ đơn giản về cách sử dụng Socket.io để tạo một server giao thức WebSocket:

Chúng ta đang sử dụng thư viện Express để tạo một ứng dụng web, sau đó tạo một đối tượng server HTTP và sử dụng thư viện Socket.io để tạo một server giao thức WebSocket.

Khi một kết nối WebSocket được thiết lập, chúng ta đang lắng nghe sự kiện connection để xử lý các tin nhắn được gửi đến từ máy khách.

Khi một tin nhắn được gửi đến từ máy khách thông qua sự kiện chat message, chúng ta đang in nội dung của tin nhắn ra console và gửi lại tin nhắn đó đến tất cả các máy khách đang kết nối thông qua sự kiện chat message. Cuối cùng, chúng ta đang lắng nghe sự kiện disconnect để xử lý các kết nối bị ngắt.

```
javascript

const express = require('express');
const app = express();
const http = require('http');
const server = http.createServer(app);
const io = require('socket.io')(server);

// Khi một kết nối WebSocket được thiết lập
io.on('connection', (socket) => {
  console.log('A user connected');

  // Lắng nghe sự kiện "chat message" từ máy khách
  socket.on('chat message', (msg) => {
    console.log(`Message: ${msg}`);

    // Gửi lại tin nhắn đến tất cả các máy khách đang kết nối
    io.emit('chat message', msg);
  });

  // Khi một kết nối WebSocket bị ngắt
  socket.on('disconnect', () => {
    console.log('A user disconnected');
  });
});

// Khởi động server và lắng nghe các kết nối tới cổng 3000
server.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

Hình 11: Code minh họa kết nối bị ngắt

Xây dựng ứng dụng Chatapp trên thiết bị di động GVHD: THS. Nguyễn Thanh Truyền

Bạn có thể tạo một máy khách để kết nối tới server này bằng cách sử dụng thư viện Socket.io trên máy khách. Bạn có thể sử dụng đoạn mã sau để tạo một kết nối WebSocket tới server và gửi một tin nhắn đến server.

```
javascript

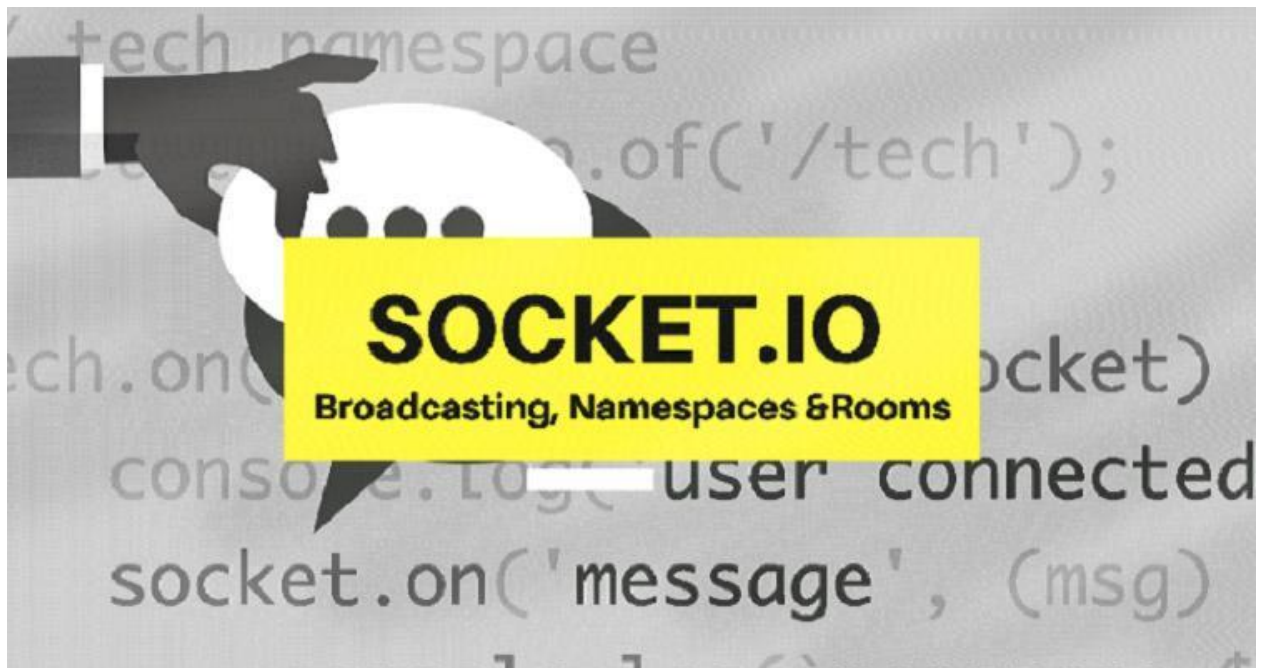
const socket = io('http://localhost:3000');

// Gửi tin nhắn từ máy khách tới server
socket.emit('chat message', 'Hello, server!');
```

Hình 12: Tin nhắn từ máy khách

Khi tin nhắn được gửi từ máy khách, server sẽ in nội dung của tin nhắn ra console và gửi lại tin nhắn đó đến tất cả các máy khách đang kết nối.

1.4.5. Thư viện Socket.io



Hình 13: Thư viện Socket.io

Socket.IO là một bộ thư viện dành cho các ứng dụng web, mobile để phát triển các ứng dụng realtime. Với đặc trưng mạnh mẽ và dễ sử dụng, Socket.IO đang ngày càng được sử dụng rộng rãi từ những trang mạng xã hội cần sự tương tác cao, đến các blog hay các trang web thương mại điện tử. Với bộ thư viện này, làm việc với WebSockets trở nên đơn giản hơn rất nhiều. Thư viện gồm 2 phần

- Phía client: gồm bộ thư viện viết cho web(JavaScript), iOS, Android
- Phía server: viết bằng JavaScript và dùng cho các máy chủ node.JS

Socket.IO hỗ trợ sử dụng rất nhiều các công nghệ realtime

- WebSocket
- Flash Socket
- AJAX long-polling
- AJAX multipart streaming
- IFrame
- JSONP polling

Nó sẽ tự động chuyển sang Websocket nếu có thể, hầu hết các trình duyệt hiện nay đã hỗ trợ websocket nên việc sử dụng socket.io trên trình duyệt cũng là đang sử dụng websocket Việc sử dụng socket.io rất đơn giản và giống nhau ở cả client lẫn server nó bao gồm 3 phần chính:

- Khởi tạo kết nối
- Lắng nghe event
- Gửi event

1.5. Ưu điểm của socket io

• Đa nền tảng: Socket.io hỗ trợ nhiều nền tảng, bao gồm trình duyệt web, Node.js và các ứng dụng di động, cho phép bạn truyền tải dữ liệu giữa các nền tảng một cách dễ dàng.

- Tự động đàm phán: Socket.io sử dụng một cơ chế đàm phán tự động để chọn phương thức truyền tải tốt nhất cho từng máy khách, bao gồm WebSocket, AJAX long polling và JSONP polling. Điều này giúp đảm bảo rằng truyền tải dữ liệu luôn được thực hiện một cách hiệu quả và ổn định.
- Xử lý lỗi: Socket.io có khả năng xử lý các tình huống lỗi và đảm bảo rằng các kết nối WebSocket không bị gián đoạn khi có sự cố xảy ra.
- Dễ sử dụng: Socket.io cung cấp một API dễ sử dụng cho phép bạn tạo và quản lý các kết nối WebSocket, xử lý các yêu cầu và truyền tải dữ liệu giữa các máy khách và máy chủ.
- Cộng đồng phát triển mạnh mẽ: Socket.io là một dự án mã nguồn mở được sử dụng rộng rãi và có một cộng đồng phát triển mạnh mẽ, đảm bảo rằng thư viện này luôn được cập nhật và bảo trì tốt nhất.

1.6. Nhược điểm của socket io

- Khó cấu hình: Socket.io có nhiều tùy chọn cấu hình và có thể khó để cấu hình chính xác cho nhu cầu của bạn. Điều này có thể dẫn đến hiệu suất kém hoặc các vấn đề khác liên quan đến kết nối và truyền tải dữ liệu.
- Không tương thích ngược: Các phiên bản Socket.io khác nhau không tương thích ngược với nhau, điều này có thể làm cho việc nâng cấp hoặc giảm cấp phiên bản trở nên phức tạp hơn.
- Có thể ảnh hưởng đến hiệu suất: Socket.io có thể ảnh hưởng đến hiệu suất của ứng dụng nếu không được cấu hình chính xác. Việc sử dụng Socket.io có thể tạo ra một số tác động tiêu cực đến hiệu suất của ứng dụng, bao gồm tốc độ mạng chậm hoặc độ trễ kết nối.
- Không hỗ trợ truyền tải dữ liệu lớn: Socket.io không được tối ưu để truyền tải dữ liệu lớn, điều này có thể gây ra các vấn đề về hiệu suất và ổn định kết nối.

- Thư viện phức tạp: Socket.io là một thư viện phức tạp và có nhiều tính năng khác nhau. Điều này có thể làm cho việc học và sử dụng Socket.io trở nên phức tạp hơn so với các thư viện khác.

1.6.1. Ví dụ đơn giản

Dưới đây là một ví dụ đơn giản về cách sử dụng Socket.io để truyền tải dữ liệu giữa máy khách và máy chủ:

Đầu tiên, bạn cần cài đặt Socket.io bằng cách thêm đoạn mã sau vào tệp HTML của bạn:

```
html

<script src="/socket.io/socket.io.js"></script>
```

Hình 14: Cài đặt Socket.io

Sau đó, bạn có thể tạo một kết nối Socket.io trên máy khách bằng cách sử dụng đoạn mã sau:

```
javascript

const socket = io();
```

Hình 15: Tiếp tục cài đặt

chúng ta đang tạo một kết nối Socket.io tới địa chỉ mặc định của máy chủ, tương đương với `http://localhost:80`.

Bây giờ, bạn có thể xử lý các sự kiện Socket.io trên máy khách bằng cách sử dụng các phương thức `emit()` và `on()`. Ví dụ, để gửi một tin nhắn từ máy khách đến máy chủ, bạn có thể sử dụng đoạn mã sau:

javascript

```
socket.emit('message', 'Hello, server!');
```

Hình 16: Gửi tin nhắn "Hello, Server!"

Ở đây, chúng ta đang gửi một sự kiện có tên là message tới máy chủ, với nội dung là chuỗi 'Hello, server!'.

Trên máy chủ, bạn có thể xử lý sự kiện message bằng cách sử dụng đoạn mã sau:

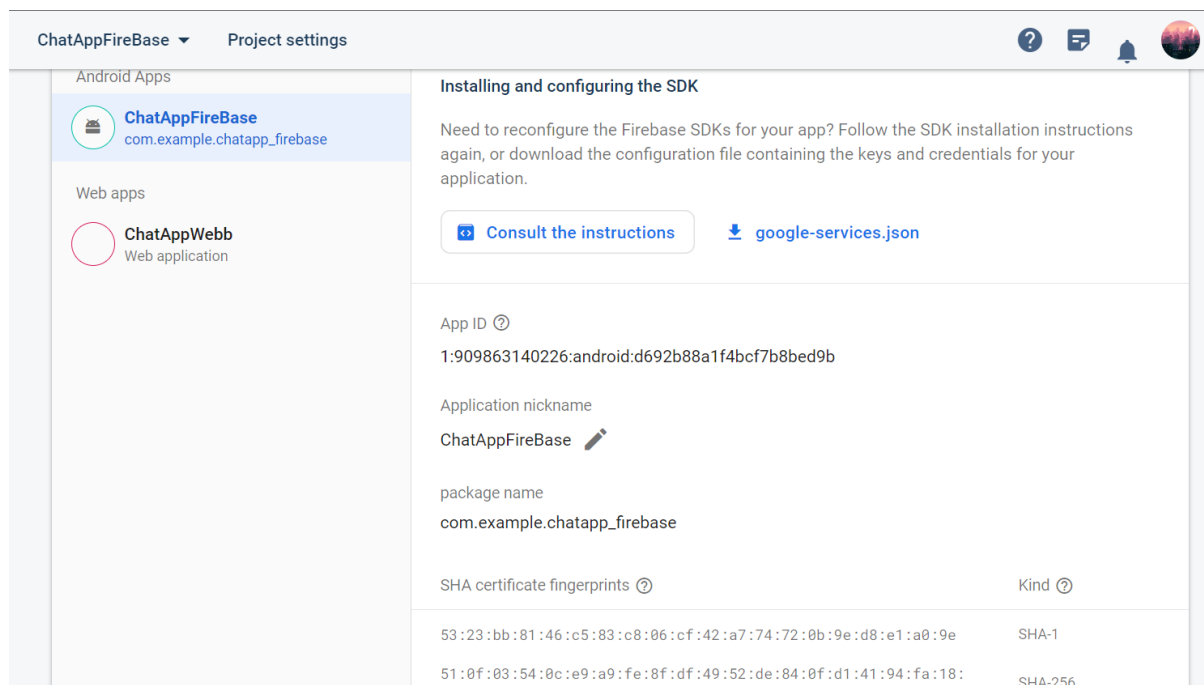
javascript

```
io.on('connection', (socket) => {  
  socket.on('message', (msg) => {  
    console.log(`Received message: ${msg}`);  
  });  
});
```

Hình 17: Xử lý sự kiện message

Ở đây, chúng ta đang lắng nghe các kết nối mới tới máy chủ thông qua sự kiện connection. Khi một kết nối được thiết lập, chúng ta đang lắng nghe sự kiện message để xử lý các tin nhắn được gửi từ máy khách. Khi một tin nhắn được gửi, chúng ta đang in nội dung của nó ra console với đoạn mã console.log().

Tóm lại, đó là một ví dụ đơn giản về cách sử dụng Socket.io để truyền tải dữ liệu giữa máy khách và máy chủ. Có rất nhiều cách để sử dụng Socket.io và thư viện này cung cấp nhiều tính năng khác nhau để giúp bạn xây dựng các ứng dụng theo thời gian thực và tương tác trực tiếp giữa máy khách và máy chủ.



Hình 19: Result

2.3. Download File JSON

Installing and configuring the SDK

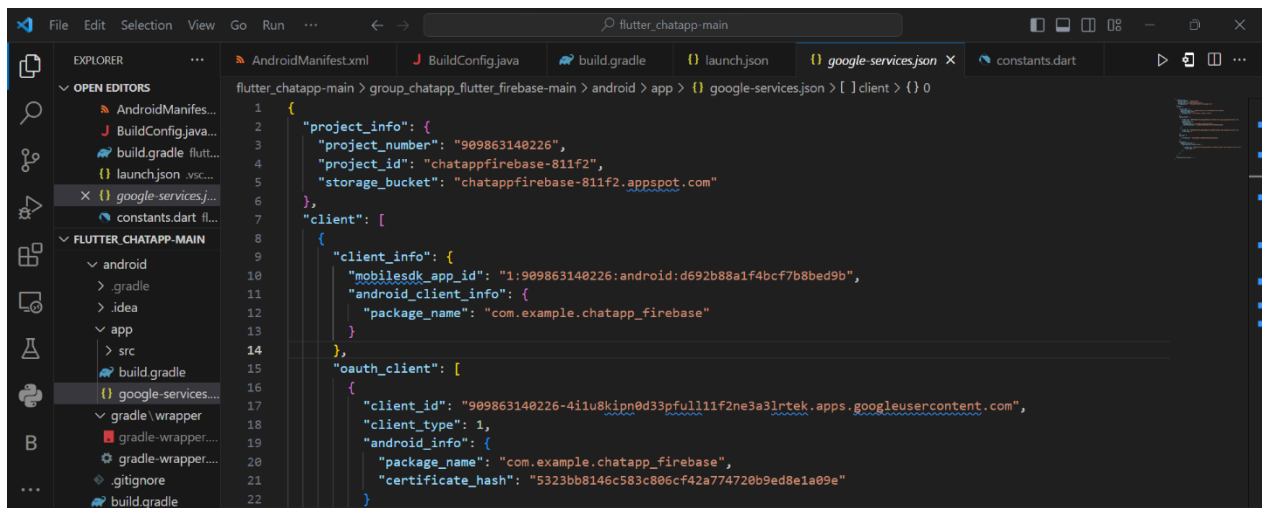
Need to reconfigure the Firebase SDKs for your app? Follow the SDK installation instructions again, or download the configuration file containing the keys and credentials for your application.

[Consult the instructions](#)

[google-services.json](#)

Hình 20: Download file JSON

2.4. Result file JSON



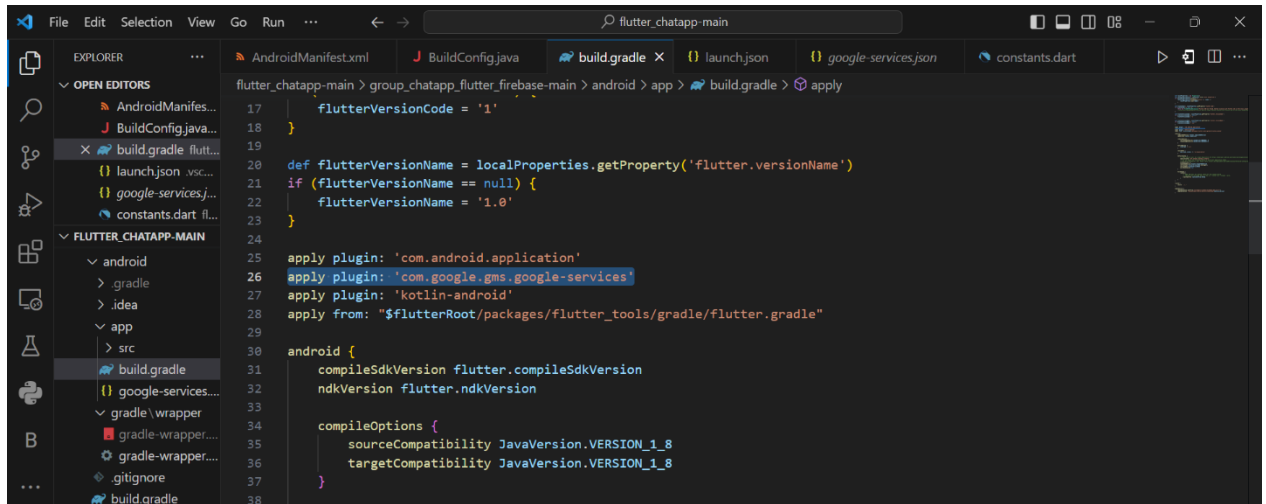
Hình 21: Download file JSON

2.5. Result file JSON



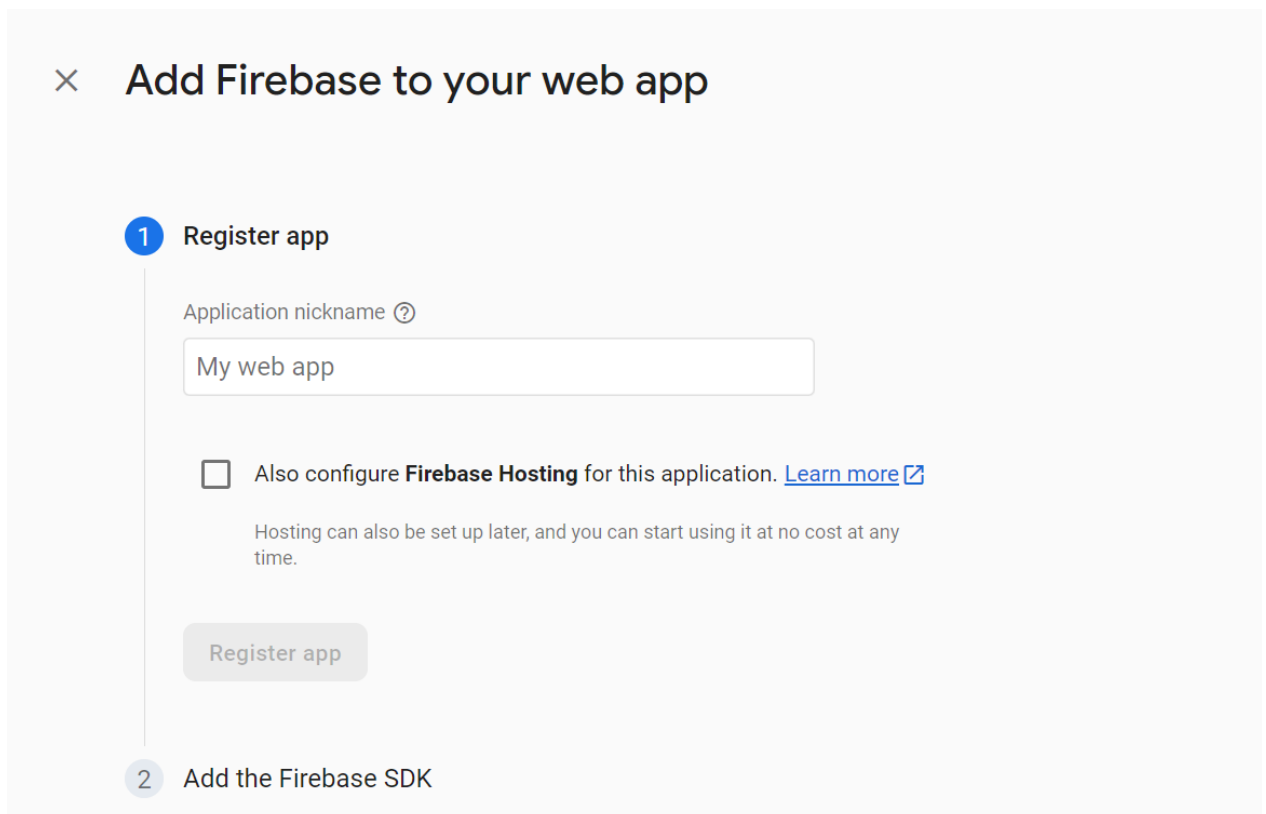
Hình 22: Result file JSON

2.6. Paste vào file build.gradle



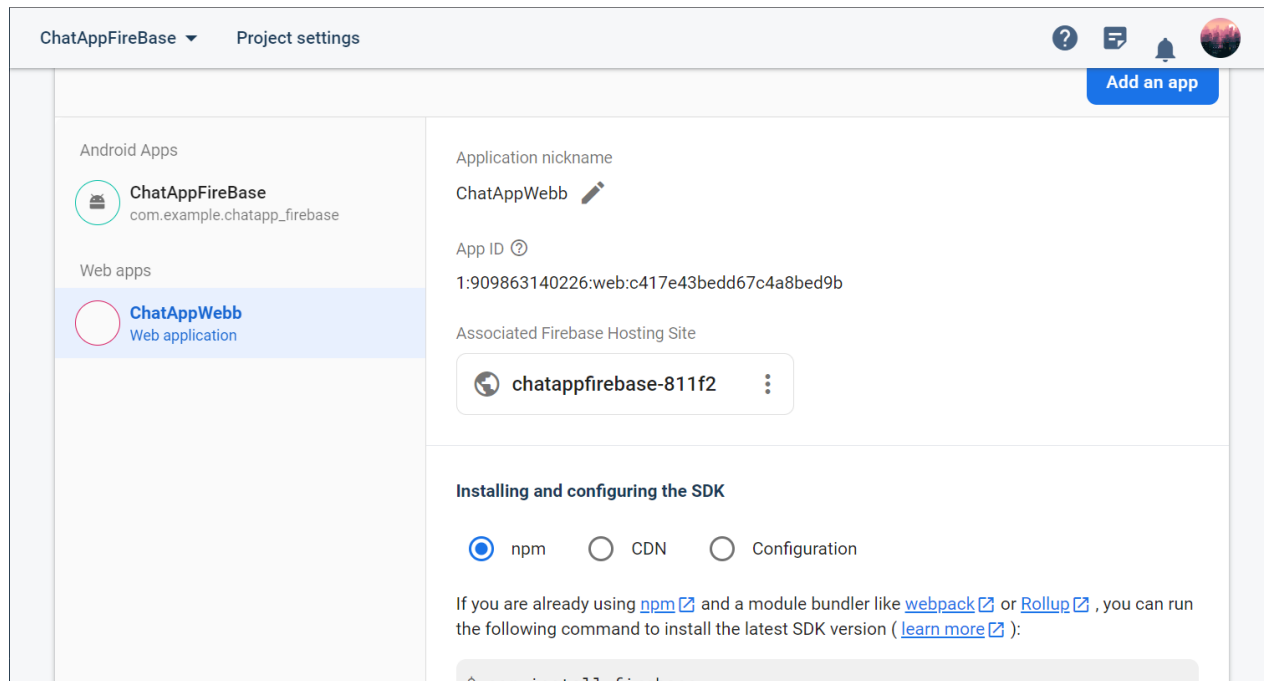
Hình 23: Paste vào file build.gradle

2.7. Add Firebase to webapp



Hình 24: Add Firebase to your web app

2.8. Result



Hình 25: Kết quả

2.9. Copy api key, projectId, appId, messagingSenderId

```
$ npm install firebase
```



Next, initialize Firebase and start mining the SDKs for the products you want to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app" ;
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey : "AIzaSyAkTK8j7r0kzHWwtQg8FG5ijzwLzo5skeI" ,
  authDomain : "chatappfirebase-811f2.firebaseio.com" ,
  projectId : "chatappfirebase-811f2" ,
  storageBucket : "chatappfirebase-811f2.appspot.com" ,
  messagingSenderId : "909863140226" ,
  appId : "1:909863140226:web:c417e43bedd67c4a8bed9b"
};

// Initialize Firebase
const app = initializeApp ( firebaseConfig );
```



Hình 26: Copy API Key

2.10. Paste vào file constants.dart

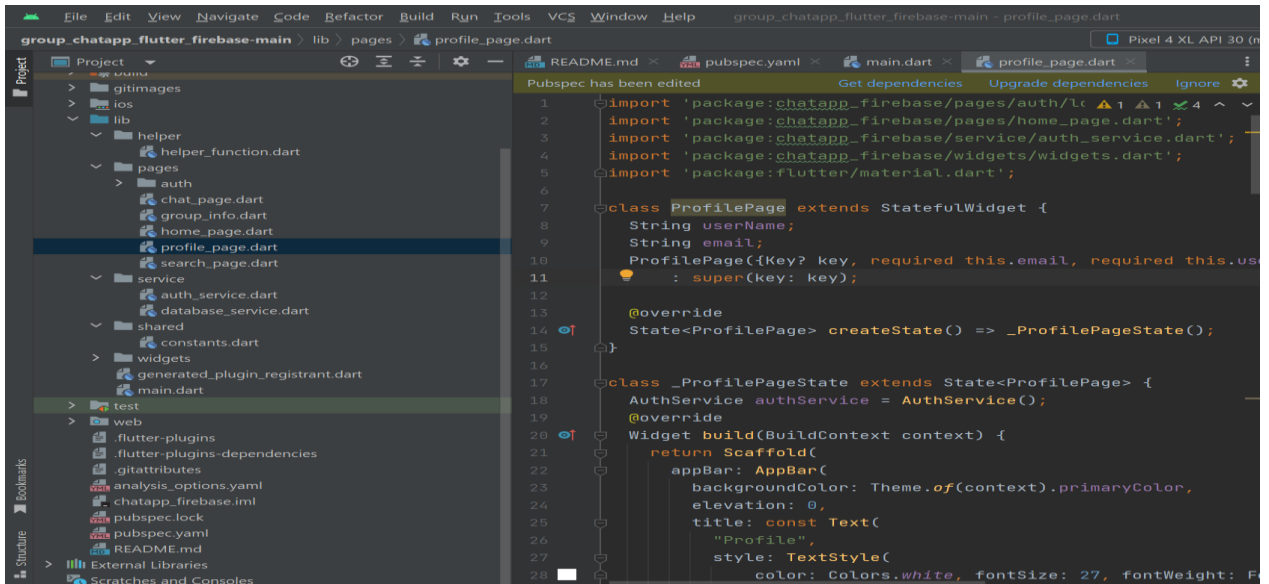
```
AndroidManifest.xml BuildConfig.java build.gradle main.dart launch.json constants.dart X
flutter_chatapp-main > group_chatapp_flutter_firebase-main > lib > shared > constants.dart
1 import 'package:flutter/material.dart';
2
3 class Constants {
4   static String appId = "1:909863140226:web:c417e43bedd67c4a8bed9b";
5   static String apiKey = "AIzaSyAkTK8j7r0kzHWwtQg8FG5ijzwLzo5skeI";
6   static String messagingSenderId = "909863140226";
7   static String projectId = "chatappfirebase-811f2";
8   final primaryColor = const Color(0xFFee7b64);
9 }
10
```

Hình 27: Paste vào file constants.dart

CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM

3.1. Mục Profile Page

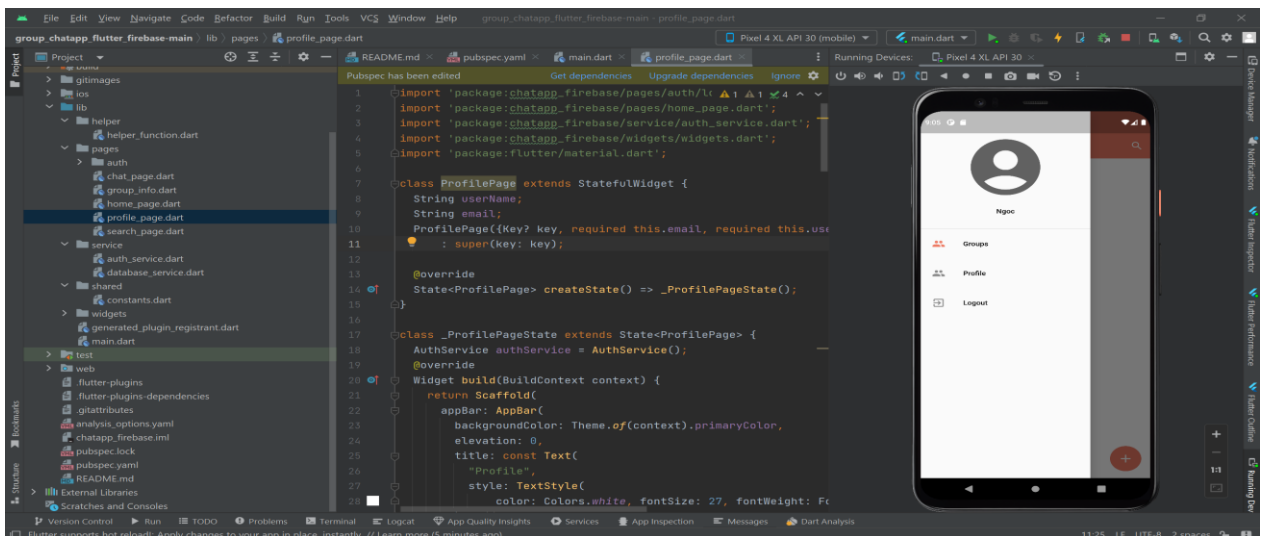
- Source code:



```
1 import 'package:chatapp_firebase/pages/auth/login_page.dart';
2 import 'package:chatapp_firebase/pages/home_page.dart';
3 import 'package:chatapp_firebase/service/auth_service.dart';
4 import 'package:chatapp_firebase/widgets/widgets.dart';
5 import 'package:flutter/material.dart';
6
7 class ProfilePage extends StatefulWidget {
8   String userName;
9   String email;
10  ProfilePage({Key? key, required this.email, required this.userName,
11    : super(key: key)});
12
13  @override
14  State<ProfilePage> createState() => _ProfilePageState();
15
16  }
17
18  class _ProfilePageState extends State<ProfilePage> {
19    AuthService authService = AuthService();
20    @override
21    Widget build(BuildContext context) {
22      return Scaffold(
23        appBar: AppBar(
24          backgroundColor: Theme.of(context).primaryColor,
25          elevation: 0,
26          title: const Text(
27            "Profile",
28            style: TextStyle(
29              color: Colors.white, fontSize: 27, fontWeight: FontWeight.bold,
```

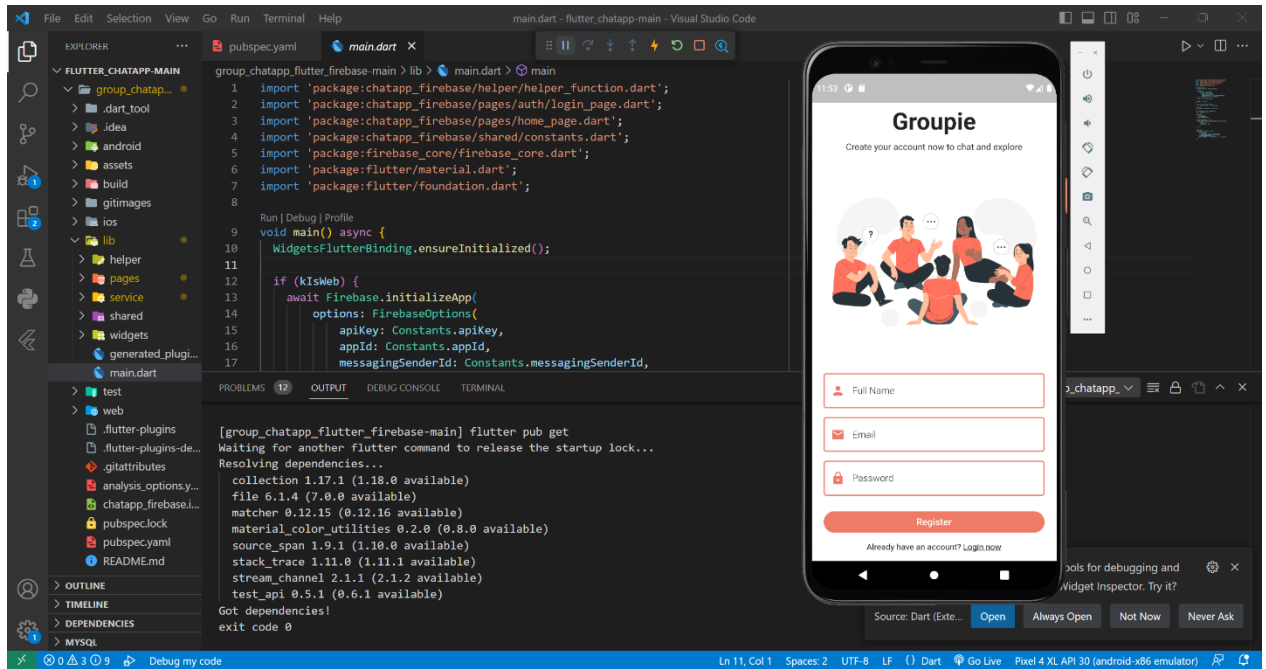
Hình 28: Source code trang Profile

- Kết quả:



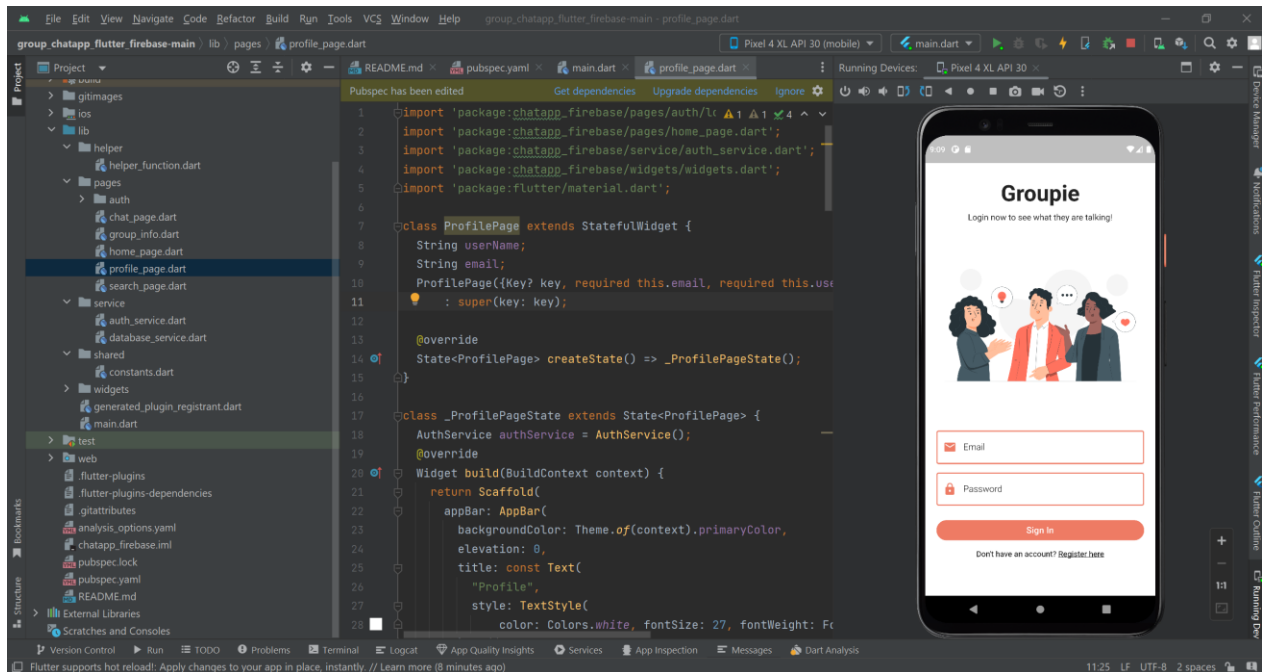
Hình 29: Kết quả Profile Page

3.2. Màn hình đăng ký



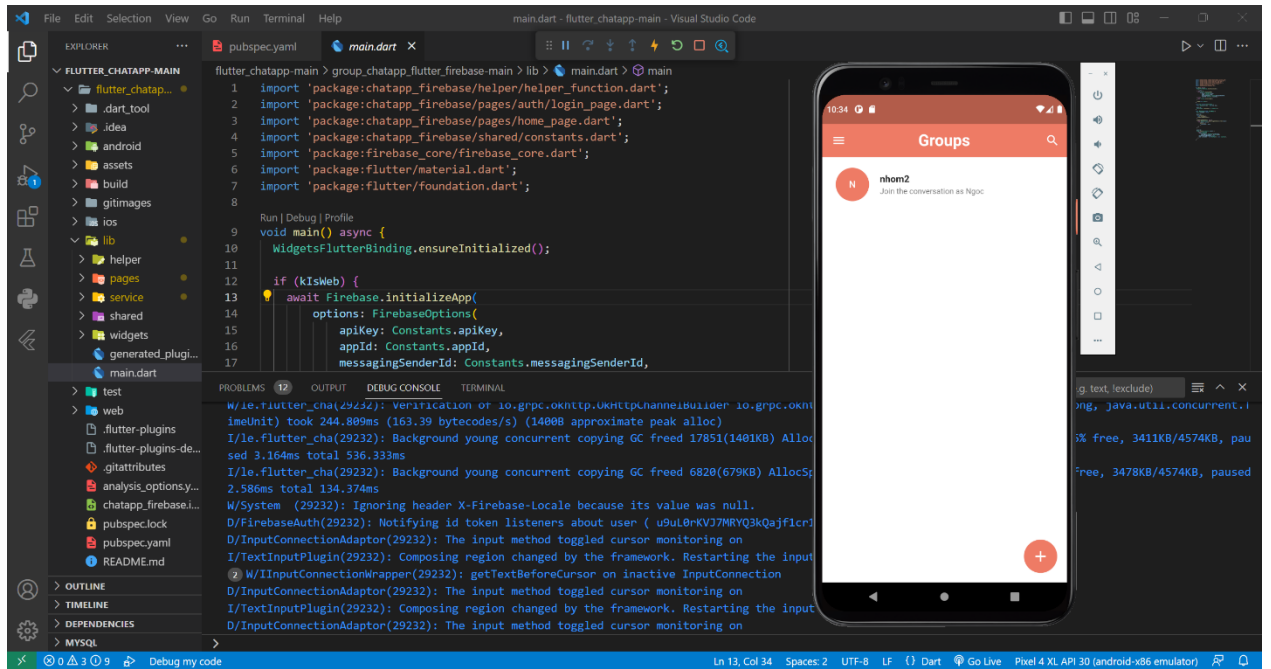
Hình 30: Màn hình đăng ký

3.3. Màn hình đăng nhập



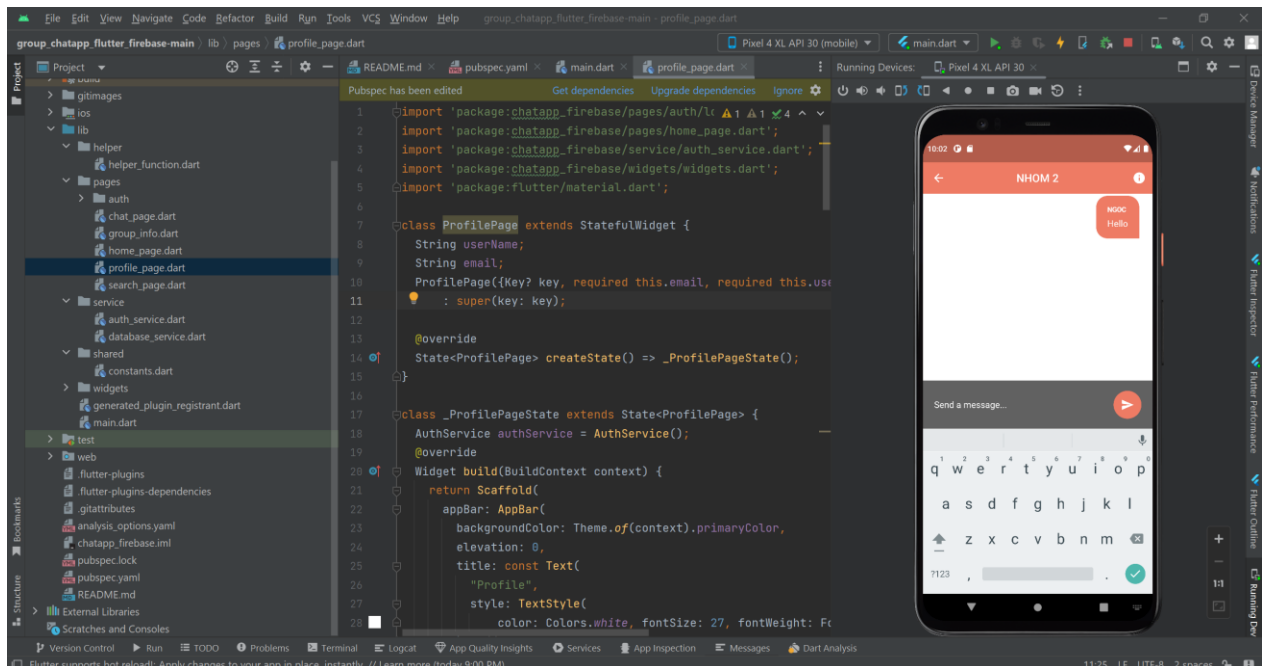
Hình 31: Màn hình đăng nhập

3.4. Màn hình group chat



Hình 32: Màn hình Group chat

3.5. Nhóm chat đã tạo



Hình 33: Màn hình nhóm chat vừa tạo

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Sau khi tiến hành chạy thử nghiệm ứng dụng chatapp nhóm chúng em đã rút ra một số kết luận về ưu điểm của ứng dụng như sau: Ứng dụng có giao diện thân thiện với người sử dụng. Bên cạnh đó ứng dụng Chatapp còn dễ sử dụng nên sẽ giúp cho người sử dụng dễ dàng thao tác trên ứng dụng mà không gây bất cứ khó khăn gì.

Bên cạnh những ưu điểm thì ứng dụng vẫn còn tồn tại những yếu điểm, sai sót như giao diện chưa thực sự bắt mắt. Ứng dụng không cho phép sử dụng tài khoản google và facebook để đăng nhập. Khiến cho ứng dụng chưa được hoàn thiện ở mức tối đa.

Trong tương lai nhóm em sẽ cố gắng hoàn thiện những yếu điểm trên để phát triển đề tài sâu hơn nữa. Tuy kiến thức còn hạn hẹp và thời gian ngắn nên đề tài không thể tránh khỏi những sai sót nhưng chúng em cũng đã cố gắng tìm tòi và học hỏi cố gắng hoàn thành những mục tiêu ban đầu đề ra. Mong thầy góp ý để nhóm chúng em rút kinh nghiệm và hoàn thành tốt hơn trong khoá luận tốt nghiệp sắp tới. Em xin trân thành cảm ơn!

DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng việt:

- [1] [Flutter là gì? Ưu, nhược điểm của Flutter - Joboko](#)
- [2] [Flutter là gì? Giải thích về Flutter – AWS \(amazon.com\)](#)
- [3] [API là gì? Những phương pháp bảo mật API đảm bảo hiệu quả \(itviec.com\)](#)
- [4] [API là gì? Những đặc điểm nổi bật của Web API | TopDev](#)
- [5] [Cơ bản về giao thức Websocket và thư viện Socket.io \(viblo.asia\)](#)
- [6] [Socket IO là gì? Hướng dẫn sử dụng socket io cơ bản \(bizflycloud.vn\)](#)
- [7] [Tạo ứng dụng Chat realtime sử dụng Socket.io - VNTALKING](#)
- [8] [Websocket là gì? Ưu - nhược điểm ra sao? Cách kết nối thế nào? - Tin tức tên miền hosting \(tenten.vn\)](#)
- [8] [Tìm hiểu về Socket.io – Ren \(ren0503.github.io\)](#)
- [9] [Lập trình Dart - Flutter \(xuanthulab.net\)](#)
- [10] [Giới thiệu về ngôn ngữ lập trình Dart - w3seo tìm hiểu về Dart \(websitehcm.com\)](#)
- [11] [\(2\) Mô hình Client - Server | Mô hình toàn bộ các websites sử dụng? - YouTube](#)
- [12] [Tìm hiểu về Firebase Realtime Database \(viblo.asia\)](#)

Tiếng anh:

- [1] [Flutter documentation | Flutter](#)
- [2] [Firebase Documentation \(google.com\)](#)