TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric

LÊ ĐỨC MẠNH
manh.ld176814@sis.hust.edu.vn

Ngành Công nghệ thông tin và truyền thông
Chuyên ngành Hệ thống thông tin

Giảng viên hướng dẫn: PGS.TS Nguyễn Bình Minh _____

Chữ kí GVHD

Khoa: Công nghệ thông tin

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 07/2022

LỜI CẢM ƠN

Lời mở đầu, em xin cảm ơn trường Đại học Bách khoa Hà Nội đã tạo một môi trường học tập và nghiên cứu lý tưởng. Em tin rằng những kiến thức chuyên sâu, những trải nhiệm quý báu có được tại đây sẽ là hành trang vững chãi trên con đường đời phía sau mái trường Đại học.

Em muốn gửi lời cảm ơn đầu tới các thầy cô viện Công nghệ Thông tin và Truyền Thông đã chỉ dạy trong 5 năm học vừa qua. Đặt biệt em xin cảm ơn PGS.TS Nguyễn Bình Minh và Th.S Đinh Hữu Hải Quân đã tận tình hướng dẫn, tạo điều kiện tốt tốt nhất để em có thể hoàn thành quyển đồ án tốt nghiệp này.

Cảm ơn những người bạn chân tình đã cùng nhau chia sẻ biết bao vui buồn. Tôi sẽ luôn trân trọng những kỷ niệm về quãng đời sinh viên không thể quên này.

Lời cuối cùng, xin cảm ơn gia đình đã luôn ủng hộ, hỗ trợ con. Cảm ơn bố và mẹ đã nuôi con trưởng thành và quan tâm chăm sóc con. Hai người luôn là điểm tựa vững chãi cuộc đời con.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Công nghệ chuỗi khối nổi lên gần đây với khả năng cho phép các giao dịch có giá trị lớn và quan trọng có thể được thực hiện một cách an toàn và chính xác mà không yêu cầu có sự xuất hiện và can thiệp của một bên thứ ba trung gian. Với lợi thế này, việc ứng dụng công nghệ chuỗi khối vào thực tiễn sẽ giúp nâng cao cải thiện hiệu suất công việc cũng như tiết kiệm chi phí vận hành cho các nghiệp vụ. Thế nhưng do là một công nghệ mới và có độ phức tạp cao, việc ứng dụng công nghệ chuỗi khối vào các nghiệp vụ doanh nghiệp là một quá trình yêu cầu nhiều công sức và thời gian. Mục tiêu của đồ án này là phát triển một hệ thống hỗ trợ đơn giản hóa nhất có thể quá trình ứng dụng này thông qua cơ chế tự động triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric - một nền tảng chuỗi khối mạnh mẽ sở hữu nhiều tính năng phù hợp với các hoạt động kinh doanh.

"Hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric" cho phép người dùng xây dựng và phát triển một cơ sở hạ tầng mạng và các ứng dụng phi tập trung để ứng dụng trong nghiệp vụ thông qua các thông số và cấu hình tổng quan. Hạ tầng mạng sẽ được triển khai trên một nền tảng Kubernetes để đảm bảo tính ổn định cùng với khả năng mở rộng dễ dàng. Từ một mô hình cơ sở dữ liệu quan hệ, một ứng dụng phi tập trung cũng sẽ được tự động viết và triển khai lên mạng. Ngoài ra việc tương tác với ứng dụng được đơn giản hóa thông qua một bộ SDK được tạo ra bởi hệ thống.

Hệ thống được triển khai theo kiến trúc Microservices với nhiều dịch vụ nhỏ nhằm đảm bảo khả năng mở rộng. Các dịch vụ này được phát triển sử dụng ngôn ngữ lập trình Python và giao diện sử dụng thư viện ReactJS. Đề tài V-chain[1] đã sử dụng hệ thống trong đồ án này như một dịch vụ xử lý các tác vụ liên quan đến mạng và ứng dụng thuộc nền tảng Hyperledger Fabric.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 1

1.1 Đặt vấn đề 1

1.2 Mục tiêu và phạm vi đề tài 2

1.2.1 Giải pháp liên quan 2

1.2.2 Mục tiêu và phạm vi 2

1.3 Định hướng giải pháp 3

1.3.1 Triển khai mạng 3

1.3.2 Triển khai ứng dụng phi tập trung 3

1.4 Bố cục đồ án 3

CHƯƠNG 2. HYPERLEDGER FABRIC 4

2.1 Tổng quan 4

2.2 Kiến trúc mạng Hyperledger Fabric 4

2.3 Khởi tạo một mạng Hyperledger Fabric 5

2.3.1 Bước 1: Khởi tạo Orderer Node 5

2.3.2 Bước 2: Thêm Tổ chức quản trị 5

2.3.3 Bước 3: Định nghĩa Consortium 6

2.3.4 Bước 4: Tạo kênh tương ứng với Consortium 6

2.3.5 Bước 5: Thêm Peer node 7

2.3.6 Bước 6: Cài đặt Hợp đồng thông minh 7

2.3.7 Bước 7: Khởi tạo mạng hoàn tất 8

CHƯƠNG 3. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU 10

3.1 Tổng quan chức năng 10

3.1.1 Biểu đồ use case tổng quan 10

3.2 Đặc tả chức năng 10
3.2.1 Đặc tả use case Tạo mạng 11
3.2.2 Đặc tả use case Thêm node ngoài 12
3.2.3 Đặc tả use case Thêm tổ chức 13
3.2.4 Đặc tả use case Xóa mạng 14
3.2.5 Đặc tả use case Tạo Ứng dụng phi tập trung 15
3.2.6 Đặc tả use case Tải SDK 16
3.2.7 Đặc tả use case Cập nhật Ứng dụng phi tập trung 17
3.2.8 Đặc tả use case Xóa Ứng dụng phi tập trung 18
3.3 Yêu cầu phi chức năng 18
CHƯƠNG 4. CÔNG NGHỆ SỬ DỤNG 19
4.1 Công nghệ triển khai mạng và ứng dụng 19
4.1.1 Container 19
4.1.2 Kubernetes 19
4.1.3 Dịch vụ điến toán đám mây 20
4.2 Công nghệ xây dựng hệ thống 20
4.2.1 MongoDB 20
4.2.2 RabbitMQ 20
4.2.3 Celery 20
CHƯƠNG 5. THỰC NGHIỆM VÀ ĐÁNH GIÁ 22
5.1 Thiết kế tổng quan 22
5.1.1 Kiến trúc tổng quan 22
5.1.2 Kiến trúc tổng quan cơ sở dữ liệu 23
5.2 Thiết kế chi tiết 24
5.2.1 Thiết kế giao diện 24
5.2.2 Thiết kế Account Service 26

5.2.3 Thiết kế Core Service 28
5.2.4 Thiết kế Network Service 32
5.2.5 Thiết kế Network Worker 33
5.2.6 Thiết kế Dapp Service 35
5.2.7 Thiết kế Dapp Worker 36
5.2.8 Các luồng hoạt động chính 37
5.3 Xây dựng hệ thống 39
5.3.1 Thư viện và công cụ sử dụng 39
5.3.2 Kết quả đạt được 41
5.3.3 Minh họa các chức năng chính 41
5.4 Kiểm thử 43
5.4.1 Kiểm thử hệ thống 43
5.4.2 Kiểm thử SDK 45
5.5 Ứng dụng của hệ thống 47
5.5.1 Tổng quan ứng dụng 47
5.5.2 Cấu hình 47
5.5.3 Luồng hoạt động 48
5.5.4 Minh họa hoạt động 49
CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 51
6.1 Kết luận 51
6.2 Hướng phát triển 51
TÀI LIỆU THAM KHẢO 52

DANH MỤC HÌNH VẼ

Hình 2.1 Kiến trúc một mạng Hyperledger Fabric cơ bản 4
Hình 2.2 Bước 1: Khởi tạo Orderer Node 5
Hình 2.3 Bước 2: Thêm Tổ chức quản trị 6
Hình 2.4 Bước 3: Định nghĩa Consortium 6
Hình 2.5 Bước 4: Tạo kênh tương ứng với Consortium 7
Hình 2.6 Bước 5: Thêm Peer node 7
Hình 2.7 Bước 6: Cài đặt Hợp đồng thông minh 8
Hình 2.8 Bước 7: Khởi tạo mạng hoàn tất 8

Hình 3.1 Biểu đồ usecase Tổng quan 10

Hình 5.1 Tổng quan các dịch vụ của hệ thống 22
Hình 5.2 Cơ sở dữ liệu tổng quan 23
Hình 5.3 Màn hình danh sách mạng 24
Hình 5.4 Màn hình thông tin mạng 24
Hình 5.5 Màn hình danh sách ứng dụng 25
Hình 5.6 Màn hình thông tin ứng dụng 25
Hình 5.7 Màn hình tạo ứng dụng 26
Hình 5.8 Biểu đồ lớp Account Service 26
Hình 5.9 Biểu đồ lớp Core Service 28
Hình 5.10 Biểu đồ lớp Network Service 32
Hình 5.11 Biểu đồ lớp Network Worker 33
Hình 5.12 Biểu đồ lớp Dapp Service 35
Hình 5.13 Biểu đồ lớp Dapp Worker 36
Hình 5.14 Biểu đồ trình tự tạo mạng 37
Hình 5.15 Biểu đồ trình tự tạo ứng dụng phi tập trung 38
Hình 5.16 Biểu đồ trình tự thêm node ngoài vào mạng 39
Hình 5.17 Màn hình danh sách mạng 41
Hình 5.18 Màn hình tạo mạng mới 41
Hình 5.19 Màn hình thông tin mạng 1 42
Hình 5.20 Màn hình thông tin mạng 2 42
Hình 5.21 Màn hình danh sách ứng dụng 42
Hình 5.22 Màn hình thiết kế cấu trúc ứng dụng 43
Hình 5.23 Màn hình thông tin ứng dụng 43
Hình 5.24 Mô-đun kiểm thử SDK 45
Hình 5.25 Cấu trúc ứng dụng để kiểm thử SDK 45

Hình 5.26 Cấu hình mạng triển khai Fabric Contract 47
Hình 5.27 Cấu hình mạng triển khai Fabric Contract 47
Hình 5.28 Luồng tạo hợp đồng của Fabric Contract 48
Hình 5.29 Luồng ký hợp đồng của Fabric Contract 48
Hình 5.30 Màn hình danh sách hợp đồng Fabric Contract 49
Hình 5.31 Màn hình chi tiết hợp đồng Fabric Contract 49
Hình 5.32 Giao dịch ký hợp đồng trên mạng chuỗi khối 50

DANH MỤC BẢNG BIỂU

Bảng 3.1 Đặc tả ca sử dụng "Tạo mạng" 11
Bảng 3.2 Đặc tả ca sử dụng "Thêm Node ngoài" 12
Bảng 3.3 Đặc tả ca sử dụng "Thêm tổ chức" 13
Bảng 3.4 Đặc tả ca sử dụng "Xóa mạng" 14
Bảng 3.5 Đặc tả ca sử dụng "Tạo Ứng dụng phi tập trung" 15
Bảng 3.6 Đặc tả ca sử dụng "Tải SDK" 16
Bảng 3.7 Đặc tả ca sử dụng "Cập nhật Ứng dụng phi tập trung" 17
Bảng 3.8 Đặc tả ca sử dụng "Xóa Ứng dụng phi tập trung" 18

Bảng 5.1 Mô tả chi tiết các lớp của Account service 26
Bảng 5.2 Chi tiết cơ sở dữ liệu của Account service 27
Bảng 5.3 Mô tả chi tiết các lớp của Core Service 28
Bảng 5.4 Chi tiết cơ sở dữ liệu của Core Service 30
Bảng 5.4 Chi tiết cơ sở dữ liệu của Core Service 31
Bảng 5.5 Mô tả chi tiết các lớp của Network service 32
Bảng 5.5 Mô tả chi tiết các lớp của Network service 33
Bảng 5.6 Mô tả chi tiết các lớp của Network Worker 33
Bảng 5.6 Mô tả chi tiết các lớp của Network Worker 34
Bảng 5.7 Mô tả chi tiết các lớp của Dapp service 35
Bảng 5.8 Mô tả chi tiết các lớp của Dapp Worker 36
Bảng 5.8 Mô tả chi tiết các lớp của Dapp Worker 37
Bảng 5.9 Danh sách thư viện và công cụ sử dụng 39
Bảng 5.9 Danh sách thư viện và công cụ sử dụng 40
Bảng 5.10 Kết quả kiểm thử tác vụ tạo mạng 44
Bảng 5.11 Kết quả kiểm thử tác vụ tạo ứng dụng phi tập trung 44
Bảng 5.12 Kết quả kiểm thử các tác vụ đọc dữ liệu sử dụng SDK 46
Bảng 5.13 Kết quả kiểm thử các tác vụ ghi dữ liệu sử dụng SDK 46

TỪ VIẾT TẮT

Viết tắt	Tên tiếng Anh	Tên tiếng Việt
API	Application Programming Interface	Giao diện lập trình ứng dụng
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
HTTP	Hyper Text Transfer Protocol	Giao thức truyền tải trên web
REST	REpresentational State Transfer	Tiêu chuẩn thiết kế API cho HTTP Server
DAPP	Decentralized Application	Ứng dụng phi tập trung
K8S	Kubernetes	Nền tảng quản lý container

THUẬT NGỮ

Thuật ngữ	Ý nghĩa
Node	Nút trong mạng
Service	Dịch vụ
Blockchain	Công nghệ/mạng chuỗi khối

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Công nghệ chuỗi khối (blockchain) được cho là một trong những công nghệ cốt lõi trong cuộc cách mạng công nghiệp lần thứ 4. Tính đột phá của công nghệ chuỗi khối nằm ở việc cho cho phép những giao dịch có giá trị lớn được thực hiện một cách minh bạch, chính xác, công bằng mà không cần đến sự chứng thực và phân xử của một bên thứ ba. Cốt lõi của công nghệ chuỗi khối nằm ở một cuốn sổ cái kỹ thuật số và các ứng dụng phi tập trung. Cuốn sổ cái cơ bản là một chuỗi các khối dữ liệu (nên có tên là chuỗi khối) được lưu trữ phân tán trên nhiều máy. Dữ liệu trong cuốn sổ cái này là không thể giả mạo và một khi đã được thêm mới vào thì sẽ không thể thay đổi hay xóa bỏ. Ứng dụng phi tập là một đoạn mã được lập trình để thực hiện thêm mới dữ liệu vào sổ cái. Đoạn mã này có thể được thực thi một cách tự động với tính chính xác cao. Dữ liệu được cập nhật bởi ứng dụng phi tập trung sẽ đảm bảo tính minh bạch cao cùng khả năng truy xuất nhờ vào đặc tính không thể bị can thiệp hay sửa đổi của sổ cái.

Những mạng chuỗi khối nổi tiếng nhất như Bitcoin và Ethereum là mạng chuỗi khối công khai (public blockchain). Đối với loại mạng này, mọi người đều có thể tham gia, thực hiện tương tác hay truy vấn dữ liệu. Tuy nhiên, đối với một số nghiệp vụ, việc thông tin giao dịch và số liệu hoạt động của các doanh nghiệp bị công khai có thể là một vấn đề lớn. Do vậy việc sử dụng loại mạng công khai vào trong nghiệp vụ giữa các tổ chức, liên doanh đôi khi là bất khả thi. Để giải quyết vấn đề này mạng chuỗi khối riêng tư (private blockchain) ra đời. Để có thể được ứng dụng vào các hoạt động giữa các doanh nghiệp, ngoài tính phân tán, bảo mật và minh bạch dữ liệu, mạng riêng tư thường thêm có các đặc điểm sau: (i) Danh tính của những bên tham gia vào mạng cần phải xác thực được.(ii) Không phải ai cũng có thể tham gia, truy vấn dữ liệu hay tương tác với mạng, chỉ các cá nhân, tổ chức có đủ quyền mới có thể thực hiện các hành động này. (iii) Tốc độ xử lý giao dịch cao hơn nhiều so với mạng chuỗi khối công khai.

Với những ưu điểm mà mạng chuỗi khối riêng tư mang lại, việc các doanh nghiệp ứng dụng mạng riêng tư vào nghiệp vụ chắc chắn có thể giúp nâng cao hiệu suất công việc. Tuy vậy quá trình ứng dụng này thường gặp phải các khó khăn sau. Đầu tiên, do là mạng riêng tư, cơ sở hạ tầng mạng sẽ cần phải được quản lý riêng biệt. Mỗi một mạng chuỗi khối sẽ có một kiến trúc riêng. Doanh nghiệp muốn sử dụng sẽ cần nghiên cứu kiến trúc đó rồi triển khai một mạng lên một cơ sở hạ tầng cụ thể. Không chỉ vậy, cơ sở hạ tầng cũng cần được cân nhắc để đảm bảo tính ổn

định và khả năng chịu lỗi để có thể được sử dụng lâu dài. Cũng giống như kiến trúc mạng, ứng dụng phi tập trung trên mạng chuỗi khối riêng tư cũng có một cấu trúc và cách thức hoạt động riêng biệt. Do vậy việc phát triển và ứng dụng các ứng dụng phi tập trung phục vụ cho nghiệp vụ cũng là một thử thách. Nhà phát triển cần tìm hiểu cách hoạt động rồi từ cách hoạt động đó phát triển một ứng dụng mới đáp ứng được nhu cầu nghiệp vụ. Thử thách này còn trở nên đặc biệt khó khăn đối với những ai chưa từng tiếp xúc với công nghệ chuỗi khối.

Có thể thấy, để ứng dụng những đặc tính nổi bật của mạng chuỗi khối riêng tư vào các nghiệp vụ thực tiễn yêu cầu những kiến thức đặc thù cùng nhiều nhân lực và thời gian. Nhận thấy vấn đề này, tôi đã quyết định phát triển một hệ thống triển khai mạng cùng với đó là ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric - một nền tảng mạng chuỗi khối riêng tư rất phổ biến hiện nay. Thông qua giao diện trực quan của hệ thống, một lập trình viên dù cho không có kiến thức về mạng chuỗi khối cũng có thể dễ dàng triển khai hạ tầng mạng và các ứng dụng phi tập trung phục vụ cho nhiều yêu cầu nghiệp vụ khác nhau.

1.2 Mục tiêu và phạm vi đề tài

1.2.1 Giải pháp liên quan

Ông lớn Amazon cung cấp một giải pháp để hỗ trợ quá trình triển khai mạng Hyperledger Fabric - Amazon Managed Blockchain[2]. Với thế mạnh về điện toán đám mây của mình, thông qua dịch vụ Managed Blockchain, người dùng có thể triển khai một mạng Hyperledger Fabric cho riêng mình trên cơ sở hạ tầng của Amazon chỉ với vài cái nhấp chuột. Quá trình quản lý và theo dõi hoạt động của mạng cũng được đơn giản hóa, tối ưu trải nhiệm người dùng.

Tuy quy trình khởi tạo và quản trị mạng có thể được thực hiện thông qua giao diện trực quan, việc phát triển ứng dụng phi tập trung phục vụ cho các nghiệp vụ trên mạng này lại không được như vậy. Người dùng phải tự mình lập trình và chạy các câu lệnh trên terminal để triển khai ứng dụng đó lên mạng. Do vậy quá trình phát triển ứng dụng phi tập trung vẫn sẽ yêu cầu kiến thức chuyên sâu về Hyperledger Fabric. Lập trình viên vẫn sẽ cần phải nghiên cứu để viết và triển khai hợp đồng thông minh, một quá trình tiêu tốn nhiều thời gian lẫn công sức.

1.2.2 Mục tiêu và phạm vi

Với mục tiêu đơn giản hóa nhất có thể quá trình ứng dụng mạng chuỗi khối riêng tư vào các nghiệp vụ doanh nghiệp, hệ thống trong đồ án này hướng đến việc cho phép những lập trình viên dù cho không có kiến thức về mạng chuỗi khối cũng có thể dễ dàng triển khai mạng cùng với đó là ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric. Để đạt được điều này, 2 mục tiêu sau được đề ra:

- Cho phép cấu hình và triển khai một mạng Hyperledger Fabric thông qua giao diện trực quan.
- Cho phép thiết kế và triển khai ứng dụng phi tập trung thông qua giao diện trực quan. Cung cấp phương thức để có thể tương tác với hợp đồng thông minh đó mà không yêu cầu kiến thức đặc thù.

1.3 Định hướng giải pháp

Để giải quyết vấn đề đã nêu ở mục 1.1, hệ thống sẽ hỗ trợ hai chức năng chính cho người dùng.

1.3.1 Triển khai mạng

Cơ sở hạ tầng để triển khai mạng cần được suy tính kỹ lượng để đảm bảo mạng hoạt động ổn định và hiệu quả nhất. Để đơn giản hóa quá trình triển khai và quản lý mạng Hyperledger Fabric, người dùng sẽ được phép tùy chỉnh cấu hình thông qua một giao diện. Sau đó, hệ thống sẽ triển khai một mạng với cấu hình tương ứng lên điện toán đám mây. Ngươi dùng sẽ không cần chú ý quá nhiều đến chi tiết phần cứng mà chỉ cần quan tâm đến cấu hình của cơ sở hạ tâng.

1.3.2 Triển khai ứng dụng phi tập trung

Điều quan trọng nhất trong việc triển khai ứng dụng là thiết kế kiến trúc sao cho phù hợp với nghiệp vụ. Để bao quát được nhiều nghiệp vụ nhất có thể, hệ thống sẽ cho phép người dùng triển khai ứng dụng dựa trên mô hình cơ sở dữ liệu quan hệ. Nhiều thực thể sở hữu các thuộc tính khác nhau có thể được định nghĩa. Các thực thể này có thể có nhiều liên kết với nhau (quan hệ một-một, một-nhiều, nhiều-nhiều). Dựa vào kiến trúc tổng quan của các thực thể này, một ứng dụng phi tập trung tương ứng sẽ được tự động sinh ra. Sau khi được triển khai, người dùng có thể tải một bộ SDK được hệ thống sinh ra về để tương tác với mạng chuỗi khối thông qua các hàm đọc ghi sửa xóa các thực thể trên. Quá trình hình thành và thay đổi của các thực thể này sẽ được lưu lại vĩnh viễn, đảm bảo việc xác thực và truy vấn dữ liệu nghiệp vụ thông qua mạng chuỗi khối.

1.4 Bố cục đồ án

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

- Chương 2: Giới thiệu kiến trúc mạng chuỗi khối Hyperledger Fabric.
- Chương 3: Khảo sát và phân tích yêu cầu.
- Chương 4: Trình bày về các công nghệ sử dụng.
- Chương 5: Trình bày về xây dựng và đánh giá hệ thống.
- Chương 6: Kết luận và định hương phát triển trong tương lai.

CHƯƠNG 2. HYPERLEDGER FABRIC

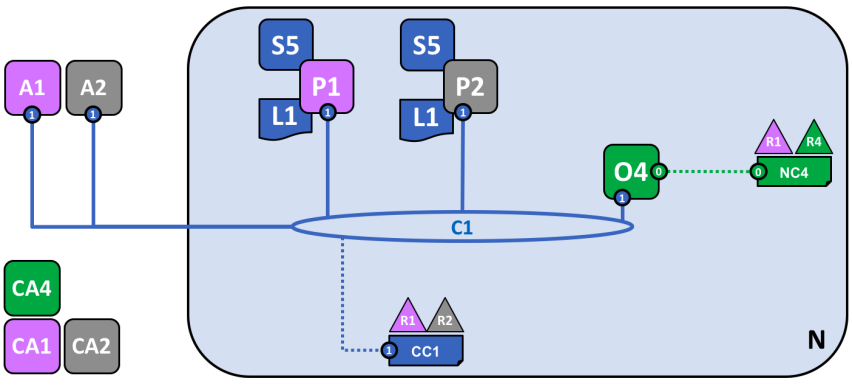
Để có thể triển khai một mạng Hyperledger Fabric, kiến thức về kiến trúc của mạng là yêu cầu tiên quyết. Chương này sẽ trình bày về tổng quan kiến trúc và các thành phần trong một mạng Hyperledger Fabric.

2.1 Tổng quan

Như đã giới thiệu, Hyperledger Fabric là một trong những nền tảng mạng chuỗi khối riêng tư nổi tiếng nhất. Mạng Hyperledger Fabric hỗ trợ các tính năng sau:

- **Quản lý định danh:** Hyperledger Fabric sử dụng hạ tầng khóa công khai và chữ ký số để xác thực, cấp quyền cho các tác nhân trong mạng.
- **Xử lý giao dịch tách biệt:** Để tăng khả năng xử lý song song, việc cập nhật giao dịch mới vào sổ cái và xác thực tính hợp lệ của các giao dịch được tách riêng biệt.
- **Chaincode:** Là ứng dụng phi tập trung (hợp đồng thông minh) mà thông qua đó có thể tương tác với sổ cái kỹ thuật số. Chaincode này có thể được viết bởi các ngôn ngữ lập trình phổ thông thay vì phải viết trên các ngôn ngữ lập trình chuyên biệt.
- **Tính mô-đun:** Các thành phần trong mạng có thể được tùy chỉnh, thay đổi.

2.2 Kiến trúc mạng Hyperledger Fabric



Hình 2.1: Kiến trúc một mạng Hyperledger Fabric cơ bản. Nguồn [3]

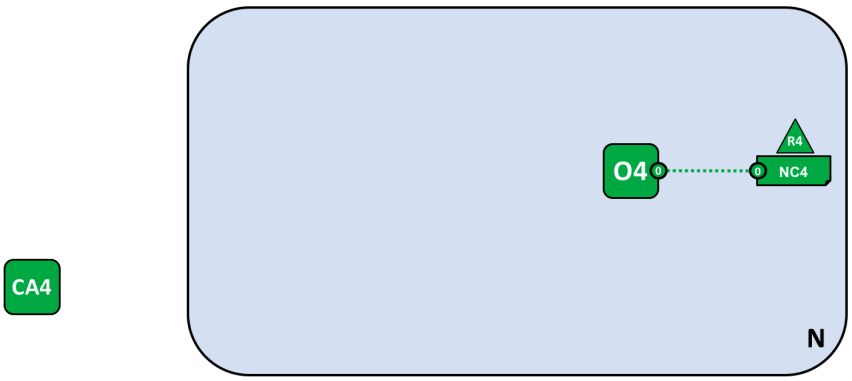
Các thành phần trong mạng ở Hình 2.1 bao gồm:

- **N:** Mạng (Network).
- **NC:** Cấu hình mạng (Network Configuration).
- **C:** Kênh (Channel): Tập trung các Tổ chức có chung mục đích kinh doanh.
- **CC:** Cấu hình kênh (Channel Configuration).

- **R:** Tổ chức (Organization).
- **O:** Orderer Node: Node duy nhất trong mạng xử lý quá trình đồng thuận.
- **P:** Peer Node: Nơi lưu trữ sổ cái kỹ thuật số. Các tương tác với hợp đồng thông minh (chaincode) đều phải thông qua Peer node.
- **S:** Hợp đồng thông minh (chaincode).
- **L:** Sổ cái kỹ thuật số (Ledger).
- **CA:** Nhà cung cấp chứng thực số (Certificate authority): Cung cấp danh tính cho tất cả các thành phần trong mạng.
- **A:** Ứng dụng hay giao diện (Application): Hỗ trợ tương tác hệ thống.

2.3 Khởi tạo một mạng Hyperledger Fabric

2.3.1 Bước 1: Khởi tạo Orderer Node

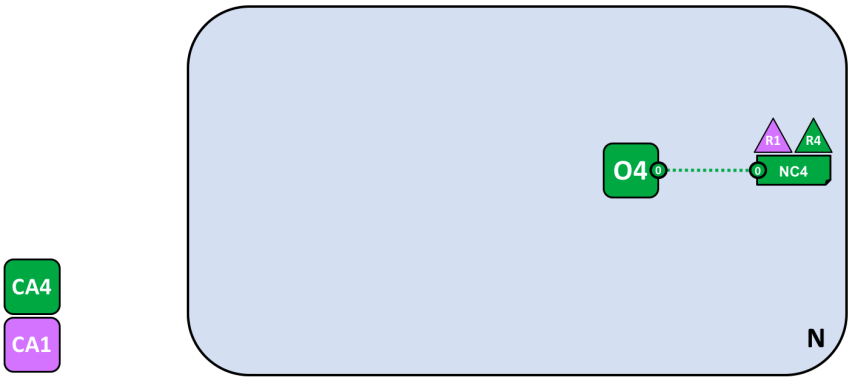


Hình 2.2: Bước 1: Khởi tạo Orderer Node. Nguồn [3]

Bước đầu tiên là khởi tạo một Orderer Node. Như ở Hình 2.2, để triển khai mạng N, trước nhất cần có Orderer Node O4 được khởi chạy và tùy chỉnh với cấu hình mạng NC4. Cấu hình mạng NC4 giao quyền quản trị mạng cho tổ chức R4. Là một mạng chuỗi khối riêng tư, việc định danh các tác nhân là bắt buộc. Nhà cung cấp chứng từ số CA4 sẽ cung cấp định danh cho các node và cá nhân thuộc tổ chức R4. O4 cũng trực thuộc tổ chức R4 và sẽ được R4 cung cấp cho một định danh. Tất cả các hành động sau này như thêm tổ chức vào vào mạng, thêm kênh, cài đặt chaincode cho kênh, khởi tạo chaincode, yêu cầu thực thi chaincode, . . . đều phải đi qua Orderer O4 này. Và trong Hyperledger Fabric, tất cả các hành động trên đều là giao dịch (transaction).

2.3.2 Bước 2: Thêm Tổ chức quản trị

Cấu hình mạng NC4 ban đầu chỉ cho phép Tổ chức R4 có quyền quản trị trên mạng. R4 có thể thêm R1 làm một tổ chức quản trị khác nữa trong mạng. Sau thời điểm này hai tổ chức R1 và R4 sẽ đều có quyền quản trị mạng tương đương nhau.

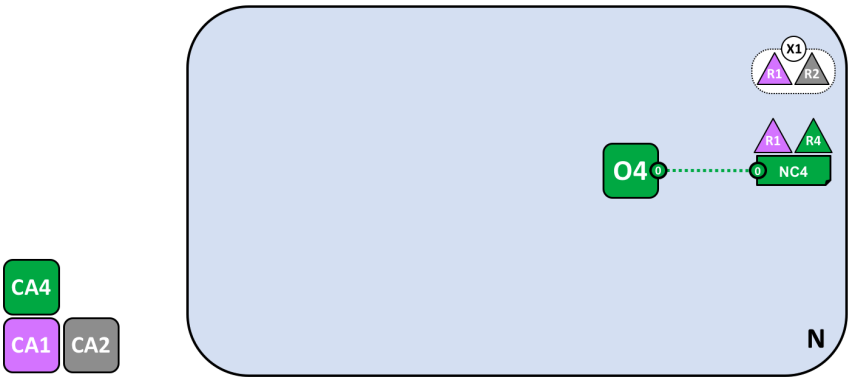


Hình 2.3: Bước 2: Thêm Tổ chức quản trị. Nguồn [3]

Tương tự như CA4, nhà cung cấp chứng từ số CA1 sẽ cung cấp chứng từ số cho các thực thể thuộc R1.

2.3.3 Bước 3: Định nghĩa Consortium

Hiện tại dù mạng có thể được quản trị bởi R1 và R4, vẫn chưa có nhiều nghiệp vụ có thể được thực hiện. Để có thể được ứng dụng vào các hoạt động kinh doanh, điều đầu tiên cần thực hiện là định nghĩa một Consortium. Consortium có thể hiểu là một liên doanh - kết nối nhiều tổ chức mà có hoạt động liên quan đến nhau.



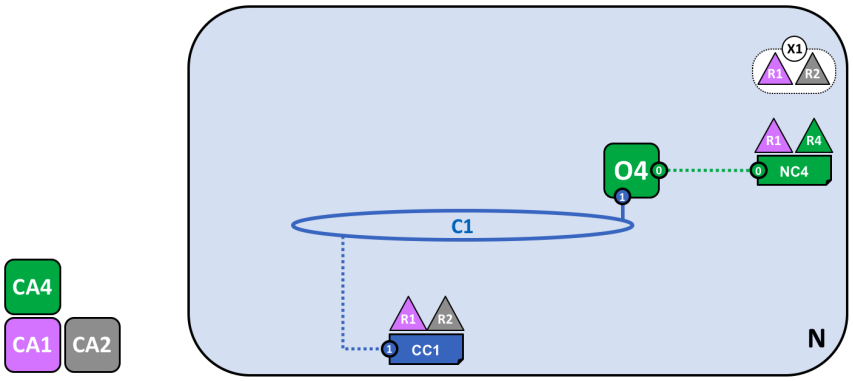
Hình 2.4: Bước 3: Định nghĩa Consortium. Nguồn [3]

Một quản trị viên mạng (R1 hoặc R4) định nghĩa một Consortium X1 với hai tổ chức, R1 và R2. Định nghĩa của consortium này được lưu trữ trong cấu hình mạng NC4 và sẽ được sử dụng ở các bước phát triển mạng tiếp theo. CA2 cũng được sử dụng để cung cấp định danh cho R2. Lưu ý là số lượng tổ chức trong một Consortium là tùy ý.

2.3.4 Bước 4: Tạo kênh tương ứng với Consortium

Kênh có thể được hiểu là một phương tiện truyền thông tin mà các tổ chức cùng chung một Consortium có thể sử dụng để giao tiếp với nhau.

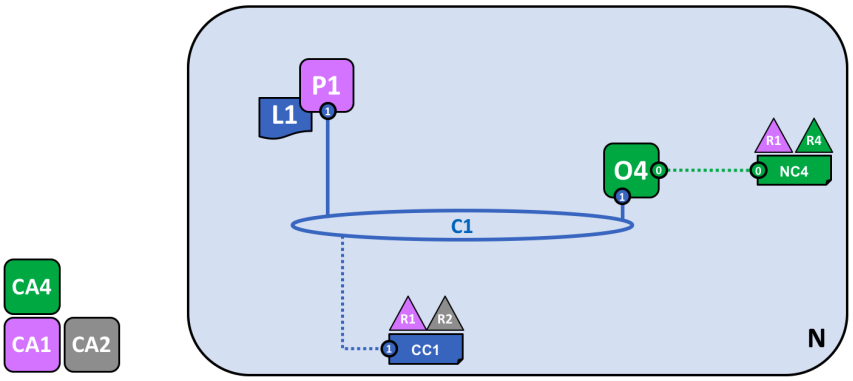
Kênh C1 được tạo sử dụng Consortium X1. Cấu hình kênh CC1 của C1 được



Hình 2.5: Bước 4: Tạo kênh tương ứng với Consortium. Nguồn [3]

tách biệt hoàn toàn với cấu hình mạng NC4. CC1 được quản lý bởi R1 và R2, 2 tổ chức này có quyền ngang nhau đối với C1. R4 dù là một quản trị mạng nhưng sẽ không có quyền gì đối với CC1.

2.3.5 Bước 5: Thêm Peer node



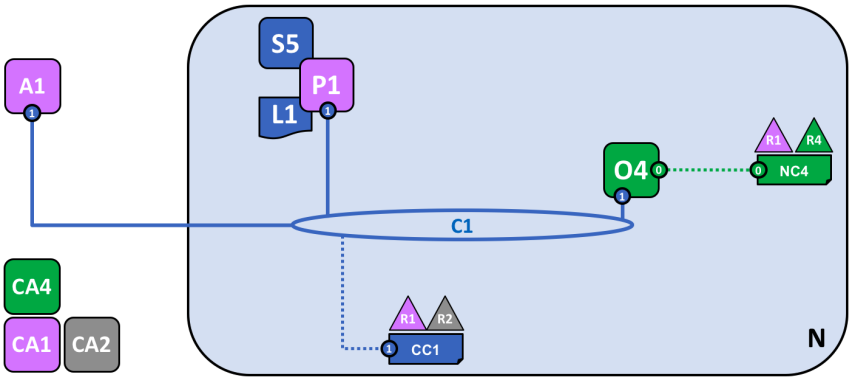
Hình 2.6: Bước 5: Thêm Peer node. Nguồn [3]

Peer P1 được thêm vào kênh C1. Trên kênh C1 sẽ tồn tại một số cái duy nhất và P1 lưu trữ một bản sao L1 của cuốn số cái này. Có thể nói, L1 được lưu trữ vật lý trên P1 và được lưu trữ một cách lô-gíc trên kênh C1. P1 thuộc tổ chức R1, do đó sẽ được cung cấp một định danh từ CA1. Khi được khởi chạy, P1 sẽ gửi yêu cầu tham gia kênh C1 tới O4. Dựa trên định danh của P1 và cấu hình kênh CC1, quyền của P1 trên kênh sẽ được xác định.

2.3.6 Bước 6: Cài đặt Hợp đồng thông minh

Sau khi đã tồn tại sổ cái, hợp đồng thông minh có thể được triển khai để ứng dụng vào nghiệp vụ.

Hợp đồng thông minh S5 được cài đặt trên Peer P1. Ứng dụng ngoài A1 có thể sử dụng S5 để truy cập vào sổ cái L1 thông qua peer P1. A1 để có thể được sử dụng cũng sẽ cần định danh, ví dụ A1 có thể sở hữu một định danh được CA1 cung cấp.



Hình 2.7: Bước 6: Cài đặt Hợp đồng thông minh. Nguồn [3]

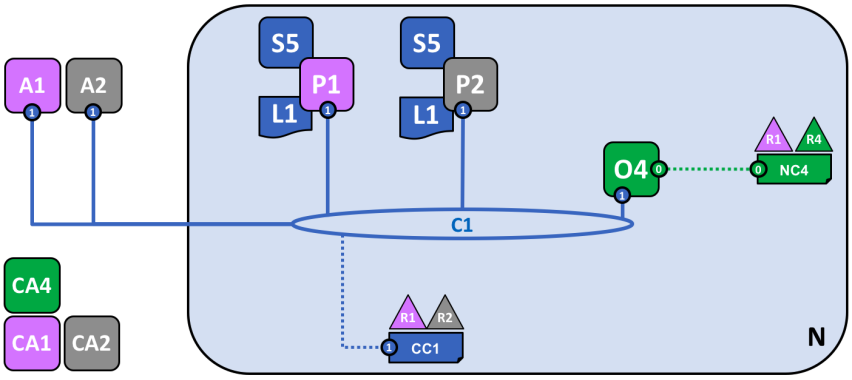
Cần lưu ý rằng A1 không thể tương tác trực tiếp sổ cái L1 trực tiếp thông qua P1, mà tất cả quyền truy cập tới sổ cái sẽ được quản lý thông qua S5. Có thể hiểu là S5 định nghĩa và cung cấp một tập hợp các phương thức mà theo đó sổ cái L1 có thể được truy vấn hay cập nhật.

S5 sẽ không chỉ được cài đặt trên P1, mà còn trên cả kênh C1. Điều này đảm bảo các thành phần kết nối với C1 sẽ biết đến S5. Tuy nhiên những thành phần khác ngoài P1 sẽ không thể nhìn thấy cấu trúc chi tiết của S5.

Một kênh có thể có nhiều Hợp đồng thông minh.

2.3.7 Bước 7: Khởi tạo mạng hoàn tất

Để hoàn tất khởi tạo mạng, Tổ chức R2 sẽ được thêm vào.



Hình 2.8: Bước 7: Khởi tạo mạng hoàn tất. Nguồn [3]

R2 sẽ thêm Peer node P2, cài đặt S5 trên đó. P2 cũng sẽ có một bản sao của sổ cái L1. A2 sử dụng định danh được cung cấp bởi CA2 có thể thương tác với L1 thông qua S5. Lúc này, hai tổ chức R1 và R2 đã có thể giao dịch với nhau thông qua kênh C1.

Chương 2 đã trình bày về tổng quan kiến trúc của một mạng Hyperledger Fabric. Đây sẽ là tiền đề để thiết kế một hệ thống có thể tự động triển khai mạng và ứng

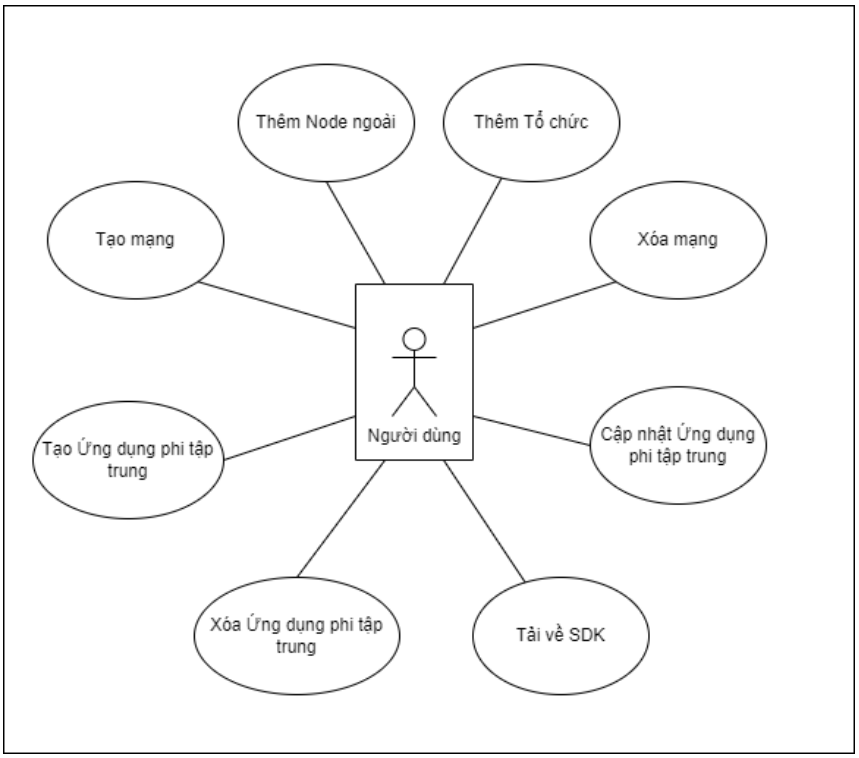
dụng phi tập trung. Chương tiếp theo sẽ phân tích các yêu cầu dựa trên quá trình khởi tạo một mạng như đã nói ở chương này.

CHƯƠNG 3. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Chương 2 đã trình bày về kiến trúc mạng Hyperledger Fabirc. Chương 3 này sẽ tiến hành phần tích yêu cầu hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric.

3.1 Tổng quan chức năng

3.1.1 Biểu đồ use case tổng quan



Hình 3.1: Biểu đồ usecase Tổng quan

Hình 3.1 là biểu đồ usecase tổng quát của hệ thống. Hệ thống chỉ bao gồm duy nhất một tác nhân, đó là người dùng. Người dùng có thể tạo, xóa mạng Hyperledger Fabric. Đối với các mạng đã được tạo, người dùng có thể thêm một node ngoài vào trong hệ thống, cùng với đó là thêm một tổ chức mới. Trên các mạng đó, người dùng còn có thể tạo ứng dụng phi tập trung mới, cập nhật và xóa các ứng dụng phi tập trung đã tồn tại. Cuối cùng, người dùng có thể tải về các SDK để tương tác với các ứng dụng phi tập trung tương ứng.

3.2 Đặc tả chức năng

3.2.1 Đặc tả use case Tạo mạng

Bảng 3.1: Đặc tả ca sử dụng "Tạo mạng"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC01</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Tạo mạng</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng tạo một mạng Hyperledger Fabric mới</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã đăng nhập vào hệ thống</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả tạo mạng</td></tr>
<tr><td>Luồng sự kiện chính:</td><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td></td><td>1</td><td>Người dùng</td><td>Nhấn nút tạo mạng trên giao diện</td></tr>
<tr><td></td><td>2</td><td>Người dùng</td><td>Nhập thông tin cấu hình mạng: (i) Tên mạng, (ii) Số máy ảo, (iii) Cấu hình mỗi máy ảo, (iv) Số Tổ chức, (v) Số peer node cho mỗi tổ chức</td></tr>
<tr><td></td><td>3</td><td>Hệ thống</td><td>Tự động triển khai mạng theo thông tin người dùng đã nhập</td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Thông tin nhập vào không hợp lệ. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.2 Đặc tả use case Thêm node ngoài

Bảng 3.2: Đặc tả ca sử dụng "Thêm Node ngoài"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC02</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Thêm Node ngoài</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng thêm một node ngoài vào một mạng đang hoạt động của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một mạng</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả thêm node ngoài</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút thêm node ngoài trên giao diện</td></tr>
<tr><td>2</td><td>Người dùng</td><td>Nhập thông tin: (i) Tên node, (ii) Tổ chức Node sẽ thuộc về, (iii) Địa chỉ IP và cổng mạng node sẽ hoạt động</td></tr>
<tr><td>3</td><td>Hệ thống</td><td>Thực hiện sinh các tệp tin để có thể cho phép một Node ngoài với thông tin người dùng đã cung cấp có thể tham gia vào mạng</td></tr>
<tr><td>4</td><td>Người dùng</td><td>Tải tệp tin hệ thống sinh ra về</td></tr>
<tr><td>5</td><td>Người dùng</td><td>Khởi chạy các tệp tin được hệ thống sinh trên máy có đại chỉ Ip và cổng mạng lúc nhập để máy đó tham gia vào mạng</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Thông tin nhập vào không hợp lệ, mạng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.3 Đặc tả use case Thêm tổ chức

Bảng 3.3: Đặc tả ca sử dụng "Thêm tổ chức"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC03</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Thêm tổ chức</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng thêm một tổ chức mới vào một mạng đang hoạt động của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một mạng</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả thêm tổ chức</td></tr>
<tr><td>Luồng sự kiện chính:</td><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td></td><td>1</td><td>Người dùng</td><td>Nhấn nút thêm tổ chức trên giao diện</td></tr>
<tr><td></td><td>2</td><td>Người dùng</td><td>Nhập thông tin tổ chức mới: (i) Tên tổ chức, (ii) Số peer thuộc tổ chức mới</td></tr>
<tr><td></td><td>3</td><td>Hệ thống</td><td>Tự động thêm một tổ chức mới vào mạng</td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Thông tin nhập vào không hợp lệ, mạng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.4 Đặc tả use case Xóa mạng

Bảng 3.4: Đặc tả ca sử dụng "Xóa mạng"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC04</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Xóa mạng</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng xóa một mạng của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một mạng</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả xóa mạng</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút xóa mạng trên giao diện</td></tr>
<tr><td>2</td><td>Người dùng</td><td>Xác nhận sẽ xóa mạng</td></tr>
<tr><td>3</td><td>Hệ thống</td><td>Gỡ bỏ toàn bỏ các thành phần trong mạng</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Mạng chưa hoạt động. Hệ thống sẽ báo lỗi</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.5 Đặc tả use case Tạo Ứng dụng phi tập trung

Bảng 3.5: Đặc tả ca sử dụng "Tạo Ứng dụng phi tập trung"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC05</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Tạo Ứng dụng phi tập trung</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng tạo Ứng dụng phi tập trung trên một mạng đang hoạt động của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một mạng</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả tạo ứng dụng phi tập trung</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút tạo ứng dụng trên giao diện</td></tr>
<tr><td>2</td><td>Người dùng</td><td>Nhập thông tin: (i) Tên ứng dụng, (ii) Mạng mà ứng dụng sẽ được triển khai lên, (iii) Kiểu mã hóa</td></tr>
<tr><td>3</td><td>Người dùng</td><td>Thao tác trên giao diện để khai báo kiến trúc của ứng dụng: (i) Tạo thực thể, (ii) Thêm thuộc tính cho các thực thể, (iii) Thêm quan hệ cho các thực thể (1-1, 1-n, n-n)</td></tr>
<tr><td>4</td><td>Người dùng</td><td>Xác nhận tạo ứng dụng</td></tr>
<tr><td>5</td><td>Hệ thống</td><td>Tự động triển khai ứng dụng tương ứng lên mạng đã chọn</td></tr>
<tr><td>6</td><td>Hệ thống</td><td>Sinh ra SDK để có thể sử dụng ứng dụng.</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Thông tin nhập vào không hợp lệ, mạng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.6 Đặc tả use case Tải SDK

Bảng 3.6: Đặc tả ca sử dụng "Tải SDK"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC06</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Tải SDK</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng tải SDK tương ứng với một ứng dụng của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một ứng dụng phi tập trung</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả tải về SDK</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút tải SDK trên giao diện</td></tr>
<tr><td>2</td><td>Hệ thống</td><td>Tải về tệp tin của SDK</td></tr>
<tr><td>3</td><td>Người dùng</td><td>Sử dụng SDK</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Ứng dụng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.7 Đặc tả use case Cập nhật Ứng dụng phi tập trung

Bảng 3.7: Đặc tả ca sử dụng "Cập nhật Ứng dụng phi tập trung"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC07</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Cập nhật Ứng dụng phi tập trung</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng cập nhật một ứng dụng phi tập trung của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một ứng dụng phi tập trung</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả cập nhật ứng dụng</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút cập nhật ứng dụng trên giao diện</td></tr>
<tr><td>2</td><td>Người dùng</td><td>Nhập thông tin mới: Tên mới cho ứng dụng</td></tr>
<tr><td>3</td><td>Người dùng</td><td>Thao tác trên giao diện để chỉnh sửa kiến trúc của ứng dụng: (i) Thêm thực thể, (ii) Thêm thuộc tính cho thực thể mới và cũ, (iii) Thêm quan hệ cho các thực thể mới và cũ (1-1, 1-n, n-n)</td></tr>
<tr><td>4</td><td>Người dùng</td><td>Xác nhận cập nhật ứng dụng</td></tr>
<tr><td>5</td><td>Hệ thống</td><td>Tự động triển khai cập nhật ứng dụng</td></tr>
<tr><td>6</td><td>Hệ thống</td><td>Cập nhật SDK mới</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Thông tin nhập vào không hợp lệ, mạng chưa hoạt động, ứng dụng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.2.8 Đặc tả use case Xóa Ứng dụng phi tập trung

Bảng 3.8: Đặc tả ca sử dụng "Xóa Ứng dụng phi tập trung"

<table>
<tr><td>Mã use case:</td><td colspan="3">UC08</td></tr>
<tr><td>Tên use case:</td><td colspan="3">Xóa Ứng dụng phi tập trung</td></tr>
<tr><td>Tác nhân:</td><td colspan="3">Người dùng</td></tr>
<tr><td>Mô tả:</td><td colspan="3">Người dùng xóa một ứng dụng phi tập trung của mình</td></tr>
<tr><td>Điều kiện tiên quyết:</td><td colspan="3">Người dùng đã tạo một ứng dụng phi tập trung</td></tr>
<tr><td>Hậu điều kiện:</td><td colspan="3">Hiện thị kết quả xóa ứng dụng</td></tr>
<tr><td>Luồng sự kiện chính:</td><td colspan="3"><table>
<tr><td>STT</td><td>Thực hiện bởi</td><td>Hành động</td></tr>
<tr><td>1</td><td>Người dùng</td><td>Nhấn nút xóa ứng dụng trên giao diện</td></tr>
<tr><td>2</td><td>Người dùng</td><td>Xác nhận xóa ứng dụng</td></tr>
<tr><td>3</td><td>Hệ thống</td><td>Xóa ứng dụng khỏi mạng</td></tr>
</table></td></tr>
<tr><td>Luồng sự kiện con:</td><td colspan="3">Không có.</td></tr>
<tr><td>Ngoại lệ:</td><td colspan="3">Mạng chưa hoạt động, ứng dụng chưa hoạt động. Hệ thống sẽ thông báo lỗi.</td></tr>
<tr><td>Bao gồm:</td><td colspan="3">Không có.</td></tr>
</table>

3.3 Yêu cầu phi chức năng

Để các mạng, ứng dụng phi tập trung có thể được ứng dụng trong nghiệp vụ doanh nghiệp, tính ổn định của hệ thống mạng sẽ cần được đảm bảo. Cụ thể đó là khả năng chịu lỗi, tính khả dụng cao và khả năng phục hồi.

Chương 3 đã trình bày phần tích yêu cầu hệ thống. Từ những phân tích này, chương 4 sẽ trình bày các công nghệ được sử dụng để hoàn thành những yêu cầu cần đạt được.

CHƯƠNG 4. CÔNG NGHỆ SỬ DỤNG

Chương 3 đã phần tích yêu cầu hệ thống. Chương 4 này sẽ trình bày về các công nghệ được sử dụng để phát triển hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric.

4.1 Công nghệ triển khai mạng và ứng dụng

Để có thể được ứng dụng vào trong các hoạt động doanh nghiệp, mạng chuỗi khối riêng tư sẽ phải hoạt động liên tục và ổn định nhất có thể. Các công nghệ sau được sử dụng để đảm bảo quá trình triển khai, quản lý mạng cũng như ứng dụng phi tập trung hoạt động trơn tru nhất có thể.

4.1.1 Container

Container là một môi trường ảo trong đó có một ứng dụng, tiến trình hoạt động. Container gần giống như máy áo khi cho phép đóng gói tiến trình, ứng dụng đó. Việc đóng gói cho phép container có thể được vận chuyển từ máy tính này sang máy tính khác mà vẫn đảm bảo tiến trình, ứng dụng bên trong container sẽ hoạt động bình thường bất kể môi trường máy chủ khác nhau. Điều này có nghĩa là miễn là các máy có thể chạy container, ứng dụng được đóng gói trong container sẽ chạy ổn định trên các máy đó. Điểm khác biệt giữa container và máy ảo là container không ảo hóa toàn bộ tài nguyên máy tính. Do đó, chi phí để chạy một container là thấp hơn rất nhiều so với máy ảo, cho phép một máy chủ dễ dàng chạy nhiều container một lúc.

Các thành phần như Orderer node, Peer node, Nhà cung cấp chứng chỉ số hay Hợp đồng thông minh của mạng Hyperledger Fabric khi được hệ thống triển khai đều sẽ nằm trong container, giúp việc khởi chạy các thành phần này dễ dàng hơn.

4.1.2 Kubernetes

Số lượng thành phần trong kiến trúc mạng Hyperledger Fabric là không nhỏ. Việc quản lý và kết nối các thành phần này cũng là một thử thách. Đó là lý do Kubernetes[4] được lựa chọn. Kubernetes là một nền tảng mã nguồn mở hỗ trợ triển khai và quản lý nhiều container một cách tối ưu. Cơ sở hạ tầng triển khai sử dụng Kubernetes sẽ có khả năng:

- **Tự động điều phối:** Container sẽ được tự động triển khai một cách hợp lý đến một trong các máy chủ trong hệ thống.
- **Tự khởi động lại:** Các container khi gặp lỗi dừng hoạt động sẽ được tự động phát hiện và tái khởi động lại.

- **Dễ dàng mở rộng:** Có thể dễ dàng triển khai nhiều container giống nhau và cân bằng tải tới các container cùng loại đó.
- **Cập nhật không thời gian chết:** Việc thực hiện cập nhật các container đang chạy sẽ không gây gián đoạn đến hệ thống.

Đồ án này sẽ sử dụng Kubernetes làm cơ sở để triển khai mạng Hyperledger Fabric nhằm nâng cao khả năng chịu lỗi và tối ưu hiệu suất của hệ thống.

4.1.3 Dịch vụ điến toán đám mây

Hệ thống trong đồ án lựa chọn máy chủ ảo cung cấp bởi các dịch vụ điện toán đám mây để trở thành hệ thống máy tính chạy mạng chuỗi khối Hyperledger Fabric. Nhà cung cấp dịch vụ điện toán đám mây sẽ là người quản lý các máy chủ vật lý, loại bỏ gánh nặng này khỏi vai người dùng. Thêm vào đó, với quy mô hoạt động linh hoạt, chi phí để duy trì hoạt động của máy ảo sẽ tương ứng với tài nguyên được thực sự sử dụng, không lo việc lãng phí tài nguyên thừa.

4.2 Công nghệ xây dựng hệ thống

Ngôn ngữ lập trình Python được sử dụng để xây dựng hệ thống, cùng với đó các công nghệ dưới đây.

4.2.1 MongoDB

MongoDB[5] là một cơ sở dữ liệu NoSQL phổ biến nhất. Cơ sở dữ liệu NoSQL trái ngược với các mô hình dữ liệu quan hệ SQL ở chỗ dữ liệu không được lưu thành bảng, cột mà sẽ được lưu tập hợp (collection) của các bản ghi (record). Điều này cho phép dữ liệu có thể được lưu trữ với cấu trúc linh hoạt. Tốc độ truy cập cao cùng hiệu năng vượt trội là lý do MongoDB được sử dụng trong hệ thống.

4.2.2 RabbitMQ

RabbitMQ[6] là một message broker sử dụng giao thức AMQP (Advanced Message Queuing Protocol) hay còn gọi là phần mềm quản lý hàng đợi yêu cầu (message). Trong kiến trúc Mỉcroservices, một thử thách lớn nhất là việc quản lý giao tiếp giữa các dịch vụ (service). RabbitMQ là công cụ được sử dụng để giải quyết vấn đề này qua các cơ chế diều phối, xử lý việc trao đổi yêu cầu giữa nhiều bên cùng với đó đảm bảo yêu cầu được gửi sẽ đến đích một cách toàn vẹn.

4.2.3 Celery

Celery[7] là một hệ thống quản lý hàng đợi công việc (task) thời gian thực. Các tác vụ như tạo mạng trong hệ thống yêu cầu thời gian xử lý dài. Để đảm bảo hiệu năng, các tác vụ nặng này sẽ được phân chia cho nhiều máy thực hiện. Celery được sử dụng để hỗ trợ quá trình phân chia công việc này giúp bảo hệ thống có thể mở

rộng dễ dàng hơn. Celery sử dụng một message broker (RabbitMQ) để hoạt động.

Chương 4 đã trình bày về các công nghệ được sử dụng trong hệ thống. Chương tiếp theo sẽ trình bày về chi tiết cấu trúc và các luồng hoạt động của hệ thống.

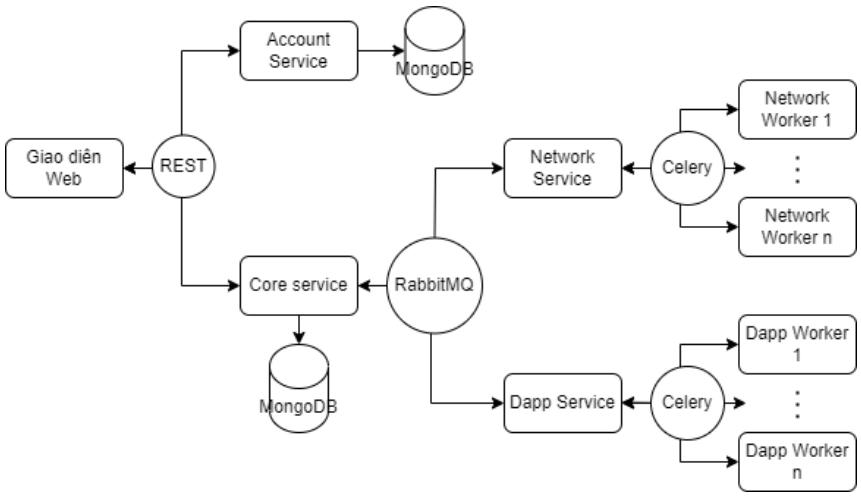
CHƯƠNG 5. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Các chương trước đã trình bày về yêu cầu và các công nghệ được sử dụng để hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric hoạt động. Chương 5 này sẽ trình bày từ tổng quan đến chi tiết thiết kế hệ thống, cùng với đó là kết quả thu được và kiểm thử.

5.1 Thiết kế tổng quan

Kiến trúc Microservice được sử dụng để triển khai hệ thống. Hệ thống sẽ được chia thành các dịch vụ hoạt động độc lập. Thông qua kiến trúc này, mỗi dịch vụ có thể được cập nhật, mở rộng mà không ảnh hưởng đến những dịch vụ khác.

5.1.1 Kiến trúc tổng quan



Hình 5.1: Tổng quan các dịch vụ của hệ thống

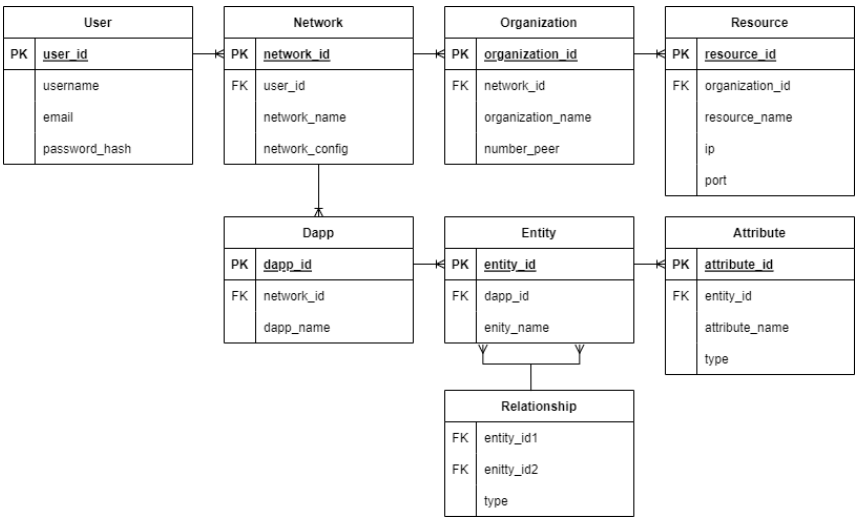
Hình 5.1 mô tả tổng quan các dịch vụ trong hệ thống. Các dịch vụ này bao gồm:

- **Giao diện web:** Giao diện web người dùng sử dụng để tương tác với hệ thống.
- **Account service:** Dịch vụ quản lý tài khoản người dùng. Dịch vụ này sở hữu một cơ sở dữ liệu riêng biệt.
- **Core Service:** Dịch vụ cung cấp các chức năng chính. Dịch vụ này sẽ tương tác với Network và Dapp Service để hoàn thành yêu cầu người dùng. Dịch vụ này sở hữu một cơ sở dữ liệu riêng biệt.
- **Network service:** Dịch vụ xử lý các yêu cầu liên quan đến triển khai và quản lý mạng Hyperledger Fabric thông qua các Network Worker.
- **Network Worker:** Đơn vị trực tiếp thực hiện các tác vụ liên quan đến triển khai và quản lý mạng.

- **Dapp service:** Dịch vụ xử lý các yêu cầu liên quan đến triển khai và quản lý ứng dụng phi tập trung thông qua các Dapp Worker.
- **Dapp Worker:** Đơn vị trực tiếp thực hiện các tác vụ liên quan đến triển khai và quản lý ứng dụng phi tập trung.

Người dùng tương tác với Account và Core Service thông qua REST API. Account service xử lý các yêu cầu liên quan đến tài khoản người dùng (đăng ký, đăng nhập, xác thực). Core Service xử lý các yêu cầu liên quan đến mạng chuỗi khối và ứng dụng phi tập trung. Core Service sẽ giao tiếp với Network và Dapp Service thông qua RabbitMQ. Để xử lý các tác vụ mạng và ứng dụng nhận từ Core service, hai service này sẽ thực hiện quản lý phân chia công việc cho các Worker thông qua Celery.

5.1.2 Kiến trúc tổng quan cơ sở dữ liệu



Hình 5.2: Cơ sở dữ liệu tổng quan

Hình 5.2 mô tả về thiết kế cơ sở dữ liệu cho các dịch vụ. Với kiến trúc Microservices, các bảng sẽ được quản lý bởi nhiều dịch vụ.

Account service quản lý người dùng (User).

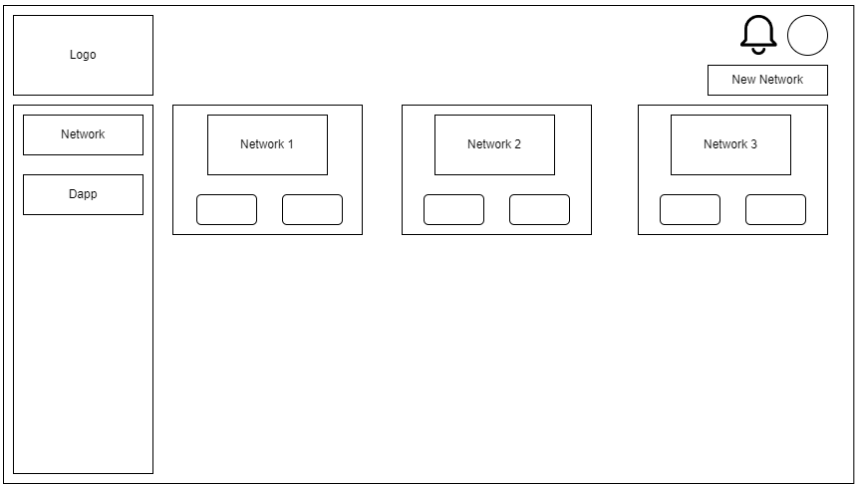
Core Service quản lý mạng (Network), tổ chức (Organization) và node ngoài (Resource). Một người dùng có nhiều mạng và một mạng có nhiều tổ chức. Mỗi một node ngoài sẽ thuộc về 1 tổ chức. Ngoài ra một mạng có nhiều ứng dụng phi tập trung (Dapp). Cấu trúc của ứng dụng phi tập trung được định nghĩa thông qua các thực thể (Entity), những thuộc tính (Attribute) của thực thể và quan hệ (Relationship) giữa các thực thể.

Chi tiết thiết kế cơ sở dữ liệu cho từng dịch vụ sẽ được trình bày trong các mục thiết kế chi tiết dịch vụ tương ứng tiếp theo.

5.2 Thiết kế chi tiết

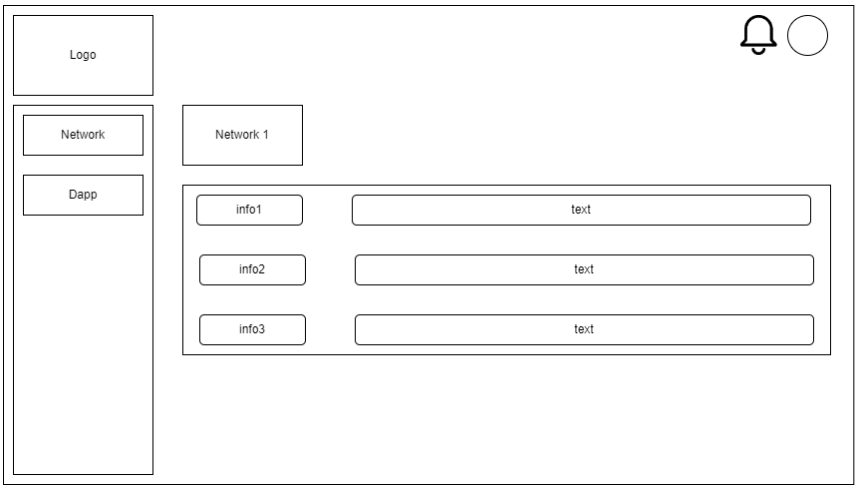
5.2.1 Thiết kế giao diện

Thiết kế các màn hình chính cho giao diện của hệ thống sẽ được trình bày trong phần này. Kích thước màn hình mà giao diện của hệ thống hướng đến là 16:9 cùng độ phân giải được sử dụng là 1920 x 1080. Kích thước và độ phân giải này hiện rất phổ biến hiện nay và thường được dùng cho màn hình máy tính, laptop.



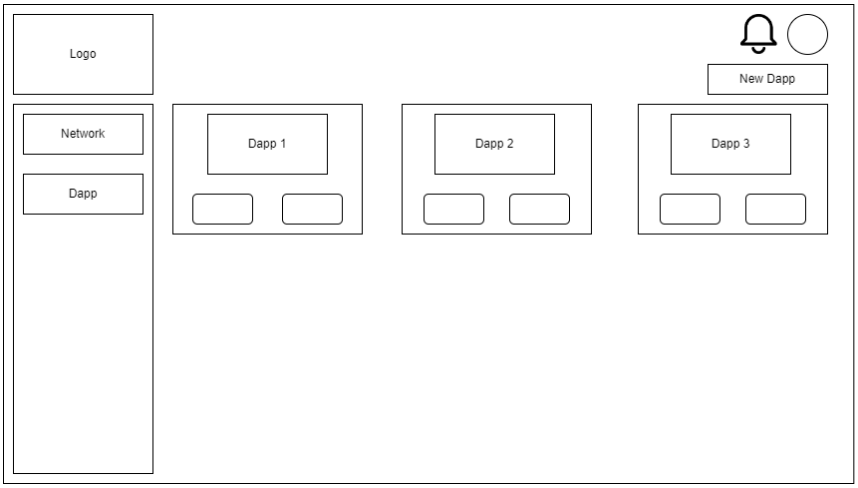
Hình 5.3: Màn hình danh sách mạng

Hình 5.3 mô tả màn hình danh sách các mạng của người dùng. Màn hình này sẽ hiện thị thông tin tổng quan về các mạng của người dùng, cùng với đó là nút "New network" để tạo một mạng mới



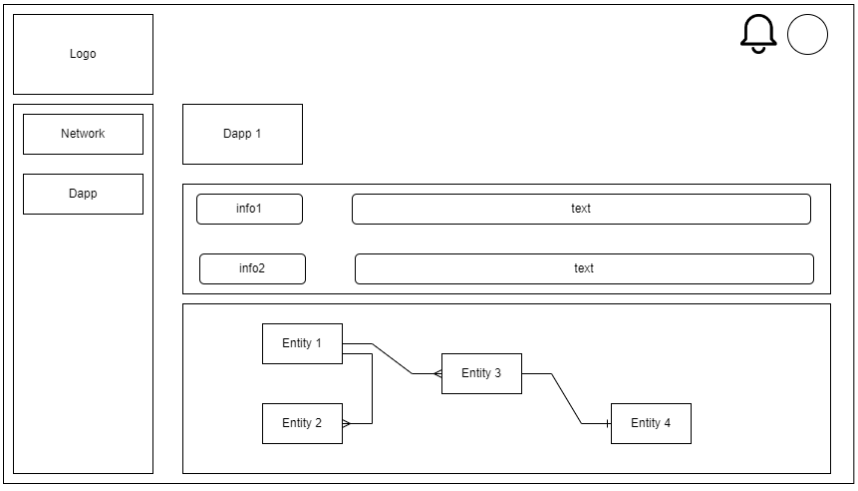
Hình 5.4: Màn hình thông tin mạng

Hình 5.4 mô tả màn hình chi tiết thông tin một mạng của người dùng.



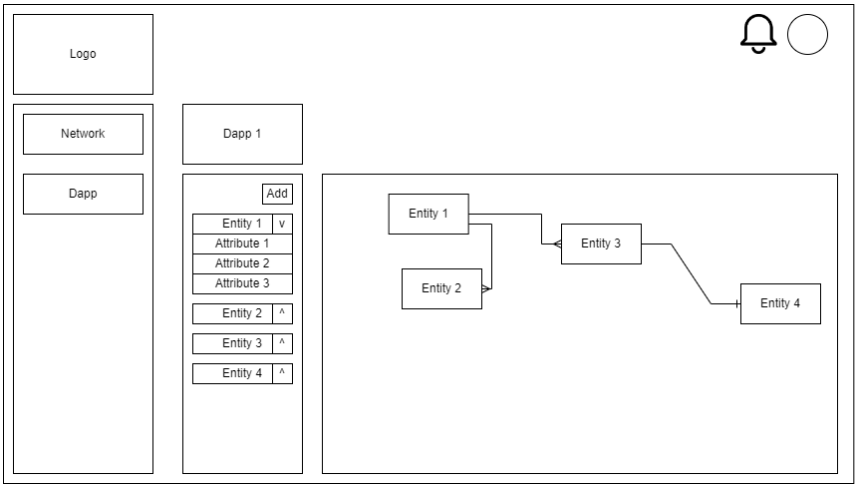
Hình 5.5: Màn hình danh sách ứng dụng

Hình 5.5 mô tả màn hình danh sách các ứng dụng của người dùng. Về cơ bản, màn hình này có cấu trúc tương tự với màn hình danh sách mạng của người dùng.



Hình 5.6: Màn hình thông tin ứng dụng

Hình 5.6 mô tả màn hình chi tiết thông tin một ứng dụng phi tập trung của người dùng. Ngoài thông tin, chi tiết về cấu trúc của ứng dụng được hiện thị trực quan thông qua một biểu đồ cơ sở dữ liệu quan hệ.

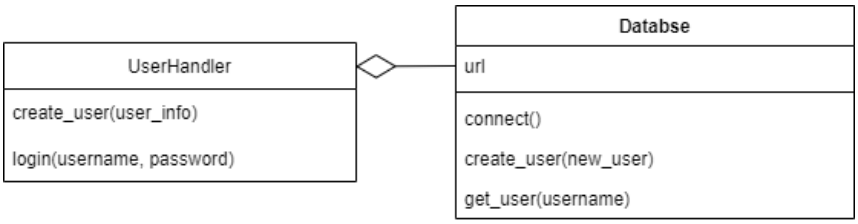


Hình 5.7: Màn hình tạo ứng dụng

Hình 5.6 mô tả màn hình tạo ứng dụng phi tập trung. Cấu trúc của ứng dụng được định nghĩa theo sơ đồ cơ sở dữ liệu quan hệ. Hệ thống sẽ cung cấp một giao diện để người dùng định nghĩa cấu trúc này một cách trực quan nhất. Người dùng có thể tạo và định nghĩa các thuộc tính cho các thực thể với thanh "Control". Tiếp đến sẽ thiết kế các quan hệ cho các thực thể bằng cách tương tác với bảng "Overview".

5.2.2 Thiết kế Account Service

a, Biểu đồ lớp



Hình 5.8: Biểu đồ lớp Account Service

Hình 5.8 mô tả biểu đồ lớp của Account service. Chi tiết các lớp được mô tả ở Bảng 5.1

Bảng 5.1: Mô tả chi tiết các lớp của Account service

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
UserHandler	Xử lý các yêu cầu thông qua API	create_user(user_info)	Tạo một user mới với thông tin tương ứng với user_info

		login(username, password)	Đăng nhập người dùng vào hệ thống. Trả về JWT Token nếu đăng nhập thành công
Database	Tương tác với cơ sở dữ liệu	url	Đường dẫn để kết nối tới cơ sở dữ liệu
		connect()	Kết nối tới cơ sở dữ liệu
		create_user(new_user)	Tạo một người dùng mới trong cơ sở dữ liệu
		get_user(user_info)	Đọc thông tin về về người dùng từ cơ sở dữ liệu

b, Thiết kế cơ sở dữ liệu

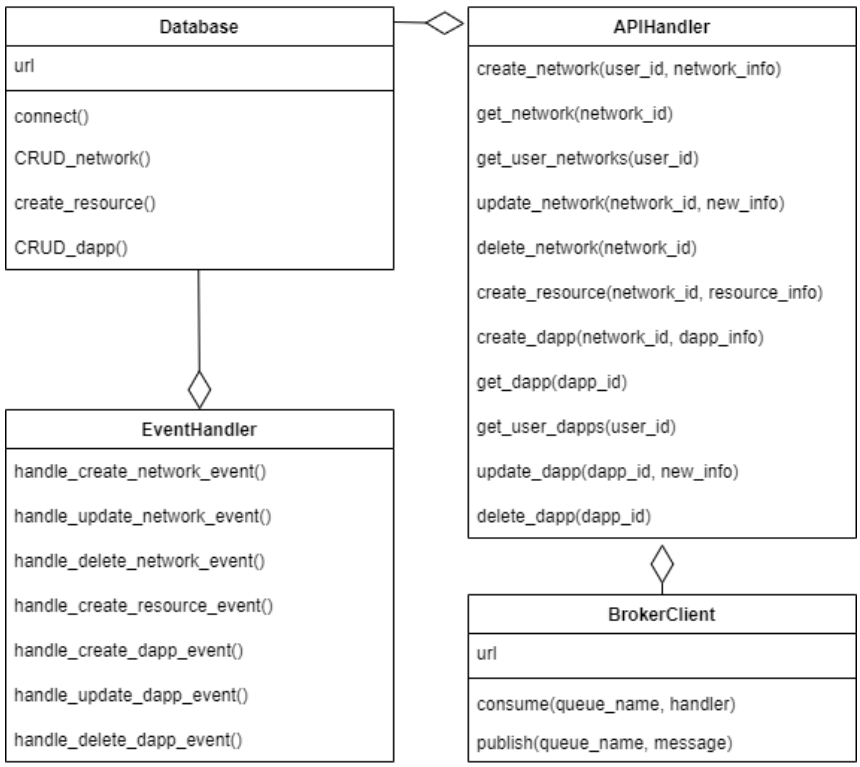
Thiết kế chi tiết cơ sở dữ liệu cho Account Service được trình bày ở Bảng 5.2

Bảng 5.2: Chi tiết cơ sở dữ liệu của Account service

Collection	Tên trường	Kiểu dữ liệu	Mô tả
User	user_id	ObjectId	ID của người dùng
	username	String	Tên của người dùng
	email	String	Email của người dùng
	password_hash	String	Giá trị băm của mật khẩu người dùng

5.2.3 Thiết kế Core Service

a, Biểu đồ lớp



Hình 5.9: Biểu đồ lớp Core Service

Hình 5.10 mô tả biểu đồ lớp của Core Service. Thông tin các lớp được mô tả chi tiết ở Bảng 5.3

Bảng 5.3: Mô tả chi tiết các lớp của Core Service

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
ApiHandler	Xử lý các yêu cầu thông qua API	create_network(user_id, network_info)	Tạo mới cho người dùng có ID user_id một mạng có cấu hình network_info
		get_network(network_id)	Lấy thông tin về mạng có ID network_id
		get_user_networks (user_id)	Lấy thông tin về tất cả mạng của người dùng có ID user_id

		update_network(network_id, new_info)	Cập nhật mạng có ID network_id cấu hình mới new_info
		delete_network(network_id)	Xóa mạng có ID network_id
		create_resource(network_id, resource_info)	Tạo một node ngoài có cấu hình resource_info cho mạng có ID network_id
		create_dapp(network_id, dapp_info)	Tạo mới một ứng dụng phi tập trung có cấu hình dapp_info cho mạng có ID network_Id
		get_dapp(dapp_id)	Lấy thông tin về ứng dụng phi tập trung có ID dapp_id
		get_user_dapps(user_id)	Lấy thông tin về tất cả ứng dụng phi tập trung của người dùng có ID user_id
		update_dapp(dapp_id, new_info)	Cập nhật mạng có ID dapp_id cấu hình mới new_info
		delete_dapp(dapp_id)	Xóa ứng dụng có ID dapp_id
EventHandler	Xử lý sự kiện được thông báo từ các dịch vụ khác	handle_create_network_event()	Xử lý sự kiện mạng mới được tạo
		handle_update_network_event()	Xử lý sự kiện mạng được cập nhật
		handle_delete_network_event()	Xử lý sự kiện mạng được xóa
		handle_create_resource_event()	Xử lý sự kiện node ngoài được tạo
		handle_create_dapp_event()	Xử lý sự kiện ứng dụng phi tập trung được tạo

		handle_update_dapp_ event()	Xử lý sự kiện ứng dụng phi tập trung được cập nhật
		handle_delete_dapp_ event()	Xử lý sự kiện ứng dụng phi tập trung bị xóa
Database	Tương tác với cơ sở dữ liệu	url	Đường dẫn kết nối tới cơ sở dữ liệu
		connect()	Kết nối tới cơ sở dữ liệu
		CRUD_network()	Đọc ghi xóa sửa thông tin mạng trong cơ sở dữ liệu
		create_resource()	Ghi thông tin node ngoài mới vào cơ sở dữ liệu
		CRUD_dapp()	Đọc ghi xóa sủa thông tin ứng dụng phi tập trung trong cơ sở dữ liệu
BrokerClient	Tương tác với message broker	url	Đường dẫn để kết nối tới message broker
		consume(queue_name, handler)	Lắng nghe sự kiện từ hàng đợi queue_name của message broker và xử lý xử kiện đó với handler
		publish(queue_name, message)	Gửi một message tới hàng đợi queue_name của message broker

b, Thiết kế cơ sở dữ liệu

Thiết kế chi tiết cơ sở dữ liệu cho Core Service được trình bày ở Bảng 5.4

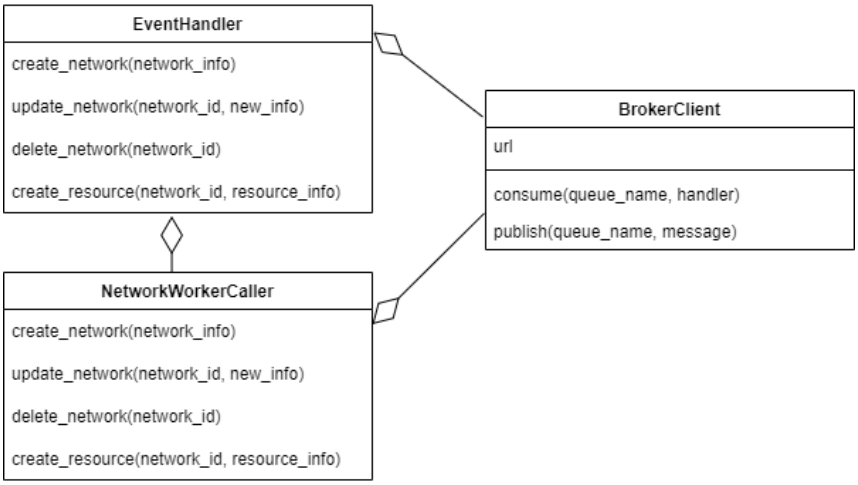
Bảng 5.4: Chi tiết cơ sở dữ liệu của Core Service

Collection	Tên trường	Kiểu dữ liệu	Mô tả
Network	network_id	ObjectId	ID của mạng
	user_id	ObjectId	ID của người dùng sở hữu mạng
	network_name	String	Tên của mạng
	network_config	Object	Cấu hình mạng
Organization	organization_id	ObjectId	ID của tổ chức

Bảng 5.4: Chi tiết cơ sở dữ liệu của Core Service

Collection	Tên trường	Kiểu dữ liệu	Mô tả
	network_id	ObjectId	ID của mạng mà tổ chức thuộc về
	organization_name	String	Tên tổ chức
	number_peer	Number	Số peer node trong tổ chức
Resource	resource_id	ObjectId	ID của node ngoài
	organization_id	ObjectId	ID của tổ chức mà node ngoài thuộc về
	resource_name	String	Tên node ngoài
	ip	String	Địa chỉ IP của máy mà node ngoài sẽ chạy
	port		Cổng mạng của máy mà node ngoài sẽ chạy
Dapp	dapp_id	ObjectId	ID của ứng dụng
	network_id	ObjectId	ID của mạng mà ứng dụng thuộc về
	dapp_name	String	Tên ứng dụng
Entity	entity_id	ObjectId	ID của thực thể
	dapp_id	ObjectId	ID của ứng dụng mà thực thể thuộc về
	entity_name	String	Tên ứng dụng
Attribute	attribute_id	ObjectId	ID của thuộc tính
	entity_id	ObjectId	ID của thực thể mà thuộc tính thuộc về
	attribute_name	String	Tên thuộc tính
	type	String	Kiểu dữ liệu của thuộc tính
Relationship	entity_id1	ObjectId	ID của thực thể đầu tiên
	entity_id2	ObjectId	ID của thực thể còn lại
	type	String	Kiểu quan hệ

5.2.4 Thiết kế Network Service



Hình 5.10: Biểu đồ lớp Network Service

Hình 5.10 mô tả biểu đồ lớp của Network Service. Chi tiết các lớp được mô tả ở Bảng 5.5

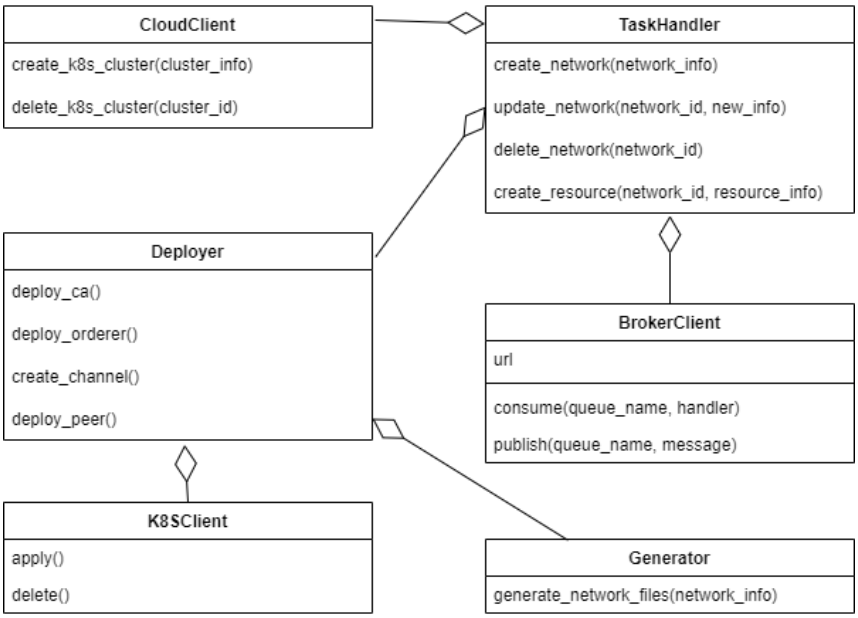
Bảng 5.5: Mô tả chi tiết các lớp của Network service

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
EventHandler	Xử lý yêu cầu từ Core Service	create_network(network_info)	Xử lý yêu cầu tạo mạng với cấu hình network_info
		update_network(network_id, new_info)	Xử lý yêu cầu cập nhật mạng có ID network_id với thông tin mới new_info
		delete_network(network_id)	Xử lý yêu cầu xóa mạng có ID network_id
		create_resource(resource_info)	Xử lý yêu cầu tạo node ngoài có cấu hình resource_info
WorkerCaller	Yêu cầu Network Worker thực hiện tác vụ	create_network(network_info)	Yêu cầu tạo mạng với cấu hình network_info
		update_network(network_id, new_info)	Yêu cầu cập nhật mạng có ID network_id với thông tin mới new_info

Bảng 5.5: Mô tả chi tiết các lớp của Network service

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
		delete_network(network_id)	Yêu cầu xóa mạng có ID network_id
		create_resource(resource_info)	Yêu cầu tạo node ngoài có cấu hình resource_info
BrokerClient	Tương tác với message broker	Tương tự Core Service	

5.2.5 Thiết kế Network Worker



Hình 5.11: Biểu đồ lớp Network Worker

Hình 5.11 mô tả biểu đồ lớp của Network Worker. Chi tiết các lớp được mô tả ở Bảng 5.6

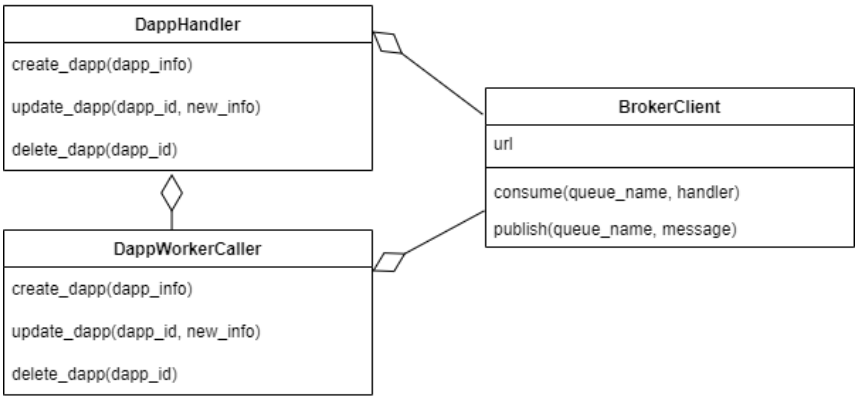
Bảng 5.6: Mô tả chi tiết các lớp của Network Worker

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
TaskHandler	Xử lý yêu cầu từ Network Service	create_network(network_info)	Xử lý yêu cầu tạo mạng với cấu hình network_info

Bảng 5.6: Mô tả chi tiết các lớp của Network Worker

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
		update_network(network_id, new_info)	Xử lý yều cầu cập nhật mạng có ID network_id với thông tin mới new_info
		delete_network(network_id)	Xử lý yêu cầu xóa mạng có ID network_id
		create_resource(resource_info)	Xử lý yêu cầu tạo node ngoài có cấu hình resource_info
CloudClient	Tương tác với dịch vụ điện toán đám mây	create_k8s_cluster (cluster_info)	Tạo một cụm Kubernetes
		delete_k8s_cluster (cluster_id)	Xóa một cụm Kubernetes
Deployer	Triển khai và quản lý mạng	deploy_ca()	Triển khai Nhà cung cấp chứng chỉ số
		deploy_orderer()	Triển khai Orderer node
		create_channel()	Tạo một kênh cho mạng
		deploy_peer()	Triển khai Peer node
K8SClient	Tương tác với cụm Kubernetes	apply()	Thực hiện tác vụ tạo hoặc cập nhật trên cụm Kubernets
		delete()	Thực hiện tác vụ tạo xóa trên cụm Kubernets
Generator	Tạo tệp tin	generate_network_files (network_info)	Tạo các tập tin cho quá trình tạo mạng
BrokerClient	Tương tác với message broker	Tương tự Core Service	Đường dẫn để kết nối tới message broker

5.2.6 Thiết kế Dapp Service



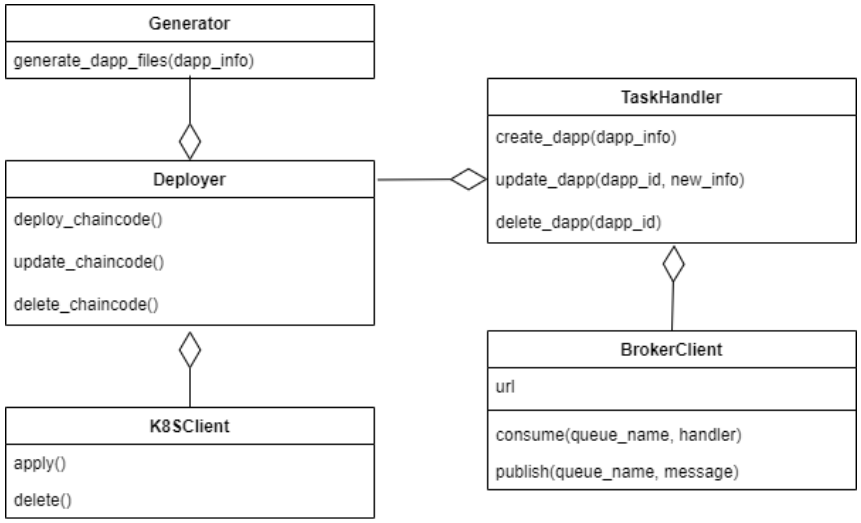
Hình 5.12: Biểu đồ lớp Dapp Service

Hình 5.12 mô tả biểu đồ lớp của Dapp Service. Thông tin các lớp được mô tả chi tiết ở Bảng 5.7

Bảng 5.7: Mô tả chi tiết các lớp của Dapp service

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
EventHandler	Xử lý yêu cầu từ Core Service	create_dapp(dapp_info)	Xử lý yêu cầu tạo ứng dụng với cấu hình dapp_info
		update_dapp(dapp_id, new_info)	Xử lý yêu cầu cập nhật ứng dụng có ID dapp_id với thông tin mới new_info
		delete_dapp(dapp_id)	Xử lý yêu cầu xóa ứng dụng có ID dapp_id
WorkerCaller	Yêu cầu Dapp Worker thực hiện tác vụ	create_dapp(dapp_info)	Yêu cầu tạo ứng dụng với cấu hình dapp_info
		update_dapp(dapp_id, new_info)	Yêu cầu cập nhật ứng dụng có ID dapp_id với thông tin mới new_info
		delete_dapp(dapp_id)	Yêu cầu xóa ứng dụng có ID dapp_id
BrokerClient	Tương tác với message broker	Tương tự Core Service	

5.2.7 Thiết kế Dapp Worker



Hình 5.13: Biểu đồ lớp Dapp Worker

Hình 5.13 mô tả biểu đồ lớp của Dapp Worker. Chi tiết các lớp được mô tả ở Bảng 5.8

Bảng 5.8: Mô tả chi tiết các lớp của Dapp Worker

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
TaskHandler	Xử lý yêu cầu từ dapp Service	create_dapp(dapp_info)	Xử lý yêu cầu tạo ứng dụng với cấu hình dapp_info
		update_dapp(dapp_id, new_info)	Xử lý yêu cầu cập nhật ứng dụng có ID dapp_id với thông tin mới new_info
		delete_dapp(dapp_id)	Xử lý yêu cầu xóa ứng dụng có ID dapp_id
Deployer	Triển khai và quản lý ứng dụng	deploy_chaincode()	Tạo và triển khai chaincode
		update_chaincode()	Cập nhật chaincode
		delete_chaincode()	Xóa chaincode
K8SClient	Tương tác với cụm Kubernetes	Tương tự Network Worker	
Generator	Tạo tệp tin	generate_dapp_files (dapp_info)	Tạo các tập tin cho quá trình tạo ứng dụng

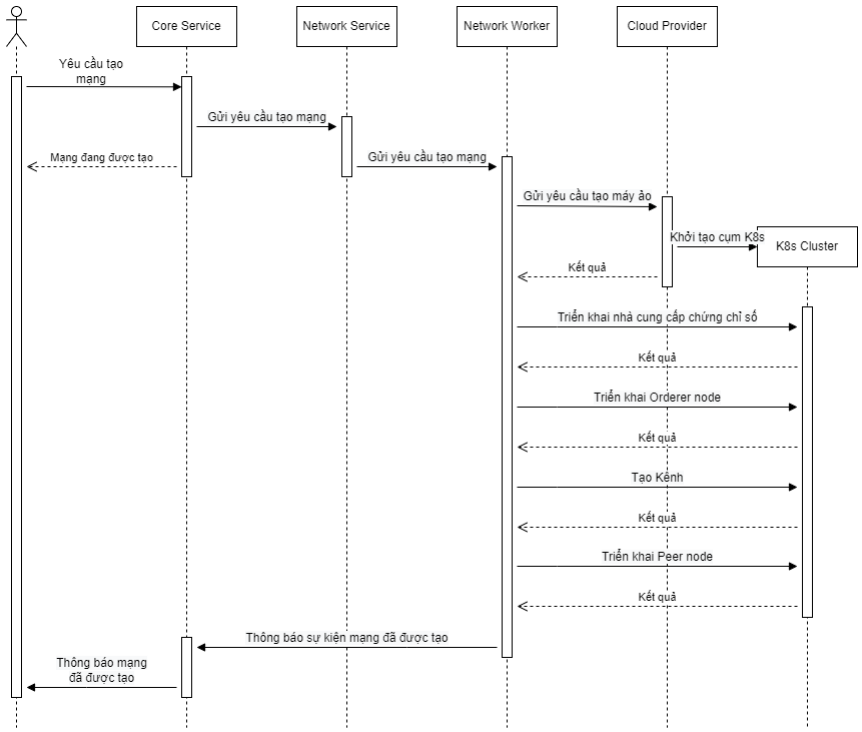
Bảng 5.8: Mô tả chi tiết các lớp của Dapp Worker

Tên lớp	Mô tả lớp	Tên thuộc tính/phương thức	Mô tả thuộc tính/phương thức
BrokerClient	Tương tác với message broker	Tương tự Core Service	

5.2.8 Các luồng hoạt động chính

Mục này sẽ mô tả trình tự tương tác giữa các dịch vụ trong hệ thống để thực hiện yêu cầu từ người dùng.

a, Tạo mạng



Hình 5.14: Biểu đồ trình tự tạo mạng

Hình 5.14 mô tả luồng hoạt động của hệ thống để triển khai một mạng chuỗi khối Hyperledger Fabric.

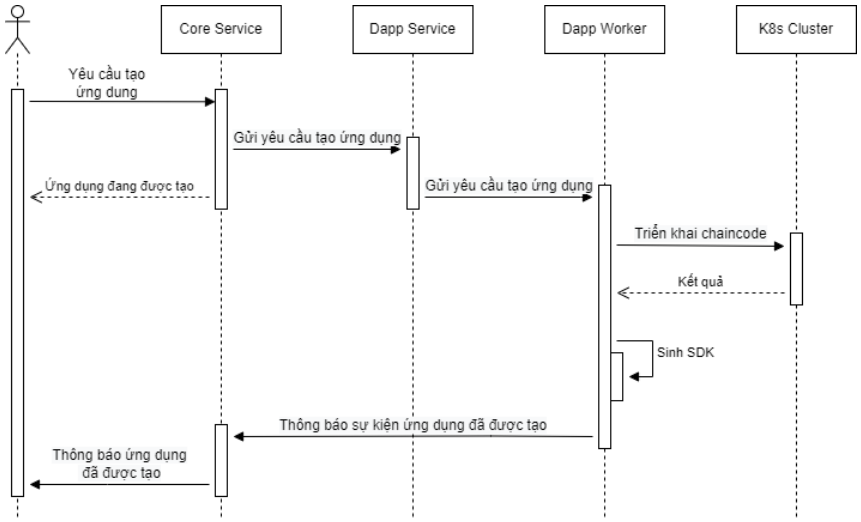
Trước hết, người dùng sử dụng giao diện để lựa chọn cấu hình cho mạng chuỗi khối mới. Giao diện sẽ gửi thông tin mạng cần tạo về cho Core Service, dịch vụ này sau đó sẽ gửi về thông báo rằng mạng đang được khởi tạo. Core Service sau đó gửi yêu cầu tạo mạng tới Network Service. Network Service yêu cầu 1 trong số các Network Worker thực hiện tác vụ tạo mạng.

Quá trình tạo mạng bắt đầu với việc Network Worker yêu cầu tạo một cụm máy ảo tương ứng với cấu hình người dùng yêu cầu tới nhà cung cấp dịch vụ điện toán

đám mây. Sau khi một cụm Kubernetes mới đã được tạo, Network Worker sẽ tiến hành triển khai một mạng Hyperledger Fabric lên trên cụm Kubernetes này. Bước đầu tiên là triển khai các Nhà cung cấp chứng chỉ số. Tiếp đó, sử dụng chứng từ số được tạo bởi các nhà cung cấp này, tiến hành triển khai Orderer Node. Tiếp đến, thực hiện cấu hình và triển khai một kênh cho các tổ chức trong mạng theo như yêu cầu người dùng đã gửi. Kế đến, các Peer node trực thuộc các tổ chức sẽ được triển khai và tham gia vào kênh này. Bước cuối cùng trong quá trình tạo mạng là triển khai một ứng dụng Explorer để có thể theo dõi các hoạt động trong mạng.

Sau khi triển khai mạng hoàn tất, Network Worker thông báo với Core Service. Core Service sẽ thông báo tới giao diện mạng đã triển khai thành công.

b, Tạo ứng dụng phi tập trung



Hình 5.15: Biểu đồ trình tự tạo ứng dụng phi tập trung

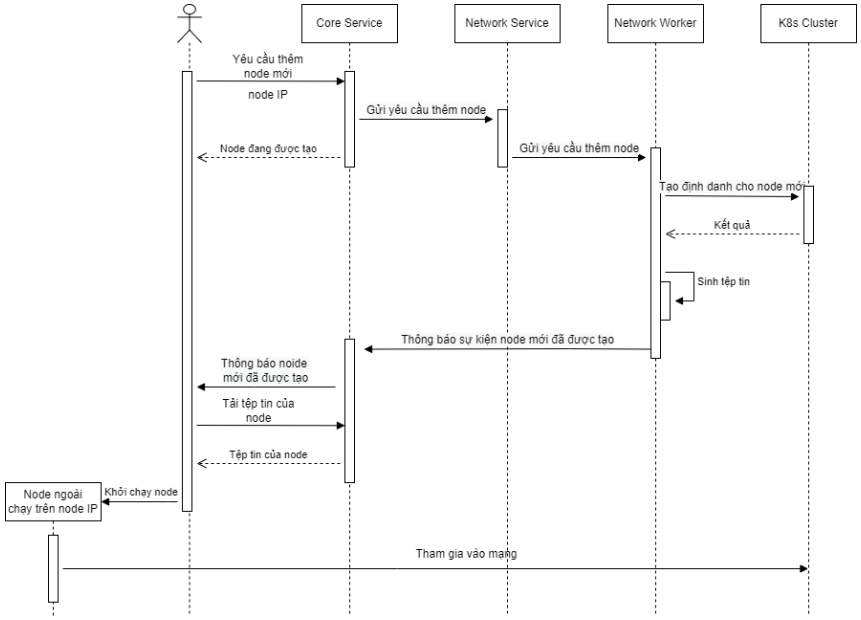
Hình 5.15 mô tả luồng hoạt động của hệ thống để triển khai một ứng dụng phi tập trung cho một mạng Hyperledger Fabric.

Tương tự như quá trình tạo mạng, người dùng gửi yêu cầu tới Core Service để tạo một ứng dụng phi tập trung mới cho một mạng của người dùng. Thông qua giao diện, cấu trúc của ứng dụng sẽ được định nghĩa với các thực thể sở hữu nhiều thuộc tính có quan hệ với nhau. Yêu cầu tạo ứng dụng này sẽ được chuyển đến Dapp Worker để tiến hành sinh và triển khai ứng dụng.

Từ cấu trúc mà người dùng yêu cầu, Dapp Worker sẽ sinh ra một chaincode tương ứng. Chaincode này sẽ được triển khai lên cụm Kubernetes mà mạng chuỗi khối đang chạy. Một SDK để tương tác với chaincode sau đó sẽ được tạo ra.

Sau khi triển khai ứng dụng thành công, Dapp Worker cũng sẽ gửi thông tin về Core Service để dịch vụ này thông báo cho người dùng.

c, Thêm node ngoài của người dùng vào mạng



Hình 5.16: Biểu đồ trình tự thêm node ngoài vào mạng

Hình 5.14 mô tả luồng hoạt động của hệ thống để thêm một node ngoài vào mạng.

Người dùng gủi yêu cầu thêm node với cấu hình và địa chỉ IP mã node sẽ chạy tới Core Service. Yêu cầu sẽ được chuyển tiếp tới Network Service. Dịch vụ này sẽ tương tác với mạng đang chạy để tạo một định danh tương ứng cho node mới này. Các tệp tin để khởi chạy node đó cũng sẽ được sinh ra.

Khi tệp tin đã được tạo người dùng sẽ nhận được thông báo từ Core Service. Người dùng từ đó tải các tệp tin này về, khởi chạy node trên máy có địa chỉ IP mà người dùng cung cấp lúc đầu. Node mới khi được khởi chạy sẽ tự động tham gia vào mạng Hyperledger Fabric của người dùng.

5.3 Xây dựng hệ thống

5.3.1 Thư viện và công cụ sử dụng

Quá trình xây dựng và phát triển các dịch vụ trong hệ thống sử dụng những công cụ và thư viện được nêu ở Bảng 5.9.

Bảng 5.9: Danh sách thư viện và công cụ sử dụng

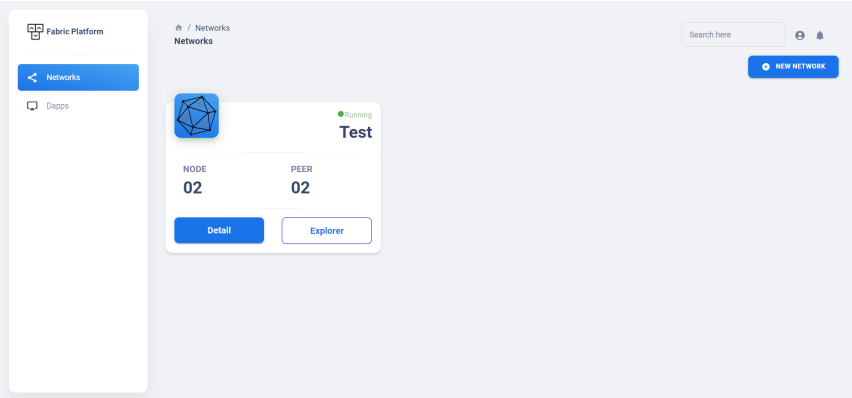
Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Visual Code Studio	`https://code.visualstudio.com/`
Triển khai hệ thống	Docker	`https://www.docker.com`

Bảng 5.9: Danh sách thư viện và công cụ sử dụng

Mục đích	**Công cụ**	**Địa chỉ URL**
Ngôn ngữ lập trình hệ thống	Python	https://www.python.org/
Thư viện xây dựng Server http	aiohttp	https://pypi.org/project/aiohttp/
Thư viện kết nối tới RabbitMQ	aio_pika	https://aio-pika.readthedocs.io/en/latest/
Thư viện sinh tệp tin theo template	Jinja	https://jinja.palletsprojects.com/en/3.1.x/
Tương tác với cụm Kubernetes	kubectl	https://kubernetes.io/docs/reference/kubectl/
Tương tác với mạng Hyperledger Fabric	Fabric CLI	https://github.com/hyperledger/fabric-cli
Xây dựng CA cho mạng Hyperledger Fabric	Fabric CA	https://github.com/hyperledger/fabric-ca
Thư viện xây dựng giao diện web	ReactJS	https://ReactJS.org
Thư viện quản lý trạng thái cho ứng dụng web	EJ2 React Diagrams	https://redux.js.org
Thư viện vẽ biểu đồ cơ sở dữ liệu quan hệ	react-flow-renderer	https://www.npmjs.com/package/react-flow-renderer
Ngôn ngữ lập trình cho SDK	Javascript	https://developer.mozilla.org/enUS/docs/Web/JavaScript
Thư viện sử dụng để xây dựng SDK	fabric-network	https://hyperledger.github.io/fabric-sdk-node/
Xây dựng explorer cho mạng Hyperledger Fabric	Hyperledger Explorer	https://github.com/hyperledger/blockchain-explorer
Xây dựng hệ thống thông báo thông qua web socket	Notification-socket.io	https://github.com/netbulls/notification-socket.io

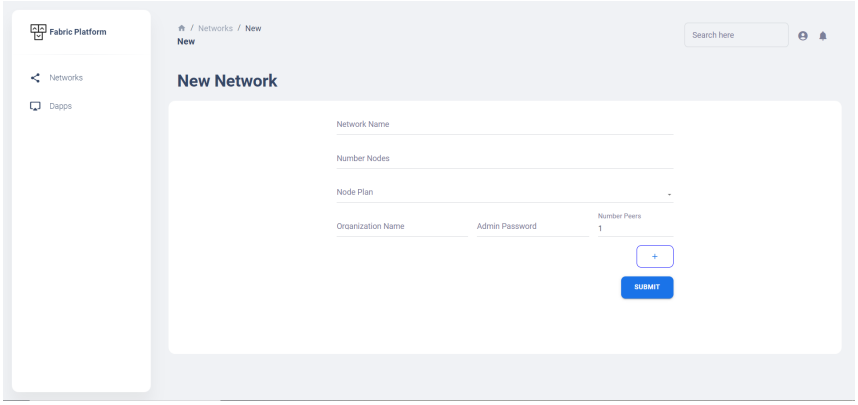
5.3.2 Kết quả đạt được

5.3.3 Minh họa các chức năng chính



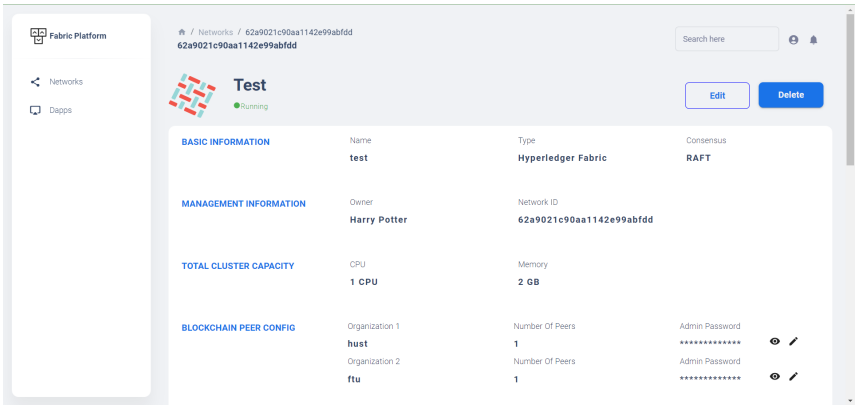
Hình 5.17: Màn hình danh sách mạng

Hình 5.17 mô tả màn hình danh sách mạng của người dùng. Tại màn hình này, người dùng có thể thấy thông tin tổng quan nhất của các mạng cùng với đó là truy cập vào Explorer của mạng. Người dùng có thể chọn "New network" để tiến hành tạo mạng mới.

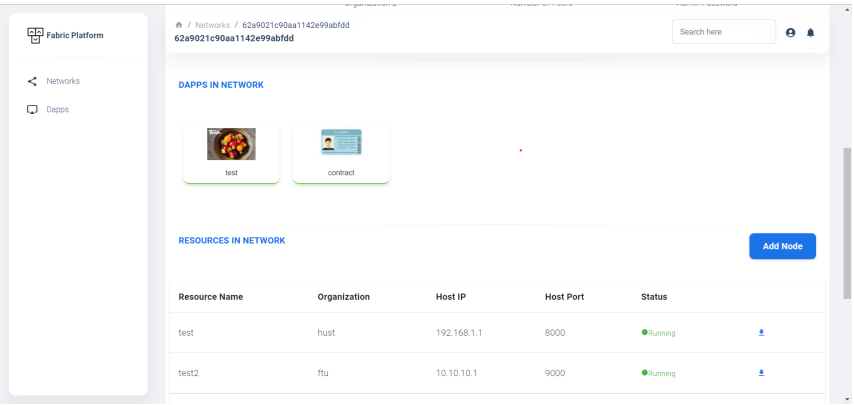


Hình 5.18: Màn hình tạo mạng mới

Hình 5.18 mô tả màn hình tạo mạng mới. Một mạng mới tương ứng với thông tin người dùng nhập sẽ được tạo sau khi nhấn Submit.

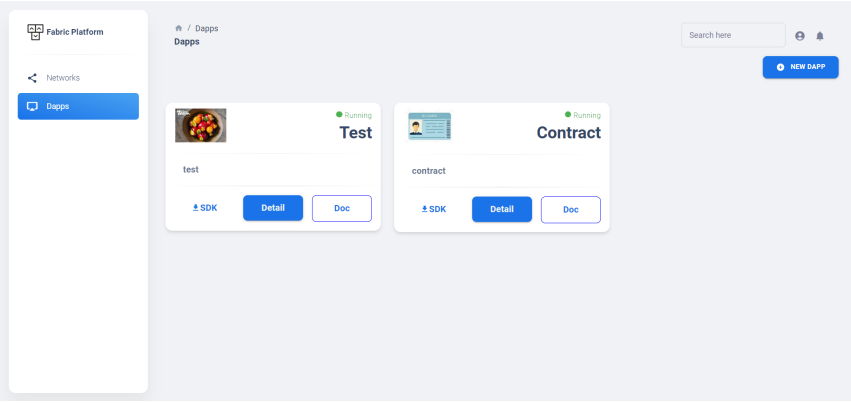


Hình 5.19: Màn hình thông tin mạng 1



Hình 5.20: Màn hình thông tin mạng 2

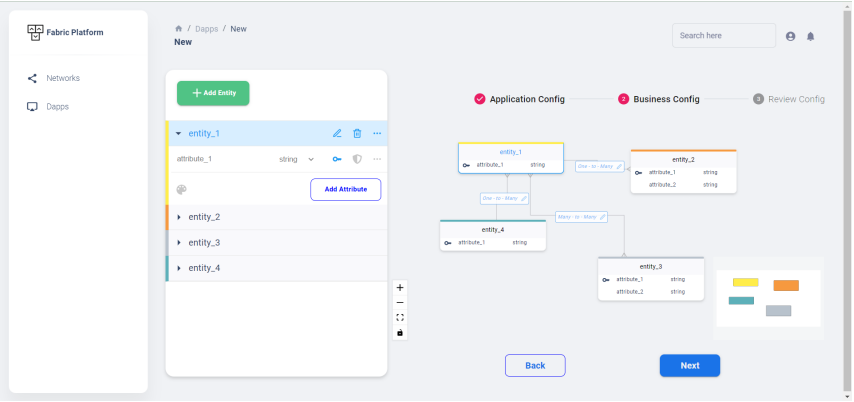
Hình 5.19 và 5.20 mô tả màn hình chi tiết thông tin của một mạng. Ngoài thông tin cơ bản của mạng, danh sách ứng dụng và node ngoài thuộc về mạng cũng được hiện thị. Ngoài ra người dùng có thể thực hiện xóa mạng thông qua nút "Delete", thêm tổ chức quá nút "Edit" và thêm node ngoài thông qua nút "Add node".



Hình 5.21: Màn hình danh sách ứng dụng

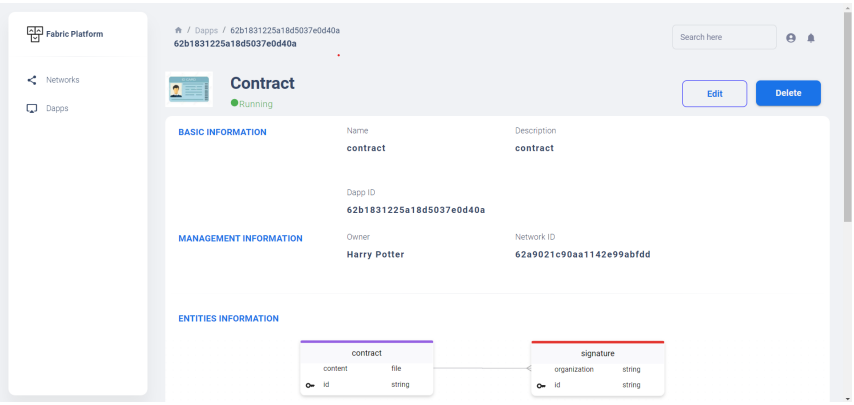
Hình 5.21 mô tả danh sách ứng dụng phi tập trung của người dùng. Ngoài thông tin cơ bản, người dùng có thể tải SDK và truy cập Tài liệu mô tả SDK. Người dùng

có thể tạo ứng dụng mới thông qua nút "New Dapp".



Hình 5.22: Màn hình thiết kế cấu trúc ứng dụng

Hình 5.22 mô tả màn hình thiết kế cấu trúc ứng dụng phi tập trung mới. Người dùng có thể tạo các thực thể với nhiều thuộc tính. Rồi kết nối các thực thể này với nhau thông qua một giao diện kéo thả.



Hình 5.23: Màn hình thông tin ứng dụng

Hình 5.23 mô tả màn hình thông tiết chi tiết của ứng dụng. Ngoài thông tin, cấu trúc tổng quan của ứng dụng cũng được hiển thị.

5.4 Kiểm thử

5.4.1 Kiểm thử hệ thống

Để kiểm thử tác vụ tạo mạng và tạo ứng dụng phi tập trung mới, các tác vụ đó sẽ được thực hiện 10 lần và đo thời gian trung bình cần để hoàn thành các tác vụ đó. Kết quả được trình bày ở Bảng 5.10 và 5.11

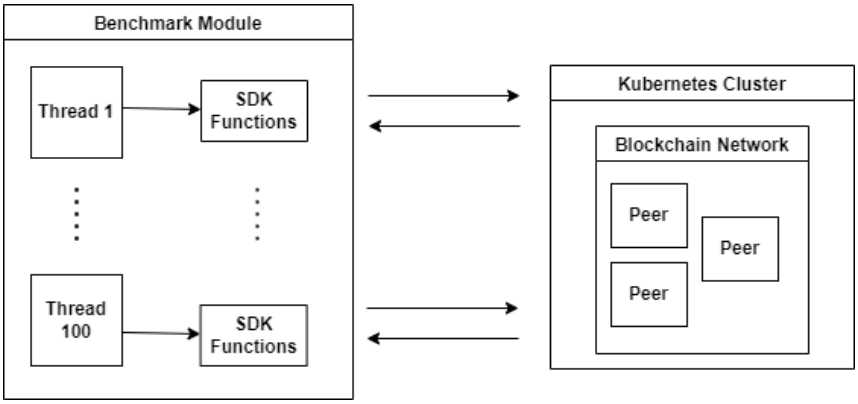
Bảng 5.10: Kết quả kiểm thử tác vụ tạo mạng

Test Case ID	TC1-1
Tác vụ	Tạo mạng
Mô tả	Đo thời gian tạo mạng trung bình
Dữ liệu kiểm thử	Số máy ảo: 3 Số tổ chức: 3 tổ chức mỗi tổ chức 3 peer Cấu hình máy ảo: 4GB Ram, 2 vCpus
Điều kiện tiền đề	Đã đăng nhập
Các bước thực hiện	Bước 1: Đăng nhập Bước 2: Chạy kiểm thử với dữ liệu kiểm thử Bước 3: Lưu kết quả
Kết quả mong muốn	Thời gian tạo mạng trung bình: 900 giây
Kết quả thực tế	Thời gian tạo mạng trung bình: 857 giây

Bảng 5.11: Kết quả kiểm thử tác vụ tạo ứng dụng phi tập trung

Test Case ID	TC1-2
Tác vụ	Tạo ứng dụng
Mô tả	Đo thời gian tạo ứng dụng trung bình
Dữ liệu kiểm thử	Một ứng dụng có 3 thực thể
Điều kiện tiền đề	Đã đăng nhập
Các bước thực hiện	Bước 1: Đăng nhập Bước 2: Chạy kiểm thử với dữ liệu kiểm thử Bước 3: Lưu kết quả
Kết quả mong muốn	Thời gian tạo ứng dụng trung bình: 200 giây
Kết quả thực tế	Thời gian tạo ứng dụng trung bình: 201 giây

5.4.2 Kiểm thử SDK



Hình 5.24: Mô-đun kiểm thử SDK

Hình 5.24 mô tả mô-đun sử dung để kiểm thử SDK. Mô-đun có thể gọi đồng thời các hàm của SDK trên nhiều luồng.

Hình 5.25: Cấu trúc ứng dụng để kiểm thử SDK

Để kiểm thử SDK, trước tiên tải về SDK của một ứng dụng có cấu trúc như hình 5.25. Tiếp đó sử dụng mô-đun kiểm thử SDK ở hình 5.25 để giả lập việc gọi đồng thời các hàm trong SDK 100 lần rồi đo thời gian cần để hoàn thành tất cả 100 lần gọi đó. Kết quả kiểm thử được mô tả ở Bảng 5.12 và **??**.

Bảng 5.12: Kết quả kiểm thử các tác vụ đọc dữ liệu sử dụng SDK

Test Case ID	TC2-1
Tác vụ	Đọc dữ liệu từ Sổ cái sử dụng SDK
Mô tả	Đo thời gian đọc dữ liệu sử dụng SDK trung bình
Điều kiện tiền đề	Đã đăng nhập
Các bước thực hiện	Bước 1: Tải SDK Bước 2: Chạy mô-đun kiểm thử cho các hàm đọc Bước 3: Lưu kết quả
Kết quả mong muốn	Tất cả các lần gọi hàm đều thành công Tổng thời gian gọi: 1 giây
Kết quả thực tế	Tất cả các lần gọi hàm đều thành công Tổng thời gian gọi: 0.41 giây Thời gian trung binh 1 lần gọi: 0.004 giây

Bảng 5.13: Kết quả kiểm thử các tác vụ ghi dữ liệu sử dụng SDK

Test Case ID	TC1-2
Tác vụ	Tạo ứng dụng
Mô tả	Đo thời gian ghi dữ liệu sử dụng SDK trung bình
Điều kiện tiền đề	Đã đăng nhập
Các bước thực hiện	Bước 1: Tải SDK Bước 2: Chạy mô-đun kiểm thử cho các hàm ghi Bước 3: Lưu kết quả
Kết quả mong muốn	Tất cả các lần gọi hàm đều thành công Tổng thời gian gọi: 2 giây
Kết quả thực tế	Tất cả các lần gọi hàm đều thành công Tổng thời gian gọi: 1.62 giây Thời gian trung binh 1 lần gọi: 0.016 giây

Kết quả kiểm thử cho thấy hệ thống hoạt động ổn định. Các yêu cầu tạo mạng và ứng dụng đều được xử lý thành công. Các tác vụ tương tác với mạng chuỗi khối thông qua SDK nhanh và chính xác.

5.5 Ứng dụng của hệ thống

Để chứng minh mạng và SDK có thể được ứng dụng vào các nghiệp vụ thực tế, tôi đã xây dựng một ứng dụng web đơn giản tương tác với mạng và ứng dụng phi tập trung được tạo bởi hệ thống. Ứng dụng được gọi tên là Fabric Contract.

5.5.1 Tổng quan ứng dụng

Fabric Contract giải quyết bài toán sử dụng công nghệ chuỗi khối để số hóa và minh bạch hóa quá trình ký hợp đồng giữa các doanh nghiệp, tổ chức. Công nghệ chuỗi khối được sử dụng như một bằng chứng không thể chối cái việc các bên liên đã đồng ý và ký vào một hợp đồng cụ thể nào đó.

Cụ thể ứng dụng cho phép một tổ chức tạo một hợp đồng dưới định dạng tệp tin pdf. Đại diện của các tổ chức sẽ thể hiện sự đồng thuận của mình với hợp đồng đó bằng cách ký lên hợp đồng đó. Các tác vụ tạo, ký hợp đồng sẽ được lưu trữ lên trên mạng chuỗi khối.

5.5.2 Cấu hình

Mạng mà ứng dụng sử dụng có cấu hình như Hình 5.26

Hình 5.26: Cấu hình mạng triển khai Fabric Contract

Fabric Contract sẽ được sử dụng trong tác vụ tạo và ký hợp đồng giữa 2 tổ chức trong mạng.

Ứng dụng phi tập trung mà Fabric Contract sử dụng có cấu hình như Hình **??**

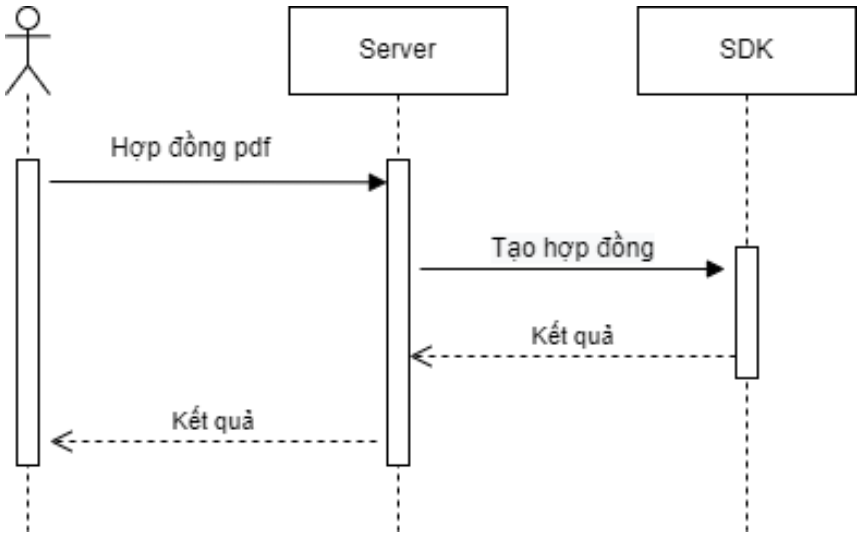
Hình 5.27: Cấu hình mạng triển khai Fabric Contract

Dữ liệu được lưu trữ trên mạng chuỗi khối bao gồm nội dung hợp đồng và các

chữ ký từ các tổ chức của hợp đồng đó. Từ các thông tin này có thể đảm bảo một hợp đồng thực sự đã được chấp thuận bởi các bên liên quan.

5.5.3 Luồng hoạt động

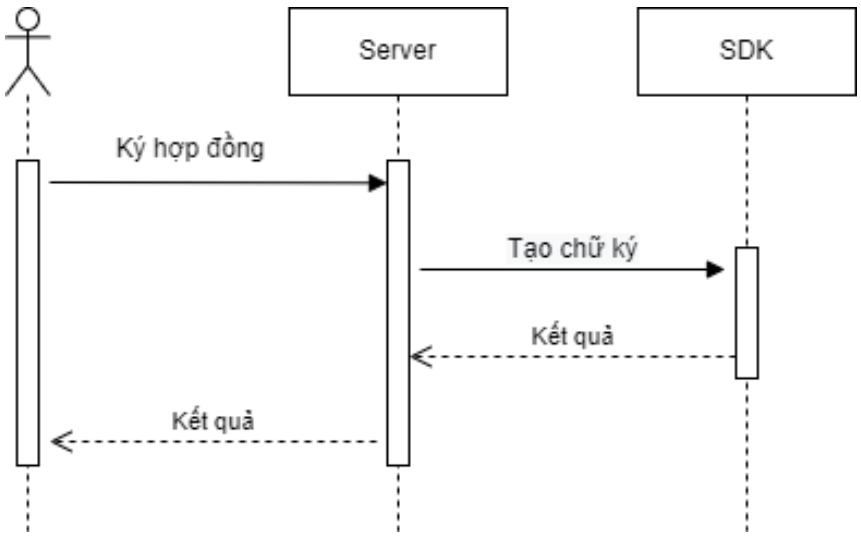
a, Tạo hợp đồng



Hình 5.28: Luồng tạo hợp đồng của Fabric Contract

Hình 5.28 mô tả luồng tạo hợp đồng của ứng dụng Fabric Contract. Người dùng tải lên một tệp tin pdf tới Server. Server sử dụng SDK sẽ tiến hành tạo một hợp đồng mới tương ứng với tệp tin đó và ghi dữ liệu lên mạng chuỗi khối.

b, Ký hợp đồng

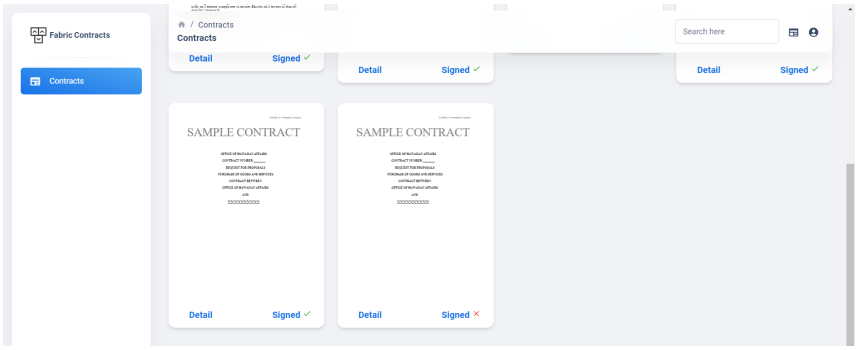


Hình 5.29: Luồng ký hợp đồng của Fabric Contract

Hình 5.29 mô tả luồng tạo hợp đồng của ứng dụng Fabric Contract. Người dùng đại diện cho một tổ chức gửi yêu cầu ký hợp đồng tới Server. Server sử dụng SDK

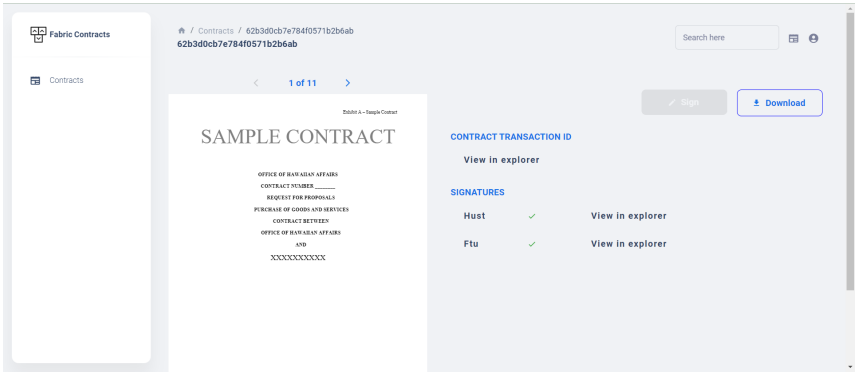
sẽ tiến hành tạo một chữ ký mới cho hợp đồng đó và cũng ghi dữ liệu lên mạng chuỗi khối.

5.5.4 Minh họa hoạt động



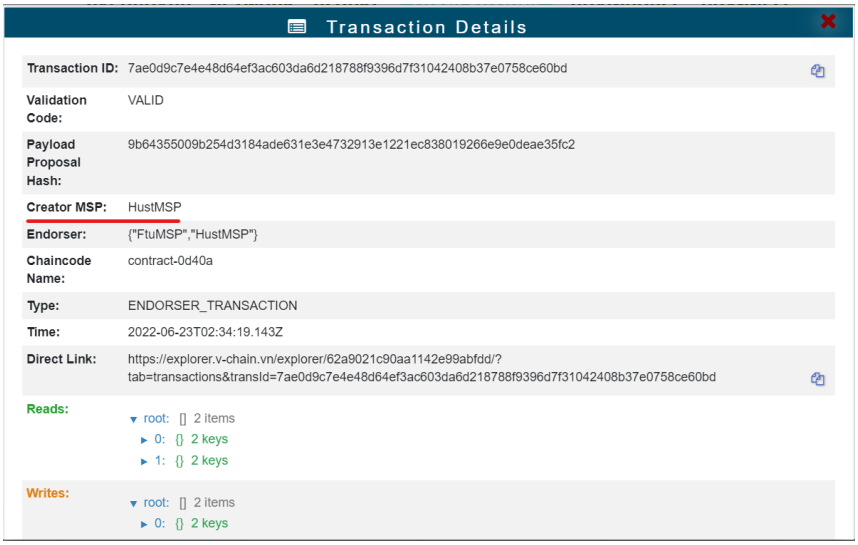
Hình 5.30: Màn hình danh sách hợp đồng Fabric Contract

Hình 5.30 mô tả danh sách các hợp đồng đã được tạo và trạng thái hợp đồng đã được ký bởi hai bên hay chưa.



Hình 5.31: Màn hình chi tiết hợp đồng Fabric Contract

Hình 5.31 mô tả mà hình chi tiết của một hợp đồng. Người dùng có thể thực hiện tác vụ ký tại màn hình này. Ngoài ra, người dùng có thể xem chi tiết các giao dịch trên mạng chuỗi khối liên quan đến việc tạo và ký hợp đồng này.



Hình 5.32: Giao dịch ký hợp đồng trên mạng chuỗi khối

Hình 5.32 mô tả thông tin về giao dịch để ký hợp đồng được xem thông qua ứng dụng Explorer của mạng Hyperledger Fabric. Có thể thấy rõ thông tin của tổ chức tạo giao dịch này thông qua CreatorMSP. Thông tin này sẽ được sử dụng để chứng minh một tổ chức đã chấp thuận và ký lên một hợp đồng.

Chương 5 đã trình bày về chi tiết thiết kế xây dựng cũng như quá trình kiểm thử đánh giá hệ thống. Đây là cơ sở quan trọng để đi đến Chương 6, chương cuối cùng.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau quá trình thực hiện đồ án, tôi đã phát triển được hệ thống triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric với khả năng đơn giản hóa quá trình ứng dụng công nghệ chuỗi khối nói chung và mạng chuỗi khối riêng tư nói riêng vào các nghiệp vụ thực tế. Để đạt được mục tiêu này, tôi đã tìm hiểu về chi tiết kiến trúc mạng, cùng với đó là cách để triển khai, vận hành hạ tầng mạng trên nền tảng Kubernetes và điện toán đám mây. Để được ứng dụng vào các nghiệp vụ cụ thể, tôi cũng đã đơn giản và trực quan hóa quá trình phát triển ứng dụng phi tập trung thông qua mô hình cơ sở dữ liệu quan hệ.

Hệ thống trong đồ án này đã được tích hợp trong nền tảng V-chain[1]. V-chain là dự án được tài trợ bởi Vingroup Innovation Fund với các dịch vụ tương tác với nhiều loại mạng chuỗi khối khác nhau từ riêng tư tới công khai. Hệ thống trong đồ án được sử dụng trong V-chain cũng với mục đích tạo và triển khai mạng và ứng dụng phi tập trung dựa trên nền tảng Hyperledger Fabric.

6.2 Hướng phát triển

Hiện nay, khi tạo một mạng mới, hệ thống mới chỉ cho phép khởi tạo mới các tổ chức. Trong tương lai, tôi muốn phát triển để các tổ chức sau khi được tạo có thể tham gia vào cả các mạng chuỗi khối khác. Thêm vào đó, một tính năng nổi bật khác của mạng Hyperledger Fabric chưa được tận dụng trong đồ án này đó là khả năng tách biệt giao dịch giữa các nhóm nhiều tổ chức ngay cả trong cùng một kênh.

Về cấu trúc của ứng dụng phi tập trung, ngoài mô hình cơ sở dữ liệu quan hệ, tôi muốn phát triển nhiều kiểu ứng dụng phi tập trung khác. Nổi bật nhất có thể kể đến các ứng dụng liên quan đến trao đổi và mua bán tài sản số. Ngôn ngữ của SDK hiện tại cũng chỉ giới hạn ở Javascript, trong tương lai tôi muốn thêm SDK hỗ trợ các ngôn ngữ lập trình phổ biến khác như Python, Golang.

TÀI LIỆU THAM KHẢO

[1] *V-chain - decentralized application development and deployment platform on blockchain technology*, https://v-chain.vn/, Last visited 17-July-2022.

[2] *Amazon managed blockchain*, https://aws.amazon.com/vi/managed-blockchain/, Last visited 17-July-2022.

[3] *Fabric guide*, https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html, Last visited 17-July-2022.

[4] *Kubernetes*, https://kubernetes.io/, Last visited 17-July-2022.

[5] *The most popular database for modern apps, mongodb*, https://www.mongodb.com, Last visited 17-July-2022.

[6] *Messaging that just works — rabbitmq*, https://www.rabbitmq.com, Last visited 17-July-2022.

[7] *Introduction to celery*, https://docs.celeryq.dev/en/stable/getting-started/introduction.html, Last visited 17-July-2022.