TRƯỜNG ĐẠI HỌC BÀ RỊA – VŨNG TÀU

KHOA CÔNG NGHỆ KỸ THUẬT -
NÔNG NGHIỆP CÔNG NGHỆ CAO

BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG KẾT NỐI DOANH NGHIỆP BKL

Trình độ đào tạo	: Đại học
Ngành	: Công nghệ thông tin
Chuyên ngành	: Lập trình ứng dụng di động, Game
Khóa học	: 2018-2022
Lớp	: DH18LT
Sinh viên thực hiện	: Nguyễn Minh Tâm
Mã số sinh viên	: 18033280
GVHD	: ThS. Nguyễn Tấn Phương

BÀ RỊA - VŨNG TÀU, NĂM 2021

LỜI CAM ĐOAN

Tôi xin cam đoan kết quả đạt được trong đồ án là sản phẩm của riêng cá nhân, không sao chép lại của người khác. Trong toàn bộ nội dung của đồ án, những điều được trình bày hoặc là của cá nhân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Vũng Tàu, ngày 01 tháng 12 năm 2021

Sinh viên thực hiện

Nguyễn Minh Tâm

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Vũng Tàu, ngày … tháng … năm 20…

Người hướng dẫn

ThS. Nguyễn Tấn Phương

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC HÌNH

LỜI NÓI ĐẦU

Công nghệ trên thế giới ngày nay đang phát triển một cách nhanh chóng, và các thiết bị di động đã không còn là một thứ xa lạ với mọi người. Chiếc điện thoại với một ứng dụng gọn nhẹ kết nối tới cơ sở dữ liệu từ xa đang làm một giải pháp tối ưu hóa cho các doanh nghiệp trên toàn cầu để cung cấp sự tiện lời tới cho khách hàng của họ.

Để bắt kịp được những nhu cầu đó, doanh nghiệp cần phải phát triển nhanh các ứng dụng điện thoại trong thời gian ngắn nhất nhưng lại tiếp cận tới được nhiều khách hàng với những nhu cầu và khẩu vị công nghệ, phong cách sống khác nhau.

Và để đáp ứng được nhu cầu đó của doanh nghiệp, các nền tảng Cross-Platform ra đời, với mục đích xây dựng một ứng dụng di động có thể đáp ứng được nhiều thiết bị với nhiều hệ điều hành khác nhau. Nhằm giảm thời gian xây dựng các ứng dụng và tối ưu hóa trải nghiệm người dùng với khách hàng.

Đồ án sau là một trong nhiều dự án mà bản thân tôi đã làm. Tại thời điểm này, dự án được chọn sau vẫn đang ở giai đoạn phát triển ở bộ phận Back-end. Nên có thể những báo cáo về dữ liệu trong đồ án sẽ không còn đúng với thực tế hoặc không còn có khả năng truy cập.

Xin trân trọng cảm ơn!

Sinh viên thực hiện

Nguyễn Minh Tâm

CHƯƠNG 1: GIỚI THIỆU

1.1. Flutter

1.1.1. Flutter là gì?

Flutter là một công cụ phát triển giao diện (UI Framework) để tạo ra các giao diện đẹp và chất lượng cao trên các nền tảng như Android, iOS, Web và Desktop. Flutter hoạt động với những mã nguồn được viết sẵn và kết hợp với những package mở để quá trình xây dựng trở nên dễ dàng và nhanh chóng. Flutter được phát triển với đội ngũ Google và hoàn toàn miễn phí.

1.1.2. Tại sao lại chọn Flutter?

- Flutter là một công cụ được phát triển bởi đội ngũ Google, và có một cộng đồng lớn mạnh theo từng ngày.
- Các ứng dụng xây dựng bằng Flutter có thể chạy trên nhiều nền tảng khác nhau như Android, iOS, Web, Desktop. Có chất lượng cao cả về giao diện lẫn hiệu suất sử dụng.
- Flutter khi chạy với 60fps, giao diện người dùng tạo ra được thực thi tốt hơn so với các công cụ khác như React Native và Ionic.
- Flutter sử dụng Dart, một ngôn ngữ được phát triển với Google, nhanh, hướng đối tượng với nhiều Chức năng như mixin, generic, isolate và static type. Flutter cũng có các thành phần giao diện riêng tương thích cho cả Android lẫn iOS, hầu hết các thành phần đó đều sẵn dùng và đều dựa theo nguyên tắc của Material Design.

1.1.3. Các đặc điểm nổi bật của Flutter

Fast Development: Tính năng Hot Reload hoạt động trong milliseconds để hiện thị giao diện. Sử dụng tập hợp các widget có thể customizable để xây dựng giao diện trong vài phút. Ngoài ra Hot Reload còn thêm các Chức năng, fix bug tiết kiệm thời gian hơn mà không cần phải thông qua máy ảo, máy android hoặc iOS.

Expressive and Flexible UI: Có rất nhiều các thành phần để xây dựng giao diện của Flutter vô cùng đẹp mắt theo phong cách Material Design và Cupertino, hỗ trợ nhiều các APIs chuyển động, smooth scrolling...

Native Performance: Các widget của flutter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

1.2. Công ty Maysoft

1.2.1. Giới thiệu

Maysoft Company Limited, tên viết tắt là Maysoft, tên gọi quốc tế là Maysoft Company Limited. Là công ty TNHH với các sản phẩm chính là về lập trình máy tính và phần mềm. Vốn điều lệ 300.000.000 VNĐ, điều hành bởi Mr. Nguyễn Minh Ý, giám đốc của công ty. Địa chỉ công ty tại Quận 7, thành phố Hồ Chí Minh.

Công ty có một sản phẩm đó là Maydental, một sản phẩm hỗ trợ quản lý các phòng khám nha khoa. Ngoài ra công ty cũng làm các sản phẩm như WordPress, gia công các phần mềm di động với Android, Swift, Flutter theo yêu cầu của khách hàng.

Với một đội ngũ lập trình vững vàng, công ty cũng được các doanh nghiệp, công ty nước ngoài bắt tay làm đối tác Outsourcing để gia công các phần mềm.

1.2.2. Ứng dụng BKL

Ứng dụng BKL là một phần mềm di động dùng để trao đổi thông tin giữa các doanh nghiệp, công ty với nhau, đồng thời có thể giúp doanh nghiệp đăng ký các sự kiện để thu hút các doanh nghiệp.

Công ty Toomba, là một công ty ở Hà Lan, làm về lĩnh vực công nghệ, gia công các phần mềm theo yêu cầu của khách hàng. Toomba đã là đối tác lâu năm của Maysoft, hai bên thường trao đổi nhân viên với nhau để cùng thực hiện các dự án lớn, có lợi ích cho cả hai bên.

Ứng dụng BKL đã bắt đầu trước đây khoảng 9 tháng, công nghệ mà Toomba dùng để phát triển là Flutter và kết nối với máy chủ qua API. Dù đã vượt thời gian dự kiến nhưng vẫn chưa thể hoàn thành do gặp rất nhiều sự cố về giao diện cũng như lỗi phần mềm. Qua đó, Maysoft đã tiến cử tôi qua để làm việc với công ty Toomba.

1.2.3. Khó khăn và vấn đề gặp phải khi tiếp quản dự án BKL

Khi mới bắt đầu được chuyển sang làm việc với công ty Toomba, sự khác nhau và múi giờ khiến cho việc giao tiếp và bắt tiếp tiến độ khá khó khăn, thời gian mà các hai bên có thể làm việc chung với nhau chỉ vỏn vẹn một buổi chiều. Dự án BKL đã bị trễ hạn hơn 6 tháng, vì thế mà áp lực về thời gian hoàn thành dự án cũng khiến cho quá trình tiếp quản trở nên khó khăn về mặt tâm lý cho cả hai bên.

Các mã nguồn cũ của dự án được viết một cách không rõ ràng, khiến cho việc sửa lỗi và thêm các Chức năng còn thiếu của BKL cũng trở thành một vấn đề, người thực hiện dự án đã rời đi và không thể trao đổi. Vì áp lực dự án mà bên Toomba cũng không muốn có một sự thay đổi lớn về mặt mà nguồn. Điều đó càng khiến cho quá trình xây dựng và tiếp quản càng khó khăn.

1.3. Lý do chọn đề tài

Một ứng dụng di động kết nối với máy chủ qua API không còn là điều gì mới lạ. Nó mang lại trải nghiệm nhanh gọn cho người sử dụng, và dễ dàng phát triển, nâng cấp đối với các công ty. Nhưng vì thế cũng cần có những kiến thức cơ bản để xây dựng ra một ứng dụng có thể đáp ứng những yêu cầu đó.

Flutter là một Framework giúp phát triển các ứng dụng Android, iOS, Web và cả Desktop được xây dựng với cùng một mã nguồn, giúp cho doanh nghiệp tiết kiệm chi phí nhân sự và cả thời gian phát triển sản phẩm. Flutter sử dụng những Component và các Design chuẩn được cấp bởi Material Design, giúp thời gian xây dựng ứng dụng trở nên nhanh chóng nhưng vẫn giữ được những quy tắc chuẩn và cho ra được một sản phẩm tốt về giao diện.

Mặc dù đã được Flutter hỗ trợ trong nhiều mặt khi xây dựng sản phẩm, nhưng khi càng đi sâu vào hệ thống, người lập trình viên cũng sẽ học được những kiến thức chuyên sâu của Framework. Để không gặp phải những khó khăn khi phát triển thì người lập trình viên cũng cần phải trao dồi những kiến thức từ các công cụ Native như chính Android, iOS, các cách một trang Web vận hành, điều hướng, v.v.

Để tài "Ứng dụng kết nối doanh nghiệp BKL", xây dựng bởi Flutter được chọn để làm đề tài vì qua quá trình xây dựng và triển khai sản phẩm đã giúp cho doanh nghiệp yêu cầu xử lý được các vấn đề khó khăn của họ. Ứng dụng đã được xây dựng trong khoảng 1 tháng, đáp ứng được các thiết kế của khách hàng, có thể giao tiếp với cơ sở dữ liệu.

1.4. Đối tượng nghiên cứu

- Nghiên cứu quy trình xây dựng giao diện cho một ứng dụng điện thoại.
- Nghiên cứu cách quản lý trạng thái của một ứng dụng điện thoại.
- Nghiên cứu quy trình liên kết với cơ sở dữ liệu qua API.
- Nghiên cứu, giải quyết các thuật toán cơ bản từ giao diện tới dữ liệu.
- Nghiên cứu các mã nguồn mở hỗ trợ cho quá trình xây dựng một ứng dụng điện thoại.

1.5. Phương pháp nghiên cứu

1.5.1. Các lý thuyết căn bản

- Tìm hiểu về ngôn ngữ Dart.
- Tìm hiểu cách hoạt động cơ bản của Flutter.
- Tìm hiểu cách quản lý trạng thái (State Management) của Flutter bằng GetX.
- Tìm hiểu quy trình kết nối tới cơ sở dữ liệu qua API bằng GetConnect.
- Tìm hiểu về Real-time chat, Push Notification và Schedule Notification.

1.5.2. Quá trình xây dựng thực tiễn

- Sử dụng Dart và Flutter làm ngôn ngữ và Framework chính để xây dựng sản phẩm.
- Sử dụng Swagger làm tài liệu hướng dẫn giao tiếp với cơ sở dữ liệu của doanh nghiệp qua API.
- Sử dụng kiến trúc DDD (Domain – Driven – Design) để quản lý và xây dựng sản phẩm.
- Sử dụng các packages để xây dựng ứng dụng như GetX, Local Notification,
- Ứng dụng các quy tắc của Material về UI/UX để xây dựng ứng dụng

CHƯƠNG 2: PHÂN TÍCH HỆ THỐNG

2.1. Chức năng

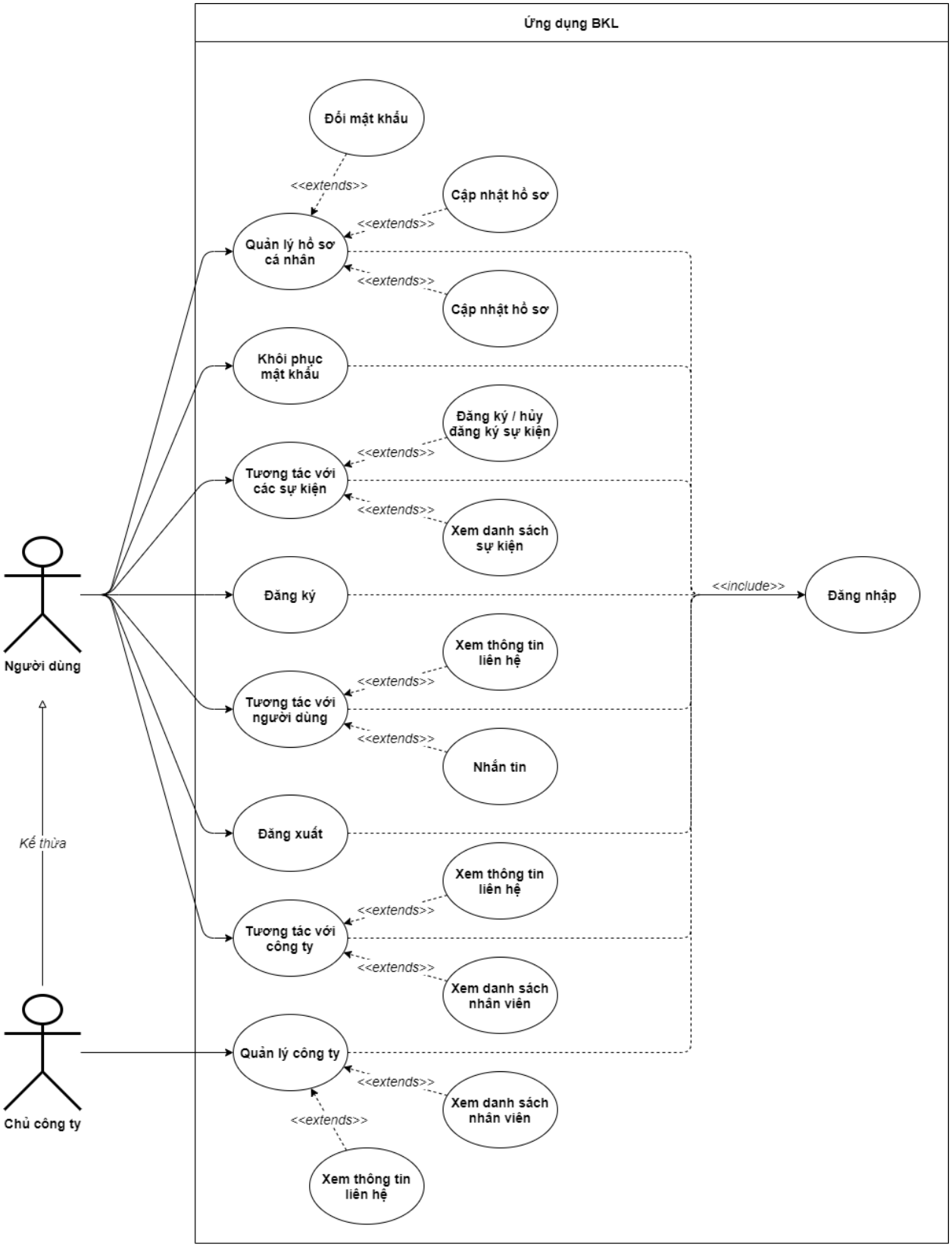
BKL là một ứng dụng chạy trên nền tảng Android và iOS, chia sẻ thông tin của các doanh nghiệp đã đăng ký trên hệ thống của BKL, đồng thời cung cấp dịch vụ tổ chức sự kiện và tham gia các sự kiện từ các doanh nghiệp để giao lưu, trao đổi kiến thức với nhau.

Doanh nghiệp yêu cầu ứng dụng BKL sẽ chỉ dùng cho người dùng để quản lý các Chức năng như:

- Xác thực người dùng.
- Quản lý, tương tác với các sự kiện
- Quản lý, tương tác với thông tin cá nhân
- Tương tác với thông tin của các người dùng và công ty khác trong hệ thống
- Nhắn tin với nhau qua ứng dụng in-app của BKL

Đối với những người dùng có đăng ký niêm yết công ty vào hệ thống của BKL, những người dùng đó sẽ vẫn có những chức năng giống như những người dùng thông thường khác và thêm các tính năng như

- Quản lý thông tin của công ty
- Quản lý nhân viên trong công ty
- Khả năng mời nhân viên qua mã QR



Hình 1: Sơ sồ Usecase tổng quát

2.2. Phân tích chức năng

2.1.1. Xác thực người dùng

- Yêu cầu:
  - Người dùng có thể đăng nhập vào ứng dụng với tài khoản và mật khẩu đã đăng ký. Sau đó chuyển tới màn hình chính.
  - Sau khi đăng nhập thành công, người dùng sẽ lấy một FCM Token của Firebase để có thể thực hiện các Chức năng như Push Notification.
  - Người dùng không thể đăng ký thông thường, phải đăng ký qua một mã QR được cấp bởi một công ty đã đăng ký trước qua trang web của BKL.
  - Sau khi đã quét được mã, người dùng phải cung cấp những thông tin cần thiết để có thể bắt đầu sử dụng ứng dụng.
  - Người dùng có thể gửi yêu cầu khôi phục lại mật khẩu với một email cung cấp
  - Ứng dụng có thể lưu đăng nhập của người dùng.
- Thực hiện.
  - Chức năng sẽ được chia ra thành các màn hình:
    - Màn hình đăng nhập
    - Màn hình khôi phục mật khẩu
    - Màn hình quét mã QR
    - Màn hình đăng ký
  - Luồng hoạt động.
    - Màn hình đăng nhập sẽ có thể điều hướng sang màn hình khôi phục mật khẩu và đăng ký, đồng thời cũng có thể chuyển sang màn hình chính nếu đăng nhập thành công
    - Màn hình khôi phục mật khẩu có thể gửi yêu cầu tới máy chủ và tự động lùi về nếu gửi thành công
    - Màn hình quét mã QR sẽ chuyển sang màn hình đăng ký nếu quét được dữ liệu theo điều kiện có sẵn
    - Màn hình đăng ký sẽ chuyển về màn hình đăng nhập sau khi đăng ký thành công
  - Dữ liệu
    - API đăng nhập
    - API đăng ký
    - API gửi yêu cầu khôi phục
    - Local storage để lưu lại thông tin đăng nhập.

2.1.2. Chức năng tương tác với sự kiện

- Yêu cầu:
  - Người dùng có thể xem các sự kiện sẽ bắt đầu trong tương lai.
  - Người dùng có thể xem thông tin chi tiết về khi chọn một sự kiện.
  - Người dùng có thể đăng ký tham gia sự kiện hoặc hủy đăng ký sự kiện với các điều kiện về mặt thời gian đăng ký.
  - Người dùng có thể xem danh sách các người dùng đã đăng ký sự kiện.
  - Người dùng có thể xem các sự kiện mà họ đăng đăng ký.
  - Người dùng có thể xem các sự đã kết thúc mà họ đã đăng ký.
  - Thiết bị sẽ thông báo cho người dùng khi sự kiện của họ sắp xảy
  - Người dùng có thể ghi thời gian của sự kiện vào ứng dụng lịch của thiết bị như Google Calendar và Lịch.
- Thực hiện.
  - Chức năng sẽ được chia ra thành các màn hình:
    - Màn hình hiển thị danh sách tất cả các sự kiện trong chưa kết thúc.
    - Màn hình hiển thị chi tiết và người tham dự của sự kiện.
    - Màn hình hiện thị các sự kiện sắp diễn ra mà người dùng đã đăng ký.
    - Màn hình hiển thị các sự kiện đã kết thúc mà người dùng đã đăng ký.
  - Luồng hoạt động.
    - Ở màn hình chính của ứng dụng, sẽ có một thành điều hướng ở dưới (Bottom Navigation Bar) và sẽ có tổng cộng 4 nút theo trình tự yêu cầu từ trái qua phải: Event, Member, Chat, Home
    - Khi bấm nút Event, sẽ hiển thị màn hình danh sách các sự kiện chưa bắt đầu. Trong đó, khi người dùng bấm chọn vào các sự kiện bất kỳ trên. Ứng dụng sẽ chuyển qua màn hình chi tiết sự kiện.
    - Khi bấm nút Home, sẽ hiển thị màn hình danh sách các sự kiện mà người dùng đã đăng ký và nút để chuyển qua màn hình các sự kiện đã kết thúc mà người dùng đã đăng ký.
    - Ở màn hình chi tiết của sự kiện, người dùng có thể lướt qua trái và phải để có thể xem thông tin sự kiện và danh sách những người tham dự
    - Trong màn hinh hiển thị thông tin, người dùng có thể đăng ký hoặc hủy đăng ký và có thể thêm thông tin sự kiện vào lịch cả nhân.
    - Khi đăng ký hay hủy đăng ký, ứng dụng sẽ cập nhật chức năng thông báo của thiết bị

- Dữ liệu
  - API lấy danh sách sự kiện
  - API đăng ký, hủy đăng ký tham gia sự kiện
  - API lấy danh sách những người tham dự *(Participant List)*
  - API lấy thông tin người tham dự *(Web-User List)*

2.1.3. Chức năng xem thông tin của Công ty và người dùng

- Yêu cầu:
  - Người dùng có thể xem danh sách các công ty đã đăng ký với ứng dụng.
  - Người dùng có thể xe, thông tin chi tiết của công ty và danh sách nhân viên của công ty.
  - Người dùng có thể xem danh sách tất cả các người dùng đã đăng ký với ứng dụng.
  - Người dùng có thể xem thông tin liên hệ của người dùng.
  - Người dùng có thể gửi tin nhắn tới người dùng.
- Thực hiện.
  - Chức năng sẽ được chia ra thành các màn hình:
    - Màn hình hiển thị danh sách tất cả các người dùng.
    - Màn hình hiển thị thông tin liên hệ của người dùng.
    - Màn hình hiển thị danh sách tất cả các công ty.
    - Màn hình hiển thị thông tin và danh sách nhân viên của công ty.
  - Luồng hoạt động.
    - Ở màn hình chính của ứng dụng, sẽ có một thành điều hướng ở dưới (Bottom Navigation Bar) và sẽ có tổng cộng 4 nút theo trình tự yêu cầu từ trái qua phải: Event, Member, Chat, Home.
    - Khi bấm nút Member, sẽ hiển thị danh sách các người dùng đăng ký với ứng dụng, nếu lướt qua trái có thể xem danh sách các công ty đăng ký với ứng dụng.
    - Nếu người dùng nhấn chọn một người dùng, ứng dụng sẽ chuyển qua màn hình hiển thị thông tin liên hệ của người dùng.
    - Tại màn hình hiển thị thông tin của người người, sẽ có nút nhắn tin để chuyển sang màn hình “Chat”.
    - Nếu người dùng nhấn chọn một công ty, ứng dụng sẽ chuyển qua màn hình hiển thị thông tin liên hệ của công tin và danh sách nhân viên.
  - Dữ liệu
    - API lấy danh sách công ty.
    - API lấy danh sách người dùng.
    - Firebase kiểm tra tin nhắn giữa hai người dùng.

2.1.4. Chức năng gửi tin nhắn giữa người dùng với người dùng

- Yêu cầu: Xây dựng một hệ thông nhắn tin cơ bản giống Messager
    - Người dùng có thể nhắn tin với một người dùng khác khi xem thông tin liên hệ của nhau
    - Người dùng có thể xem danh sách những người họ đã nhắn tin.
    - Người dùng có thể biết được có tin nhắn mới qua cách làm nổi bật giao diện và qua thông báo đẩy (Push Notification) từ Firebase.
    - Người dùng có thể nhắn tin cho nhau.
    - Người dùng có thể xem thông tin liên hệ của người mình đang nhắn.
    - Người dùng có thể lưu trữ nhóm trò chuyện, các tin nhắn sẽ không biến mất.
- Thực hiện.
    - Chức năng sẽ được chia ra thành các màn hình:
        - Màn hình hiển thị danh sách những người dùng đã nhắn tin hoặc được nhắn tới
        - Màn hình nhắn tin
    - Luồng hoạt động.
        - Ở màn hình chính của ứng dụng, sẽ có một thành điều hướng ở dưới (Bottom Navigation Bar) và sẽ có tổng cộng 4 nút theo trình tự yêu cầu từ trái qua phải: Event, Member, Chat, Home.
        - Khi bấm nút Chat, sẽ hiển thị danh sách các những nhóm trò chuyện mà người dùng đã nhắn hoặc được nhắn.
        - Ở màn hình thông tin liên hệ của một người dùng khác. Người dùng có thể nhắn tin hoặc bắt đầu nhắn tin qua một nút ở trong màn hình. Qua đó ứng dụng sẽ di chuyển qua màn hình trò chuyện.
        - Ở màn hình hiển thị các nhóm trò chuyển, nếu người dùng chọn một nhóm, ứng dụng sẽ di chuyển tới nhóm trò chuyện đó.
        - Ở trong màn hình trò chuyện, người dùng có thể bấm vào Avatar của người mình đang nhắn tin để xem thông tin của họ.
    - Dữ liệu
        - API lấy danh sách người dùng theo ID.
        - API gửi thông báo đẩy tới người dùng.
        - Firebase lấy danh sách nhóm trò chuyện.
        - Firebase gửi tin nhắn.

2.1.5. Chức năng xem, chỉnh sửa thông tin cá nhân

- Yêu cầu:
  - Người dùng có thể xem thông tin cá nhân
  - Người dùng có thể chỉnh sửa thông tin cá nhân
  - Người dùng có thể đổi mật khẩu
  - Người dùng có thể gửi đơn hỗ trợ về cho nhà phát triển ứng dụng
  - Đăng xuất
- Thực hiện.
  - Chức năng sẽ được chia ra thành các màn hình:
    - Màn hình hồ sơ cá nhân, hiển thị các Chức năng mở rộng.
    - Màn hình chỉnh sửa thông tin cá nhân.
    - Màn hình đổi mật khẩu.
    - Màn hình gửi đơn hỗ trợ.
  - Luồng hoạt động.
    - Ở màn hình chính của ứng dụng, sẽ có một thành điều hướng ở dưới (Bottom Navigation Bar) và sẽ có tổng cộng 4 nút theo trình tự yêu cầu từ trái qua phải: Event, Member, Chat, Home.
    - Trong màn hình Home, người dùng có thể bấm vào Avatar phía trên để di chuyển tới màn hồ sơ cá nhân
    - Ở màn hình hồ sơ, người dùng có thể di chuyển qua các màn hình chỉnh sửa thông tin người dùng, đổi mật khẩu và cả gửi đơn hỗ trợ
    - Ở cuối màn hình hồ sơ, người dùng có thể đăng xuất khỏi ứng dụng, và di chuyển ra màn hình đăng nhập và xóa Token FCM của người dùng.
  - Dữ liệu
    - API lấy thông tin công ty theo ID của người chủ.
    - API thay đổi thông tin người dùng, đổi mật khẩu.
    - API gửi đơn hỗ trợ.
    - API đăng xuất.

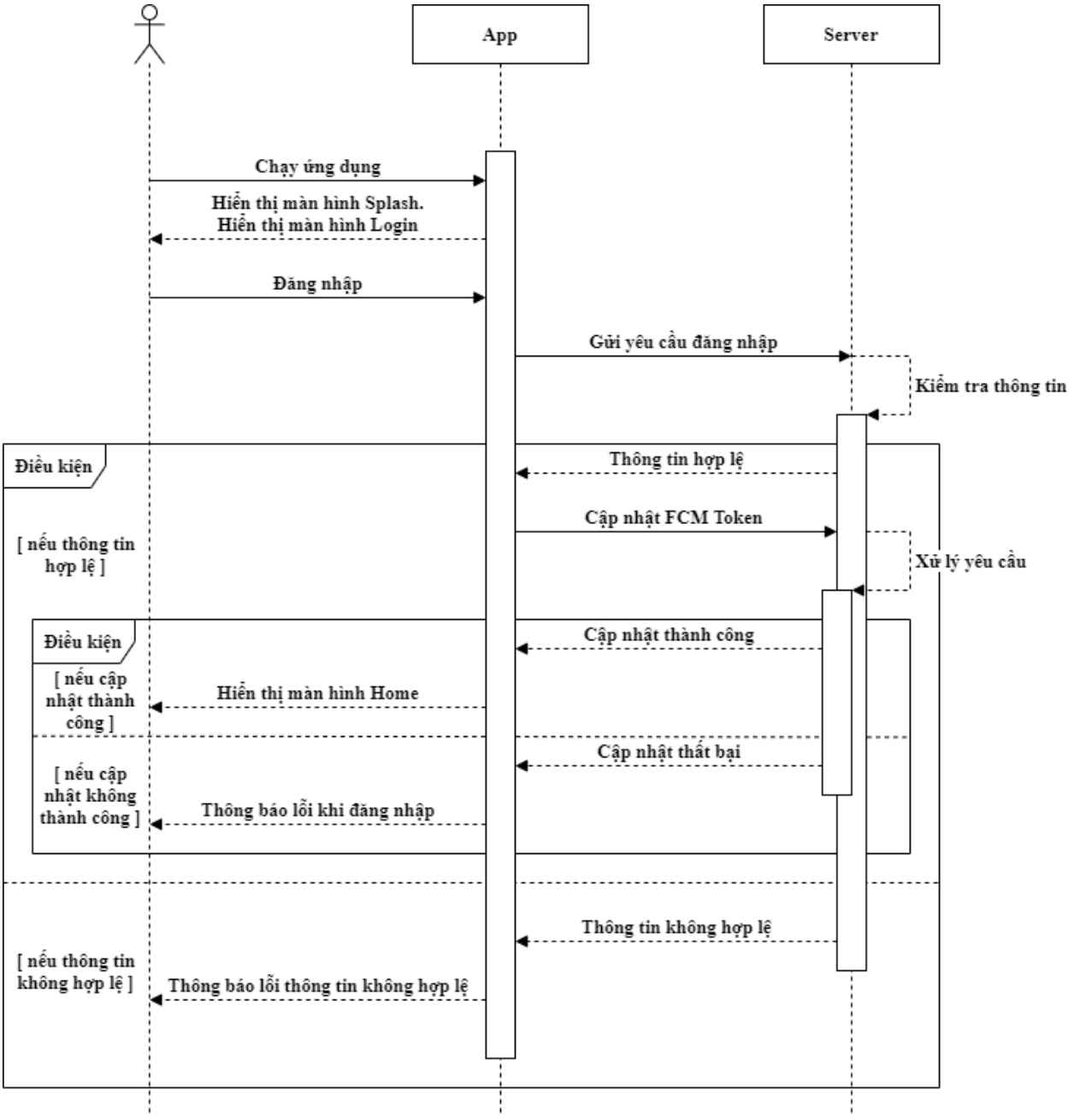
2.1.6. Chức năng xem, chỉnh sửa thông tin doanh nghiệp

- Yêu cầu: Chỉ dành cho người dùng là chủ của một doanh nghiệp.
    - Người dùng có thể xem thông tin của doanh nghiệp
    - Người dùng có thể chỉnh sửa thông tin doanh nghiệp
    - Người dùng có thể đổi logo của doanh nghiệp
    - Người dùng có thể hiện thị mã QR để cho các người khác dùng để đăng ký, thay đổi thông tin của công ty và xóa nhân viên khỏi công ty.
- Thực hiện.
    - Chức năng sẽ được chia ra thành các màn hình:
        - Màn hình hiển thị thông tin và danh sách nhân viên của công ty.
        - Màn hình chỉnh sửa thông tin công ty.
    - Luồng hoạt động.
        - Ở màn hình chính của ứng dụng, sẽ có một thành điều hướng ở dưới (Bottom Navigation Bar) và sẽ có tổng cộng 4 nút theo trình tự yêu cầu từ trái qua phải: Event, Member, Chat, Home.
        - Trong màn hình Home, người dùng có thể bấm vào Avatar phía trên để di chuyển tới màn hồ sơ cá nhân
        - Nếu người dùng là chủ của một công ty, màn hình sẽ hiển thị một nút để di chuyển tới màn hinh thông tin công ty. Trong màn hình đó sẽ có nút để di chuyển tới màn hình chỉnh sửa thông tin công ty.
        - Trong màn hình thông tin, danh sách nhân viên của công ty, người dùng có thể chọn một nhân viên để có thể xóa khỏi công ty.
    - Dữ liệu
        - API lấy thông tin công ty theo ID của người chủ.
        - API thay đổi thông tin công ty, xóa nhân viên khỏi công ty.

2.3. Sơ đồ phân tích các chức năng

2.3.1. Chức năng đăng nhập

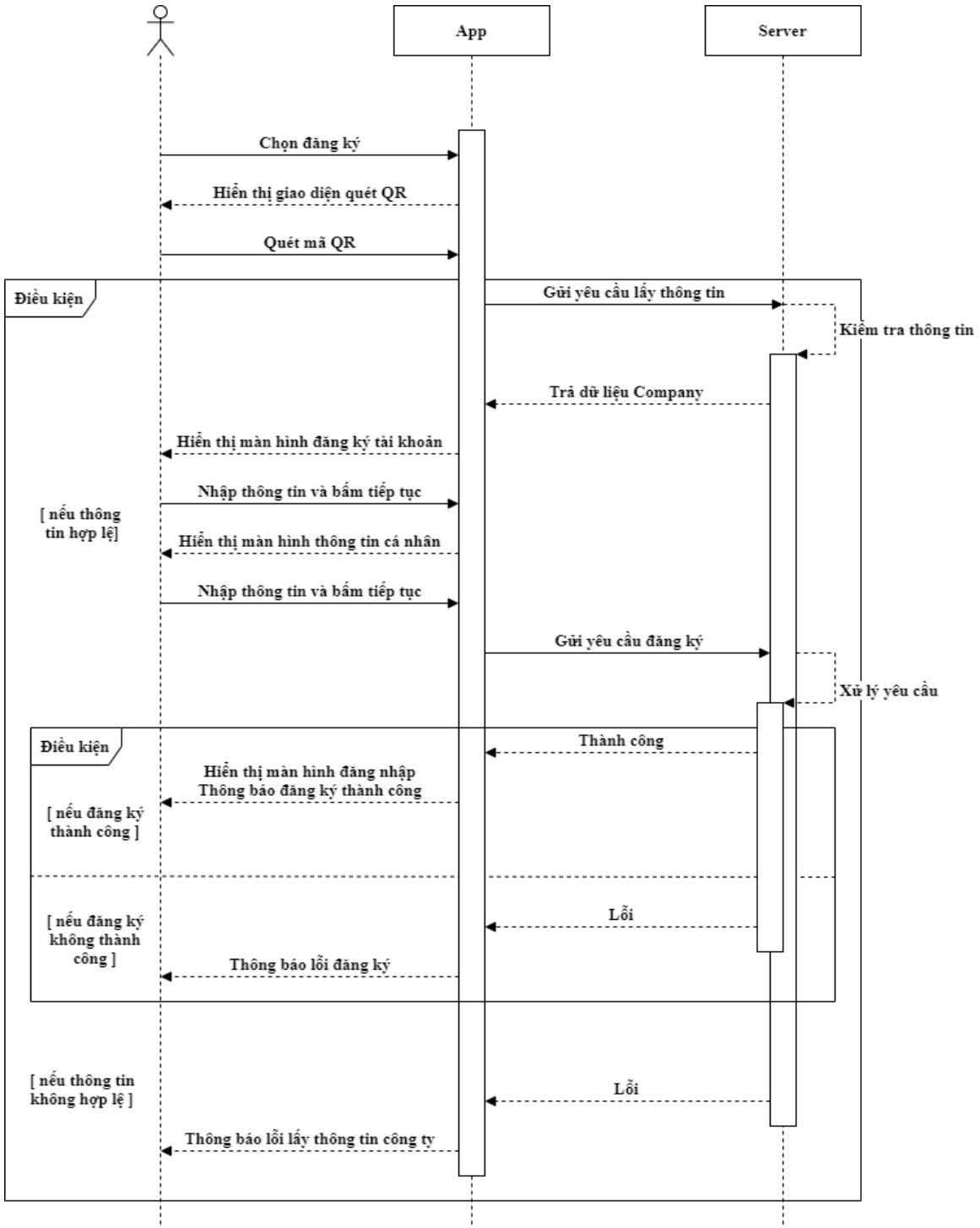
- **Mô tả:** Người dùng có thể đăng nhập vào ứng dụng bằng email và mật khẩu đã đăng ký. Nếu đăng nhập thành công, ứng dụng sẽ lưu đăng nhập và di chuyển tới màn hình Home
- **Sơ đồ tuần tự:**



Hình 2: Sơ đồ tuần tự chức năng đăng nhập

2.3.2. Chức năng đăng ký

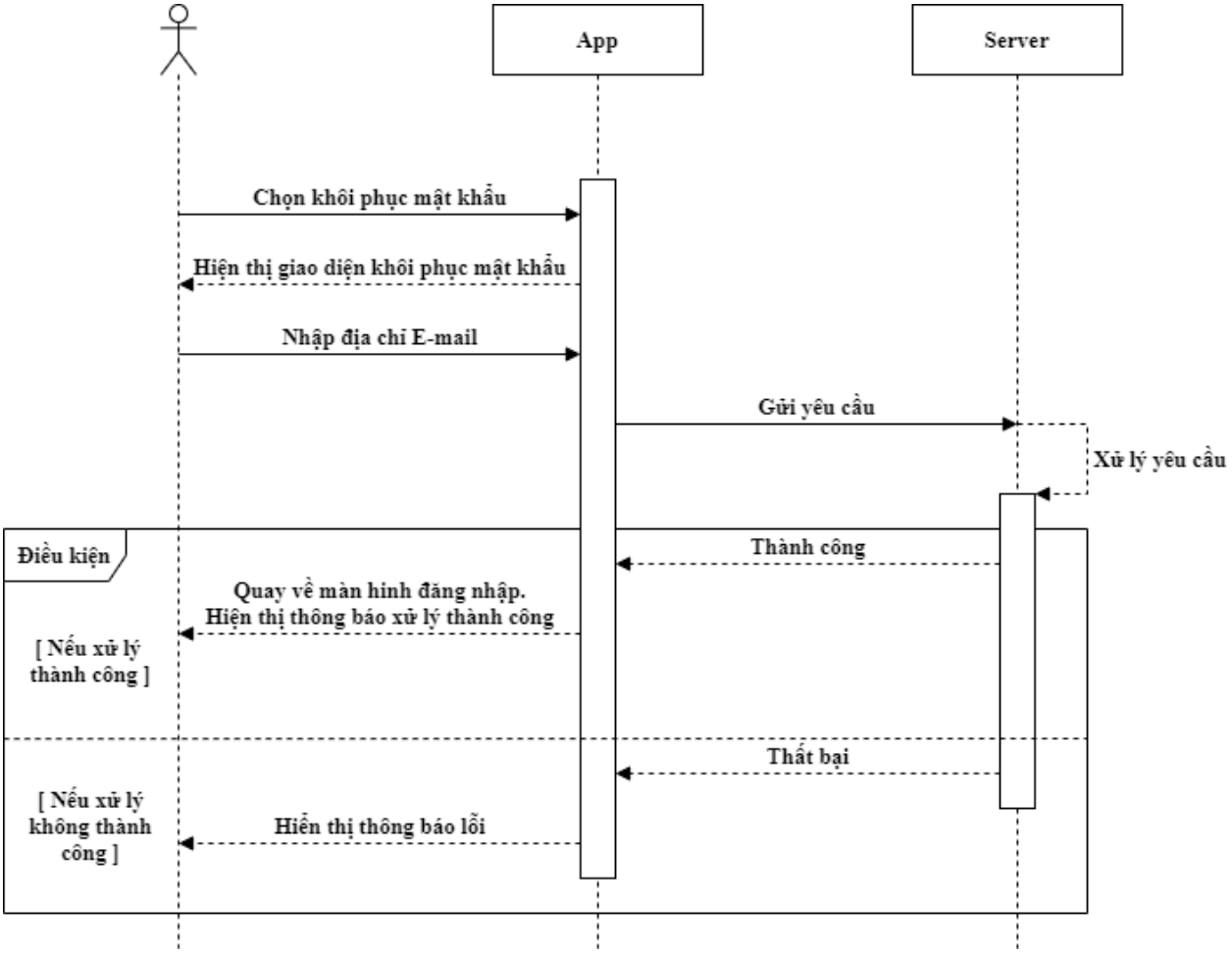
- **Mô tả:** Người dùng khi đăng ký phải có một mã QR từ một công ty đã đăng ký với hệ thông, với mã QR này, người dùng có thể đăng ký dưới quyền người dùng và là nhân viên của công ty mà họ đã đăng ký.
- **Sơ đồ tuần tự:**



Hình 3: Sơ đồ tuần tự chức năng đăng ký

2.3.3. Chức năng khôi phục mật khẩu

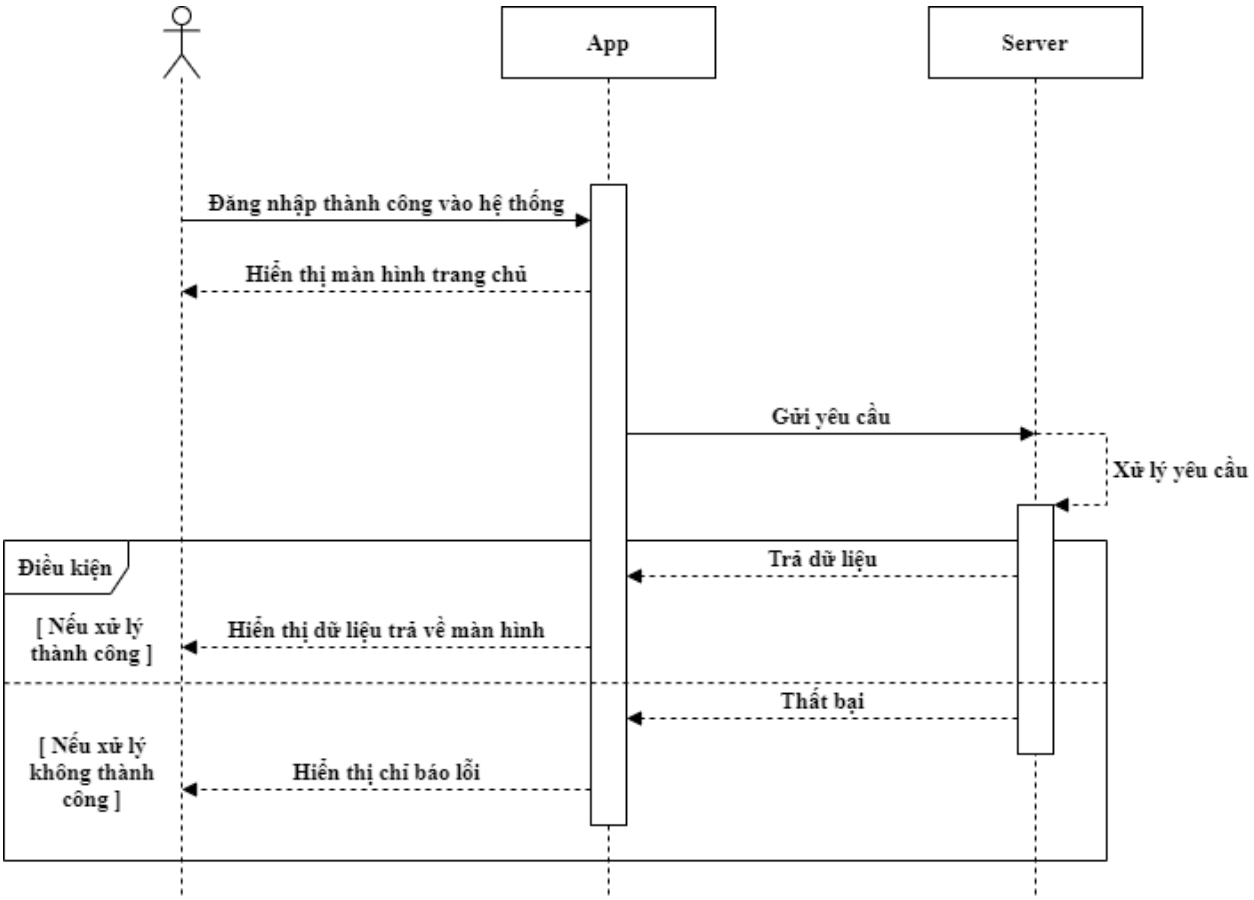
- **Mô tả:** Khi ở màn hình đăng nhập, người dùng có thể yêu cầu khôi phục tài khoản bằng địa chỉ E-mail mà họ đã đăng ký, hệ thống sẽ gửi hướng dẫn khôi phục về email đó nếu tồn tại.
- **Sơ đồ tuần tự:**



Hình 4: Sơ đồ tuần tự chức năng khôi phục mất khẩu

2.3.4. Chức năng xem sự kiện đã đăng ký

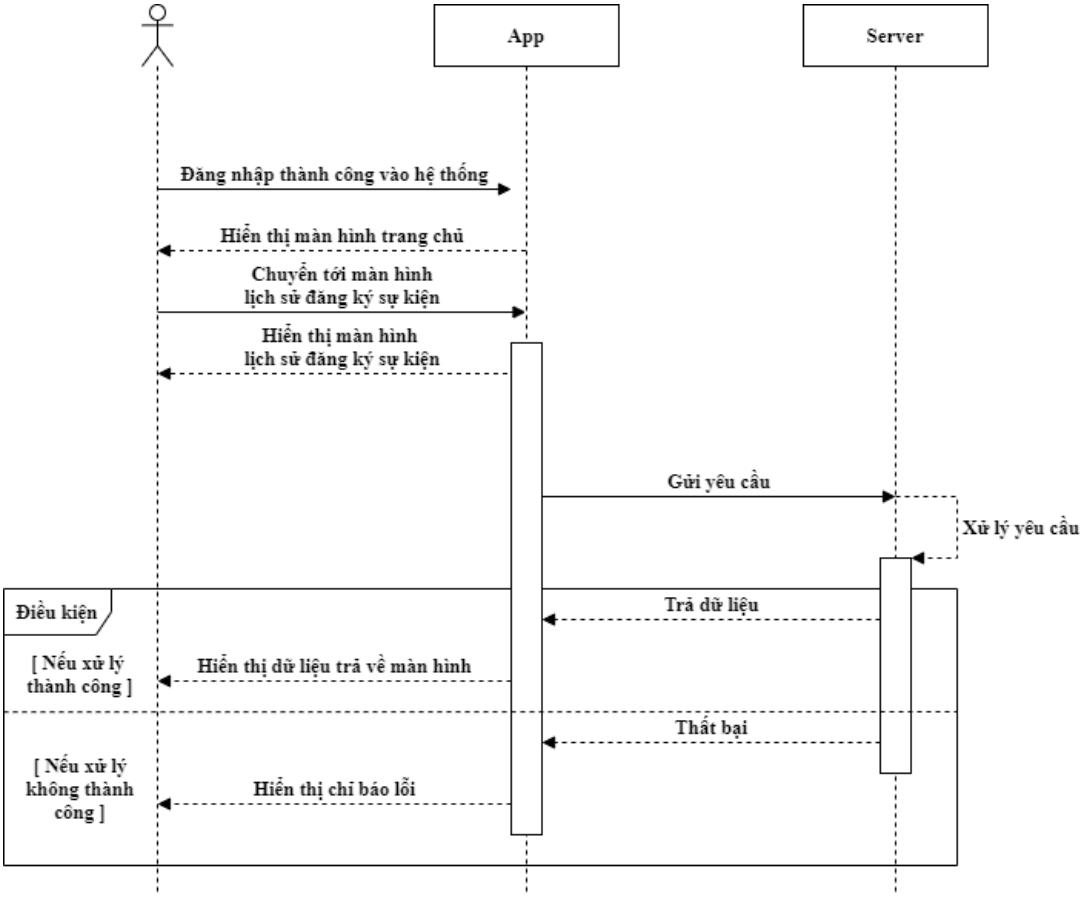
- **Mô tả:** Sau khi đăng nhập, người dùng có thể xem những sự kiện mà họ đã đăng ký tại màn hình trang chủ, người dùng cũng có thể xem những sự kiện đã kết thúc mà họ đã đăng nhập.
- **Sơ đồ tuần tự:**



Hình 5: Sơ đồ tuần tự chức năng xem sự kiện đã đăng ký

2.3.5. Chức năng xem lịch sử các sự kiện đã đăng ký

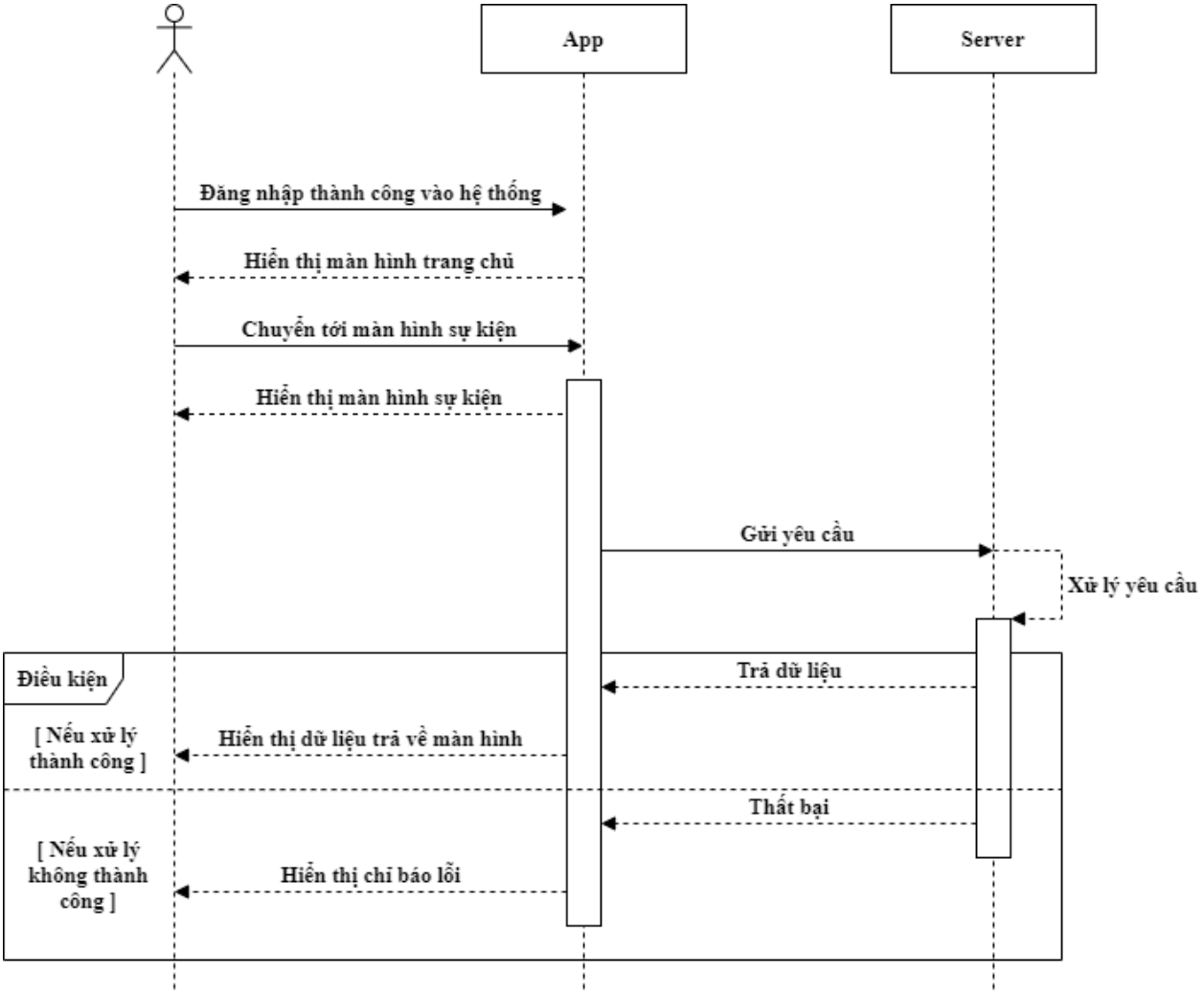
- **Mô tả:** Sau khi đăng nhập, người dùng có thể xem những sự kiện đã kết thúc mà họ đã đăng ký tại màn hình lịch sử sự kiện đã đăng ký.
- **Sơ đồ tuần tự:**



Hình 6: Sơ đồ tuần tự chức năng xem lịch sử các sự kiện đã đăng ký

2.3.6. Chức năng xem danh sách các sự kiện đang có hiệu lực

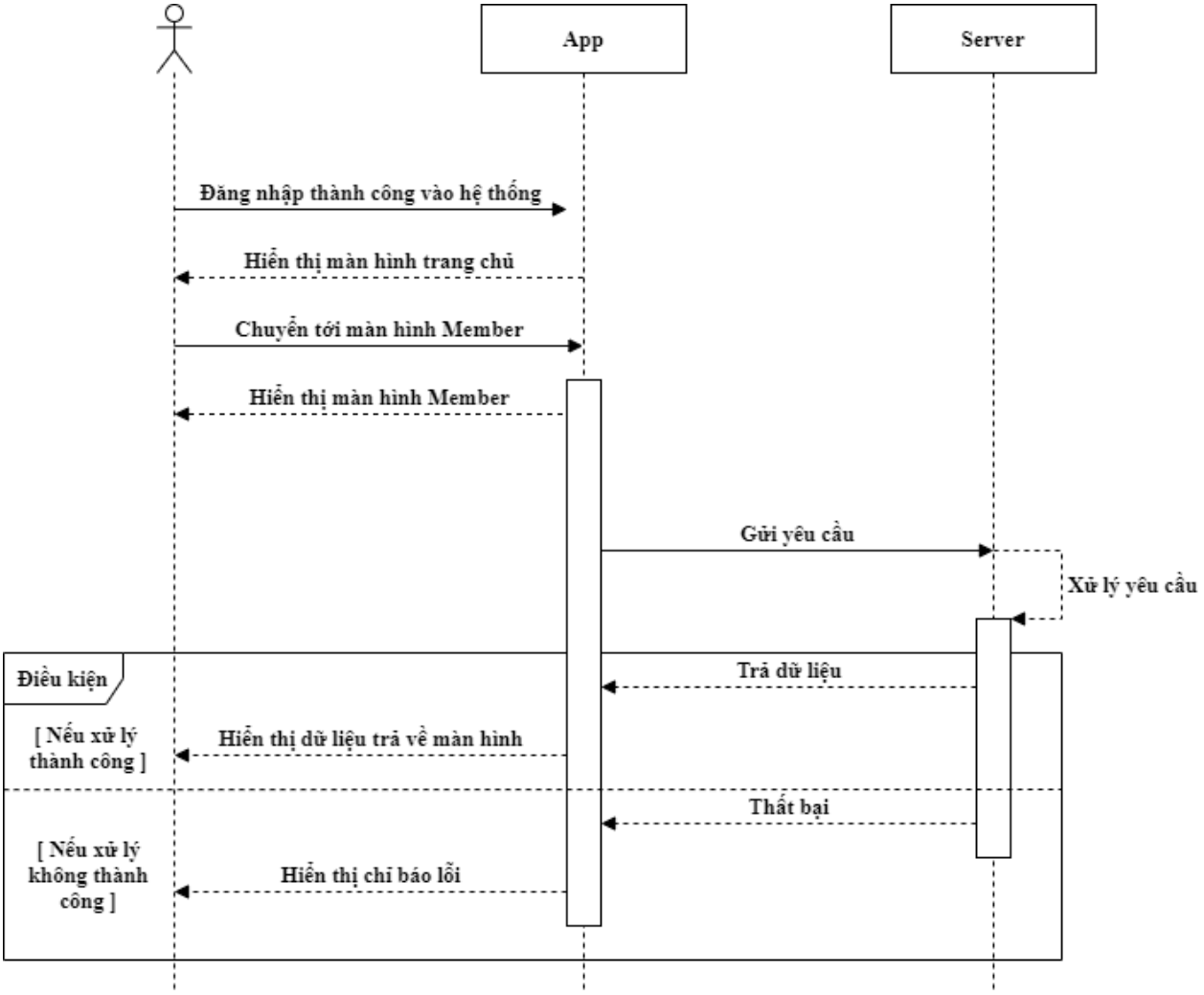
- **Mô tả:** Sau khi đăng nhập, người dùng có thể di chuyển tới màn hình danh sách sự kiện để có thể xem những sự kiện đang active.
- **Sơ đồ tuần tự:**



Hình 7: Sơ đồ tuần tự chức năng xem sự kiện đã đăng ký

2.3.6. Chức năng xem danh sách những người dùng trong hệ thống

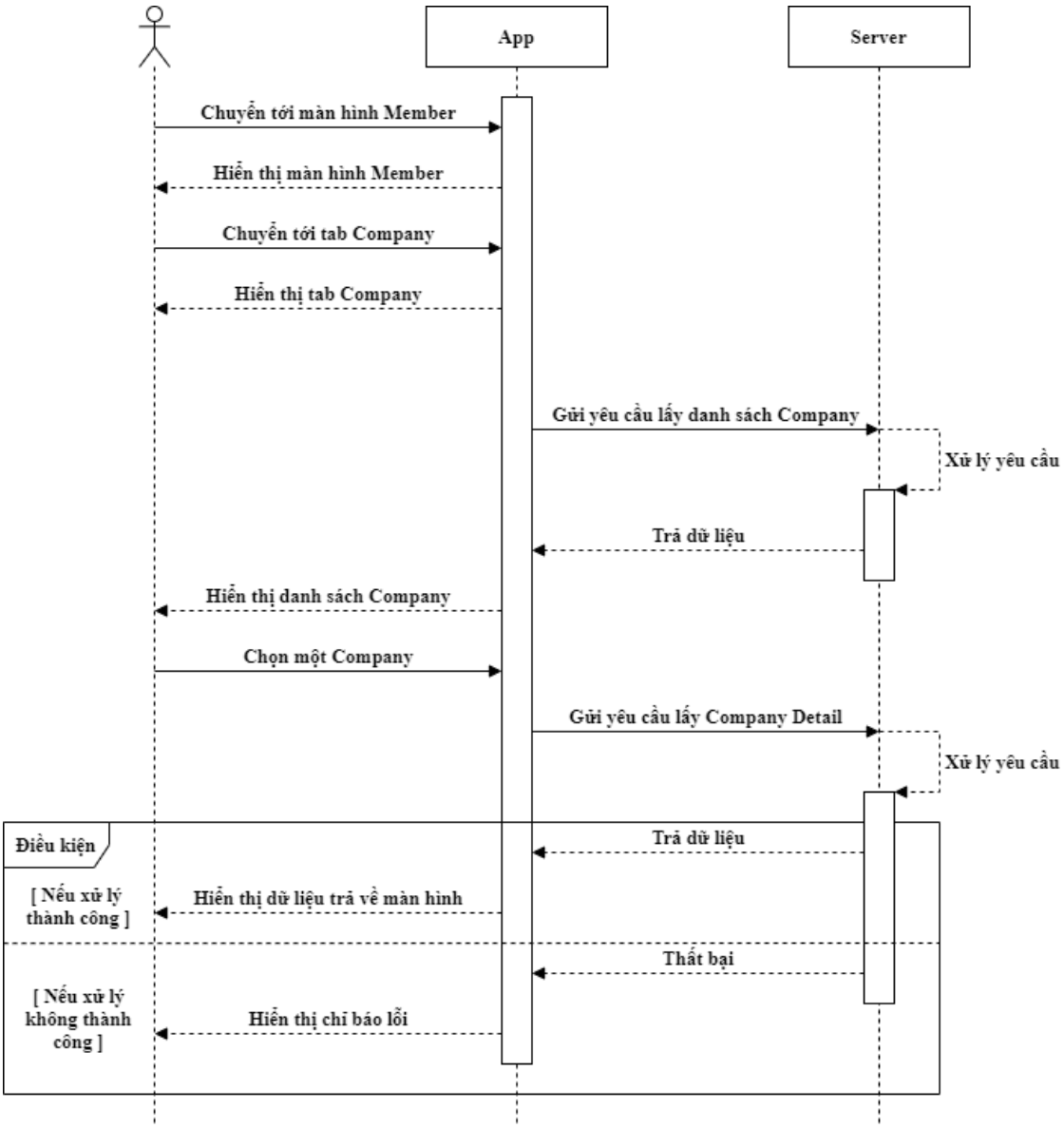
- **Mô tả:** Sau khi đăng nhập, người dùng có thể xem danh sách tất cả những người dùng trong hệ thống ở màn hình Member, người dùng có thể
- **Sơ đồ tuần tự:**



Hình 8: Sơ đồ tuần tự chức năng xem sự kiện đã đăng ký

2.3.7. Chức năng xem danh sách những doanh nghiệp trong hệ thống

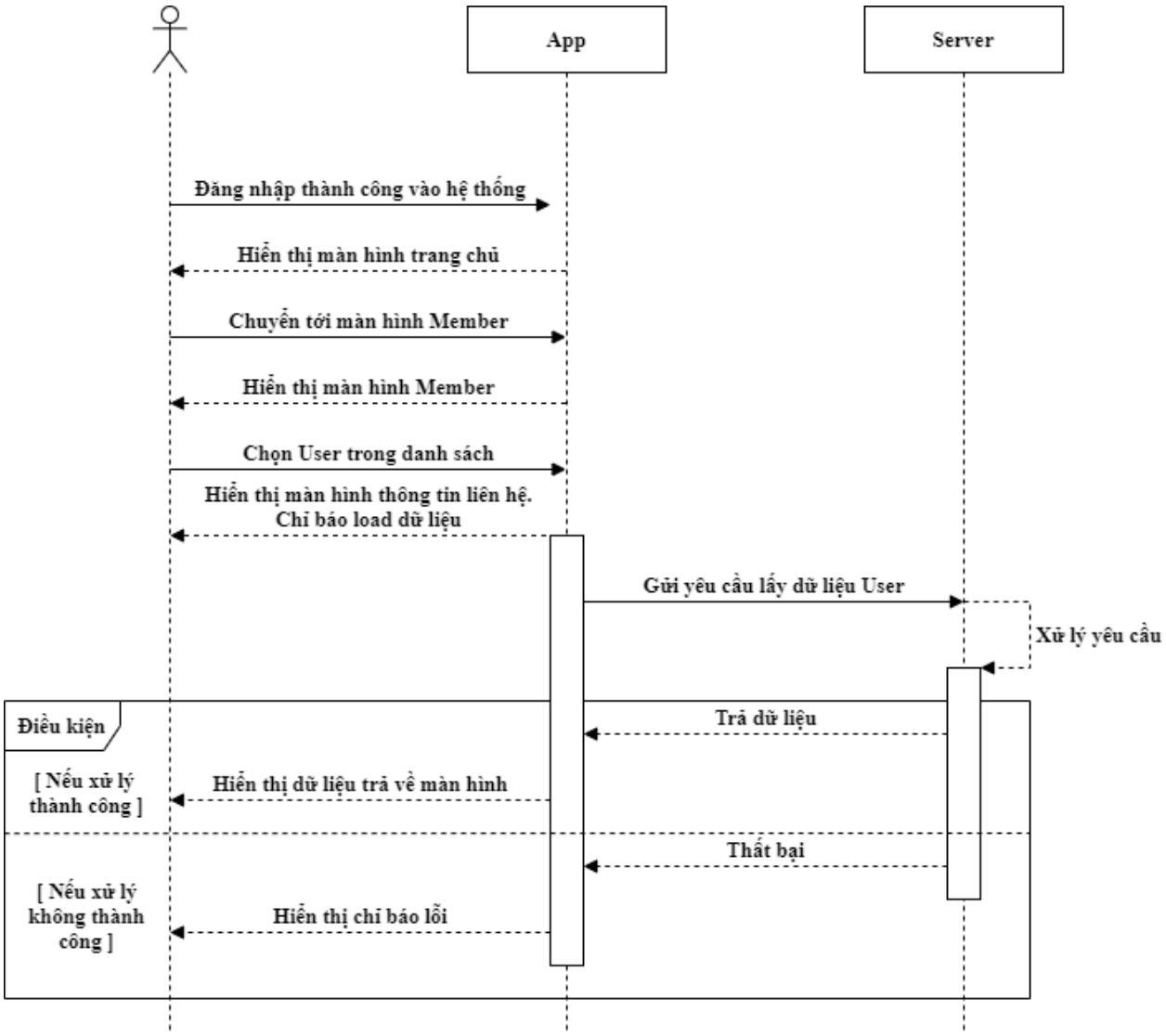
- **Mô tả:** Sau khi đăng nhập, người dùng có thể xem danh sách tất cả những doanh nghiệp trong hệ thống ở màn hình Member ở Tab Company, người dùng có thể
- **Sơ đồ tuần tự:**



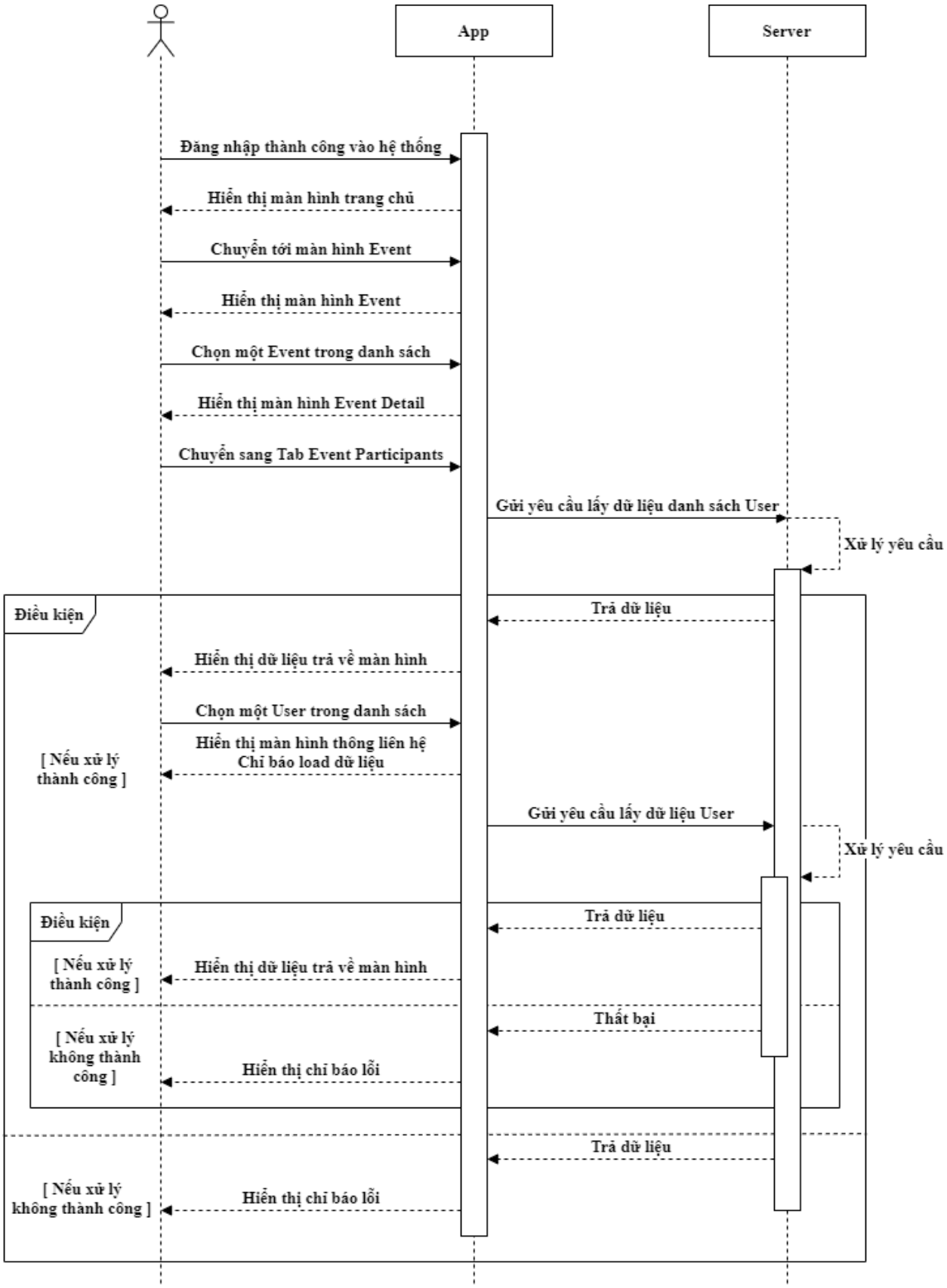
Hình 9: Sơ đồ tuần tự chức năng xem sự kiện đã đăng ký

2.3.8. Chức năng xem thông tin liên hệ của người dùng

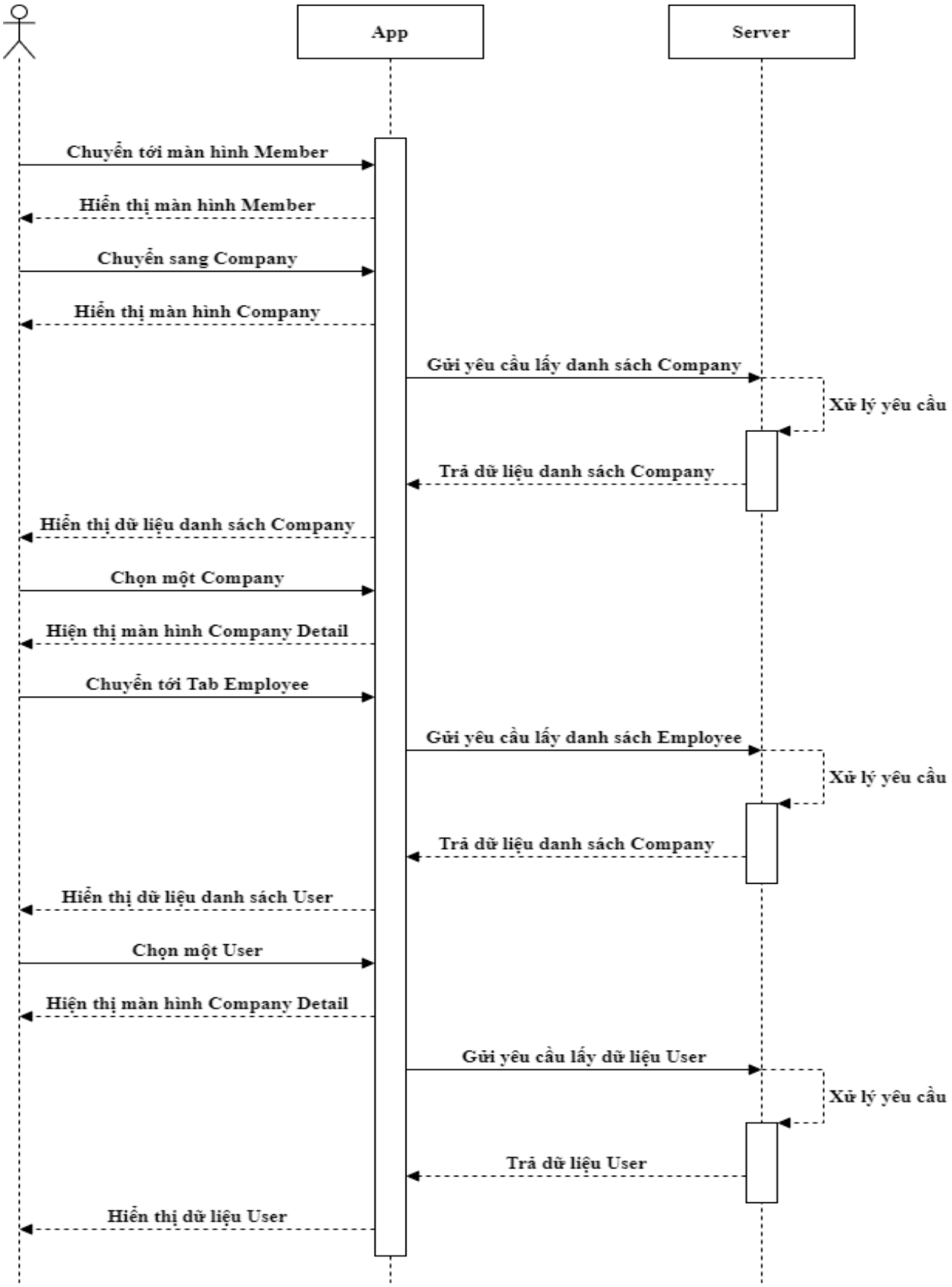
- **Mô tả:** Người dùng có thể xem thông tin liên hệ của người dùng nếu bấm vào item trong danh sách user *(Member screen, Event Participant tab của Event Detail Screen và employee tab của Company Detail Screen)*
- **Sơ đồ tuần tự:**



Hình 10: Sơ đồ tuần tự chức năng xem thông tin liên hệ của người dùng (Member Screen)



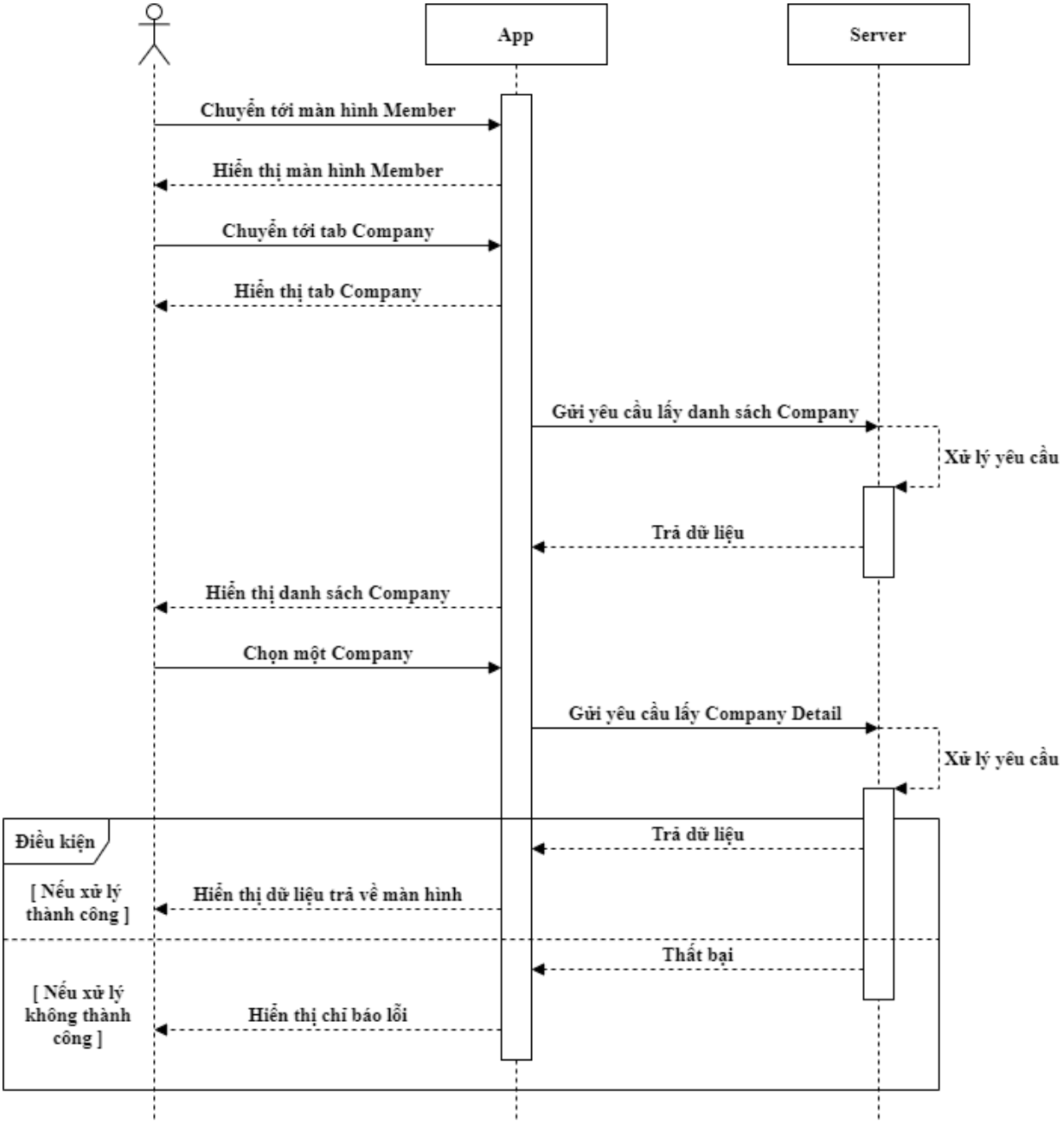
Hình 11: Sơ đồ tuần tự chức năng xem thông tin liên hệ của người dùng (Event Participants Screen)



Hình 12: Sơ đồ tuần tự chức năng xem thông tin liên hệ của người dùng (Company Employee Screen)

2.3.9. Chức năng xem thông tin liên hệ của công ty

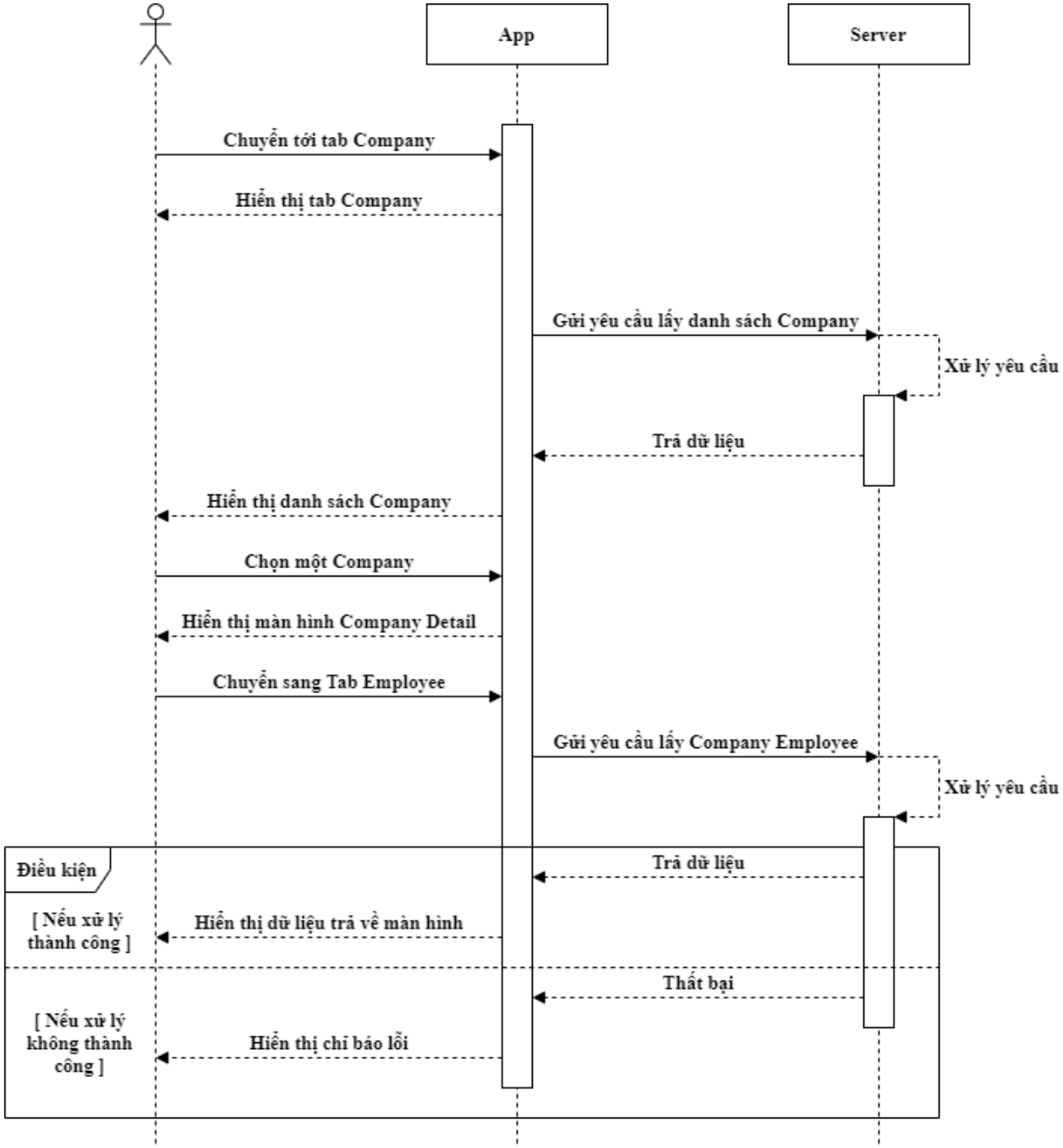
- **Mô tả:** Người dùng có thể xem thông tin liên hệ của công ty khi chọn một công tin ở màn hình Company
- **Sơ đồ tuần tự:**



Hình 13: Sơ đồ tuần tự chức năng xem thông tin liên hệ của doanh nghiệp

2.3.10. Chức năng xem danh sách nhân viên của công ty

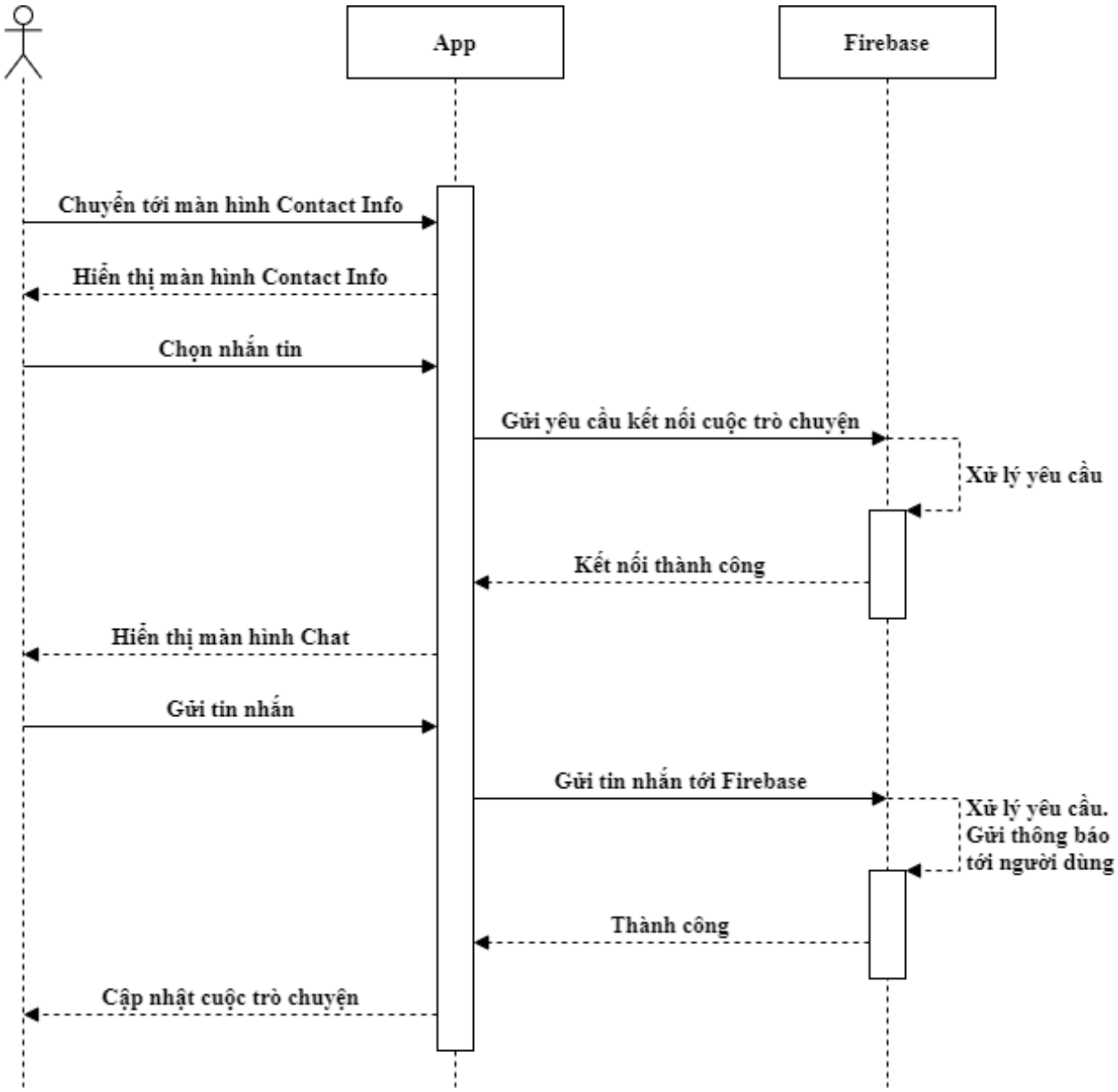
- **Mô tả:** Người dùng có thể xem danh sách nhân viên của công ty khi chọn một công tin ở màn hình Company Employee
- **Sơ đồ tuần tự:**



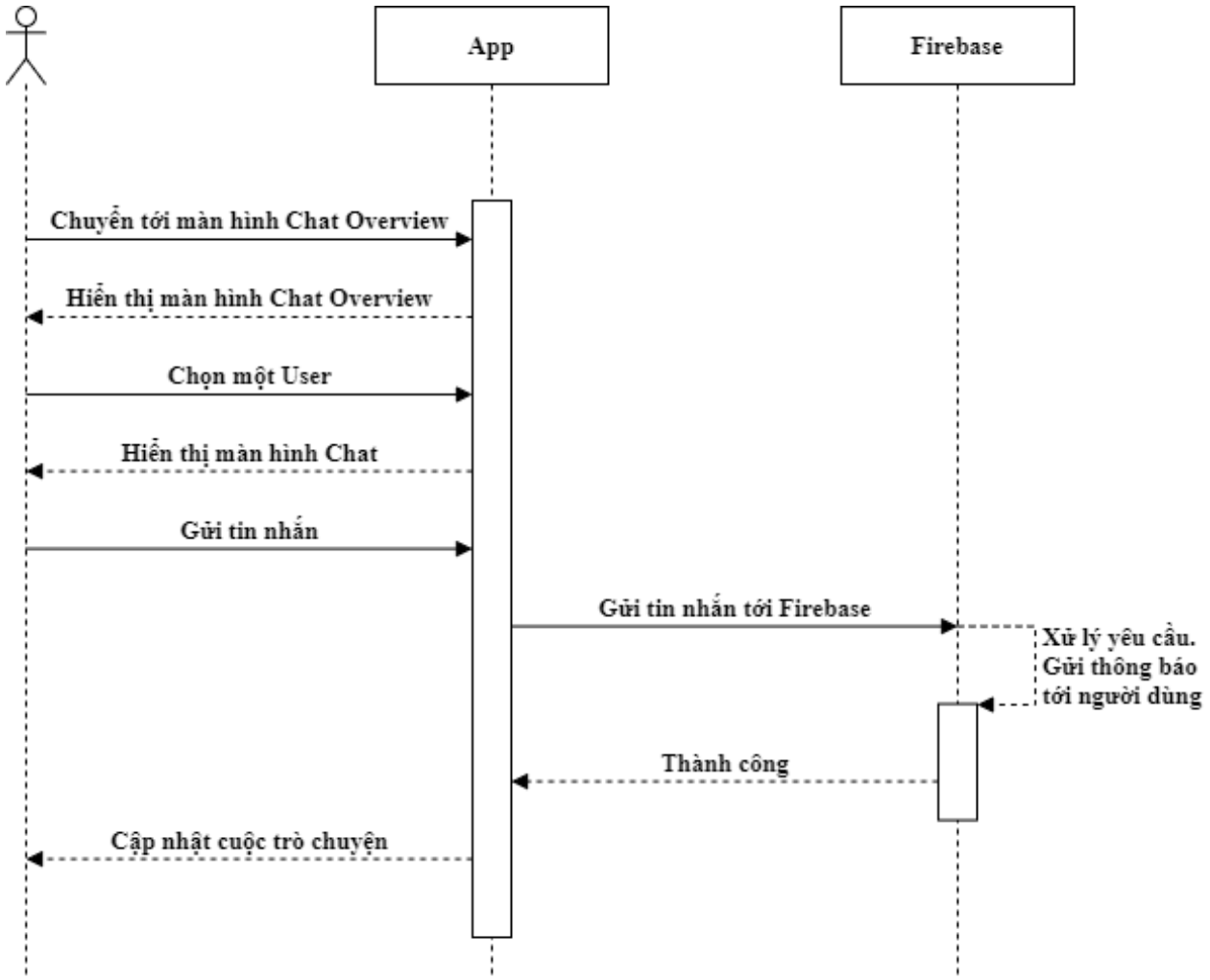
Hình 14: Sơ đồ tuần tự chức năng xem thông tin liên hệ của doanh nghiệp

2.3.11. Chức năng nhắn tin với người dùng

- **Mô tả:** Người dùng có thể nhắn tin với một người dùng khác bằng 2 cách, khi họ chưa có có nhóm trò chuyện, người dùng có thể bắt chuyện bằng cách tìm tới thông tin liên lệ của họ và chọn nhắn tin. Nếu họ đã trò chuyện, người dùng có thể nhắn tin nhanh với họ ở màn hình Chat Overview.
- **Sơ đồ tuần tự:**



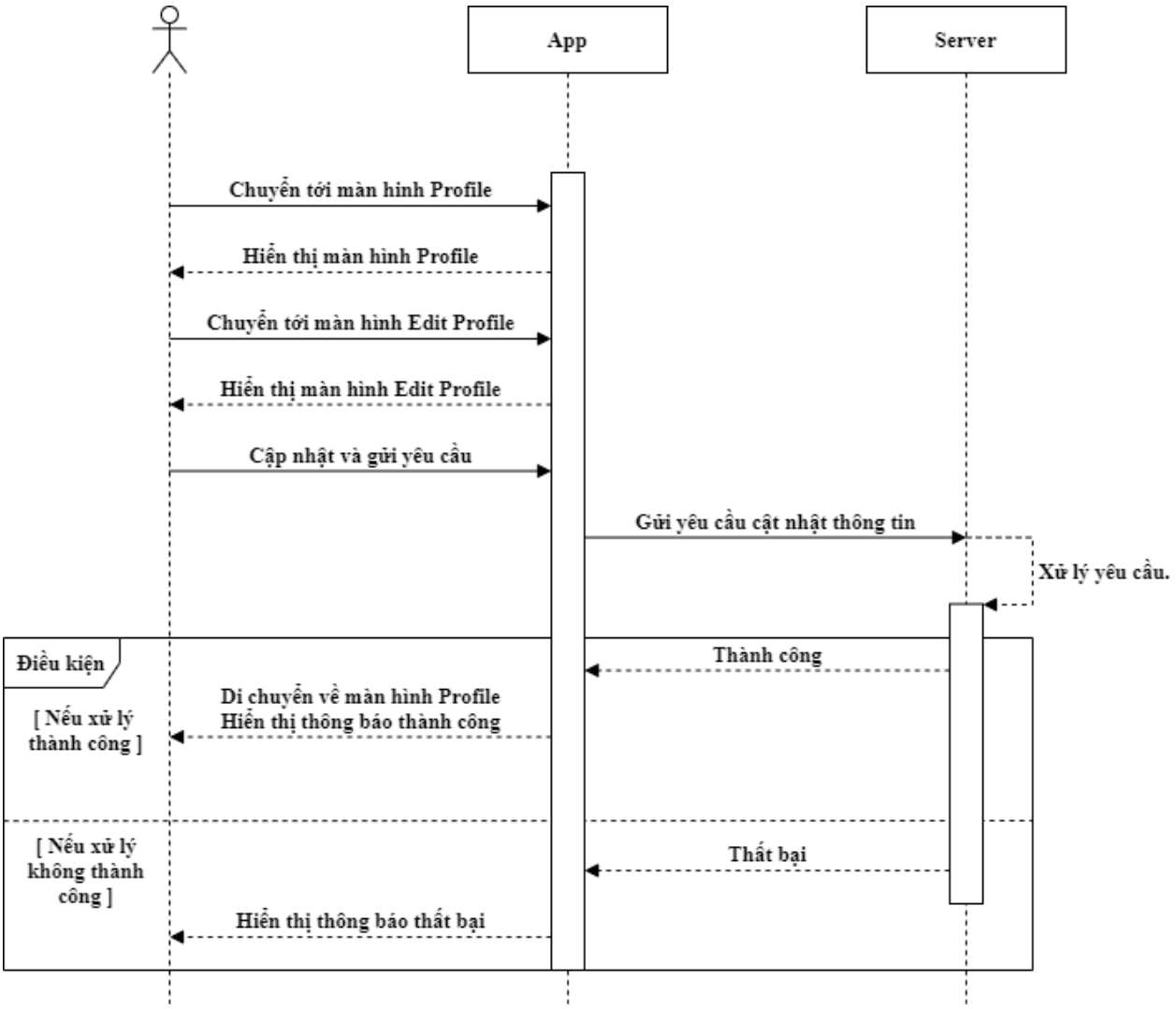
Hình 15: Sơ đồ tuần tự chức năng nhắn tin (Contact Info)



Hình 16: Sơ đồ tuần tự chức năng nhắn tin (Chat Overview)

2.3.12. Chức năng cập nhật thông tin cá nhân

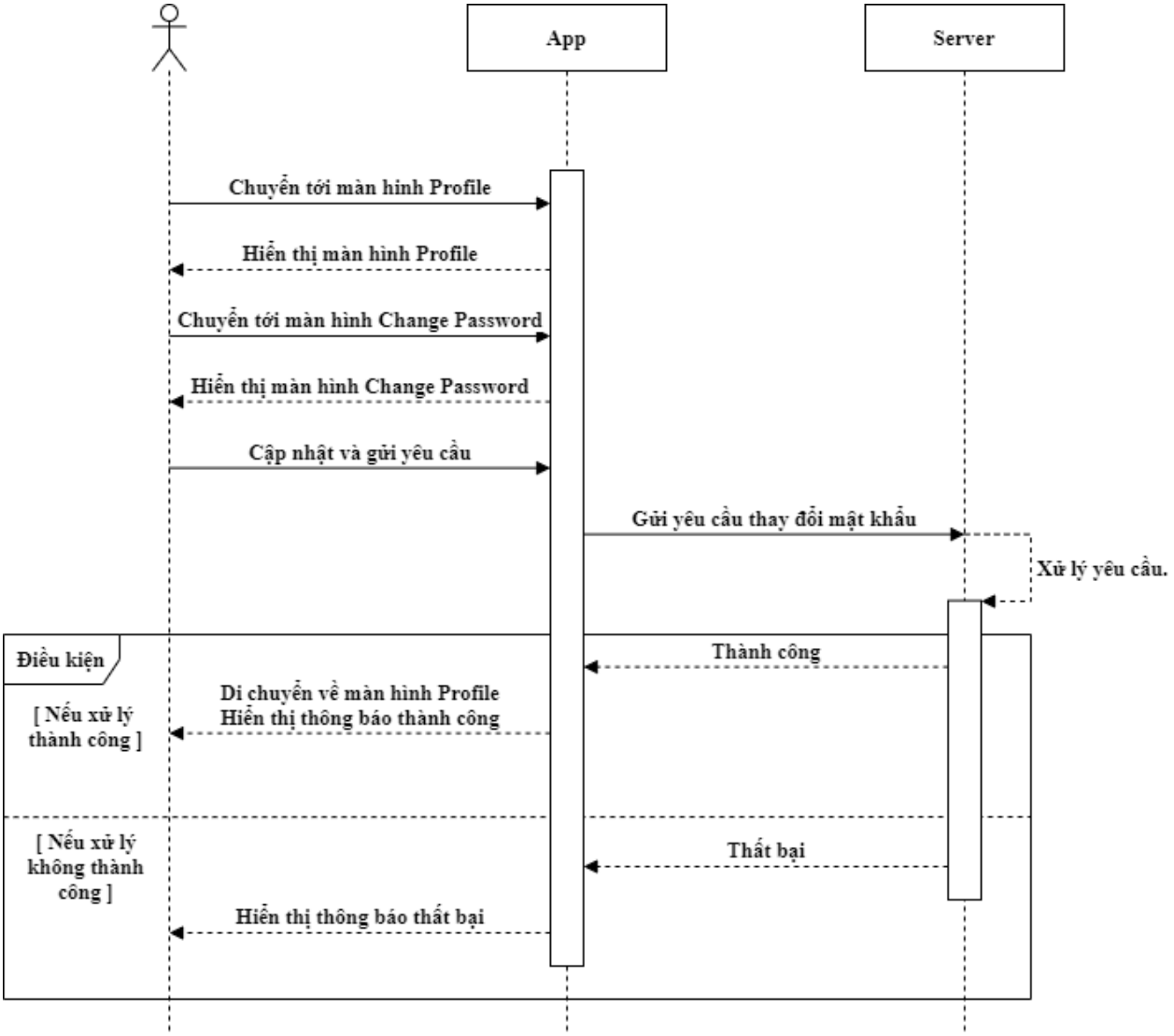
- **Mô tả:** Người dùng có thể cập nhật thông tin cá nhân.
- **Sơ đồ tuần tự:**



Hình 17: Sơ đồ tuần tự chức năng cập nhật thông tin cá nhân

2.3.13. Chức năng thay đổi mật khẩu

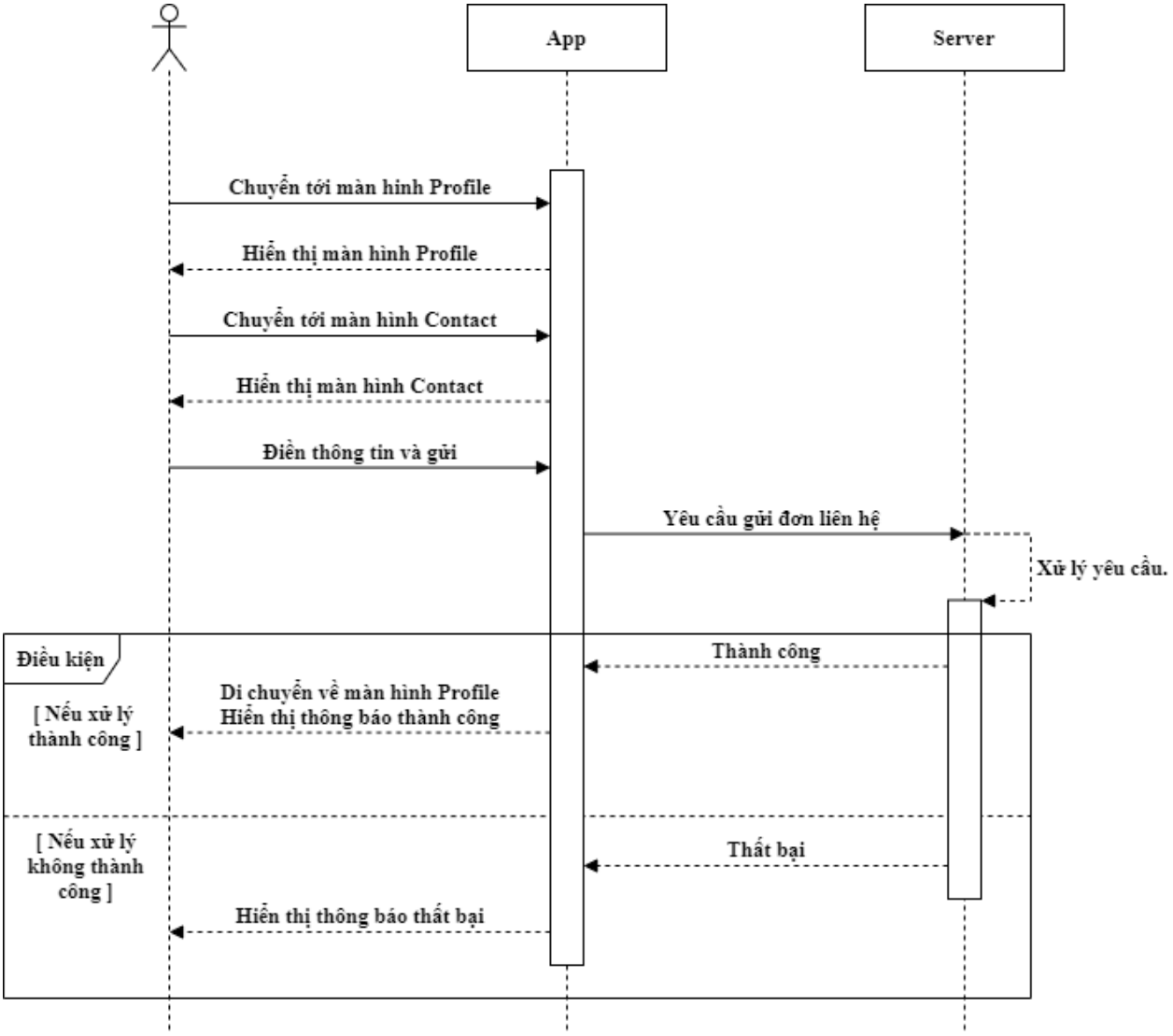
- **Mô tả:** Người dùng có thể thay đổi mật khẩu
- **Sơ đồ tuần tự:**



Hình 18: Sơ đồ tuần tự chức năng thay đổi mật khẩu

2.3.14. Chức năng gửi đơn hỗ trợ

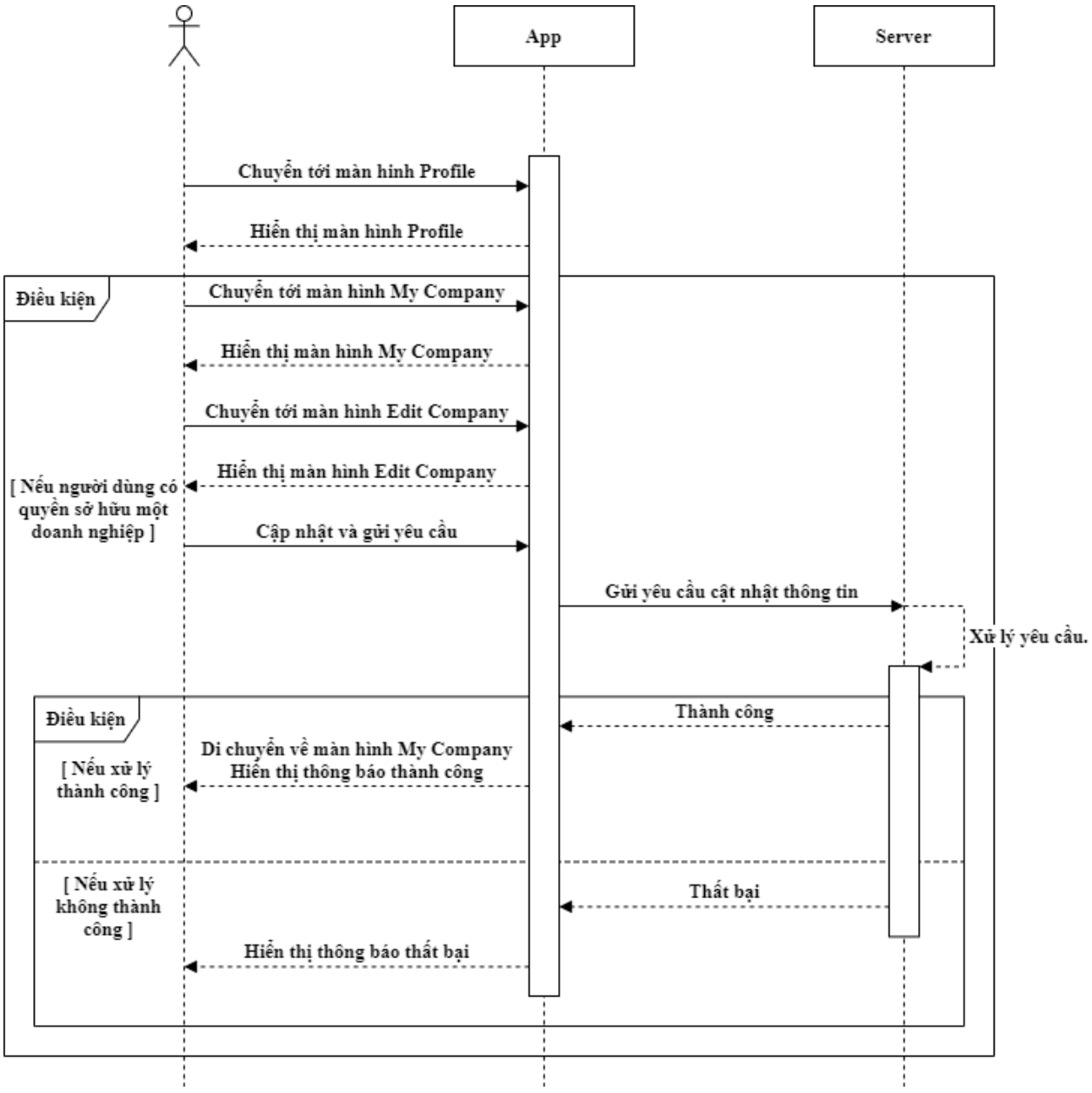
- **Mô tả:** Người dùng có thể gửi đơn liên hệ tới nhà phát triển để nhận được sự hỗ trợ.
- **Sơ đồ tuần tự:**



Hình 19: Sơ đồ tuần tự chức năng thay đổi mật khẩu

2.3.15. Chức năng cập nhật thông tin doanh nghiệp

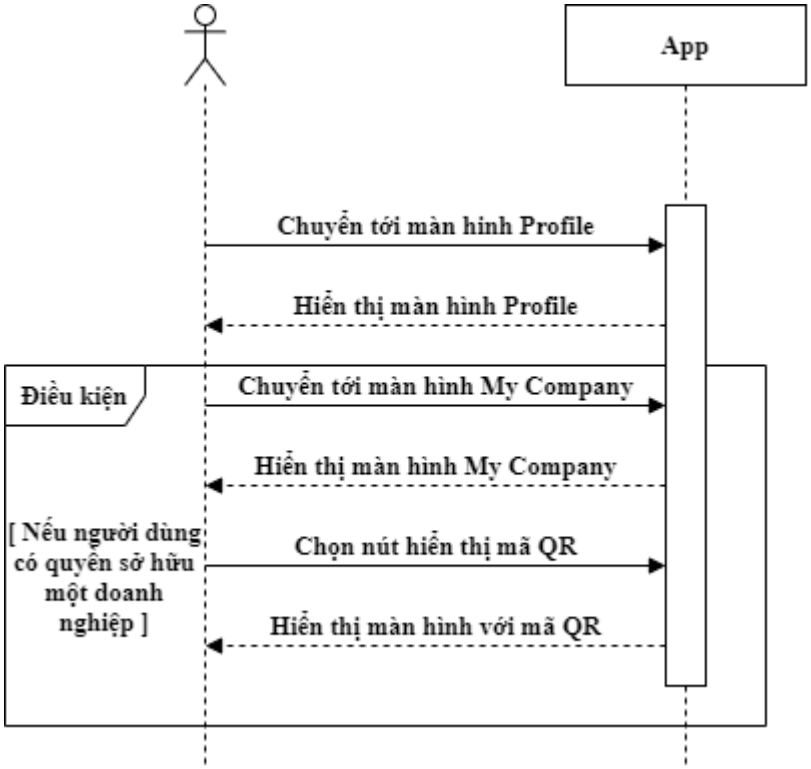
- **Mô tả:** Khi người dùng sở hữu một doanh nghiệp trên hệ thống của BKL, người dùng có quyền được thay đổi những thông tin liên hệ và những thông tin cơ bản của doanh nghiệp.
- **Sơ đồ tuần tự:**



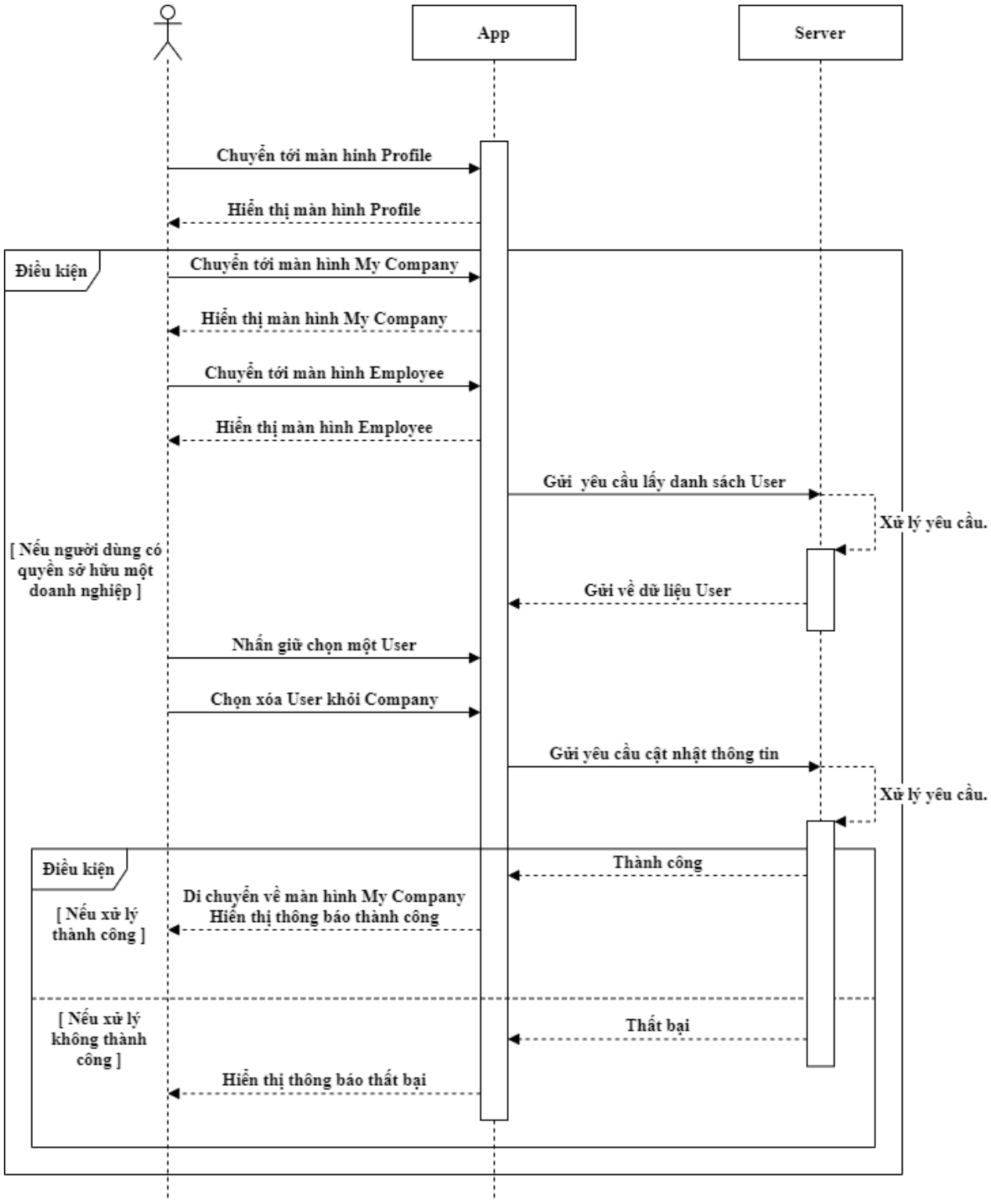
Hình 20: Sơ đồ tuần tự chức năng cập nhật thông tin doanh nghiệp

2.3.16. Chức năng quản lý nhân viên trong doanh nghiệp

- **Mô tả:** Người chủ doanh nghiệp có thể thêm nhân viên bằng cách hiển thị một mã QR cho người dùng khác để họ dùng để quét khi đăng ký tài khoản. Sau khi đang ký họ sẽ là nhân viên của doanh nghiệp. Người dùng cũng có thể xóa nhân viên ra khỏi doanh nghiệp
- **Sơ đồ tuần tự:**



Hình 21: Sơ đồ tuần tự chức năng hiển thị QR đăng ký

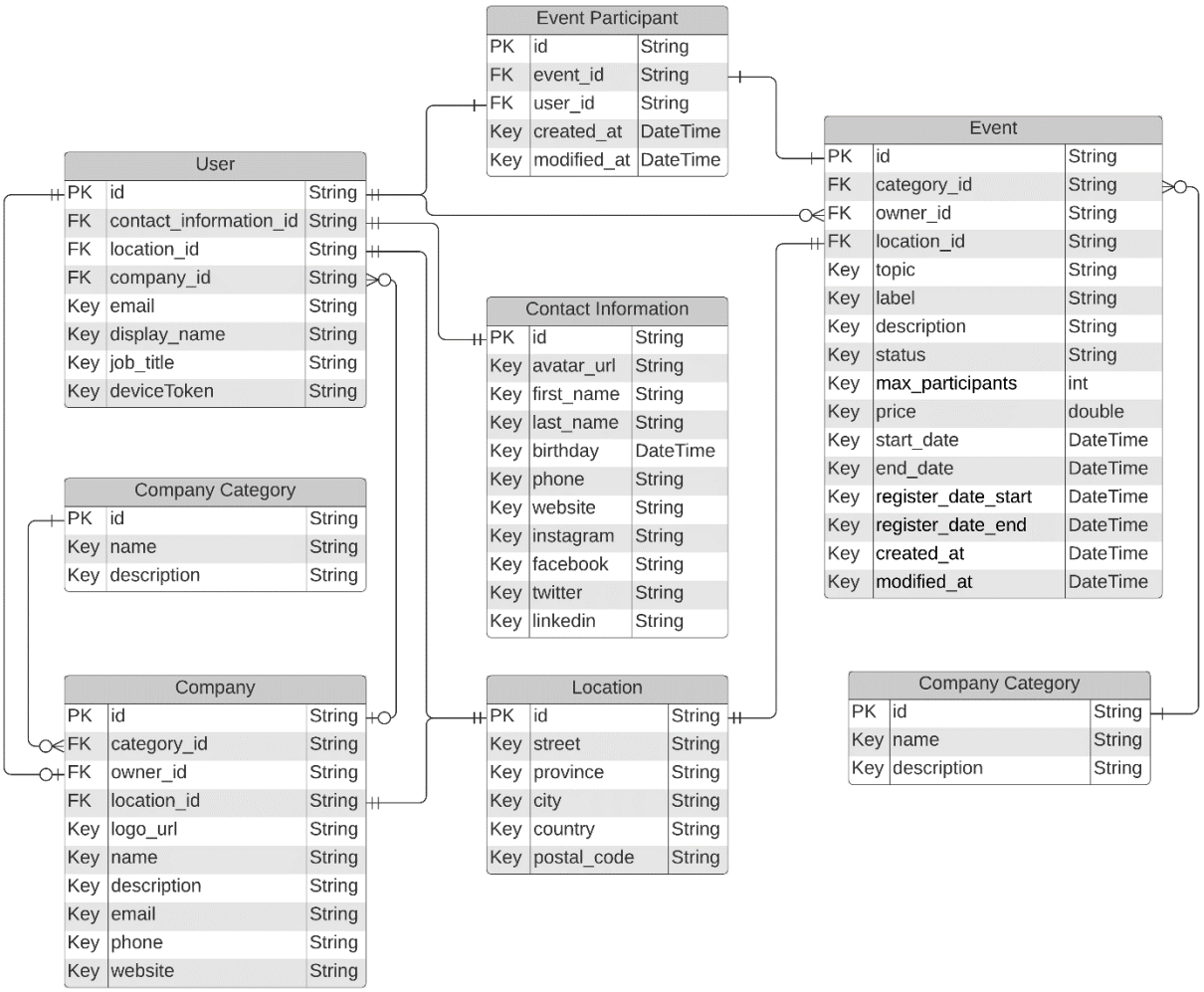


Hình 22: Sơ đồ tuần tự chức năng xóa nhân viên khỏi doanh nghiệp

2.4. Phân tích cơ sở dữ liệu

Cơ sở dữ liệu đã được phân tích về thiết kế bởi đội ngũ Toomba, trong quá trình tiếp quản dự án BKL, và họ không cấp quyền truy cấp vào Database vì lý do bảo mật. Do vậy, giao tiếp chính giữa bộ phận làm Mobile với Back-end là qua một API Document đã được viết sẵn.

Qua các API docs được cung cấp, ta có thể phân tích ngược lại thành một cấu trúc dữ liệu như sau:

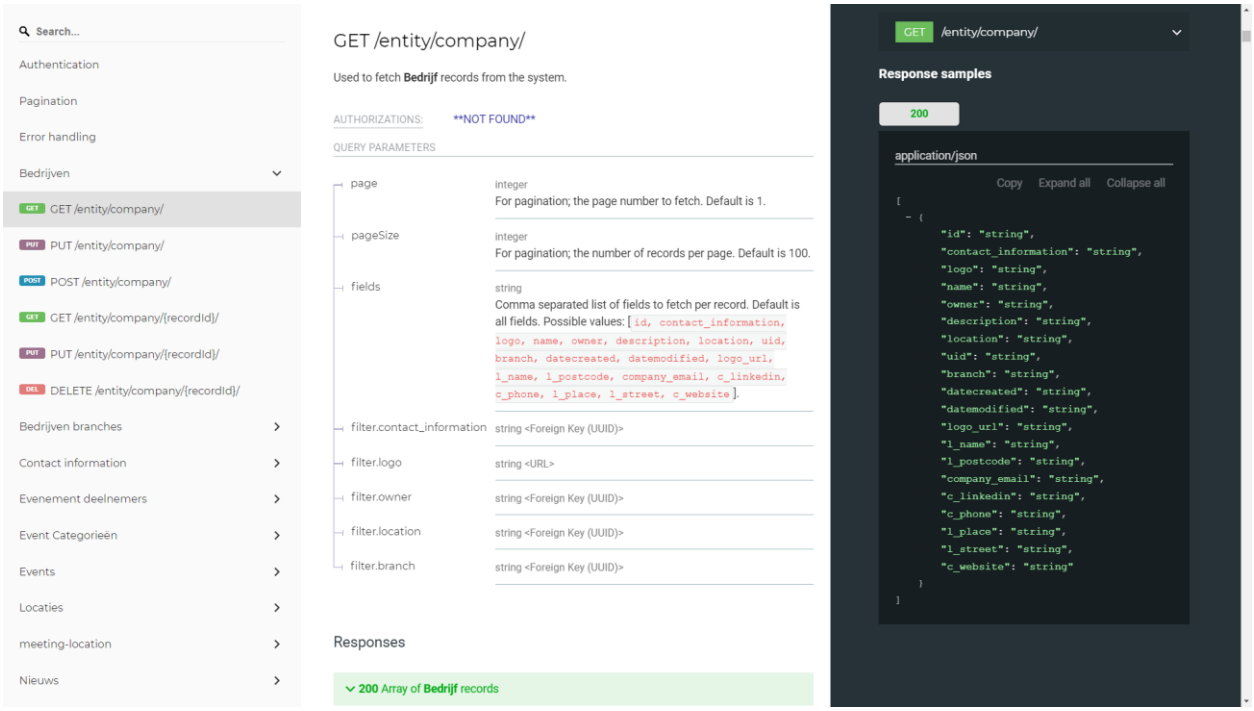


Hình 23: Mô hình ER của hệ thống

Những API được sử dụng chủ yếu trong ứng dụng gồm những API sau:

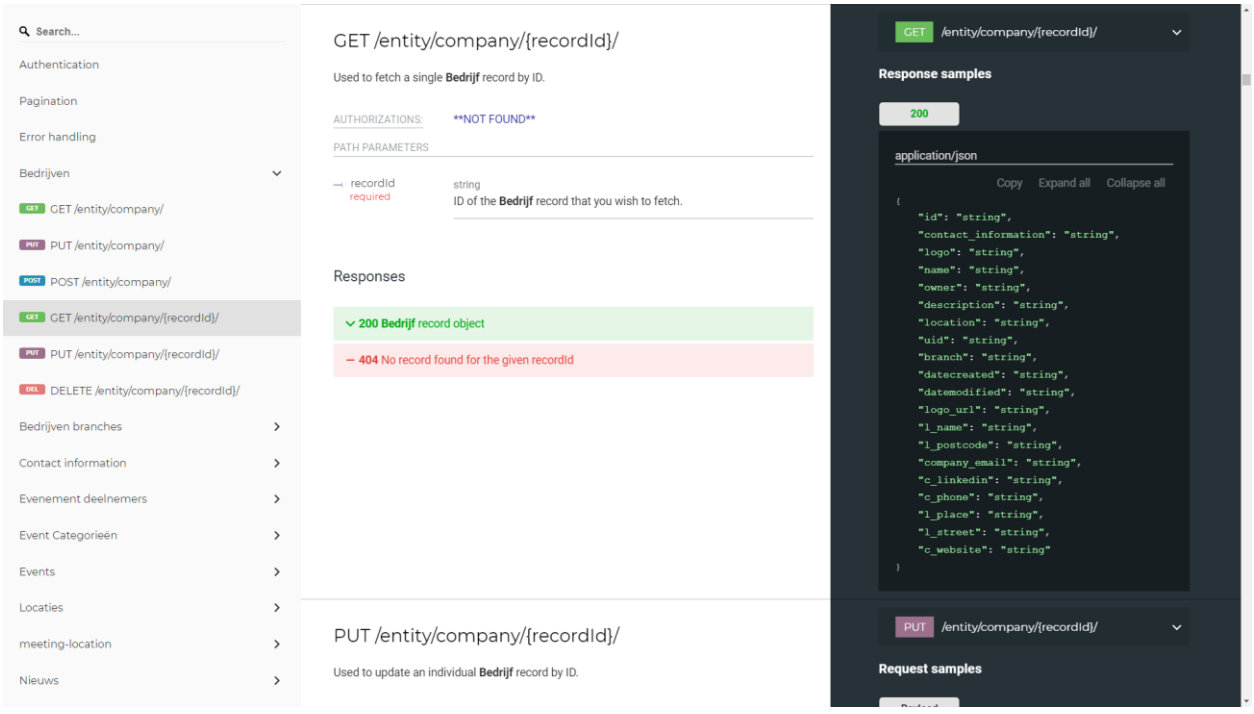
2.4.1. Company API

- Lấy danh sách các Comapny



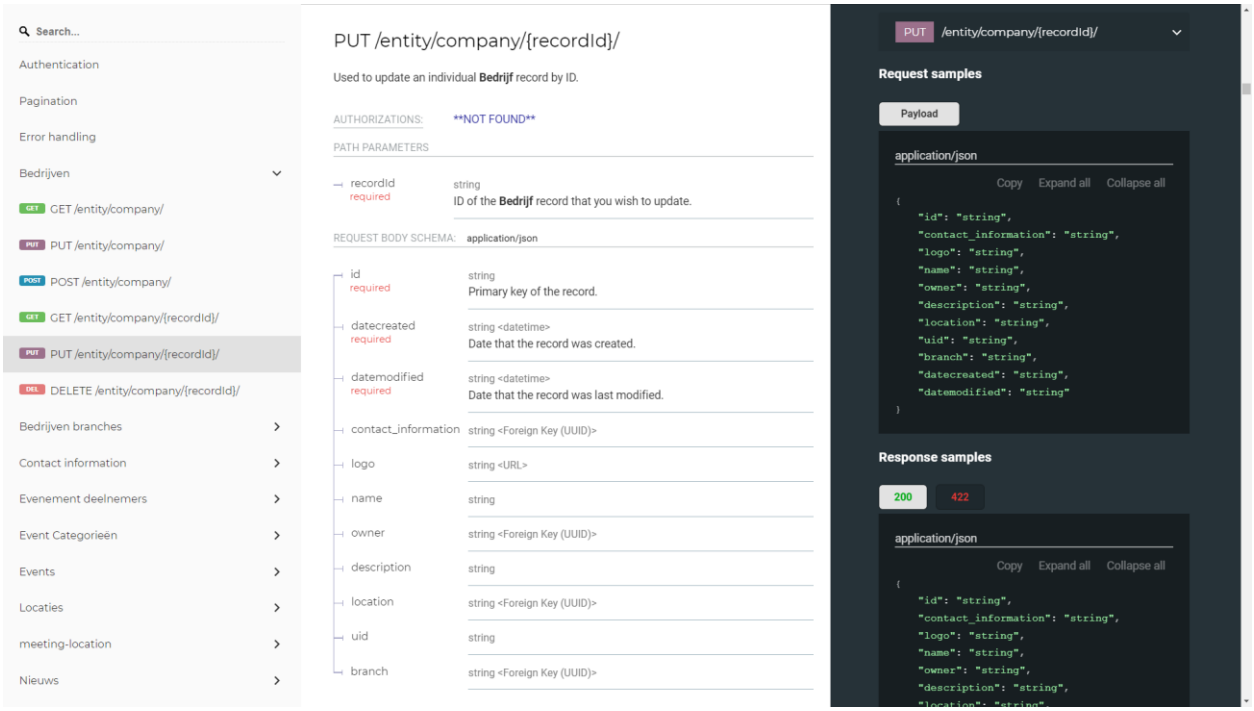
Hình 24: API lấy danh sách Company

- Lấy một Company dựa theo ID



Hình 25: API lấy Copany dựa theo ID

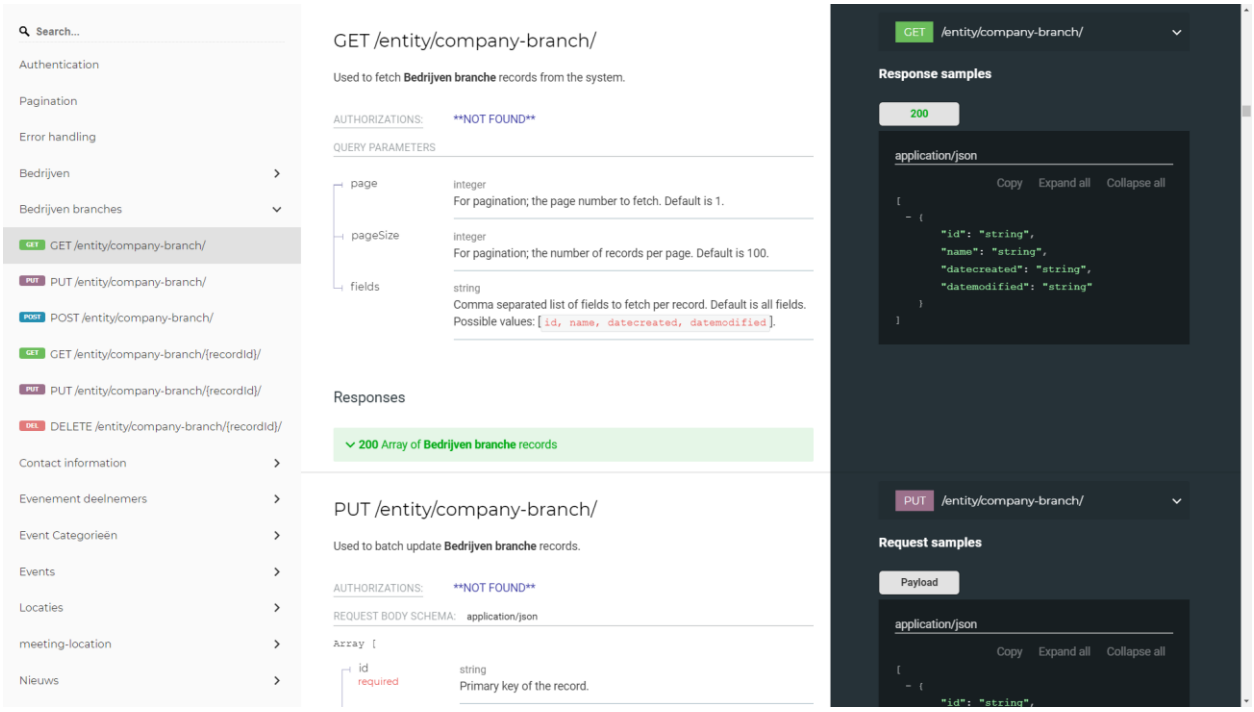
- Cập nhật một Company theo ID



Hình 26: API cập nhật Company theo ID

2.4.2. Company Branch API

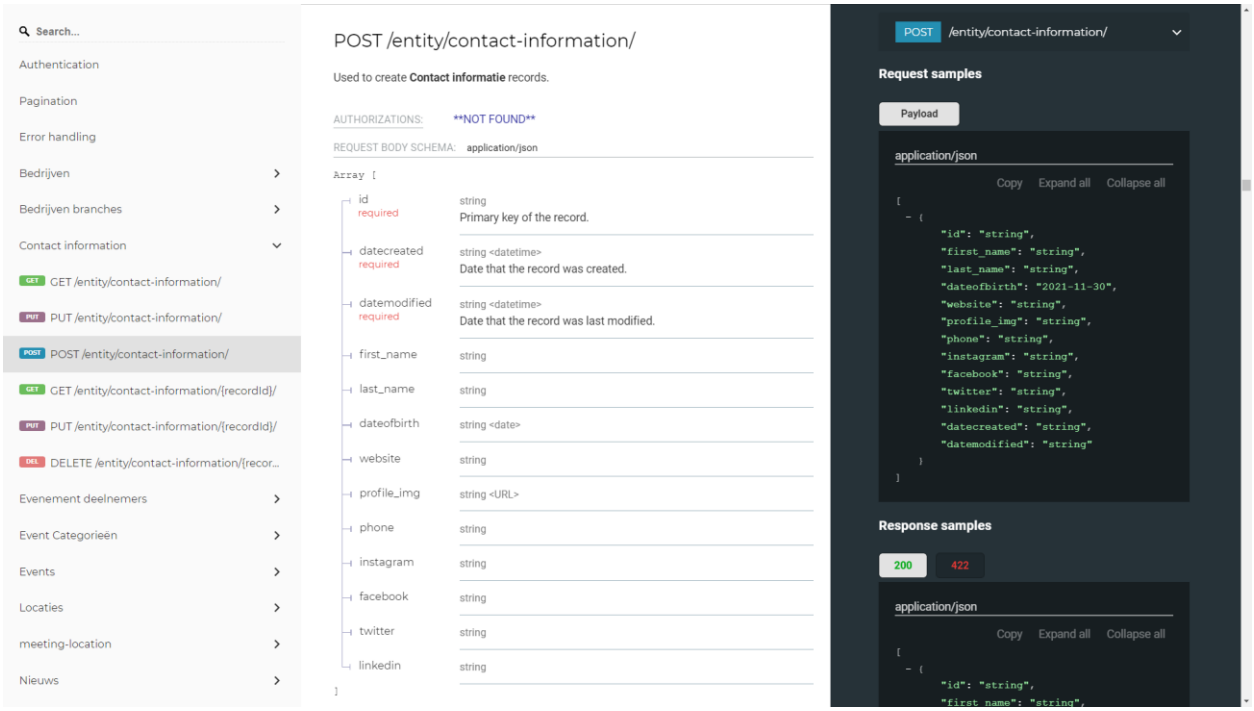
- Lấy danh sách các Company Branch



Hình 27: API lấy danh sách Company Branch

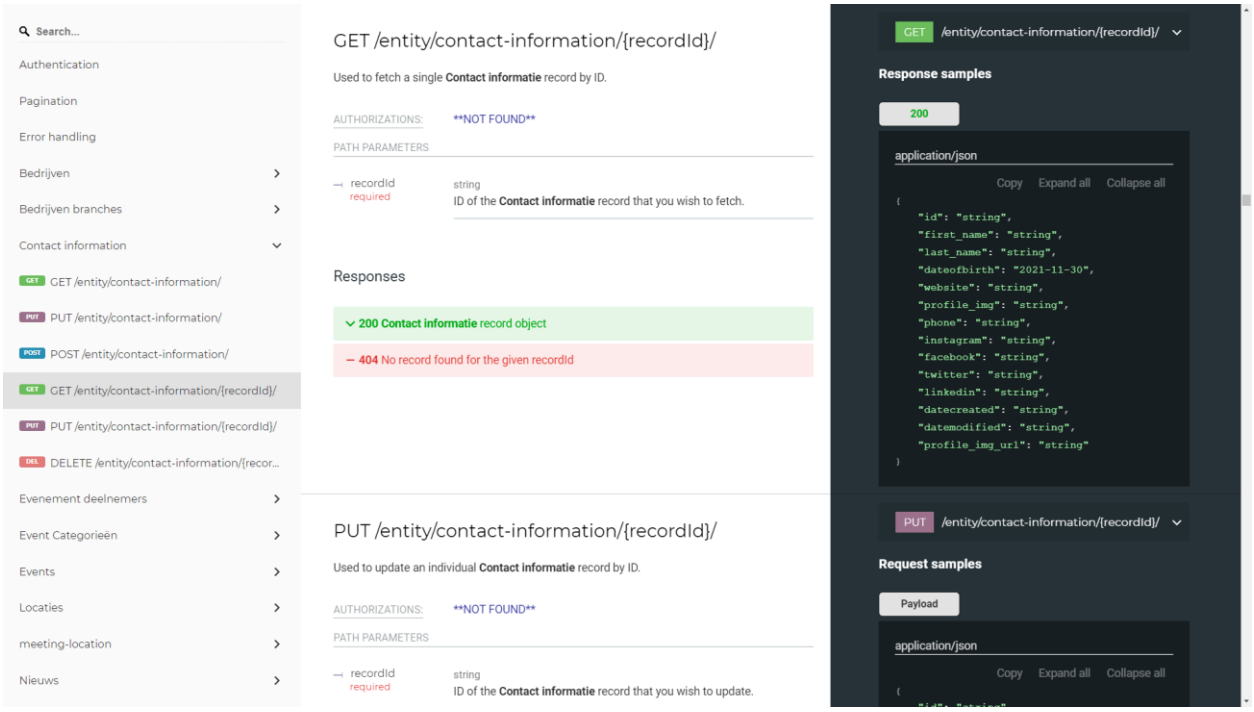
2.4.3. Contact Information API

- Tạo một Contact Information



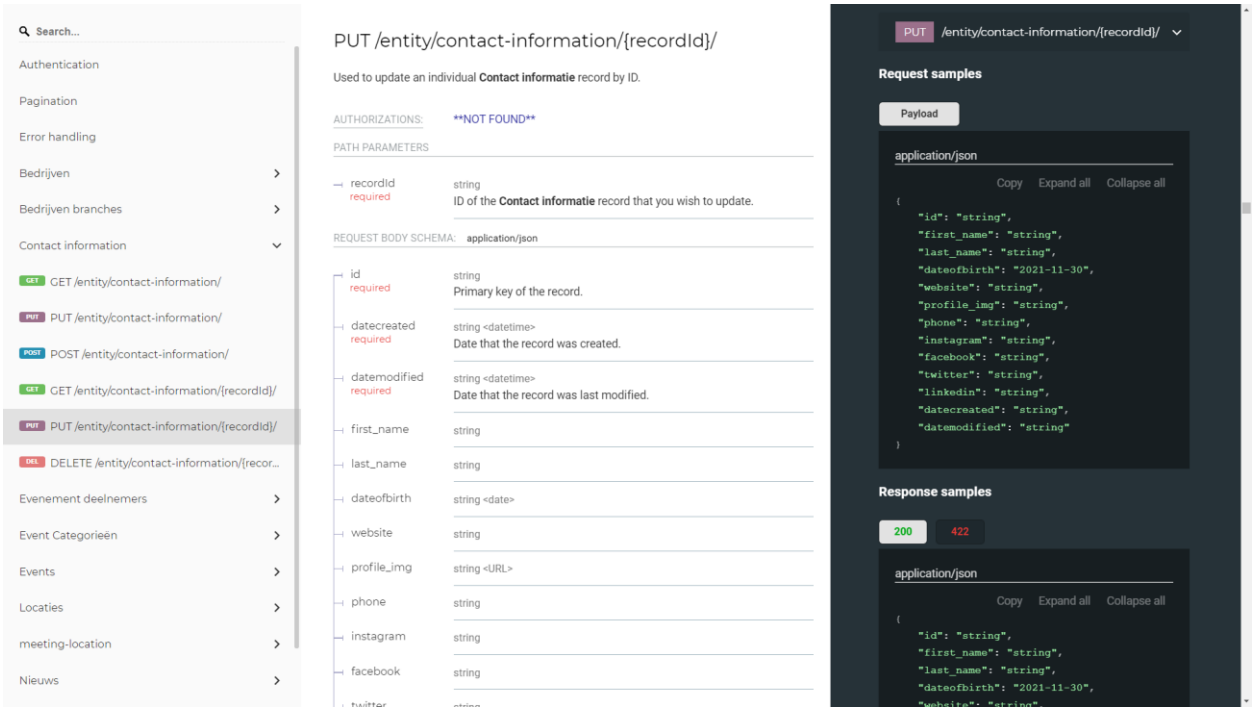
Hình 28: API tạo một Contact Information

- Lấy một Contact Information dựa theo ID



Hình 29: API lấy Contact Information theo ID

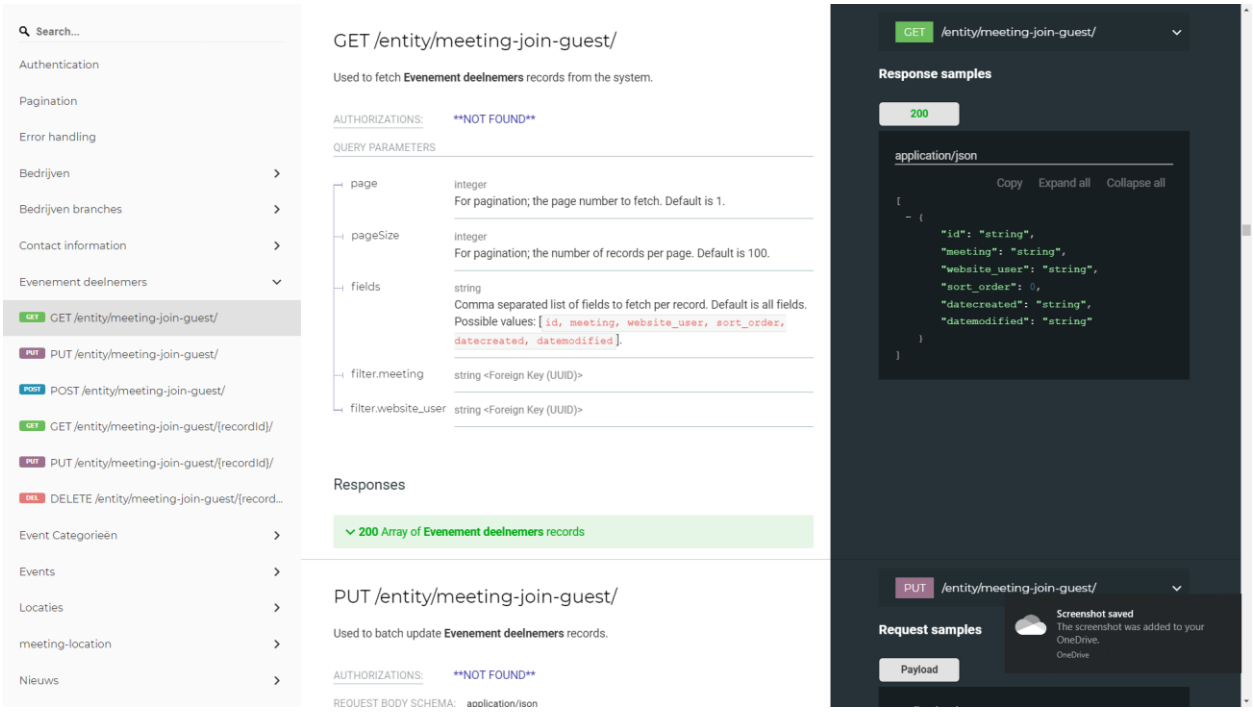
- Cập nhật một Contact Information theo ID



Hình 30: API cập nhật một Contaction Information theo ID

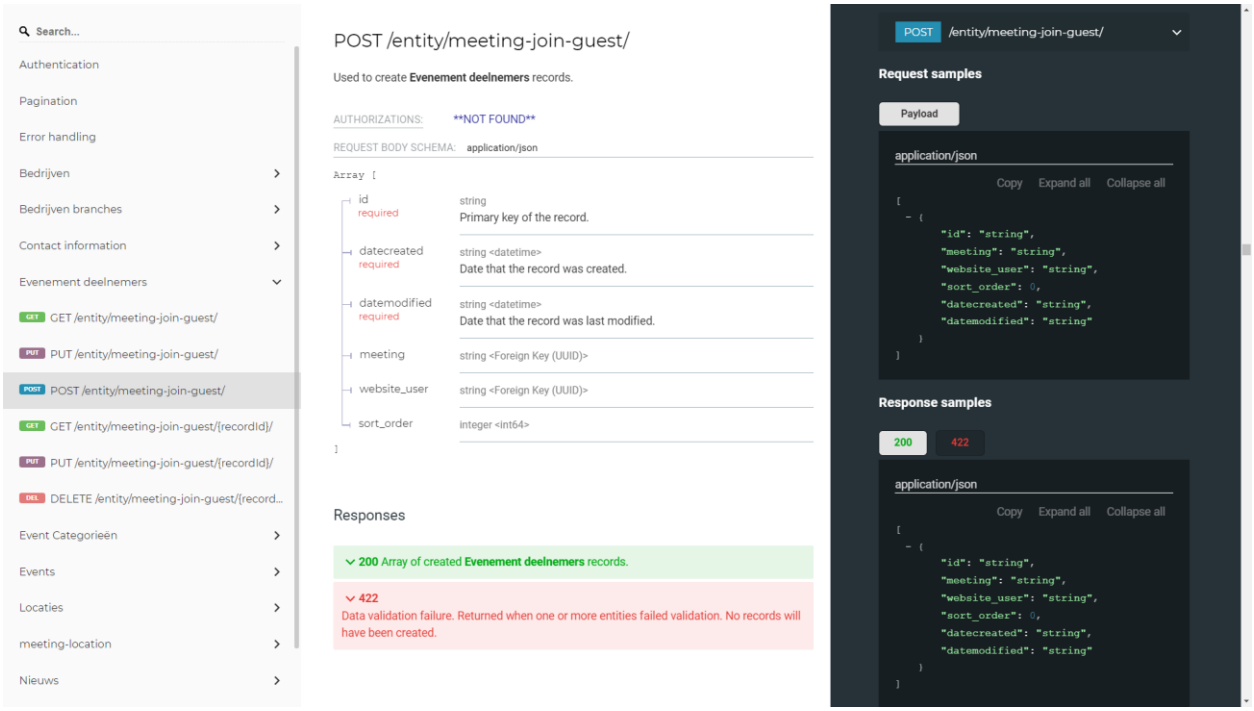
2.4.4. Event Participant API

- Lấy danh sách các Event Participant



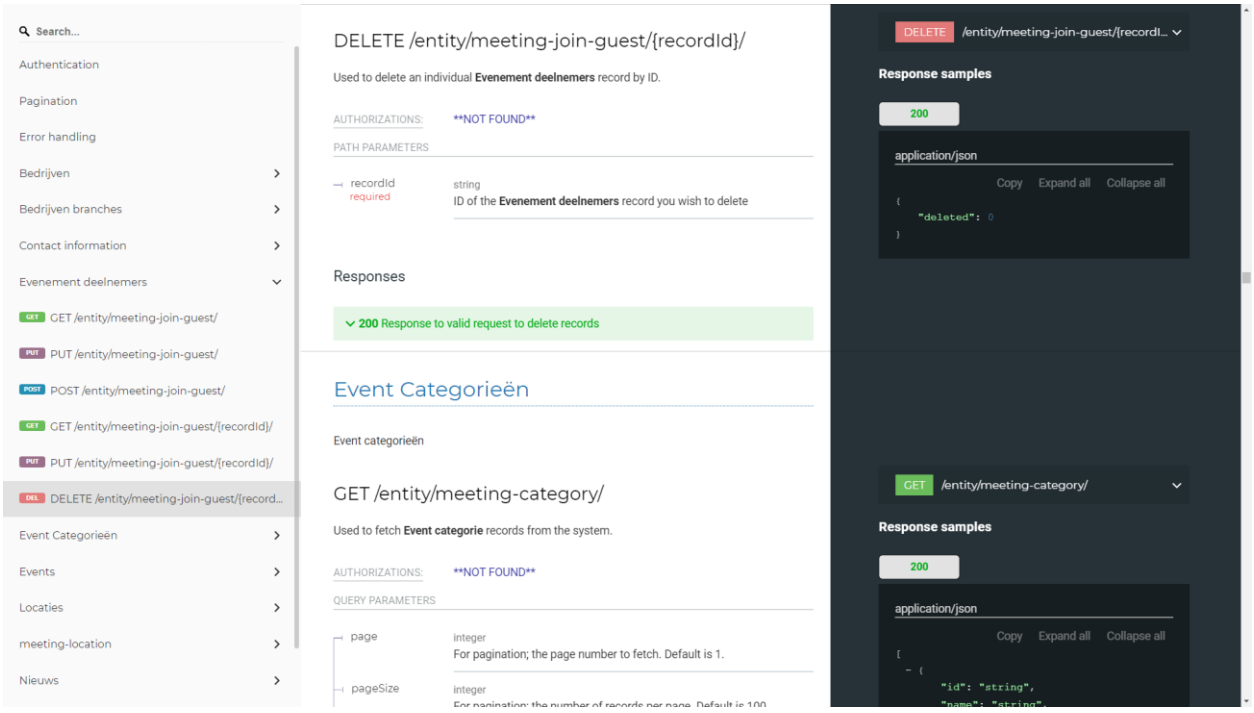
Hình 31: API lấy danh sách Event Participants

- Tạo một Event Participant



Hình 32: API tạo một Event Participant

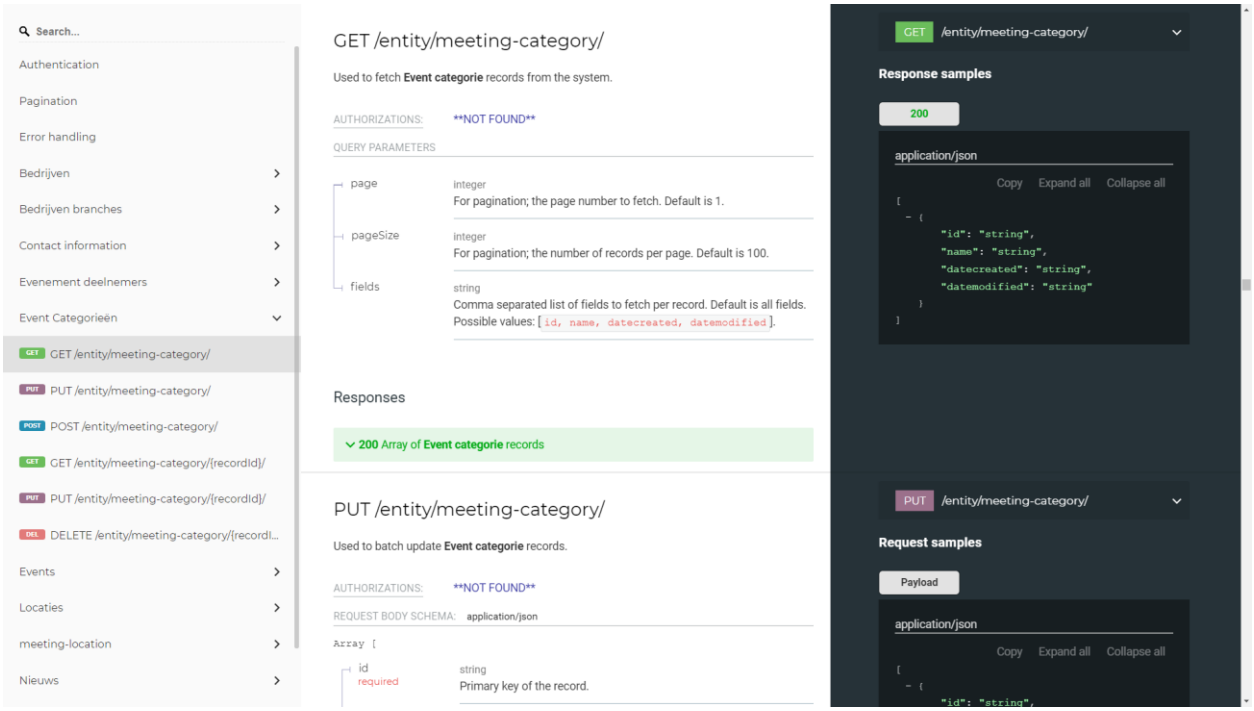
- Xóa một Event Participant theo ID



Hình 33: API xóa một Event Participant

2.4.5. Event Category API

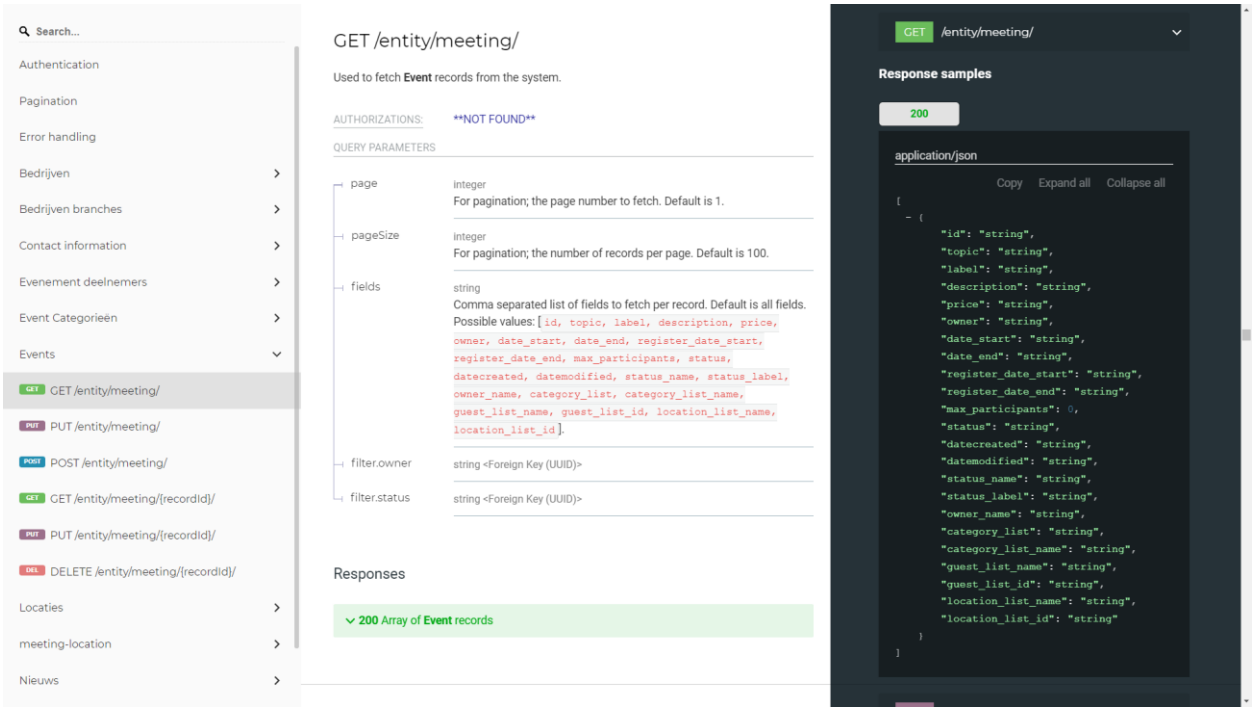
- Lấy danh sách Event Category



Hình 34: API lấy danh sách Event Category

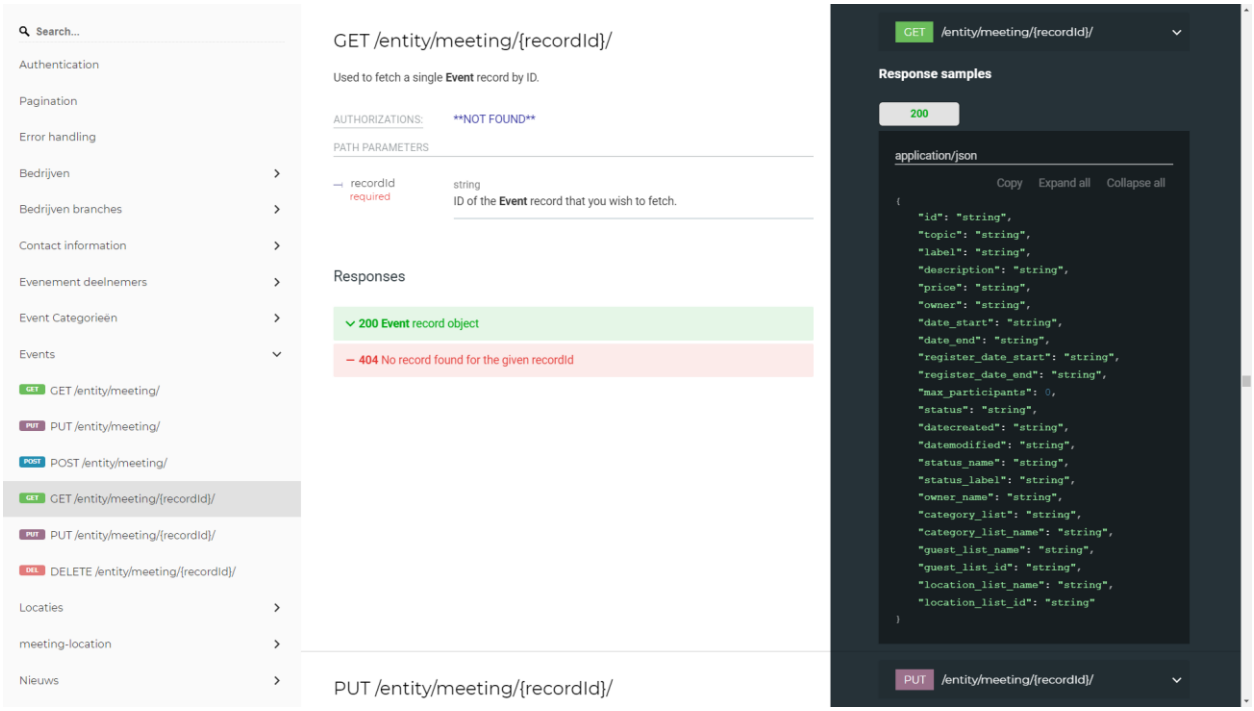
2.4.7. Event API

- Lấy danh sách các Event



Hình 35: API lấy danh sách Event

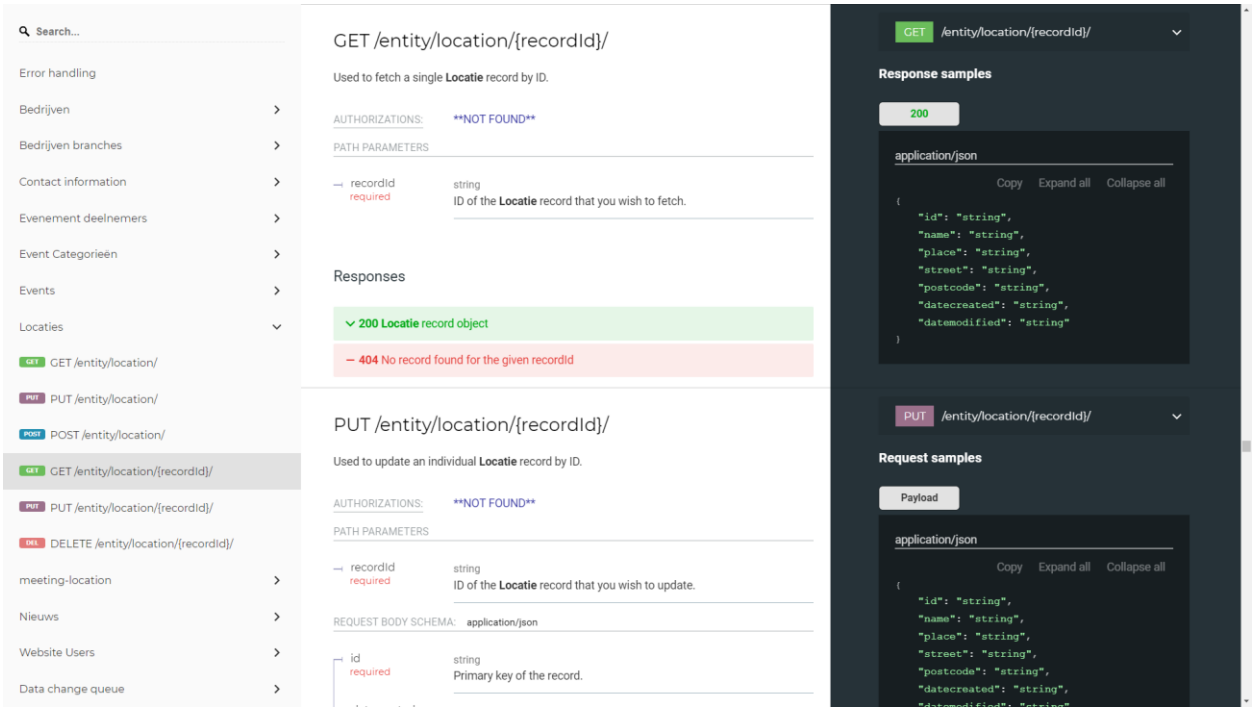
- Lấy một Event dựa theo ID



Hình 36: API lấy một sự kiện theo ID

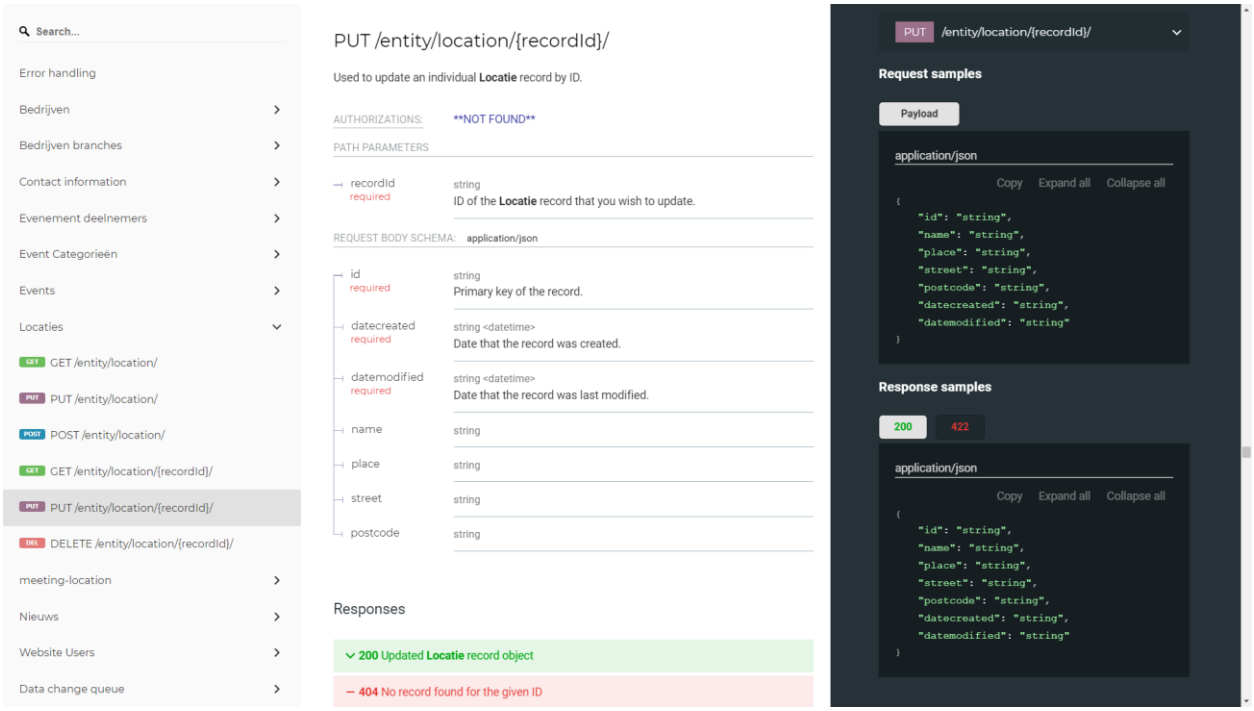
2.4.8. Location API

- Lấy một Location theo ID



Hình 37: API lấy một Location theo ID

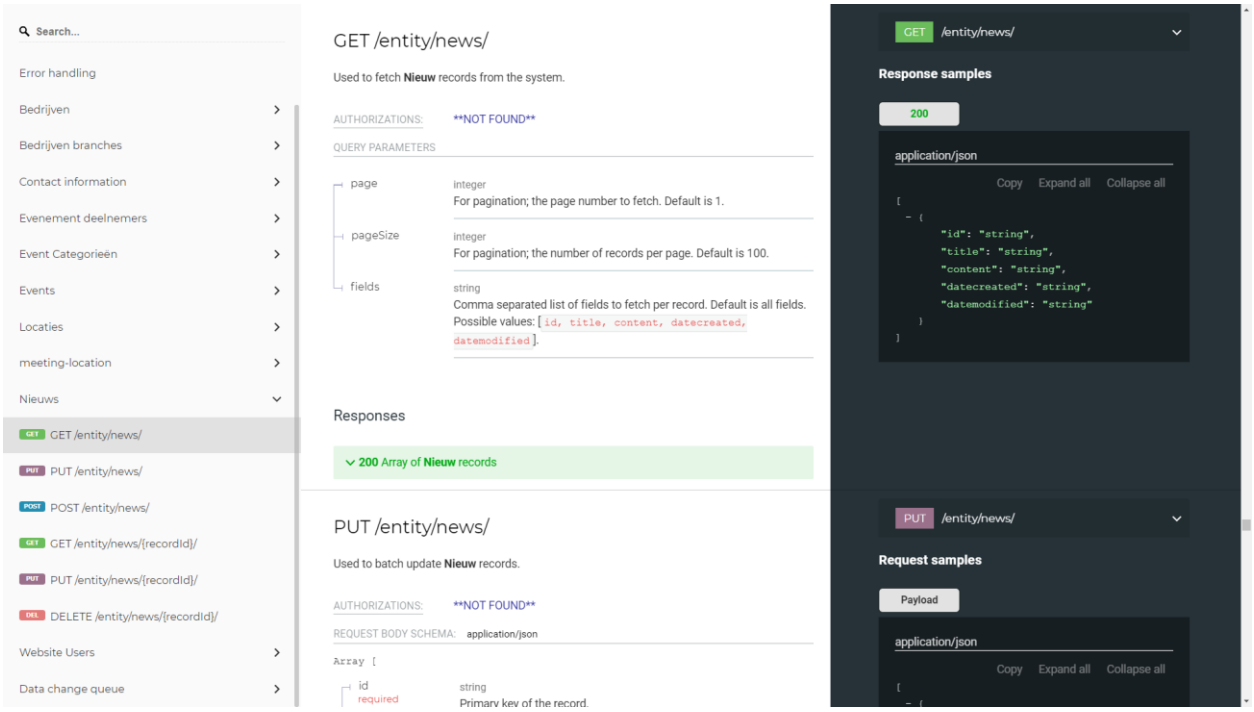
- Cập nhật một Location theo ID



Hình 38: API cập nhật một Location theo ID

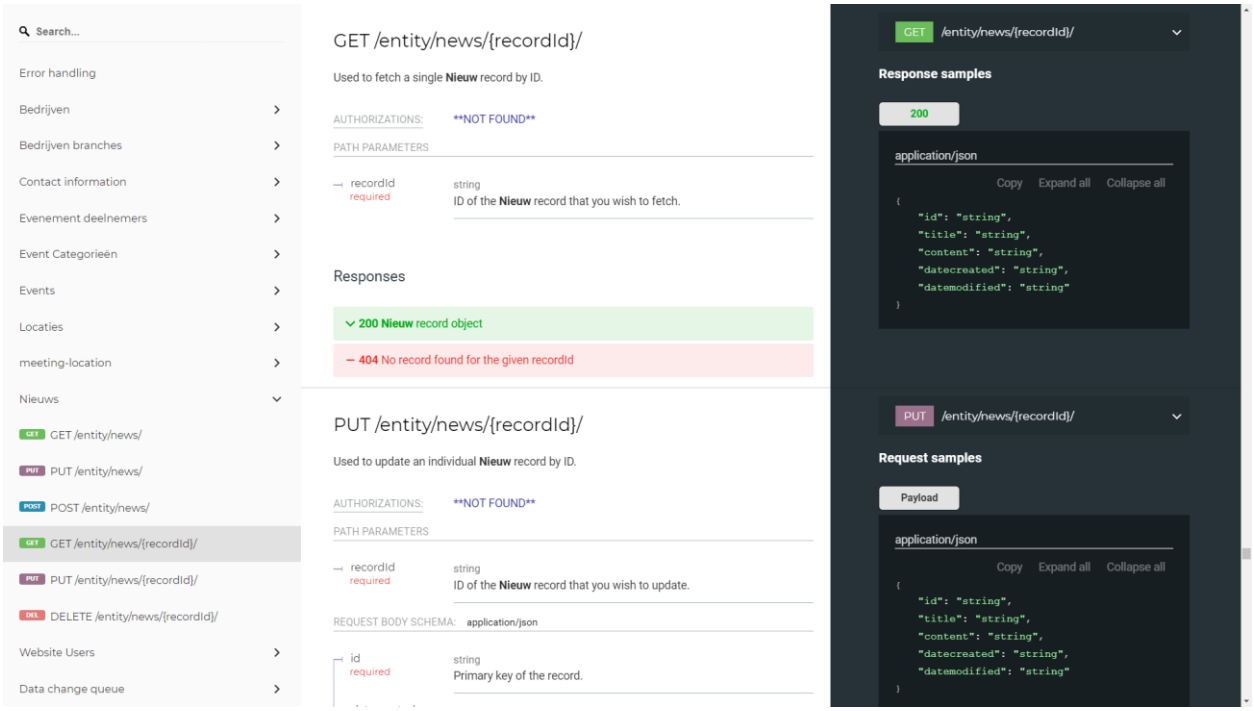
2.4.9. News API

- Lấy danh sách các News



Hình 39: API lấy danh sách News

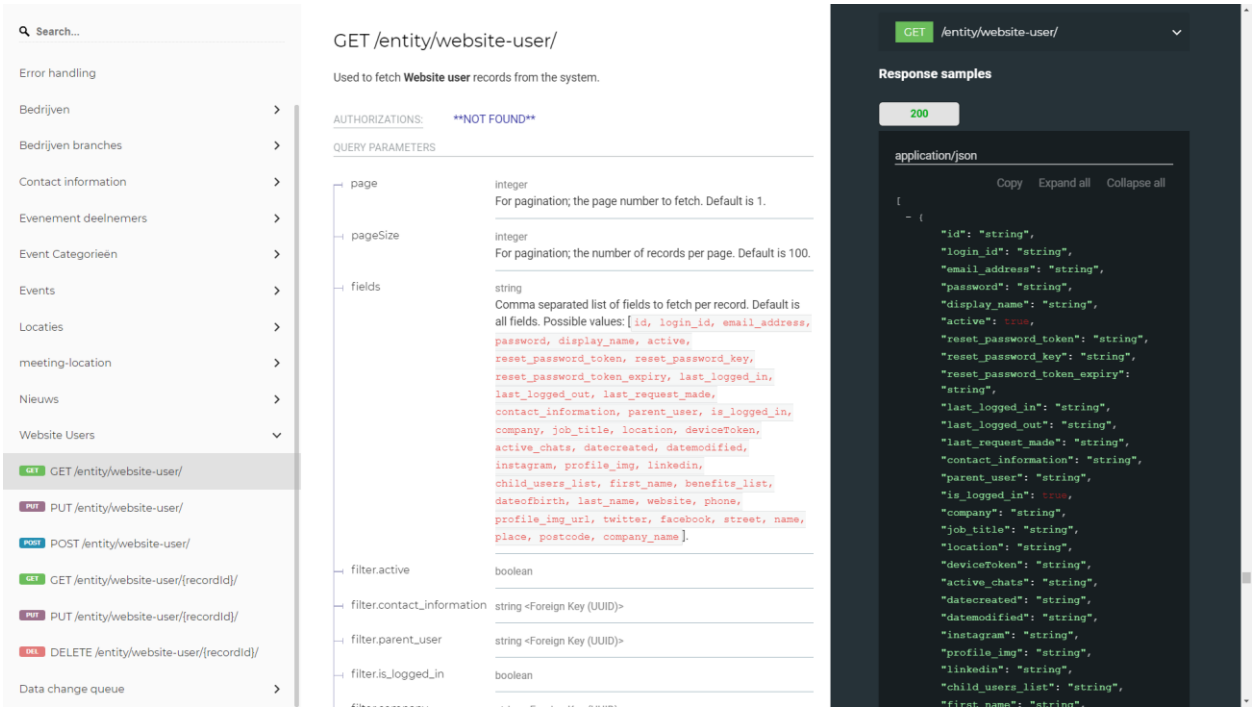
- Lấy một News theo ID



Hình 40: API lấy một News theo ID

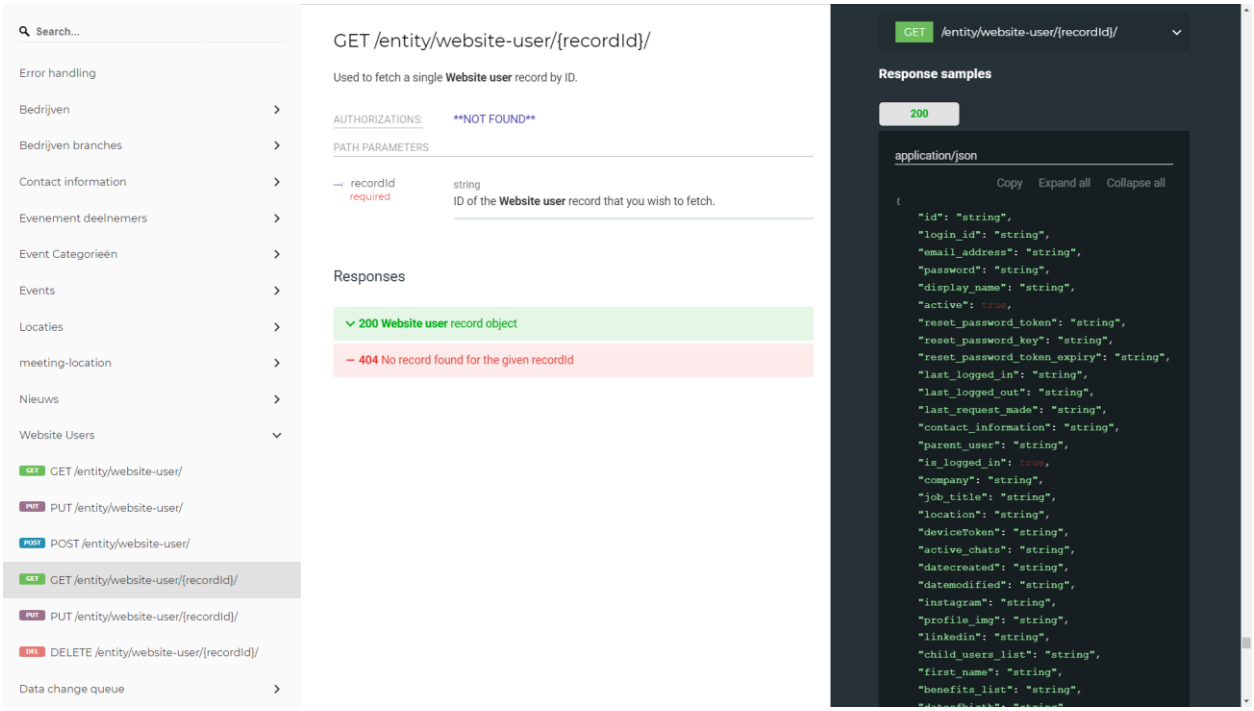
2.4.10. Website User API

- Lấy danh sách Website User



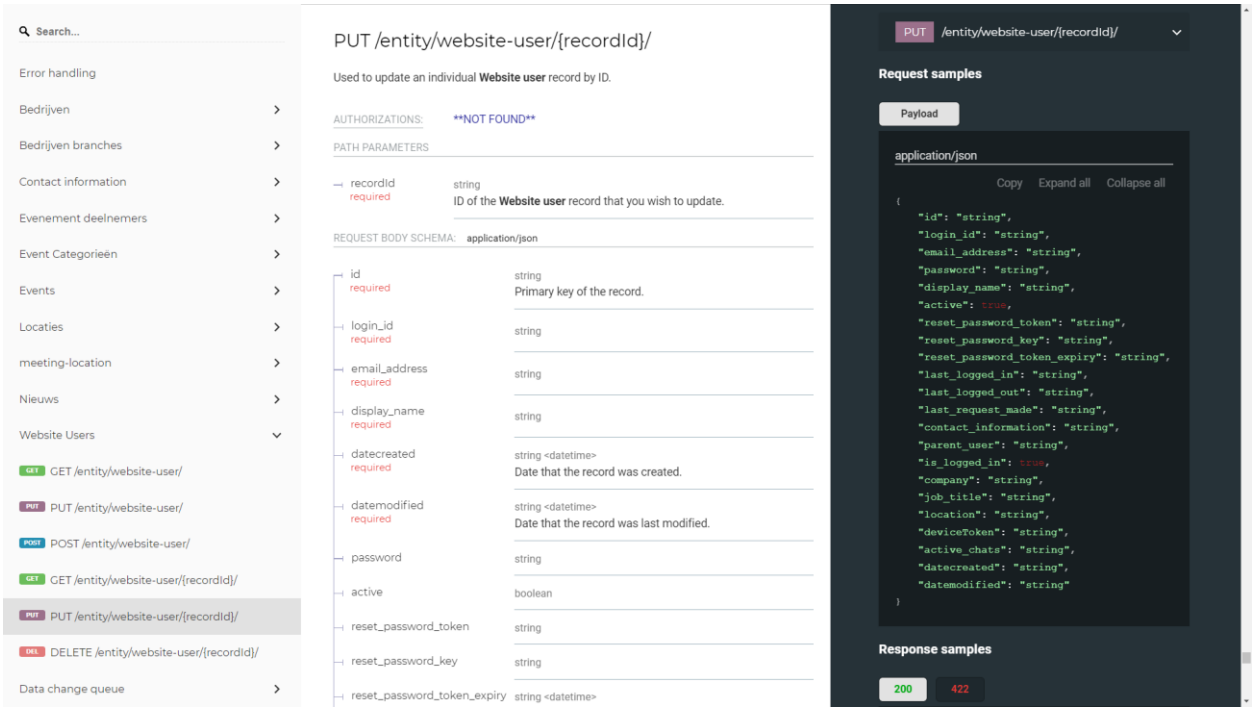
Hình 41: API lấy danh sách User

- Lấy một Website User theo ID



Hình 42: API lấy một User theo ID

- Cập nhật một Website User theo ID



Hình 43: API cập nhật một User theo ID

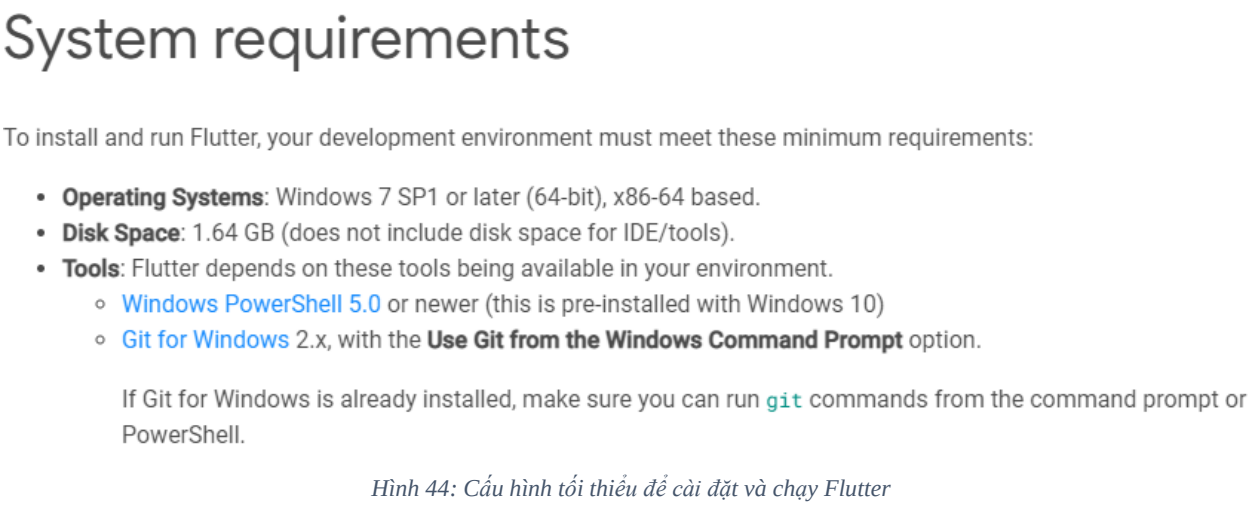
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

3.1. Sử dụng Flutter

3.1.1. Cách cài đặt Flutter

Ta có thể truy cập vào đường link chính thức của Flutter để làm theo hướng dẫn cách cài đặt Flutter tại đây: https://docs.flutter.dev/get-started/install

Tại thời điểm hiện tại, dự án được xây dựng trên hệ điều hành Window 10, nên những cách làm hoặc câu lệnh đưa vào dựa theo hướng dẫn dành cho hệ điều hành Window.



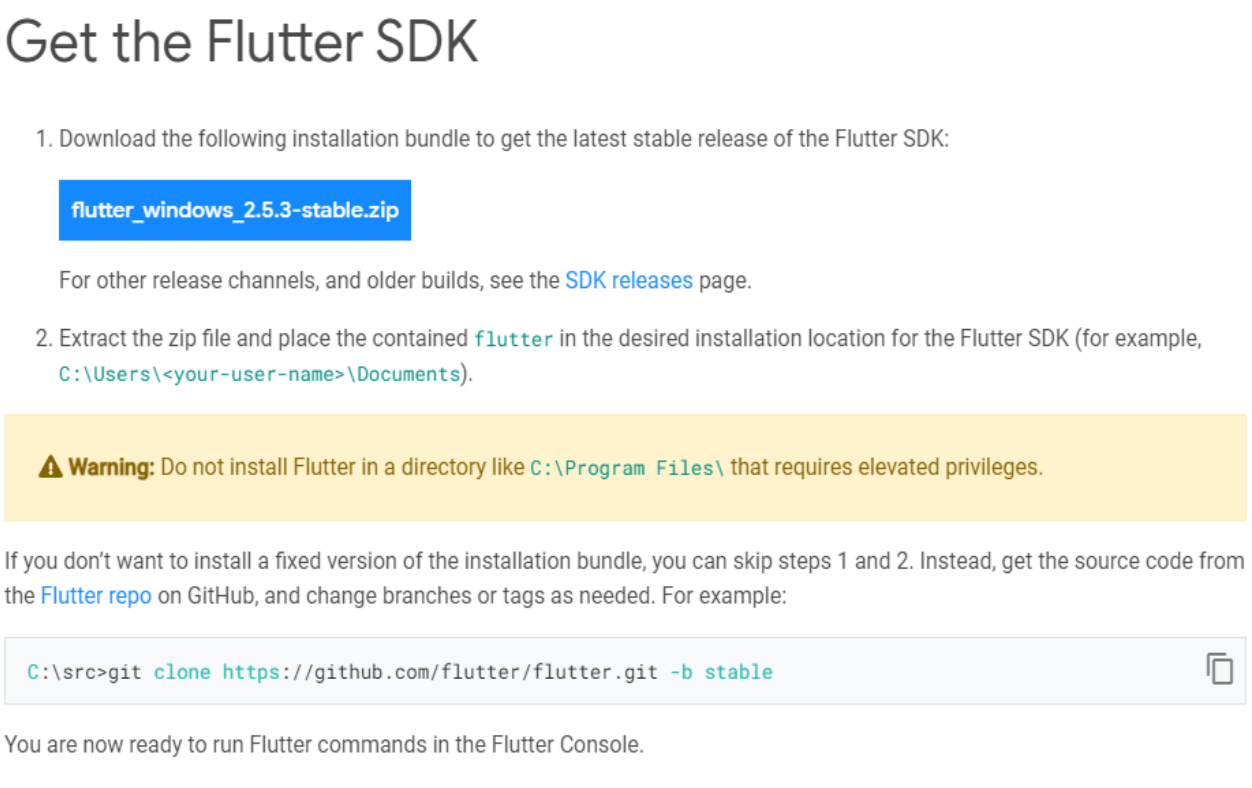
System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Windows 7 SP1 or later (64-bit), x86-64 based.
- **Disk Space**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
  - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  - Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.

    If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Hình 44: Cấu hình tối thiểu để cài đặt và chạy Flutter



Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

   flutter_windows_2.5.3-stable.zip

   For other release channels, and older builds, see the SDK releases page.

2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\Users\<your-user-name>\Documents`).

⚠ **Warning:** Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the Flutter repo on GitHub, and change branches or tags as needed. For example:

```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

You are now ready to run Flutter commands in the Flutter Console.

Hình 45: Cách tải Flutter SDK về máy

Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the `PATH` environment variable:

- From the Start search bar, enter 'env' and select **Edit environment variables for your account**.
- Under **User variables** check if there is an entry called **Path**:
  - If the entry exists, append the full path to `flutter\bin` using `;` as a separator from existing values.
  - If the entry doesn't exist, create a new user variable named `Path` with the full path to `flutter\bin` as its value.

You have to close and reopen any existing console windows for these changes to take effect.

Hình 46: Hướng dẫn cập nhật Variable Path cho Flutter

Run `flutter doctor`

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

```
C:\src\flutter>flutter doctor
```

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

```
[-] Android toolchain - develop for Android devices
    • Android SDK at D:\Android\sdk
    ✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://docs.flutter.dev/setup/#android-setup for detailed instructions.
```

The following sections describe how to perform these tasks and finish the setup process. Once you have installed any missing dependencies, you can run the `flutter doctor` command again to verify that you've set everything up correctly.

Hình 47: Kiểm tra các yêu cầu bằng terminal

Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the Android documentation.
2. Windows-only: Install the Google USB Driver.
3. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
4. In the terminal, run the `flutter devices` command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your `adb` tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the `ANDROID_SDK_ROOT` environment variable to that installation directory.

Hình 48: Hướng dẫn setup thiết bị Android

Set up the Android emulator

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

1. Enable VM acceleration on your machine.
2. Launch **Android Studio**, click the **AVD Manager** icon, and select **Create Virtual Device...**
   - In older versions of Android Studio, you should instead launch **Android Studio > Tools > Android > AVD Manager** and select **Create Virtual Device...**. (The **Android** submenu is only present when inside an Android project.)
   - If you do not have a project open, you can choose **Configure > AVD Manager** and select **Create Virtual Device...**
3. Choose a device definition and select **Next**.
4. Select one or more system images for the Android versions you want to emulate, and select **Next**. An *x86* or *x86_64* image is recommended.
5. Under Emulated Performance, select **Hardware - GLES 2.0** to enable hardware acceleration.
6. Verify the AVD configuration is correct, and select **Finish**.

   For details on the above steps, see Managing AVDs.

7. In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

Agree to Android Licenses

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

1. Make sure that you have a version of Java 8 installed and that your `JAVA_HOME` environment variable is set to the JDK's folder.

   Android Studio versions 2.2 and higher come with a JDK, so this should already be done.

2. Open an elevated console window and run the following command to begin signing licenses.

```
$ flutter doctor --android-licenses
```

3. Review the terms of each license carefully before agreeing to them.
4. Once you are done agreeing with licenses, run `flutter doctor` again to confirm that you are ready to use Flutter.

Hình 49: Hướng dẫn setup máy ảo

3.1.2. Các Package hỗ trợ chính dùng để xây dựng ứng dụng

Lưu ý: Các phiên bản có thể đã được nâng cấp lên phiên bản cao hơn hoặc không còn hỗ trợ cho Framework.

- **Quản lý State**
  - *get: ^4.3.8*
- **Push Notification và Schedule Notification**
  - *flutter_native_timezone: ^2.0.0*
  - *rxdart: ^0.27.2*
  - *flutter_local_notifications: ^9.0.2*
- **Kiến trúc hệ thống, Functional Programming**
  - *dartz: ^0.10.0*
- **Bộ nhớ thiết bị**
  - *get_storage: ^2.0.3*
  - *internet_connection_checker: ^0.0.1+3*
- **Xây dựng giao diện**
  - *flutter_html: ^2.1.0*
  - *url_launcher: ^6.0.12*
  - *uni_links: ^0.5.1*
  - *image_picker: ^0.8.4+3*
  - *intl: ^0.17.0*
  - *google_fonts: ^2.0.0*
  - *qr_code_scanner: ^0.5.2*
  - *qr_flutter: ^4.0.0*
  - *flutter_form_builder: ^6.0.1*
  - *form_field_validator: ^1.1.0*
  - *font_awesome_flutter: ^9.1.0*
  - *cupertino_icons: ^1.0.0*
- **Sử dụng Firebase**
  - *firebase_core: ^1.2.1*
  - *firebase_auth: ^3.0.1*
  - *firebase_analytics: ^8.1.1*
  - *firebase_messaging: ^10.0.8*
  - *cloud_firestore: ^2.2.1*

3.2. Sử dụng GetX khi xây dựng ứng dụng bằng Flutter

3.2.1. GetX là gì?

Đối với những người mới tiếp cận với flutter việc tiềm kiếm cho mình một structure, pattern để theo là cực kỳ quan trọng. Với kinh nghiệm sử dụng và làm việc với cấu trúc thư mục Laravel. GetX trở nên quen thuộc với các khái niệm như Model, View, Controller.

Lưu ý: để bắt đầu sử dụng Plugin này những người mới cũng nên học cơ cản Flutter trước để hiểu các component trong Futter hoạt động như thế nào. Một lần nữa công cụ chỉ giúp chúng ta dễ dàng hơn trong công việc nhưng để giải quyết được những vấn đề phức tạp và chuyên sâu hơn thì việc hiểu về Flutter là bắt buộc

Vậy Get làm được những gì:

- State Manager: Quản lý state trong Flutter
- Navigation Manager: Quản lý việc điều hướng
- Dependencies Manager: Cung cấp giải pháp dependencies injection tuyệt vời
- Utils function: Các hàm tiện ích cực kỳ hữu ích trong lập trình Flutter

3.2.2. Quản lý State bằng GetX

Đầu tiên ta sẽ nêu các vấn đề gặp phải với các kiểu quản lý state phổ biến hiện tại

- BLoC pattern cũng là một cách quản lý state hiệu quả và an toàn. Tuy nhiên code khá rối rắm và tốn nhiều thời gian để implement, trong một số trường hợp thì khó handle. Khó tiếp cận với những người mới.
- MobX cũng là một thư viện tuyệt vời, dễ sử dụng hơn BLoC tuy nhiên lại quá phụ thuộc vào code generation. Có thể gây khó hiểu với người sử dụng và làm phình scope của dự án.
- Provider là một kiểu quản lý state cơ bản và hiệu quả đối với nhiều người. Khi mới bắt đầu mọi người nên sử dụng thuần thục Provider. Tuy nhiên Provider sử dụng InheritedWidget và nó chỉ có thể sử dụng trong widget tree nên nhiều trường hợp sẽ không thể handle được.

Tiếp theo là những ưu điểm của Get - State manager

- Chỉ update những widget cần thiết
- Sử dụng ít bộ nhớ hơn so với các kiểu quản lý state khác
- Với Get chúng ta không phải suy nghĩ sử dụng StatefulWidget hay StateLessWidget nữa. Bây giờ chúng ta chỉ việc một component duy nhất là GetWidget
- Việc tổ chức cấu trúc project sẽ cực kỳ rõ ràng, phần code logic được tách hẳn hoàn toàn so với UI.
- Update widgets mà không cần tiêu tốn RAM.
- Tối ưu hoá bộ nhớ, không phải lo lắng việc Out Memory, Get sẽ tự động thu dọn những component không cần thiết.

3.3. Giới thiệu về kiến trúc Domain – Driven – Design (DDD)

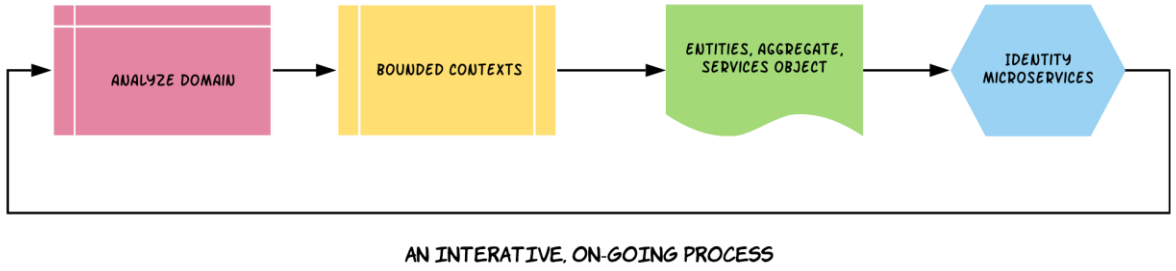
3.3.1. Giới thiệu

Domain-Driven Design là một phương pháp tiếp cận trong việc phân tích và phát triển phần mềm khi giải quyết những vấn đề nghiệp vụ phức tạp. Ý tưởng cơ bản của phương pháp này là việc xây dựng sự kết nối chặt chẽ giữa thiết kế phần mềm và mô hình nghiệp vụ trong suốt vòng đời phát triển sản phẩm. Để tạo nên sự kết nối này, DDD đưa ra 3 yêu cầu cơ bản:

- Trọng tâm của dự án là những nguyên tắc và logic nghiệp vụ.
- Thiết kế phần mềm cần phải phản ánh chính xác mô hình nghiệp vụ.
- Sự cộng tác liên tục giữa kĩ sư và chuyên gia nghiệp vụ.

Kết quả của việc phân tích hệ thống dựa trên phương pháp DDD, kết hợp cùng mô hình kiến trúc Microservices giúp chúng ta tổ chức và phát triển những microservices giải quyết các vấn đề nghiệp vụ một cách tương ứng. Hoạt động kết hợp này được phản ánh qua một qui trình bao gồm các bước:

- Phân tích nghiệp vụ - domain model.
- Định nghĩa ngữ cảnh - bounded context.
- Định nghĩa đối tượng (entities), tập hợp (aggregate) và dịch vụ (service).
- Xác định microservices cần xây dựng.



3.3.2. Khái niệm

DDD được tác giả Eric Evan đề xuất vào vào năm 2003 trong một quyển sách phần mềm nổi tiếng "Domain Driven Design - Tackling complexity in the heart of software". Phương pháp này nhanh chóng nhận được sự đón nhận, phát triển từ cộng động và được áp dụng sâu rộng trong qui trình phát triển phần mềm ngày nay.

Để giới thiệu các khái niệm cơ bản trong DDD, chúng ta lấy ví dụ về phát triển hệ thống quản lý vận tải theo mô hình Uber. Trong quá trình thiết kế và xây dựng hệ thống, những kĩ sư phần mềm phối hợp chặt chẽ với những chuyên gia nghiệp vụ - **domain experts**. Việc trao đổi thông tin giữa những thành viên này cần dựa trên một ngôn ngữ mô tả chính xác các thuật ngữ, vấn đề hoặc qui trình cần giải quyết - **ubiqitous language**.

Quá trình cộng tác liên tục tạo nên một mô hình nghiệp vụ - **domain model** - một hình thức trừu tượng hoá từ mô hình kinh doanh doanh nghiệp - **business domain**. Khi nghiệp vụ phức tạp, mô hình này tiếp tục được phân chia thành nhiều thành phần nghiệp vụ nhỏ hơn, bao gồm:

- Core Domains - nghiệp vụ cốt lõi, cơ bản nhất.
- Sub Domains - nghiệp vụ có tính chất phụ trợ

Điểm quan trọng của việc chia tách này là tạo ra được những phạm vi ngữ cảnh - **bounded context**, giúp xác định rõ ranh giới giữa các nghiệp vụ, đồng thời thể hiện được chính xác ý nghĩa của những thực thể - **entity** trong mỗi phạm vi đó.

Việc phân chia phạm vi các ngữ cảnh cũng giúp giảm thiểu sự phức tạp không cần thiết khi mô hình hoá các thực thể trong thực tế.

Trong một số trường hợp, các entities không tồn tại độc lập và có vòng đời phụ thuộc vào một tập hợp - **aggregate**. Ví dụ khi việc đánh giá chất lượng phục vụ luôn phụ thuộc vào một chuyến đi, và trong trường hợp này thực thể biểu diễn mỗi chuyến đi được gọi là **root entity**.

Entity được định nghĩa là các đối tượng mang dữ liệu với khả năng định danh duy nhất. Bên cạnh Entity, chúng ta cũng sử dụng khái niệm **value object** để định nghĩa các đối tượng chứa dữ liệu đơn thuần. Ví dụ, trong ngữ cảnh Invoice, hoá đơn là những thực thể chứa giá trị **identity** giúp chúng ta có sự phân biệt về tính duy nhất. Mỗi hoá đơn cũng bao gồm đối tượng - **value object** chứa thông tin về thời gian đi, quãng đường. Những đối tượng này chỉ chứa giá trị thông tin đơn thuần và không cần mang tính chất duy nhất.

Trong phát triển phần mềm, có nhiều công nghệ cơ sở dữ liệu khác nhau để lưu trữ những đối tượng entity hay aggregate. Tuy nhiên, việc truy cập các đối tượng

thông tin này thường được "đóng gói" qua một lớp trung gian - **repository**. Việc đóng gói này giảm tính sự phụ thuộc của mô hình nghiệp vụ vào công nghệ lưu trữ, và nâng cao khả năng nâng cấp, thay thế công nghệ khi cần thiết.

Bên cạnh các loại hình đối tượng liên quan đến dữ liệu và truy cập, chúng ta cũng có thêm một khái niệm khác về **services**. Services là những qui trình hoạt động nghiệp vụ, chính sách doanh nghiệp etc…liên quan đến nhiều đối tượng khác nhau. Vì ý nghĩa đó, các đối tượng service có tính chất *stateless* - các hoạt động service thực thi và không lưu giữ lại bất kì trạng thái nào.

3.3.3. Kiến trúc

Sau khi phân tách các phạm vi ngữ cảnh, xác định các đối tượng nghiệp vụ bên trong chúng, bước tiếp theo là việc định nghĩa và xây dựng những microservices tương ứng.

Dựa trên nguyên tắc của phương pháp DDD, để tập trung vào việc xây dựng mô hình nghiệp vụ đồng thời giảm thiểu sự phụ thuộc đối với những thành phần khác của ứng dụng, mỗi services được chia thành nhiều lớp -layers.

Tác giả Eric Evan đề xuất kiến trúc **multi-layers** bao gồm:

- Domain Layer
- Application Layer
- Infrastructure Layer

3.3.3.1. Domain Layer

Là thành phần quan trọng nhất trong kiến trúc bởi việc đóng gói các qui tắc và mô hình nghiệp vụ. Tầng nghiệp vụ này có các đặc điểm:

- Biểu diễn mô hình nghiệp vụ qua entities dưới hình thức các lớp POJO
- Kiểm soát và thể hiện trạng thái hoạt động nghiệp vụ
- Sử dụng Domain Event để thông báo cho các module khác khi một sự kiện xảy ra.
- Hoạt động độc lập với tầng lưu trữ vật lý, không phụ thuộc trực tiếp ORM framework.

3.3.3.2. Application Layer

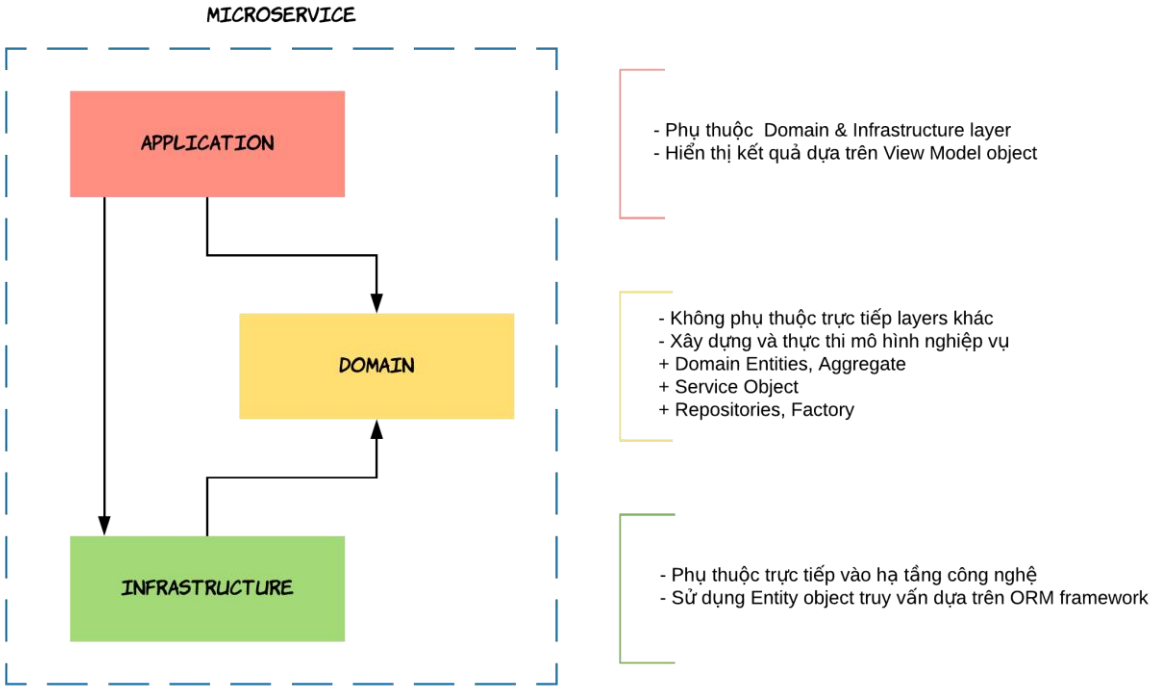
Chức năng chính của tầng này là quản lý các tác vụ yêu cầu từ bên ngoài và chuyển giao - delegate các thực thi nghiệp vụ xuống tầng Domain. Với ý nghĩa đó, tầng ứng dụng có khả năng:

- Presentation interface - cung cấp chức năng ứng cho phía client
- Application interface - giao tiếp ứng dụng với những services khác

3.3.3.3. Infrastructure Layer

Infrastructure layer làm việc trực tiếp với hạ tầng công nghệ của hệ thống phần mềm. Các chức năng chính của tầng này bao gồm:

- Truy vấn, thay đổi thông tin trong cơ sở dữ liệu
- Quản lý dữ liệu trên vùng nhớ tạm thời (cache data)
- Bảo mật, giám sát, lưu trữ thông tin hoạt động (log, monitor)



3.4. Xây dựng lớp Domain (Business Logic Layer)

3.4.1. Cấu hình chung cho hệ thống (Core)

3.4.1.1. Configs Data

Trong file này, ta có thể cấu hình các giá trị như:

- Đường dẫn API
- Khóa API
- Đường dẫn lưu dữ liệu trong thiết bị

Ví dụ:

```
class DataConfigs {
    static const String apiUrl = ‘’
    static const String storagePath = ‘’;
    static const String apiKey = ‘’;
}
```

3.4.1.2. Usecase

Usecase là cách mà giao diện sẽ giao tiếp với tầng dữ liệu qua tầng logic. Usecase sẽ trả về hai trường hợp.

- Thất bại: Quá trình thực hiện yêu cầu xảy ra lỗi
- Thành công: Quá trình thực hiện thành công và trả về dữ liệu

Trong các trường hợp, Usecase cũng sẽ cần có tham số để có thể thực hiện như Usecase đăng nhập, đăng ký, v.v. Các trường hợp khác thì lại không.

Đối với dự án BKL, do thời gian gấp rút, nên đã đơn gian hóa kiến trúc và bỏ qua các phần sử dụng của Usecase trong dự án.

```
class Usecase<T, Params> {
    Future<Either<Failure, T>> call(Params params);
}

Class NoParam {}
```

3.4.1.3. Alert Type

Alert Type là các loại thông báo có thể trả về, dựa theo Bootstrap Alert. Trong Flutter ta sẽ cấu hình như sau:

```
enum AlertType {
    primary,
    secondary,
    success,
    danger,
    warning,
    info,
}
```

Việc cấu hình lớp Alert Type này nhằm mục đích để hiển thị các chỉ báo, thông báo khác nhau lên giao diện người dùng.

3.4.1.4. Failure

Failure là một trong hai loại dữ liệu trả về của Usecase, Failure sẽ được trả khi quá trình thực hiện một Usecase gặp phải lỗi. Failure sẽ gồm tiêu đề, tin nhắn và loại thông báo.

Các loại Failure được cấu hình trong dự án gồm:

- ***Network Failure:*** Gọi khi gặp sự cố mất kết nối hoặc không có kết nối với Internet
- ***Application Failure:*** Gọi khi gặp sự cố xử lý dữ liệu trong hệ thống, như convert sang Json gặp lỗi, chuyển một loại dữ liệu không khớp với dữ liệu định sẵn, v.v.
- ***Server Failure***: Gọi chủ yếu khi gặp phải lỗi khi gọi API (Firebase có code gọi lỗi riêng nên đôi khi không sử dụng Server Failure để bắt lỗi cho các phương thức liên quan tới Firebase). Các lỗi có thể như 401, 404, các lỗi Server 500, … Những lỗi này thường phải được dữ liệu trả về một thông báo lỗi cụ thế lý do vì sao lỗi, do vậy phía Front-end và Back-end cần phải có sự giao đổi về vấn đề này.

Lớp Failure

```
class Failure {
    String title;
    String message;
    AlertType type;

    Failure({
        this.title = ‘Error’,
        this.message = ‘Failed to handle the request’,
        this.type = AlertType.danger,
    ``
```

Lớp Network Failure

```
class NetworkFailure extends Failure {
    NetworkFailure({
        String title = ‘Network Error’,
        String message = ‘Failed to connect to the
internet’,
        AlertType type = AlertType.warning,
    }) : super(
            title: title,
            message: message,
            type: type,
```

Lớp Application Failure

```
class ApplicationFailure extends Failure {
    ApplicationFailure({
        String title = ‘Application Error’,
        String message = ‘Failed to process the request,
        AlertType type = AlertType.danger,
    }) : super(
            title: title,
            message: message,
            type: type,
        `
```

Lớp Server Failure

```
class ServerFailure extends Failure {
    ServerFailure({
        String title = ‘Server Error’,
        String message = ‘Server failed to handle the
request,
        AlertType type = AlertType.warning,
    }) : super(
            title: title,
            message: message,
            type: type,
```

3.4.1.5. Success

Success là một trong hai loại dữ liệu trả về của Usecase, Success sẽ được gọi thì Usecase được thực hiện thành công, khi trả về sẽ gồm tiêu đề, tin nhắn và loại thông

báo giống với Failure, nhưng sẽ đi kèm với một loại dữ liệu trả về, ví dụ như một User hoặc một List các User.

Lớp Success

```
class Success {
     String title;
     String message;
     AlertType type;

     Success ({
          this.title = ‘Success’,
          this.message = ‘Successfully handle the request’,
          this.type = AlertType.success,
     })
}
```

3.4.2. Models

Entity được định nghĩa chỉ bao gồm các thuộc tính và hàm khởi tạo. Sẽ không bị thay đổi nếu Busisness Logic không bị tác động, nghĩa là nếu thay đổi cách gọi dữ liệu, máy chủ, v.v. Thì Entity và các thuộc tính vên trong của nó sẽ vẫn vậy.

Model được định nghĩa tại tầng Infrastructure, Model sẽ kế thừa từ một Entity có sẽ và cấu hình tùy vào các cách gọi dữ liệu như json hoặc Firebase Document, nhưng phải trả về đủ các dữ liệu theo yêu cầu của Entity

Theo kiến trúc DDD thì Models ở đây sẽ được xác định thành Entity, nhưng do thời gian gấp rút cần phải rút gọi lại những phần không cần thiết, vì vậy Models sẽ được viết trực tiếp vào Entity gộp các thuộc tính lẫn các hàm chuyển dữ liệu như ***toJson()***, ***fromJson()*** chung với nhau.

Quá trình viết Models có thể được thực hiện nhanh hơn bằng công cụ ***quicktype.io***

Ví dụ: Company Models

```
class CompanyModel {
      CompanyModel({
            required this.id,
            required this.name,
      })
      String id;
      String name;
      factory CompanyModel.fromJson(Map<String, dynamic> json) {
            return ComapnyModel(
                  id: json[‘id’],
                  name: json[‘name’],
            );
      }
}
```

3.4.3. Repositories

Repository tại tầng Domain sẽ cấu hình các phương thức của một Model hoặc một Chức năng. Các hàm của mỗi Repository không thực hiện một chức năng nhất định, nhưng sẽ khai báo để cho tầng dữ liệu dùng đề cấu hình riêng. Điều này khiến rằng nếu ta thay đổi cách dữ liệu được xử lý từ Server hay từ phía Client tới Server, thì sẽ không ảnh hưởng tới Business Logic của hệ thống.

Ví dụ như ta sẽ cấu hình cho một Company Model được tương tác hoàn toàn qua API bằng các hàm sau:

```
abstract class CompanyRepository {
   Future<Either<Failure, List<CompanyModel>>> list();
   Future<Either<Failure, CompanyModel>>> get(String id);
   Future<Either<Failure, CompanyModel>>> create(CompanyModel data);
   Future<Either<Failure, CompanyModel>>> update(CompanyModel data);
   Future<Either<Failure, Success>>> delete(Company data);
}
```

3.5. Xây dựng lớp Infratructure (Data Layer)

3.5.1. Các Class hỗ trợ (Data Helpers)

3.5.1.1. API Helper

Lớp API Helper được dùng để hỗ trợ thực hiện những chức năng chung khi gửi một Request tới Server, ví dụ nếu ta muốn mỗi lần một request được gửi lên và khi trả về, màn hình console sẽ in ra các thông tin của Response để giúp lập trình viên debug, hoặc bắt các lỗi như 401, 404, 500 và thực hiện một hành động nhất định hoặc trả về một giá trị nhất định.

API Helper cũng được dùm để cấu hình đường dẫn prefix của một API url. Giúp việc gọi request của những lớp API kết thừa từ nó trở nên dể dàng hơn.

3.5.1.2. Json Formatter

Trước khi Flutter và Dart có Chức năng null-safety (Một chức năng kiểm tra biến hoặc giá trị có thể là null hay không giống như ngôn ngữ Swift khi phát triển iOS). Các dự án bằng Flutter thường có trường hợp bị crash hoặc lỗi do dữ liệu trả về từ server có thể bị null do map dữ liệu bằng Json.

Do đó, người lập trình viên có 2 phương án. Một là cấu hình cho các biến của model trở thành "nullable", nghĩa là các biến này có thể trống, không có dữ liệu. Nhưng điều này cũng khiến quá trình gắn data vào giao diện thêm một bước phức tạp, vì giờ đây giao diện sẽ phải kiểm tra rằng giá trị này có null hay không, nếu null thì làm gì, nếu không thì sao…

Để tránh phải bỏ thời gian để xử lý các logic giao diện đó, ta sẽ cấu hình cho các thuộc tính trong model trở thành không thể null, nghĩa là biến đó khi gọi về sẽ phải có dữ liệu, nếu không có thì phải gán cho thuộc tính đó một dữ liệu mặc định. Và để đạt được yêu cầu trên, ta sẽ tạo ra một lớp Json Formatter để hỗ trợ xử lý các giá trị null khi Server trả về.

Lưu ý: Các thuộc tính của model không nên cấu hình un-nullable cho tất cả, mà phải xem xét về cách các thực thể liên hệ với các thực thể khác. Ví dụ như một User có thể có hoặc không có số điện thoại ghi trong phần thông tin liên lệ, vì vậy có thể để thuộc tính này là nullable.

```
class JsonFormater {
  static double asDouble(dynamic value) {
    if (value is double) {
      return value;
    } else if (value is int) {
      return value.toDouble();
    } else {
      return double.tryParse(value ?? '') ?? 0;
    }
  }

  static int asInt(dynamic value) {
    if (value is int) {
      return value;
    } else if (value is int) {
      return value.toInt();
    } else {
      return int.tryParse(value ?? '') ?? 0;
    }
  }

  static String asString(dynamic value) {
    if (value is String) {
      return value;
    } else if (value == null) {
      return '';
    } else {
      return value.toString();
    }
  }

  static DateTime asDateTime(dynamic value) {
    if (value is DateTime) {
      return value;
    }

    try {
      return DateFormat("MMMM, dd yyyy HH:mm:ss").parse(value);
    } catch (error) {
      return DateTime.now();
    }
  }
}
```

3.5.2. Cấu hình API (Sources)

Các API gọi trực tiếp tới server cũng được chia ra theo các Models và thường được viết sẵn theo phương thức CRUD, nhưng tùy vào các loại API khác nhau mà đường liên kết sẽ khác nhau.

```
Future<CompanyModel> getCompany(String companyId) async {
    Response response = await get(
                'entity/company/$companyId/',
                headers: ApiHelper.headersAuth
            );

    if (response.statusCode == 200) {
      try {
        return CompanyModel.fromJson(response.body);
      } catch (e) {
        throw ServerException(
          url: 'entity/company/$companyId/',
          response: response,
          message: e.toString(),
        );
      }
    } else {
      throw ServerException(
        url: 'entity/company/$companyId/',
        response: response,
        message: "Failed to get company",
      );
    }
  }
```

3.5.3. Cấu hình Repository (Repository Implement)

Tại lớp Repository Implement, ta bắt đầu xử lý các yêu cầu về mặt logic và kết nối chúng với API. Tại tầng này, chúng ta cũng bắt các lỗi như Network Failure, hay Application Failure và trả về các Server Failure tùy thuộc vào các lớp gọi API có throw về Server Exception hay không.

```
class CompanyRepoImpl implements CompanyRepo {
  CompanyApi _companyApi = CompanyApi();
  CompanyBranchApi _companyBranchApi = CompanyBranchApi();
  LocationApi _locationApi = LocationApi();
  FileApi _fileApi = FileApi();
  UserApi _userApi = UserApi();

  @override
  Future<Either<Failure, CompanyModel>> getCompany(String companyId)
async {
    if (await NetworkConnection.isConnected) {
      try {
        return Right(await _companyApi.getCompany(companyId));
      } on ServerException catch (e) {
        return Left(ServerFailure(message: e.message));
      }
    } else {
      return Left(NetworkFailure());
    }
  }
}
```

3.6. Xây dựng lớp Presentation (UI Layer)

3.6.1. Xây dựng Controller

Khi xây dựng controller cho một chức năng, một màn hình hay một Model nhất định, phải phân tích các khả năng như sau:

- Dữ liệu này có được giữ lại hay không?
- Dữ liệu này có được sử dụng qua nhiều màn hình khác hay không?
- Dữ liệu này có cập nhật liên tục hay không?

Với các dữ liệu được đổi hoặc được thay thể, cập nhật mới sau khi quay lại. Ví dụ như khi chọn một Event Detail, mặc dù cùng một màn hình nhưng dữ liệu phải được gọi về mới từ Server để khớp với Event mà người dùng đã chọn.

Đối với trường hợp như vậy, chúng ta sẽ không lưu thông tin biến đó vào Controller mà sử dụng FutureBuilder (Một Widget dùng để gọi dữ liệu mỗi khi được chạy) để cập nhật giao diện.

Với các dữ liệu được sử dụng qua nhiều màn hình. Đổi với ứng dụng BKL, danh sách Event và danh sách người dùng luôn được dùng qua lại ở nhiều màn hình khác nhau.

Vậy để tiết kiệm được các request tới Server, ta có thể lưu danh sách này vào Controller để giữ trạng thái cho danh sách, điều này cũng một phần giúp cho các thông tin được đồng bộ qua nhiều màn hình.

Đi kèm với Controller cũng có một khái niệm từ GetX đó là Binding, khi binding một controller với một hoặc nhiều màn hình trong dự án, sẽ đảm bảo được rằng Controller sẽ không bị rỗng (Chưa được khởi tạo) khi hiển thị màn hình.

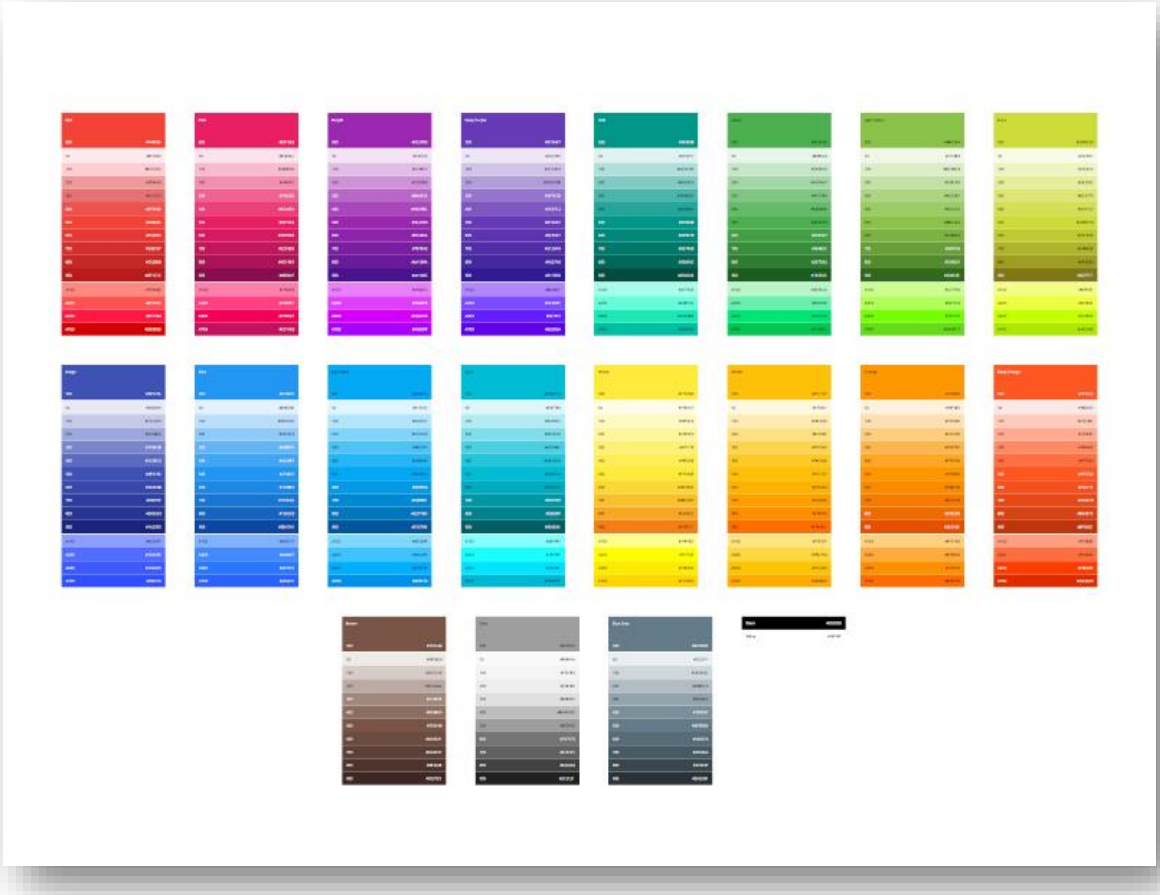
- Các bước để xây dựng một Controller đầy đủ bao gồm:
- Khai báo Repository
- Khai báo các biến dùng để hiển thị lên màn hình
- Cấu hình các hàm liên quan tới chức năng của Controller hoặc Models
- Cấu hình Class Binding

3.6.2. Cấu hình các thuộc tính chung cho giao diện (Global)

Khi xây dựng một dự án, một trong những điều kiện giúp cho quá trình xây dựng trở nên trôi chảy đó là tính nhất quán, đồng bộ. Và để đạt được điều đó, ta sẽ cấu hình những thuộc tính chung của giao diện trước khi bắt đầu xây dựng, mặc dù trong quá trình phát triển sẽ có những thay đổi, nhưng đây là một cách tốt để giúp nâng cao trình độ và khả năng viết Clean Code.

Các thuộc tính nên được cấu hình trước khi bắt đầu đó chính là:

- App Config: Cấu hình tên của App, những thông tin liên lạc, version của ứng dụng, …
- App Color: Màu sắc của ứng dụng, bao gồm những màu quan trọng như primary color và accent color, cũng có thể cấu hình qua colorScheme để phát triển Chức năng đổi tông màu cho ứng dụng
- App Size: Kích thước và các thông số giao diện của ứng dụng, chủ yếu là padding, margin, border radius của các nút bấm, text field, …



3.6.3. Xây dựng giao diện

3.6.3.1. Màn hình chào mừng

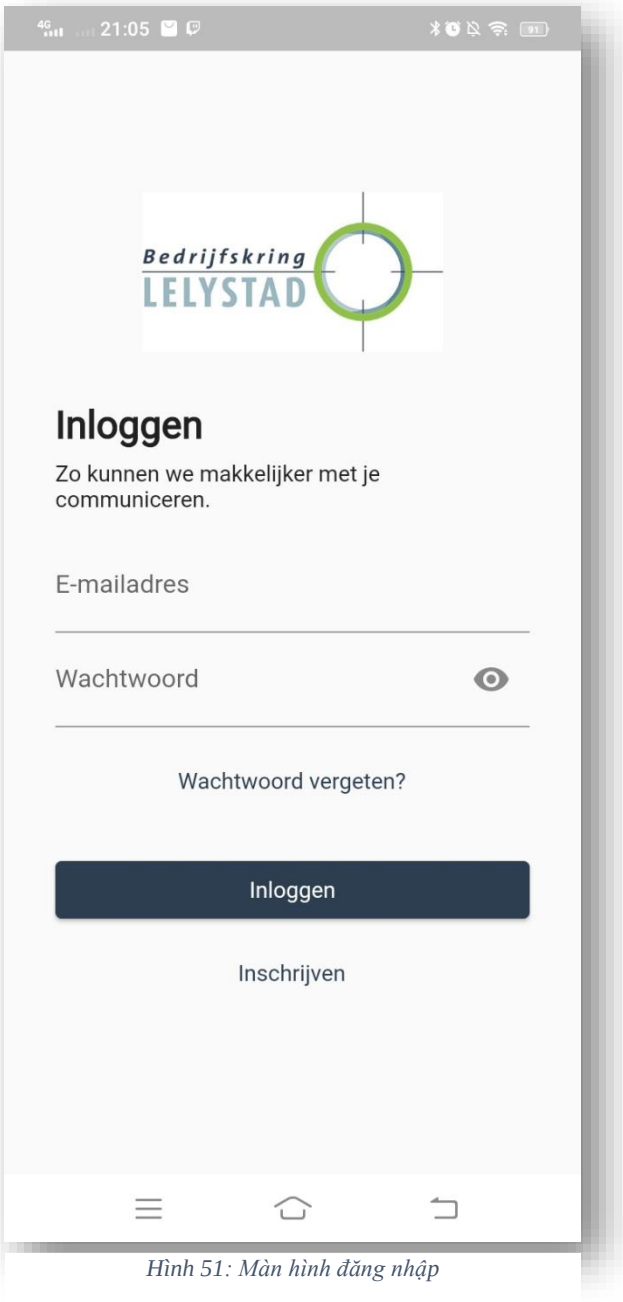
Màn hình Chào mừng (Splash) dùng để hiển thị logo, thương hiệu của khách hàng. Thường xuất hiện khi khởi chạy App và hiển thị trong khoảng từ 3 – 5 giây, trong thời gian đó cũng có thể dùng để load trước dữ liệu hoặc kiểm tra trạng thái đăng nhập của người dùng (Keep Login)

Hình 50: Màn hình Splash

3.6.3.2. Màn hình đăng nhập

Màn hình Login, hiển thị sau khi màn hình Splash kết thúc với điều kiện là nếu người dùng chưa đăng nhập vào ứng dụng hoặc Chức năng lưu đăng nhập đang tách.

Tại màn hình này, người dùng sẽ thực hiện chức năng đăng nhập bằng email và mật khẩu. Người dùng cũng có thể di chuyển tới các màn hình khác như Đăng ký và khôi phục mật khẩu.

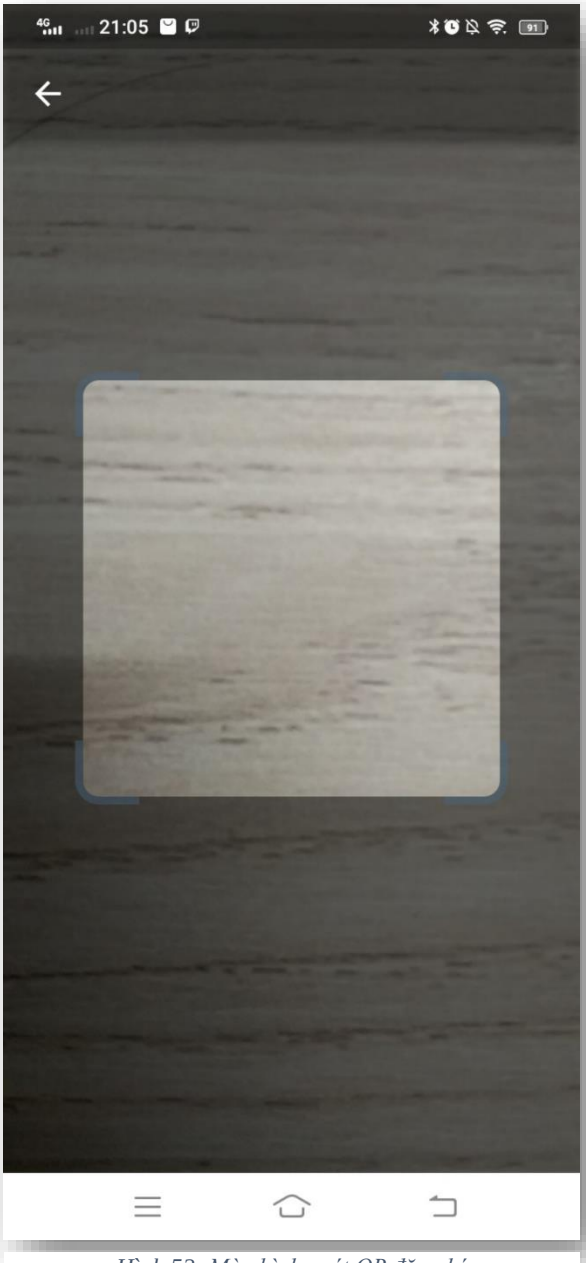


Hình 51: Màn hình đăng nhập

3.6.3.3. Màn hình quét mã QR đăng ký

Khi người dùng bấm nút đăng ký tạo màn hình Login, ứng dụng sẽ mở giao diện quét mã QR đăng ký, trước khi hiển thị sẽ kiểm tra quyền sử dụng Camera của thiết bị ra chỉ tiếp tục hoạt động khi có phép.

Nếu người dùng quét ra một mã QR phù hợp, ứng dụng sẽ gửi thông tin tới Server để lấy thông tin Company mà người dùng quét được. Sau đó chuyển qua màn hình đăng ký tài khoản.

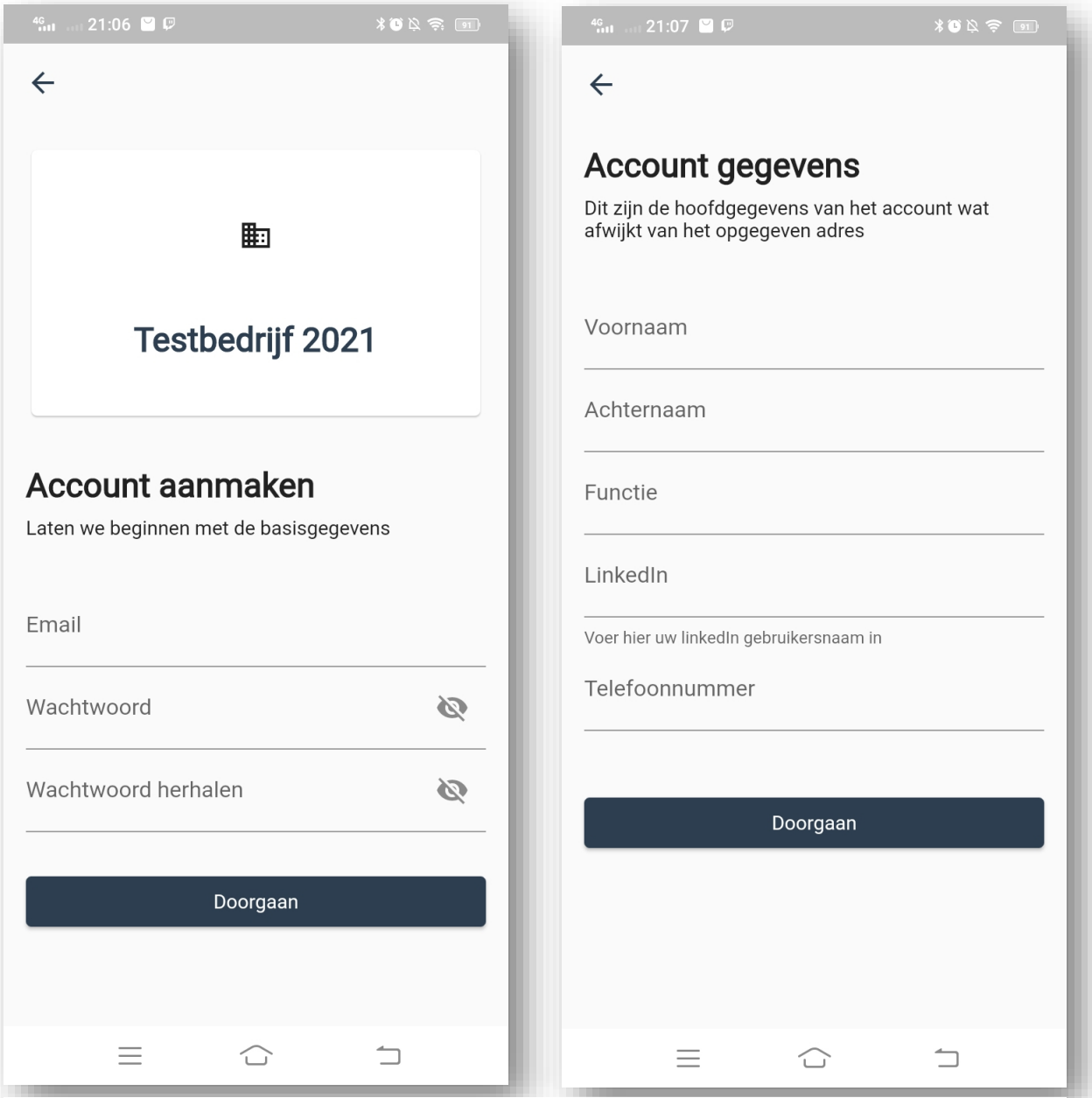


Hình 52: Màn hình quét QR đăng ký

3.6.3.4. Màn hình đăng ký

Khi quét mã và lấy dữ liệu thành công, màn hình đăng ký tài khoản sẽ hiển thị thông tin của doanh nghiệp mà người dùng đã quét. Ứng dụng sẽ yêu cầu người dùng nhập thông tin đăng nhập và tiếp theo đó là thông tin cá nhân.

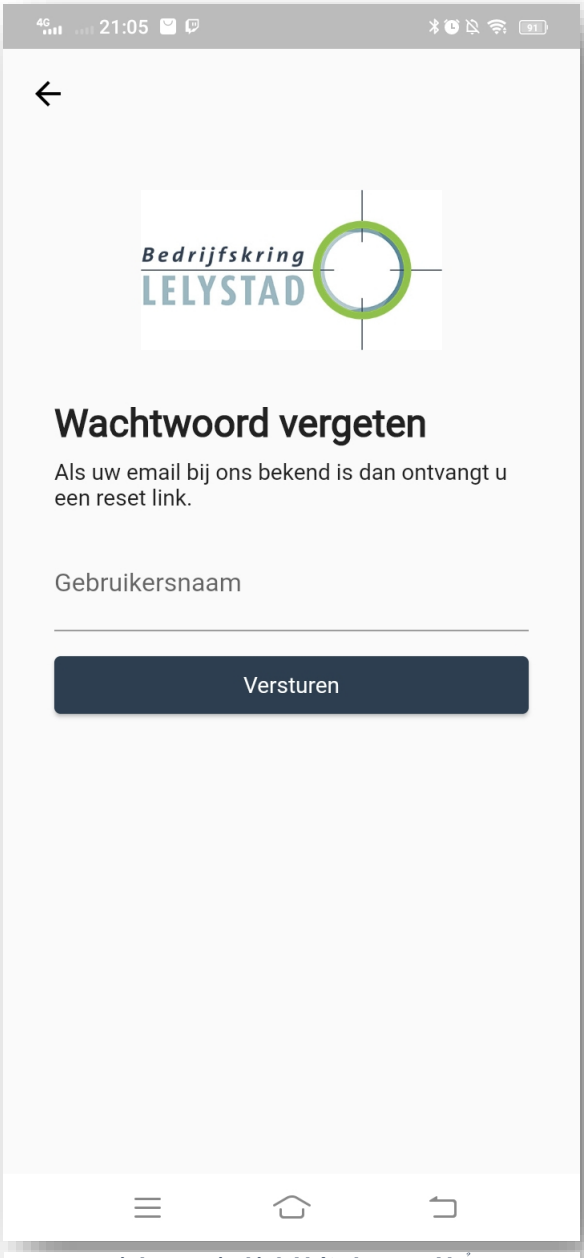
Đây là màn hình thực hiện chức năng đăng nhập và tạo một thông tin liên hệ mới cho người dùng. Sau đó sẽ quay về lại màn hình đăng nhập.



Hình 53: Màn hình đăng ký tài khoản

3.6.3.5. Màn hình khôi phục mật khẩu

Người dùng có thể gửi yêu cầu khôi phục mật khẩu tại màn hình Forgot Password. Người dùng sẽ cần cung cấp địa chỉ email, nếu địa chỉ cấp tồn tại, ứng dụng sẽ gửi chỉ báo thành công lại về người dùng.

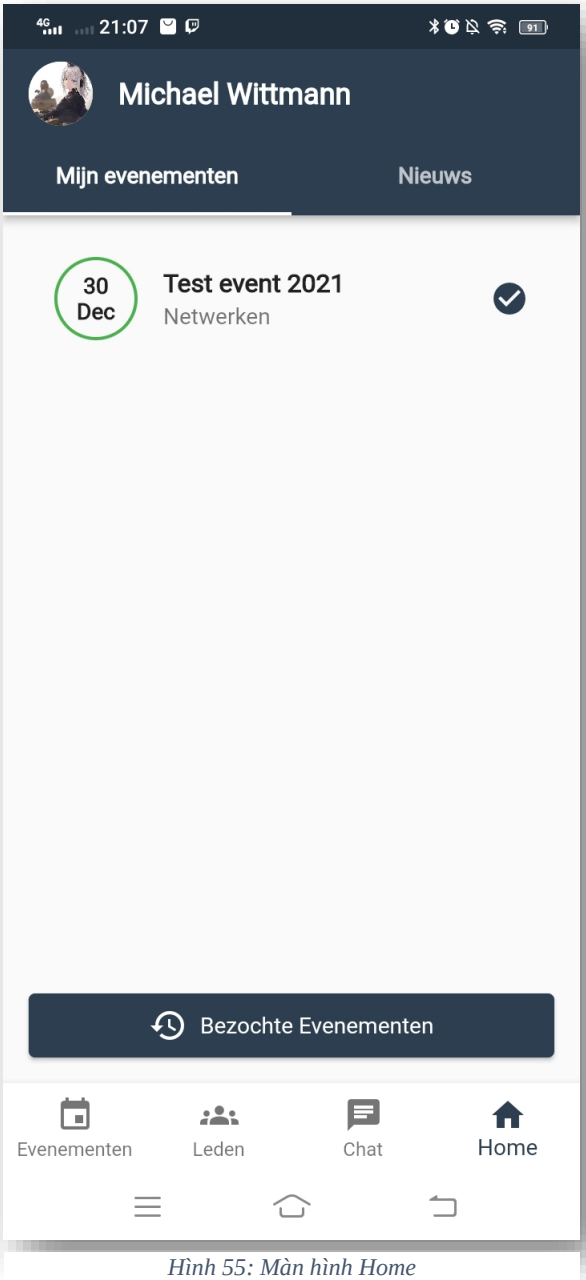


Hình 54: Màn hình khôi phục mật khẩu

3.6.3.6. Màn hình trang chủ

Sau khi đăng nhập thành công, người dùng sẽ được chuyển tới màn hình Home, tại đây, dữ liệu của Event sẽ được load và sau đó sẽ hiển thị danh sách các sự kiện mà người dùng đã đăng ký.

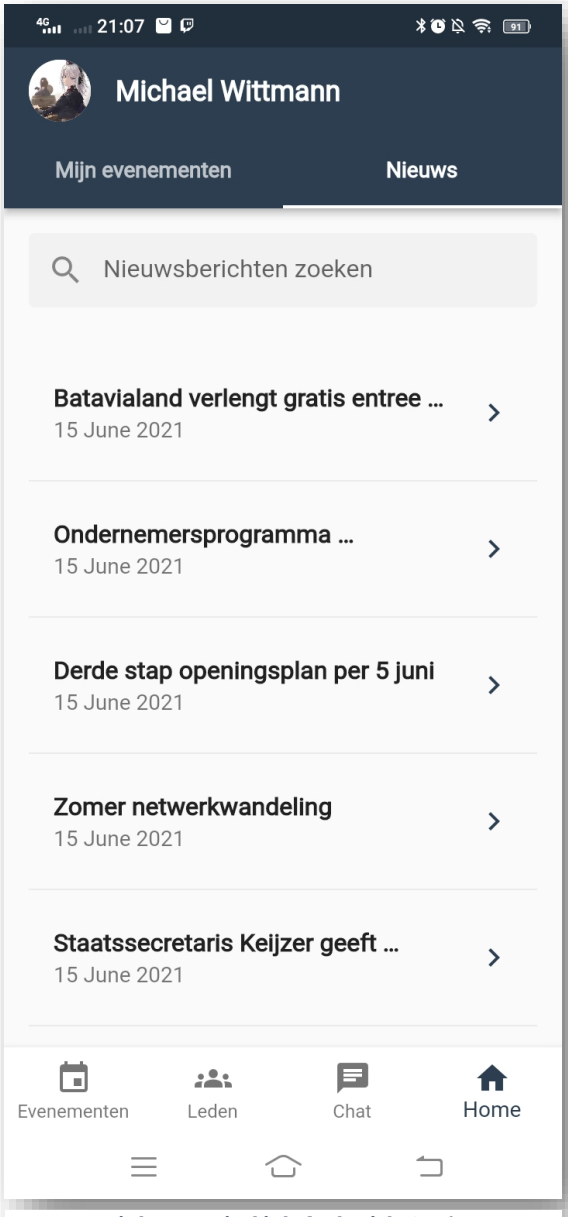
Tại màn hình người dùng có thể điều hướng qua các màn hình như Profile, News, Events, Event History, Member, Chat Overview



Hình 55: Màn hình Home

3.6.3.7. Màn hình danh sách tin tức

Người dùng có thể xem các tin tức mới của BKL tại Tab News của màn hình Home, khi chọn vào một tin tức thì ứng dụng sẽ mở thông tin chi tiết của tin tức đó.

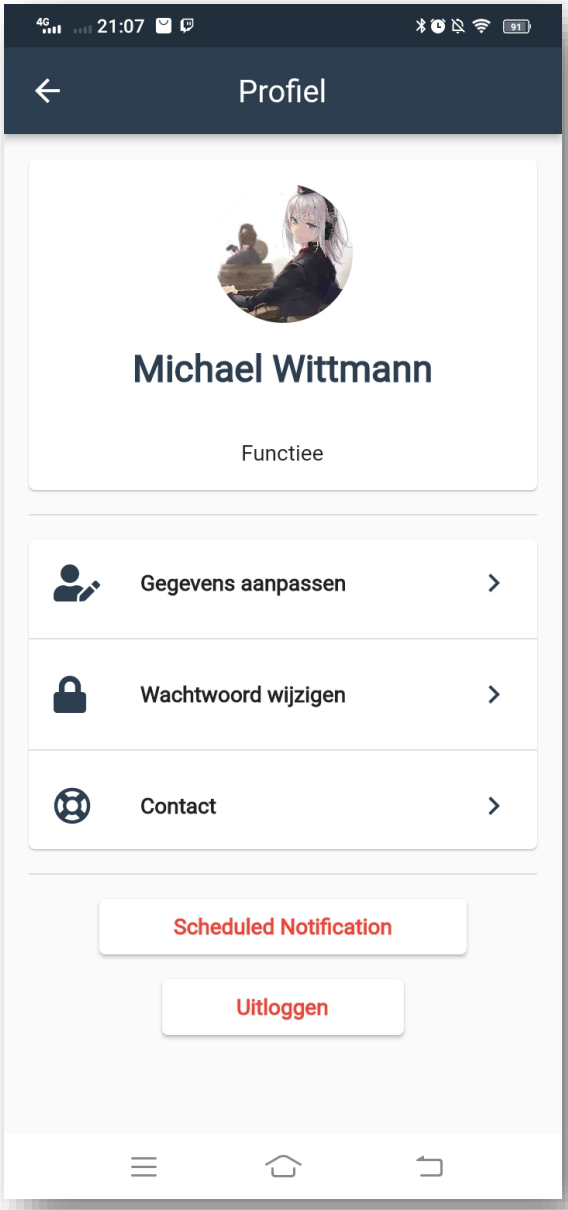


Hình 57:: Màn hình danh sách tin tức

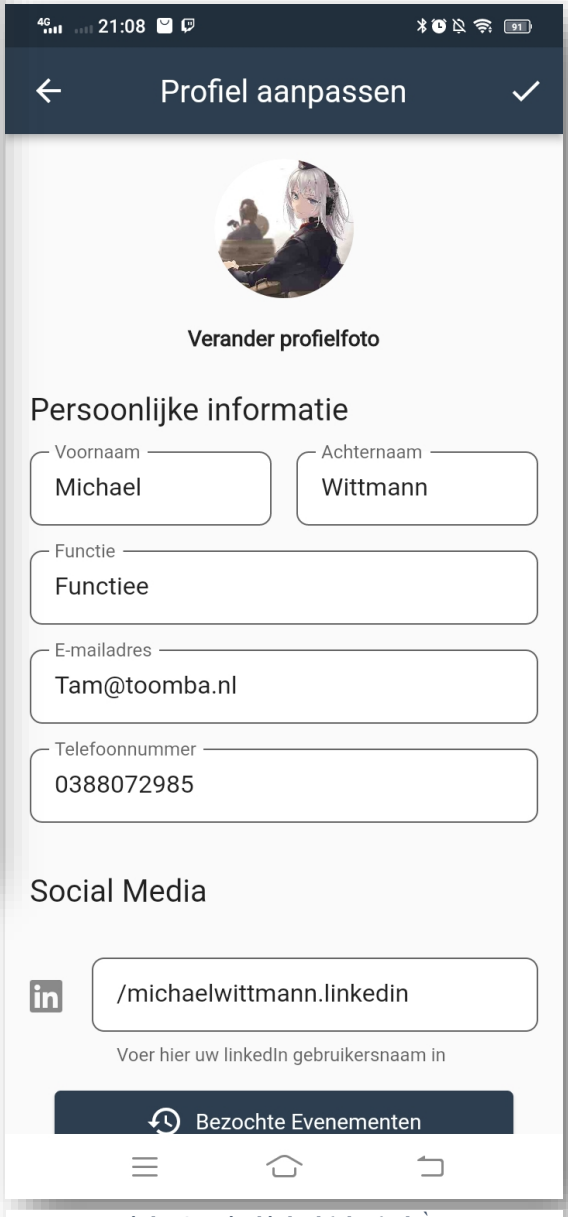
Hình 56: Màn hình đọc tin tức

3.6.3.8. Màn hình hồ sơ cá nhân và chỉnh sửa hồ sơ

Từ màn hình Home, người dùng có thể chuyển tới màn hình Profile và chọn Edit Profile để có thể cập nhật thông tin cá nhân.



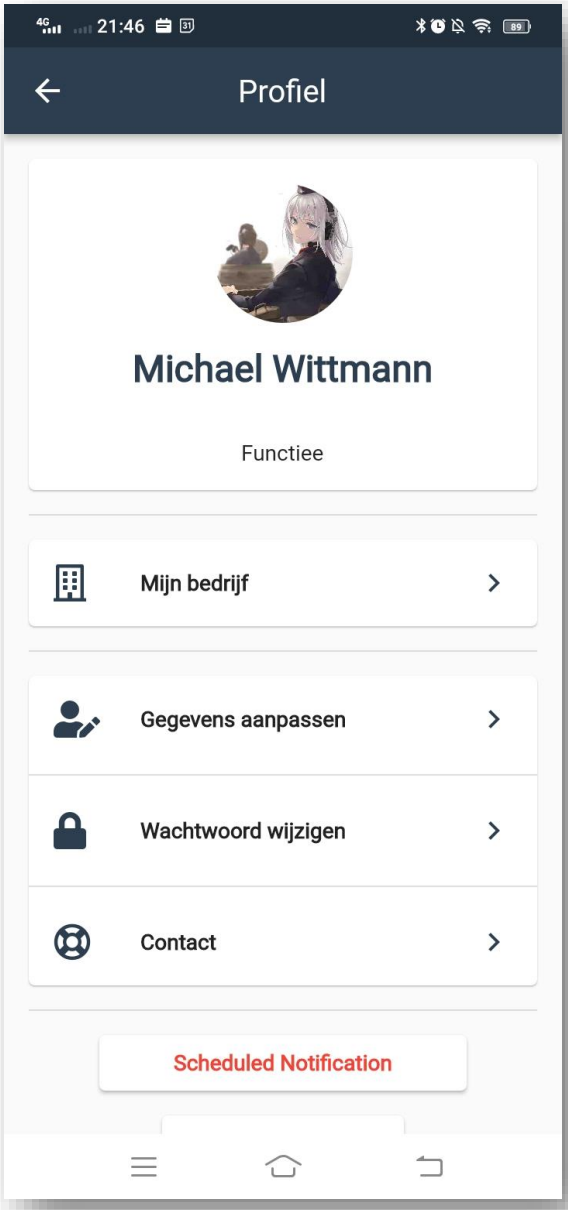
Hình 59: Màn hình Profile (Quá trình Testing)



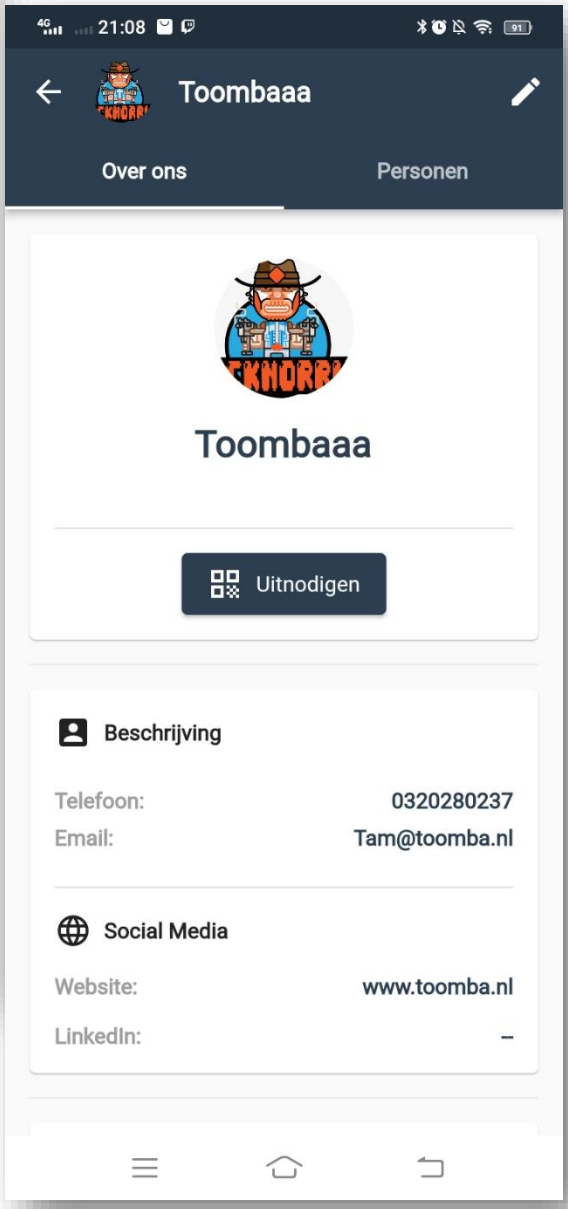
Hình 58: Màn hình chỉnh sửa hồ sơ

3.6.3.9. Màn hình hồ sơ và chỉnh sửa hồ sơ doanh nghiệp của người dùng

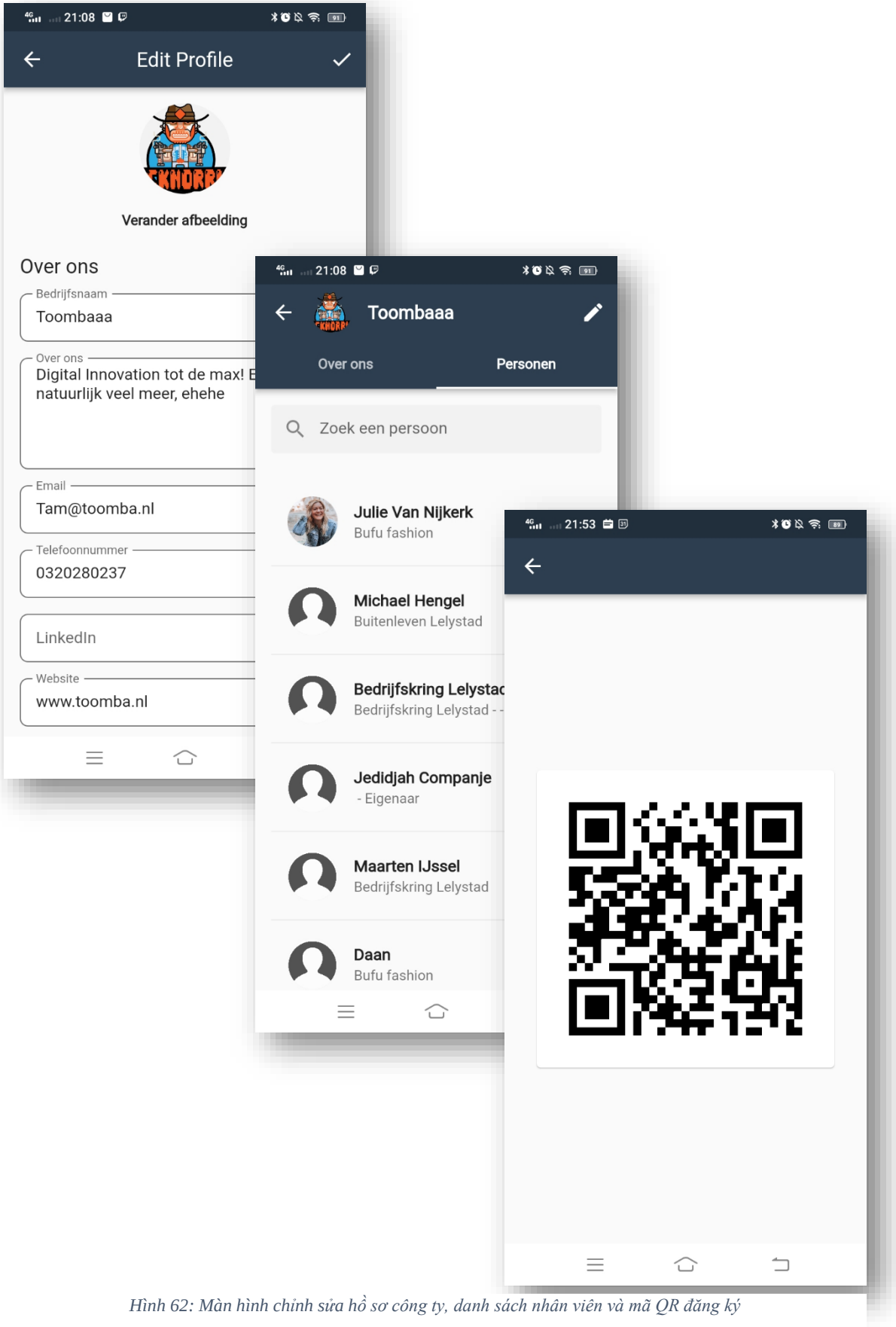
Nếu người dùng là chủ của một doanh nghiệp trên hệ thống, giao diện sẽ hiển thị thêm nút bấm My Company, giúp người dùng chuyển tới màn hình quản lý doanh nghiệp của họ



Hình 60: Màn hình Profile với My Company



Hình 61: Màn hình quản lý Company của người dùng

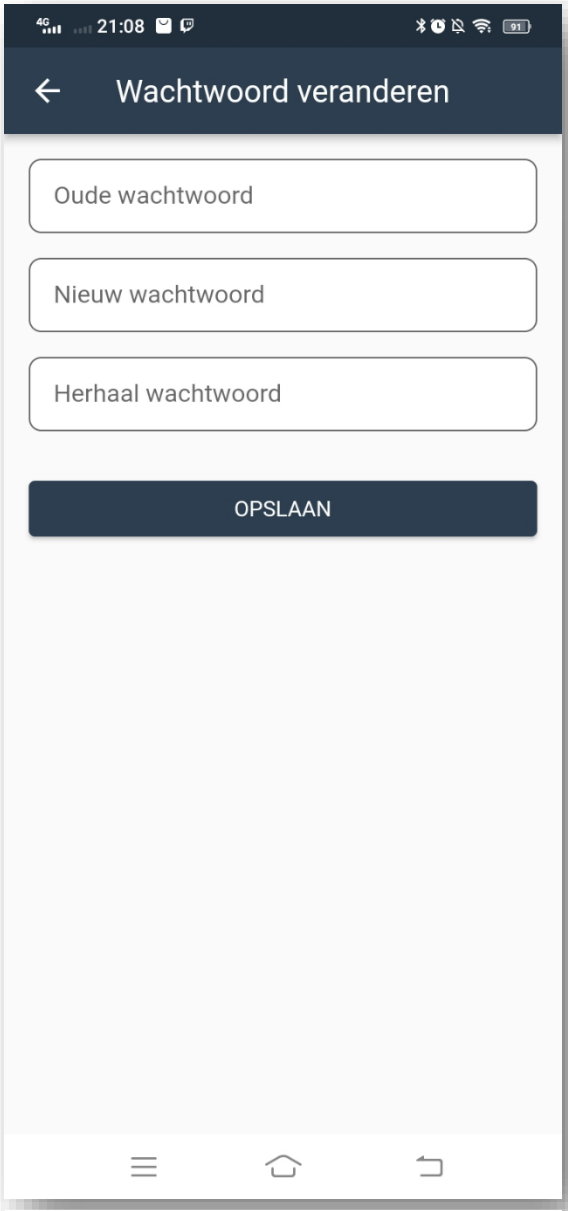


Hình 62: Màn hình chỉnh sửa hồ sơ công ty, danh sách nhân viên và mã QR đăng ký

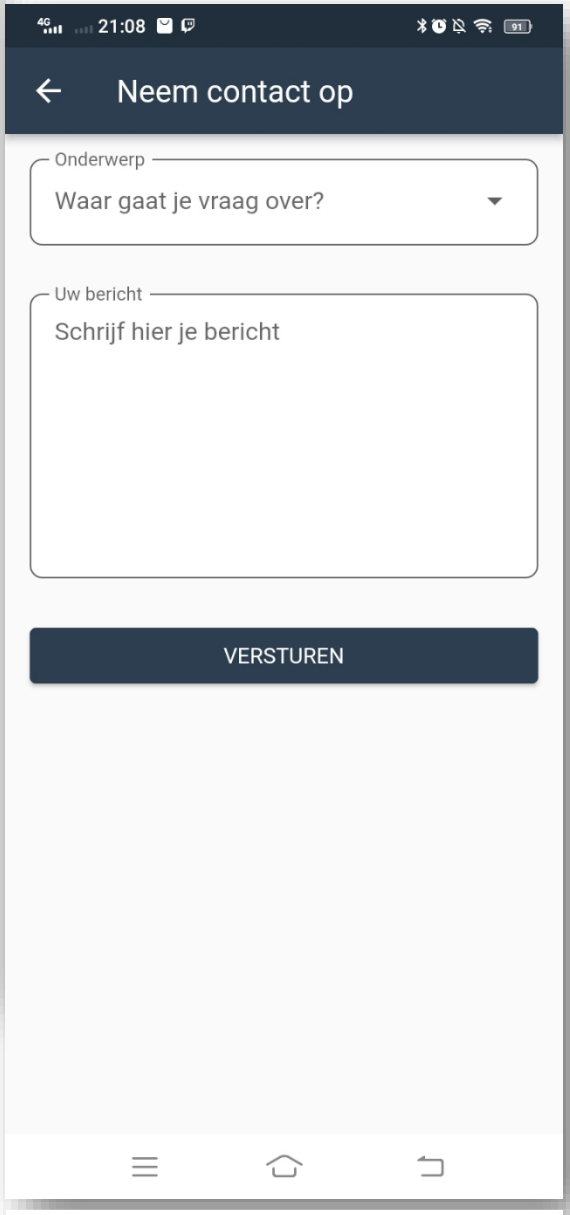
3.6.3.10. Màn hình đổi mật khẩu và gửi đơn hỗ trợ.

Từ màn hình Profile, người dùng có thể tới màn hình Change Passoword Screen để thực hiện chức năng thay đổi một khẩu.

Cũng từ màn hình Profile người dùng có thể tới màn hình Contact để gửi đơn hỗ trợ tới nhà phát triển ứng dụng

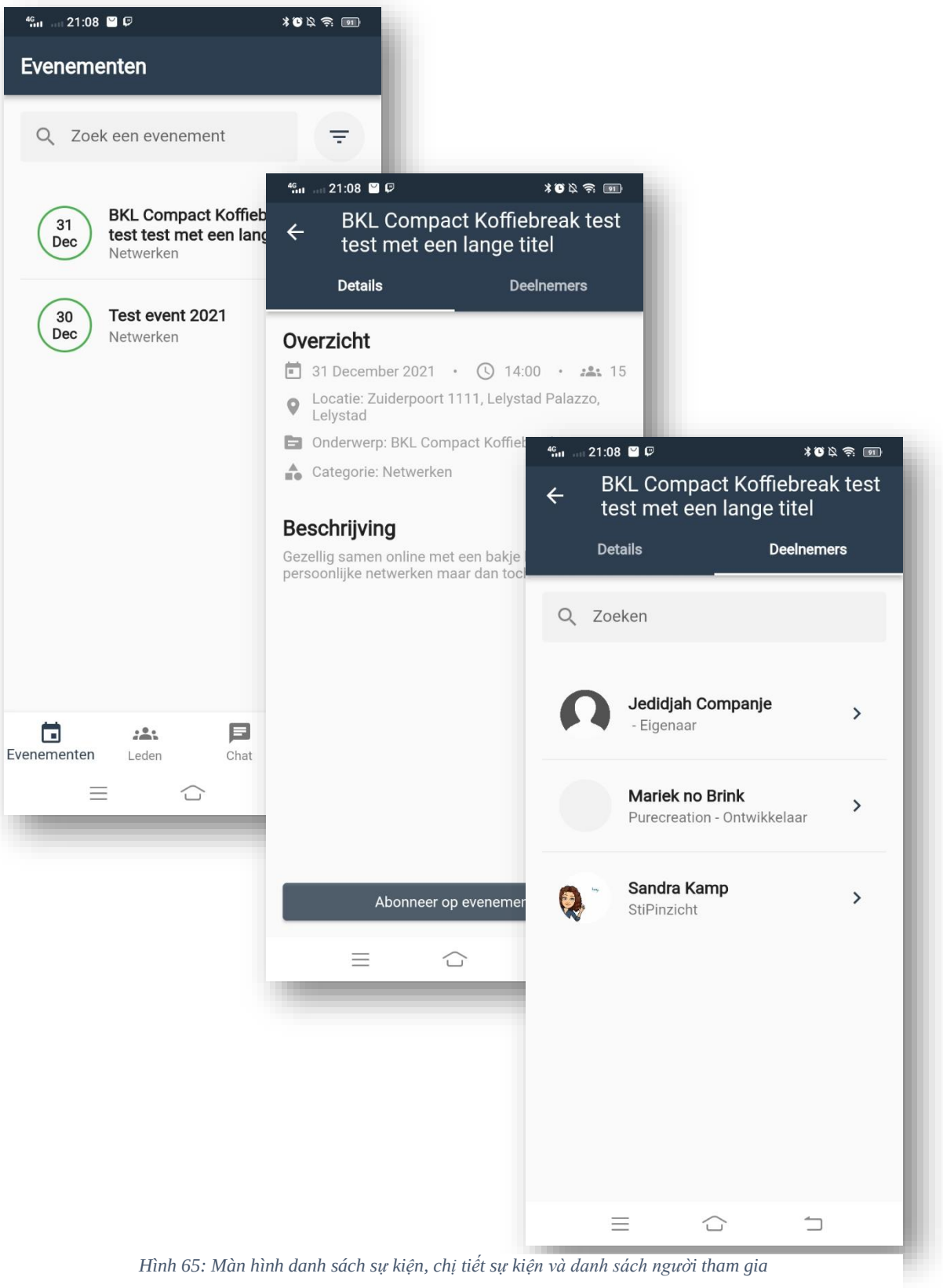


Hình 64: Màn hình đổi mật khẩu



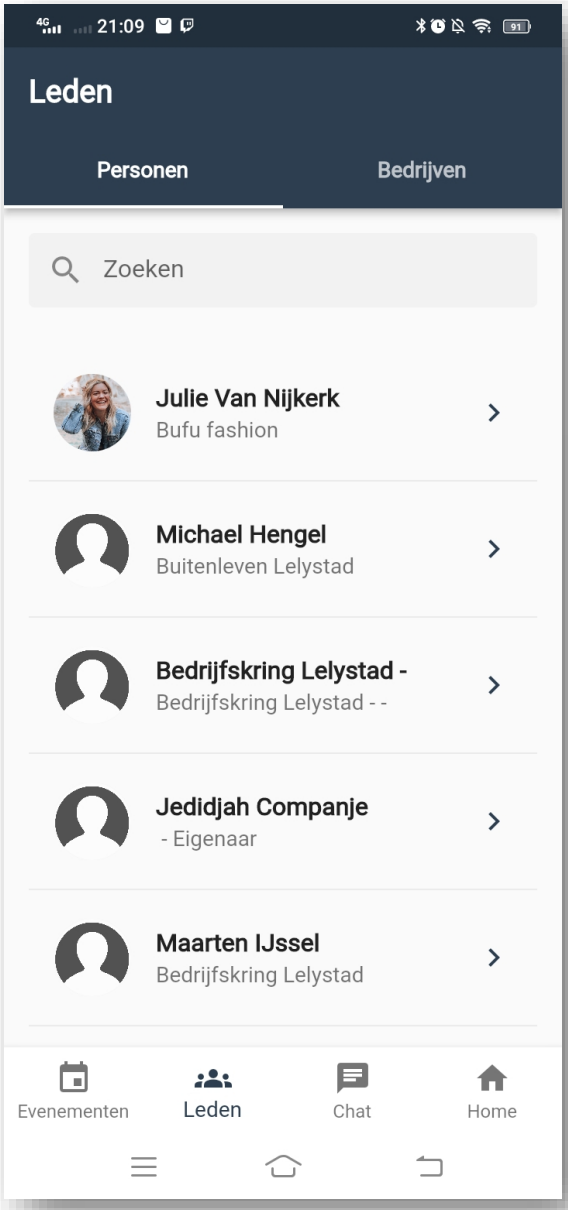
Hình 63: Màn hình gửi đơn liên hệ

3.6.3.11. Màn hình danh sách sự kiện và chi tiết sự kiện

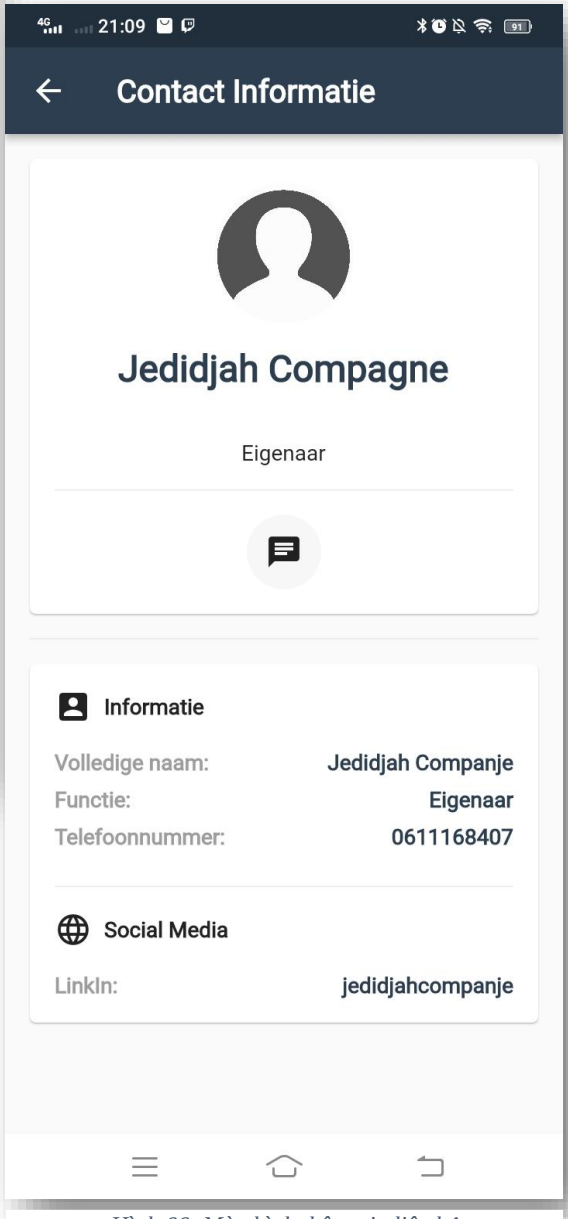


Hình 65: Màn hình danh sách sự kiện, chị tiết sự kiện và danh sách người tham gia

3.6.3.12. Màn hình danh sách người dùng thông tin liên hệ

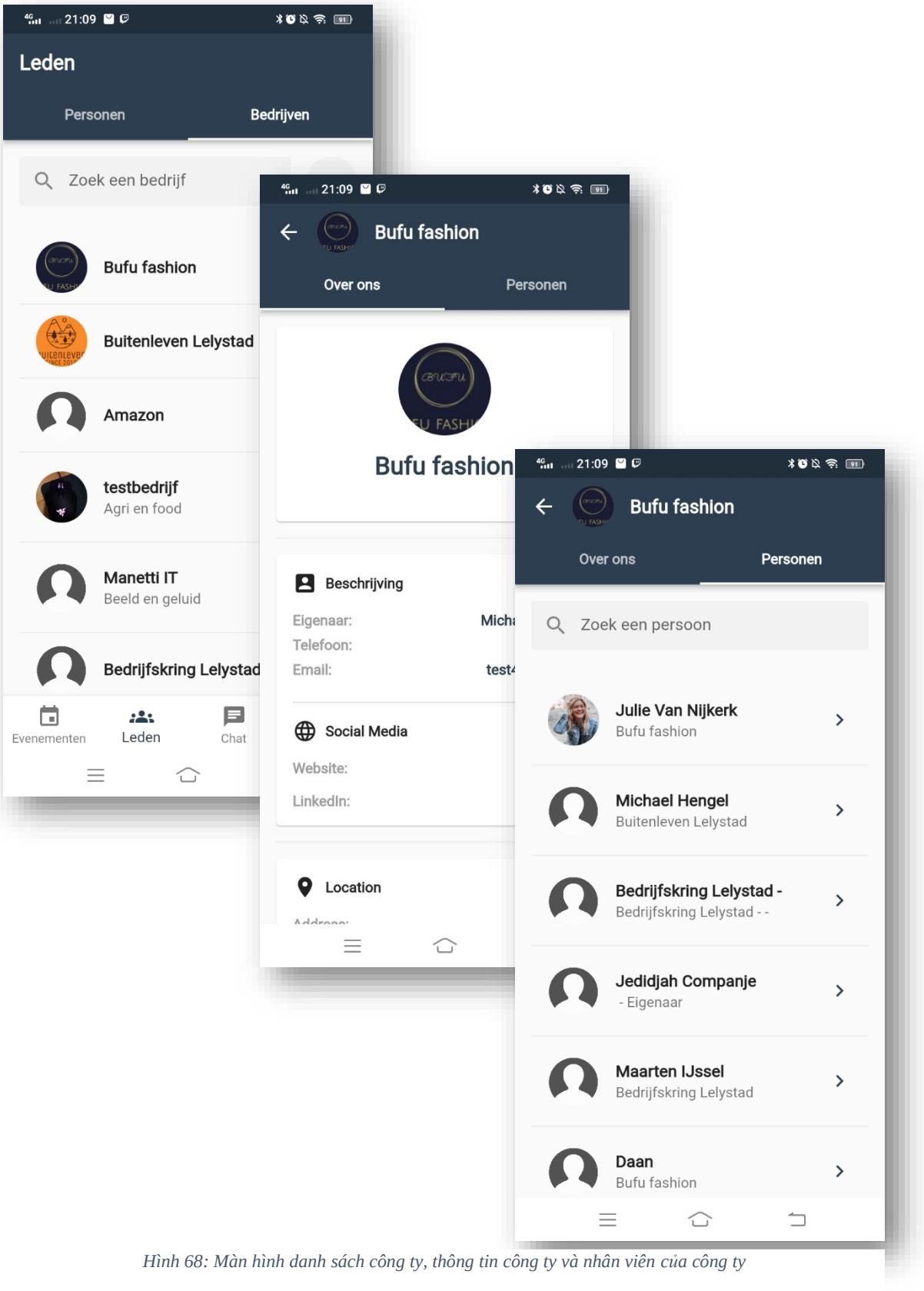


Hình 67: Màn hình danh sách người dùng trong hệ thống



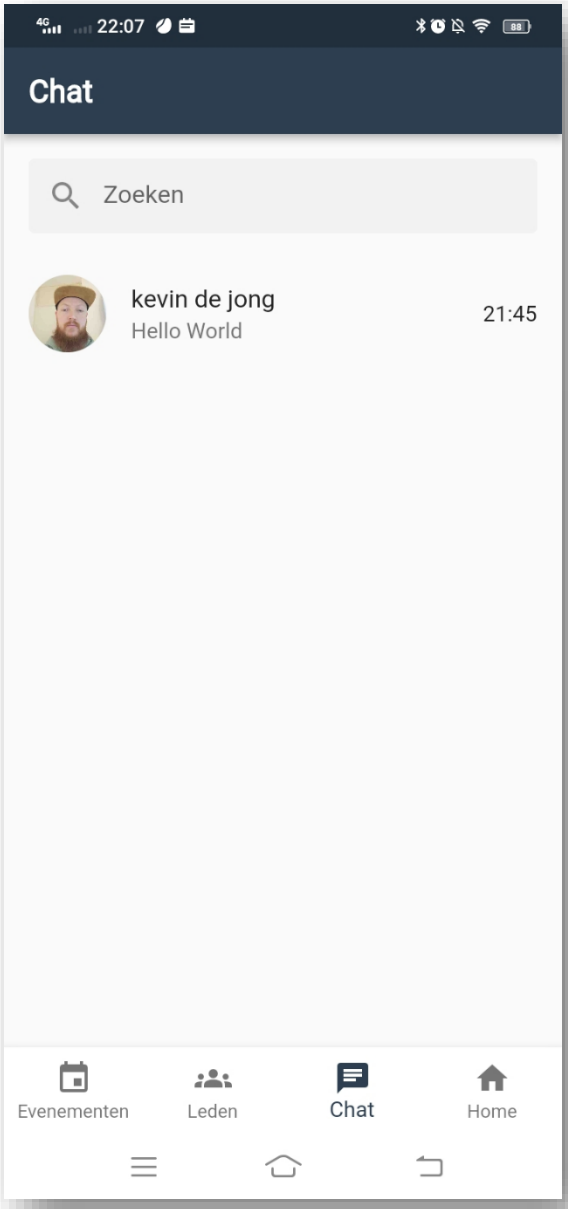
Hình 66: Màn hình thông tin liên hệ

3.6.3.13. Màn hình danh sách doanh nghiệp và thông tin của doanh nghiệp trong hệ thống

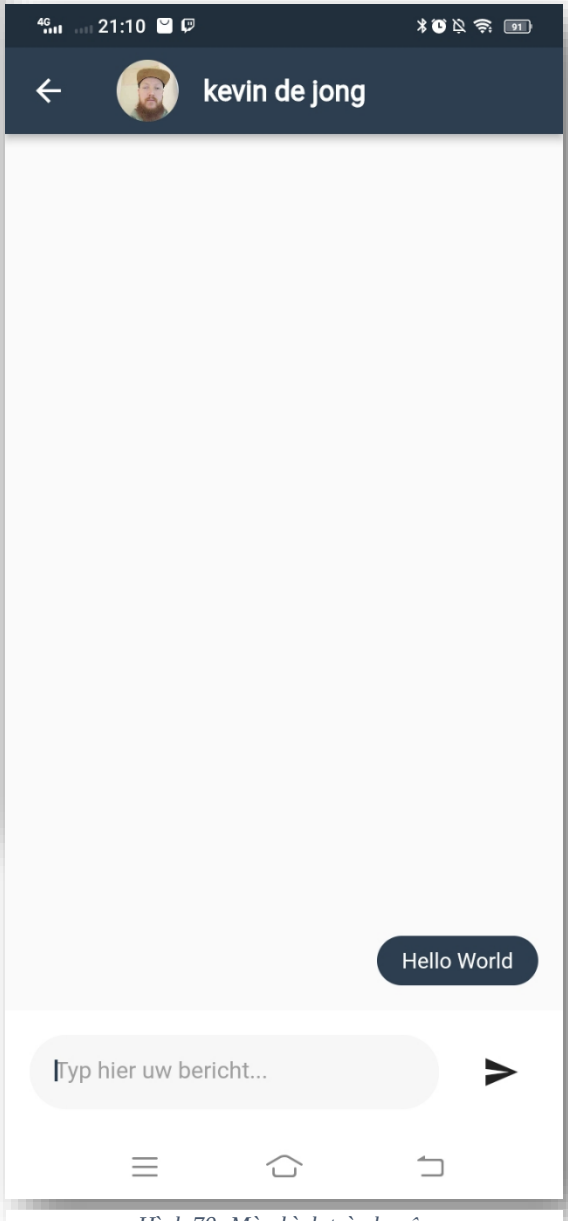


Hình 68: Màn hình danh sách công ty, thông tin công ty và nhân viên của công ty

3.6.3.14. Màn hình danh sách các nhóm trò chuyện và trò chuyện



Hình 69: Màn hình danh sách nhóm trò chuyện



Hình 70: Màn hình trò chuyện

CHƯƠNG 4: KẾT LUẬN

4.1. Kết quả đạt được

Dự án BKL đã giúp tôi nâng cao được kinh nghiệm làm việc nhóm và kiến thức chuyên môn về Flutter.

Qua việc giao tiếp và làm việc cho cả đội Việt Nam và Hà Lan, giúp tôi sắp xếp thời gian và công việc hiệu quả, đồng thời cải thiện kỹ năng giao tiếp và trình độ ngoại ngữ.

Với các yêu cầu của khách hàng, đã giúp ôn lại những kiến thức về Notification và đồng thời nâng cao kiến thức về nó. Đặc biệt là khi thực hiện Chức năng thông báo đẩy Push Notification và Real-time chat với FCM.

4.2. Hướng phát triển

Trong quá trình phát triển ứng dụng BKL, đã có những điểm không đồng bộ và tối ưu trong cách lưu trữ dữ liệu, quản lý dữ liệu và quản lý State của hệ thống.

Chưa thể phát huy tối ưu kiến trúc DDD theo như ý muốn, dẫn đến một số các vấn để và lỗi phát sinh nhỏ trong quá trình xây dựng.

Qua đó, tôi sẽ tiếp tục nghiên cứu và áp dụng kiến trúc DDD vào dự án và tối ưu hóa khái niệm và các đối tương liên kết trong dự án để tạo nên một cấu trúc chặt chẽ nhưng dễ thay đổi, thích nghi và dễ hiểu trong tương lai

4.3. Tài liệu tham khảo

- https://flutter.dev/
- https://material.io/design
- https://pub.dev/