**LỜI MỞ ĐẦU**

Để hoàn thành được đồ án này, nhóm em xin gửi lời cảm ơn sâu sắc nhất của chúng em tới thầy Hoàng Tuấn Hảo, bộ môn An toàn thông tin khoa Công Nghệ Thông Tin, Học Viện Kỹ Thuật Quân Sự đã nhận hướng dẫn và tin tưởng để giao đề tài này cho chúng em. Trong thời gian thực hiện đồ án, thầy đã nhiệt tình hướng dẫn và giúp đỡ chúng em rất nhiều. Nhờ những sự hiểu biết sâu rộng và kinh nghiệm nghiên cứu khoa học của thầy đã giúp nhóm em vượt qua những khó khăn khi thực hiện đồ án này.

Em cũng muốn bày tỏ sự cảm ơn của mình tới các quý thầy cô trong bộ môn Công nghệ mạng cũng như các thầy, các cô trong trường đã giảng dạy, giúp đỡ em trong suốt những năm học vừa qua. Nhờ có các thầy, các cô cung cấp cho chúng em những kiến thức nền tảng và kiến thức chuyên môn mà em có thể hoàn thành đồ án và xa hơn là phục vụ cho công việc sau này.

*Em xin chân thành cảm ơn!*

**BẢNG GIẢI THÍCH CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Từ đầy đủ** | **Nghĩa của từ** |
| HF | Hyperledger Fabric |  |
| HC | Hyperledger Composer | Chaincode trong hệ thống Hyperledger |
| MSP | Membership Service Provider | Nhà cung cấp dịch vụ thành viên |
| CA | Certificate Authority | Cơ quan cấp chứng chỉ |
| DLT | Distributed Ledger Technology | Công nghệ sổ cái phân tán |
| P2P | Peer to Peer | Mạng máy tính ngang hàng |
| MSP | Membership Service Provider | Cung cấp dịch vụ thành viên |
| REST | Representational State Transfer | Dạng chuyển đổi cấu trúc dữ liệu |

**DANH MỤC HÌNH ẢNH**

[Hình 1. 1. Ứng dụng blockchain 3](#_Toc46310157)

[Hình 1. 2. Mô hình chuỗi khối 4](#_Toc46310158)

[Hình 1. 3. Tình bất biến của blockchain 4](#_Toc46310159)

[Hình 1. 4. Ledger ghi lại giao dịch trong mạng 5](#_Toc46310160)

[Hình 1. 5. Mã hóa thông tin 6](#_Toc46310161)

[Hình 1. 6. Version blockchain 6](#_Toc46310162)

[Hình 1. 7. Hyperledger Umbrella 9](#_Toc46310163)

[Hình 1. 8. Luồng giao dịch 11](#_Toc46310164)

[Hình 1. 9. Cấu trúc của sổ cái Hyperledger Fabric 13](#_Toc46310165)

[Hình 1. 10. Cấu trúc blockchain trong Hyperledger Fabric 14](#_Toc46310166)

[Hình 1. 11. Công nghệ IPFS 17](#_Toc46310167)

[Hình 1. 12. Môi trường thời gian chạy NodeJS – Đơn giản 21](#_Toc46310168)

[Hình 1. 13. Phụ thuộc gói cục bộ NodeJS được quản lý bởi NPM trong tệp package.json 22](#_Toc46310169)

[Hình 1. 14. Giới thiệu ExpressJS 23](#_Toc46310170)

[Hình 1. 15. Luồng công việc ExpressJS 24](#_Toc46310171)

[Hình 1. 16. REST API với ExpressJS 24](#_Toc46310172)

[Hình 1. 17. Chức năng Core AngularJS 25](#_Toc46310173)

[Hình 1. 18. Dự án ứng dụng web mã tự sinh yeoman 26](#_Toc46310174)

[Hình 2. 1. Biểu đồ phân rã chức năng 39](#_Toc46051939)

[Hình 3. 1. Kiến trúc tổng quan hệ thống Hyperledger 39](#_Toc46328419)

[Hình 3. 2. Mô hình hoạt động tổng quan hệ thống Hyperledger 39](#_Toc46328420)

[Hình 3. 3. Mô hình kiến trúc hệ thống quản lý lưu trữ công văn 40](#_Toc46328421)

[Hình 3. 4. Các thành phần định business network 41](#_Toc46328422)

[Hình 3. 5. Các bước để bắt đầu phát triển một ứng dụng 41](#_Toc46328423)

[Hình 3. 6. Thên công văn đi – Sơ đồ trình tự 42](#_Toc46328424)

[Hình 3. 7. Duyệt/hủy công văn đi – Sơ đồ trình tự 43](#_Toc46328425)

[Hình 3. 8. Transaction gửi công văn đi – Sơ đồ trình tự 43](#_Toc46328426)

[Hình 3. 9. Transaction cập nhật thông tin công văn đi – Sơ đồ trình tự 44](#_Toc46328427)

[Hình 3. 10. Transaction xóa công văn đi 44](#_Toc46328428)

[Hình 3. 11. Một đoạn chaincode dùng để thêm mới công văn đi 45](#_Toc46328429)

[Hình 3. 12. Phần quyền cho user 46](#_Toc46328430)

[Hình 3. 13. Phần quyền cho user (phòng ban) 47](#_Toc46328431)

[Hình 3. 14. Phần quyền văn thư 47](#_Toc46328432)

[Hình 3. 15. Phần quyền cho lãnh đạo 48](#_Toc46328433)

[Hình 3. 16. Hệ thống Use Case 48](#_Toc46328434)

[Hình 3. 17. Quy trình xử lý công văn đi 49](#_Toc46328435)

[Hình 3. 18. Định nghĩa các đối tượng công văn 52](#_Toc46328436)

[Hình 3. 19. Thể hiện của asset (công văn đi) cần để lưu vào blockchain. 53](#_Toc46328437)

[Hình 3. 20. Khai báo dữ liệu kiểu enum 54](#_Toc46328438)

[Hình 3. 21. Kiểu khai báo relationship trong asset (class) 54](#_Toc46328439)

[Hình 3. 22. Thể hiện kiểu relationship trong class 54](#_Toc46328440)

[Hình 3. 23. Khai báo kiểu mảng trong file model 54](#_Toc46328441)

[Hình 3. 24. Cấu trúc lưu trữ transactions khi tạo công văn 60](#_Toc46328442)

[Hình 3. 25. Cấu trúc lưu trữ transactions khi cập nhật công văn 60](#_Toc46328443)

[Hình 3. 26. Cấu trúc lưu trữ transactions khi xóa công văn 61](#_Toc46328444)

[Hình 3. 27. Cấu trúc lưu trữ transactions khi duyệt công văn 61](#_Toc46328445)

[Hình 3. 28. Cấu trúc lưu trữ transactions khi gửi công văn 61](#_Toc46328446)

[Hình 3. 29. Cấu trúc lưu trữ transactions khi hủy công văn 61](#_Toc46328447)

[Hình 3. 30. Giao diện đăng nhập 62](#_Toc46328448)

[Hình 3. 31. Giao điện tạo công văn đi 62](#_Toc46328449)

[Hình 3. 32. Giao diện duyệt / hủy công văn đi 63](#_Toc46328450)

[Hình 3. 33. Giao diện gửi công văn đi 63](#_Toc46328451)

[Hình 3. 34. Giao diện xóa công văn đi 63](#_Toc46328452)

[Hình 3. 35. Giao diện cập thông tin công văn đi 64](#_Toc46328453)

[Hình 3. 36. Giao diện xem thông tin cá nhân 64](#_Toc46328454)

**DANH MỤC BẢNG BIỂU**

[Bảng 2. 1. Yêu cầu chức năng đăng nhập 31](#_Toc46310189)

[Bảng 2. 2. Chức năng tạo công văn đi 32](#_Toc46310190)

[Bảng 2. 3. Chức năng cập nhật công văn đi 32](#_Toc46310191)

[Bảng 2. 4. Chức năng xóa công văn đi 33](#_Toc46310192)

[Bảng 2. 5. Chức năng duyệt công văn đi 34](#_Toc46310193)

[Bảng 2. 6. Chức năng gửi công văn đi 35](#_Toc46310194)

[Bảng 2. 7. Chức năng xem thông tin công văn đi 35](#_Toc46310195)

[Bảng 2. 8. Chức năng xem thông tin cá nhân 36](#_Toc46310196)

[Bảng 2. 9. Chức năng xem thông tin các lần giao dịch 36](#_Toc46310197)

[Bảng 3. 1. Usecase cho chức năng tạo công văn 51](#_Toc46310201)

[Bảng 3. 2. Usecase cho chức năng duyệt/hủy công văn 51](#_Toc46310202)

[Bảng 3. 3. Usecase cho chức năng gửi công văn 52](#_Toc46310203)

[Bảng 3. 4. Bảng chứa thông tin công văn đi 55](#_Toc46310204)

[Bảng 3. 5. Bảng chứa thông tin văn thư với quyền tạo/gửi công văn đi 57](#_Toc46310205)

[Bảng 3. 6. Bảng chứa thông tin lãnh đạo với quyền duyệt công văn đi 57](#_Toc46310206)

[Bảng 3. 7. Bảng chứa thông tin người dùng nhận với quyền xem công văn đi 58](#_Toc46310207)

[Bảng 3. 8. Bảng kiểu enumchứa thông tin nơi nhận phòng ban công văn 58](#_Toc46310208)

[Bảng 3. 9. Kiểu enum chứa thông tin đơn vị công văn 59](#_Toc46310209)

[Bảng 3. 10. Kiểu enum chứa thông tin lĩnh vực của công văn 59](#_Toc46310210)

[Bảng 3. 11. Kiểu enum chứa loại tài liệu của công văn 59](#_Toc46310211)

[Bảng 3. 12. Kiểu enum chứa trang thái xử lý của công văn 60](#_Toc46310212)

[Bảng 3. 13. Kiểu enum chứa thông tin giới tính user 60](#_Toc46310213)

[Bảng 4. 1. Kế hoạch kiểm thử 67](#_Toc46308122)

[Bảng 4. 2. Các rủi ro có thể xảy ra 67](#_Toc46308123)

[Bảng 4. 3. Kiểm thử chức năng đăng nhập 68](#_Toc46308124)

[Bảng 4. 4. Kiểm thử chức năng tạo công văn 69](#_Toc46308125)

[Bảng 4. 5. Kiểm thử chức năng duyệt / hủy công văn 72](#_Toc46308126)

[Bảng 4. 6. Kiểm thử chức năng gửi công văn 73](#_Toc46308127)

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc46677528)

[Chương 1 CƠ SỞ LÝ THUYẾT 3](#_Toc46677529)

[1.1. Giới thiệu công nghệ blockchain nền tảng 3](#_Toc46677530)

[1.1.1. Công nghệ blockchain và các đặc trưng liên quan 3](#_Toc46677531)

[1.1.2. Công nghệ Blockchain Hyperledger Fabric 8](#_Toc46677532)

[1.1.3. Công nghệ IPFS 16](#_Toc46677533)

[1.2. Công nghệ sử dụng xây dựng hệ thống 18](#_Toc46677534)

[1.2.1. Hyperledger Composer 18](#_Toc46677535)

[1.2.2. Kiến trúc RESTful 19](#_Toc46677536)

[1.2.2. NodeJS (running environment) 19](#_Toc46677537)

[1.2.3. NPM 20](#_Toc46677538)

[1.2.4. ExpressJS 22](#_Toc46677539)

[1.2.5. AngularJS 23](#_Toc46677540)

[1.2.6. Yeoman 25](#_Toc46677541)

[1.2.7. .NET Framework 26](#_Toc46677542)

[1.2.8. Công nghệ cài đặt hệ thống Hyperledger 27](#_Toc46677543)

[1.3. Công cụ hỗ trợ 27](#_Toc46677544)

[1.3.1. VMware Workstation 27](#_Toc46677545)

[1.3.2. Postman 27](#_Toc46677546)

[1.3.3. Visual Studio 2019 27](#_Toc46677547)

[Tổng luận chương 28](#_Toc46677548)

[Chương 2 YÊU CẦU HỆ THỐNG 29](#_Toc46677549)

[2.1. Mô tả hệ thống 29](#_Toc46677550)

[2.1.1. Các tính năng chính của ứng dụng 29](#_Toc46677551)

[2.1.2. Đặc điểm người sử dụng 29](#_Toc46677552)

[2.1.3. Môi trường vận hành 29](#_Toc46677553)

[2.1.4. Các ràng buộc về thực thi và thiết kế 30](#_Toc46677554)

[2.1.5. Các yêu cầu giao tiếp bên ngoài 30](#_Toc46677555)

[2.2. Phân rã chức năng 31](#_Toc46677556)

[2.2. Các yêu cầu chức năng 31](#_Toc46677557)

[2.2.1. Đăng nhập 31](#_Toc46677558)

[2.2.2. Tạo công văn đi 32](#_Toc46677559)

[2.2.3. Cập nhật công văn đi 32](#_Toc46677560)

[2.2.4. Xóa công văn đi 33](#_Toc46677561)

[2.2.5. Duyệt công văn điện tử 34](#_Toc46677562)

[2.2.6. Gửi công văn đi 34](#_Toc46677563)

[2.2.7. Xem thông công văn đi 35](#_Toc46677564)

[2.2.8. Xem thông tin cá nhân 36](#_Toc46677565)

[2.2.9. Xem thông tin các lần giao dịch 36](#_Toc46677566)

[2.4. Các yêu cầu phi chức năng 37](#_Toc46677567)

[2.4.1. Yêu cầu thực thi 37](#_Toc46677568)

[2.4.2. Yêu cầu an toàn 37](#_Toc46677569)

[2.4.3. Yêu cầu bảo mật 37](#_Toc46677570)

[2.4.4 Các đặc điểm chất lượng phần mềm 37](#_Toc46677571)

[Tổng kết chương 38](#_Toc46677572)

[Chương 3: THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG 39](#_Toc46677573)

[3.1. Kiến trúc hệ thống 39](#_Toc46677574)

[3.1.1. Thiết kế kiến trúc 39](#_Toc46677575)

[3.1.2. Systems workflow 43](#_Toc46677576)

[3.1.3. Các quy trình xử lý 45](#_Toc46677577)

[3.2. Thiết kế dữ liệu 51](#_Toc46677578)

[3.2.1. Các thành phần cơ bản của file model 51](#_Toc46677579)

[3.2.2. Các kiểu dữ liệu được dùng trong model file 53](#_Toc46677580)

[3.2.3. Thiết kế cơ sở dữ liệu 55](#_Toc46677581)

[3.2.4. Thiêt kế transactions 61](#_Toc46677582)

[3.3. Thiết kế giao diện 62](#_Toc46677583)

[3.3.1. Đăng nhập 62](#_Toc46677584)

[3.3.2. Tạo công văn đi 63](#_Toc46677585)

[3.3.3. Duyệt công văn đi 64](#_Toc46677586)

[3.3.4. Xác nhận gửi công văn đi 64](#_Toc46677587)

[3.3.5. Xác nhận xóa 64](#_Toc46677588)

[3.3.6. Cập nhật thông tin công văn 65](#_Toc46677589)

[3.3.7. Xem thông tin cá nhân 65](#_Toc46677590)

[Tổng kết chương 65](#_Toc46677591)

[Chương 4 KIỂM THỬ VÀ ĐÁNH GIÁ 66](#_Toc46677592)

[4.1. Giới thiệu 66](#_Toc46677593)

[4.1.1. Mục tiêu 66](#_Toc46677594)

[4.1.2. Phạm vi kiếm thử 66](#_Toc46677595)

[4.2. Chi tiết kế hoạch kiểm thử 66](#_Toc46677596)

[4.2.1. Các chức năng sẽ được kiểm thử 66](#_Toc46677597)

[4.2.2. Các chức năng sẽ không được kiểm thử 66](#_Toc46677598)

[4.2.3. Tiêu chí kiểm thử thành công/thất bại 66](#_Toc46677599)

[4.3. Quản lý kiểm thử 66](#_Toc46677600)

[4.3.1. Tiến hành kiểm thử 66](#_Toc46677601)

[4.3.2. Môi trường kiểm thử 67](#_Toc46677602)

[4.4. Kế hoạch dự đoán và chi phí 67](#_Toc46677603)

[4.5. Các rủi ro 67](#_Toc46677604)

[4.6. Các trường hợp kiểm thử 67](#_Toc46677605)

[4.6.1. Chức năng đăng nhập 67](#_Toc46677606)

[4.6.2. Chức năng tạo công văn 68](#_Toc46677607)

[4.6.3. Chức năng duyệt/hủy công văn 72](#_Toc46677608)

[4.6.4. Chức năng gửi công văn 73](#_Toc46677609)

[Tổng kết chương 74](#_Toc46677610)

[KẾT LUẬN 75](#_Toc46677611)

[TÀI LIỆU THAM KHẢO 77](#_Toc46677612)

[PHỤ LỤC 79](#_Toc46677613)

# MỞ ĐẦU

**1. Đặt vấn đề**

Quản lý, lưu trữ công văn, tài liệu trên hệ thống blockchain là một chủ để còn khá là mới ở Việt Nam, chúng được lưu trữ trên hệ thống phi tập chung, phân tán dễ dàng cho việc truy xuất dữ liệu một cách nhanh chóng, đảm bảo được tính toàn vẹn, tính sẵn sàng của dữ liệu. Quản lý công văn theo một quy trình công văn đi là một quy trình luôn phải đảm bảo tính chính xác, an toàn từ nơi gửi đến nơi nhận và không bị thay đổi nội dung cũng như việc đánh mất thông tin.

Trong làn sóng cách mạng công nghiệp 4.0, công nghệ Blockchain được xem là công nghệ "chìa khóa" cho chuyển đổi số và xây dựng nền tảng công nghệ tương lai. Một ứng cho phép người dùng có thể quản lý công văn, tài liệu trên hệ thống mạng blockchain một cách dễ dàng, thuận tiện sẽ giúp tổ chức, chính phủ an toàn cho việc sử dụng công nghệ này.

**2. Lý do chọn đề tài**

Công nghệ lưu trữ, quản lý theo mô hình tập trung truyền thống client-server có nhiều hạn chế mà công nghệ này mang lại. Đầu tiên vấn đề về chi phí khi phải xây dựng một hệ thống server, hoặc thuê một server mất rất nhiều chi phí vận hành, nhân công. Tính an toàn và bảo mật trên hệ thống cần rất nhiều thiết bị vật lý có thể đảm bảo hệ thống an toàn, cùng với đó có nhiều lỗ hổng mà hacker có thể xâm nhập. Khi mà hệ thống server với lượng truy cập người dùng lớn thì vấn đề not response server sẽ xảy ra hoặc có thể do những tấn công DOS từ bên ngoài làm cho việc truy xuất thông tin bị gián đoạn, dữ liệu không thể lấy được. Đặc biệt dữ liệu bị đổi làm sai lệch thông tin gốc khi áp dụng với tài liệu công văn nhà nước sẽ dẫn đến hậu quả nặng nề.

Để giải quyết vấn đề trên thì blockchain là sự lựa chọn hợp lý để bảo đảm về tính an toàn, bất biến cửa tài liệu, công văn cũng như tính sẵn sàng của dữ liệu khi mình cần truy xuất một cách nhanh chóng.

Vậy công nghệ Blockchain là gì ? Tiềm năng của công nghệ Blockchain lớn như thế nào ? Thông qua đề tài “Xây dựng hệ thống quản lý, lưu trữ công văn an toàn ứng dụng blockchain : Quy trình công văn gửi đi” này sẽ giúp em hiểu hơn về công nghệ Blockchain và việc ứng dụng nó vào hệ thống quản lý công văn thực tế sẽ thay đổi như thế nào.

**3. Mục tiêu**

Mục tiêu chính của đồ án này là cung cấp kiến thức cơ bản về công nghệ Blockchain và ưu điểm của công nghệ này so với phát triển ứng dụng Web truyền thống.

Bài viết này chứa yêu cầu hệ thống, trong đó cho thấy một số tiêu chí hệ thống phải được đáp ứng, sau đó hiển thị kiến trúc hệ thống và thiết kế hệ thống. Cuối cùng nhưng không kém phần quan trọng, cho thấy tất cả việc thực hiện cách xây dựng một bản trình bày cho đồ án này.

Trong đồ án này, em sẽ tạo phiên bản đầu tiên của ứng dụng quản lý công văn thực hiện tất cả giao dịch bằng nền tảng Blockchain.

**4. Giả thiết**

Tuy nhiên, việc sử dụng Blockchain trong lưu trữ công văn chưa được áp dụng trong thực tế, chính phủ vẫn chưa áp dụng công nghệ này trong thế giới thực.

Tóm lại, đồ án này giả định tất cả các thuộc tính trong hệ thống blockchain là hợp pháp và có thể trao đổi giao dịch khi chúng được triển khai.

**5. Kết cấu đề tài**

Nội dung đồ án được chia thành bốn chương:

Chương 1: Cơ sở lý thuyết

Chương 2: Yêu cầu hệ thống

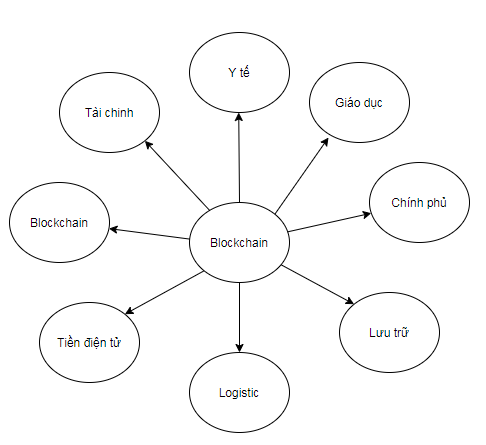
Chương 3: Thiết kế và xây dựng hệ thống

Chương 4: Kiểm thử và đánh giá

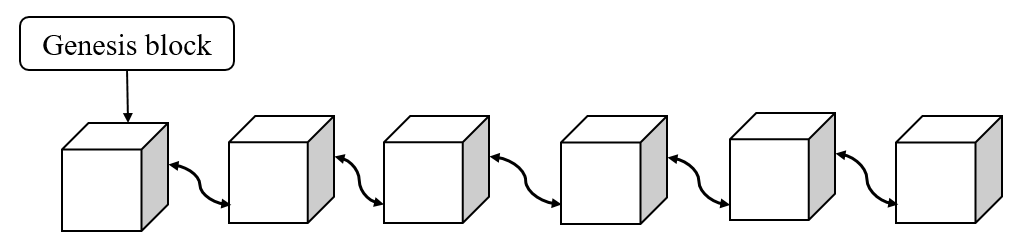
# Chương 1 CƠ SỞ LÝ THUYẾT

## 1.1. Giới thiệu công nghệ blockchain nền tảng

### 1.1.1. Công nghệ blockchain và các đặc trưng liên quan



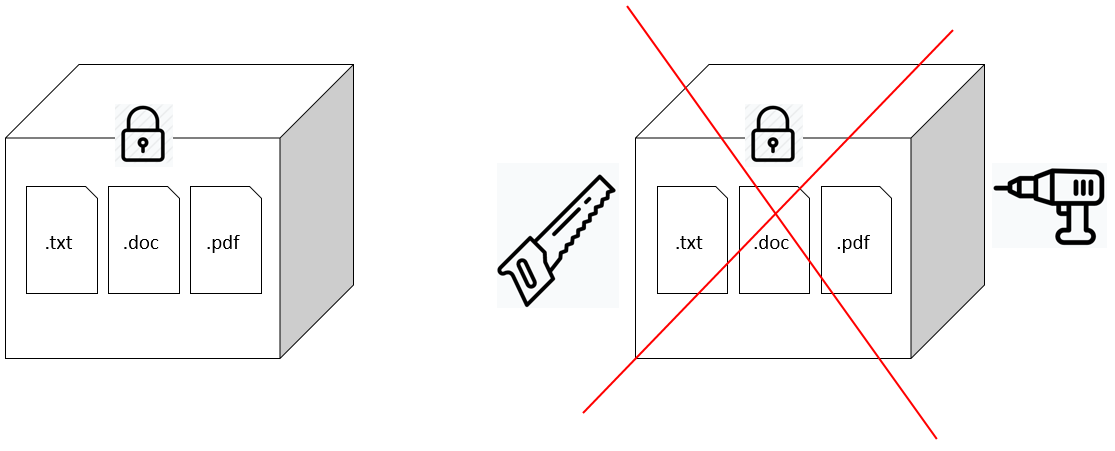
Hình 1. . Ứng dụng blockchain

**Blockchain** (chuỗi khối) là một cơ sở dữ liệu phân cấp lưu trữ thông tin trong các khối thông tin được liên kết với nhau bằng mã hóa và mở rộng theo thời gian. Mỗi khối thông tin đều chứa thông tin về thời gian khởi tạo và được liên kết tới khối trước đó, kèm một mã thời gian và dữ liệu giao dịch. Blockchain được thiết kế để chống lại việc thay đổi của dữ liệu. Một khi dữ liệu đã được mạng lưới chấp nhận thì sẽ không có cách nào thay đổi được nó.

Hình 1. . Mô hình chuỗi khối

**Block** cũng được định nghĩa như một cấu trúc dữ liệu, là một danh sách liên kết sử dụng hàm băm con trỏ thay vì các con trỏ thông thường. Hàm băm con trỏ này được sử dụng để trỏ tới khối trước đó.

Khối dữ liệu (block) là đơn vị lưu trữ cơ bản của Blockchain. Dữ liệu sẽ được đóng gói thành 1 khối. Sau đó khối này sẽ được khóa lại bằng 1 thuật toán mã hóa. Khi đã khóa xong thì dữ liệu sẽ không thể thay đổi được nữa.



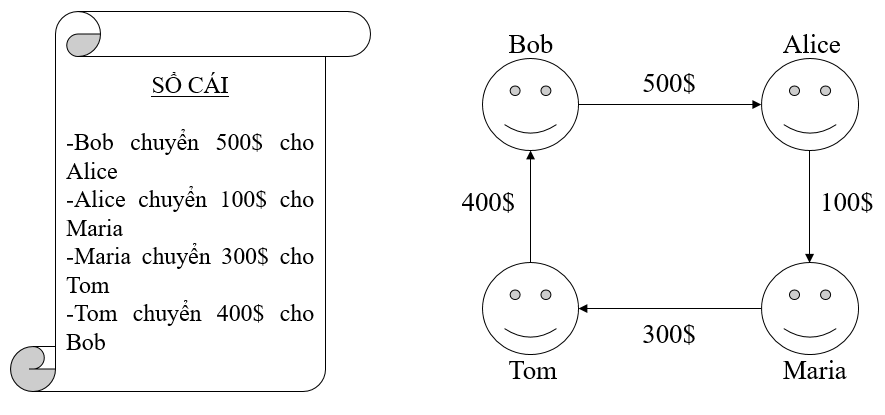
Hình 1. . Tình bất biến của blockchain

**Cấu trúc khối điển hình của block**

* *Version*: số phiên bản dùng để nâng cấp phần mềm và giao thức.
* *Previos Block hash*: hàm băm của khối trước nó.
* *Merkle root*: hàm băm của tất cả các giao dịch trong khối.
* *Timestamp*: dấu thời gian mà hàm băm được sinh ra.
* *Difficulty*: độ khó của hàm băm - tương ứng với con số 0 trên header của block.
* *Nonce*:Con số may mắn dùng để băm header sao cho kết quả phải nhỏ hơn hoặc bằng header hash trước đó thì mới được thêm vào.

**Sổ cái (Ledger)**

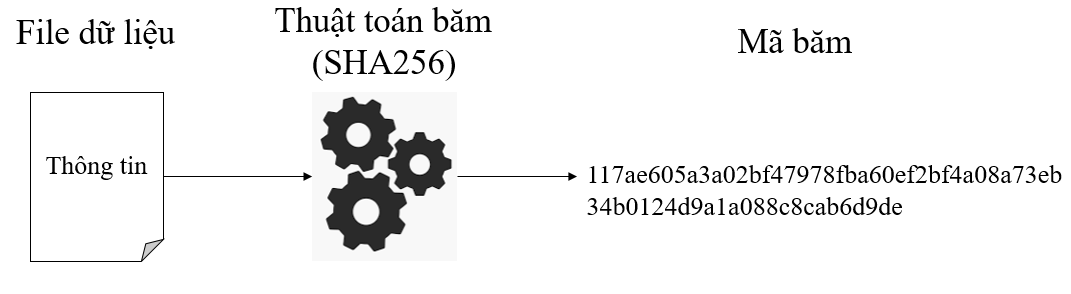
* Sổ cái là việc ghi lại các giao dịch giữa các cá nhân và các tổ chức với nhau.Đây là nhu cầu thiết yếu về chứng nhận quyền sở hữu cá nhân đối với các tài sản.



Hình 1. . Ledger ghi lại giao dịch trong mạng

**Hàm băm (Hash)**

* **Hàm băm** là một thuật toán hết sức quan trọng trong công nghệ Blockchain.
* Hàm băm thực chất là một thuật toán xáo trộn mã nhị phân của dữ liệu kỹ thuật số. Sao cho dữ liệu đầu vào dù dài hay ngắn thì khi băm xong cũng đều sẽ cho ra kết quả là một chuỗi có chiều dài cố định và không trùng lặp. Nghĩa là không bao giờ có chuyện hai dữ liệu đầu vào khác nhau mà lại cho ra hai mã băm giống nhau. Đây chính là cách mà người ta tạo ra mã số định danh duy nhất cho mỗi khối trong chuỗi, nó giống như số chứng minh thư của một khối vậy.

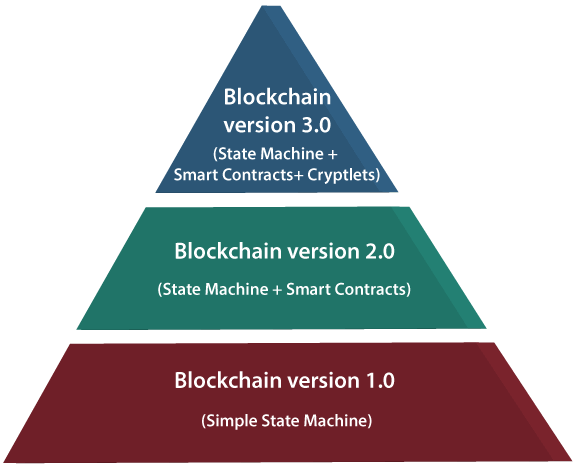


Hình 1. . Mã hóa thông tin

**Mã hóa khóa công khai (RSA)**

* Ngoài việc sử dụng hàm băm ra, công nghệ Blockchain còn sử dụng một công nghệ mã khóa khác để đảm bảo sự bí mật của dữ liệu đó là thuật toán mã hóa bất đối xứng hay còn gọi là cặp khóa công khai – bí mật RSA.

**Các phiên bản của Blockchain**



Hình 1. . Version blockchain

**Mạng private Blockchain và mạng public Blockchain**

Bảng 1. . Bảng so sánh hai mạng blockchain privte và public

|  |  |
| --- | --- |
| **Private** | **Public** |
| Là một Blockchain được phép | Là một Blockchain không được phép |
| Kiểm soát và hạn chế người tham gia | Bất cứ ai cũng có thể tham gia |
| Có một hoặc nhiều thực thể kiểm soát mạng dẫn đến phụ thuộc vào các bên thứ 3 để giao dịch | Không có một thực thể duy nhất kiểm soát mạng |
| Tất cả danh tính được biết nên tin cậy tuyệt đối | Không đáng tin cậy |
| Ví dụ: HF | Ví dụ: Bitcoin, Ethereum |

**Các tính chất của Blockchain**

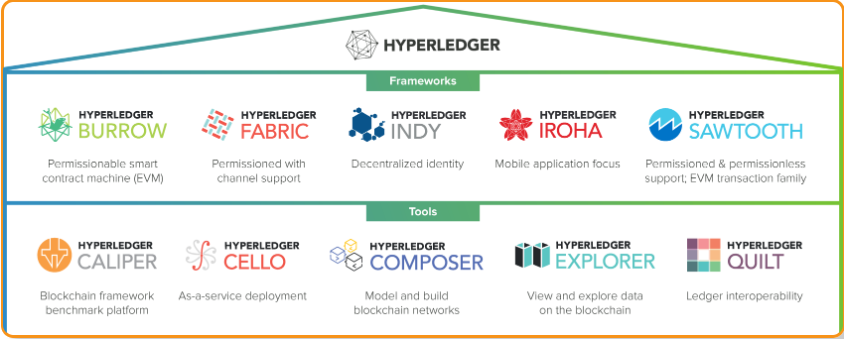
***Tính tin cậy:*** Khi nhắc đến blockchain, tính chất đầu tiên sẽ được nhắc tới là tính tin cậy. Giao dịch trong blockchain được ký số bởi thuật toán ECDSA – một thuật toán ký số mà khả năng giả mạo chữ ký là rất nhỏ, bởi yêu cầu tài nguyên tính toán lớn. Thêm nữa, việc sử dụng hàm băm mật mã trong cả quá trình từ lúc khởi tạo giao dịch tới lúc vào sổ là một yếu tố gia tăng thêm sự tin cây. Toàn bộ các giao dịch đều được tóm lược và lưu trữ trong header của block. Quá trình kiểm tra các block và giao dịch cũng diễn ra nhiều lần với nhiều lớp. Với tính chất này làm cho độ tin cậy của hệ thống được tăng lên.

***Tính bất biến:*** Dữ liệu giao dịch sẽ không thể thay đổi sau khi vào sổ xác nhận đủ lâu, giao dịch được lưu trữ trong block, các block lại được liên kết với nhau. Trong ID của một block có “dấu vết” của toàn bộ các giao dịch từ đầu, cộng thêm các yếu tố ngẫu nhiên. Do đó, việc làm lại sổ sẽ đòi hỏi tài nguyên tính toán rất lớn, vì liên quan đến các block trước và chi phối toàn bộ các block sau. Tính bất biến của dữ liệu là điểm mấu chốt khi lựa chọn công nghệ này.

***Tính bền vững:*** Hệ thống blockchain hình thành và phát triển đến khi đủ lớn thì sẽ trở nên không thể phá vỡ. Nguyên tắc đồng thuận phân tán khiến việc kiểm tra và xác thực ngày càng được gia tăng theo số nút mạng tham gia. Hơn thế, mạng ngang hàng mang lại khả năng tính toán cao, không có nút cổ chai nên việc tấn công DDoS rất khó xảy ra. Nếu có sự cố ở một nút thì các nút còn lại vẫn có thể hoạt động bình thường với dữ liệu đã được đồng bộ. Các dữ liệu này cũng không thể xóa, chỉ ghi và đọc dữ liệu. Với tính chất này thì dữ liệu của sản phẩm có thể được khôi phục dễ dàng.

***Tính sẵn sàng:*** Các nút mạng của blockchain được đặt ở nhiều nơi, nhiều khu vực địa lý khác nhau, nên đảm bảo tính sẵn sàng. Việc tham gia mạng blockchain không đòi hỏi sự cấp phép của một đơn vị quản lý tập trung, nên thường dẫn tới số nút mạng ngày càng gia tăng. Khi một máy bị lỗi và khôi phục lại, sẽ tự động kết nối và đồng bộ lại dữ liệu. Dữ liệu được lan truyền đi rộng khắp trên thế giới không phát sinh thêm chi phí địa lý, vận chuyển, … tiết kiệm thời gian và công sức. Tính chất này làm giảm khả năng sản phẩm bị thay đổi trong quá trình vận chuyển.

### 1.1.2. Công nghệ Blockchain Hyperledger Fabric



Hình 1. . Hyperledger Umbrella

**Hyperledger Fabric** là một nền tảng công nghệ sổ cái phân tán (DLT) cấp phép doanh nghiệp mã nguồn mở, được thiết kế để sử dụng trong bối cảnh doanh nghiệp, cung cấp một số khả năng khác biệt chính so với các nền tảng sổ cái phân tán hoặc blockchain phổ biến khác [1].

Một điểm khác biệt chính là Hyperledger được thành lập bởi **Linux Foundation**, nơi có lịch sử lâu dài và rất thành công trong việc nuôi dưỡng các dự án mã nguồn mở dưới sự quản trị mở của cộng đồng phát triển bền vững và hệ sinh thái thịnh vượng.

**Fabric** có kiến trúc **mô đun** và **cấu hình cao**, cho phép đổi mới, linh hoạt và tối ưu hóa cho nhiều trường hợp sử dụng công nghiệp bao gồm ngân hàng, tài chính, bảo hiểm, y tế, nguồn nhân lực, chuỗi cung ứng và thậm chí cả âm nhạc kỹ thuật số.

Fabric là nền tảng sổ cái phân tán đầu tiên hỗ trợ các hợp đồng thông minh được tạo ra bằng các ngôn ngữ lập trình cấp cao như Java, Go và Node.js, thay vì các DSL - ngôn ngữ chuyên biệt (ví dụ như ngôn ngữ dùng để viết smart contract trong nền tảng Ethereum là Solidity). Có nghĩa là các doanh nghiệp chỉ cần các nhà phát triển có kinh nghiệm về ngôn ngữ lập trình cấp cao là có thể phát triển được.

Nền tảng Fabric là nền tảng **được cho phép** (permissioned) có nghĩa là, không giống như mạng **permissionless/công khai**, những người tham gia được biết đến nhau, thay vì ẩn danh và do đó hoàn toàn không tin cậy. Điều này giúp đảm bảo hơn cho sự an toàn của dữ liệu.

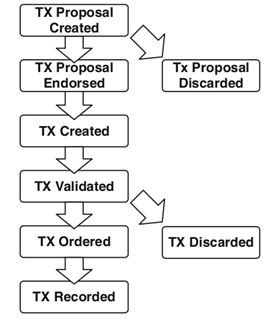
Một trong những yếu tố quan trọng nhất của nền tảng là sự hỗ trợ của nó đối với các giao thức đồng thuận **có thể tích hợp được**, cho phép nền tảng được tùy chỉnh hiệu quả hơn để phù hợp với các trường hợp sử dụng cụ thể và mô hình tin cậy.

**Fabric** có thể tận dụng các giao thức đồng thuận **không yêu cầu tiền điện tử riêng** để khuyến khích khai thác tốn kém hoặc thúc đẩy thực hiện hợp đồng thông minh.

Sự kết hợp của các tính năng thiết kế khác biệt này làm cho **Fabric** trở thành một trong những nền tảng hoạt động tốt hơn hiện nay cả về xử lý giao dịch và độ trễ xác nhận giao dịch, và nó cho phép bảo mật và riêng tư các giao dịch cũng như hợp đồng thông minh (smart contract).

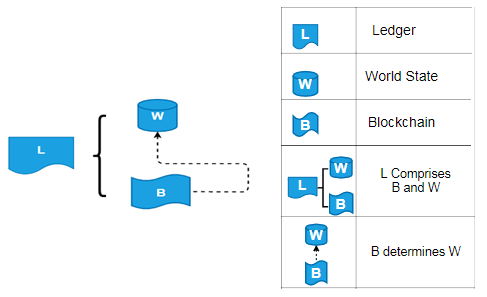
* ***Permissioned Blockchain:*** Hyperledger Fabric là một blockchain riêng tư và được cho phép. Thành viên của mạng cần phải ghi danh thông qua Membership Service Provider (MSP). Tất cả người tham gia của mạng đều phải biết được danh tính. Khóa công khai được sử dụng như chứng chỉ mật mã gắn liền với các tổ chức, thành phần mạng và người dùng cuối. Kiểm soát truy cập dữ liệu được áp dụng trên cấp độ mạng và kênh. Một sổ cái bao gồm 2 thành phần: World state và Transaction log (Blockchain), phần này sẽ được trình bày ở phần sau. Sổ cái cũng chứa các file cấu hình cũng như chính sách và danh sách kiểm soát truy cập.
* ***Chaincode (Hợp đồng thông minh):*** Hyperledger Fabric tận dụng công nghệ container để lưu trữ các hợp đồng thông minh được gọi là chaincode bao gồm logic ứng dụng của hệ thống. Chaincode có thể được được triển khai trong các ngôn ngữ lập trình như Go hoặc Java và được gọi thông qua một đề xuất giao dịch. Việc thực thi chaincode dựa trên World state của sổ cái.
* ***Nodes:*** có 3 kiểu nodes trong hệ thống Hyperledger Fabric: Client, peer và Oderer.
* **Client:** Một client hoạt động như một người dùng cuối. Nó kết nối với các peer để kết nối tới blockchain. Một Client node có thể tạo giao dịch và broadcast tin nhắn tới Oderers thông qua kênh giao tiếp.
* **Peer:** Một peer node nhận được trạng thái cập nhật theo thứ tự dưới dạng giao dịch từ oderers, commit giao dịch và duy trì trạng thái của sổ cái.
  + **Oderer**: Một Oderer node xác nhận các giao dịch dựa trên các chính sách chứng thực và yêu cầu các giao dịch thành một chuỗi trước khi **broadcast** chúng vào mạng.
* **Transactions:** Vòng đời của giao dịch Hyperledger Fabric bắt đầu từ một đề xuất được tạo bởi ứng dụng client và được gửi đến các peer cụ thể với tư cách là các **endorsers** (người chứng thực). Endorsers sẽ xác minh chữ ký của người khởi tạo và thực hiện các chaincode để chuẩn bị giao dịch. Kết quả của việc thực hiện là một tập hợp các cặp key-value được đọc từ chaincode và chúng được ghi vào chaincode. Đề xuất phản hồi (The proposal response) với chữ ký của người chứng thực được gửi lại cho khách hàng.

Client kết hợp các chứng thực thành payload của giao dịch trước khi broadcast giao dịch tới oderer. Oderer chịu trách nhiệm xác nhận giao dịch, sắp xếp các giao dịch thành khối và phân phối các khối cho tất cả các peer. Một khi các peer nhận được các giao dịch, chúng sẽ kiểm tra chính sách chứng thực để đảm bảo rằng các Endorsers chính xác là người đã ký kết quả và xác thực chữ ký đối với các chứng thực có trong payload giao dịch. Các peer đảm bảo tính toàn vẹn của dữ liệu giao dịch thông qua thông qua việc kiểm tra dữ liệu từ khi chaincode được thực thi cho đến lúc kiểm tra chứng thực. Nếu dữ liệu bị thay đổi thì sẽ được đánh dấu là không hợp lệ và gởi một tin nhắn thông báo lỗi đến client [4].

******

Hình 1. . Luồng giao dịch

* ***Consensus (Cơ chế đồng thuận):*** Dựa trên vòng đời của giao dịch, Consensus trong Hyperledger Fabric yêu cầu xác minh đầy đủ các giao dịch đạt được nếu giao dịch và kết quả thực hiện của chaincode tương ứng trong một khối đáp ứng các tiêu chí chính sách kiểm tra. Những kiểm tra này diễn ra trong vòng đời của một giao dịch và được chia thành 3 giai đoạn: Chứng thực (Endorserment), Đặt hàng (Ordering) và Xác nhận (Validation).
* Sự chứng thực được thúc đẩy bởi chính sách peer xác nhận giao dịch nhất định.
* Đặt hàng chấp nhận các giao dịch đã được chứng thực và yêu cầu các giao dịch thành một trình tự được cam kết vào sổ cái tương ứng.
* Các khối giao dịch được vận chuyển đến tất cả các peer trên kênh. Xác nhận kiểm tra tính chính xác của một tập hợp các giao dịch được đặt hàng trong một khối, xem sét chính sách chứng thực và kiểm tra phiên bản cho tính toàn vẹn dữ liệu.
* **Ledger (Sổ cái)**



Hình 1. . Cấu trúc của sổ cái Hyperledger Fabric

* **World state**: Mỗi chaincode có *world state* và *blockchain* riêng. Nó là một DsB giữ trạng thái hiện tại của trạng thái sổ cái. Các trạng thái này được thể hiện dưới dạng cặp *key-value*. Nó giữ thực tế của một đối tượng kinh doanh. Nó có thể được tạo, cập nhật và xóa. Khi ứng dụng gửi giao dịch và khi thực hiện giao dịch hợp lệ, trước tiên, nó sẽ commit ở world state và sau đó được cập nhật vào sổ cái. State cũng có một số phiên bản. Số phiên bản này thay đổi mỗi khi trạng thái được cập nhật. Khi một sổ cái được tạo ra, world state trống rỗng, vì bất kỳ giao dịch nào thể hiện sự thay đổi hợp lệ trong world state cũng được ghi lại trong blockchain.

Điều này có nghĩa là world state có thể được tạo bất cứ lúc nào từ blockchain. State chỉ được duy trì bởi Peers. Có thể hiểu đơn giản bằng ví dụ sau:

+ A muốn bán xe cho B và dữ liệu được lưu có thể là:

**{**

**“id”: “car1”** 🡪 Đây là id của xe, vị trí lưu trong sổ cái

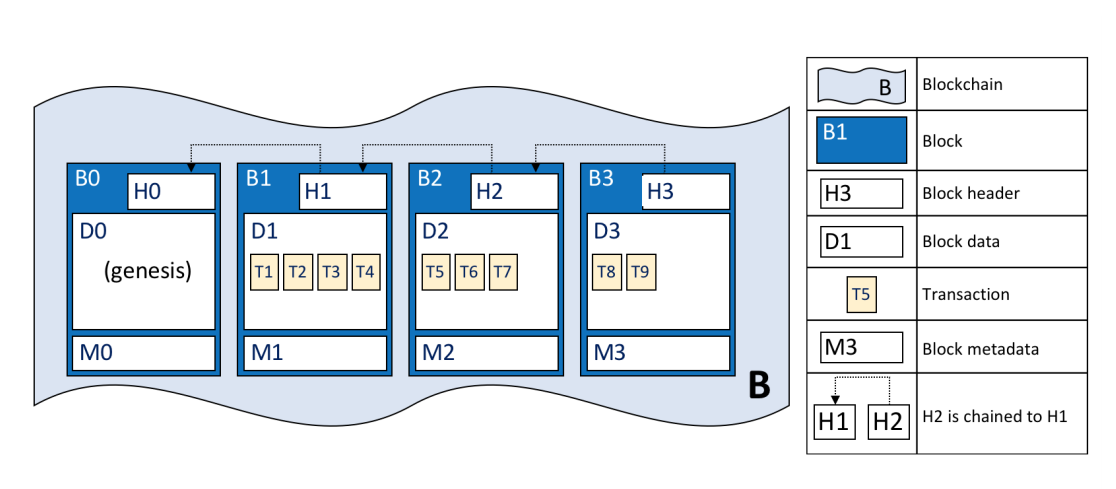
**“owner”: “A”** 🡪 Người sở hữu

**}**

+ Sau khi giao dịch xong thì B trở thành người sở hữu xe tương ứng với việc sổ cái sẽ thêm khối mới (do không thể thay đổi dữ liệu). Về lâu dài số lượng người sở hữu sẽ tăng lên dẫn đến câu truy vấn vào sổ cái sẽ mất nhiều thời gian. Do đó, World state dùng để lưu trạng thái dữ liệu tức là chỉ lưu thông tin giao dịch cuối cùng.

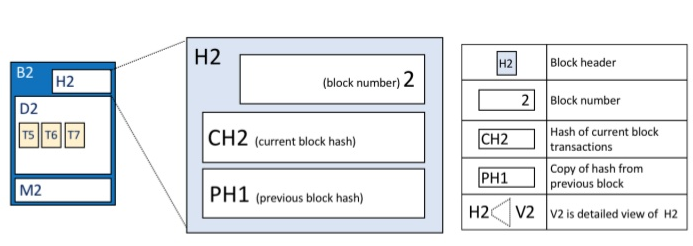
* **Blockchain**: Nhật ký giao dịch ghi lại tất cả các thay đổi đã dẫn đến world state hiện tại. Nó được cấu trúc như một bản ghi tuần tự của các khối được liên kết với nhau, trong đó mỗi khối chứa một chuỗi các giao dịch, mỗi giao dịch đại diện cho một truy vấn hoặc cập nhật lên world state.

Mỗi tiêu đề khối bao gồm một hàm băm của các giao dịch khối, cũng như một bản sao của hàm băm của tiêu đề khối khối trước đó. Blockchain luôn được triển khai dưới dạng một tệp, không giống như **World state** sử dụng cơ sở dữ liệu. Khối đầu tiên trong chuỗi là khối genesis. Mặc dù đây là một điểm khởi đầu của sổ cái. Nó không chứa bất kỳ chi tiết giao dịch nào. Thay vào đó, nó chứa các chi tiết giao dịch cấu hình chứa trạng thái ban đầu của kênh mạng.



Hình 1. . Cấu trúc blockchain trong Hyperledger Fabric

* **Mỗi khối bao gồm**



Hình 1.12. Cấu trúc khối trong Hyperledger Fabric.

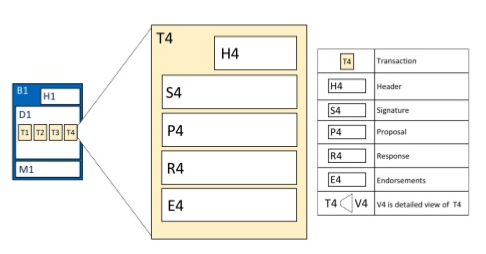
*- Header (H)*: Bao gồm 3 trường:

* Block number: Một số nguyên bắt đầu từ 0, khối genesis và tăng thêm 1 cho mỗi khối mới được nối thêm.
* Current Block Hash (CH): Hash của tất cả các giao dịch có trong một khối
* Previuos Hash (PH): Hash của khối trước.

*- Data (D)*: Bao gồm một danh sách các giao dịch được sắp xếp theo thứ tự.

*- Metadata (M)***:** Nó chứa thời gian khi khối được viết, chứng chỉ, khóa chung và chữ ký của người viết khối. Một committer khối cũng thêm một chỉ báo hợp lệ / không hợp lệ cho mỗi giao dịch. Thông tin này không được bao gồm trong hàm băm của khối.

* **Mỗi giao dịch trong một khối bao gồm**

**

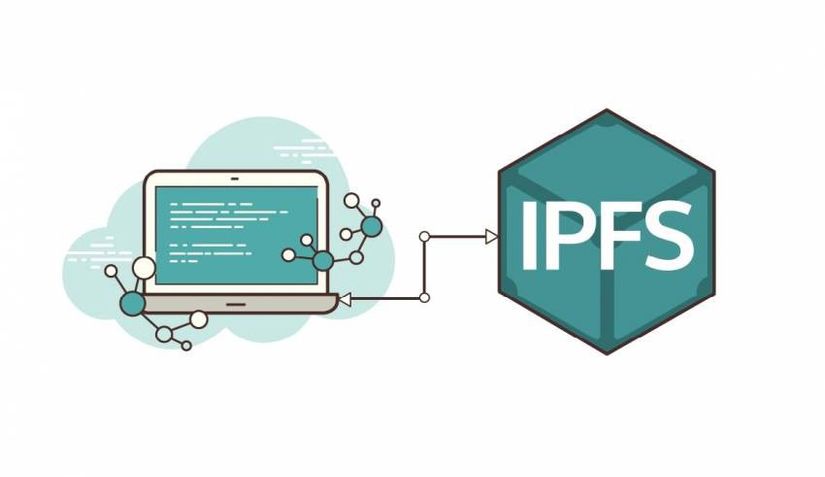
*Hình 1.13. Cấu trúc 1 giao dịch trong Hyperledger Fabric*

* ***Header (H)***: Chứa metadata cần thiết về giao dịch. Ví dụ: Tên của mã chuỗi có liên quan và phiên bản của nó.
* ***Signature (S)***: Chứa chữ ký mã hóa được tạo bởi ứng dụng khách. Nó được sử dụng để kiểm tra xem các chi tiết giao dịch có bị giả mạo hay không và yêu cầu khóa riêng của ứng dụng để tạo ra nó.
* ***Proposal (P)***: Mã hóa các tham số đầu vào được cung cấp bởi ứng dụng cho hợp đồng thông minh tạo ra cập nhật sổ cái được đề xuất. Khi hợp đồng thông minh chạy, đề xuất này cung cấp một tập hợp các tham số đầu vào, kết hợp với world state hiện tại, xác định world state mới.
* ***Response (R)***: Nó nắm bắt các giá trị trước và sau của world state dưới dạng cài đặt Đọc / Ghi (RW). Nó là đầu ra của hợp đồng thông minh.
* ***Endorsement (E)***: Đây là danh sách các phản hồi giao dịch đã ký từ mỗi tổ chức được yêu cầu đủ để đáp ứng chính sách chứng thực. Vì vậy, ở đây cho một đề xuất giao dịch, chúng ta có nhiều phản hồi chứng thực.

Với những khái niệm kể trên thì công nghệ này cung cấp những tính năng cần thiết đối với việc quản lý công văn bởi nếu người tham gia tham gia vào mạng, họ cần phải ghi danh và cung cấp thông tin chính xác thì mới được tạo giao dịch mới. Giúp hạn chế được dữ liệu cung cấp thiếu cho việc thực hiện quy trình gửi công văn. Bên cạnh đó thì cũng có nhược điểm là số lượng người tham gia ít do phải cung cấp thông tin giấy tờ liên quan mới có thể tham gia vào chuỗi.

Ở mức độ phát triển ứng dụng chúng ta sẽ chỉ phát triển chaincode để xử lý dữ liệu. Còn kiến trúc bên dưới thì chúng ta sẽ sử dụng Docker image mẫu do các kỹ sư của Fabric cung cấp. Phần docker này sẽ bao gồm 4 phần là Oderer, CouchDB, CA, Peer. Mô tả chi tiết xem ở phần cài đặt [1].

### 1.1.3. Công nghệ IPFS



Hình 1. . Công nghệ IPFS

**InterPlanetary File System (IPFS)** là một giao thức và mạng ngang hàng để lưu trữ và chia sẻ dữ liệu trong một hệ thống tệp phân tán. IPFS sử dụng địa chỉ nội dung để xác định duy nhất mỗi tệp trong không gian tên toàn cầu kết nối tất cả các thiết bị máy tính [6].

Bài toán đặt ra: Những hạn chế mà HTTP hiện tại đang gặp phải đó là phụ thuộc rất nhiều vào server (đại diện cho mô hình client-server). Và chính vì phụ thuộc vào server mà nó có những hạn chế như nếu server sập client sẽ không thể truy cập vào, hay server nắm quá nhiều data của người dùng dẫn đến những bê bối liên quan đến dữ liệu cá nhân.

Chính vì thế IPFS được sinh ra với tư tưởng về mô hình P2P thay vì Client-server. IPFS (viết tắt của InterPlanetary File System) là một giao thức phân phối mã nguồn của giao thức hypermedia hoạt động dựa trên nội dung và danh tính. Là sự kết hợp của rất nhiều công nghệ như **DHT**, **Git** và **BitTorrent.**

- Chúng ta có thể tổng hợp lại một số lợi ích sau:

* Tránh sự phụ thuộc vào máy chủ.
* Không còn mô hình tập trung.
* Giảm bớt chi phí (cho cả nhà phát triển ví dụ tiền server, ...).
* Cải tiến về tốc độ (cái này càng nhiều ng dùng thì càng nhanh giống như BitTorrent ý).

Đầu tiên chúng ta sẽ phải nhìn nhận lại mọi thứ đó là thay vì các kiểu đối tượng mà server lưu trữ như ảnh, text, videos, ... thì trên IPFS chúng sẽ được lưu dưới dạng mã hash hết (còn gọi là đối tượng IPFS). Ý tưởng đó là nếu trình duyệt của bạn muốn truy cập một trang nào đó trên IPFS thì chỉ cần đưa ra mã hash rồi mạng sẽ tìm máy có lưu trữ dữ liệu vs cùng mã hash và sau đó tải trang đó về từ máy tính đấy về cho bạn.

- Vì thế IPFS sẽ có 2 phần:

* Xác định tệp có địa chỉ nội dung (giá trị hash của tệp đó).
* Tìm nó (khi bạn có đoạn hash của file hay trang cần tải mạng sẽ tìm và connect tới máy tốt nhất để tải xuống cho bạn).

Đó là kết quả của một P2P overlay cung cấp khả năng định tuyến nhanh.

Sâu hơn thì các đối tượng IPFS tạo thành một cấu trúc dữ liệu được xác thực bằng mật mã được gọi là DAG Merkle và cấu trúc dữ liệu này có thể được sử dụng để mô hình hóa nhiều cấu trúc dữ liệu khác. Mình sẽ giới thiệu các đối tượng IPFS và Merkle DAG và đưa ra các ví dụ về các cấu trúc có thể được mô hình hóa bằng IPFS.

* **IPFS Objects**

- Một đối tượng IPFS là một cấu trúc dữ liệu với hai trường:

* Data: một đống dữ liệu nhị phân không cấu trúc có kích thước <256 kB.
* Links: một mảng các cấu trúc Links. Đây là các liên kết đến các đối tượng IPFS khác.

- Cấu trúc của Link có ba trường:

* Name: Tên của Link.
* Hash: hàm băm của đối tượng IPFS được link đến.
* Size: kích thước tích lũy của đối tượng IPFS được liên kết, bao gồm cả các liên kết sau của nó (Trường size chủ yếu được sử dụng để tối ưu hóa mạng P2P) [6].

## 1.2. Công nghệ sử dụng xây dựng hệ thống

### 1.2.1. Hyperledger Composer

**Hyperledger Composer** là một bộ công cụ và framework phát triển mở rộng, giúp phát triển các ứng dụng blockchain dễ dàng hơn. Mục tiêu chính của nó là tăng tốc thời gian phát triển hệ thống và giúp tích hợp các ứng dụng blockchain với các hệ thống hiện tại. Chúng ta có thể sử dụng Composer để nhanh chóng phát triển các trường hợp sử dụng và **triển khai giải pháp blockchain trong vài tuần thay vì vài tháng**. Nhờ sự thuận lợi trên nên nó được sử dụng trong khuôn khổ đề tài luận văn này [3].

**Hyperledger Composer bao gồm các thành phần sau:**

* **Execution Runtimes**: Có thể chạy trên 3 cơ chế khác nhau:
* Hyperledger Fabric State được lưu trên sổ cái phân tán.
* Web, được thực thi bên trong một trang web (playground). State được lưu ở local storage của trình duyệt
* Embedded, được thực thi trong tiến trình NodeJS dùng để kiểm tra logic. State được lưu trong bộ nhớ theo dạng key-value.
* **JavaScript SDK**: Là một tập hợp các API cho phép nhà phát triển tạo ứng dụng để quản lý và tương tác với hệ thống mạng đã cài đặt.
* **Command Line Interface**: Cho phép nhà phát triển và quản trị viên quản lý các khai báo mạng doanh nghiệp.
* **REST Server**: Công cụ tự động tạo các Open API (Swagger) REST API cho mạng doanh nghiệp (đọc phần kế tiếp).
* **LoopBack Connector**: là một tập hợp các khối xây dựng đơn giản hóa việc thực hiện các trình kết nối dành riêng cho nguồn dữ liệu như REST.
* **Playground Web User Interface**: Dùng để khai báo và kiểm tra mạng doanh nghiệp.
* **Yeoman code generator**: Dùng để tạo ra các skeleton project.
* **VSCode and Atom editor plugins:** Các plugin hỗ trợ việc phát triển ứng dụng.

### 1.2.2. Kiến trúc RESTful

**Web service** là tài nguyên phần mềm có thể xác định bằng địa chỉ URL/URI, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu. Một Web service được tạo nên bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng nhìn thấy và có thể truy cập đến những dịch vụ mà nó thực hiện, đồng thời có thể yêu cầu thông tin từ Web service khác. Giữa Web service và các nền tảng khác có thể giao tiếp với nhau bằng XML hoặc JSON. Đây là một công nghệ giúp cho ứng dụng người dùng cuối có thể chạy trên các nền tảng khác nhau nhưng vẫn chỉ cần một nơi để xử lý. Thích hợp cho việc mở rộng quy mô của ứng dụng.

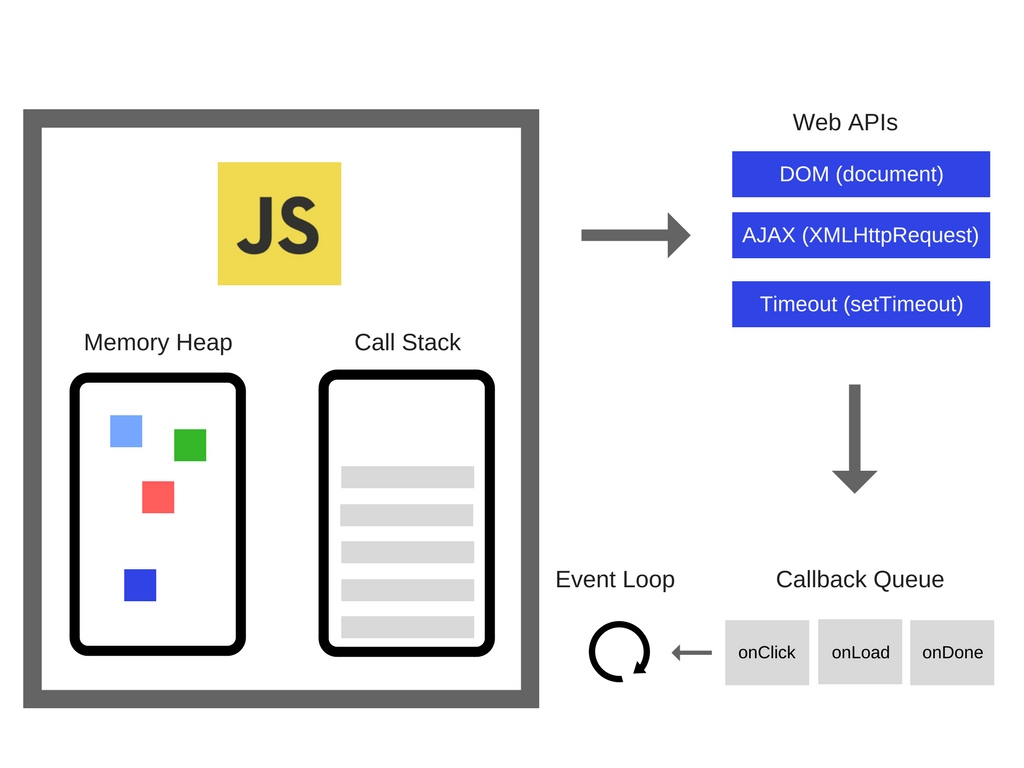
**REST** (Representational State Transfer) là một cách tiếp cận, một kiểu kiến trúc viết API nhưng không tuân theo một chuẩn hay một giao thức cụ thể nào. RESTful là một bộ quy tắc để tạo ra một ứng dụng Web Service, mà nó tuân thủ 4 nguyên tắc thiết kế cơ bản sau:

* Sử dụng các phương thức HTTP một cách rõ ràng: Thêm tài nguyên sử dụng phương thức POST, truy xuất tài nguyên sử dụng GET, cập nhật tài nguyên sử dụng PUT và xóa tài nguyên sử dụng DELETE.
* Phi trạng thái: Không lưu giữ thông tin của client, không quản lý phiên làm việc (Session).
* Hiển thị cấu trúc thư mục như các URls: để giúp cho người dùng dễ đoán đang trỏ đến và cung cấp tài nguyên gì.
* Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai: Dựa vào JSON hoặc XML, các phần mềm client có thể dễ dàng đọc và xử lý dữ liệu hiệu quả hơn.

### 1.2.2. NodeJS (running environment)

**Node.js** là một hệ thống phần mềm được thiết kế để viết các ứng dụng Internet có khả năng mở rộng, đặc biệt là máy chủ Web. Chương trình được viết bằng JavaScript, sử dụng kỹ thuật điều khiển theo sự kiện, nhập/xuất không đồng bộ để tối tiểu tổng chi phí và tối đại khả năng mở rộng. Node.js bao gồm có V8 JavaScript engine của Google, libUV, và vài thư viện khác.

**Node.js** cũng cung cấp một thư viện phong phú gồm nhiều module JavaScript khác nhau, giúp đơn giản hóa việc phát triển các ứng dụng Web bằng Node.js đến một mức độ lớn. Hình dưới đây cho thấy sự ra đời của JavaScript Engine trong phối cảnh cơ bản: Ngăn xếp cuộc gọi, Vòng lặp sự kiện và Hàng đợi nhiệm vụ



Hình 1. . Môi trường thời gian chạy NodeJS – Đơn giản

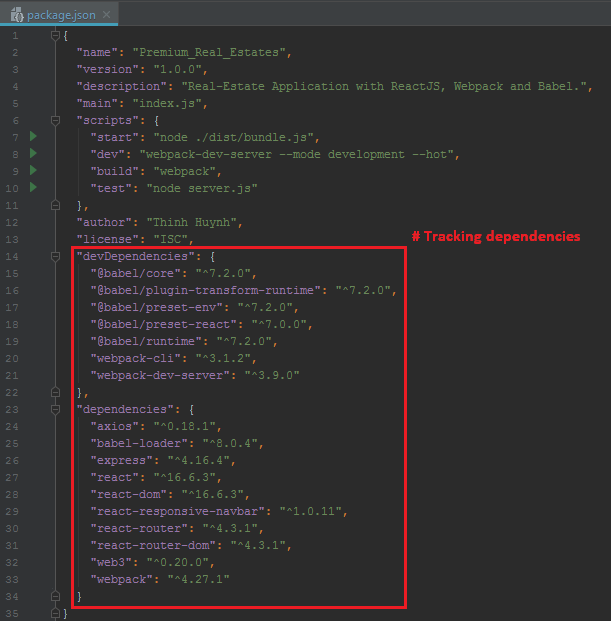
### 1.2.3. NPM

Khi bạn cài đặt Node.js, nó đi kèm với NPM, nó chuyển thành Trình quản lý gói Node của Node. NPM cuối cùng là một gói lưu trữ đăng ký trực tuyến để bạn tải xuống và là người quản lý các gói đó và các phụ thuộc của chúng trên máy của bạn để bạn có thể sử dụng chúng bất cứ khi nào cần trong dự án của bạn.

Sau khi bạn cài đặt Node.js nếu bạn muốn theo dõi các phụ thuộc gói dự án của mình, bạn phải tạo một tệp có tên là package.json, bạn có thể làm điều đó bằng cách chỉ cần thực hiện “npm init” và trả lời các câu hỏi được nhắc.

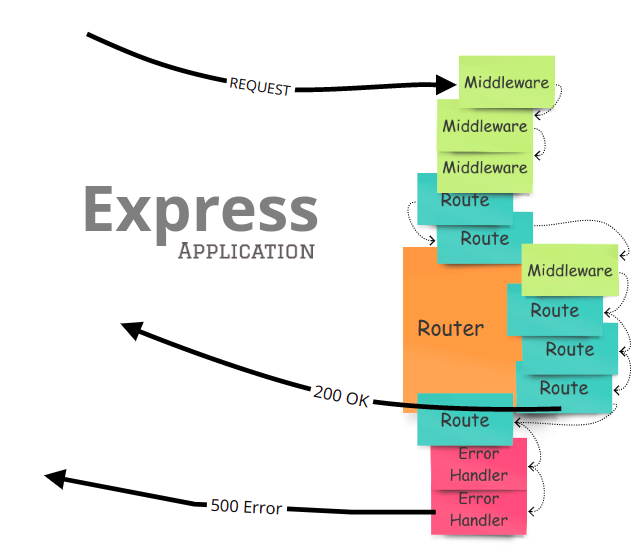
Có 2 cách chính để sử dụng NPM, đó là cài đặt các gói trên toàn cầu hoặc cục bộ. Sự khác biệt chính giữa các gói cục bộ và toàn cầu là:

* Các gói cục bộ được cài đặt trong thư mục nơi bạn chạy “npm install <tên gói>” và chúng được đặt trong thư mục node\_modules.
* Các gói toàn cầu được đặt ở một nơi duy nhất trong hệ thống của bạn (chính xác là nơi phụ thuộc vào thiết lập của bạn), bất kể bạn chạy “npm install -g <tên gói>” ở đâu.



*Hình 1. 13. Phụ thuộc gói cục bộ NodeJS được quản lý bởi NPM trong tệp package.json*

### 1.2.4. ExpressJS

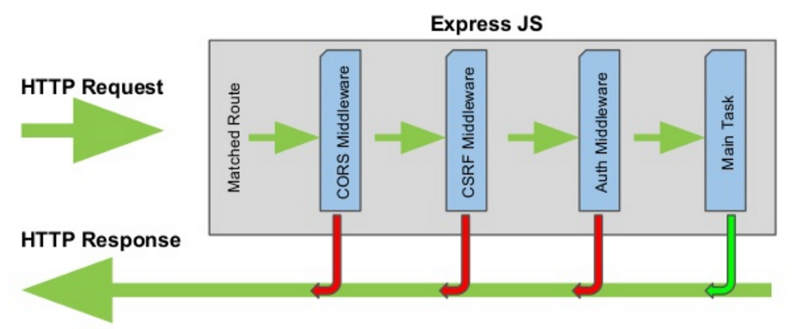


Hình 1. . Giới thiệu ExpressJS

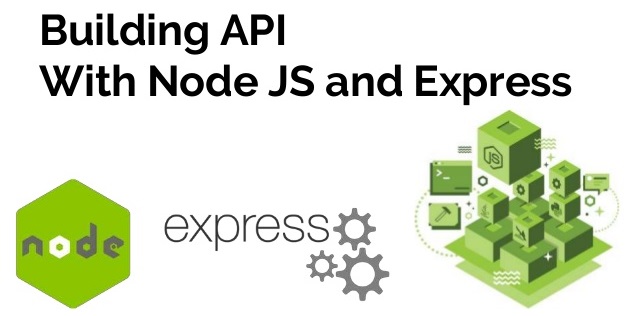
**Express** là khung Web Node phổ biến nhất và là thư viện cơ bản cho một số khung Web Node phổ biến khác. Nó cung cấp các cơ chế để:

* Viết trình xử lý cho các yêu cầu HTTP khác nhau tại các đường dẫn URL (tuyến đường) khác nhau.
* Tích hợp với các công cụ "view" để tạo phản hồi bằng cách chèn dữ liệu vào các mẫu.
* Đặt các cài đặt ứng dụng Web phổ biến như cổng sẽ sử dụng để kết nối và vị trí của các mẫu được sử dụng để hiển thị phản hồi.
* Thêm xử lý yêu cầu bổ sung "phần mềm trung gian" tại bất kỳ điểm nào trong đường ống xử lý yêu cầu.

Mặc dù bản thân Express khá tối giản, các nhà phát triển đã tạo ra các gói phần mềm trung gian tương thích để giải quyết hầu hết mọi vấn đề phát triển Web. Có thư viện để làm việc với cookie, session, đăng nhập người dùng, tham số URL, dữ liệu POST, tiêu đề bảo mật và nhiều hơn nữa. Bạn có thể tìm thấy danh sách các gói phần mềm trung gian được duy trì bởi nhóm Express tại Express Middleware (cùng với danh sách một số gói bên thứ 3 phổ biến).



Hình 1. . Luồng công việc ExpressJS



Hình 1. . REST API với ExpressJS

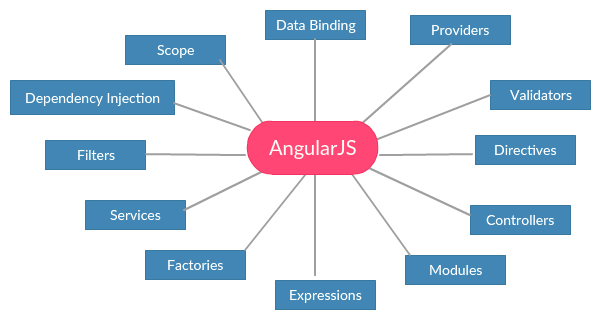
### 1.2.5. AngularJS

**Angular 6** là một UI framework được phát triển bởi Google cho phép chúng ta tạo ra một trang web dạng single page application. Các tính năng nổi bật:

* Two-way data binding: Đây là 1 trong những tính năng tuyệt với nhất trong Angular 2. Dữ liệu được binding một cách tự động và nhanh chóng, những thay đổi trong view sẽ được tự động cập nhật vào trong các component class.
* Typescript: Angular 6 hoàn toàn được viết bằng Typescript. Điều đó đồng nghĩa là nó hỗ trợ cho ES6 Modules.
* Expressive HTML: Angular 6 cho phép chúng ta dùng các cấu trúc lập trình như câu lệnh if, vòng lặp for,... để render và kiểm soát các trang HTML.
* Services: Services Angular là các đối tượng singleton hoặc các hàm được sử dụng để thực hiện các nhiệm vụ cụ thể. Nó nắm giữ một số logic và chức năng này có thể được gọi là bộ điều khiển, chỉ thị, bộ lọc. Có thể dùng nó để xử lý các http request đến RESTful server.

Những tính năng kể trên giúp cho người thực hiện có thể dễ dàng phát triển ứng dụng, giúp cho trải nghiệm của người sử dụng được tốt hơn.

AngularJS là một thư viện JavaScript rất mạnh. Nó được sử dụng trong các dự án Ứng dụng trang đơn (SPA - Single Page Application). Nó mở rộng HTML DOM với các thuộc tính bổ sung và làm cho nó phản ứng nhanh hơn với các hành động của người dùng. AngularJS là mã nguồn mở, hoàn toàn miễn phí và được sử dụng bởi hàng ngàn nhà phát triển trên toàn thế giới. Nó được cấp phép theo phiên bản giấy phép Apache 2.0.

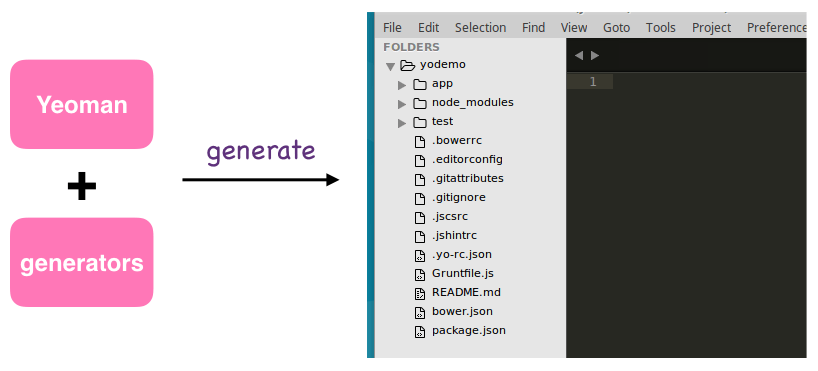


Hình 1. . Chức năng Core AngularJS

**Đặc điểm chung**

* AngularJS là một khung hiệu quả có thể tạo ra các ứng dụng Internet phong phú.
* AngularJS cung cấp cho các nhà phát triển một tùy chọn để viết các ứng dụng phía máy khách (Client Side) với kiến trúc Model View Controller (MVC) và Model View View Model (MVVM) .
* Các ứng dụng được viết bằng AngularJS tuân thủ nhiều trình duyệt.
* AngularJS tự động xử lý mã JavaScript phù hợp với từng trình duyệt.
* AngularJS là mã nguồn mở, hoàn toàn miễn phí.

### 1.2.6. Yeoman



Hình 1. . Dự án ứng dụng web mã tự sinh yeoman

**Yeoman** (yo) là 1 công cụ tạo khung, lên khung các app mới, hỗ trợ bạn generate các đoạn script, frameworks, ... bằng Nodejs, Angularjs, ...

**Yeoman** không chỉ là tools mà còn được sử dụng như là 1 luồng làm việc (workflow), tập hợp các "best practices – thực hành tốt nhất" giúp cho việc phát triển ứng dụng Nodejs một cách nhanh chóng và dễ dàng hơn. Dễ dàng tạo phần khung sườn cho những dự án mới, với các template tuỳ chỉnh được thông qua dòng lệnh. Yeoman còn tạo config cho Grunt và đưa vào các task Grunt cần cho việc build app.

* Yeoman hiện tại có các bộ generator cho Angular, Backbone, React, Polymer và hơn 1500+ projects khác.
* Hệ thống build process hỗ trợ minify script và css, tối ưu ảnh và HTML, biên dịch coffeescript, sass, less.
* Tất cả các script và js đều được tự động chạy qua JSHint để đảm bảo script được viết best-practices. Các bạn có thể tìm hiểu thêm về JSHint và CssHint.
* Tối ưu ảnh cực tốt (OptiPNG và JPEGTran).
* Package Management: có thể tìm kiếm, cài đặt packages mới thông quan terminal mà không cần mở trình duyệt.
* PhantomJS Unit Testing.

### 1.2.7. .NET Framework

.NET Framework là một nền tảng lập trình và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows được phát triển bởi Microsoft. Các chương trình được viết trên nền.NET Framework sẽ được triển khai trong môi trường phần mềm (ngược lại với môi trường phần cứng) được biết đến với tên Common Language Runtime (CLR). Môi trường phần mềm này là một máy ảo trong đó cung cấp các dịch vụ như an ninh phần mềm (security), quản lý bộ nhớ (memory management), và các xử lý lỗi ngoại lệ (exception handling).

.NET framework bao gồm tập các thư viện lập trình lớn, và những thư viện này hỗ trợ việc xây dựng các chương trình phần mềm như lập trình giao diện; truy cập, kết nối cơ sở dữ liệu; ứng dụng web; các giải thuật, cấu trúc dữ liệu; giao tiếp mạng... CLR cùng với bộ thư viện này là 2 thành phần chính của.NET framework.

.NET framework đơn giản hóa việc viết ứng dụng bằng cách cung cấp nhiều thành phần được thiết kế sẵn, người lập trình chỉ cần học cách sử dụng và tùy theo sự sáng tạo mà gắn kết các thành phần đó lại với nhau. Nhiều công cụ được tạo ra để hỗ trợ xây dựng ứng dụng.NET, và IDE (Integrated Developement Environment) được phát triển và hỗ trợ bởi chính Microsoft là Visual Studio.

### 1.2.8. Công nghệ cài đặt hệ thống Hyperledger

* Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
* Docker Engine: Version 17.03 or higher
* Docker-Compose: Version 1.8 or higher
* Node: 8.9
* npm: v5.x
* git: 2.9.x or higher
* Python: 2.7.x

## 1.3. Công cụ hỗ trợ

### 1.3.1. VMware Workstation

**VMware Workstation** là một phần mềm tạo máy ảo nổi tiếng, nó có thể cấu hình các tham số cho máy ảo và đặc biệt có thể clone các máy đã tạo. Rất thuận lợi cho việc cài đặt hệ thống.

### 1.3.2. Postman

**Postman** là một công cụ cho phép chúng ta làm việc với API, nhất là REST, hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, PATCH, DELETE, ...), cho phép lưu lại lịch sử các lần request, rất tiện cho việc sử dụng lại khi cần.

### 1.3.3. Visual Studio 2019

**Microsoft Visual Studio** là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

**Visual Studio 2019** là IDE để xây dựng các ứng dụng web application một cách dễ dàng và nhành chóng, cùng với mỗi trường fix bug được hỗ trợ rất mạnh mẽ cho các developer.

## Tổng luận chương

Bằng những cơ sở trên, em đã đưa ra được giải pháp và lựa chọn công nghệ cho hệ thống.

1. Xây dựng ứng dụng front-end AngularJS để call API tới hệ thống Composer Rest Sever thực hiện cho việc quản trị công văn và người dùng theo quy trình công văn gửi đi.
2. Xây dựng hệ thống back-end với công cụ Hyperledger Composer sử dụng NodeJS viết chaincode trên server và triển khai trên Hyperledger Fabric.
3. Sử dụng Web MVC làm website cho người dùng với múc đích mở rộng
4. Xử lý việc lưu trữ công văn trên hệ thống IPFS, upload và download công văn trên hệ thống IPFS.

# Chương 2 YÊU CẦU HỆ THỐNG

## 2.1. Mô tả hệ thống

### 2.1.1. Các tính năng chính của ứng dụng

**Các tính nâng chính:**

* Đăng nhập (để phân chia về vai trò của người tham gia).
* Xem thông tin cá nhân (biết được đã đăng nhập hay chưa).
* Soạn thảo công văn từ văn bản giấy (văn thư là người soạn công văn sau khi nhận được văn bản chính thức người quản lý) bao gồm: (thêm, sửa, xóa, tìm kiếm).
* Kiểm duyệt công văn để trùng khớp với thông tin công văn của văn bả giấy.
* Xác nhận và gửi công văn đến các phòng ban, tổ chức được nhận.
* Xem thông tin công văn.

### 2.1.2. Đặc điểm người sử dụng

* **Người vận hành:** Người vận hành sử dụng các chức năng của Composer-cli hoặc giao diện explorer để quản lý các thông tin cho người tham gia.
* **Người tham gia (lãnh đạo, văn thư, user (khoa, phòng ban)):** liên hệ với người vận hành để cung cấp thông tin và nhận ID card tham gia vào mạng để thực hiện các task vụ của mình.
* **Người tham gia (user (tất cả mọi người truy cập trang chủ)):** Người muốn xem thông tin công văn, chỉ thị của cấp trên đưa xuống.

### 2.1.3. Môi trường vận hành

* Server:
* Hệ điều hành Ubuntu 16.04.
* Node: phiên bản 8.17.
* Ngoài ra còn có các điều kiện ở phần phụ lục cài đặt.
* Ram tối thiểu 4GB.
* Máy người sử dụng:
* Trình duyệt web Chrome, Firefox…
* Có kết nối Internet.

### 2.1.4. Các ràng buộc về thực thi và thiết kế

* Vì cơ sở dữ liệu là một loại đặc biệt, HF Blockchain, nên yêu cầu chính của hệ thống phải đạt được các tiêu chí sau:
* Hệ thống phải được kết nối với mạng HF, trong tài liệu đồ án này.
* Hệ thống phải được thêm vào các phụ thuộc để tương tác với mạng HF, trong tài liệu này là Rest Server Composer.
* Giao diện người dùng phải rõ ràng, đơn giản, trực quan và nhanh nhạy để giúp người dùng hoàn thành nhiệm vụ với ít nỗ lực và thời gian hơn, do đó Angular được áp dụng cho Client Side trong đồ án này.
* Hiệu suất (Performance) của phía khách hàng phải thích ứng với nhược điểm của SPA, do đó, trong tài liệu đồ án án này, kết xuất phía máy chủ cho máy khách được áp dụng.
* API phải đơn giản để lưu trữ các hợp đồng Hyperledger được biên dịch dưới định dạng JSON, do đó ExpressJS được áp dụng cho Phía máy chủ trong bài luận văn này.
* Đảm bảo dữ liệu được lưu trữ trên blockchain qua đó đảm bảo được sự an toàn của dữ liệu.
* Đảm bảo tính dễ sử dụng: Thân thiện với người dùng không chuyên, dễ dàng làm quen và sử dụng ngay lần đầu tiếp xúc. Chỉ cần thực hiện một số lượng ít các thao tác để hoàn thành một chức năng nhất định.
* Đảm bảo tính bảo mật: Cần cung cấp thông tin hợp lệ mới có thể đăng nhập và sử dụng hệ thống.
* Đảm bảo các chất lượng của phần mềm: Dễ sử dụng, module hóa, dễ bảo trì, mở rộng.
* Kỹ thuật và công nghệ: Sử dụng nền tảng blockchain Hyperledger Fabric, tool hỗ trợ Hyperledger Composer và Angular 6.

### 2.1.5. Các yêu cầu giao tiếp bên ngoài

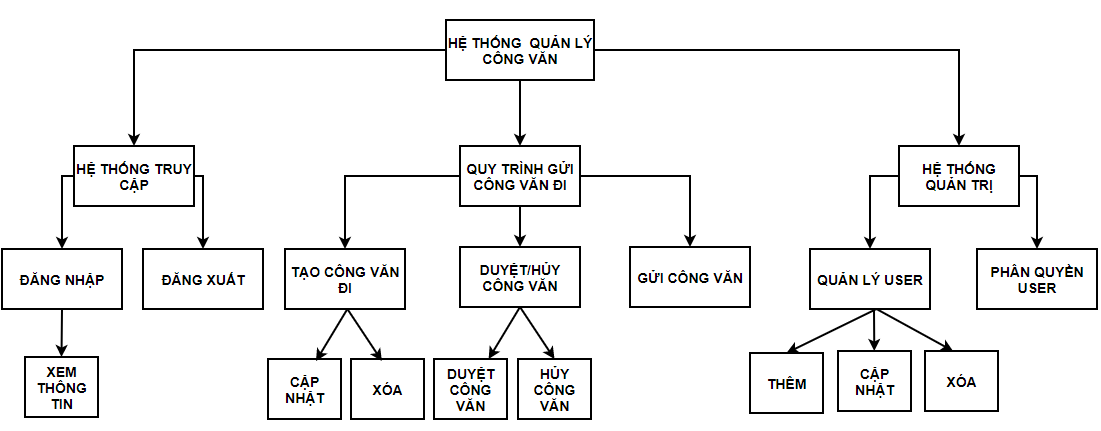
**Giao diện người dùng:**

* Màu nền sáng và phối màu hài hòa giữa các phần tử trên giao diện.
* Font chữ đơn giản, rõ ràng.
* Sử dụng ngôn ngữ dễ hiểu phù hợp người sử dụng.
* Không sử dụng nhiều hiệu ứng giữa các thao tác sử dụng.
* Hiển thị tốt và dễ dàng thao tác trên các thiết bị di động có màn hình nhỏ.

**Giao tiếp phần mềm:**

* Ứng dụng hoạt động trên trình duyệt Google Chrome.
* Môi trường vận hành: Web Application.
* Quá trình truyền dữ liệu giữa Backend và Frontend của ứng dụng sử dụng định dạng Json với phương thức xác thực JwtToken.

## 2.2. Phân rã chức năng



Hình 2. . Biểu đồ phân rã chức năng

## 2.2. Các yêu cầu chức năng

### 2.2.1. Đăng nhập

Bảng 2. . Yêu cầu chức năng đăng nhập

|  |  |
| --- | --- |
| Tên yêu cầu | Đăng nhập |
| Đối tượng sử dụng | Người tham gia vào quy trình gửi công văn đi |
| Điều kiện | Đã được cũng cấp thông tin name card, password , đơn vị. |
| Các thao tác xử lý | B1: Người dùng chọn nút “Login/Logout’’ phía trên.  B2: Nhập tên card, mật khẩu, đơn vị của mình.  B3: Nhấn nút “Đăng Nhập”. |
| Kết quả | Đăng nhập thành công, hoặc thất bại. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.2. Tạo công văn đi

Bảng 2. 2. Chức năng tạo công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Tạo công văn điện tử |
| Đối tượng sử dụng | Văn thư được tham gia vào trong mạng. |
| Điều kiện | Văn bản nội bộ thì đăng nhập thành công, văn bản công khai không cần đăng nhập. |
| Các thao tác xử lý | B1: Văn thư đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Nhập tên card, mật khẩu, đơn vị của mình.  B4: Đăng nhập thành công chuyển sang danh sách công văn  B5: Văn thư chọn nút thêm công văn, giao diện chuyển sang trang soạn thảo công văn.  B6: Soạn công văn, văn thư ấn nút tạo công văn. |
| Kết quả | Tạo thành công, hoặc thất bại. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.3. Cập nhật công văn đi

Bảng 2. 3. Chức năng cập nhật công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Cập nhật công văn |
| Đối tượng sử dụng | Văn thư được tham gia vào trong mạng. |
| Điều kiện | Văn bản nội bộ thì đăng nhập thành công, văn bản công khai không cần đăng nhập. |
| Các thao tác xử lý | B1: Văn thư đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Căn thư chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chuyển sang danh sách công văn  B5: Văn thư chọn nút sửa công văn, giao diện chuyển sang trang cập nhật công văn.  B6: Cập nhật công văn, văn thư ấn nút tạo công văn. |
| Kết quả | Cập nhật thành công, hoặc thất bại. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.4. Xóa công văn đi

Bảng 2. 4. Chức năng xóa công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Xóa công văn |
| Đối tượng sử dụng | Văn thư được tham gia vào trong mạng. |
| Điều kiện | Văn bản nội bộ thì đăng nhập thành công, văn bản công khai không cần đăng nhập. |
| Các thao tác xử lý | B1: Văn thư đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Văn thư chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chuyển sang danh sách công văn  B5: Văn thư chọn nút xóa công văn, có thông báo xác nhận.  B6: Chọn xóa công văn. |
| Kết quả | Xóa thành công, hoặc thất bại. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.5. Duyệt công văn điện tử

Bảng 2. 5. Chức năng duyệt công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Duyệt hoặc hủy công văn đi. |
| Đối tượng sử dụng | Lãnh đạo được tham gia vào trong mạng. |
| Điều kiện | Đăng nhập thành công. |
| Các thao tác xử lý | B1: Lãnh đạo đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Lãnh đạo chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chuyển sang danh sách công văn đi.  B5: Chọn nút công văn cần duyệt, danh sách công văn cần duyệt sẽ hiện thị.  B6: Xác nhận duyệt , hoặc hủy công văn. |
| Kết quả | Duyệt thành công, hoặc hủy công văn. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.6. Gửi công văn đi

Bảng 2. 6. Chức năng gửi công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Gửi công văn đi. |
| Đối tượng sử dụng | Văn thư được tham gia vào trong mạng. |
| Điều kiện | Đăng nhập thành công. |
| Các thao tác xử lý | B1: Văn thư đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chuyển sang danh sách công văn đi.  B5: Chọn nút công văn cần gửi, danh sách công văn cần gửi sẽ hiện thị.  B6: Xác nhận gửi. |
| Kết quả | Gửi thành công. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.7. Xem thông công văn đi

Bảng 2. 7. Chức năng xem thông tin công văn đi

|  |  |
| --- | --- |
| Tên yêu cầu | Xem thông tin công văn đi. |
| Đối tượng sử dụng | User (khoa, phòng ban) được tham gia vào mạnng.  Tất cả mọi người truy cập vào trang chủ |
| Điều kiện | Đăng nhập thành công hoặc không cần đăng nhập. |
| Các thao tác xử lý | B1: User đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chuyển sang thông tin công văn đi. |
| Kết quả | Nhìn thấy thông tin công văn đi . |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.8. Xem thông tin cá nhân

Bảng 2. . Chức năng xem thông tin cá nhân

|  |  |
| --- | --- |
| Tên yêu cầu | Xem thông tin cá nhân. |
| Đối tượng sử dụng | Participant được tham gia vào mạnng. |
| Điều kiện | Đăng nhập thành công. |
| Các thao tác xử lý | B1: Participant đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công chọn thông tin property |
| Kết quả | Nhìn thấy thông tin cá nhân. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

### 2.2.9. Xem thông tin các lần giao dịch

Bảng 2. . Chức năng xem thông tin các lần giao dịch

|  |  |
| --- | --- |
| Tên yêu cầu | Xem thông tin tất cả thông tin giao dịch trên mạng. |
| Đối tượng sử dụng | Admin được tham gia vào mạnng. |
| Điều kiện | Đăng nhập thành công. |
| Các thao tác xử lý | B1: Admin đăng nhập vào ứng dụng, yêu cầu quản trị viên import card.  B2: Chọn đăng nhập để thực hiện login vào mạng  B3: Chọn tên card, mật khẩu, đơn vị của mình  B4: Đăng nhập thành công có thể chọn tất cả thông tin. |
| Kết quả | Xem tất cá thông tin. |
| Ghi chú | Sau khi hoàn thành quy trình nhập thông tin cần đăng xuất để tải về card mới cho lần sử dụng kế tiếp. |

## 2.4. Các yêu cầu phi chức năng

### 2.4.1. Yêu cầu thực thi

Mỗi thao tác chức năng không tốn quá lâu để hoàn thành. Có khả năng cập nhật trạng thái sản phẩm Real-time với websocket mỗi khi quản trị viên thêm mới sản phẩm hoặc cập nhật thông tin giai đoạn.

### 2.4.2. Yêu cầu an toàn

Đảm bảo an toàn CSDL trước các nguy hại trên Internet, các hành vi tấn công phá hoại của hacker.

### 2.4.3. Yêu cầu bảo mật

* Yêu cầu người dùng có thông tin hợp lệ mới có thể truy cập và thực hiện thao tác thêm thông tin trong ứng dụng.
* Trao đổi dữ liệu giữa backend và fronend cần phải chứng thực thông qua JwtToken.
* Người dùng được phân vai trò và các quyền truy cập để sử dụng tài nguyên hệ thống.

### 2.4.4 Các đặc điểm chất lượng phần mềm

* **Tính tương thích**

Ứng dụng có thể hoạt động trên các kích thước màn hình cũng như các nền tảng hệ điều hành, phiên bản trình duyệt khác nhau.

* **Tính tin cậy**
* Chỉ người dùng có thông tin hợp lệ được cung cấp bởi quản trị viên mới có thể truy cập và thêm dữ liệu thông qua ứng dụng.
* Tất cả thao tác liên quan tới dữ liệu đều được ghi nhận lại.
* Dữ liệu không thể bị thay đổi từ bên ngoài.
* **Tính linh hoạt**

Ứng dụng cần đảo bảo hoạt động bình thường và phản hồi nhanh khi thao tác với lượng dữ liệu lớn.

* **Tính dễ bảo trì**
* Đảm bảo cấu trúc phần mềm và lập trình hướng đối tượng.
* Ứng dụng cần được xây dựng theo hướng phân hóa thành các component riêng biệt và theo các mức trừu tượng nhất định, tránh trùng lặp code để dễ dàng thay đổi, bảo trì về sau.
* Mô hình ứng dụng được chia thành hai phần với backend hoạt động như một webservice là một cơ sở tốt để phát triển thêm các ứng dụng với nền tảng khác như Android Native, iOS Native khi có nhu cầu.
* **Tính có thể kiểm thử**

Có thể thực hiện kiểm thử chấp nhận thông qua việc kiểm thử chức năng hệ thống trên các trình duyệt.

* **Tính theo dõi được**

Tất cả thao tác liên quan tới dữ liệu đều được ghi nhận lại.

## Tổng kết chương

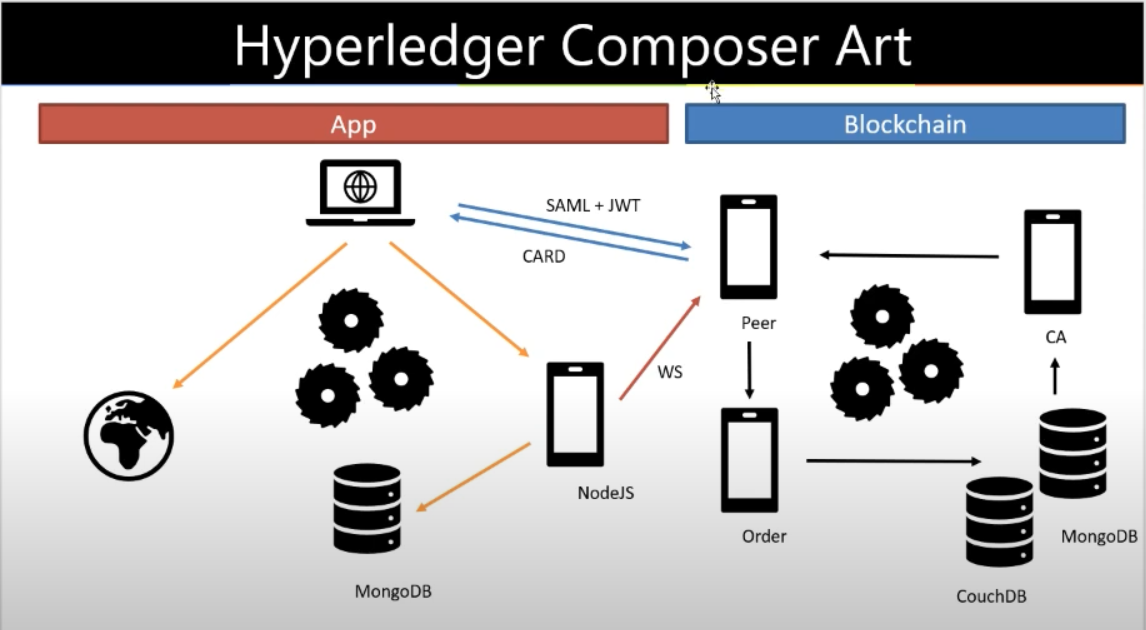
Ở chương 2 em đã đưa ra những yêu cầu, tiếp cận về bài toán quản lý và lưu trữ công văn trệ hệ thoogns blockchain Hyperledger Fabric và từ đó đưa ra những phân tích cho hệ thống của mình. Bắt đầu từ mô hình tổng quan của hệ thống, các chức năng của hệ thống để phục vụ người dùng. Tiếp đến là đưa ra được các mô hình phân rã chức năng,thông tin nghiệp vụ phản ánh tương tác của người dùng với hệ thống.

# Chương 3: THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

## 3.1. Kiến trúc hệ thống

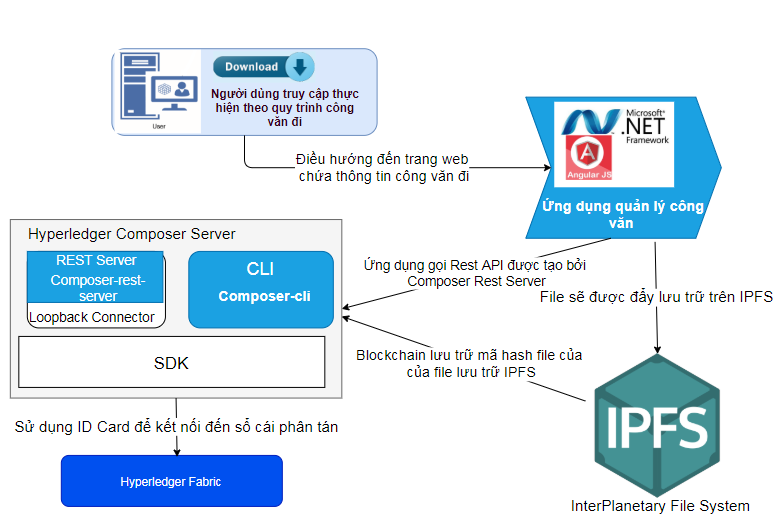
An external file that holds a picture, illustration, etc.
Object name is sensors-19-02228-g008.jpg

Hình 3. . Kiến trúc tổng quan hệ thống Hyperledger

****

Hình 3. . Mô hình hoạt động tổng quan hệ thống Hyperledger

### 3.1.1. Thiết kế kiến trúc

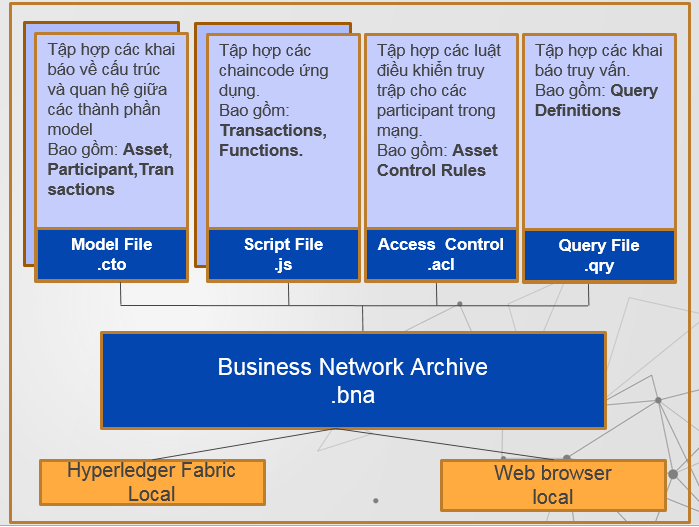


Hình 3. . Mô hình kiến trúc hệ thống quản lý lưu trữ công văn

Dựa vào mô hình này chúng ta có thể hình dung ra được hệ thống gồm 4 thành phần chính:

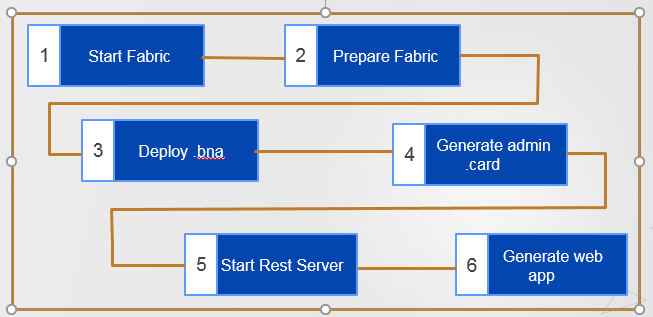
* **Client**: Web API .Net Framework, AngularJS framework ở phía front-end sẽ đảm nhận vai trò hiển thị dữ liệu cũng như gửi dữ liệu giai đoạn lên server.
* **Composer-rest-server:** Đảm nhiệm vai trò là một máy chủ REST, xử lý dữ liệu đầu vào là các phương thức giao tiếp HTTP và tương tác với mạng fabric có sẵn. Bên cạnh nó còn có **composer-cli** dùng để tạo ID card cho người tham gia và cài đặt chaincode lên hệ thống mạng Fabric.
* **Hyperledger Fabric**: Mạng fabric mẫu có sẵn đã được cài đặt.
* **InterPlanetary File System**: Hệ thống tập tin phan tán ngang hàng(P2P). Cho phép lưu trữ nội dung file trang hệ thống phân tán.

**Hyperledger Composer** cung cấp bộ công cụ để viết hợp đồng thông minh (chaincode) [3]:



Hình 3. . Các thành phần định business network

* **Model file** dùng để khai báo domain của business network và các mô hình dữ liệu muốn lưu trữ.
* **Script file** là các chaincode của ứng dụng xử lý các mô hình được khai báo sau đó lưu vào sổ cái.
* **Access control file** dùng để phân quyền người tham gia (participant).
* **Query file** dùng để lọc dữ liệu từ sổ cái.
* Ngoài Script file được viết bằng Javascript (ES6) thì 3 file còn lại được viết theo cấu trúc của tài liệu Hyperledger Composer.
* Các bước để bắt đầu phát triển một ứng dụng



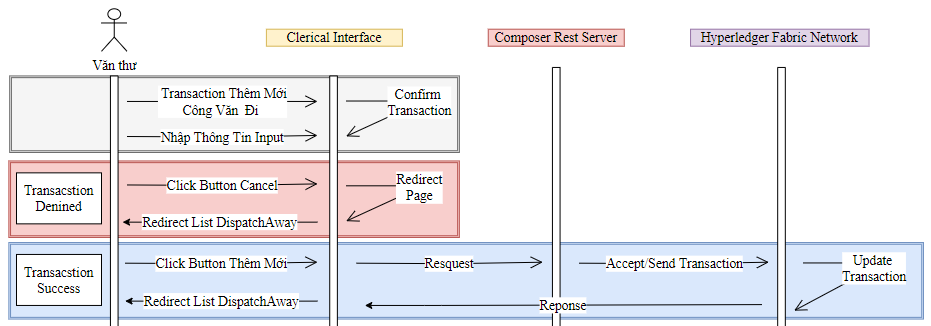
Hình 3. . Các bước để bắt đầu phát triển một ứng dụng

1. Khởi động các thành phần tham gia mạng như là Peers, Oderers, CA, CouchDB.
2. Chuẩn bị các file cấu hình cũng như các chứng chỉ mẫu.
3. Deploy file bna gồm các khai báo mạng lên mạng Fabric.
4. Tạo admin ID card (bao gồm thông tin kết nối và chứng chỉ) để tương tác với mạng.
5. Khởi chạy REST server: Các mô hình dữ liệu đã tạo sẽ tự động có thêm các phương thức HTTP (GET, POST, PUT, DELETE, ...).
6. Tạo ứng dụng web .net với angularjs để làm ứng dụng phía front-end, có thể phát triển thêm các tính năng không cần thiết với mạng blockchain trong tương lai.

* Đối với REST server thì ứng dụng cần có 3 chế độ:
* Một để cho người xem thông tin công văn dành cho tất cả, không cần đăng nhập - Chỉ có thể xem.
* Một để cho người tham gia vào quá trình xử lý quy trình công văn đi- Multiuser mode (có chứng thực).
* Có thể có thêm chế độ dành cho người quản trị để họ có thể tạo user (người tham gia), truy xuất thông tin giao dịch quy trình gửi công văn đi.

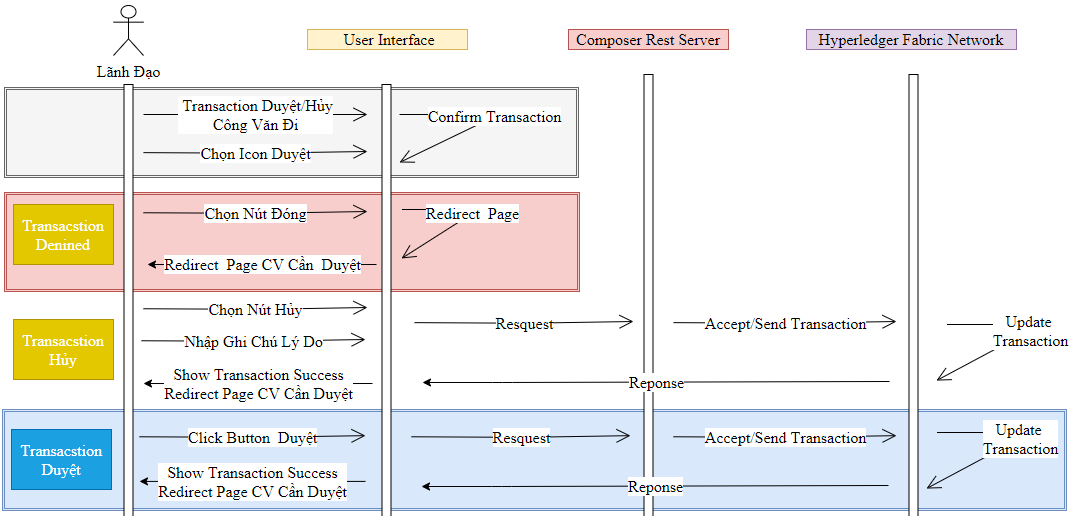
### 3.1.2. Systems workflow

* Hình dưới minh hoa sơ đồ làm việc theo quy trình làm việc ở trên:



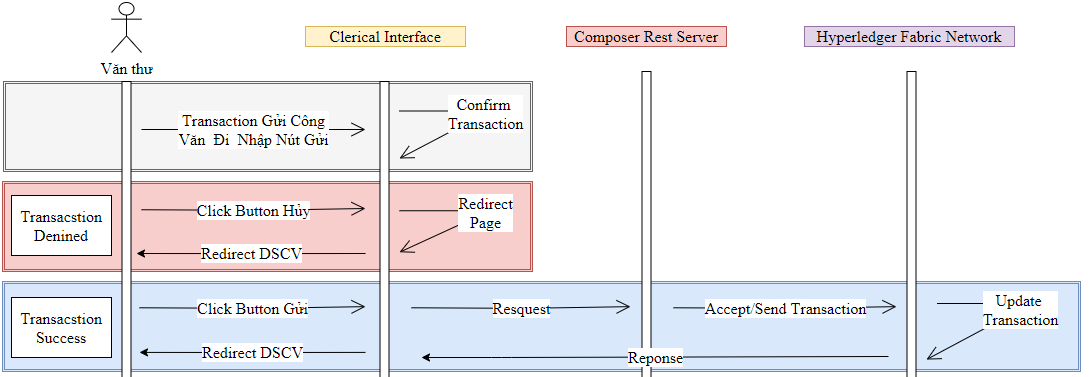
Hình 3. . Thên công văn đi – Sơ đồ trình tự

* Văn thư thực hiện soạn thảo công văn điện tử so công văn giấy, tải công văn bản gốc lên lên hệ thống IPFS để lưu trữ, chờ quá trình lãnh đạo duyệt.



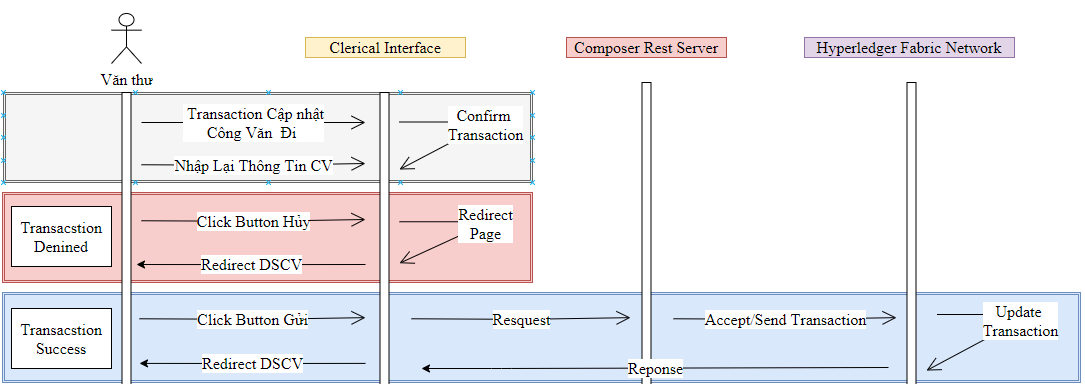
Hình 3. . Duyệt/hủy công văn đi – Sơ đồ trình tự

* Lãnh đạo quản lý đơn vị thực hiện kiểm tra thông tin và duyệt công văn với thông tin là chính xác và hủy công văn với thông tin chưa đúng.



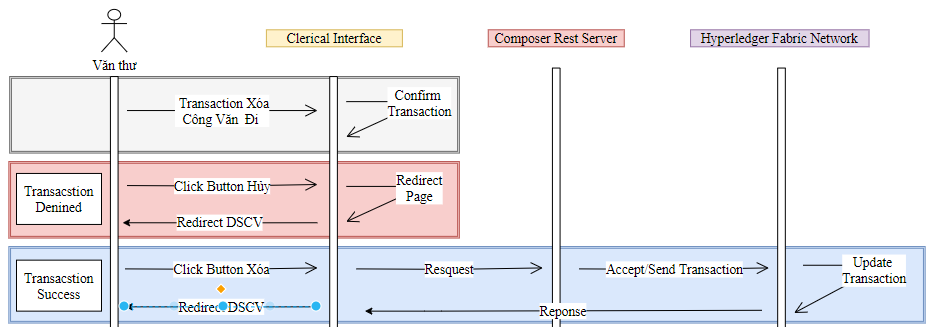
Hình 3. . Transaction gửi công văn đi – Sơ đồ trình tự

* Tiếp theo vă thư tiếp nhận, thực hiện và gửi công văn với công văn đã được duyệt và cập nhật lại thông tin công văn với công văn bị hủy.



Hình 3. . Transaction cập nhật thông tin công văn đi – Sơ đồ trình tự

* Văn thư cập nhật thông tin công văn với thông tin cần sửa đổi.



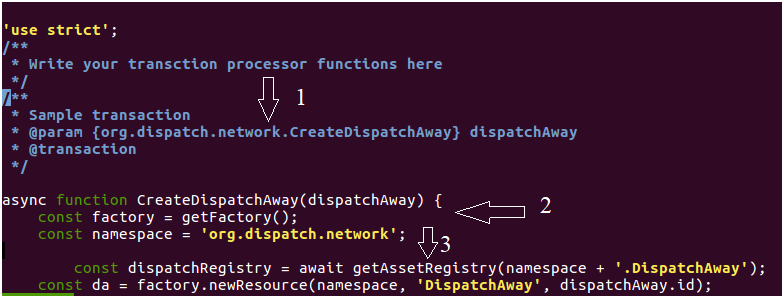
Hình 3. . Transaction xóa công văn đi

### 3.1.3. Các quy trình xử lý

* **Xử lý**

Ở giai đoạn xử lý này thì tương ứng với Script file là các **chaincode** của ứng dụng.

Về cơ bản đa số các giao dịch đều sử dụng function **getFactory()** để truy cập kho chứa blockchain tiếp theo là sử dụng hàm **get<x>Registry()** để lấy world state của asset/participant trên sổ cái, sử dụng hàm **newResource()** để tạo mới một đối tượng, sau đó gán hoặc xử lý dữ liệu từ request vào đối tượng và cuối cùng là sử dụng hàm **add()** để thêm đối tượng vào sổ cái.



Hình 3. . Một đoạn chaincode dùng để thêm mới công văn đi

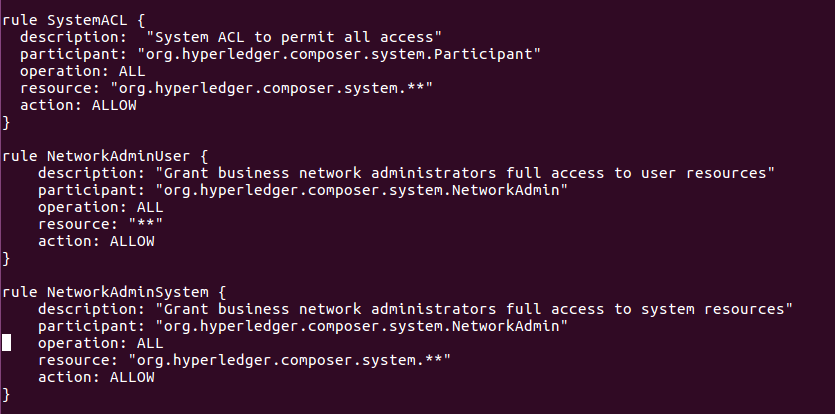
(1) Dòng đầu tiên là mô tả chức năng của hàm, dòng tiếp theo là tag **@param** khai báo namespace + tên Transaction được lưu trong model file.

Dòng chữ **dispatcAway** màu xanh nước biển kế bên dùng làm tham số khi có request gởi đến. Nó được truyền vào tham số của hàm function dùng làm request object để xử lý. Giao dịch có tên **CreateDispatcAway** này dùng để tạo các asset (công văn đi) cần truyền object **dispatcAway** vào. Admin chỉ cần gởi object **dispatcAway** tới url API này thì hàm sẽ tự động được thực thi (các hàm xử lý khác vẫn gửi dữ liệu theo khai báo). **@transaction** tag là bắt buộc phải có.

(2) Để tương tác với blockchain thì bất cứ hàm xử lý giao dịch nào cũng cần phải có **getFactory().**

(3) Xử lý các đối tượng sau đó lưu vào blockchain.

* **Phân quyền**

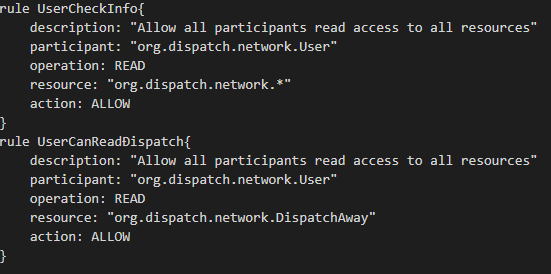


Hình 3. . Phần quyền cho user

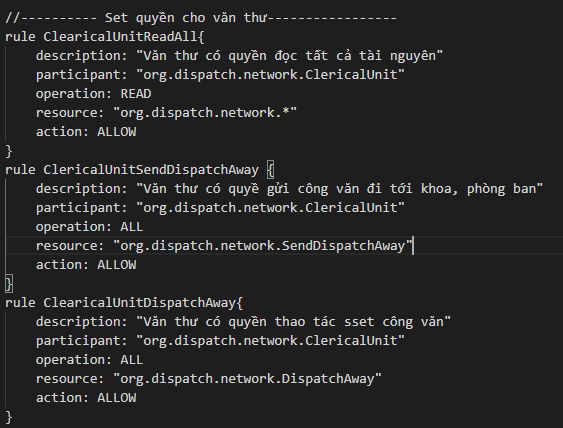
Với tất cả người tham gia sẽ có quyền READ các tài nguyên đã được định nghĩa trong **file model**. Mặc định hệ thống sẽ là **DENY** tất cả hành động đối với tất cả người tham gia đã được khai báo và bao gồm cả **admin** (Được tạo khi deploy file bna vào mạng).

Với các văn thư(ClericalUnit) thì sẽ có thêm quyền CREATE, DELETE, UPDATE đối với các bảng công văn (DispatchAway) và quyền UPDATE đối với bảng lãnh đạo(ManagerUnit) để cập nhật status khi duyệt/hủy thông tin theo quy trình công văn:

Xét quyền cho user (khoa, phòng ban) có quyền đọc thông tin công văn trong hệ thống mạng.

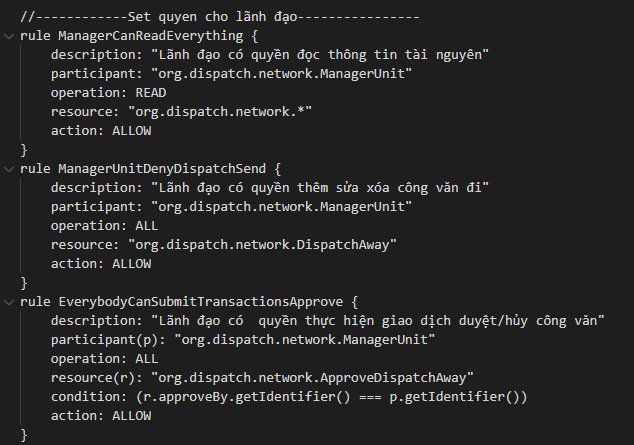


Hình 3. . Phần quyền cho user (phòng ban)



Hình 3. . Phần quyền văn thư

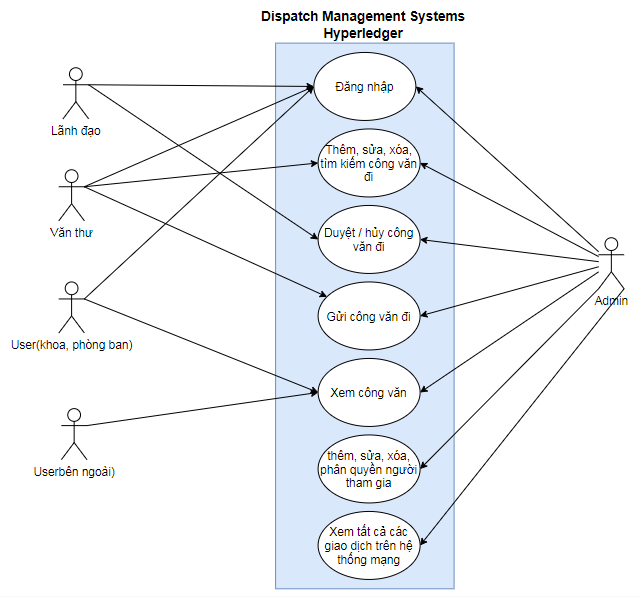
Xét quyền cho văn thư đơn vị có quyền thực hiện full quyền với thông tin công văn trong hệ thống mạng.



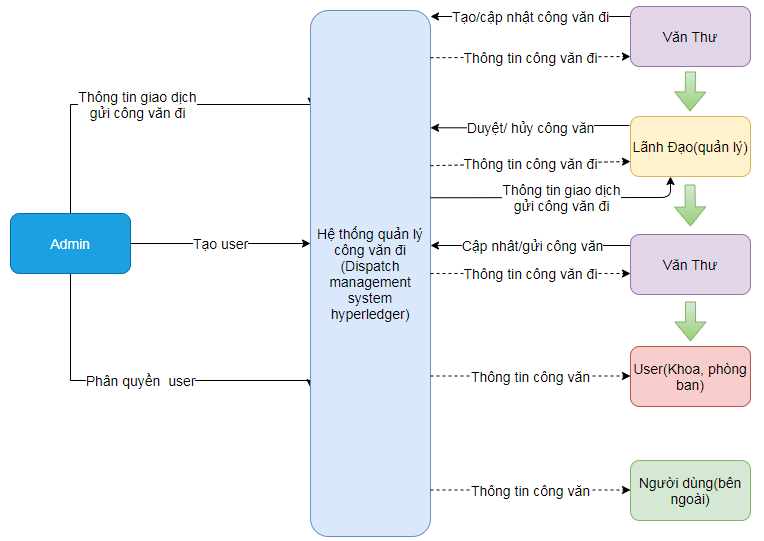
Hình 3. . Phần quyền cho lãnh đạo

Xét quyền cho lãnh đạo có quyền thực hiện full quyền với thông tin công văn trong hệ thống mạng.

* ***Quá trình xử lý quy trình công văn đi***



Hình 3. . Hệ thống Use Case



Hình 3. . Quy trình xử lý công văn đi

Do có liên quan đến quy trình nghiệp vụ để đảm bảo tính xác thực của công văn cũng như tình chính xác của tài liệu nên quy trìn được phân quyền theo từng chức vụ, phòng ban. Tức là mỗi giai đoạn sẽ có một người đảm nhiệm và quyền của mỗi giai đoạn là khác nhau. Các role tương ứng:

* Tạo công văn (Văn thư).
* Duyệt/Hủy công văn (Lãnh đạo, giám đốc/phó giám đốc học viện KTQS).
* Gửi công văn (Văn thư).

Để bắt đầu một quy trình gửi công văn đi thì người quản trị cần thêm các thông tin của người tham gia và cung cấp card cho họ, có thể add card họ vào trong mạng để họ tham gia vào mạng. Sau khi nhận được thông báo từ người quản trị thì người tham gia sẽ tiến hành nhập thông tin của mình.

Người quản trị có thể nhìn thấy toàn bộ thông tin mà người tham gia đã nhập cũng như trang thái công văn đang ở giai đoạn nào.

Cuối cùng khi công văn hoàn thành quy trình thì sẽ chuyển sang trạng thái hoàn tất. Bên nhận sẽ có thể check thông tin khi đăng nhập vào hệ thống mạng hoặc với user (từ bên ngoài) không cần đăng nhập mà truy cập trang chủ sẽ hiện thông tin với công văn dành cho tất cả mọi người.

**Các giai đoạn và người đảm nhiệm mỗi giai đoạn trong quy trình**

**1. Tạo công văn đi**

Bảng 3. . Usecase cho chức năng tạo công văn

|  |  |
| --- | --- |
| Tên Usecase | Tạo công văn đi |
| Mô tả | Tạo công văn đi |
| Tác nhân | Văn thư |
| Điều kiện trước xử lý | Đăng nhập thành công |
| Các bước thực hiện | B1: Văn thư đăng nhập vào hệ thống.  B2: Vào danh sách công văn đi, chọn nút thêm công văn đi.  B3: Nhập đầy đủ thông tin công văn nhưu trên văn bản gốc, cùng với các trường liên quan.  B4:Nhấn nút submit để hoàn tất. |
| Sau xử lý | Chuyển sang trang danh sách công văn mới tạo. |

**2. Duyệt/hủy công văn**

Bảng 3. . Usecase cho chức năng duyệt/hủy công văn

|  |  |
| --- | --- |
| Tên Usecase | Duyệt/hủy công văn đi |
| Mô tả | Kiểm tra thông tin nhập đã đúng với thông tin văn bản gốc cùng với các thông tin liên quan. |
| Tác nhân | Lãnh đạo(Giám đốc/phó giám đốc học viện) |
| Điều kiện trước xử lý | Đăng nhập thành công |
| Các bước thực hiện | B1: Lãnh đạo đăng nhập vào hệ thống.  B2: Vào danh sách công văn cần duyệt, chọn nút duyệt công văn.  B3: Kiểm tra thông tin:  Nếu thông tin đúng thì chọn nút duyệt.  Nếu thông tin sai chọn nút hủy.  B4:Nhấn nút submit để hoàn tất. |
| Sau xử lý | Chuyển sang trang danh sách công văn cần duyệt. |

**3. Gửi công văn**

Bảng 3. . Usecase cho chức năng gửi công văn

|  |  |
| --- | --- |
| Tên Usecase | Gửi công văn đi |
| Mô tả | Gửi công văn tới đơn vị, phòng ban hoặc user (bên ngoài). |
| Tác nhân | Văn thư (Học viện) |
| Điều kiện trước xử lý | Đăng nhập thành công |
| Các bước thực hiện | B1: Văn thư đăng nhập vào hệ thống.  B2: Vào danh sách công văn cần gửi, chọn nút gửi công văn.  B3: Kiểm tra lại trang thái thông tin:  Nếu thông tin đúng thì chọn nút gửi.  B4:Nhấn nút submit để hoàn tất. |
| Sau xử lý | Chuyển sang trang danh sách công văn cần gửi. |

## 3.2. Thiết kế dữ liệu

Hyperledger Composer cung cấp ngôn ngữ mô hình dữ liệu dùng để khai báo định nghĩa các tài nguyên trong mạng [3].

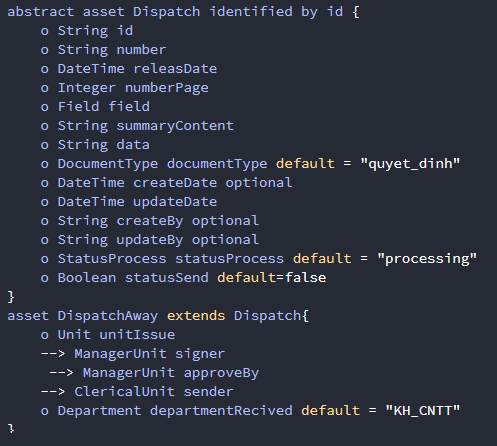
### 3.2.1. Các thành phần cơ bản của file model

- **Namespace**: Các tài nguyên được khai báo bên trong đều được dùng kèm với namespace này.

- Một tập hợp khai báo các tài nguyên bao gồm: assets, participant, transaction và event. Tất cả chúng đều là class và sẽ có các chức năng khác nhau.

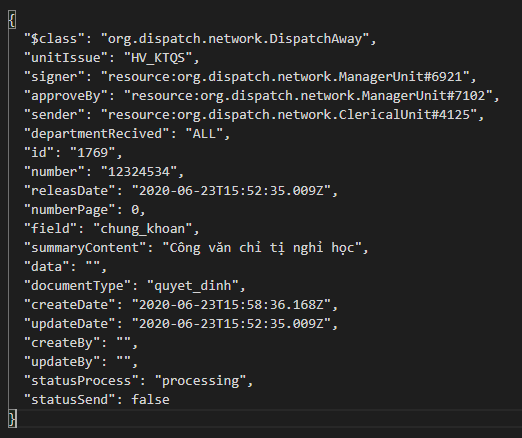
* **Asset**: là lớp đơn giản nhất dùng để lưu các thông tin như là dữ liệu,
* **Participant**: là lớp dùng để khai báo thông tin của người tham gia. Sau khi đã khai báo thì được dùng để tạo identity, tiếp tục dùng identity để tạo ID card và dùng ID card tương tác với mạng theo quy định của **file Access Control List**. Khi sử dụng ID card thì nó sẽ thể hiện người sử dụng như một participant được định nghĩa trong **file Model** và có các quyền được định nghĩa tương ứng trong **file** **Access Control List**.
* **Transaction**: là lớp tương tự như asset nhưng id và timestamp sẽ tự động được sinh ra khi có dữ liệu được xử lý thành công. Lớp này được dùng để xử lý các chaincode thay vì asset vì nó được liên kết với event.
* **Event**: Khi transaction được xử lý thành công thì nó sẽ phát ra một tin nhắn dưới dạng **websocket** dùng để xử lý realtime phía front-end.

Ví dụ :



Hình 3. . Định nghĩa các đối tượng công văn

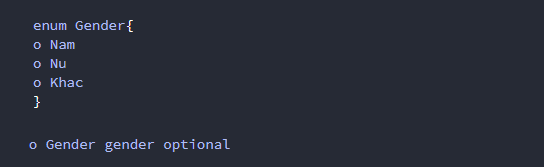
* Hình trên định nghĩa classs DispatchAway (công văn đi) có các thành phần :
* Khai báo một **abstract class Dispatch** để định nghĩa class base (cha) cho các class khác kế thừa tương tự như trong các ngôn ngữkhác.
* Class công văn đi(**DispatchAway**) kế thừa(**extends**) từ **abstract class** **Dispatch.**
* Chữ “**o**” phía trước là ký hiệu khai báo thuộc tính, sau đó là kiểu dữ liệu
* Từ khóa **identified by** dùng để làm khóa chính của đối tượng.
* Một thể hiện tương ứng:



Hình 3. . Thể hiện của asset (công văn đi) cần để lưu vào blockchain.

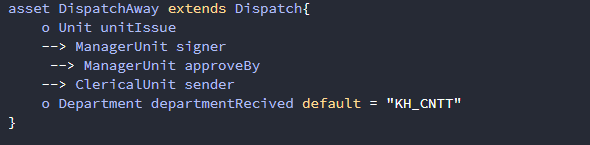
### 3.2.2. Các kiểu dữ liệu được dùng trong model file

* Các kiểu dữ liệu nguyên thủy: String, Double, Integer, Long, DateTime, Boolean.
* Kiểu Enum là danh sách các giá trị có thể sử dụng.

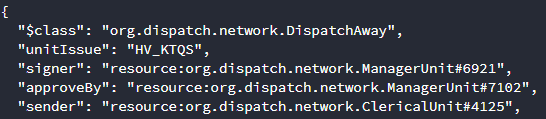


Hình 3. . Khai báo dữ liệu kiểu enum

* Hình. Khai báo kiểu dữ liệu enum và khai báo thuộc tính trong class.
* Kiểu **Relationship** dùng để tham chiếu tới lớp khác, có ký hiệu là **-->**. Mặc dù nó chỉ là thể hiện nhưng nó giúp chúng ta biết đến đường dẫn của API liên quan.



Hình 3. . Kiểu khai báo relationship trong asset (class)



Hình 3. . Thể hiện kiểu relationship trong class

* Kiểu Array được khai báo theo định dạng sau



Hình 3. . Khai báo kiểu mảng trong file model

### 3.2.3. Thiết kế cơ sở dữ liệu

Bảng 3. . Bảng chứa thông tin công văn đi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Asset: DispatchAway** – Quản lý công văn đi | | | | |
| **Tên** | **Kiêu dữ liệu** | **Khóa chính** | **Rỗng** | **Chú thích** |
| id | String | PK | NN | Mã văn bản |
| number | String |  | NN | Số văn bản |
| releasDate | DateTime |  | NN | Ngày phát hành |
| numberPage | Integer |  | NN | Số trang |
| field | Field |  | NN | Lĩnh vực văn bản |
| summaryContent | String |  | NN | Tóm tắt nội dung công văn |
| data | String |  | NN | Dữ liệu mã hóa |
| documentType | DocumentType |  | NN | Loại văn bản |
| createDate | DateTime |  | optional | Ngày tạo |
| updateDate | DateTime |  | optional | Ngày cập nhật |
| createBy | String |  |  | Người tạo |
| updateBy | String |  | optional | Người cập nhật |
| statusProcess | StatusProcess |  | NN | Trạng thái xử lý |
| statusSend | Boolean |  | NN | Trạng thái gửi |
| unitIssue | Enum(Unit) |  | NN | Đơn vị phát hành |
| signer | Relationship (ManagerUnit) |  | NN | Liên kết với lãnh đạo |
| approveBy | Relationship (ManagerUnit) |  | NN | Liên kết với lãnh đạo |
| sender | Relationship (ClericalUnit) |  | NN | Liên kết văn thư |
| departmentRecived | Enum(Department) |  | NN | Phòng bạn nhận |

Bảng 3. . Bảng chứa thông tin văn thư với quyền tạo/gửi công văn đi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Participant: ClericalUnit** – Văn thư | | | | |
| **Tên** | **Kiêu dữ liệu** | **Khóa chính** | **Rỗng** | **Chú thích** |
| id | String | PK | NN | Mã văn thư |
| fullName | String |  | NN | Họ và tên văn thư |
| userName | String |  | NN | Tên đăng nhập |
| password | String |  | NN | Mật khẩu |
| address | String |  | optional | Địa chỉ |
| email | String |  | optional | Gmail đăng ký |
| gender | Enum(Gender) |  | optional | Giới tính |
| unit | Enum(Unit) |  | NN | Thuộc đơn vị nào |

Bảng 3. . Bảng chứa thông tin lãnh đạo với quyền duyệt công văn đi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Participant: ManagerUnit** – Lãnh đạo | | | | |
| **Tên** | **Kiêu dữ liệu** | **Khóa chính** | **Rỗng** | **Chú thích** |
| id | String | PK | NN | Mã lãnh đạo, giám đốc |
| fullName | String |  | NN | Họ và tên văn thư |
| userName | String |  | NN | Tên đăng nhập |
| password | String |  | NN | Mật khẩu |
| address | String |  | optional | Địa chỉ |
| email | String |  | optional | Gmail đăng ký |
| gender | Enum(Gender) |  | optional | Giới tính |
| unit | Enum(Unit) |  | NN | Thuộc đơn vị nào |

Bảng 3. . Bảng chứa thông tin người dùng nhận với quyền xem công văn đi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Participant: User**– User(Khoa, phòng ban) | | | | |
| **Tên** | **Kiêu dữ liệu** | **Khóa chính** | **Rỗng** | **Chú thích** |
| id | String | PK | NN | Mã user của khoa, phòng ban |
| fullName | String |  | NN | Họ và tên văn thư |
| userName | String |  | NN | Tên đăng nhập |
| password | String |  | NN | Mật khẩu |
| address | String |  | optional | Địa chỉ |
| email | String |  | optional | Gmail đăng ký |
| gender | Enum(Gender) |  | optional | Giới tính |
| department | Enum(Department) |  | NN | Thuộc đơn vị nào |

Bảng 3. . Bảng kiểu enumchứa thông tin nơi nhận phòng ban công văn

|  |  |
| --- | --- |
| **Enum: Department** – Tên Các đơn vị phòng ban | |
| **Tên** | **Chú thích** |
| KH\_CNTT | Khoa công nghệ thông tin |
| KH\_XD | Khoa xây dựng |
| KH\_TOAN | Khoa toán |
| KH\_CDT | Khoa cơ điện tử |
| KH\_CK | Khoa cơ khí |
| KH\_HHVT | Khoa hang không vũ trụ |
| ALL | Tất cả |

Bảng 3. . Kiểu enum chứa thông tin đơn vị công văn

|  |  |
| --- | --- |
| **Enum: Unit** – Đợn vị | |
| **Tên** | **Chú thích** |
| HV\_KTQS | HVKT Quân Sự |
| HV\_KTMM | HVKT Mật Mã |
| DH\_QGHN | ĐH QG Hà Nội |
| HV\_TAICHINH | HV Tài Chính |
| HV\_BCVT | HVBC Viễn Thông |

Bảng 3. . Kiểu enum chứa thông tin lĩnh vực của công văn

|  |  |
| --- | --- |
| **Enum: Field** – Lĩnh vực | |
| **Tên** | **Chú thích** |
| chung\_khoan | Lĩnh vực chứng khoán |
| doanh\_nghiep | Lĩnh vực doanh nghiệp |
| giao\_duc | Lĩnh vực giáo dục |
| lao\_dong | Lĩnh vực lao động |
| tien\_luong | Lĩnh vực tiền lương |
| bao\_hiem | Lĩnh vực bảo hiểm |
| cntt | Lĩnh vực công nghệ thông tin |
| tn\_moi\_truong | Lĩnh vực tài nguyên và môi trường |
| van\_tai |  |
| linh\_vu\_khac |  |

Bảng 3. . Kiểu enum chứa loại tài liệu của công văn

|  |  |
| --- | --- |
| **Enum: DocumentType** – Loại tài liệu | |
| **Tên** | **Chú thích** |
| chi\_thi | Văn bản chỉ thị |
| hien\_phap | Hiến pháp |
| lenh | Lệnh |
| luat | Văn bản luật |
| nghi\_dinh | Văn bản nghị định |
| nghi\_quyet | Văn bản nghị quyết |
| phap\_lenh | Văn bản pháp lệnh |
| quyet\_dinh | Văn bản quyết định |
| sac\_lenh | Văn bản sắc lệnh |
| thong\_tu | Văn bản thông tư |

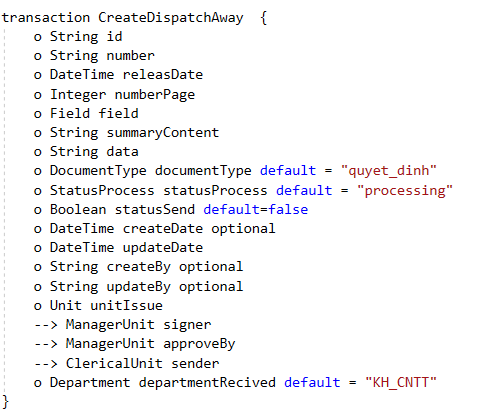
Bảng 3. . Kiểu enum chứa trang thái xử lý của công văn

|  |  |
| --- | --- |
| **Enum: StatusProcess** – Trạng thái xử lý công văn | |
| **Tên** | **Chú thích** |
| processed | Công văn đã được duyệt |
| processing | Công văn đang chờ duyệt |
| cancel | Công văn bị hủy |

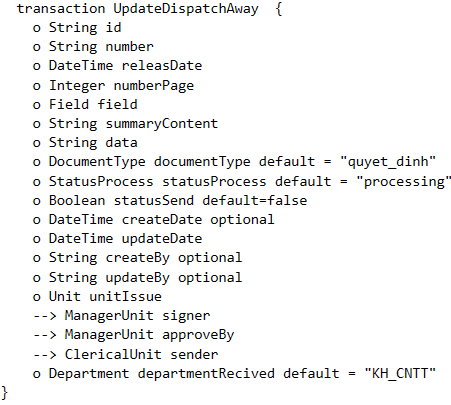
Bảng 3. . Kiểu enum chứa thông tin giới tính user

|  |  |
| --- | --- |
| **Enum: Gender** – Giới tính | |
| **Tên** | **Chú thích** |
| Nam | Giới tính nam |
| Nu | Giới tính nữ |
| Khac | Giới tính thứ ba |

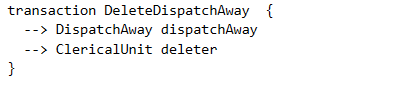
### 3.2.4. Thiêt kế transactions



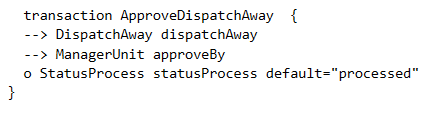
Hình 3. . Cấu trúc lưu trữ transactions khi tạo công văn



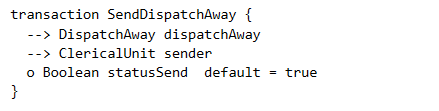
Hình 3. . Cấu trúc lưu trữ transactions khi cập nhật công văn



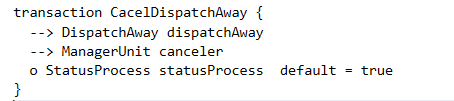
Hình 3. . Cấu trúc lưu trữ transactions khi xóa công văn



Hình 3. . Cấu trúc lưu trữ transactions khi duyệt công văn



Hình 3. . Cấu trúc lưu trữ transactions khi gửi công văn

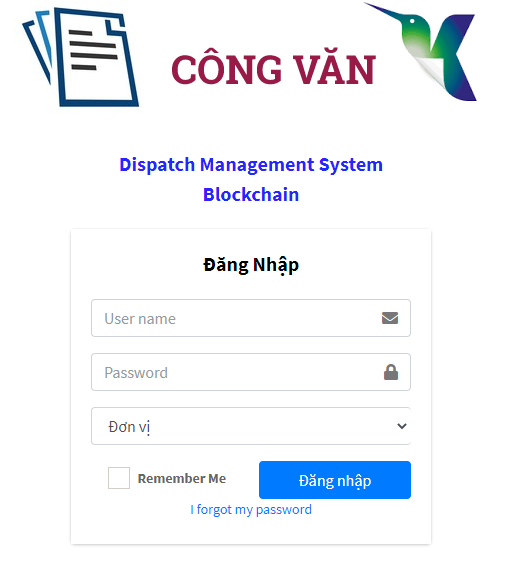


Hình 3. . Cấu trúc lưu trữ transactions khi hủy công văn

## 3.3. Thiết kế giao diện

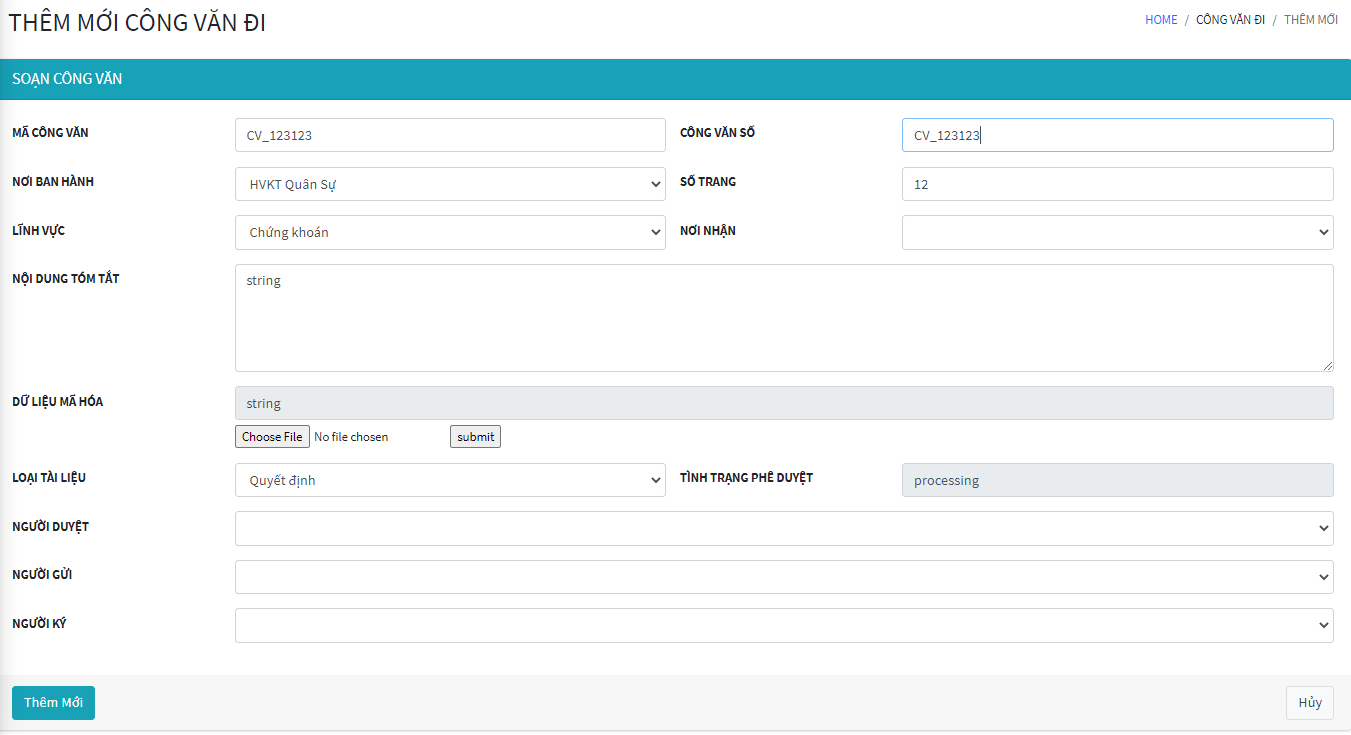
### 3.3.1. Đăng nhập

Giao diện dành cho các người tham gia trong quy trình gửi công văn đăng nhập để gửi công văn theo quy trình. Họ sẽ nhập thông tin được cung cấp bởi người quản trị bao gồm ID card.

****

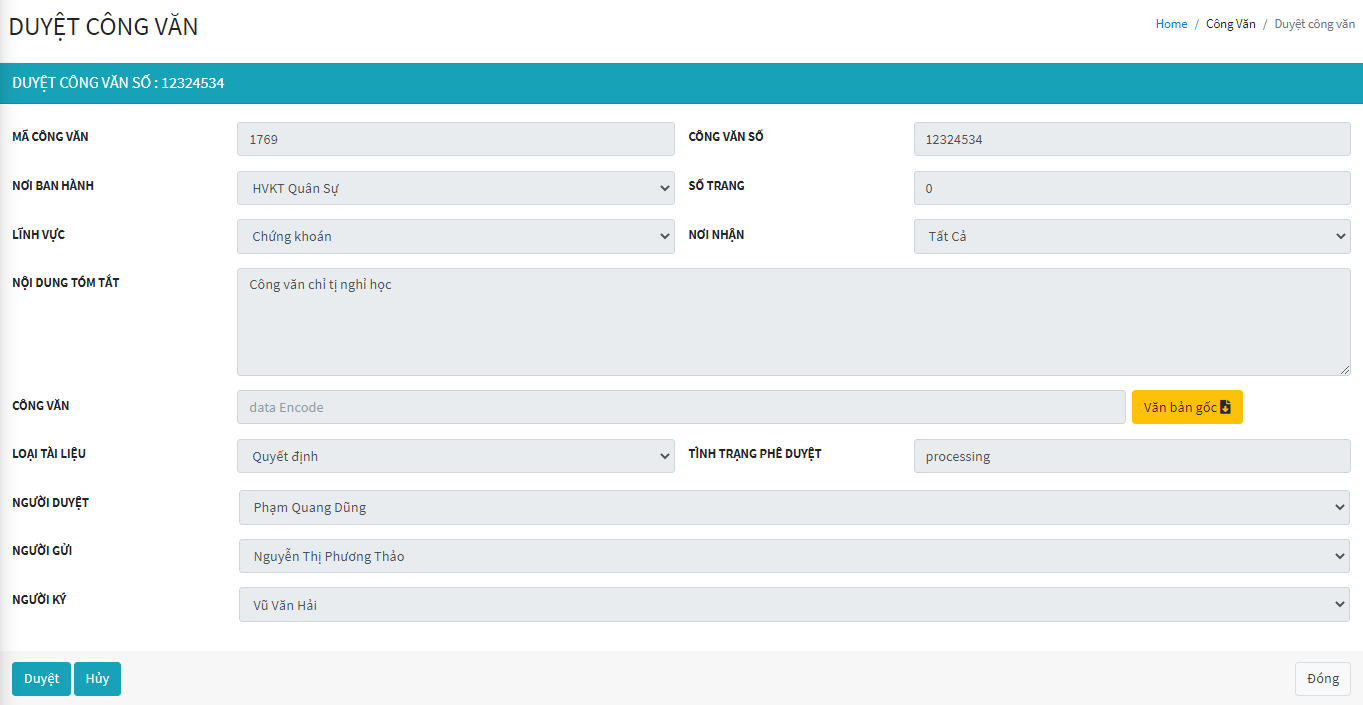
Hình 3. . Giao diện đăng nhập

### 3.3.2. Tạo công văn đi



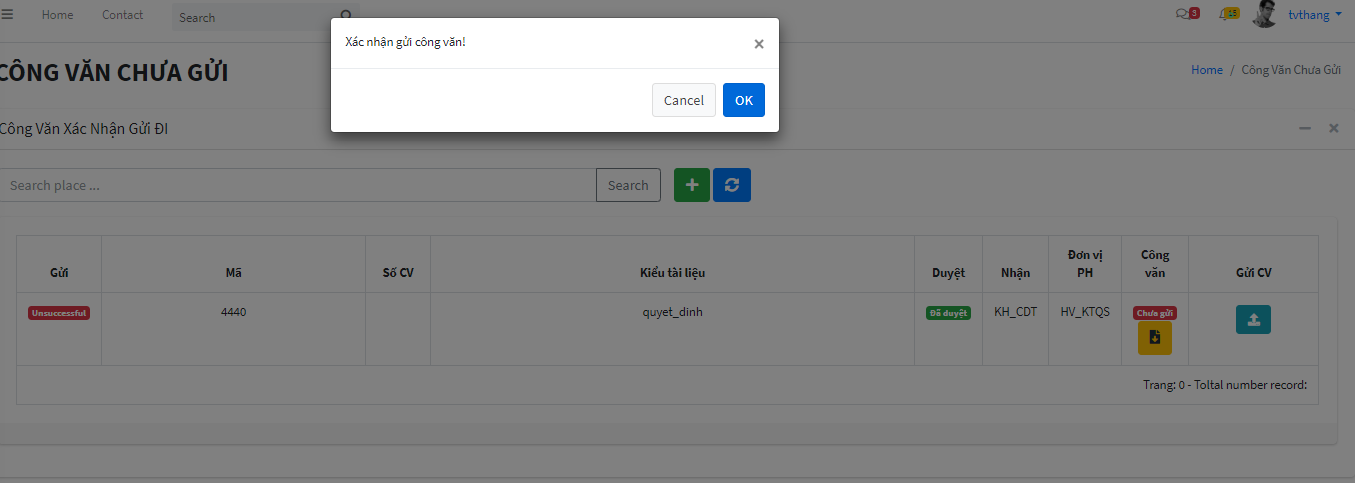
Hình 3. . Giao điện tạo công văn đi

### 3.3.3. Duyệt công văn đi



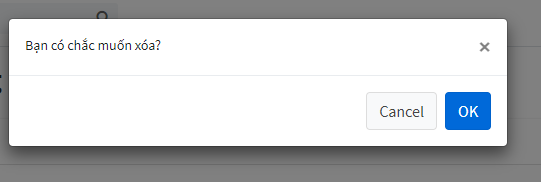
Hình 3. . Giao diện duyệt / hủy công văn đi

### 3.3.4. Xác nhận gửi công văn đi



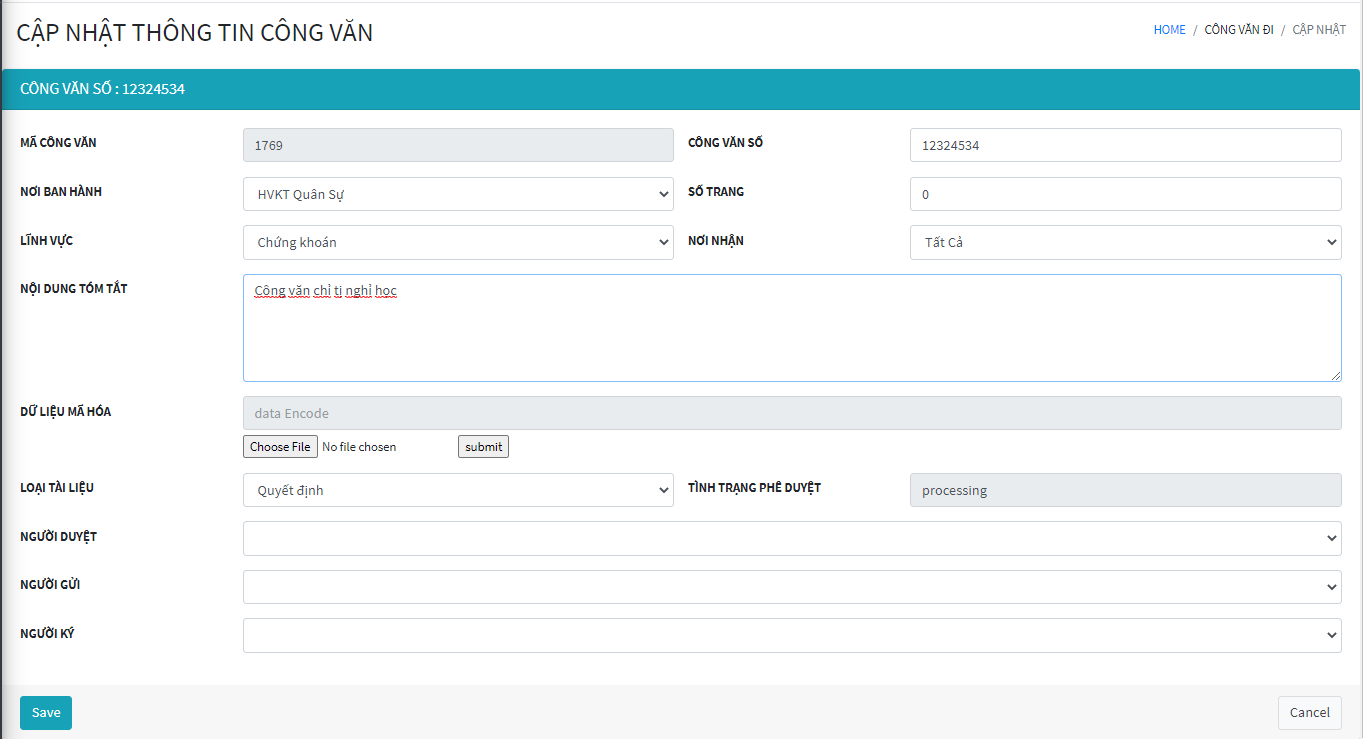
Hình 3. . Giao diện gửi công văn đi

### 3.3.5. Xác nhận xóa



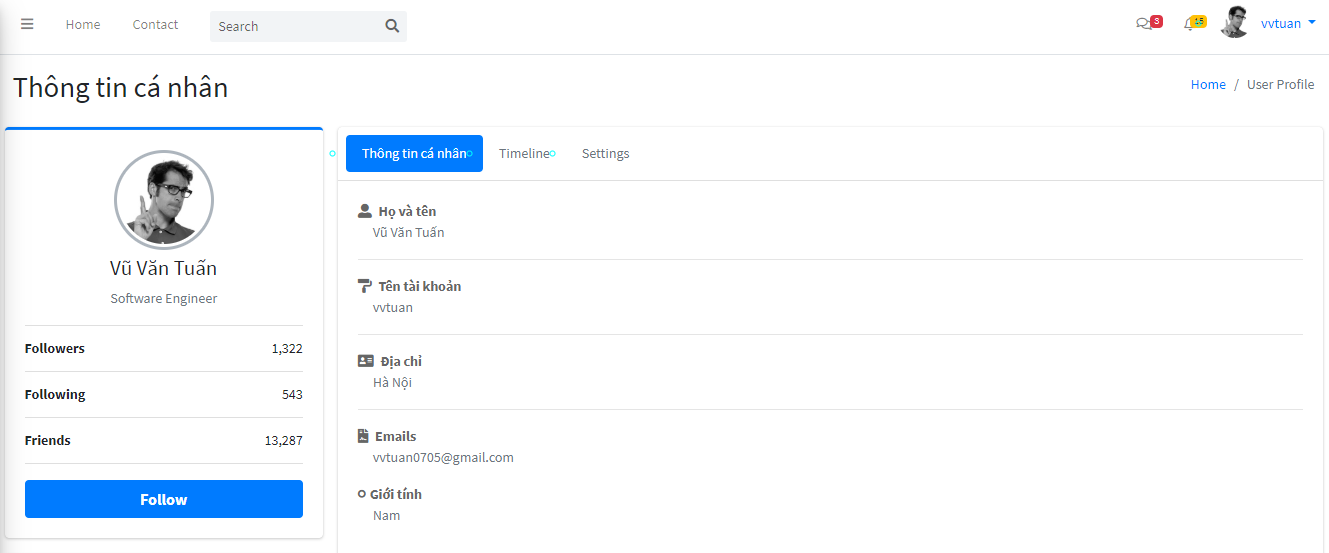
Hình 3. . Giao diện xóa công văn đi

### 3.3.6. Cập nhật thông tin công văn



Hình 3. . Giao diện cập thông tin công văn đi

### 3.3.7. Xem thông tin cá nhân



Hình 3. . Giao diện xem thông tin cá nhân

## Tổng kết chương

Từ những phân tích đã có ở chương 1 và chương 2 nên em đã thực hiện thiết kế business model và cơ sở dữ liệu cho hệ thống blockchain Hyperledger và thực hiện thiết kế giao diện cho hệ thống ở nền tảng web quản trị công văn và người dùng.

# Chương 4 KIỂM THỬ VÀ ĐÁNH GIÁ

## 4.1. Giới thiệu

### 4.1.1. Mục tiêu

* Nhằm kiểm soát và phát hiện các lỗi tiềm ẩn trong hệ thống và kiểm tra các tính năng có hoạt động đúng như kỳ vọng hay không.
* Tạo cơ sở tốt cho giai đoạn bảo trì hệ thống.

### 4.1.2. Phạm vi kiếm thử

* Chỉ kiểm tra hạn chế các chức năng với kiểm thử chấp nhận.

## 4.2. Chi tiết kế hoạch kiểm thử

### 4.2.1. Các chức năng sẽ được kiểm thử

* Đăng nhập phần quyền
* Thêm thông tin công văn đi
* Duyệt công văn đi
* Gửi công văn

### 4.2.2. Các chức năng sẽ không được kiểm thử

* Kiểm thử về hiệu năng, kiểm thử tải, kiểm thử bảo mật.

### 4.2.3. Tiêu chí kiểm thử thành công/thất bại

* Kiểm thử thành công khi kết quả cuối cùng của chuỗi thao tác đúng như mong đợi đặt ra ban đầu.
* Kiểm thử thất bại khi xảy ra lỗi không định trước hoặc kết quả cuối cùng sai lệch so với kỳ vọng ban đầu.

## 4.3. Quản lý kiểm thử

### 4.3.1. Tiến hành kiểm thử

* Lập kế hoạch kiểm thử
* Tạo test case
* Tiến hành kiểm thử
* Tạo kết quả kiểm thử

### 4.3.2. Môi trường kiểm thử

* Hệ điều hành Ubuntu 16.04 và windows 10 pro
* Trình duyệt chrome phiên bản mới nhất 2020

## 4.4. Kế hoạch dự đoán và chi phí

Bảng 4. . Kế hoạch kiểm thử

|  |  |  |
| --- | --- | --- |
| **Tên công việc** | **Thời gian bắt đầu** | **Thời gian kết thúc** |
| **Lập kế hoạch kiểm thử** | 12/7/2020 | 12/7/2020 |
| **Kiểm thử lần 1** | 15/7/2020 | 15/7/2020 |
| **Kiểm thử lần 2** | 17/7/2020 | 17/7/2020 |
| **Đánh giá** | 18/7/2020 | 18/7/2020 |

## 4.5. Các rủi ro

Bảng 4. . Các rủi ro có thể xảy ra

|  |  |  |
| --- | --- | --- |
| **STT** | **Các rủi ro có thể xảy ra** | **Kế hoạch làm giảm bớt hoặc tránh** |
| **1** | Thời gian kiểm thử thực tế dài hơn thời gian dự đoán | Tăng thời gian thảo luận nhóm, phân chia lại công việc |
| **2** | Lỗi hệ điều hành | Chuẩn bị trước máy tính dự phòng để thay thế |
| **3** | Chưa có nhiều kinh nghiệm trong việc kiểm thử | Tham khảo thêm từ các tài liệu liên quan trên diễn đàn, website chuyên về kiểm thử chức năng |

## 4.6. Các trường hợp kiểm thử

### 4.6.1. Chức năng đăng nhập

* Mục đích kiểm thử: Kiểm tra đăng nhập theo các trường hợp để đảm bảo chức năng hoạt động chính xác.
* Tiền điều kiện: Mở được ứng dụng quản công văn vào được phần đăng nhập.
* Mô tả
* Mở ứng dụng quản lý công văn
* Chọn nút Đăng nhập/Đăng xuất
* Nhập thông tin đã được cung cấp bởi admin
* Nhấn nút Đăng Nhập
* Kịch bản:

Bảng 4. . Kiểm thử chức năng đăng nhập

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mô tả dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Thành công/Thất bại** |
| 1 | Nhập thông tin và id card hợp lệ | Đăng nhập thành công, chuyển về trang chủ Quản Lý Công Văn | Đăng nhập thành công, chuyển về trang chủ Quản Lý Công Văn | Thành công |
| 2 | Nhập thông tin và id card không hợp lệ | Đăng nhập không thành công và hiện thông báo | Đăng nhập không thành công và hiện thông báo | Thành công |

### 4.6.2. Chức năng tạo công văn

* Mục đích kiểm thử: Kiểm tra chức năng thêm thông tin công văn theo các trường hợp để đảm bảo chức năng hoạt động chính xác.
* Tiền điều kiện: Đã đăng nhập và tìm thấy danh mục công văn đi.
* Mô tả
* Mở ứng dụng quản lý công văn
* Chọn nút đăng nhập/đăng xuất
* Đăng nhập
* Tìm kiếm danh mục công văn đi
* Chọn nút Thêm
* Kịch bản

Bảng 4. . Kiểm thử chức năng tạo công văn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mô tả dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Thành công/thất bại** |
| 1 | id: null,  number: null,  releasDate: "2020-06-14T15:20:46.674Z",  numberPage: 12,  field: ''chung\_khoan'',  summaryContent: ''string'',  unitIssue: 'HV\_KTQS',  data: null,  documentType: ''quyet\_dinh'',  statusProcess: ''processing'',  departmentRecived: ''cntt'',  approveBy: '',  sender: null,  signer: null,  createBy: null,  statusSend: ''false'',  createDate:null | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Thành công |
| 2 | id: null,  number: null,  releasDate: "2020-06-14T15:20:46.674Z",  numberPage: 12,  field: null,  summaryContent: ''string'',  unitIssue: ''HV\_KTQS',  data: null,  documentType: ''quyet\_dinh'',  statusProcess: ''processing'',  departmentRecived: null,  approveBy: null,  sender: null,  signer: null,  createBy: null,  statusSend: ''false'',  createDate: null,  updatedate: null | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Thành công |
| 3 | id: ''CV\_HN'',  number: ''CV\_123'',  releasDate: new Date(),  numberPage: ''12'',  field: ''giao\_duc'',  summaryContent: ''tóm tắt nd'',  unitIssue: ''HV\_KTQS'',  data: ''chuỗi mã hóa'',  documentType:''quyet\_dinh'',  statusProcess: ''processing'',  departmentRecived: ''KH\_CNTT'',  approveBy: ''resource:org.dispatch.network.ManagerUnit#vvtuan'',  sender: ''resource:org.dispatch.network.ClericalUnit#tvthang'',  signer: ''resource:org.dispatch.network.ManagerUnit#vvtuan'',  createBy: ''vvtuan'',  statusSend: ''false'',  createDate: new Date(),  updatedate: new Date() | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Thành công |

### 4.6.3. Chức năng duyệt/hủy công văn

* Mục đích kiểm thử: Kiểm tra chức năng duyệt hoặc hủy thông tin công văn theo các trường hợp để đảm bảo chức năng hoạt động chính xác.
* Tiền điều kiện: Đã đăng nhập và tìm thấy danh mục danh sách công văn cần duyệt với user là lãnh đạo cấp học viện.
* Mô tả
* Mở ứng dụng quản lý công văn
* Chọn nút đăng nhập/đăng xuất
* Đăng nhập
* Tìm kiếm danh mục danh sách công văn cần duyệt với user là lãnh đạo cấp học viện
* Chọn icon biểu tượng duyệt
* Check nội dung đúng với thông tin điện tử, nhấn duyệt, hoặc hủy
* Kịch bản

Bảng 4. . Kiểm thử chức năng duyệt / hủy công văn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mô tả dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Thành công/thất bại** |
| 1 | dispatchAway: null,  approveBy: null,  statusProcess: null | Báo lỗi trả về từ hệ thống trường rỗng, thực hiện không thành công | Báo lỗi trả về từ hệ thống trường rỗng, thực hiện không thành công | Thành công |
| 2 | dispatchAway: ''org.dispatch.network.DispatchAway#1769'',  approveBy: ''org.dispatch.network.ManagerUnit#vvtuan'',  statusProcess: true | Không báo lỗi trường rỗng, cho phép thực hiện tác vụ | Không báo lỗi trường rỗng, cho phép thực hiện tác vụ | Thành công |

### 4.6.4. Chức năng gửi công văn

* Mục đích kiểm thử: Kiểm tra chức năng gửi thông tin công văn cho nơi nhận theo các trường hợp để đảm bảo chức năng hoạt động chính xác.
* Tiền điều kiện: Đã đăng nhập và tìm thấy công văn cần gửi.
* Mô tả
* Mở ứng dụng quản lý công văn
* Chọn nút đăng nhập/đăng xuất
* Đăng nhập
* Tìm kiếm danh mục công văn cần gửi
* Chọn nút xác nhận gửi đi
* Kịch bản

Bảng 4. . Kiểm thử chức năng gửi công văn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mô tả dữ liệu kiểm thử** | **Kết quả mong đợi** | **Kết quả thực tế** | **Thành công/thất bại** |
| 1 | dispatchAway: null,  sender: null,  statusSend: null | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Thành công |
| 2 | dispatchAway: ''org.dispatch.network.DispatchAway#1769'',  sender: ''org.dispatch.network.ClericalUnit#vvtuan'',  statusSend: true | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Báo lỗi trường rỗng, không cho thực hiện tiếp tác vụ | Thành công |

## Tổng kết chương

Trong chương 4 em đã thực hiện kiểm thử với một vài dữ liệu để đánh giá hoạt động của ứng dụng web cũng như hệ thống Hyperledger Fabric và đưa ra kết quả đúng như nghiệp vụ mong muốn đề ra.

# KẾT LUẬN

**Kết quả đạt được**

**1. Về lý thuyết**

Qua quá trình nghiên cứu công nghệ, tìm hiểu quy trình quản lý công văn và phát triển ứng dụng đã giúp tôi có thêm nhiều kiến thức để phát triển ứng dụng với công nghệ blockchain Hyperledger Fabric và Angular framework. Nhờ đó cũng biết cách xây dựng một ứng dụng từ khâu thiết kế cấu trúc hệ thống, mô hình dữ liệu cho đến các bước thiết kế và cài đặt để dần hình thành một hệ thống hoàn chỉnh.

Với kiến thức đã tìm hiểu về Hyperledger Fabric có thể nói công nghệ blockchain này sẽ được phát triển mạnh mẽ ở nước ta. Nếu được đầu tư bài bản và có kế hoạch chi tiết thì vấn đề quản lý lưu trữ công văn sẽ được giải quyết đáng kể nhờ công nghệ này.

Nắm được cách lưu trữ dữ liệu và truy xuất dữ liệu sử dụng công cụ Hyperledger Composer với các API mà nó cung cấp.

Với Angular, từ việc thiết kế giao diện cho đến tạo các Component, sử dụng các module giúp tôi có thêm kinh nghiệm trong việc phát triển ứng dụng kiểu single page application, phần nào hiển thị được đầy đủ thông tin đã được thiết kế ban đầu.

**2. Về ứng dụng**

Sau thời gian phát triển thì ứng dụng cũng đã đáp ứng được vấn đề ứng dụng blockchain vào quản lý công văn. Tính năng quản lý, lưu trữ công văn, phân quyền nhanh chóng đáp ứng yêu cầu đặt ra.

**3. Tính thực tiễn**

Ứng dụng có thể cơ bản đáp ứng được cho việc lưu trữ và quản lý công văn.

Các công nghệ sử dụng trong đề tài là các công nghệ mới, mang nhiều ưu điểm vượt trội chính là nền tảng tốt cho việc mở rộng và nâng cấp về sau. Cùng với khả năng hoạt động đa nền tảng, Webservice được cung cấp theo chuẩn RESTful mang lại tiềm năng phát triển thêm các ứng dụng ở nền tảng khác rất khả quan.

**4. Hạn chế**

Bên cạnh việc học được thêm nhiều kiến thức về công nghệ thì cũng có thêm nhiều hạn chế:

- Công nghệ blockchain Hyperledger Fabric là một công nghệ mới xuất hiện gần đây vì vậy sẽ mất thời gian để tìm cách triển khai ứng dụng và sửa lỗi khi gặp phải.

- Lưu trữ quản lý công văn trên hệ thống blockchain là một đề tài mới mẻ nên việc thiết kế cấu trúc dữ liệu không được tốt dẫn đến khó phát triển về sau khi dữ liệu lớn dần.

- Một vài phần tử vẫn chưa hiển thị được tốt trên các kích thước màn hình khác nhau.

- Việc ứng dụng block vào việc lưu trữ công văn trên thực tế không phải là ý tưởng hay cho công việc này, chưa hoàn toàn phù hợp trong việc quản lý, và đây là đề tài chỉ đến mức nghiên cứu và có thể phát triển ứng dụng vào cuộc sống với nhiều lĩnh vực và ý tưởng khác sẽ phù hợp hơn.

**5. Hướng phát triển**

Do đề tài được thực hiện ở mức nghiên cứu, học hỏi để hoàn thiện kỹ năng chuyên môn và đáp ứng các yêu cầu thiết yếu nên vẫn còn thiết sót nhiều tính năng. Với hướng phát triển trong tương lai thì có thể mở rộng các chức năng sau:

- Phân quyền trong hệ thống mạng 1 cách tự động phía client không cần phụ thuộc vào admin mỗi lần đăng nhập.

- Hoàn thiện hiển thị các giao diện để hiển thị tốt hơn trên các trình duyệt cũng như kích thước màn hình khác nhau.

- Chuyển ứng dụng backend về dạng NodeJS application để có thể dễ dàng sử dụng middle-ware thay cho cơ chế chứng thực có sẵn.

# TÀI LIỆU THAM KHẢO

[1] ARAF KARSH HAMID Co-Founder/CTO MetaMagic Global Inc., NJ, USA, BlockChain HyperLedger Fabric, World Blockchain Summit December 6-7, 2018, The Ritz-Carlton, Bengaluru.

[2] Elli Androulaki Artem Barger Vita Bortnikov Christian Cachin Konstantinos Christidis Angelo De Caro David Enyeart Christopher Ferris Gennady Laventman Yacov Manevich Srinivasan Muralidharan∗ Chet Murthy† Binh Nguyen ∗ Manish Sethi Gari Singh Keith Smith Alessandro Sorniotti Chrysoula Stathakopoulou Marko Vukoli´c Sharon Weed Cocco Jason Yellick I B M, arXiv:1801.10228v1 [cs.DC] 30 Jan 2018

[3] <https://hyperledger.github.io/composer/v0.19/installing/installing-index>

[4] <https://medium.com/coinmonks/how-does-hyperledger-fabric-works-cdb68e6066f5#:~:text=Orderer%20node%20includes%20the%20transaction,peers%20inside%20their%20own%20organization.>

[5] <https://hyperledger.github.io/composer/v0.19/integrating/enabling-rest-authentication>

[6] <https://mraddon.blog/2018/07/15/how-to-push-load-image-file-from-to-ipfs-using-javascript-examples-part-iii/>

[7] <https://medium.com/@CazChurchUk/developing-multi-user-application-using-the-hyperledger-composer-rest-server-b3b88e857ccc>

[8] <https://angular.io/>

[9] <https://medium.com/coinmonks/hyperledger-fabric-composer-errors-solutions-827112a3fce6>

[10]<https://hyperledger.github.io/composer/latest/tutorials/queries?fbclid=IwAR205CPFJDDW0mljizDb27Nr_-lwg-EYM0eS_ZHyslhpPkFOUKUHjNKlZXQ>

[11] <https://www.udemy.com/course/hyperledger/>

[12] <https://hyperledger-fabric.readthedocs.io/en/release-1.2/network/network.html>

# PHỤ LỤC

**Phụ lục 1: Hướng dẫn cài đặt hệ thống**

**1.1. Cài đặt hệ thống hyperledger**

* Câu lệnh cài đặt các công cụ cần
* Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
* Docker Engine: Version 17.03 or higher
* Docker-Compose: Version 1.8 or higher
* Node: 8.9
* npm: v5.x
* git: 2.9.x or higher
* Python: 2.7.x
* Có thể sử dụng script có sẵn để tự động cài đặt:

**curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh**

**chmod u+x prereqs-ubuntu.sh**

**./prereqs-ubuntu.sh**

* **Cài đặt môi trường phát triển**

Hyperledger Composer (chủ yếu được sử dụng để tạo Mạng doanh nghiệp) và thiết lập Hyperledger Fabric (chủ yếu được sử dụng để chạy / triển khai Mạng doanh nghiệp cục bộ). Lưu ý rằng business network cũng có thể được triển khai vào thời gian chạy của Hyperledger Fabric trong các môi trường khác, ví dụ: trên nền tảng đám mây.

* Cài đặt thành phần
* instsall the CLI tool:

**npm install -g composer-cli@0.19**

* Install tiện ích để chạy máy chủ REST để hiện thị business network dưới dang API Restful:

**npm install -g composer-rest-server@0.19**

* Tiện ích tạp ra các assets:

**npm install -g generator-hyperledger-composer@0.19**

* Yeoman là công cụ tạo ra các ứng dụng

**npm install -g yo**

* Cài đặt playground

Playground cung cấp giao diện người dùng có thể xem và tương tác với mạng

* Browser app có thể chỉnh sửa và testing dễ dàng với mạng

**npm install -g composer-playground@0.19**

* Cài đặt IDE

Install visual studio code cho việc phát triển ứng dụng và visual studio code cho việc xây dựng ứng dụng.

* Cài đặt Hyperledger Fabric

Cung cấp môi trường để có thể deploy business network

* Tạo thư mục ~/Fabric-dev-server, clone file trên github chứa file công cụ để cài đặt Hyperledger Fabric

**mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers**

**curl -O** [**https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz**](https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz) **tar -xvf fabric-dev-servers.tar.gz**

* Giải nén và chạy file để cài đặt Hyperledger Fabric

**cd ~/fabric-dev-servers**

**export FABRIC\_VERSION=hlfv12**

**./downloadFabric.sh**

* Kiểm soát môi trường phát triển
* Start hoặc stop quá trình runtime

**Khởi chạy hệ thống:** ~/fabric-dev-servers/startFabric.sh

**Đừng hệ thống:** ~/fabric-dev-servers/stopFabric.sh

**Phá hủy hệ thống:** ~/fabric-dev-servers/teardownFabric.sh

* Khởi chạy web app cho phép deveper edit và testing project đang phát triển

**cd ~/fabric-dev-servers**

**export FABRIC\_VERSION=hlfv12**

**./startFabric.sh**

**./createPeerAdminCard.sh**

* Start web app editor develop

**composer-playground**

* Cài đặt file **bna** lên mạng Fabric:
* Tải source code của đề tài tại :
* Giải nén và di chuyển tới thư mục của file bna. Sau đó thực hiện cài đặt theo hướng dẫn sau (từ bước 4) :

<https://hyperledger.github.io/composer/v0.19/tutorials/developer-tutorial.html>

* Chi tiết: Cài đặt chaincode vào mạng:

**$composer network install --card PeerAdmin@hlfv1 --archiveFile** **dispatchnetwork.bna**

* Chạy mạng đã cài (*chú ý version được hiển thị trên terminal*):

**$composer network start --networkName dispatch-network --networkVersion 0.0.12-final --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card.**

* Sau khi chạy thành công sẽ có 1 file **networkadmin.card** được tạo.
* Import card sử dụng để tương tác với mạng:

**$composer card import --file networkadmin.card**

* Kiểm tra mạng:

**$composer network ping --card admin@dispatch-network**

* Chạy Composer-rest-server:

Có 2 chế độ chính là **anonymous** dùng cho người truy xuất và **admin** dùng cho người tham gia quá trình sản xuất sản phẩm:

* Đầu tiên cần tạo người tham gia mẫu bằng cách chạy rest server với admin card:

**$composer-rest-server -c admin@dispatch-network**

Sau đó vào giao diện **explorer** tại url: **<điachỉ>:3000/explorer**

Chọn transaction là **createDemoParticipant**, phương thức POST và nhập object rỗng và nhấn “Try it out”. Nhập các thông tin liên quan trong thư mục source code ở các api tương ứng. Sau khi giao dịch thành công thì tắt server đi.

* Kích hoạt **multiuser mode** bằng hướng dẫn:

<https://www.codementor.io/gangachris125/passport-jwt-authentication-for-hyperledger-composer-rest-server-jqfgkoljn>

* Tạo identity để sử dụng rest server với vai trò là user anonymous

**$composer identity issue -c admin@dispatch-network -f name.card -u user1 -a "resource:org.dispatch.network.ManagerUnit#IdParticipant"**

* Import card vào mạng

**$composer card import --file user.card**

* Chạy composer-rest-server với chế độ anonymous (chỉ được xem)

**$composer-rest-server -c admin@dispatch-network -w true -p 3001**

* Chạy composer-rest-server với chế độ chứng thực

**$composer-rest-server -c admin@dispatch-network -m true -w true**

Với chế độ chứng thực thì người tham gia cần có card (identity) ứng với participant trong file model:

**$composer identity issue -c admin@dispatch-network -f name.card –u tanvan1 -a "resource:org.dispatch-network.ManagerUnit#IdParticipant".**

Sử dụng card **name.card** để tương tác với hệ thống bằng vai trò là lãnh đạo với trách nhiệm duyệt công văn.

**Chú ý**: Khi sử dụng mạng đã có dữ liệu thì cần sử dụng lênh sau để tránh mất dữ liệu:

* Di chuyển vào đường dẫn sau:

**$cd fabric-dev-servers/fabric-scripts/hlfv11/composer/**

**$docker-compose stop** - Dừng trạng thái mạng và không làm việc nữa.

Hoặc **$docker-compose start** để làm việc tiếp.

**1.2. Cài đặt Web API**

**Điều kiện tiên quyết**:

* Cài đặt visual studio code 2019
* Hệ điều hành windows 10

Tải source code web app : <https://github.com/vvthuy0705/dispatch>

Chạy chương trình với IDE Visual Studio 2019, lấy thông tin token sau khi authen git với composer rest server và địa chỉ IP máy ảo nhập vào file gethost.js project, tiến hành run và đăng nhập.