TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP
Hệ thống quản lý và điều phối tin nhắn đa kênh

Lý Bá Tuấn
Tuan.lb173438@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn: TS.Nguyễn Thị Thu Trang _____

Bộ môn: Công nghệ phần mềm
Viện: Công nghệ thông tin – Truyền thông

HÀ NỘI, 06/2021

Lời cam kết

Họ và tên sinh viên: . . . . . .

Điện thoại liên lạc: . . . . . Email: . . . . . .

Lớp: . . . . . Hệ đào tạo: . . . . . .

Tôi – *Lý Bá Tuấn* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *tiến sỹ Nguyễn Thị Thu Trang*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày tháng năm

Tác giả ĐATN

Họ và tên sinh viên

Lời cảm ơn

Lời đầu tiên em xin gửi lời cảm ơn chân thành nhất tới TS. Nguyễn Thị Thu Trang đã luôn nhiệt tình hướng dẫn, chỉ bảo và giúp đỡ em trong suốt quá trình tham gia làm đồ án tốt nghiệp.

Lời tiếp theo em xin dành lời cảm ơn tới các anh chị, bạn bè, các thành viên trong Lab 914 – Thư viện Tạ Quang Bửu, đặc biệt là hai anh chị Phạm Quang Minh và Phí Khánh Huyền đã luôn trao đổi, chia sẻ kinh nghiệm và góp ý cho em, giúp em giải quyết được nhiều khúc mắc trong quá trình làm đồ án. Cám ơn mọi người đã luôn ủng hộ em về mặt tinh thần, để em có động lực học tập và hoàn thành ĐATN này.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, bạn bè đã luôn giúp đỡ động viên và tạo điều kiện tốt nhất để em có thể thực hiện hoàn thành đồ án tốt nghiệp này.

Trong quá trình xây dựng và hoàn thiện báo cáo cũng như đồ án tốt nghiệp, em sẽ không tránh khỏi những sai sót, vì thế em rất mong các thầy cô và các bạn đọc góp ý để em có thể hoàn thiện hơn nữa sản phẩm này.

Em xin chân thành cảm ơn !

Tóm tắt

Với sự bùng nổ của mạng xã hội và sự phát triển của nền kinh tế số, các doanh nghiệp đã không còn tiếp cận tới khách hàng 1 cách trực tiếp hay phải thông qua các dịch vụ gọi điện qua số điện thoại thông thường nữa. Ngày nay, các doanh nghiệp phải tiếp cận tới các tệp khách hàng từ nhiều kênh khác nhau, với đa dạng các hình thức từ nhắn tin đến trò chuyện trực tiếp thông qua các thiết bị thông minh. Việc chăm sóc khách hàng một cách kịp thời, nhanh chóng và hiệu quả trên nhiều kênh (mạng xã hội Facebook, Zalo, Viber, Telegram, Website,...) sẽ giúp doanh nghiệp mở rộng tệp khách hàng tốt hơn, được đánh giá cao trong việc chăm sóc khách hàng. Đi cùng với việc chăm sóc khách hàng là việc quản lý và theo dõi năng suất làm việc của các tổng đài viên theo thời gian và các ca làm việc, để có thể kiểm soát tốt từng cá nhân trong quá trình làm việc, thúc đẩy năng suất làm việc của mỗi người một các hợp lý. Vì thế, mục đích của đồ án tốt nghiệp là phát triển một hệ thống nhắn tin giữa các tổng đài viên với khách hàng từ nhiều kênh khác nhau, tập trung tại một nguồn và có thể điều phối tin nhắn tới các tổng đài viên sao cho phù hợp nhất. Hơn nữa, hệ thống còn được tích hợp với chatbot để tự động trả lời các câu hỏi thường gặp của khách hàng. Hệ thống được xây dựng dựa trên kiến trúc Microservices và chủ yếu tập trung vào nghiệp vụ điều phối tin nhắn từ khách hàng tới các tổng đài viên cũng như việc thống kê giám sát hiệu suất làm việc của các tổng đài viên. Có rất nhiều công nghệ mới được sử dụng trong đồ án, ví dụ như: NodeJS, Express.js, MongoDB, Redis, RabbitMQ để xây dựng các module phía backend; thư viện ReactJS nổi tiếng để xây dựng giao diện người dùng; Docker để triển khai hệ thống. Hiện tại, hệ thống đã được triển khai thử nghiệm tại: https://portal.iristech.club. Hệ thống vẫn đang trong quá trình thử nghiệm, phát triển và tối ưu các chức năng để chuẩn bị đưa tới tay người dùng.

Mục lục

Danh mục hình vẽ

Danh mục bảng

Danh mục các từ viết tắt

API	Application Programming Interface Giao diện lập trình ứng dụng
HTML	HyperText Markup Language Ngôn ngữ đánh dấu siêu văn bản
CNTT	Công nghệ thông tin
ĐATN	Đồ án tốt nghiệp
UI	User Interface
JSX	JavaScript XML
AMQP	Advanced Message Queue Protocol
HTTP	HyperText Transfer Protocol
DOM	Document Object Model
URL	Uniform Resource Locator

Danh mục thuật ngữ

Browser	Trình duyệt
Request	Yêu cầu truy xuất dữ liệu
Framework	Bộ khung làm việc được xây dựng sẵn
Chatbot	Hệ thống kết hợp với trí tuệ nhân tạo để tương tác với con người
Livechat	Công cụ nhắn tin trực tuyến trên website

Chương 1 Giới thiệu đề tài

Chương 1 này giới thiệu các vấn đề thực tế dẫn tới việc em lựa chọn đề tài, tổng quan về hệ thống quản lý và điều phối tin nhắn đa kênh. Sau đó nêu ra mục tiêu và phạm vi đề tài, định hướng giải pháp và bố cục của đồ án.

1.1 Đặt vấn đề

Việc chăm sóc khách hàng nhanh chóng và kịp thời sẽ ghi được rất nhiều điểm trong mắt khách hàng của doanh nghiệp. Hiện nay có rất nhiều hình thức chăm sóc khách hàng trực tuyến như: chăm sóc qua email, chăm sóc qua điện thoại, chăm sóc qua livechat. Trong số đó, chăm sóc khách hàng qua livechat được cho là mới nhất, nhanh chóng nhất. Livechat là một giải pháp nhắn tin trực tuyến giữa khách hàng và các tư vấn viên khi khách hàng truy cập vào website của doanh nghiệp hoặc nhắn tin thông qua các mạng xã hội nếu hệ thống livechat đó cho phép tích hợp. Trên thị trường hiện nay cũng có nhiều ứng dụng hỗ trợ các doanh nghiệp nhắn tin với khách hàng như Livechat.com, tawk.to, subiz.com.vn. Tuy nhiên, các ứng dụng này còn nhiều mặt hạn chế như: hình thức trả lời là trực tiếp giữa các tư vấn viên hỗ trợ với khách hàng, hạn chế về mặt ngôn ngữ, đồng thời các kênh nhắn tin được tích hợp vào ứng dụng không phù hợp với đại đa số người dùng trong nước. Từ những nhu cầu thực tế và các hạn chế của những sản phẩm livechat trên thị trường, em thực hiên đề tài "xây dựng hệ thống điều phối và quản lý tin nhắn đa kênh". Mục đích của đề tài là xây dựng hệ thống nhắn tin có thể tự động chia các tin được gửi tới từ nhiều kênh khác nhau cho các nhân viên chăm sóc khách hàng sao cho hiệu quả và đều đặn nhất. Đồng thời cho phép người quản lý giám sát và thống kê báo cáo hiệu suất làm việc của các nhân viên một cách nhanh chóng, dễ dàng và chính xác.

1.2 Mục tiêu và phạm vi đề tài

Với những vấn đề đã trình bày trong mục 1.1, đồ án này sẽ giải quyết một số vấn đề như sau: Thứ nhất, hệ thống có chức năng cho phép quản lý lên lịch làm việc cho các nhân viên tư vấn. Thứ hai, hệ thống cho phép người dùng nhắn tin và nhận tin nhắn từ các ứng dụng đã tích hợp như website, Facebook, Zalo, Viber, Telegram. Thứ ba, hệ thống có chức năng tự động phân chia tin nhắn cho các nhân viên chăm sóc khách hàng đã đăng ký ca làm việc trước đó và đang trực tuyến và chức năng tự nhận tin nhắn từ danh sách tin đang chờ để thực hiện chăm sóc khách hàng. Thứ tư, hệ thống phân chia quyền, chỉ người có quyền là quản

lý mới có thể xem toàn bộ tin nhắn mà Bot đang chat và các tin nhắn mà các nhân viên đang chăm sóc. Riêng các nhân viên thì chỉ có thể xem được các tin đang ở trạng thái chờ và các tin mà nhân viên đó đang phục vụ. Cuối cùng, hệ thống có chức năng giám sát theo thời gian thực các nhân viên trong ca làm việc. Có chức năng thống kê báo cáo, tổng hợp số liệu đối với từng nhân viên và trên toàn bộ cả hệ thống.

1.3 Định hướng giải pháp

Theo cách phát triển ứng dụng truyền thống, toàn bộ mã nguồn của ứng dụng sẽ được viết trong 1 project, tất cả các tính năng sẽ được gom lại thành một khối. Cách phát triển này thì đơn giản, phù hợp với hầu hết các dự án. Tuy nhiên khi ứng dụng cần mở rộng tính năng, gia tăng số lượng người dùng, yêu cầu giao tiếp với các hệ thống khác tăng lên thì lúc này mã nguồn sẽ ngày một phình to ra, khiến cho dự án ngày một chậm đi về cả mặt hiệu năng lẫn việc phát triển và sửa lỗi. Chính vì thế, em lựa chọn cách tiếp cận theo mô hình kiến trúc Microservices. Theo đó, các tính năng có liên quan tới nhau sẽ được gom lại thành một dịch vụ tương tự như mô hình nguyên khối truyền thống. Các dịch vụ con này hoạt động độc lập, tạo sự liên kết lỏng lẻo giữa các thành phần, không phục thuộc quá nhiều lẫn nhau. Tính năng quản lý ca làm việc sẽ do dịch vụ quản lý đảm nhiệm, hệ thống chat sẽ do dịch vụ nhắn tin thực hiện riêng.

Do phần backend được thiết kế theo mô hình nhiều dịch vụ, các dịch vụ cần phải giao tiếp được với nhau, vì thế em đã sử dụng công nghệ của RabbitMQ, giúp các dịch vụ dễ dàng giao tiếp với nhau thông qua hàng đợi tin nhắn. Em cũng sử dụng hàng đợi tin nhắn để thực hiện điều phối tin nhắn tự động. dịch vụ nhắn tin cũng được em sử dụng Websocket để tạo các kết nối tới máy của người dùng thực hiện các chức năng như gửi tin nhắn, giám sát thay đổi theo thời gian thực. Việc lưu trữ dữ liệu ở các dịch vụ em đã chọn hệ quản trị cơ sở dữ liệu MongoDB vì đây là một kiểu cơ sở dữ liệu phi quan hệ, có cấu trúc linh hoạt, tốc độ đọc, ghi nhanh hơn so với các kiểu dữ liệu SQL, rất phù hợp với một hệ thống nhắn tin cần cập nhật dữ liệu liên tục. Em cũng sử dụng thêm công nghệ của Redis, cũng là một công nghệ lưu trữ dữ liệu để có thể tăng tốc độ cho hệ thống và quản lý phiên người dùng truy cập.

Đối với thành phần giao diện tức phần frontend, em lựa chọn cách phát triển ứng dụng theo kiểu Single-page Application (SPA). Bằng cách này, phần frontend của hệ thống sẽ được tách biệt khỏi backend và giao tiếp thông qua API. Phần frontend em chọn sử dụng thư viện ReactJS, một thư viện đang rất nổi tiếng cho lập trình giao diện trên website hoạt động bằng cách phân chia các mọi thứ trên giao diện thành các thành phần nhỏ để thuận tiện cho việc quản lý, tái sử dụng và mở rộng ứng dụng.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 em sẽ trình bày về tổng quan các chức năng trong hệ thống đồng thời đặc tả một số usecase chính.

Trong chương 3, em sẽ trình bày về các công nghệ và công cụ em sử dụng xuyên suốt quá trình làm đồ án.

Trong chương 4, em sẽ trình bày chi tiết về kiến trúc hệ thống, quá trình xây dựng và triển khai hệ thống.

Trong chương 5 em sẽ trình bày các giải pháp và đóng góp nổi bật của bản than trong suốt quá trình làm ĐATN như đưa ra giải pháp, xây dựng giao diện, kiến trúc và cách em giải quyết các vấn đề trong quá trình làm đồ án.

Trong chương 6, em sẽ kết luận nội dung của đồ án, nêu ra các ưu nhược điểm, những gì em đã và chưa làm được. Đồng thời, em sẽ đề xuất các định hướng em muốn phát triển hệ thống trong tương lai.

Chương 2 Khảo sát và phân tích yêu cầu

Chương 2 em sẽ trình bày khảo sát của em về một số ứng dụng livechat hiện có trên thị trường. Từ đó đưa ra tổng quan chức năng của hệ thống em sẽ xây dựng và làm rõ chi tiết từng chức năng.

2.1 Khảo sát hiện trạng

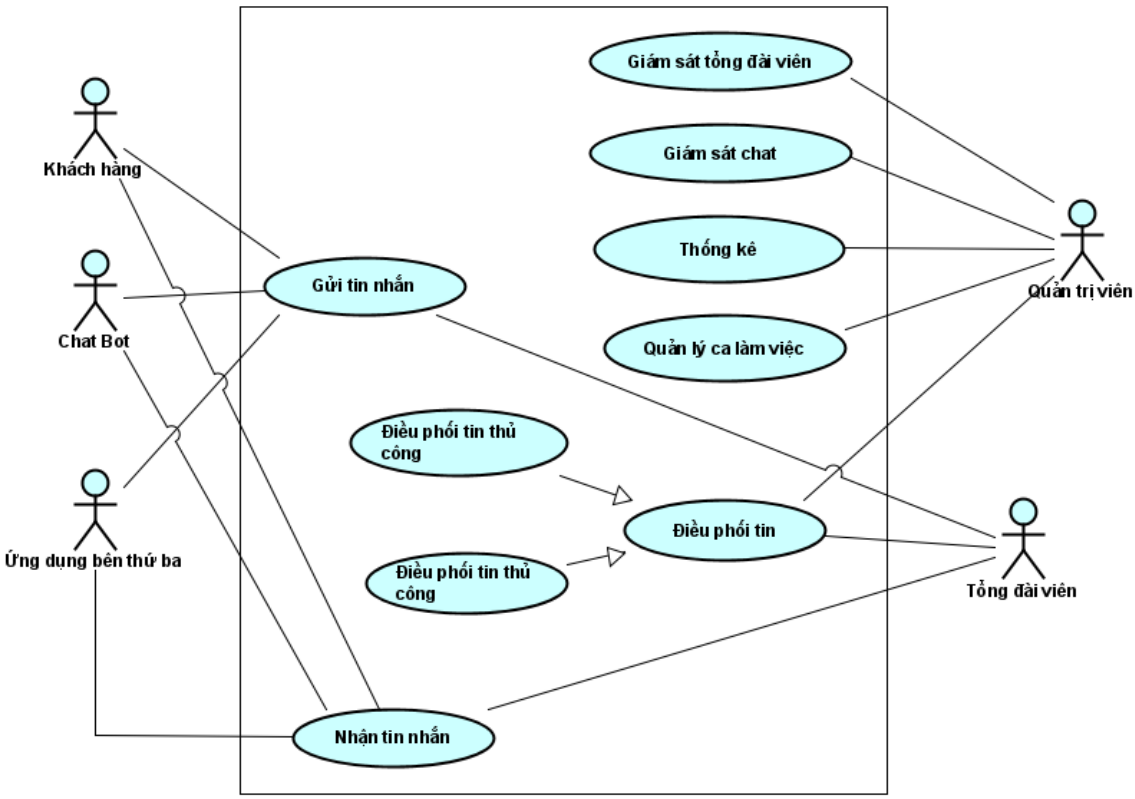
Hiện nay trên thị trường có rất nhiều ứng dụng livechat với các tính năng đa dạng. Trong đó, Livechat.com và Tawk.to là hai ứng dụng nổi bật hơn cả, tuy nhiên đó là các ứng dụng rất lớn của các công ty đa quốc gia, không có hỗ trợ tiếng việt như Livechat và có hỗ trợ tiếng việt như Tawk.to thì lại không rõ ràng và đầy đủ. Livechat thì có rất nhiều tính năng, nhưng trong những tính năng đấy sẽ trở nên thừa hoặc rất ít khi sử dụng đồng thời không phù hợp cho đại đa số nhu cầu của người Việt. Tawk.to thì lại không có tích hợp các kênh nhắn tin từ các mạng xã hội như Messenger của Facebook, hoặc Zalo, …

Ở thị trường Việt Nam, em có biết đến một vài ứng dụng về mảng livechat cũng đang cạnh tranh nhau trên thị trường. Cụ thể là sản phẩm của Subiz livechat cung cấp các chức năng nhắn tin, chăm sóc khách hàng cho doanh nghiệp. Sản phẩm này có nhiều tính năng giúp doanh nghiệp nắm bắt được nhu cầu, thói quen của khách hàng, giúp doanh nghiệp tư vấn và hỗ trợ khách hàng của họ nhanh chóng và tiện lợi hơn. Tuy nhiên Subiz Livechat hiện chỉ cho phép nhắn tin qua kênh tùy chỉnh, email và Messager của Facebook.

2.2 Tổng quan chức năng

2.2.1 Biểu đồ use case tổng quan

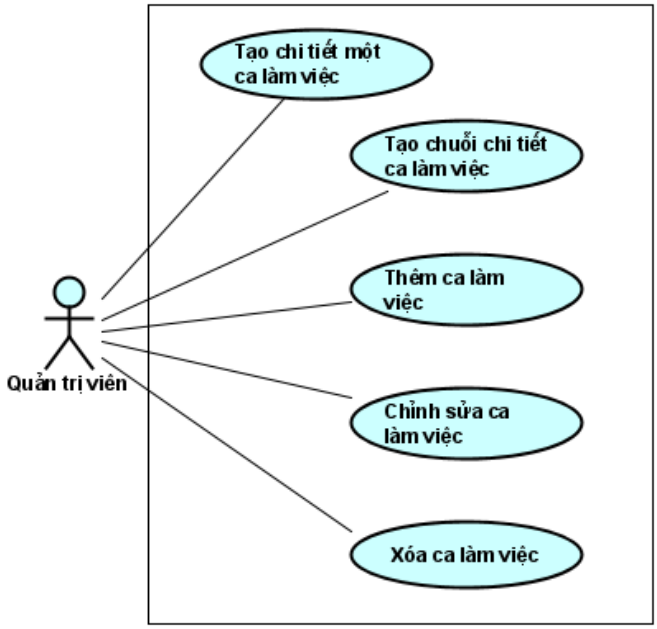
Hệ thống quản lý điều phối tin nhắn đa kênh gồm nhiều thành phần chức năng khác nhau. **Hình 1** mô tả tổng quan toàn bộ chức năng của hệ thống.



Hình 1 Biểu đồ usecase tổng quan

Hệ thống gồm có 5 tác nhân là quản trị viên, tổng đài viên, Chatbot, khách hàng, ứng dụng bên thứ ba như Facebook, Zalo, Viber, Telegram. Khách hàng và ứng dụng bên thứ 3 có thể gửi và nhận tin nhắn đến cho chatbot hoặc tổng đài viên. Khi các tin nhắn được trực tiếp chuyển hướng tới cho tổng đài viên thì tổng đài viên có thể thực hiện các chức năng của điều phối tin nhắn. Quản trị viên có thể xem các thống kê mà hệ thống cung cấp, giúp có cái nhìn tổng quan và đánh giá một cách khách quan năng suất làm việc của các tổng đài viên. Quản trị viên cũng có thể quản lý ca làm việc cho các tổng đài viên, giúp cho hệ thống biết được những tổng đài viên đang trong ca làm việc để có thể phân chia tin và tính toán hiệu năng hợp lý. Ngoài ra, quản trị viên cũng có chức năng giám sát theo thời gian thực tổng đài viên và các tin nhắn được bắn vào hệ thống.

2.2.2 Biểu đồ use case phân rã “quản lý ca làm việc”

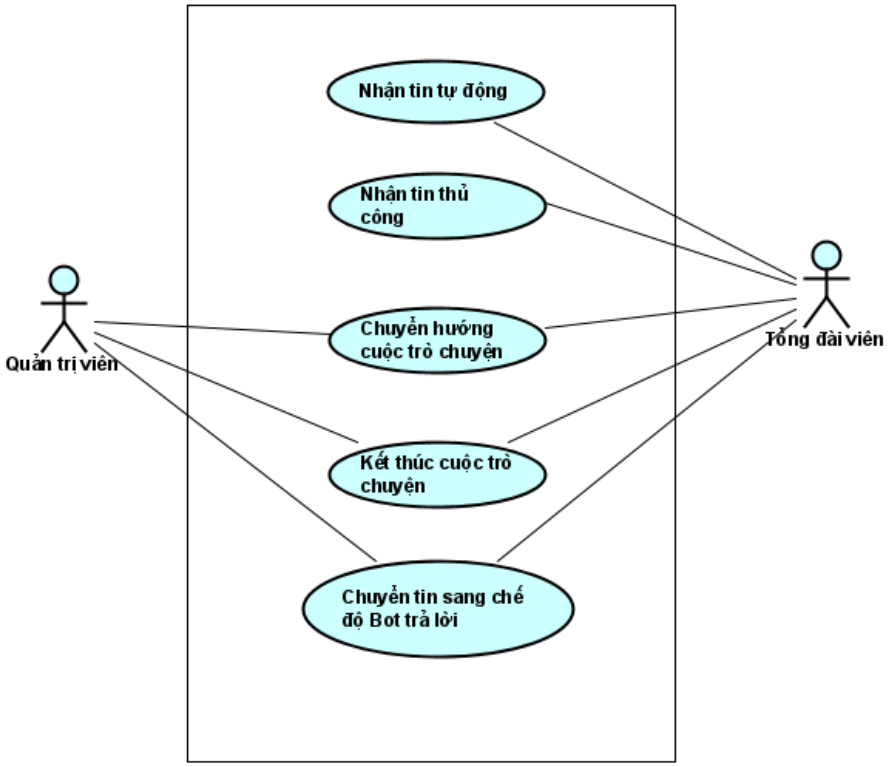


Hình 2 Biểu đồ usecase phân rã “quản lý ca làm việc”

Hệ thống có chức năng quản lý ca làm việc cho tổng đài viên, giúp người quản trị viên dễ dàng hơn trong việc theo dõi công việc hằng ngày, hằng giờ theo ca làm cụ thể và đánh giá tốt hơn trong việc theo dõi hiệu suất làm việc. Quản trị viên có thể Thêm ca làm việc trong ngày, chỉnh sửa ca làm việc, xóa ca làm việc, thực hiện tạo các chuỗi chi tiết ca làm việc cho các tổng đài viên.

2.2.3 Biểu đồ use case phân rã “điều phối tin”

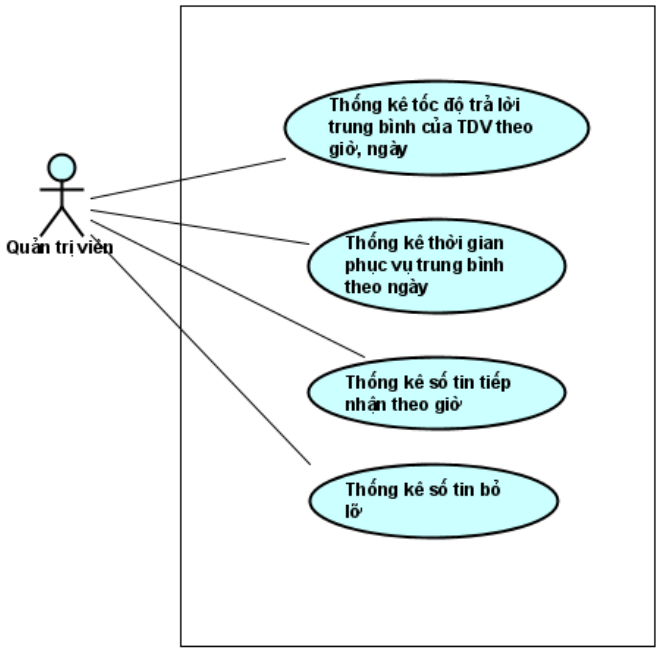
Tổng đài viên có thể tự mình nhận các tin nhắn về để chat khi hệ thống đang ở chế độ điều hướng thủ công hoặc tự động được phân tin hay nhận tin tự động khi hệ thống đang ở chế độ điều hướng tin tự động. Hai chức năng nhận tin tự động và nhận tin thủ công sẽ không dành cho quản trị viên vì quản trị viên đã có hoàn toàn quyền tham gia vào mọi cuộc trò chuyện rồi. Trong quá trình trò chuyện với khách hàng, tổng đài viên có thể thực hiện các hành động như chuyển hướng cuộc trò chuyện, kết thúc cuộc trò chuyện, chuyển tin sang chế độ Bot trả lời. Ngoài tổng đài viên ra thì quản trị viên cũng có thể thực hiện các hành động kết thúc hay chuyển tiếp cuộc trò chuyện bất cứ lúc nào và bất cứ khách hàng nào mà không có điều kiện ràng buộc nào cả.



Hình 3 Biểu đồ usecase phân rã "điều phối tin"

2.2.4 Biểu đồ use case phân rã "thống kê"

Hệ thống cung cấp chức năng thống kê giúp cho quản trị viên có cái nhìn tổng quan về hệ thống cũng như năng suất làm việc của các tổng đài viên, từ đó đưa ra các biện pháp phù hợp và kịp thời cho đội ngũ chăm sóc khách hàng khi cần thiết. Các chức năng thống kê mà hệ thống cung cấp gồm có: thống kê tốc độ trả lời của tổng đài viên theo ngày, theo giờ; thống kê thời gian phục vụ trung bình và số khách hàng đã phục vụ theo giờ; thống kê số tin tiếp nhận theo giờ, theo ca, theo ngày; thống kê số tin bị bỏ lỡ theo giờ theo ngày, theo ca.

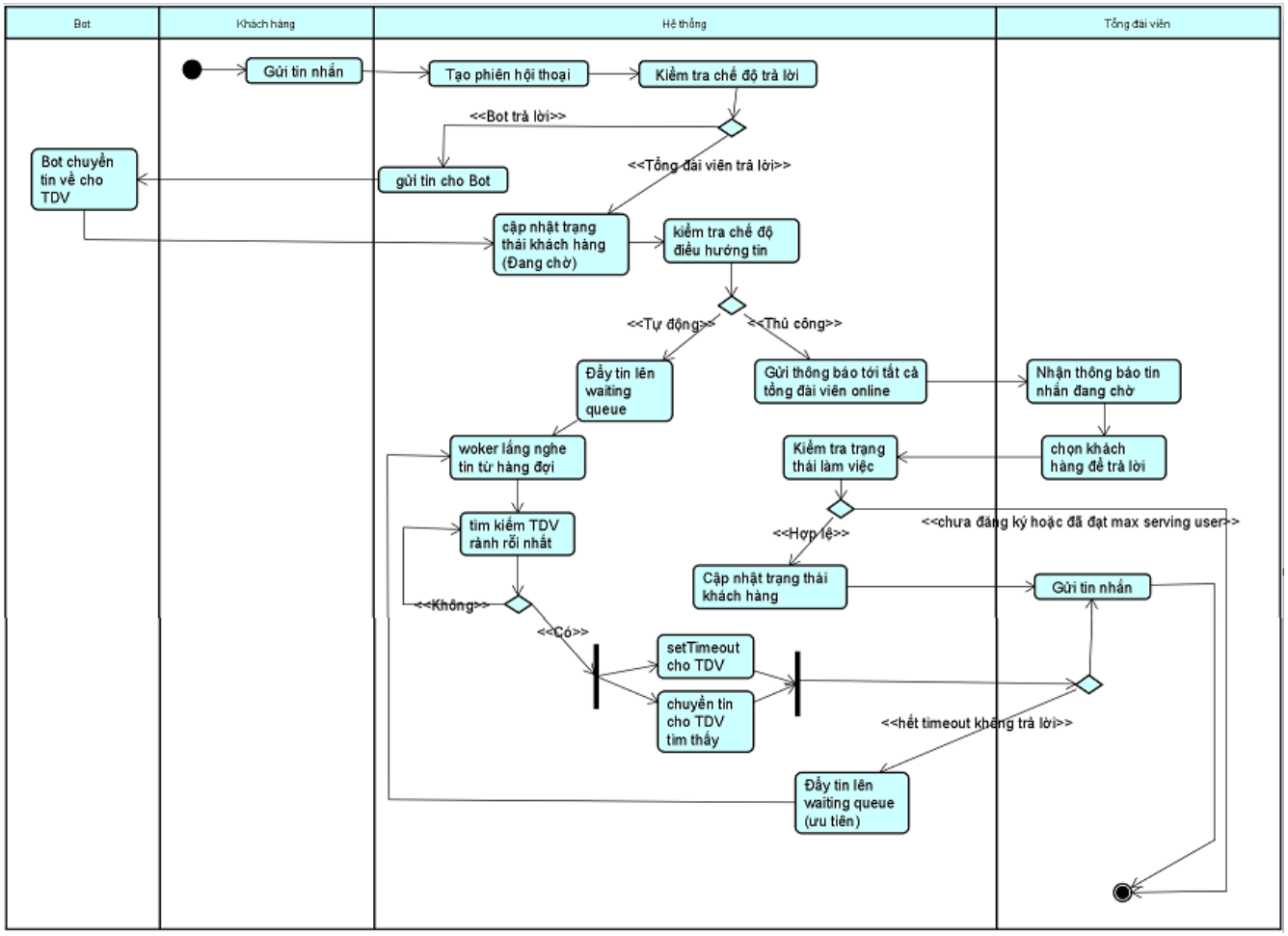


Hình 4 Biểu đồ usecase phân rã thống kê

2.2.5 Quy trình nghiệp vụ

2.2.5.1 Quy trình nghiệp vụ điều phối tin

Hình 5 mô tả quy trình nghiệp vụ của chức năng điều phối tin. Hệ thống cho phép cấu hình hai chế độ điều phối là tự động và thủ công. Khi các tin nhắn được gửi từ khách hàng hoặc được chuyển tiếp từ Bot trả về cho tổng đài viên thì hệ thống sẽ cập nhật trạng thái của khách hàng trên hệ thống là đang chờ và sau đó tùy theo chế độ điều phối tin được cài đặt để điều phối tin cho hợp lý. Nếu là chế độ điều phối thủ công, thì hệ thống sẽ gửi cho những tổng đài viên đang trực tuyến danh sách các khách hàng đang chờ. Từ đó, các tổng đài viên sẽ lựa chọn những khách hàng mà mình muốn trả lời. Còn nếu là chế độ điều phối tin tự động, các tổng đài viên sẽ không được chọn các khách hàng để mà phục vụ, thay vào đó thì hệ thống sẽ tự động tìm kiếm tổng đài viên đang rảnh rỗi nhất để chỉ định cho người đó. Sau khi được chỉ định mà tổng đài viên không trả lời khách hàng sau 3 phút thì tin nhắn đó sẽ lập tức được tự động chuyển hướng cho tổng đài viên khác. Nội dung này em sẽ trình bày cụ thể và rõ ràng hơn ở Chương 5.



Hình 5 Quy trình nghiệp vụ điều phối tin

2.3 Đặc tả chức năng

Phần này sẽ đặc tả một số chức năng chính của hệ thống. **Bảng 1** liệt kê 19 usecase được sử dụng. Do kích thước báo cáo có hạn nên em sẽ trình bày 3 usecase: Nhận tin thủ công, Chuyển hướng cuộc trò chuyện, Tạo chuỗi chi tiết ca làm việc.

Bảng 1 Danh sách usecase

Hệ thống	**Nhóm use case**	**Mã use case**	**Tên use case**
Hệ thống quản lý và điều phối tin nhắn đa kênh	Điều phối tin	UC001	**Nhận tin thủ công**
		UC002	Nhận tin tự động
		UC003	**Chuyển hướng cuộc trò chuyện**
		UC004	Kết thúc cuộc trò chuyện
		UC005	Xem thông tin chi tiết khách hàng
		UC006	Tắt chế độ bot trả lời
	Giám sát	UC007	Giám sát tổng đài viên thời gian thực
		UC008	Giám sát cuộc trò chuyện thời gian thực
	Thống kê	UC009	Thống kê tốc độ trả lời trung bình của TDV.
		UC010	Thống kê thời gian phục vụ trung bình.
		UC011	Thống kê số tin tiếp nhận của TDV.

		UC012	Thống kê số tin TDV bỏ lỡ.
		UC013	Thống kê số tin chuyển hướng.
Hệ thống quản lý nhân viên		UC014	Tạo ca làm việc
		UC015	Chỉnh sửa ca làm việc
		UC016	Xóa ca làm việc
		UC017	**Thiết lập chuỗi các chi tiết ca làm việc**
		UC018	Tạo một chi tiết ca làm việc
		UC019	Hủy chi tiết ca làm việc

2.3.1 Đặc tả use case "Nhận tin thủ công"

Bảng 2 Đặc tả usecase "nhận tin thủ công"

<table>
<tr><td>Mã use case</td><td>UC001</td><td>Tên use case</td><td>Nhận tin thủ công</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Tổng đài viên</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Đã đăng ký ca làm việc</td></tr>
<tr><td>Luồng sự kiện chính (Thành công)</td><td colspan="3"><table>
<tr><th>STT</th><th>Thực hiện bởi</th><th>Hành động</th></tr>
<tr><td>1.</td><td>Tổng đài viên</td><td>Chọn mục danh sách đang chờ</td></tr>
<tr><td>2.</td><td>Hệ thống</td><td>Hiển thị danh sách khách hàng đang chờ</td></tr>
<tr><td>3.</td><td>Tổng đài viên</td><td>Chọn một khách hàng để trả lời</td></tr>
</table></td></tr>
</table>

	4.	Hệ thống	Kiểm tra số lượng khách hàng mà TDV đang phục vụ.
	5.	Hệ thống	Cập nhật trạng thái khách hàng trên hệ thống sang đang TDV đang phục vụ.
	6.	Hệ thống	Thông báo tới tất cả những TDV khác khách hàng này đã được chọn.
	7.	Hệ thống	Gửi thông tin giám sát
Luồng sự kiện thay thế	**STT**	**Thực hiện bởi**	**Hành động**
	4a.	Hệ thống	Thông báo lỗi nếu TDV đang phục vụ tối đa khách hàng cho phép
	5a.	Hệ thống	Thông báo lỗi nếu cập nhật trạng thái khách hàng thất bại.

2.3.2 Đặc tả use case "chuyển hướng cuộc trò chuyện"

Bảng 3 Đặc tả usecase "chuyển hướng cuộc trò chuyện"

Mã use case	UC003	**Tên use case**	Chuyển hướng cuộc trò chuyện
Tác nhân	Người quản trị, tổng đài viên		
Tiền điều kiện	Tồn tại ca làm việc, và tổng đài viên		
Luồng sự kiện chính (Thành công)	**STT**	**Thực hiện bởi**	**Hành động**
	1.	Người dùng	Chọn chức năng chuyển hướng tin
	2.	Hệ thống	Lấy danh sách TDV có thể chuyển hướng

	3.	Hệ thống	Hiển thị danh sách TDV có thể chuyển hướng
	4.	Người dùng	Chọn một TDV muốn chuyển hướng
	5.	Hệ thống	Kiểm tra tính hợp lệ
	6.	Hệ thống	Cập nhật phiên.
	7.	Hệ thống	Lưu dữ liệu thống kê
	8.	Hệ thống	Gửi thông tin giám sát
Luồng sự kiện thay thế	**STT**	**Thực hiện bởi**	**Hành động**
	5a.	Hệ thống	Thông báo TDV được chuyển hướng đang phục vụ tối đa khách hàng cho phép
	5b.	Hệ thống	Thông báo TDV không có quyền chuyển hướng tin nếu không phải là quản trị hoặc đang không phục vụ khách hàng đó
	3a.	Hệ thống	Hệ thống hiển thị danh sách rỗng và kết thúc usecase

2.3.3 Đặc tả use case “thiết lập ca làm việc”

Bảng 4 Đặc tả usecase “thiết lập ca làm việc”

Mã use case	UC017	**Tên use case**	Thiết lập ca làm việc
Tác nhân	Quản trị viên		
Tiền điều kiện			

Luồng sự kiện chính (Thành công)	**STT**	**Thực hiện bởi**	**Hành động**
	1.	Quản trị viên	Chọn chức năng thiết lập ca làm việc
	2.	Hệ thống	Lấy danh sách tổng đài viên và danh sách các ca làm việc
	3.	Quản trị viên	Nhập các thông tin tạo chuỗi chi tiết ca làm việc (mô tả dưới *)
	4.	Quản trị viên	Chọn “Thiết lập”
	5.	Hệ thống	Xử lý và lưu trữ thông tin các ca làm việc vào cơ sở dữ liệu.
	6.	Hệ thống	Thông báo tạo thành công và hiển thị các chi tiết ca làm việc vừa đăng ký thành công.
Luồng sự kiện thay thế	**STT**	**Thực hiện bởi**	**Hành động**
	5a.	Hệ thống	Thông báo tạo chi tiết ca làm việc thất bại

* Dữ liệu đầu vào khi thiết lập chuỗi ca làm việc

STT	Trường dữ liệu	Mô tả	Bắt buộc
1.	Tổng đài viên	Các tổng đài viên muốn tạo các chi tiết ca làm việc	Có
2.	Ca làm việc	Các ca làm trong ngày muốn tạo	Có
3.	Ngày bắt đầu	Ngày bắt đầu tạo chi tiết ca làm	Có

4.	Ngày kết thúc	Ngày kết thúc tạo chi tiết ca làm	Có
5.	Làm việc thứ 7	Tùy chọn có tạo chi tiết các ca làm việc vào cả ngày thứ 7	Không
6.	Làm việc chủ nhật	Tùy chọn có tạo chi tiết các ca làm việc vào cả ngày chủ nhật	Không

2.4 Yêu cầu phi chức năng

2.4.1 Tính dễ dùng

Vì người sử dụng hệ thống quản lý và điều phối tin nhắn đa kênh là những người đã có hiểu biết đôi chút về máy tính và công nghệ nên hệ thống tập trung tối giản hóa giao diện, loại bỏ các thao tác thừa không cần thiết. Giao diện chọn lọc các icon phổ biến và dễ hiểu, giúp người dùng dễ tiếp cận nhất có thể.

2.4.2 Tính dễ bảo trì

Hệ thống vẫn còn đang trong quá trình xây dựng và phát triển nên cần được thiết kế dể có thể dễ dàng sửa đổi, nâng cấp, cập nhật các tính năng sao cho phù hợp với môi trường và các yêu cầu mới. Các tính năng mới được thêm vào sẽ phải được đảm bảo không ảnh hưởng đến sự hoạt động ổn định của hệ thống trước đó.

2.4.3 Tính khả chuyển

Hệ thống được xây dựng để có thể hoạt động tốt trên nhiều môi trường, nhiều tỷ lệ màn hình khác nhau như laptop, máy tính bảng, điện thoại thông minh.

Chương 2 đã mô tả yêu cầu, giới thiệu tổng quan về chức năng, tiến hành phân tích một số usecase và đặc tả một số chức năng chính của hệ thống. Ngoài ra, các yêu cầu phi chức năng cũng là một phần quan trọng đối với hệ thống nhằm nâng cao trải nghiệm người dùng. Để đạt được các yêu cầu như vậy, Chương 3 em sẽ trình bày các công nghệ và công cụ sử dụng để xây dựng ứng dụng.

Chương 3 Công nghệ sử dụng

Trong chương 2 em đã trình bày về tổng quan các chức năng và đặc tả các chức năng chính của hệ thống quản lý và điều phối tin nhắn đa kênh. Trong chương 3 này em sẽ trình bày các công nghệ sử dụng trong quá trình phát triển hệ thống. Nội dung này bao gồm: (i) công nghệ được sử dụng phía Frontend, (ii) công nghệ sử dụng phía Backend.

3.1 Frontend

3.1.1 ReactJS

ReactJS là một thư viện Javascript mã nguồn mở được thiết kế bởi Facebook hỗ trợ xây dựng các thành phần giao diện nhanh, gọn và tiện lợi. ReactJS sinh ra với mục đích cốt lõi là không chỉ khiến cho trang web phải thật nhanh mà còn phải có khả năng mở rộng dễ dàng. Cách làm việc của React đúng theo nguyên tắc chia để trị. Các lập trình viên với React chỉ cần tập trung vào cách thành phần giao diện riêng lẻ của toàn bộ ứng dụng web, giúp cho việc phát triển dễ dàng hơn, nhanh chóng hơn.

JSX

Trong React, thay vì thường xuyên sử dụng Javascript để thiết kế bố cục trang web thì React sẽ dùng JSX. JSX cho phép nhúng các đoạn mã HTML trực tiếp vào trong mã của javascript. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javascipt. JSX sẽ được biên dịch trước khi chạy vị thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch.

Virtual DOM

ReactJS sử dụng Virtual DOM để tăng hiệu năng cho ứng dụng. Việc chỉ nút gốc mới thay đổi trạng thái và khi có thay đổi sẽ tái cấu trúc lại toàn bộ đồng kĩa với cây DOM sẽ phải thay đổi một phần dẫn đến ảnh hưởng tới tốc độ sử lý. Với Virtual DOM, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi, nó sẽ tính toán sự thay đổi giữa object và cây DOM thật, để tránh việc xuất lại những phần giao diện không cần thiết. Với những đặc điểm như vậy, React thường dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian.

Components

React được xây dựng dựa trên các component. Mỗi component trong React có các trạng thái và các thành phần giao diện khác nhau. Các trạng thái trong các component có thể được thay đổi, khi các trạng thái thay đổi, React sẽ thực hiện so sánh và cập nhật giao diện cho chính component ấy. Việc tách ra và quản lý các trạng thái theo các component này giúp mã nguồn trở nên dễ quản lý, mở rộng, sửa lỗi khiến cho React luôn là lựa chọn hàng đầu cho các dự án lớn.

3.1.2 Material UI

Material UI là một thư viện các React Component được thiết bởi Google. Material UI được thiết kế theo phong cách của Material Design, chủ yếu tập trung vào những đường nét đơn giản, sử dụng những gam màu đậm, nổi bật. Đồng thời, cách thiết kế này thường sử dụng các yếu tố đồ họa, tạo cảm giác "nổi" lên trên giao diện. Material UI rất phù hợp để xây dựng các ứng dụng của React một cách nhanh chóng thì nó được thiết kế để đáp ứng hầu hết tất cả các components thường được sử dụng như Button, Layout, Icon, ….

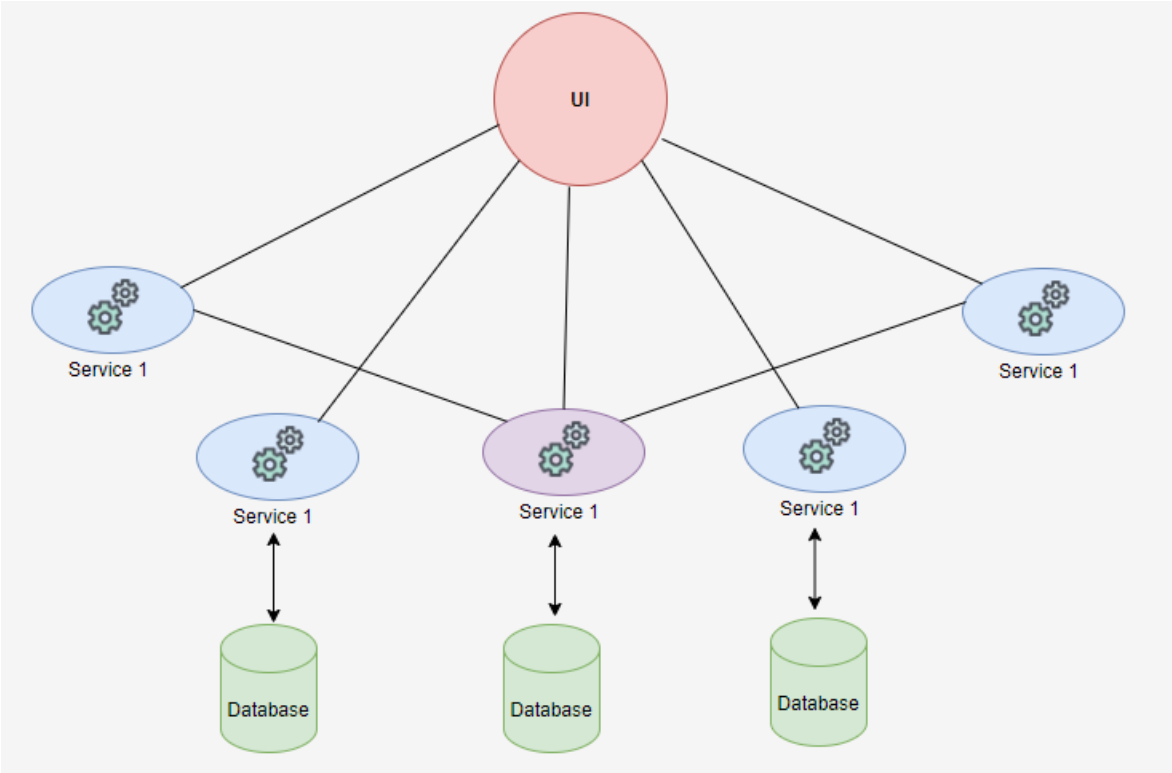
3.2 Backend

3.2.1 Microservices

Mô hình kiến trúc Microservices dựa trên ý tưởng là chia một hệ thống thành nhiều dịch vụ nhỏ kết nối với nhau, mỗi dịch vụ sẽ có một cơ sở dữ liệu riêng, thực hiện những nhóm chức năng riêng như **Hình 6**. Các dịch vụ nhỏ có thể sẽ cung cấp ra ngoài các API để giao tiếp với dịch vụ khác hoặc để phía client gọi vào. Để các dịch vụ giao tiếp được với nhau thì các nhà phát triển sẽ cần phải thống nhất phương thức giao tiếp và kiểu dữ liệu cho các dịch vụ.

Ưu điểm của Microservices là giảm thiểu sự phức tạp trong các hệ thống lớn, tách rời nghiệp vụ cho mỗi dịch vụ nhỏ giúp dễ nắm bắt nghiệp vụ, dễ bảo trì. Hơn nữa, tách riêng ra thành nhiều dịch vụ nhỏ như vậy, ta còn có thể sử dụng nhiều công nghệ, nhiều ngôn ngữ để xây dựng nên các dịch vụ tùy theo nhu cầu của dịch vụ đó. Điều này giúp cho việc xây dựng các hệ thống hay ứng dụng với kiến trúc Microservices trở nên linh hoạt và ít bị ràng buộc hơn bởi các công nghệ đã dùng trước đó.

Bên cạnh những ưu điểm đó, Microservices cũng có những nhược điểm nhất định. Khi tách ra thành các dịch vụ nhỏ cần giao tiếp với nhau qua hệ thống mạng như vậy sẽ khiến cho toàn bộ hệ thống có độ trễ cao hơn kiến trúc nguyên khối thông thường. Thêm nữa là các nhà phát triển sẽ phải giải quyết các bài toán khó hơn với tính chất của một hệ thống phân tán, cần đảm bảo được tính toàn vẹn dữ liệu cho toàn bộ hệ thống.



Hình 6 Mô hình microservices

3.2.2 NodeJS

NodeJS là một nền tảng được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript, giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng. Các công ty lớn như Amazon, Ebay, Linkedin, Microsoft, Paypal, … đều đang sử dụng NodeJS. Theo như Paypal thì sử dụng NodeJS làm giảm thời gian đáp ứng lên tới 35%. Còn về phía Linkedin, khi chuyển từ Ruby sang sử dụng NodeJS để xử lý các truy cập từ mobile, con số Server sử dụng giảm từ 30 xuống còn 3, nghĩa làm giảm gần 90%. Những điều này cho thấy ưu điểm của NodeJS thực sự là tốc độ thực thi và khả năng mở rộng. NodeJS chạy đa nền tảng phía Server sử dụng kiến trúc hướng sự kiện và cơ chế Non-blocking I/O làm cho nó nhẹ và hiệu quả.

Expressjs là một framework được xây dựng trên nền tảng của NodeJS. Nó cung cập các tính năng mạnh mẽ để phát triển web hoặc mobile. Việc xây dựng các API giờ đây trở nên dễ dàng hơn khi sử dụng Express do framework này tương đối nhỏ gọn và linh hoạt, cung cấp nhiều phương thức HTTP và các middleware.

3.2.3 MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là cơ sở dữ liệu thuộc kiểu dữ liệu phi quan hệ NoSQL. MongoDB là một cơ sở dữ liệu hướng tài liệu, các dữ liệu được lưu trữ trong các tài liệu (documents) dưới dạng JSON thay vì dạng bảng như các dạng cơ sở dữ

liệu quan hệ vì thế nên việc truy vấn trở nên rất nhanh. MongoDB sử dụng khái niệm collection thay vì table, document thay vì row trong các dạng cơ sở dữ liệu quan hệ khác. Dữ liệu trong MongoDB không có sự ràng buộc nào cả khi thêm, sửa, xóa. Chính vì thế, khi thao tác với MongoDB phải hết sức cẩn trọng vì có thể sẽ hệ thống bị lỗi do dữ liệu bị xóa nhầm hoặc không kiểm tra đầy đủ. Bên cạnh đó, vì dữ liệu được lưu dưới dạng key – value nên các documents chỉ khác nhau về value sẽ dẫn đến các key bị lặp lại, gây dư thừa dữ liệu.

3.2.4 Websocket

Websocket là một giao thức giao tiếp máy tính, cung cấp các kênh giao tiếp hai chiều thông qua một kết nối TCP. Websocket API là một công nghệ nâng cao cho phép mở một kết nối 2 chiều giữa trình duyệt của client và một server. Thông qua kết nối đã được thiết lập này, client có thể gửi một tin nhắn tới server bất kỳ lúc nào mà không cần đợi phía server phản hồi hay ngược lại, server cũng có thể gửi tin nhắn tới client bất cứ khi nào cần mà không cần một yêu cầu nào trước đó từ phía client. Điều này khác biệt hoàn toàn so với HTTP, đó là các clients sẽ chủ động gửi yêu cầu tới các server và đợi server phản hồi.

3.2.5 RabbitMQ

RabbitMQ là một message broker sử dụng giao thức AMQP (Advanced Message Queue Protocol – giao thức giao nhận tin nhắn sử dụng hàng đợi). Đây là một chương trình trung gian giữa các thành phần trong một hệ thống, có thể nhận tin nhắn từ một thành phần, lưu trữ, xử lý rồi chuyển tới các thành phần khác trong hệ thống. Điều này đặc biệt hữu ích với đặc điểm của các hệ thống phân tán khi mà các thành phần trong hệ thống cần giao tiếp với nhau thông qua mạng. Các thành phần đóng vai trò là bên gửi tin nhắn (producer) sẽ chỉ cần quan tâm tới việc gửi tin nhắn lên message broker là RabbitMQ, còn lại việc gửi tin nhắn đi đến đâu, bên nhận (consumer) sẽ nhận được như thế nào sẽ do broker xử lý hết, việc điều hướng sẽ được cấu hình ngay từ đầu. Đây là một đặc tính bất đồng bộ của RabbitMQ giúp cho việc gọi chéo giữa các dịch vụ trong kiến trúc Microservices hiệu quản hơn, hiệu năng tốt hơn.

3.2.6 Redis

Redis cũng là một trong số các hệ quản trị cơ sở dữ liệu theo kiểu phi quan hệ NoSQL. Tuy nhiên, Redis thường được sử dụng cho mục đích là lưu trữ bộ nhớ đệm để tăng tốc độ cho server xử lý, quản lý phiên, phân tích dữ liệu theo thời gian thực. Mới đây, Redis cũng ra mắt thêm tính năng Pub/Sub, có khả năng chuyển tiếp tin nhắn giống như cơ chế của RabbitMQ. Toàn bộ dữ liệu của redis được nằm trên bộ nhớ Ram, trái với các cơ sở dữ liệu khác thường được lưu dữ liệu trên ổ cứng. Bằng cách loại bỏ việc truy cập vào ổ cứng, kho dữ liệu trong bộ nhớ như Redis tránh được sự chậm trễ do thời gian tìm kiếm và có thể truy cập dữ liệu chỉ trong vài micro giây.

3.2.7 Docker

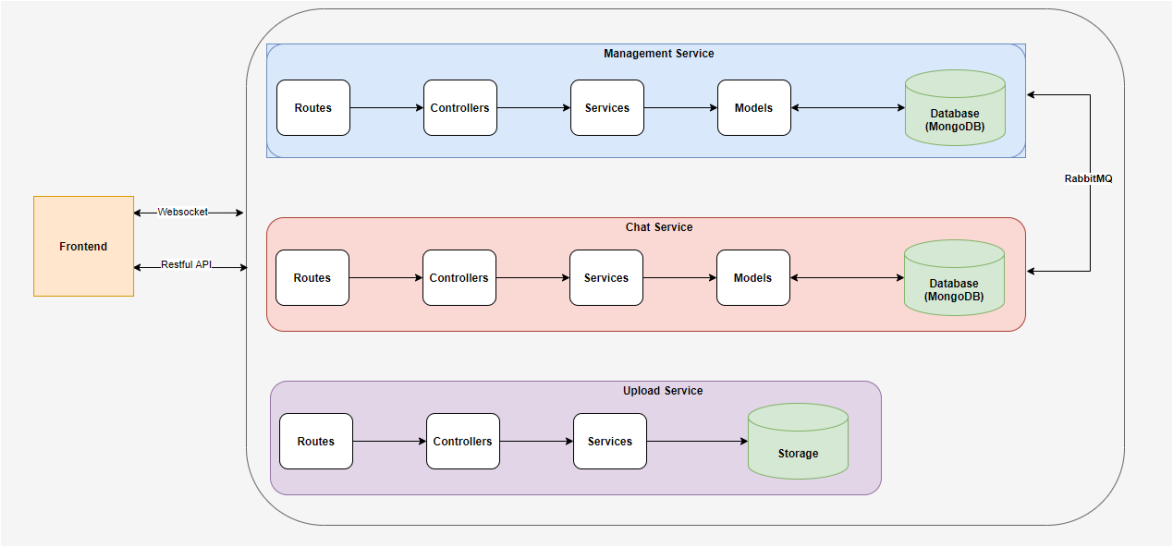
Docker là một dự án mã nguồn mở cho phép các nhà phát triển có thể đóng gói và triển khai ứng dụng trên các thiết bị, cấu hình khác nhau mà không cần trải qua các bước cấu hình và cài đặt môi trường phức tạp, dài dòng. Docker có hai thành phần chính là docker engine và docker hub. Docker engine dùng để tạo ra các docker image và chạy các docker container. Docker hub là nơi chứa các Docker image. Image là một dạng tập hợp các tệp của ứng dụng được tạo ra bởi Docker engine. Nội dung của các docker image sẽ không bị thay đổi khi di chuyển và chúng được dùng để chạy các docker container.

Trong chương 3 em đã trình bày các công nghệ chủ yếu để xây dựng hệ thống. Trong chương 4, em sẽ trình bày chi tiết về cách xây dựng, kiến trúc, cách tổ chức và lưu trữ thông tin trong hệ thống quản lý và điều phối tin đa kênh.

Chương 4 Phát triển và triển khai ứng dụng

Chương 4 em sẽ trình bày chi tiết về kiến trúc và thiết kế của hệ thống bao gồm cả phần frontend và backend.

4.1 Thiết kế kiến trúc



Hình 7 Thiết kế kiến trúc tổng quan hệ thống điều phối nhắn tin đa kênh

Hình 7 mô tả kiến trúc tổng quan của hệ thống quản lý và điều phối tin đa kênh. Như đã trình bày ở Chương 3, hệ thống được tách riêng là hai phần: Backend và Frontend.

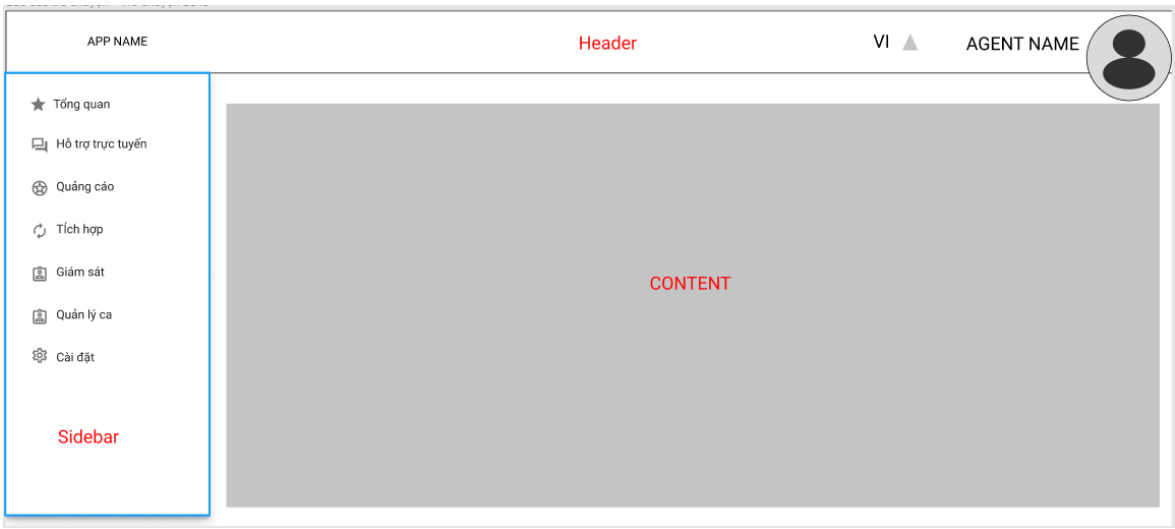
Backend được thiết kế theo mô hình kiến trúc Microservices, bao gồm các dịch vụ: dịch vụ nhắn tin (Chat service), dịch vụ quản lý (Management Service) và dịch vụ tải lên tệp tin (Upload service). Các dịch vụ đều có cơ sở dữ liệu riêng và sẽ được trình bày ở trong phần 5.1.2.1, 5.1.2.2 và 5.1.2.3. để kết nối với cơ sở dữ liệu thì em sử dụng thư viện mongoose được viết riêng cho ngôn ngữ javascript. Thư viện này được thiết kế để kết nối với cơ sở dữ liệu MongoDB thông qua Object Data Model (ODM) – một dạng biểu diễn dữ liệu dưới dạng đối tượng trong javascript thay vì sử dụng cú pháp thuần túy viết riêng cho cơ sở dữ liệu. Không những vậy, thư viện còn cung cấp những tính năng để xác nhận, kiểm tra và truy vấn dữ liệu dễ dàng. Các dịch vụ giao tiếp, trao đổi dữ liệu với nhau bằng giao thức AMQP, sử dụng công nghệ của RabbitMQ, giúp cho việc phát triển tách biệt các thành phần hệ thống dễ dàng.

Frontend sử dụng thư viện ReactJS để xây dựng nên các thành phần giao diện và gửi nhận dữ liệu đến máy chủ thông qua API và kết nối Websocket. Thiết kế chi tiết frontend em sẽ trình bày trong mục 5.1.2.4.

4.2 Thiết kế chi tiết Frontend

4.2.1 Thiết kế mockup bố cục giao diện của hệ thống

Các màn trên giao diện được thiết kế bằng ReactJS nên chúng được chia thành các component. Giao diện được thiết kế trên màn hình có độ phân giải 1920 x 1080 với bố cục là 3 phần chính là Sidebar, Header, Content được mô tả như **Hình 8**. Header và Sidebar sẽ luôn hiển được hiển thị trên màn hình. Header để hiển thị các thông tin cố định của ứng dụng như tên ứng dụng, ngôn ngữ đang hiển thị, tên người đang sử dụng và ảnh đại diện của họ. Chức năng đăng xuất cũng có thể được sử dụng bằng cách bấm vào ảnh đại diện. Sidebar là khu vực người dùng có thể tùy chọn nội dung mà mình muốn hiển thị, đây là nơi sẽ điều hướng tới các trang tương ứng với các chức năng của hệ thống.



Hình 8 Mockup bố cục của hệ thống quản lý tin nhắn

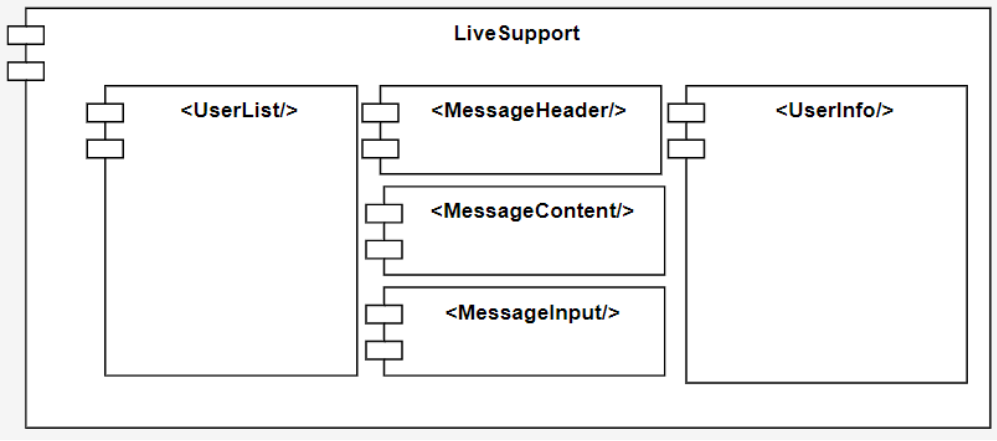
Font chữ, màu sắc, cách hiển thị thông báo, cách thiết kế bảng, bộ lọc, nút bấm, đều được thiết kế đồng nhất nhằm đạt được tính nhất quán cho toàn bộ ứng dụng.

Thuộc tính	**Cấu hình**
Màu sắc	Màu chính: #000034

	Màu khác: #f6a61f, #f16a73, #4991e2 #4a4a4a,
Font chữ	Helvetica, Aldrich
Bo góc	10px
Đổ bóng	0px 3px 3px -2px rgba(0,0,0,0.2), 0px 3px 4px 0px rgba(0,0,0,0.14), 0px 1px 8px 0px rgba(0,0,0,0.12)
Vị trí hiển trị thông báo	Góc trái phía dưới màn hình

4.2.2 Thiết kế thành phần giao diện màn hình hỗ trợ trực tuyến

Màn hình hỗ trợ trực tuyến được chia thành 6 thành phần con như **Hình 9**. UserList là thành phần hiển thị danh sách khách hàng có thêm cả các thành phần lọc nhanh các khách hàng như lọc theo kiểu khách hàng và lọc theo trạng thái của khách hàng. MessageHeader là thành phần giao diện hiển thị tên của khách hàng đang mở và các hành động có thể thực hiện đối với khách hàng đó. MessageContent là khung hiển thị nội dung cuộc trò chuyện, MessageInput là khu vực để tổng đài viên nhập tin nhắn trả lời khách hàng. Cuối cùng là UserInfo, nơi hiển thị thông tin chi tiết của khách hàng.



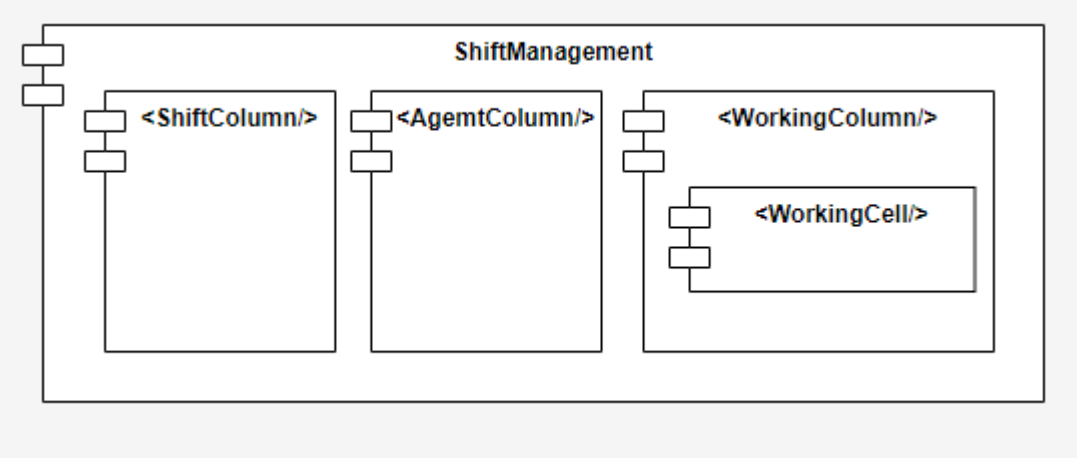
Hình 9 Thiết kế thành phần giao diện màn hình hỗ trợ trực tuyến

4.2.3 Thiết kế thành phần giao diện màn hình quản lý ca làm việc

Màn hình quản lý ca làm việc được chia thành 3 phần chính như **Hình 10**, gồm có:

- ShiftColumn: hiển thị danh sách các ca làm việc trong ngày.
- AgentColumn: hiển thị danh sách các tổng đài viên đã đăng ký trên mỗi ca làm việc.

- WorkingColumn: hiển thị dòng thời gian theo tháng, mỗi cột trong WorkingColumn thể hiện cho một ngày. Bên trong WorkingColumn là các WorkingCell có nhiệm vụ hiển thị các thông tin đăng ký trên mỗi ca làm việc của tổng đài viên.



Hình 10 Thiết kế thành phần giao diện màn hình quản lý ca làm việc

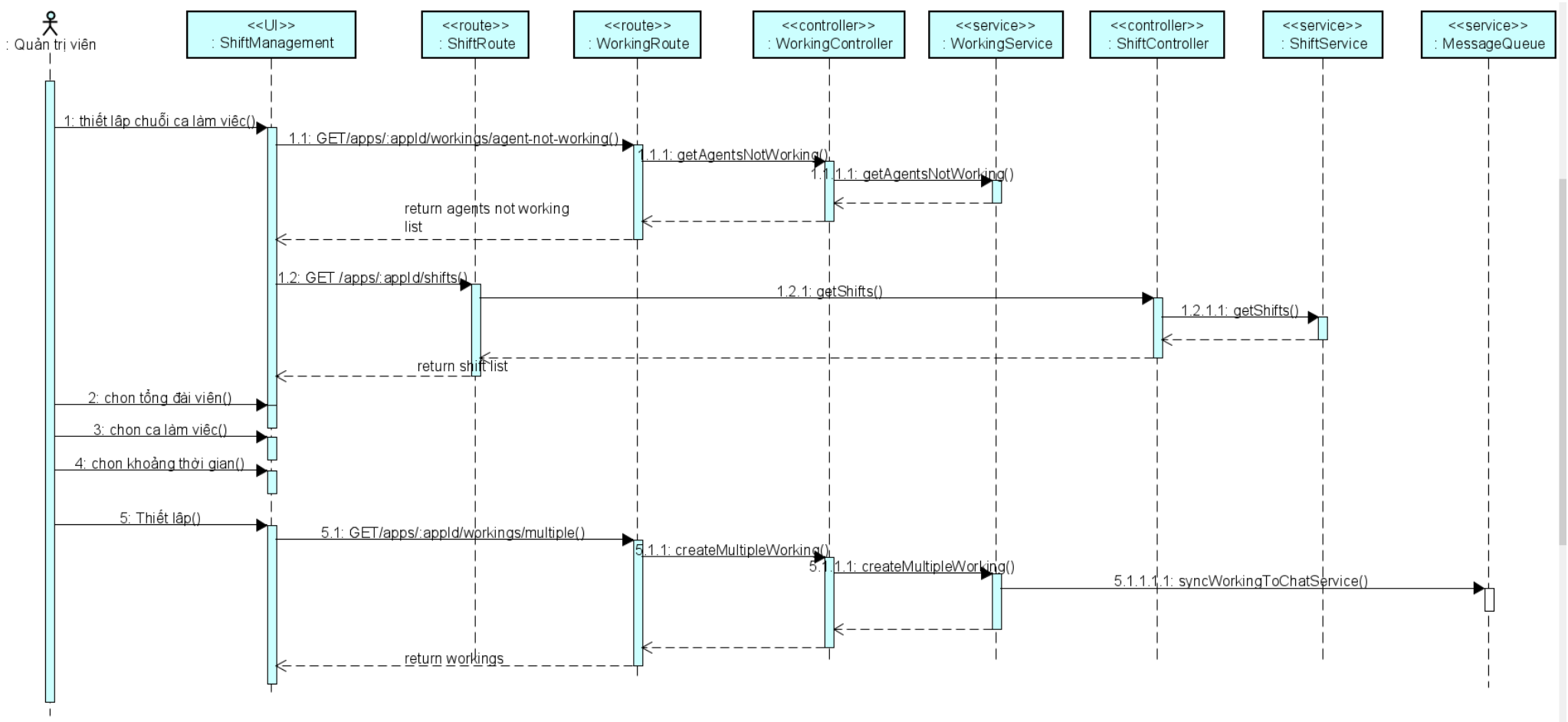
4.3 Thiết kế chi tiết Backend

4.3.1 Luồng hoạt động

Trong phần này, em sẽ trình bày biểu đồ trình tự cho các chức năng chính của hệ thống quản lý và điều phối tin nhắn đa kênh. Do phạm vi báo cáo có giới hạn nên em sẽ trình bày biểu đồ trình tự của hai chức năng tạo chuỗi chi tiết ca làm việc và chuyển hướng cuộc trò chuyện.

4.3.1.1 Luồng hoạt động tạo chuỗi chi tiết ca làm việc

Hình 11 mô tả trình tự xử lý khi người quản trị viên thiết lập chuỗi ca làm việc. Từ màn hình quản lý ca làm việc, khi chọn chức năng thiết lập ca làm việc, trình duyệt sẽ gửi yêu cầu lên server dịch vụ quản lý để lấy về danh sách các tổng đài viên chưa đăng ký ca làm việc (phương thức getAgentsNotWorking) và danh sách các ca làm việc hiện có trong ngày (phương thức getShifts). Sau đó, quản trị viên sẽ phải thực hiện các thao tác: (i) chọn các tổng đài viên, (ii) chọn các ca làm việc, (iii) chọn khoảng thời gian cài đặt. Khi chọn xong, tổng đài viên chọn tạo rồi trình duyệt sẽ gửi yêu cầu lên hệ thống qua API apps/:appId/workings/multiple của dịch vụ quản lý. Khi việc xử lý và lưu dữ liệu vào cơ sở dữ liệu thành công thì cũng là lúc hệ thống gọi phương thức asyncWorkingToChatService() để đồng bộ dữ liệu các ca làm việc sang cơ sở dữ liệu của dịch vụ nhắn tin thông qua RabbitMQ. Chi tiết về phần hiển thị em sẽ trình bày ở phần 5.2.3.

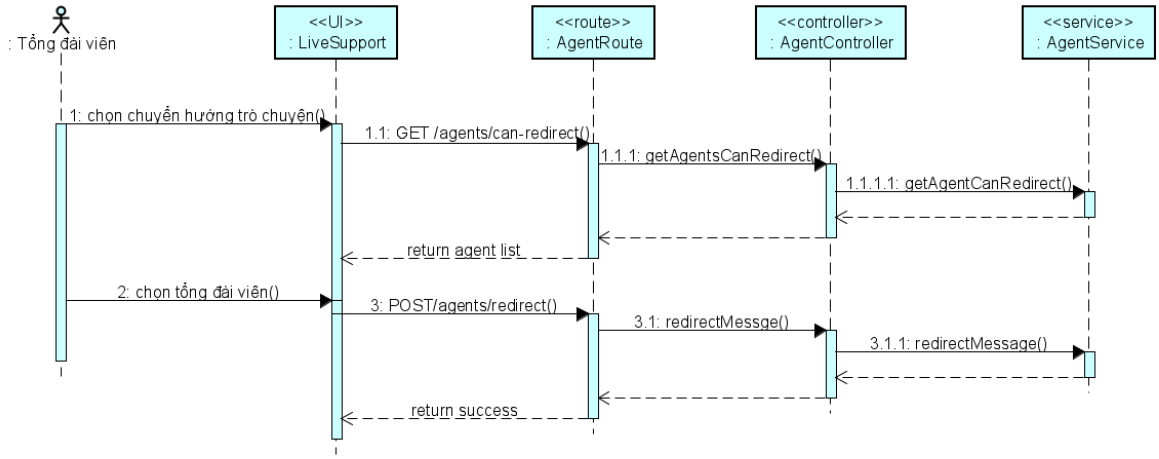


Hình 11 Biểu đồ trình tự tạo chuỗi chi tiết ca làm việc

4.3.1.2 Luồng hoạt động chuyển hướng cuộc trò chuyện

Hình 12 mô tả luồng hoạt động của chương trình một tổng đài viên thực hiện chuyển hướng cuộc trò chuyện tới một tổng đài viên khác. Từ một khách hàng đang được tổng đài viên phục vụ, tổng đài viên lựa chọn chức năng chuyển hướng cuộc trò chuyện, trình duyệt sẽ gửi yêu cầu tới dịch vụ nhắn tin theo API GET agents/can-redirect để lấy ra danh sách các tổng đài viên có thể thực hiện chuyển tiếp. Khi yêu cầu được đưa vào đến AgentService, tại đây sẽ kiểm tra và lấy ra tất cả những tổng đài viên thuộc quản lý của ứng dụng đã đăng kí ca làm việc, đang trực tuyến và đang không phục vụ số lượng người dùng ở mức tối đa.

Khi tổng đài viên bấm chọn một tổng đài viên từ danh sách, trình duyệt lại gửi yêu cầu tới dịch vụ nhắn tin thông qua API POST agents/redirect để yêu cầu chuyển tiếp. Chi tiết về giao diện em sẽ trình bày ở phần 4.4.3.2.



Hình 12 Biểu đồ trình tự chuyển hướng cuộc trò chuyện

4.3.2 Thiết kế các dịch vụ

Trong kiến trúc backend của dịch vụ nhắn tin cung cấp nhiều dịch vụ xử lý logic nghiệp vụ trong hệ thống. Do khối lượng báo cáo có hạn, em xin trình bày một vài dịch vụ quan trọng nhất mà em đã làm đó là: Worker Service, Message Queue Service, Monitor Service.

4.3.2.1 Worker Service

Worker service được sử dụng cho tính năng điều phối tin tự động. Dịch vụ này có nhiệm vụ tìm kiếm tổng đài viên rảnh rỗi nhất rồi thực hiện chỉ định các tin nhắn đang chờ tới tổng đài viên đó. Ngoài ra, Worker service còn có chức năng đếm thời gian hết hạn trong một khoảng thời gian nhất định, nếu tổng đài viên không trả lời sẽ được Worker service chuyển tiếp tin tới một tổng đài viên khác.

Bảng 5 Các phương thức của Worker Service

STT	Tên hàm	Tham số	Mục đích
1	getAvailableAgents	appId: mã ứng dụng workingAgents: danh sách các tổng đài viên đã đăng ký ca làm việc delayingAgentId: mã tổng đài viên đang quá hạn trả lời (có trong trường hợp chuyển tiếp tin tự động) onilneAgents: danh sách tổng đài viên đang trực tuyến maxServingUser: số lượng khách hàng tối đa tổng đài viên có thể phục vụ trong ứng dụng	Lấy ra danh sách các tổng đài viên đủ điều kiện làm việc như: đã đăng ký ca làm, đang trực tuyến và đang không phục vụ quá số lượng khách hàng.
	getLeastWorkingAgent	appId: mã ứng dụng maxServingUser: số lượng khách hàng tối đa tổng đài viên có thể phục vụ trong ứng dụng delayingAgentId: mã tổng đài viên đang quá hạn trả lời	Lấy ra tổng đài viên đang làm việc ít nhất. Phương thức trả về id của tổng đài viên.
	assignChatToAgent	appId: mã ứng dụng userId: mã khách hàng cần chỉ định phục vụ	Thực hiện chỉ định cho một tổng đài viên khi có khách hàng nhắn tin đến.

		fromAgentId: mã tổng đài viên đang quá hạn trả lời.	
	autoPushToWaitingQueue	appId: mã ứng dụng userId: mã khách hàng fromAgentId: mã tổng đài viên đang quá hạn trả lời	Phương thức được gọi ra khi tổng đài khi quá hạn trả lời tin nhắn của khách hàng, hệ thống tự động đẩy lại vào hàng chờ để chuyển cho tổng đài viên khách

4.3.2.2 Message Queue Service

Message Queue Service có vai trò gửi và nhận thông điệp từ hệ thống ChatBot và các dịch vụ khác chẳng hạn như dịch vụ quản lý thông qua hàng đợi tin nhắn từ RabbitMQ. Ngoài ra, dịch vụ này còn có chức năng đẩy các tin nhắn khách hàng đang chờ vào hàng đợi tin nhắn đang chờ theo định danh của từng ứng dụng, đồng thời cũng là nơi tạo ra các consumer (tiêu thụ) lắng nghe các tin nhắn được gửi từ RabbitMQ Broker rồi chuyển về cho Worker Service xử lý.

Bảng 6 Các phương thức của Message Queue Service

STT	Tên hàm	Tham số	Mục đích
	handleResponse	Data: thông tin tin nhắn gửi đi. isAgentChat: kiểm tra có phải tổng đài viên gửi tin nhắn hay không	Xử lý thông điệp phản hồi từ chatbot hoặc tin nhắn tổng đài viên gửi tới.
	createWaitingChatConsumer	appId: mã ứng dụng	Tạo ra consumer theo mỗi ứng dụng để nhận tin nhắn đang chờ về xử lý (chế độ điều phối tự động)

	cancelConsumerAndQueue	appId: mã ứng dụng	Hủy consumer và hàng đợi tương ứng của ứng dụng khi ứng dụng bị xóa

4.3.2.3 Monitor Service

Monitor Service sẽ hoạt động bất cứ khi nào có một sự kiện xảy ra làm thay đổi trạng thái của khách hàng hoặc tổng đài viên như: tạo mới một người dùng, chatbot không trả lời được nên chuyển tiếp về cho tổng đài viên, tổng đài viên chuyển tiếp cuộc trò chuyện, tổng đài viên kết thúc cuộc trò chuyện, tổng đài viên online hoặc offline … Mỗi khi có một sự kiện xảy ra, Monitor service tính toán và lấy lại các thông tin mới nhất của tổng đài viên và khách hàng vừa được cập nhật rồi gửi thông tin qua kết nối websocket cho các quản trị viên.

Bảng 7 Các phương thức của Monitor Service

STT	Tên hàm	Tham số	Mục đích
1	handleMonitorInfo	appId: mã ứng dụng userId: mã khách hàng agentIds: mảng các id của tổng đài viên thực hiện thay đổi changeUserStatus: kiểm tra xem sự kiện có làm thay đổi trạng thái của khách hàng không	Xử lý thông tin khách hàng và tổng đài viên khi có sự kiện xảy ra làm thay đổi trạng thái của khách hàng hoặc trạng thái của tổng đài viên.

4.3.2.4 Thiết kế API

Trong phần 4.1 em đã trình bày rằng các dịch vụ giao tiếp với nhau thông qua API hoặc tin nhắn hàng đợi của RabbitMQ. Hệ thống cung cấp các API và hàng đợi được em mô tả cụ thể trong **Bảng 8** và **Bảng 9**.

Bảng 8 Danh sách api

STT	Dịch vụ	Địa chỉ	Phương thức	Mục đích
1	Dịch vụ quản lý	/api/v1/apps	POST	Tạo ứng dụng
2		/api/v1/apps/:appId	PUT	Cập nhật ứng dụng
3		/api/v1/apps/:appId	GET	Lấy thông tin ứng dụng
4		/api/v1/apps/:appId	DELETE	Xóa ứng dụng
5		/api/v1/ apps/:appId/shifts	POST	Tạo ca làm việc cho ứng dụng
6		/api/v1/apps/:appId/shifts/:shiftId	PUT	Cập nhật ca làm việc
7		/api/v1/apps/:appId/shifts/:shift	GET	Lấy tất cả danh sách ca làm việc của ứng dụng
8		/api/v1/apps/:appId/shifts/:shift	DELETE	Xóa ca làm việc
9		/api/v1/apps/:appId/workings	POST	Tạo chi tiết một ca làm việc
10		/api/v1/apps/:appId/workings	GET	Lấy danh sách chi tiết các ca làm việc theo ứng dụng
11		api/v1/apps/:appId/workings/:workingId	DELETE	Hủy một chi tiết ca làm việc

12		api/v1/apps/:appId/workings/multiple	POST	Tạo chuỗi chi tiết các ca làm việc
13		/api/v1/apps/:appId/workings/agent-not-working	GET	Lấy ra danh sách các tổng đài viên chưa đăng ký ca làm việc trên ứng dụng
14	Dịch vụ nhắn tin	/api/v1/pick-user	POST	Nhận tin thủ công
15		/api/v1/end-chat	POST	Kết thúc cuộc trò chuyện
16		/api/v1/redirect-message	POST	Chuyển hướng cuộc trò chuyện
17		/api/v1/can-redirect	GET	Lấy ra danh sách các tổng đài viên có thể chuyển hướng tin nhắn
18		/api/v1/users	GET	Lấy danh sách khác hàng
19		/api/v1/users/:userId	GET	Lấy danh sách khách hàng theo Id
20		/api/v1/users/skip-user	GET	Lấy ra danh sách khách hàng kể từ một userId, sắp xếp theo thời gian tin nhắn cuối đến hệ thống
21		/api/v1/users/:userId/reply-mode	PUT	Thay đổi chế độ trả lời cho user

Bảng 9 Danh sách hàng đợi tin nhắn

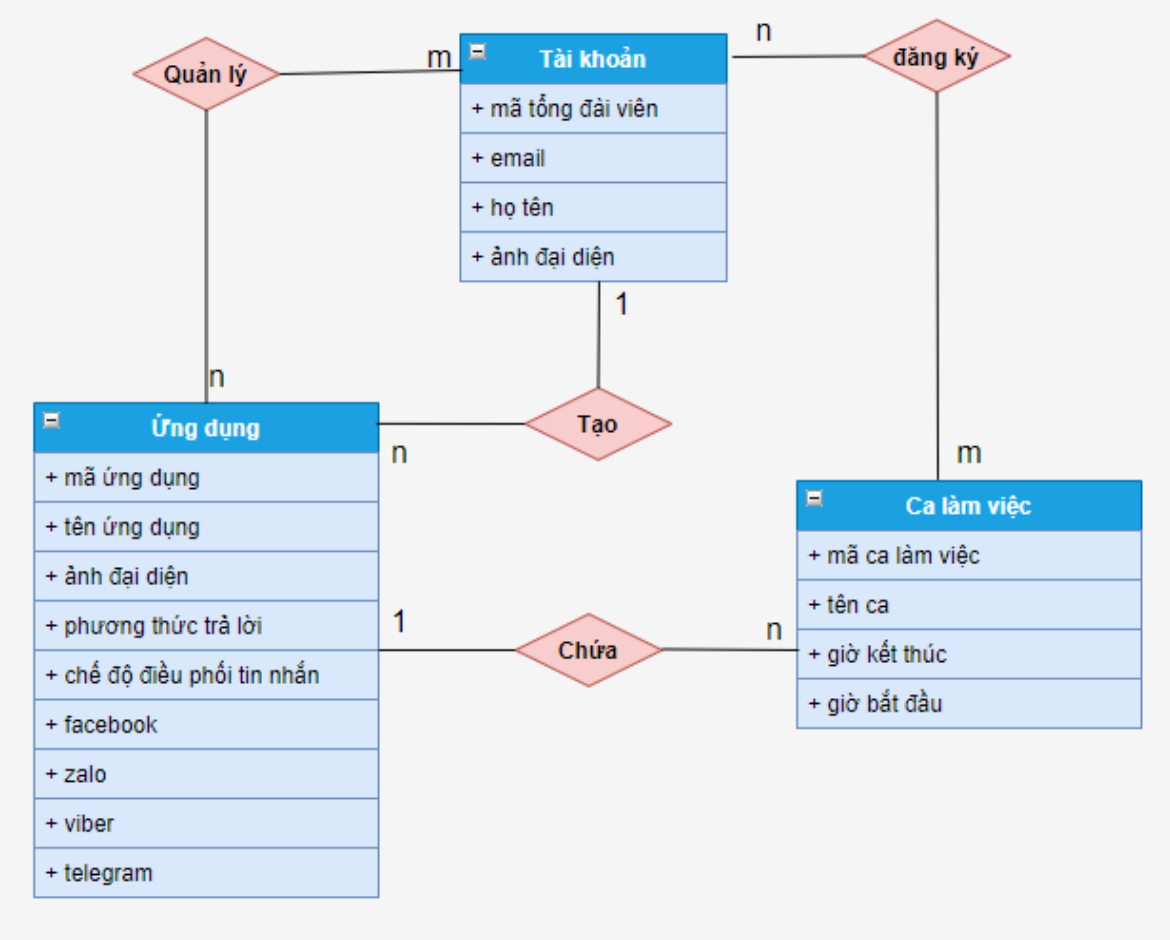
STT	Hàng đợi	Mục đích
1	CHAT_QUEUE	Dịch vụ quản lý gửi yêu cầu và trả kết quả cho dịch vụ nhắn tin
2	RESPONSE_CHAT_QUEUE	Dịch vụ nhắn tin gửi yêu cầu và kết quả cho dịch vụ quản lý
3	WAITING_CHAT_QUEUE_appID	Dịch vụ nhắn tin gửi lên các tin nhắn đang trong trạng thái chờ tổng đài viên phục vụ trong trường hợp ứng dụng để chế độ routing tự động.

4.3.3 Thiết kế cơ sở dữ liệu

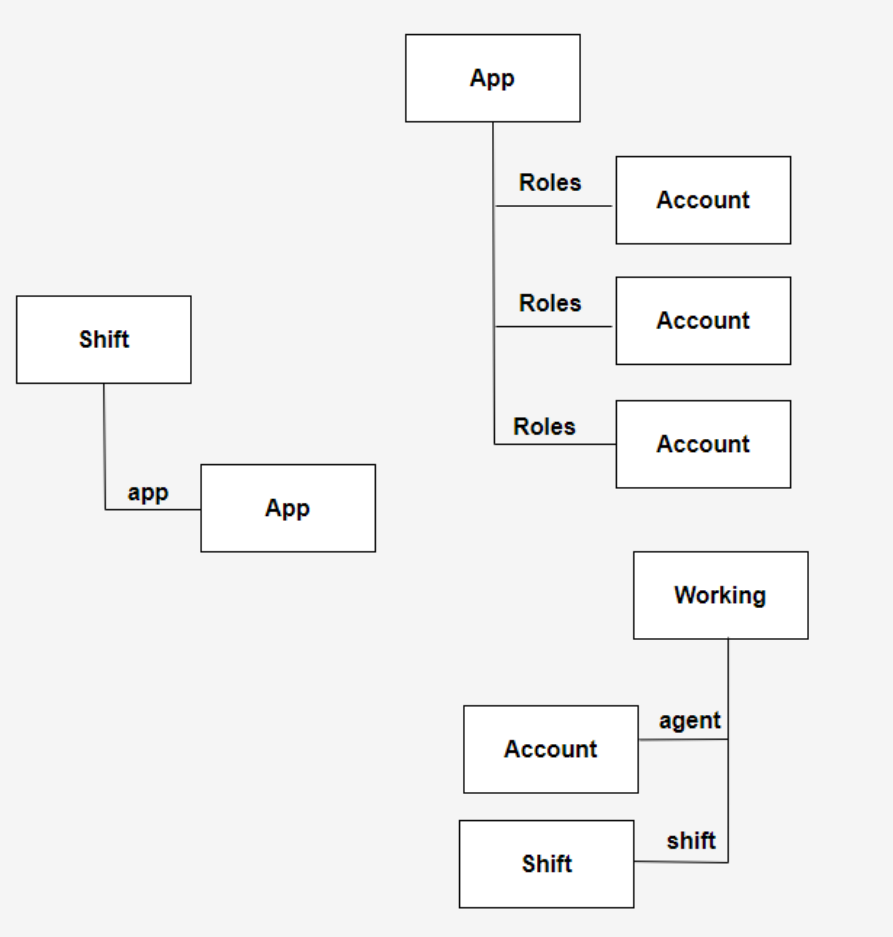
Như đã trình bày trong phần 4.1 về kiến trúc, các dịch vụ sẽ có cơ sở dữ liệu riêng. Trong phần này em xin được phép trình bày 2 cơ sở dữ liệu của dịch vụ nhắn tin và dịch vụ quản lý. Riêng dịch vụ tải lên tệp tin là kho chứa các tệp tin đa phương tiện hiện chưa cần có cơ sở dữ liệu nên em sẽ không trình bày.

4.3.3.1 Cơ sở dữ liệu dịch vụ quản lý

Hình 13 là biểu đồ thực thể liên kết của dịch vụ quản lý thể hiện mối liên hệ giữa Ứng dụng, Tài khoản và Ca làm việc. Theo đó, một tài khoản có thể tạo nhiều ứng dụng, một ứng dụng có thể có quản lý nhiều tài khoản và có nhiều ca làm việc. Đồng thời, một tài khoản cũng có thể đăng ký nhiều ca làm việc khác nhau và mỗi ca làm việc có thể được đăng ký bởi nhiều tài khoản.



Hình 13 Biểu đồ thực thể liên kết dịch vụ quản lý



Hình 14 Thiết kế tổng quan cơ sở dữ liệu quản lý

Từ biểu đồ thực thể liên kết trên, em đã thiết kế cơ sở dữ liệu phi quan hệ NoSQL sử dụng MongoDB như **Hình 14**. Cơ sở dữ liệu gồm có bốn collections chính là App, Account, Shift và Working. App Collection sẽ tham chiếu đến _id của Account Collection. Shift Collection sẽ tham chiếu tới _id của App có ý nghĩa là định nghĩ ca làm việc trong ngày của ứng dụng. Working Collection có ý nghĩa là thể hiện chi tiết ca làm việc của tổng đài viên đăng ký làm việc vào một ca làm việc trong một ngày cố định. Working Collection tham chiếu tới _id của Account và Shift Collection. Chi tiết của các trường trong các collection em xin được trình bày cụ thể ở bảng

Bảng 10 Thiết kế chi tiết cơ sở dữ liệu quản lý

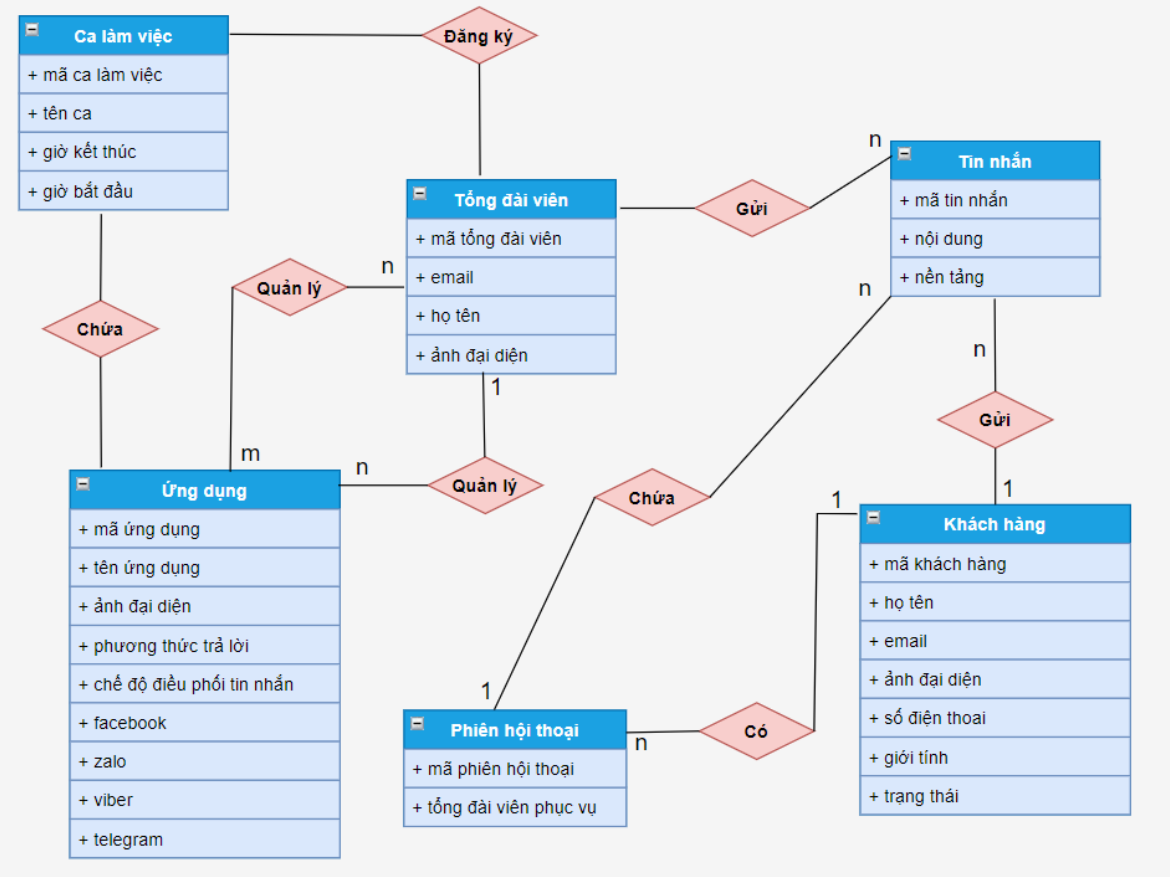
Collection	**Tên trường**	**Kiểu dữ liệu**	**Ý nghĩa**
App	_id	String	Id của ứng dụng

	Name	String	Tên ứng dụng
	Avatar	String	Đường dẫn đến ảnh của ứng dụng
	Roles	Array	Các quyền hiện có trong ứng dụng
	replyMode	String	Phương thức trả lời hiện tại của ứng dụng
	routingMode	String	Chế độ điều phối tin
	maxServingUser	Number	Số lượng khách hàng tối đa mà tổng đài viên có thể phục vụ cùng lúc
	waitingTimeout	Number	Thời gian chờ tối đa khi tổng đài viên không trả lời sẽ tự động chuyển hướng cho tổng đài viên khác.
	botToken	String	Token của chatbot bên hệ thống smartdialog
	Facebook	Object	Thông tin tích hợp Facebook Messager
	Zalo	Object	Thông tin tích hợp Zalo
	Viber	Object	Thông tin tích hợp Viver
	Telegram	Object	Thông tin tích hợp Telegram
Shift	_id	Object	ID của ca làm việc
	Name	String	Tên ca làm việc
	startTime	String	Giờ bắt đầu ca làm việc

	endTime	String	Giờ kết thúc ca làm việc
	App	String	ID của ứng dung
Working	_id	ObjectId	ID của chi tiết ca làm việc
	Agent	ObjectId	Tham chiếu ID của Agent đăng ký
	Shift	ObejctId	Ca làm việc đăng ký
	Day	ISODate	Ngày đăng ký làm
Account	_id	ObjectId	ID của người dùng
	Name	String	Tên người dùng
	Avatar	String	Đường dẫn tới hình ảnh đại diện
	Email	String	Email của người dùng

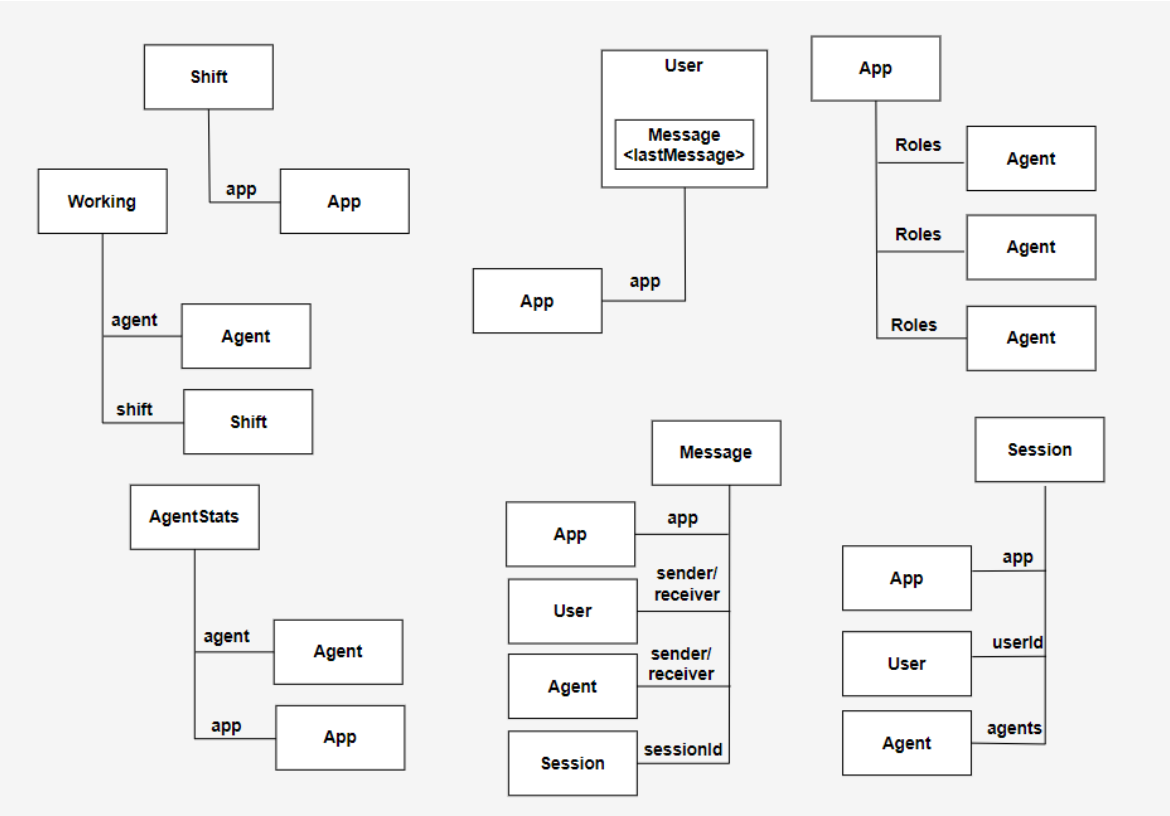
4.3.3.2 Cơ sở dữ liệu dịch vụ nhắn tin

Hình 15 là biểu đồ thực thể liên kết của dịch vụ nhắn tin mô tả mối quan hệ giữa ứng dụng, ca làm việc, tổng đài viên, tin nhắn, khách hàng và phiên hội thoại. Mối quan hệ giữa tổng đài viên, ca làm việc và ứng dụng cũng là mối quan hệ ở dịch vụ quản lý được em đồng bộ sang dịch vụ nhắn tin. Ở dịch vụ nhắn tin, tổng đài viên và khách hàng có thể gửi nhiều tin nhắn, các tin nhắn sẽ được chứa bởi phiên hội thoại. Các khách hàng khi nhắn tin đến thì sẽ tạo ra các phiên hội thoại.



Hình 15 Biểu đồ thực thể liên kết dịch vụ nhắn tin

Từ biểu đồ thực thể liên kết, em đã thiết kế cơ sở dữ liệu phi quan hệ NoSQL MongoDB bao gồm tám collection đó là: App, Agent, Shift, Working, User, Session, Message, AgentStats. App, Agent, Shift, Working là bốn collection được đồng bộ từ dịch vụ quản lý nên em sẽ không trình bày lại ở đây. User Collection sẽ có một trường là lastMessage chứa toàn bộ thông tin của một bản ghi thuộc Message Collection thể hiện đây là tin nhắn mới nhất. Message Collection chứa _id của ứng dụng, các trường sender hoặc receiver sẽ tham chiếu tới các collection là Agent hoặc User còn tùy vào người gửi và người nhận, Message Collection cũng phải có id của session tham chiếu tới Session Collection. Session Collection tham chiếu tới App, User và cả Agent. AgentStats Collection được em tạo ra để lưu trữ các dữ liệu thống kê hiệu suất làm việc của tổng đài viên đối với mỗi ứng dụng theo từng ngày vì thế có id tham chiếu tới App collection và Agent collection. Em sẽ mô tả chi tiết các trường trong mỗi collection ở **Bảng 11**.



Hình 16 Thiết kế tổng quan cơ sở dữ liệu nhắn tin

Bảng 11 Thiết kế chi tiết cơ sở dữ liệu dịch vụ nhắn tin

Collection	Tên trường	Kiểu dữ liệu	Ý nghĩa
User	App	String	ID của ứng dụng quản lý chứa khách hàng
	Type	String	Loại khách hàng (Anonymous, Facebook, Zalo, Viber, Telegram)
	Profile	Object	Thông tin chung của khách hàng (fullname, dateOfBirth, gender, avatar, gender)
	Status	Object	Value: trạng thái của người dùng

			startedAt: thời gian bắt đầu trạng thái này của khách hàng
	Facebook	Object	Thông tin của khách hàng trên Facebook
	Zalo	Object	Thông tin của khách hàng trên Zalo
	Viber	Object	Thông tin của khách hàng trên Viber
	Telegram	Object	Thông tin của khách hàng trên Telegram
	lastMessaage	Object	Tin nhắn mới nhất của khách hàng
	firstWaitingMsgTime	Number	Thời gian tin nhắn đầu tiên người dùng gửi đến đang chờ được phản hồi. dùng để tính hiệu suất phản hồi của tổng đài viên
	replyMode	String	Chế độ trả lời đối với khách hàng (chatbot hoặc tổng đài viên)
Session	_id	String	ID của phiên hội thoại
	userId	ObjectId	ID của khách hàng tương ứng với phiên hội thoại
	App	String	ID của ứng dụng chứa phiên hội thoại
	Agents	Object	-served: là mảng chứa các id của tổng đài viên đã từng phục vụ khách hàng trong phiên này.

			- serving: là một ObjectId collection Agent biểu thị tổng đài viên đang phục vụ.
Message	_id	String	ID của tin nhắn
	Content	Object	Nội dung tin nhắn
	sessionId	String	ID của phiên hội thoại chứa tin nhắn
	App	String	ID của ứng dụng chứa tin nhắn
	Sender	ObjectId	ID của một người gửi (User, Agent, Bot)
	Receiver	ObjectId	ID của một người nhật (có thể là Agent, User)
	Platform	String	Nền tảng mà tin nhắn này đã được nhận vào trước khi được đưa nào cơ sở dữ liệu
	isFirst	Boolean	Xác định là tin nhắn đầu tiên của phiên hội thoại
	Status	String	Trạng thái của tin nhắn
AgentStats	App	String	Id của ứng dụng chứa thông tin thống kê của tổng đài viên
	Agent	String	ID của tổng đài viên
	avgReplyTime	Number	Thời gian phải hồi trung bình
	totalFirstReply	Number	Tổng số lần tin nhắn đầu tiên mà khách hàng gửi tới sau khi tổng đài viên phản hồi.

	totalSession	Number	Tổng số phiên mà tổng đài viên đã phục vụ
	totalRedirect	Number	Tổng số chat mà tổng đài viên đã chuyển hướng tin nhắn
	totalAsssigned	Number	Tổng số chat mà tổng đài viên được chỉ định phục vụ hoặc tổng đài viên chọn phục vụ.
	totalMissed	Number	Tổng số chat mà tổng đài viên không trả lời kịp để quá hạn và hệ thống phải tự động chuyển sang cho tổng đài viên khác.

4.4 Xây dựng ứng dụng

4.4.1 Thư viện và công cụ sử dụng

Trong quá trình phát triển hệ thống, em đã sử dụng một số thư viện và công cụ hỗ trợ được em liệt kê trong **Bảng 12**.

Sinh viên liệt kê các công cụ, ngôn ngữ lập trình, API, thư viện, IDE, công cụ kiểm thử, v.v. mà mình sử dụng để phát triển ứng dụng. Mỗi công cụ phải được chỉ rõ phiên bản sử dụng. SV nên kẻ bảng mô tả tương tự như Bảng 12. Nếu có nhiều nội dung trình bày, sinh viên cần xoay ngang bảng.

Bảng 12 Danh sách thư viện và công cụ sử dụng

Công cụ	**Mục đích**	**Địa chỉ URL**
Visual Studio Code	IDE lập trình	https://code.visualstudio.com/
Robo 3T	Công cụ quản trị cơ sở dữ liệu	https://robomongo.org
Postman	Công cụ gọi api	https://postman.com

Javascript	Ngôn ngữ lập trình cho toàn bộ hệ thống	https://developer.mozilla.org/en-US/docs/Web/JavaScript
Express	Framework lập trình phía backend	https://expressjs.com
ReactJS	Thư viện lập trình giao diện phía frontend	https://ReactJS.org
MongoDB	Hệ quản trị cơ sở dữ liệu	https://mongodb.com
Material UI	Thư viện React Component	https://material-ui.com/
Docker	Triển khai hệ thống	https://docker.com

Ngoài ra em còn sử dụng một số thư viện bên thứ 3 việc phát triển hệ thống được dễ dàng, tập trung tối đa vào phần nghiệp vụ. Các thư viện được em liệt kê ở

Bảng 13 Danh sách thư viện sử dụng trong dịch vụ nhắn tin

Thư viện	**Phiên bản**	**Mục đích**
mongoose	5.6.9	Thư viện lập trình kết nối và thao tác truy vấn cơ sở dữ liệu với MongoDB
amqplib	0.5.5	Thư viện tạo client kết nối với RabbitMQ
redis	2.8.0	Thư viện hỗ trợ kết nối với redis server
ws	7.1.2	Thư viện hỗ trợ tạo kết nối websocket
Axios	0.19.0	Thư viện gửi HTTP request
Moment	2.24.0	Thư viện hỗ trợ xử lý thời gian

4.4.2 Kết quả đạt được

Sau khi triển khai ứng dụng, hệ thống có các chức năng đã được phát triển khá hoàn thiện như điều phối tin từ nhiều kênh một cách tự động, nhận tin nhắn thủ công, chuyển hướng tin nhắn cho các tổng đài viên khác, có chức năng giám sát tin nhắn, tổng đài viên cho các quản trị viên, quản lý ca làm việc, thống kê hiệu suất của tổng đài viên.

Dưới đây là bảng thống kê thông tin về ứng dụng.

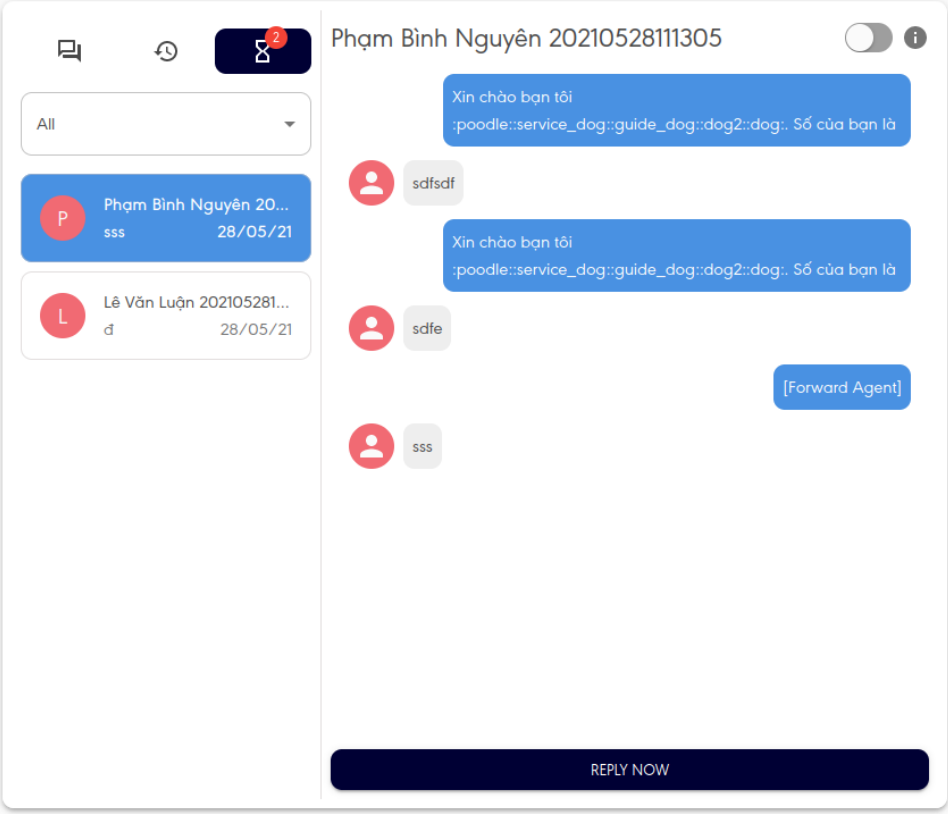
Thông tin	Thống kê
Số file trong mã nguồn dịch vụ nhắn tin	
Số dòng code	
Số bảng sử dụng	

4.4.3 Minh hoạ các chức năng chính

Trong phần này em sẽ minh họa một số chức năng chính của hệ thống quản lý và điều phối tin nhắn đa kênh. Cụ thể là giao diện và các chức năng (i) chọn tin thủ công, (ii) chuyển tiếp cuộc trò chuyện, (iii) giám sát chat, (iv) giám sát tổng đài viên, (v) thống kê hiệu suất tổng đài viên. Giao diện và chức năng của phần hỗ trợ trực tuyến điều phối tin nhắn và quản lý ca làm việc sẽ được em trình bày trong phần 5.2.3 và 5.3.3.

4.4.3.1 Chọn tin thủ công

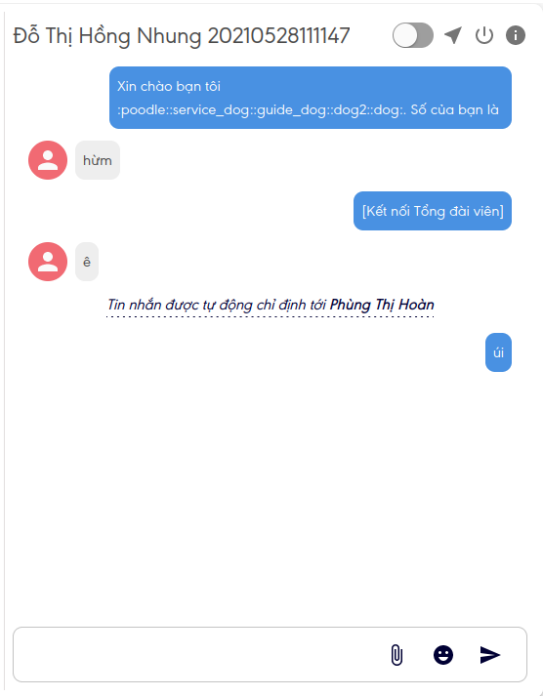
Hình 17 là màn hình khi ứng dụng đang ở chế độ điều phối tin thủ công và người đăng nhập có vai trò là tổng đài viên. Trên thanh điều hướng của danh sách các khách hàng sẽ có 3 tab đó là: (i) khách hàng đang phục vụ, (ii) lịch sử cách khách hàng mà tổng đài viên đã phục vụ, (iii) danh sách đang chờ. Riêng tab danh sách đang chờ thì khi ứng dụng ở chế độ điều phối tự động thì trên giao diện sẽ không tồn tại. Ở dưới dùng phần nội dung tin nhắn có một nút “Trả lời ngay” để cho phép các tổng đài viên thực hiện chức năng chọn một khách hàng về phía mình để phục vụ.



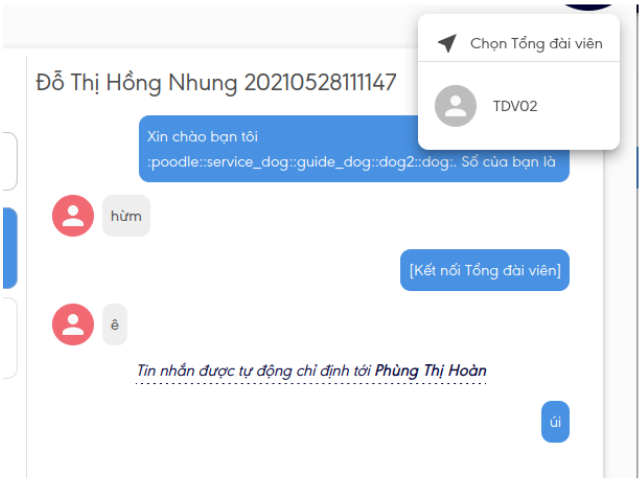
Hình 17 giao diện chọn tin thủ công

4.4.3.2 Chuyển tiếp cuộc trò chuyện

Hình 18 là một phần màn hình hỗ trợ trực tuyến khu vực nội dung nhắn tin khi khách hàng đang được tổng đài viên phục vụ. Lúc này, ở góc trên bên phải khung hình sẽ có các hành động mà tổng đài viên có thể thực hiện theo thứ tự: chuyển đổi trạng thái trả lời, chuyển tiếp cuộc trò chuyện, kết thúc cuộc trò chuyện, thông tin chi tiết khách hàng. **Hình 19** là giao diện màn hình khi tổng đài viên nhấn vào icon chuyển tiếp cuộc trò chuyện. Khi nhấn vào sẽ hiển thị ra danh sách tổng đài viên có thể chuyển tiếp tại thời điểm bấm nút.



Hình 18 màn hình khách hàng đang được tổng đài viên phục vụ

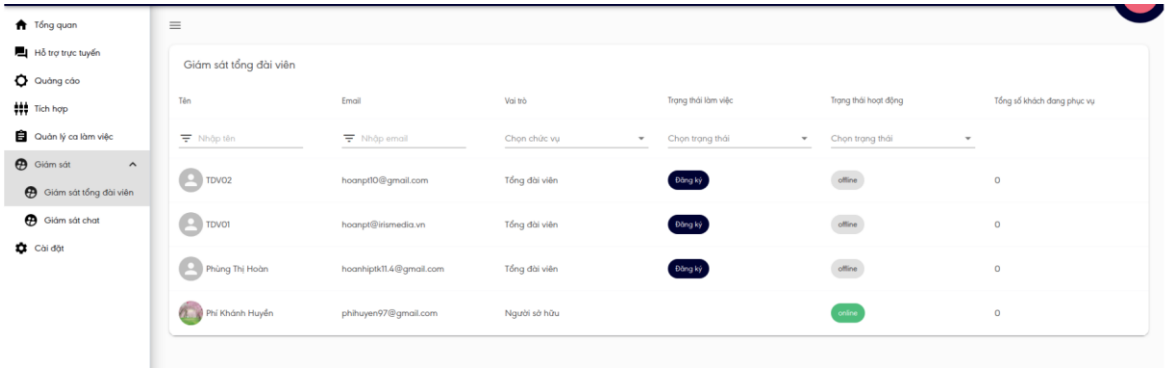


Hình 19 màn hình khi chọn chức năng chuyển tiếp cuộc trò chuyện

4.4.3.3 Giám sát tổng đài viên

Hình 20 là giao diện giám sát tổng đài viên theo thời gian thực. Tại màn hình này, quản trị viên có thể quan sát được sự thay đổi ngay lập tức nếu tổng đài viên online hoặc offline. Đồng thời, số lượng khách hàng mà mỗi tổng đài viên cũng sẽ thay đổi theo thời gian thực đúng như số lượng khách hàng mà tổng đài viên đó đang phục vụ. Điều này dễ giúp cho

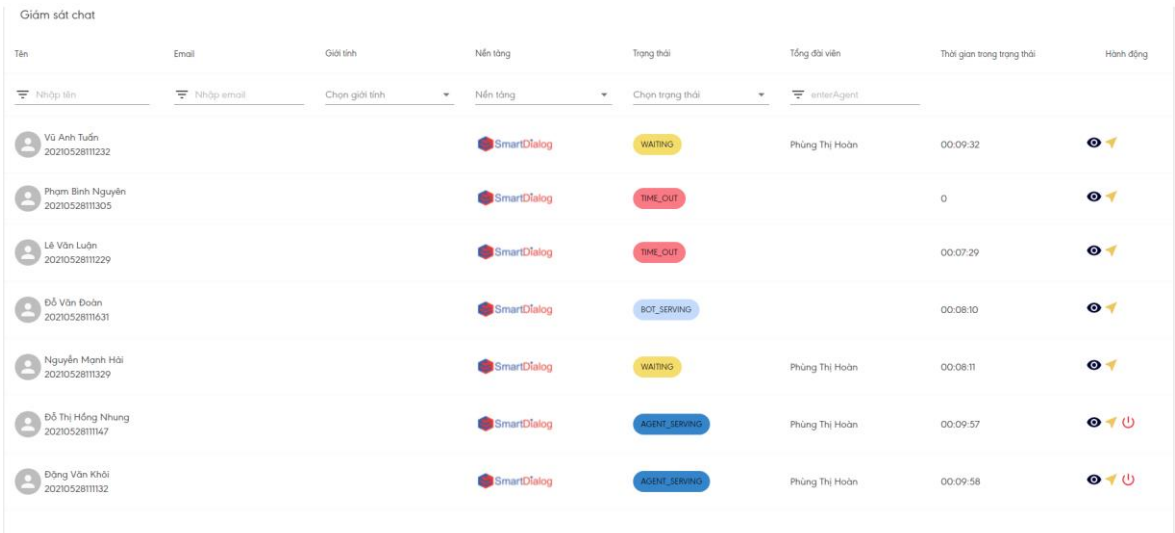
người quản trị viên có cái nhìn tổng quan về trạng thái làm việc của các tổng đài viên trong thời điểm hiện tại.



Hình 20 giao diện giám sát tổng đài viên

4.4.3.4 Giám sát chat

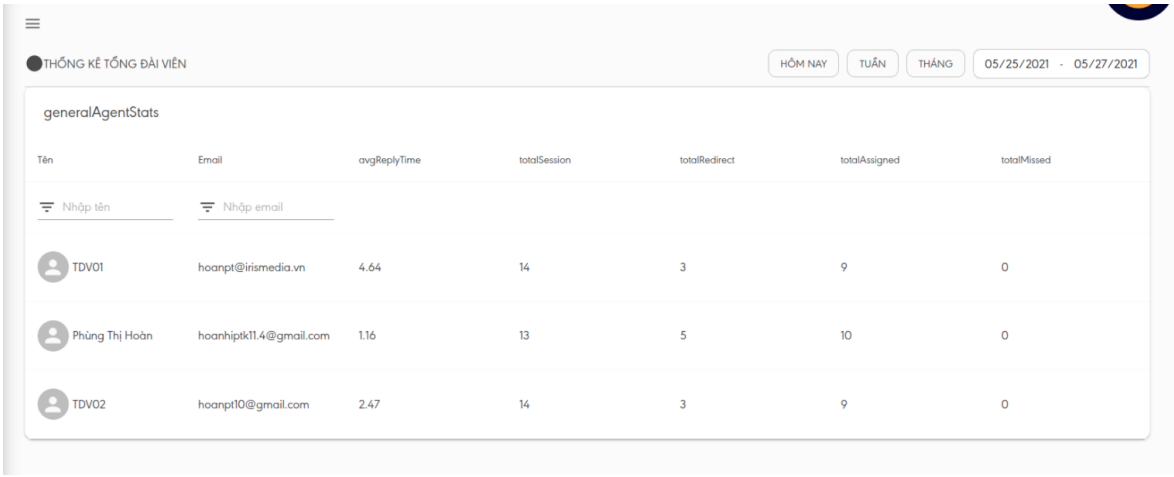
Hình 21 là giao diện giám sát các cuộc trò chuyện cũng như giám sát các khách hàng nhắn tin vào hệ thống theo thời gian thực. Bất kể có sự kiện nào xảy ra mà làm thay đổi trạng thái của khách hàng thì trên màn hình giám sát chat đều sẽ được cập nhật ngay lập tức. Trên màn hình có hiển thị thông tin "thời gian trong trạng thái" là quãng thời gian mà khách hàng này đổi trạng thái mới nhất. Điều này đặc biệt hữu ích khi quan sát các trạng thái đang chờ (waiting), quá hạn (time_out) hoặc tổng đài viên đang phục vụ (agent_serving) để quản trị viên có thể đưa ra những quyết định và phương án phù hợp.



Hình 21 giao diện giám sát chat

4.4.3.5 Thống kê hiệu suất tổng đài viên

Hình 22 là giao diện thống kê hiệu suất tổng đài viên. Bảng thống kê này hiển thị các số liệu mà hệ thống ghi nhận trong suốt quá trình chăm sóc khách hàng của tổng đài viên. Các số liệu: thời gian phản hồi trung bình, tổng số phiên phục vụ, tổng số cuộc trò chuyện mà tổng đài viên đã chuyển tiếp, tổng số cuộc trò chuyện mà tổng đài viên đã nhận, tổng số cuộc trò chuyện mà tổng đài viên bỏ lỡ đề có thể giúp người quản trị viên có cái nhìn khách quan nhất về năng suất làm việc của các tổng đài viên, từ đó đưa ra các chính sách, các giải pháp hợp lý với mỗi người.



Hình 22 màn hình thống kê hiệu suất tổng đài viên

4.5 Kiểm thử và triển khai

Để kiểm thử cho hệ thống quản lý và điều phối tin nhắn đa kênh, em sử dụng phương pháp kiểm thử hộ đen dựa trên các tính năng trên giao diện hiển thị. Trong **Bảng 14**, em sẽ trình bày tất cả các test case mà em đã tiến hành kiểm thử.

Bảng 14 Danh sách các test case

STT	Màn hình	Tên test case
1	Hỗ trợ trực tuyến	Kiểm tra thông tin hiển thị
2		Kiểm tra khả năng cập nhật thông tin khi nhắn tin
3		Kiểm tra khả năng thêm mới khách hàng đang chờ trong hàng đợi

4		Kiểm tra tính chính xác của bộ đếm số tin đang chờ và số tin đang phục vụ
5		Kiểm tra khả năng điều phối tin tự động trên các kênh nhắn tin
6		Kiểm tra tải thêm tin nhắn
7		Kiểm tra tải thêm khách hàng
8		Kiểm tra chuyển hướng cuộc trò chuyện
9		Kiểm tra chức năng nhận tin thủ công
10		Kiểm tra chức năng kết thúc cuộc trò chuyện
11		Kiểm tra chức năng điều hướng tin khi chatbot chuyển tiếp tin nhắn về tổng đài viên
12	Quản lý ca làm việc	Kiểm tra hiển thị danh sách chi tiết các ca làm việc
13		Kiểm tra chức năng thêm một chi tiết ca làm việc
14		Kiểm tra chức năng thêm ca làm việc
15		Kiểm tra chức năng chỉnh sửa ca làm việc
16		Kiểm tra chức năng xóa ca làm việc
17		Kiểm tra chức năng thiết lập chuỗi ca làm việc
18		Kiểm tra chức năng hủy một chi tiết ca làm việc
19	Giám sát	Kiểm tra trạng thái online/offline với tổng đài viên
20		Kiểm tra số lượng khách hàng phục vụ có thay đổi chính xác hay không
21		Kiểm tra hiển thị thông tin tổng đài viên

22		Kiểm tra hiển thị thay đổi trạng thái khách hàng
23		Kiểm tra hiển thị đếm giờ trong trạng thái
24		Kiểm tra hiển thị các hành động trên từng khách hàng
25		Kiểm tra chức năng chuyển hướng tin khi giám sát
26		Kiểm tra chức năng kết thúc trò chuyện khi giám sát
27	Cài đặt	Kiểm tra thay đổi chế độ điều phối tin
28		Kiểm tra thay đổi thời gian quá hạn của tổng đài viên
29		Kiểm tra thay đổi số lượng khách hàng phục vụ tối đa cho tổng đài viên

Chương 4 đã trình bày quá trình thiết kế, xây dựng, kiểm thử và triển khai hệ thống dựa trên các công nghệ em đã trình bày ở chương 3 để đáp ứng các yêu cầu ở chương 2. Trong chương 5, em sẽ trình bày các giải pháp và đóng góp nổi bật của bản thân trong quá trình hoàn thành đồ án.

Chương 5 Các giải pháp và đóng góp nổi bật

Chương 5 em sẽ trình bày những điều em thấy tự hào nhất trong suốt quá trình thực hiện đồ án cũng như phát triển hệ thống này. Cụ thể đó là thiết kế hệ thống theo kiến trúc Microservices, đưa ra giải pháp điều phối tin nhắn tự động, giải pháp quản lý ca làm việc và giải pháp giúp quản trị viên giám sát tổng đài viên và giám sát các khách hàng.

5.1 Thiết kế hệ thống theo kiến trúc microservices

5.1.1 Vấn đề

Theo truyền thống, các ứng dụng hay các dự án phần mềm thường được xây dựng theo kiến trúc nguyên khối (monolithic). Điều này có nghĩa là tất cả các thành phần, chức năng của project đều được xây dựng và phát triển trên một dự án. Cách phát triển này rất phổ biến đặc biệt là với các ứng dụng nhỏ với số lượng người dùng ít. Chúng dễ dàng để phát triển trên một IDE hoặc một công cụ nào đó xuyên xuốt vòng đời của ứng dụng. Điểm cộng nữa của những ứng dụng xây dựng theo kiến trúc này là vào những thời điểm ban đầu triển khai, chúng hoạt động vô cùng tốt khi mà số lượng người dùng chưa tăng lên đáng kể hay ứng dụng chưa phát sinh nhiều tính năng khác. Qua thời gian, ứng dụng cần mở rộng, thêm tính năng, lượng người dùng tăng, dữ liệu tăng, … Lúc này, dự án bắt đầu trở nên phức tạp, chỉ một thay đổi nhỏ thôi là cũng đã ảnh hưởng đến toàn bộ cả hệ thống, khiến việc bảo trì, tìm lỗi, phát triển ngày một trở nên khó khăn. Đối với các ứng dụng xây dựng theo kiến trúc nguyên khối, chúng có một sự chặt chẽ về kiến trúc, nhưng nó lại tiềm ẩn nhiều mối nguy khác. Chi phí, thời gian, công sức phát triển, tìm lỗi, kiểm tra tính năng sẽ đều tăng theo cấp số nhân cùng với kích thước của ứng dụng. Rồi còn vấn đề về tính sẵn sàng của các ứng dụng kiểu này khi mà tất cả các thành phần của hệ thống đều đang chạy trên cùng một tiến trình, chỉ với một lỗi xảy ra mở một thành phần thôi là có thể dẫn đến toàn bộ hệ thống dừng hoạt động ngay.

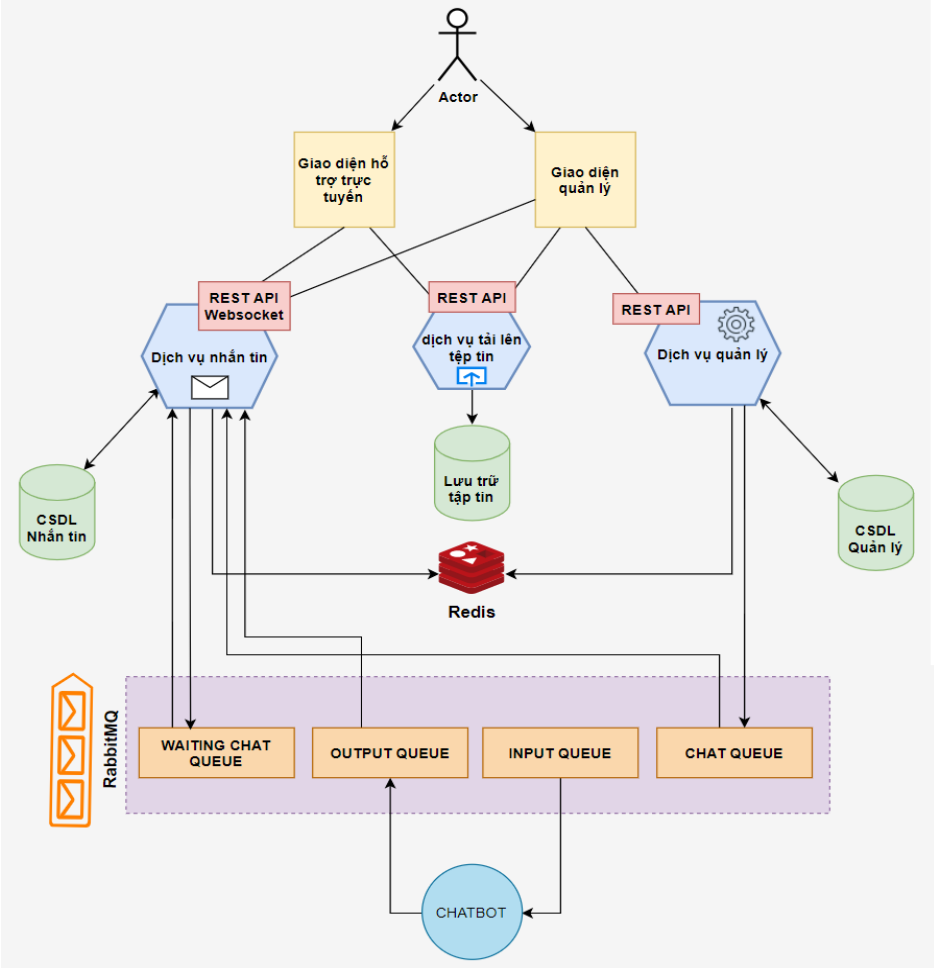
Đối với hệ thống quản lý điều phối tin nhắn đa kênh, thì người dùng có thể được chia làm hai nhóm là nhóm người nhắn tin đến hệ thống qua các kênh nhắn tin (website của doanh nghiệp, Facebook, zalo, …) và nhóm người quản lý. Dễ dàng nhận thấy nhóm người quản lý có số lượng rất ít so với nhóm người sử dụng hệ thống để nhắn tin. Khi số lượng khách hàng nhắn tin đến hệ thống tăng lên, hệ thống sẽ dễ có khả năng bị quá tải và bị sập. Ta có

thể giải quyết vấn đề này bằng cách tăng bộ nhớ và sức mạnh cho server xử lý, nhưng các làm này thường không hiệu quả do lãng phí tài nguyên khi mà tài nguyên cho người dùng nhắn tin thì không đủ còn tài nguyên cho người quản lý thì lại không dùng hết.

Khác với kiến trúc nguyên khối, mô hình kiến trúc microservices sẽ chia hệ thống thành các dịch vụ nhỏ đảm nhiệm các tính năng cụ thể đối với từng dịch vụ đồng thời được phát triển và hoạt động một cách độc lập nhất có thể. Vì lợi ích này, em đã thiết kế hệ thống theo kiến trúc microserivces, gộp những tính năng liên quan đến nhắn tin vào một dịch vụ, những tính năng liên quan tới quản lý vào một dịch vụ. Lúc này, khi mà số lượng người dùng nhắn tin đến hệ thống tăng lên thì ta có thể mở rộng dịch vụ nhắn tin tùy ý mà không cần phải lo lắng về dịch vụ quản lý sẽ bị thừa tài nguyên.

5.1.2 Giải pháp

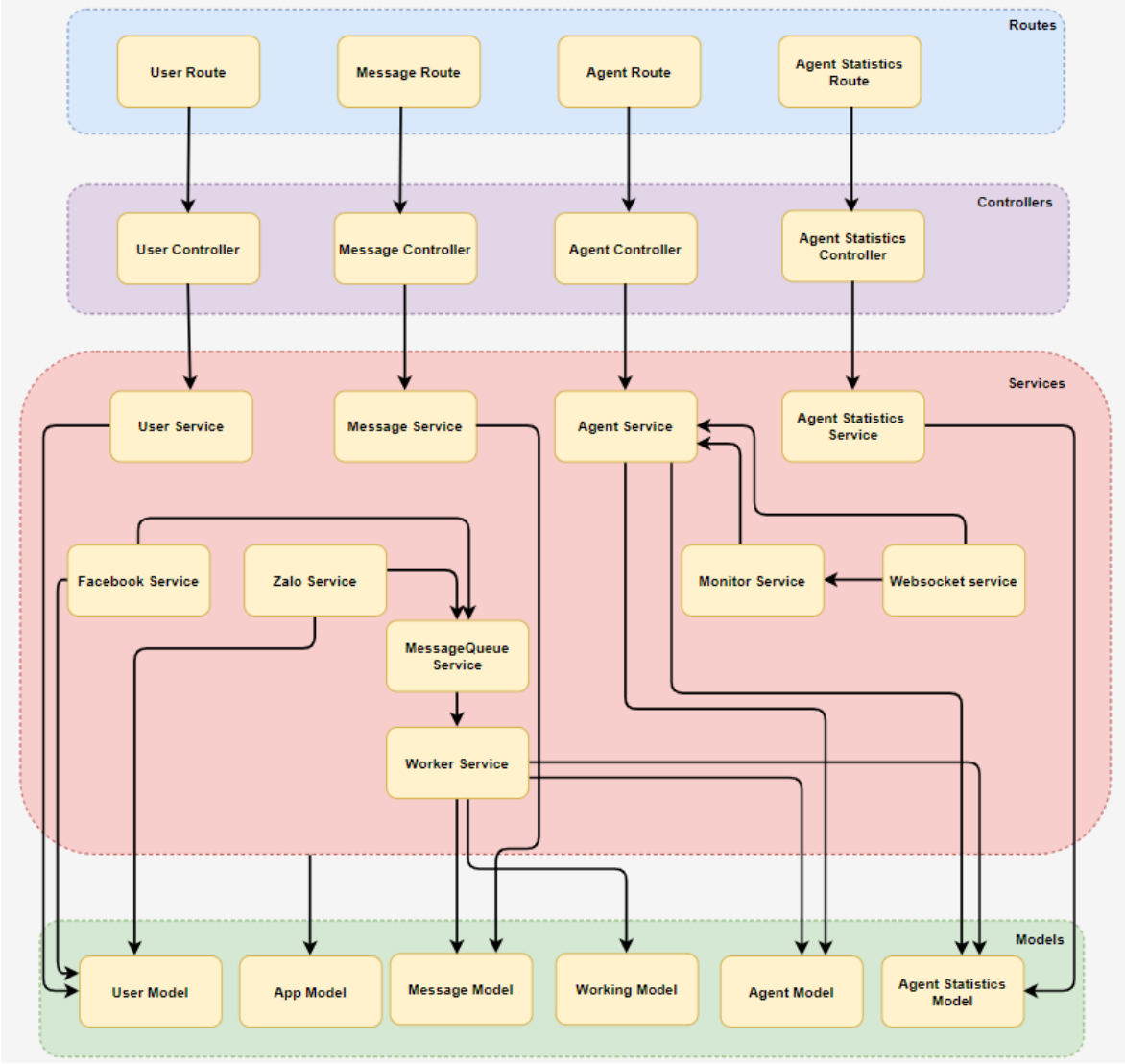
Nhận thấy ưu điểm mà kiến trúc Microservices mang lại, em đã thiết kế hệ thống dựa trên kiến trúc này. Hệ thống được chia ra làm ba dịch vụ con là (i) dịch vụ quản lý, (ii) dịch vụ nhắn tin, (iii) dịch vụ tải lên tệp tin như **Hình 23**.



Hình 23 Thiết kế hệ thống theo kiến trúc Microservices

5.1.2.1 Dịch vụ nhắn tin

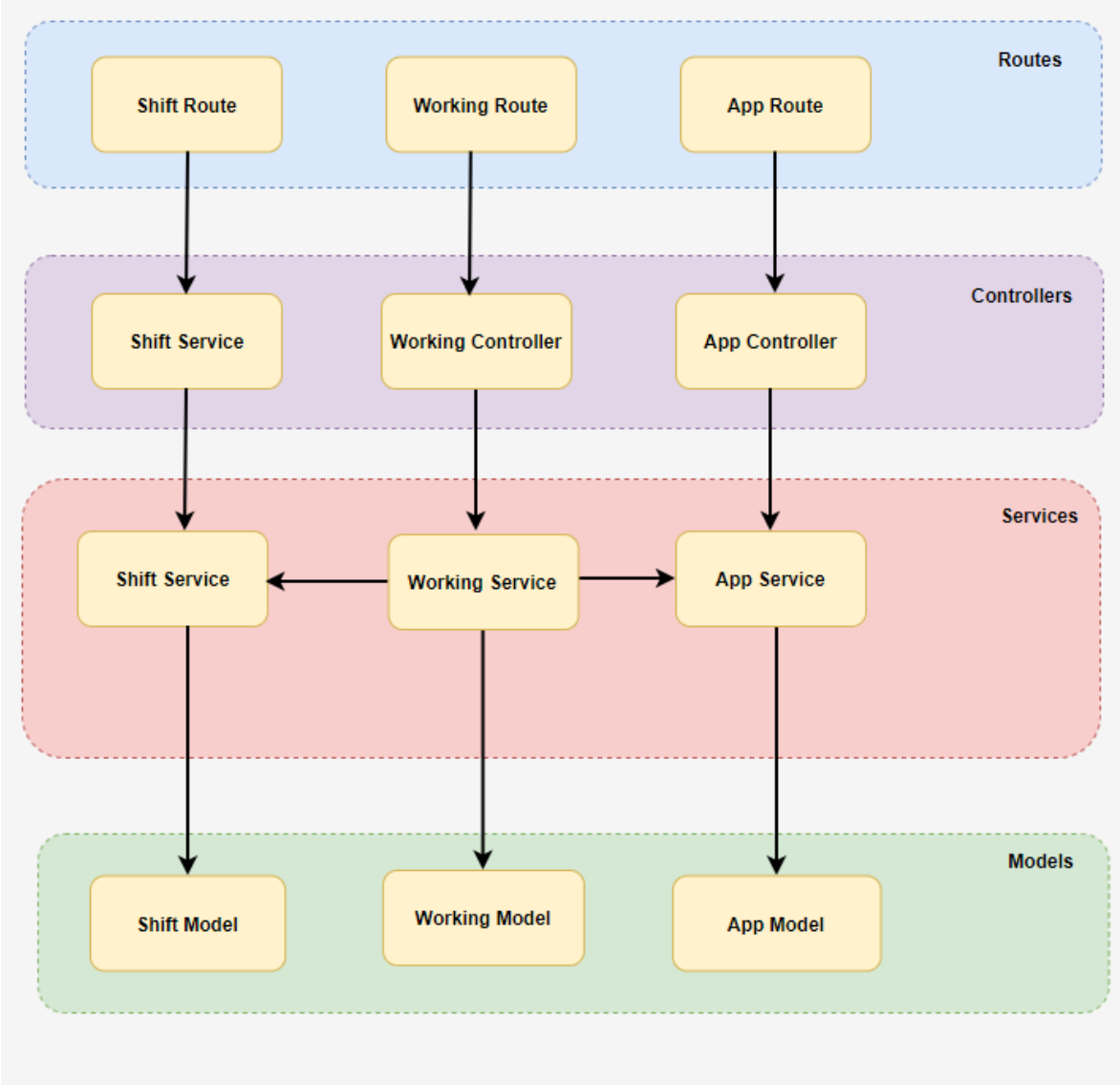
Dịch vụ nhắn tin tập trung vào việc quản lý khách hàng và các tin nhắn mà khách hàng gửi đến. Đồng thời, dịch vụ nhắn tin thực hiện việc điều phối tin nhắn đến cho các tổng đài viên, thực hiện các chức năng thống kê số liệu cần thiết. Dịch vụ nhắn tin gồm 4 thành phần chính là Routes, Controllers, Services và Models. Trong đó, Routes tạo ra các kết nối REST API, nhận vào các yêu cầu đầu vào để chuyển tới cho Controllers. Controllers sẽ có nhiệm vụ lấy các dữ liệu đầu vào cần thiết và chuyển tới Services. Service sẽ là nơi xử lý tất cả những logic liên quan đến nghiệp vụ của ứng dụng và cũng thực hiện truy vấn tới cơ sở dữ liệu MongoDB thông qua Models. Dịch vụ nhắn tin còn kết nối tới RabbitMQ để thực hiện điều hướng tin nhắn và cũng là để giao tiếp với các dịch vụ khác thông qua hàng đợi tin nhắn. Dịch vụ nhắn tin cũng cung cấp Websocket để phục vụ cho việc gửi nhận tin nhắn với client. Kiến trúc chi tiết của dịch vụ nhắn tin được mô tả chi tiết trong **Hình 24**.



Hình 24 Kiến trúc chi tiết dịch vụ nhắn tin

5.1.2.2 Dịch vụ quản lý

Dịch vụ quản lý cũng có các phần tương tự như dịch vụ nhắn tin nhưng nhiệm vụ thì tập trung vào việc quản lý tổng đài viên, cấu hình ứng dụng, tích hợp các ứng dụng bên thứ 3. Dịch vụ quản lý cũng được kết nối với RabbitMQ để thực hiện giao tiếp với các dịch vụ khác và cung cấp các REST API ra ngoài hệ thống. Kiến trúc chi tiết của dịch vụ nhắn tin được mô tả chi tiết trong **Hình 25**.



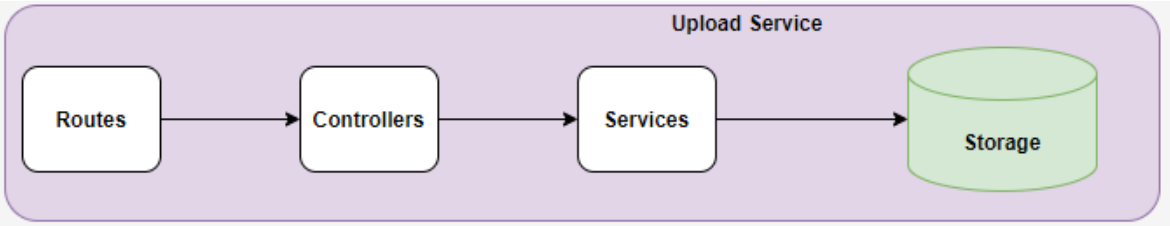
Hình 25 Kiến trúc chi tiết dịch vụ quản lý

5.1.2.3 Dịch vụ tải lên tệp tin

Dịch vụ tải lên tập tin có nhiệm vụ lưu trữ các tập tin đa phương tiện. chức năng tải lên tập tin này được tách ra thành một dịch vụ riêng vì không có nghiệp vụ phức tạp nhưng lại cần kho lưu trữ lớn. Bên cạnh đó, khi hệ thống mở rộng lên, các dịch vụ nhắn tin và quản lý được nhân lên, thì chỉ cần gọi đến dịch vụ này để tải lên các tập tin khi cần thiết mà không

cần phải lo đến việc dư thừa tài nguyên. Dịch vụ tải lên tập tin cung cấp ra ngoài các REST API phục vụ cho việc tải lên tập tin với các định dạng đa dạng và ảnh dưới dạng base64.

Kiến trúc của dịch vụ thì đơn giản, bao gồm 3 thành phần là Routes, Controllers và Services hoạt động tương tự dịch vụ nhắn tin và dịch vụ quản lý.

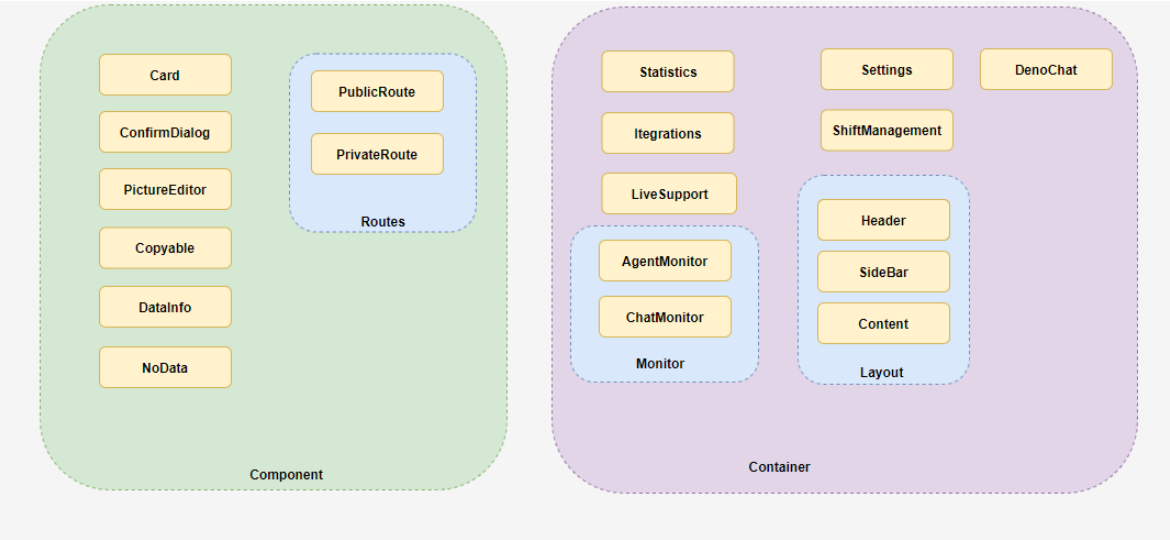


Hình 26 kiến trúc dịch vụ tải lên tập tin

5.1.2.4 Kiến trúc frontend

Kiến trúc frontend được chia ra làm 2 phần chính là Containers và Components như **Hình 27**.

Phần containers có 8 phần: (i) Statistics kết xuất giao diện thống kê thông tin khách hàng, (ii) Itegrations kết xuất giao diện tích hợp các ứng dụng bên thứ 3, (iii) Settings kết xuất giao diện cài đặt, cấu hình của ứng dụng, (iv) ShiftManagement kết xuất giao diện quản lý ca làm việc, (v) Monitor kết xuất 2 giao diện giám sát tổng đài viên và giám sát các cuộc trò chuyện theo thời gian thực, (vi) Layout là bố cục của toàn bộ màn hình, (vii) LiveSupport kết xuất giao diện nhắn tin hỗ trợ khách hàng, (viii) DemoChat kết xuất giao diện nhắn tin thông qua của sổ livechat.



Hình 27 Kiến trúc frontend

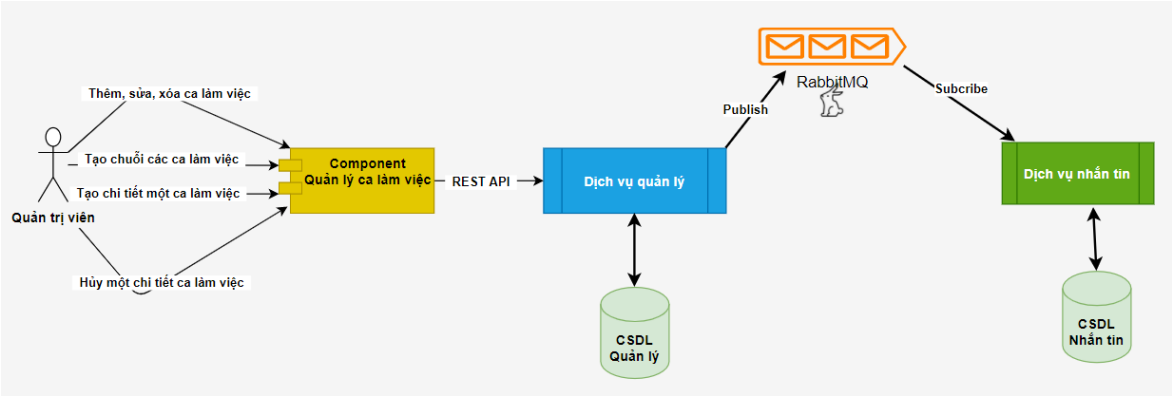
Phần components là các thành phần được sử dụng nhiều lần ở các màn hình. Khi cần, các màn ở containers sẽ gọi tới các component. Có tất cả 7 component đó là Card, ConfirmDialog, PictureEditor, Copyable, DataInfo, NoData và Routes. Trong đó, Route có thêm 2 thành phần là PublicRoute và PrivateRoute chịu trách nhiệm điều hướng và kết xuất ra màn hình giao diện tương ứng với địa chỉ trên thanh URL mà người dùng yêu cầu.

5.2 Phát triển tính năng quản lý ca làm việc

5.2.1 Vấn đề

Để có thể kiểm soát tốt chất lượng chăm sóc khách hàng, không bị thừa nhân lực khi ít khách hàng cần phục vụ hay tối ưu hóa nguồn lực tổng đài viên vào những thời điểm thích hợp, đồng thời dễ dàng quản lý vào theo dõi tổng đài viên thì em nhận thấy cần phải phân chia ra các ca làm việc cho các tổng đài viên. Chính vì thế em đã phát triển tính năng quản lý ca làm việc của các tổng đài viên để người quản lý có thể thực hiện theo dõi và sắp xếp công việc một cách hợp lý.

5.2.2 Giải pháp



Hình 28 sơ đồ tính năng quản lý ca làm việc

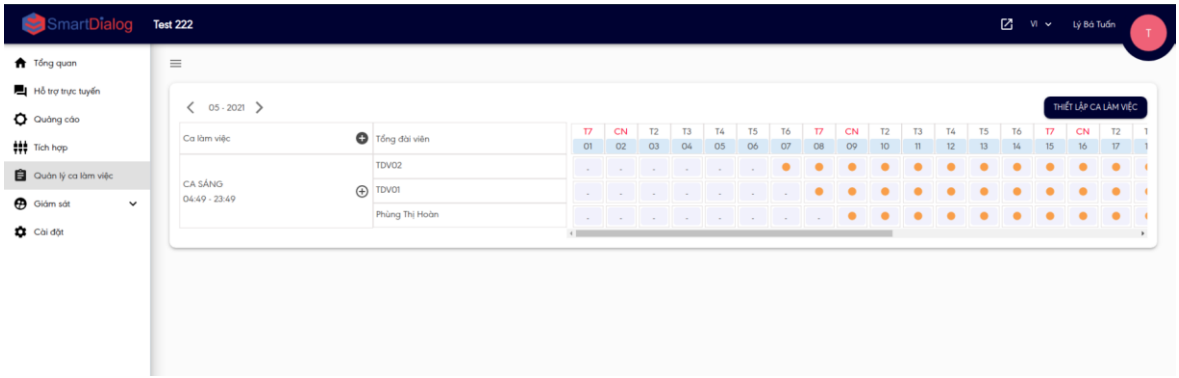
Quản trị viên sẽ có thể thực các hành động thêm, sửa, xóa ca làm việc, tạo chuỗi các ca làm việc, tạo chi tiết một ca làm việc, hủy một chi tiết ca làm việc của một tổng đài viên thông qua giao diện quản lý ca làm việc một cách dễ dàng. Mọi yêu cầu sẽ được gửi về dịch vụ quản lý thông qua giao thức REST API, dịch vụ quản lý sẽ xử lý các nghiệp vụ logic cần thiết và lưu dữ liệu cuối cùng vào cơ sở dữ liệu đồng thời sẽ đẩy dữ liệu đó lên CHAT_QUEUE trên RabbitMQ để đồng bộ với dữ liệu bên dịch vụ nhắn tin.

5.2.3 Kết quả đạt được

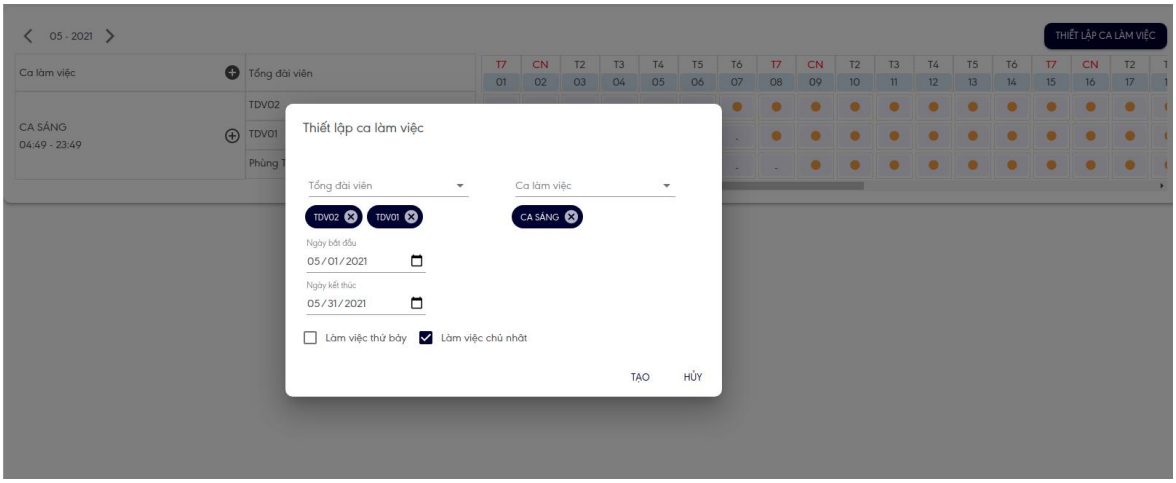
Kết quả đạt được sau khi triển khai các chức năng được thể hiện ở các màn hình dưới đây.

Hình 29 là màn hình quản lý ca làm việc. Ở màn hình này, em đang thiết kế hiển thị danh sách các chi tiết đăng ký ca làm việc theo từng tháng. Người dùng có thể điều chỉnh hiển thị tiến hoặc lùi tháng bằng cách bấm vào hai phím mũi tên ở góc trên bên trái của bảng hiển thị.

Hình 30 là giao diện khi quản trị viên chọn chức năng "Thiết lập ca làm việc". Đây là một dạng popup hiển thị, người quản trị có thể thực hiện chọn các tổng đài viên, các ca làm việc, chọn khoảng ngày cần cài đặt, thêm nữa là có thể tùy chọn có làm vào ngày thứ bảy hay chủ nhật hay không.



Hình 29 màn hình giao diện quản lý ca làm việc



Hình 30 màn hình giao diện thiết lập ca làm việc

5.3 Đề xuất giải pháp điều phối tin nhắn đa kênh

5.3.1 Vấn đề

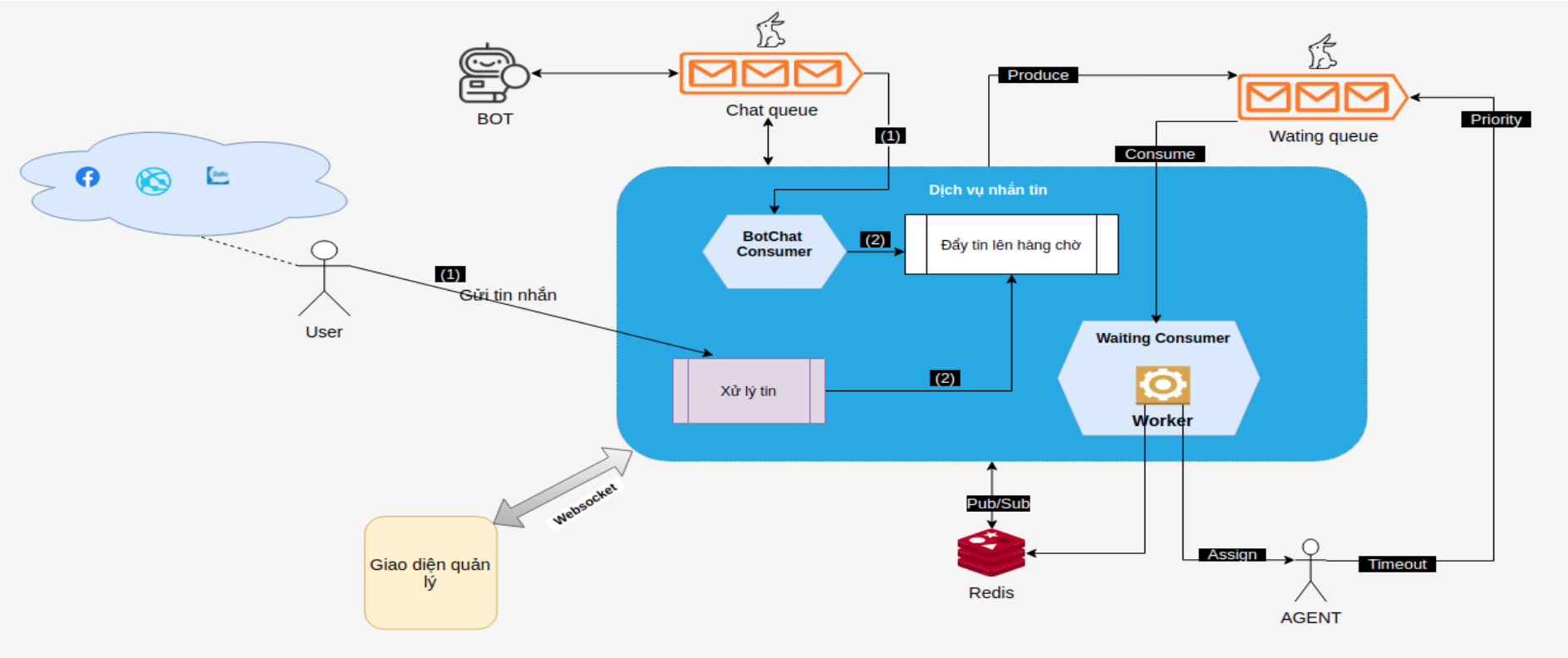
Ngày nay, khách hàng dễ dàng tìm thấy nhiều lựa chọn cho các loại sản phẩm mà họ cần đến, các doanh nghiệp cũng vì thế mà vẫn luôn tìm nhiều cách để tiếp cận đến nhiều tập

khách hàng nhanh và hiệu quả nhất có thể. Trong bối cảnh thời đại 4.0, khách hàng tiếp xúc với doanh nghiệp ở mọi nơi từ website doanh nghiệp, các trang mạng xã hội phổ biến thì chăm sóc khách hàng đa kênh là một lựa chọn không thể bỏ qua. Thêm nữa, giữa cuộc sống đầy hối hả, con người ta làm gì cũng cần được phản hồi nhanh chóng. Cụ thể, đối với việc chăm sóc khách hàng bằng kênh nhắn tin thì lại càng cần phải tối ưu vấn đề đó. Mỗi doanh nghiệp đều có một đội tư vấn viên thường trực để chăm sóc khách hàng, vì thế nếu các tin nhắn được đưa đến các tư vấn viên một cách tùy tiện thì rất dễ xảy ra tình trang bỏ qua các khách hàng đã đợi lâu nhưng chưa được chăm sóc hoặc chăm sóc không đồng đều dẫn đến việc đánh mất nhiều khách hàng tiềm năng. Việc điều phối tin nhắn từ khách hàng được chuyển đến các tư vấn viên sao cho tối ưu nhất có thể về mặt nguồn lực và thời gian là rất quan trọng. Chính vì vậy, em đã mạnh dạn đề xuất giải pháp điều phối tin nhắn đa kênh cho hệ thống nhắn tin.

5.3.2 Giải pháp

Em xây dựng tính năng cho phép ứng dụng tùy chỉnh giữa hai chế độ điều phối tin nhắn đó là điều phối tự động và điều phối thủ công. Đối với điều phối thủ công thì khi có tin nhắn mới từ người dùng nhắn đến hoặc từ Bot trả về cho các tư vấn viên thì tin sẽ được đẩy vào hàng chờ. Lúc này, các tư vấn viên đã đăng ký ca làm việc trước đó có thể vào danh sách các khách hàng đang chờ được phục vụ trên màn hình vào lựa chọn những người dùng mà mình muốn trả lời.

Còn với điều hướng tự động, các tư vấn viên sẽ không thể vào danh sách đang chờ để lựa chọn ra những khách hàng theo ý mình mà hệ thống sẽ có một bộ phận được gọi là worker chuyên chịu trách nhiệm phân bổ tin nhắn tới các tư vấn viên đang rảnh rỗi nhất. Các tin nhắn đang chờ sẽ được đưa đến worker bằng cách đẩy các tin đó lên hàng đợi trên RabbitMQ Broker xử lý (**Hình 31**). RabbitMQ Broker sẽ có nhiệm vụ phân phối các tin nhắn này tới các dịch vụ nhắn tin trên khắp hệ thống mà có Waiting Consumer đang chờ được bắn tin tới để tiêu thụ. Các tin sẽ được bắn lần lượt vào các Waiting Consumer để xử lý theo cơ chế là chỉ khi worker tìm được tổng đài viên rảnh rỗi để phân tin thì consumer mới có thể nhận được tin tiếp theo.

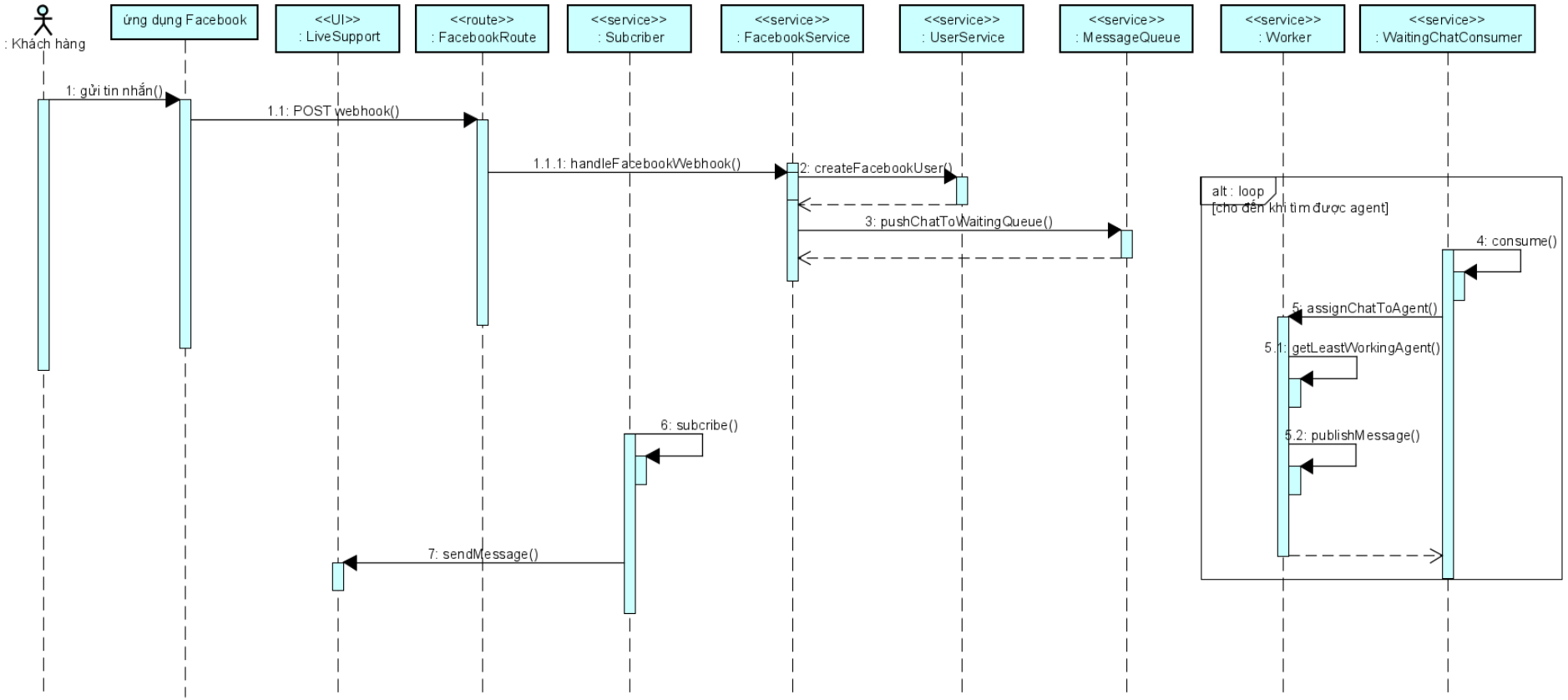


Hình 31 Sơ đồ điều hướng tin tự động

Trước tiên, worker sẽ lấy những tổng đài viên đã đăng ký ca làm việc trước đó rồi kiểm tra xem những ai đang trực tuyến. Nếu trong số người trực tuyến đều đang phục vụ số lượng người dùng tối đa thì worker sẽ tạo một vòng lặp cứ sau 3 giây sẽ tìm lại một lần cho đến khi tìm được thì thôi. Trong trường hợp có tổng đài viên đang rảnh rỗi (phục vụ ít khách hàng nhất), thì worker sẽ lập tức chỉ định khách hàng đang chờ tới tổng đài viên đó. Nếu có hai tổng đài viên trở lên đang phục vụ số khách hàng ít nhất bằng nhau, em sẽ kiểm tra lần lượt số lượng phiên phục vụ và số lượng tin nhắn mà những tổng đài viên đó đã gửi trong ca làm việc hiện tại để so sánh xem ai là người có số lượng phiên và tin nhắn ít nhất thì sẽ chỉ định cho người đó. Khi một tổng đài viên được chỉ định khách hàng, hệ thống sẽ đếm thời gian kể từ khi hệ thống chỉ định, nếu sau thời gian chờ đã được cài đặt mà tổng đài viên không trả lời, hệ thống sẽ tự động chuyển tin lại vào hàng đợi đang chờ (waiting queue) trên RabbitMQ nhưng với trạng thái là ưu tiên (priority) để đảm bảo khách hàng này sẽ được phục vụ sớm nhất có thể.

Hình 32 mô tả biểu đồ trình tự điều phối tin tự động khi mà một người dùng từ mạng xã hội Facebook nhắn tin tới một ứng dụng của hệ thống. luồng hoạt động của biểu đồ giả sử là ứng dụng đã được tích hợp với Facebook, để chế độ điều phối tin tự động và tin nhắn gửi tới ở trạng thái là chuyển tiếp cho tổng đài viên. Theo đó, khi gửi tin nhắn tới một trang trên Facebook được tích hợp với ứng dụng thì Facebook sẽ gửi một webhook về cho ứng dụng. Sau đó FacebookService sẽ lưu dữ liệu khách hàng trên Facebook vào cơ sở dữ liệu của hệ thống (createFacebookUser). Tiếp đến sẽ đóng gói tin nhắn và mã khách hàng đó thành một tin nhắn và đẩy lên hàng đợi tin nhắn đang chờ (pushChatToWaitingQueue) đồng thời kết thúc phương thức handleFacebookWebhook. Tin nhắn sau khi được đẩy lên hàng đợi sẽ ngay lập tức được RabbitMQ Broker chuyển đi tới các consumer. Khi nhận được một tin nhắn từ Broker, WaitingChatConsumer sẽ xử lý tin nhắn đó, rồi gọi WorkerService ra xử lý. Lúc này phương thức assignChatToAgent() của worker sẽ hoạt động. nếu phương thức này trả về một đối tượng rỗng, chứng tỏ là worker không tìm được tổng đài viên nào rảnh rỗi để chỉ định tin nhắn. Do đó, WaitingChatConsumer là một vòng lặp, sẽ lặp lại phương thức assignChatToAgent của worker cho đến khi nó trả về thông tin của một user thì kết thúc và gửi một tín hiệu (ack) cho RabbitMQ Broker biết là tin nhắn đã được xử lý xong.

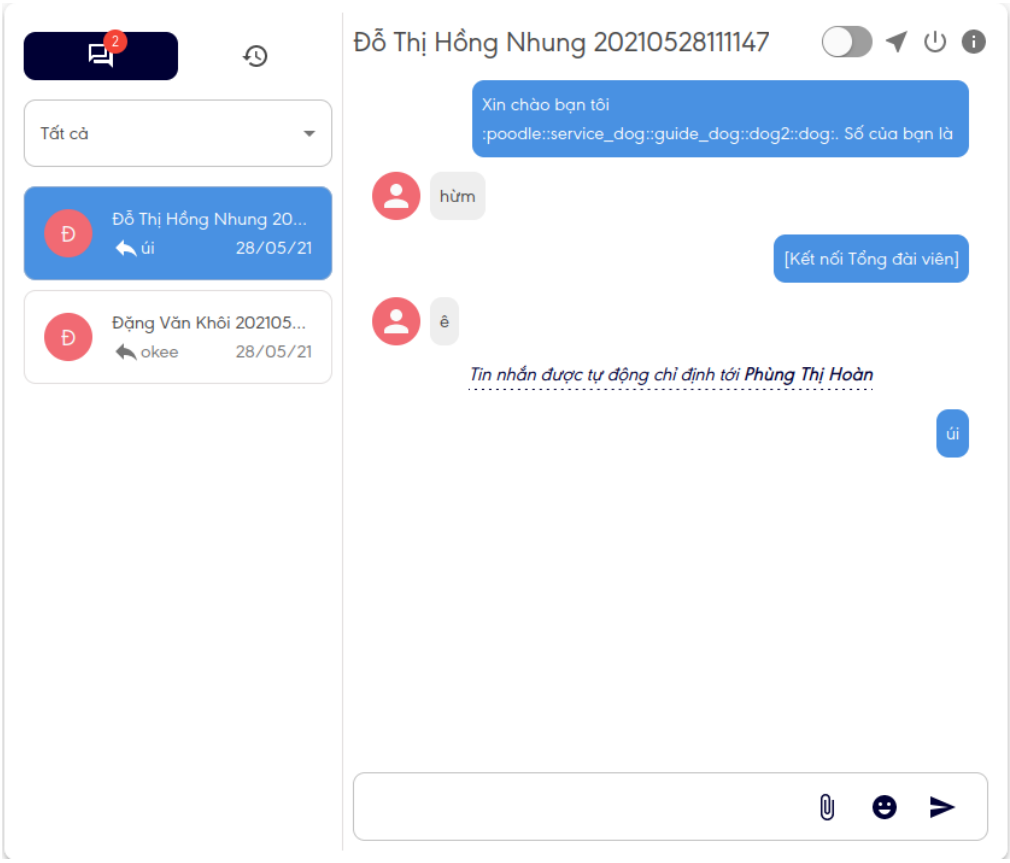
Đối với các trường hợp chatbot chuyển tiếp tin nhắn về cho tổng đài viên và các ứng dụng bên thứ ba khác như Viber, Zalo, Telegram cũng hoạt động tương tự facebook vì thế em sẽ không diễn giải thêm.



Hình 32 Biểu đồ trình tự điều hướng phối tin tự động từ Facebook

5.3.3 Kết quả đạt được

Kết quả đạt được sau khi triển khai chức năng thể hiện ở màn hình dưới đây.



Hình 33 giao diện điều phối tin tự động

Hình 33 là giao diện hiển thị của màn hình hỗ trợ trực tuyến với quyền đăng nhập ứng dụng là tổng đài viên. Ứng dụng lúc này đang ở trạng thái điều phối tin tự động, vì thế sẽ không có tab hiển thị danh sách khách hàng đang trong hàng chờ giống như đã được trình bày ở mục 4.4.3.1 chọn tin thủ công. Khi tổng đài viên được hệ thống chỉ định khách hàng phục vụ, trên màn hình sẽ hiển thị một dòng thông báo là: "Tin nhắn được chỉ định tới …" để người tổng đài viên có thể dễ dàng nhận biết.

5.4 Đề xuất giải pháp giám sát thời gian thực

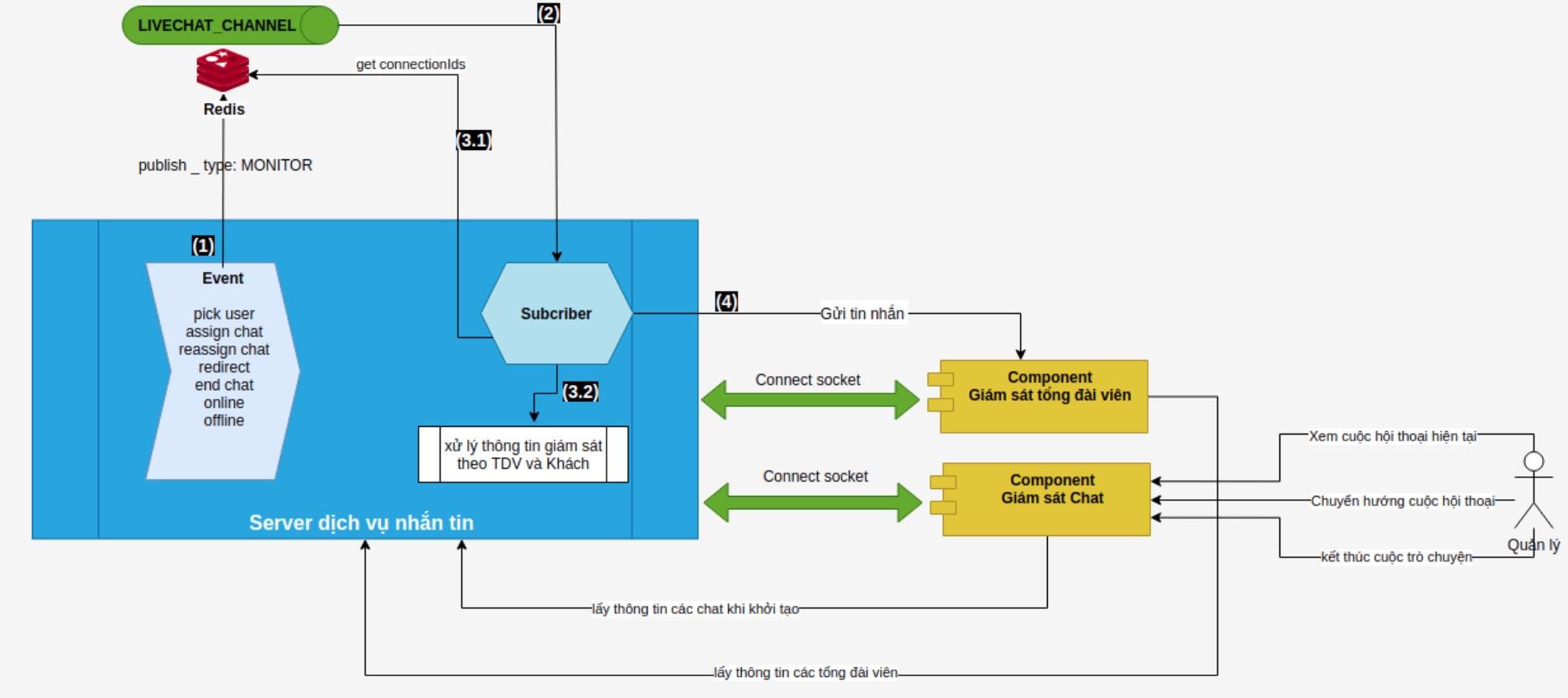
5.4.1 Vấn đề

Việc quản lý, giám sát và quan sát tổng quan các tin nhắn đang được phục vụ là rất quan trọng. Đồng thời, các tư vấn viên cũng phải được theo dõi sát sao tình trạng làm việc.

5.4.2 Giải pháp

Để giải quyết vấn đề trên, em đã xây dựng hai màn hình cho quản lý có thể giám sát nhân viên và các tin nhắn được gửi tới hệ thống. Hai màn hình giao diện này sẽ được kết nối Websoket với server của dịch vụ nhắn tin để trao đổi dữ liệu theo thời gian thực mỗi khi mà có sự thay đổi trạng thái của người dùng, xem **Hình 34**.

Mỗi khi có một sự kiện xảy ra, chẳng hạn như người dùng chọn khách hàng phục vụ (pick_user), chuyển hướng cuộc trò chuyện (redirect), kết thúc cuộc trò chuyện (end_chat), tổng đài viên online hay offline, … thì đều sẽ được hệ thống ghi nhận và đẩy một tin nhắn sự kiện lên kênh LIVECHAT_CHANNEL của Redis với kiểu (type) là MONITOR. Với cơ chế publish/subcribe, Redis sẽ từ đó mà đẩy các tin nhắn sự kiện tới các dịch vụ đã đăng ký kênh LIVECHAT_CHANNEL gọi là các Subcriber. Các subscriber khi nhận được tin sẽ xử lý thông tin giám sát theo tổng đài viên và khách hàng sau đó thực hiện gửi tin nhắn tới các thành phần giao diện giám sát đã được kết nối websocket từ trước. Đối với giám sát trò chuyện, quản trị viên có thể thực hiện các hành động như chuyển hướng cuộc hội thoại tới một tổng đài viên khác cho dù khách hàng đó đang là chatbot trả lời hay có trạng thái phục vụ nào đi nữa, hay có thể kết thúc cuộc trò chuyện ngay lập tức, … Các hành động này có thể được sử dụng tùy vào tình trạng, hoàn cảnh ở thời điểm mà người quản trị đang quan sát, điều này có thể giúp giải quyết nhanh chóng nhiều tình trạng, vấn đề không mong muốn, chẳng hạn như khách hàng chờ quá lâu, khách hàng được tư vấn quá lâu,…



Hình 34 Sơ đồ mô tả quy trình giám sát

Chương 6 Kết luận và hướng phát triển

6.1 Kết luận

Qua quá trình xây dựng và phát triển hệ thống "quản lý và điều phối tin nhắn đa kênh" dưới sự hướng dẫn của TS. Nguyễn Thị Thu Trang và sự nỗ lực của bản thân em, hệ thống đã đạt được các mục tiêu đề ra ban đầu. Thứ nhất, hệ thống được thiết kế theo mô hình kiến trúc Microservices giúp phát triển độc lập các module, dễ dàng mở rộng và bảo trì. Thứ hai, em đã xây dựng được chức năng điều phối tin nhắn tự động và thủ công, giúp cho người dùng dễ dàng tiếp cận và chăm sóc khách hàng nhanh nhất có thể. Thêm nữa, em cũng phát triển thêm được tính năng quản lý ca làm việc và giám sát tổng đài viên để người quản trị có thể sắp xếp công việc và đánh giá nhân viên của mình một cách hợp lý.

Tuy đã xây dựng được nhiều chức năng, nhưng em vẫn gặp phải một vài vấn đề chưa xử lý được, ví dụ như em vẫn chưa xử lý được việc nhận diện thông tin người dùng ẩn danh trên trình duyệt khi kết nối websocket, dẫn đến việc lưu trữ thông tin người dùng không chính xác.

Trong quá trình phát triển hệ thống, làm ĐATN lần này em cũng học hỏi và tích lũy thêm được rất nhiều kinh nghiệm và những điều bổ ích như việc tái sử dụng mã nguồn, tái cấu trúc mã nguồn, quy trình phát triển ra một phần mềm hoàn chỉnh đến tay người dùng, sử dụng các công nghệ quản lý mã nguồn như github. Bên cạnh đó là các kỹ năng làm việc nhóm, kỹ năng báo cáo và kỹ năng giải quyết vấn đề nữa.

6.2 Hướng phát triển

Do hạn chế về thời gian nên một số chức năng chưa được đạt tới mức hoàn thành so với thực tế đặt ra của hệ thống. Vì thế nên trong tương lai, hệ thống vẫn còn cần phải được xây dựng và bồi đắp thêm các tính năng để hệ thống trở nên hoàn thiện hơn.

- Với giao diện quản lý ca làm việc, sẽ cho người xem có thể tùy chọn thêm dạng hiển thị theo tuần thay vì chỉ hiển thị theo tháng, gây khó khăn khi quan sát trong trường hợp vào các ngày cuối tháng.
- Bộ lọc của phần giám sát chat hiện tại chưa được tối ưu và chính xác, nên em sẽ cần thời gian để chỉnh sửa bộ lọc.

- Bổ sung thêm các tính năng gợi ý các câu phổ biến cho tổng đài viên, để chăm sóc khách hàng nhanh chóng hơn.

Tài liệu tham khảo

[1] Kirupa Chinnathambi, *Learning React: A hands-on guide to building web applications using React and Redux*, tái bản lần 2, Addison-Wesley, 2018

[2] Kasun Indrasiri và Prabath Siriwardena, *Microservices for the Enterprise: Designing, Developing, and Deploying*, tái bản lần 2, Apress, 2018

[3] Chris Richardson, *Microservices Patterns*, tái bản lần 1, Manning Publications, 2018

[4] NodeJS, *https://nodejs.org*, truy cập lần cuối 30/05/2021

[5] ReactJS, *https://reactjs.org*, truy cập lần cuối 30/05/2021

[6] Websocket, *https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API*, lần truy cập cuối 30/05/2021

[7] Material UI, *https://material-ui.com*, lần truy cập cuối 30/05/2021

[8] RabbitMQ, https://rabbitmq.com, lần truy cập cuối 30/05/2021

[9] Redis, https://redis.io, lần truy cập cuối 30/05/2021

[10] LiveChat, *https://www.livechat.com*, truy cập lần cuối 30/05/2021

[11] tawk.to, *https://www.tawk.to*, truy cập lần cuối 30/05/2021