 **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: ỨNG DỤNG CÔNG NGHỆ BLOCKCHAIN TRONG CÔNG TÁC THI CỬ**

**Giảng viên hướng dẫn : TS. ĐẶNG MINH TUẤN**

**Sinh viên thực hiện : NGUYỄN MINH VŨ**

**Lớp : D15CQAT03**

**Mã sinh viên : B15CQAT195**

**Khóa : 2015-2020**

**Hệ : Chính quy**

**Hà Nội – 2019**

# **LỜI CẢM ƠN**

Lời đầu tiên, em xin được gửi lời cảm ơn sâu sắc tới TS. Đặng Minh Tuấn - người thầy đã luôn tận tình giúp đỡ em định hướng nghiên cứu trong suốt quá trình làm đồ án. Đồng thời, thầy cũng là người truyền đạt cho em rất nhiều những kiến thức bổ ích về các lĩnh vực an toàn thông tin.

Em xin được cảm ơn các thầy, cô đang công tác giảng dạy tại Bộ môn an toàn thông tin – Học viện công nghệ bưu chính viễn thông. Trong suốt quá trình học đại học, em đã được các thầy, cô truyền đạt rất nhiều tri thức quí báu.

Em cũng xin được cảm ơn gia đình, người thân và bạn bè của em đã giúp đỡ và hỗ trợ cho em rất nhiều trong suốt thời gian qua.

Em xin chân thành cảm ơn!

Hà Nội, ngày 7 tháng 12 năm 2019

**Nguyễn Minh Vũ**

**MỤC LỤC**

[LỜI CẢM ƠN ii](#_Toc27083091)

[DANH MỤC CÁC HÌNH VẼ, SƠ ĐỒ i](#_Toc27083092)

[DANH MỤC CÁC BẢNG iv](#_Toc27083093)

[MỞ ĐẦU 1](#_Toc27083094)

[CHƯƠNG 1. ỨNG DỤNG BLOCKCHAIN TRONG THI CỬ 3](#_Toc27083095)

[1.1. Giới thiệu bài toán 3](#_Toc27083096)

[1.2. Giới thiệu về công nghệ blockchain 3](#_Toc27083097)

[1.2.1. Các khái niệm 4](#_Toc27083098)

[1.2.2. Cấu trúc 12](#_Toc27083099)

[1.2.3. Đặc điểm và ứng dụng 20](#_Toc27083100)

[1.2.4. Phân loại 24](#_Toc27083101)

[1.3. Giới hạn phạm vi bài toán 25](#_Toc27083102)

[CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG 26](#_Toc27083103)

[2.1. Tổng quan về hệ thống 26](#_Toc27083104)

[2.2. Giới thiệu về Hyperledger Fabric 28](#_Toc27083105)

[2.3. Các công nghệ khác được dùng trong hệ thống 36](#_Toc27083106)

[2.3.1. Docker 36](#_Toc27083107)

[2.3.2. Mongodb 40](#_Toc27083108)

[2.3.3. Kubernetes 46](#_Toc27083109)

[2.4. Xây dựng mạng lưới fabric blockchain 51](#_Toc27083110)

[CHƯƠNG 3. LẬP TRÌNH VÀ ĐÁNH GIÁ 53](#_Toc27083111)

[3.1. Lập trình 53](#_Toc27083112)

[3.1.1. Lập trình mạng blockchain 53](#_Toc27083113)

[3.1.2. Xây dựng api backend 57](#_Toc27083114)

[3.1.3. Xây dựng phần hệ thống 59](#_Toc27083115)

[3.2. Đánh giá 68](#_Toc27083116)

[**3.2.1.** **Hệ thống** 68](#_Toc27083117)

[**3.2.2.** **Giao diện chương trình** 68](#_Toc27083118)

[KẾT LUẬN 73](#_Toc27083119)

[TÀI LIỆU THAM KHẢO 74](#_Toc27083120)

# 

# DANH MỤC CÁC HÌNH VẼ, SƠ ĐỒ

[Hình 1- 1: Sơ đồ tổng quát quát của hệ thống 2](#_Toc27079575)

[Hình 1- 2:Hợp đồng thông minh trong fabric fabcar 5](#_Toc27079576)

[Hình 1- 3: Mạng phân bố P2P Network vs Centralized Network 6](#_Toc27079577)

[Hình 1- 4: Cây Merkle với lá chẵn 7](#_Toc27079578)

[Hình 1- 5: Cây Merke với lá số lẻ 7](#_Toc27079579)

[Hình 1- 6: Mã hóa đối xứng 8](#_Toc27079580)

[Hình 1- 7: Mã hóa bất đối xứng với khóa công khai và khóa bí mật 8](#_Toc27079581)

[Hình 1- 8: Cấu trúc block 10](#_Toc27079582)

[Hình 1- 9: Chuỗi liên kết trong blockchain 10](#_Toc27079583)

[Hình 1- 10: Cấu trúc một block trong code 10](#_Toc27079584)

[Hình 1- 11: Cây Merkle liên kết với nhau bằng mã Hash của cây trước đó 11](#_Toc27079585)

[Hình 1- 12: Cấu trúc của một khối dữ liệu 11](#_Toc27079586)

[Hình 1- 13: Mô phỏng một liên kết trong blockchain 12](#_Toc27079587)

[Hình 1- 14: Mô hình Proof-Of-Work 13](#_Toc27079588)

[Hình 1- 15: Mạng phân bố Blockchain 13](#_Toc27079589)

[Hình 1- 16: Mạng blockchain với hai nhánh song song cùng tồn tại 14](#_Toc27079590)

[Hình 1- 17: Blockchain công nhận nhánh dài hơn 15](#_Toc27079591)

[Hình 1- 18: Blockchain công nhận nhánh có khối lượng tính toán lớn nhất 16](#_Toc27079592)

[Hình 2- 1: Thiết kế chi tiết hệ thống tổ chức thi 21](file:///C:\Users\minhvu\Desktop\Đồ_án_vunm.docx#_Toc27079594)

[Hình 2- 2: Kiến trúc đơn giản của một mạng Hyperledger Fabric 27](#_Toc27079595)

[Hình 2- 3: Sơ đồ bắt đầu một transaction 28](#_Toc27079596)

[Hình 2- 4: Format của dockerfile 29](#_Toc27079597)

[Hình 2- 5: Dockerfile xây dựng với base image ubuntu kết hợp với nginx 30](#_Toc27079598)

[Hình 2- 6: Docker compose file 31](#_Toc27079599)

[Hình 2- 7: Thiết kế database trong Mysql 33](#_Toc27079600)

[Hình 2- 8: Cơ sở dữ liệu trong mongodb 33](#_Toc27079601)

[Hình 2- 9: Replication trong mongodb 35](#_Toc27079602)

[Hình 2- 10: Cơ chế automatic failover trong mongo 36](#_Toc27079603)

[Hình 2- 11: Mô hình node trong kubernetes 38](#_Toc27079604)

[Hình 2- 12: Mô tả các node trong kubernetes 38](#_Toc27079605)

[Hình 2- 13: Mô hình pod trong kubernetes 39](#_Toc27079606)

[Hình 2- 14: pods trong node kubernetes 40](#_Toc27079607)

[Hình 2- 15: Mô hình deployment trong kubernetes 40](#_Toc27079608)

[Hình 2- 16: Mô hình mạng fabric 42](#_Toc27079609)

[Hình 2- 17: Mô hình client tương tác với mạng blockchain 43](#_Toc27079610)

[Hình 3- 1: Định dạng cơ sở dữ liệu trong mạng blockchain 44](#_Toc27079611)

[Hình 3- 2: Hàm init trong chaincode 44](#_Toc27079612)

[Hình 3- 3: Hàm invoke trong chaicode 44](#_Toc27079613)

[Hình 3- 4: Hàm lấy thông tin của một thí sinh 45](#_Toc27079614)

[Hình 3- 5: Hàm thêm mới thí sinh và thay đổi thông tin thí sinh 46](#_Toc27079615)

[Hình 3- 6: Những file và thư mục cần thiết trong mạng fabricblockchain 46](#_Toc27079616)

[Hình 3- 7: Mô tả cơ sở dữ liệu của thí sinh 47](#_Toc27079617)

[Hình 3- 8: Mô tả cơ sở dữ liệu đề thi 48](#_Toc27079618)

[Hình 3- 9: Mô hình hệ thống sử dụng kubernetes 49](#_Toc27079619)

[Hình 3- 10: Dockerfile cho code backend 50](#_Toc27079620)

[Hình 3- 11: Dockercompose file cho backend 50](#_Toc27079621)

[Hình 3- 12: File config kubernetes pod một cho backend 50](#_Toc27079622)

[Hình 3- 13: File config kubernetes pod hai cho backend 51](#_Toc27079623)

[Hình 3- 14: File service kubernetes cho backend 51](#_Toc27079624)

[Hình 3- 15: Kết quả tạo service backend 52](#_Toc27079625)

[Hình 3- 16: File config nginx.conf cho frontend 52](#_Toc27079626)

[Hình 3- 17: File dockerfile cho frontend 53](#_Toc27079627)

[Hình 3- 18: File dockercompose của frontend 53](#_Toc27079628)

[Hình 3- 19: Các pods của frontend và backend sau khi tạo 54](#_Toc27079629)

[Hình 3- 20: Service sau khi tạo bằng kubernetes 54](#_Toc27079630)

[Hình 3- 21: Dockerfile cho mongodb 55](#_Toc27079631)

[Hình 3- 22: Docker compose file cho mongodb 56](#_Toc27079632)

[Hình 3- 23: Kết quả replication mongodb 56](#_Toc27079633)

[Hình 3- 24: Giao diện phần đăng nhập 57](#_Toc27079634)

[Hình 3- 25: Giao diện phần xác thực thông tin 57](#_Toc27079635)

[Hình 3- 26: Giao diện hướng dẫn làm bài thi 58](#_Toc27079636)

[Hình 3- 27: Giao diện xác thực hai bước bằng otp 58](#_Toc27079637)

[Hình 3- 28: Giao diện chọn môn thi 58](#_Toc27079638)

[Hình 3- 29: Giao diện làm bài thi 59](#_Toc27079639)

[Hình 3- 30: Giao diện nộp bài thi 59](#_Toc27079640)

# DANH MỤC CÁC BẢNG

[Bảng 1- 1: Bảng Hash Puzzle 12](#_Toc27079641)

[Bảng 1- 2: Sự khác biệt giữa private blockchain và public blockchain 20](#_Toc27079642)

[Bảng 2- 1: So sánh mongodb và mysql 34](#_Toc27079643)

# MỞ ĐẦU

Vụ gian lận thi 2018 (hay còn gọi là Vụ gian lận thi Trung học phổ thông Quốc gia 2018) là vụ sai phạm trong công tác tổ chức, chấm thi ở [Kỳ thi THPT quốc gia](https://vi.wikipedia.org/wiki/K%E1%BB%B3_thi_trung_h%E1%BB%8Dc_ph%E1%BB%95_th%C3%B4ng_qu%E1%BB%91c_gia_(Vi%E1%BB%87t_Nam)) năm 2018. Vụ việc liên quan đến hàng loạt bài thi của thí sinh được nâng điểm ở các tỉnh [Hà Giang](https://vi.wikipedia.org/wiki/H%C3%A0_Giang), [Sơn La](https://vi.wikipedia.org/wiki/S%C6%A1n_La), [Hòa Bình](https://vi.wikipedia.org/wiki/H%C3%B2a_B%C3%ACnh). Vụ gian lận này được cho là nghiêm trọng nhất từ xưa đến nay, sau nhiều năm [Bộ Giáo dục và Đào tạo](https://vi.wikipedia.org/wiki/B%E1%BB%99_Gi%C3%A1o_d%E1%BB%A5c_v%C3%A0_%C4%90%C3%A0o_t%E1%BA%A1o) tổ chức các kỳ thi tuyển sinh và THPT Quốc gia.

Giải quyết vấn đề trên là một bài toán lớn đối với các nước đang phát triển như Việt Nam, Quy chế và cách thi luôn được xem xét và thay đổi nhưng cho đến nay thì tính hiều quả vẫn không cao. Một trong những giải pháp mang tính thời đại chính là áp dụng khoa học kỹ thuật tiên tiến, các hệ thống có tính tự động hóa cao, vừa đảm bảo được tính chính xác, vừa đáp ứng được tính toàn vẹn về điểm của các thí sinh. Tiết kiệm kinh phí tổ chức thi, cách thi đơn giản không gâp áp lực đến các thí sinh, và tính công bằng luôn luôn được đảm bảo. Một trong sơ đó có thể kể đến là công nghệ Blockchain, công nghệ này có thể giúp chúng ta quản lý thông tin về các thí sinh như: Họ tên, chứng minh nhân dân, địa chỉ quê quán và điểm thi …

Trong lĩnh vực thi cử này thì blockchain sẽ có ứng dụng trong việc lưu trữ, các thông tin cần thiết và quan trọng về thí sinh sẽ được lưu trữ trên blockchain giúp gia tăng đáng kể tính minh bạch và sẵn có của dữ liệu, giảm thiểu việc xử lý, lưu trữ, cũng như sửa đổi dữ liệu khi không được phép.

Trong phạm vi đồ án của mình, em xin trình bày hệ thống áp dụng công nghệ blockchain cho việc lưu trữ điểm và thông tin cả các thí sinh. Ứng dụng thi cử sử dụng công nghệ block chain được tách làm ba thành phần chính bao gồm phần giao diện là một trang web để làm bài thi, phần backend để tính điểm cho thí sinh và phần mạng lưới blockchain để lưu điểm và thông tin cần thiết của các thí sinh. Trong đó, phần backend sẽ bao gồm 2 phần là cơ sở dữ liệu: Thông tin đăng nhập, và đề thi…và phần code để lấy đề thi gửi lên giao diện làm bài và nhận kết quả sau đó tính điểm cho thí sinh, và gửi lên mạng blockchain.

Với mục tiêu như vậy, bố cục của đồ án sẽ bao gồm bốn chương theo cấu trúc như sau:

**Chương 1: Bài toán tránh gian lận điểm trong thi cử**

Trình bày mục tiêu và các thành phần trong bài toán tránh gian lận điểm trong thi cử.

**Chương 2: Giải pháp, thiết kế và xây dựng hệ thống**

Trình bày giải pháp đưa ra, các thiết kế và cách xây dựng hệ thống tổ chức thi, bao gồm các thành phần, cơ sở dữ liệu, mạng lưới, mô hình…

**Chương 3: Thực nghiệm và xây dựng ứng dụng demo**

Đánh giá hiệu quả của hệ thống rút ra nhận xét. Sau đó trình bày quá trình xây dựng ứng dụng demo.

**Kết luận và hướng phát triển**

Tổng kết, tóm tắt những kết quả đã đạt được và còn chưa đạt được. Từ đó đề xuất các hướng phát triển tiếp theo.

# 

# CHƯƠNG 1. ỨNG DỤNG BLOCKCHAIN TRONG THI CỬ

Chương này sẽ giới thiệu một cách tổng quan về mục tiêu, cấu trúc, đặc điểm, phạm vi của bài toán ứn dụng blockchain trong công tác thi cử.

* 1. Giới thiệu bài toán

Bài toán ứnng dụng blockchain trong công tác thi cử có nhiệm vụ chính là tổ chức một kì thi trên máy tính, các thí sinh sẽ là bài thi trên trang web, kết quả sẽ được chấm một cách tự động và sau đó lưu lên trên mạng blockchain. Khi thí sinh muốn biết điểm thì có thể tra cứu trên hệ thống, điểm sẽ được bảo đảm về tính toàn vẹn và minh bạch. Quy trình hoạt động của hệ thống gồm ba phần chính và được mô tả theo hình dưới đây.

Giao diện làm bài thi trắc nhiệm: gồm đănng nhập và phần làm bài thi

Backend gồm các phần: đăng nhập, lấy đề thi trong cơ sở dữ liệu, nhận bài làm và tính điểm

Mạng blockchain lưu thông tin thí sinh và điểm

Hình 1- 1: Sơ đồ tổng quát quát của hệ thống

* 1. Giới thiệu về công nghệ blockchain

Mặc dù đã xuất hiện và được đề cập đến từ giữa 2008 nhưng đến giờ Blockchain vẫn còn là một khái niệm xa lạ đối với nhiều người và chưa có nhiều ứng dụng thực tiễn ngoài việc được sử dụng trong giao dịch và thanh toán bằng tiền điện tử. Với việc phát triển nhanh chóng của thương mại điện tử, trí tuệ nhân tạo, thực tế ảo, Internet vạn vật, blockchain đang nổi lên là một công nghệ có sức mạnh đột phá làm thay đổi cách thức chúng ta trao đổi thông tin qua mạng Internet.

Blockchain (chuỗi khối), tên ban đầu block chain 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 rất khó và tốn nhiều tài nguyên để thay đổi được nó. Số lượng ký tự trong một biển số xe có thể là 6, 7, 8 hoặc 9; trong đó số lượng chữ cái có thể là 1,2 hoặc 3 ký tự. Chuỗi ký tự có thể được sắp xếp thành dạng biển dài (biển một hàng) và dạng biển ngắn (biển hai hàng).

Blockchain được đảm bảo nhờ cách thiết kế sử dụng hệ thống tính toán phân cấp với khả năng chịu lỗi byzantine cao. Vì vậy sự đồng thuận phân cấp có thể đạt được nhờ Blockchain. Vì vậy Blockchain phù hợp để ghi lại những sự kiện, hồ sơ y tế, xử lý giao dịch, công chứng, danh tính và chứng minh nguồn gốc. Việc này có tiềm năng giúp xóa bỏ các hậu quả lớn khi dữ liệu bị thay đổi trong bối cảnh thương mại toàn cầu.

Blockchain đầu tiên được phát minh và thiết kế bới [Satoshi Nakamoto](https://vi.wikipedia.org/wiki/Satoshi_Nakamoto) vào năm 2008 và được hiện thực hóa vào năm sau đó như là một phần cốt lõi của [Bitcoin](https://vi.wikipedia.org/wiki/Bitcoin), khi công nghệ blockchain đóng vai trò như là một cuốn sổ cái cho tất cả các giao dịch. Qua việc sử dụng mạng lưới ngang hàng và một hệ thống dữ liệu phân cấp, [Bitcoin](https://vi.wikipedia.org/wiki/Bitcoin) blockchain được quản lý tự động. Việc phát minh ra blockchain cho [Bitcoin](https://vi.wikipedia.org/wiki/Bitcoin) đã làm cho nó trở thành loại tiền tệ kỹ thuật số đầu tiên giải quyết được vấn đề double spending (chi tiêu gian lận khi 1 lượng tiền được dùng 2 lần). Công nghệ này của [Bitcoin](https://vi.wikipedia.org/wiki/Bitcoin) đã trở thành nguồn cảm hứng cho một loạt các ứng dụng khác

Công nghệ blockchain tương đồng với cơ sở dữ liệu, chỉ khác ở việc tương tác với cơ sở dữ liệu. Để hiểu blockchain, cần nắm được năm định nghĩa sau: chuỗi khối (blockchain), cơ chế đồng thuận phân tán đồng đẳng (Distributed), tính toán tin cậy (trusted computing), hợp đồng thông minh (smart contracts) và bằng chứng công việc (proof of work).  Mô hình tính toán này là nền tảng của việc tạo ra các ứng dụng phân tán

* + 1. Các khái niệm

Phần này sẽ giới thiệu và giải thích các khái niệm được đề cập tới trong một mạng lưới blockchain cơ bản.

Sổ cái phân tán

Sổ cái không phải là khái niệm mới, sổ sách kế toán kép đã được sử dụng từ thế kỷ 13. Khi được chia sẻ thì một khái niệm mới ra đời đó là *sổ cái phân tán*. Sổ cái là một cấu trúc dữ liệu bao gồm danh sách các giao dịch đã được sắp xếp. Trong blockchain, sổ cái được sao chép trên tất cả các nút. Hơn nữa, các giao dịch được nhóm trong các khối, các khối này sau đó được nối với nhau thành một chuỗi. Vì vậy, sổ cái phân tán cơ bản là một cấu trúc dữ liệu chỉ nối thêm được nhân rộng khắp các nút mạng trong hệ thống. Trong blockchain, sổ cái phân tán bắt đầu với một số trạng thái ban đầu và ghi lại toàn bộ các trạng thái xảy ra khi hệ thống hoạt động. Sổ cái phân tán trong hệ thống blockchain có các đặc điểm sau:

* Các nút trong mạng có quan hệ ngang hàng và đồng sở hữu sổ cái nên dữ liệu được bảo toàn.
* Chỉ cần một nút tồn tại thi toàn bộ dữ liệu có thể đồng bộ cho toàn mạng nhờ đó mà hệ thống luôn có tính sẵn sàng cao.
* Dữ liệu chỉ có giá trị khi được tất cả các nút đồng thuận.
* Các sổ cái giống y hệt nhau ở tất cả các nút.

**Block**

Mỗi block (khối) là một tập hợp các giao dịch được nhóm lại với nhau, bao gồm *block header* và *block body*. Kích thước khối tùy thuộc vào loại và thiết kế của blockchain được sử dụng. Cấu trúc của một khối cũng phụ thuộc vào loại và thiết kế của blockchain, phần block header bao gồm:

* Block Version: Phiên bản của khối.
* Merkle Tree Root Hash: Giá trị băm của tất cả các giao dịch trong khối được tính theo cây Merkle.
* Timestamp: Nhãn thời gian khởi tạo khối sử dụng định dạng UNIX.
* nBits: Ngưỡng mục tiêu của một mã băm khối hợp lệ.
* Nonce: Một giá trị có kích thước 4 byte, thường bắt đầu bằng 0 và thay đỗi trong mỗi lần tính mã băm.
* Parent Block Hash: Giá trị băm 256-bit của khối liền trước đó.

**Chuỗi khối ( The blockchain ) và dịch vụ chuỗi khối:**

Blockchain là cơ sở dữ liệu được lưu trữ phân tán trong các khối thông tin về các giao dịch được liên kết với nhau và mở rộng theo thời gian, các thông tin trên mạng lưới phân tán được mã hóa mà chỉ có người nắm giữ khóa cá nhân mới truy cập được. Mỗi khối thông tin giao dịch đều có tiền tố chứa thông tin về thời gian khởi tạo và được liên kết tới khối trước đó để chống lại việc thay đổi của dữ liệu (mỗi khi có sự thay đổi mã niêm phong sẽ bị sai lệch và hệ thống sẽ phát hiện ra hoặc sẽ mất liên kết); một khi dữ liệu đã được ghi thì sẽ không thể thay đổi được hoặc phải tốn một tài nguyên rất lớn để “đào” lại các khối giao dịch cho đến khi khối bị thay đổi.

Về công nghệ blockchain, đây là một dạng cụ thể của công nghệ sổ cái phân tán (DLT) - một cách thức lưu trữ và đồng bộ dữ liệu (là các khối dữ liệu chứa thông tin của các giao dịch, sau khi được niêm phong bằng thuật toán băm SHA256 sẽ được móc xích với nhau qua mã định danh của khối trước đó) giữa các thành viên/địa chỉ lưu trữ trong đó mỗi thành viên/địa chỉ lưu trữ đều giữ các bản lưu giống hệt nhau. Blockchain sử dụng phương pháp ký số và các thuật toán niêm phong số bằng hàm băm SHA256 nhằm tạo ra và xác thực các giao dịch và xác thực các khối giao dịch mới liên tục mở rộng, trong đó thông tin được thêm vào cấu trúc dữ liệu hiện có và không thể bị xóa bỏ - đây là việc hình thành nên các khối giao dịch (transaction blocks).

Hiểu một cách đơn giản, blockchain là một sổ cái kỹ thuật số được phân tán; sổ cái luôn được niêm phong và móc xích giữa các khối và đồng bộ cho những người tham gia vào mạng lưới. Tất cả các bản sao này được cập nhật khi dữ liệu hoặc giao dịch mới được ghi vào blockchain thông qua một thuật toán đồng thuận của tất cả mọi người tham gia. Đây là một hệ thống ngang hàng (P2P), giảm thiểu sự tham gia của các khâu trung gian, tăng cường an ninh, tính minh bạch và sự ổn định, vì vậy, giảm thiểu chi phí và lỗi do con người gây ra

**Smart contract (hợp đồng thông minh)**

[Smart Contract](https://blogtienao.com/smart-contract-hop-dong-thong-minh-la-gi/)**(Hợp Đồng Thông Minh)** là một thuật ngữ mô tả một bộ giao thức đặc biệt có khả năng tự động thực hiện các điều khoản, các thoả thuận giữa các bên trong hợp đồng (ở trường hợp này là các hệ thống máy tính) nhờ sự hỗ trợ của [**công nghệ Blockchain**](https://blogtienao.com/blockchain-la-gi/). Toàn bộ hoạt động của Smart Contract được thực hiện một cách tự động và không có sự can thiệp từ bên ngoài, hay thông qua một bên thứ ba trung gian. Những giao dịch được thực hiện bằng các hợp đồng thông minh rất minh bạch, có thể dễ dàng truy xuất được và không thể bị can thiệp hoặc đảo chiều. Các điều khoản trong Smart Contract tương đương với một hợp đồng có pháp lý và được ghi lại dưới ngôn ngữ của lập trình. Điểm nổi bật nhất của **Smart Contract** là cho phép hai bên tham gia thực hiện hợp đồng một cách chính xác, an toàn và nhanh chóng; mà không cần các bên biết nhau từ trước, cũng không cần phải gặp trực tiếp để có thể làm việc với nhau, hay một bên trung gian thứ ba mà chỉ cần có kết nối Internet. Khái niệm về Smart Contract được biết đến lần đầu tiên năm bởi Nick Szabo vào 1993. Tại thời điểm đó ông đã nêu ra những nguyên tắc hoạt động chính, nhưng cũng ở thời điểm ấy thì vẫn chưa có đủ công nghệ và môi trường thích hợp để hiện thực hóa nó. Thế nhưng mọi chuyện đã thay đổi với sự ra đời và phát triển của công nghệ Blockchain. Cơ chế hoạt động của các **Smart Contract**có thể nói là giống như với một chiếc máy bán hàng tự động. Có nghĩa là chúng chỉ tự động thực hiện những điều khoản đã được lập trình sẵn từ trước khi được điều khoản đó đã đáp ứng đủ những yêu cầu cần thiết. Đầu tiên, các điều khoản trong hợp đồng sẽ được viết bằng ngôn ngữ lập trình, sau đó được mã hóa và chuyển vào một block thuộc Blockchain. Sau khi chuyển vào block, Smart Contract này sẽ được phân phối và sao chép lại bởi các node đang hoạt động trên nền tảng đó. Sau khi có nhận lệnh triển khai thì hợp đồng sẽ được triển khai theo đúng như điều khoản định sẵn. Đồng thời, Smart Contract cũng sẽ tự động kiểm tra quá trình thực hiện những cam kết, điều khoản được nêu trong hợp đồng. **Smart Contract** là một ứng dụng tận dụng tất cả những điểm mạnh của công nghệ Blockchain mang lại vì vậy nó có rất nhiều lợi ích, dưới đây là các lợi ích chính của nó: Tự động hóa, không bị thất lạc, an toàn, tốc độ, tiết kiệm, chính xác.

Tất cả các mô hình blockchain đều có hợp đồng thông minh được tích hợp trong các giao dịch. Ví dụ trong các loại tiền mã hóa, hợp đồng thông minh xác minh các giao dịch đầu vào bằng cách kiểm tra chữ ký của chúng. Sau đó, xác minh số dư của các địa chỉ gửi có khớp với địa chỉ nhận không. Cuối cùng, hợp đồng thông minh sẽ áp dụng các thay đổi tới các trạng thái. Người dùng cũng có thể viết các hợp đồng thông minh thực thi trong giao dịch.



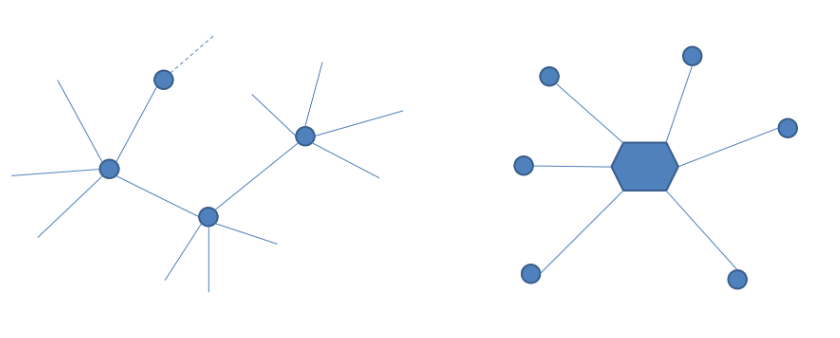
Hình 1- 2:Hợp đồng thông minh trong fabric fabcar

Cơ chế đồng thuận phân tán đồng đẳng:

Cơ chế đồng thuận phân tán đồng đẳng (hay còn gọi là cơ chế đồng thuận ngang hàng) là cơ chế ngược lại với mô hình cổ điển về cơ chế đồng thuận tập trung – nghĩa là khi một cơ sở dữ liệu tập trung được dùng để quản lý việc xác thực giao dịch. Một sơ đồ phân tán đồng đẳng chuyển giao quyền lực và sự tin tưởng cho một mạng lưới phân tán đồng đẳng và cho phép các nút của mạng lưới đó liên tục lưu trữ các giao dịch trên một khối (block) công cộng, tạo nên một chuỗi (chain) độc nhất: chuỗi khối (blockchain). Mỗi khối kế tiếp chứa một mã “hash” (một dấu tay độc nhất) của mã trước nó; vì thế mã hóa (thông qua hàm hash) được sử dụng để bảo đảm tính xác thực của nguồn giao dịch và loại bỏ sự cần thiết phải có một trung gian tập trung. Sự kết hợp của mã hóa và công nghệ blockchain lại đảm bảo rằng sẽ không bao giờ một giao dịch được lưu trữ lại hai lần. Khi có một chuối được thêm vào block thì tất cả các nút trên mạng sẽ tham gia vào quá trình đồng thuận vì vậy nó được gọi là đồng đẳng.

**Mạng phân bố ngang hàng (distributed peer-2-peer) và mạng tập trung (centralized network):**

Một mạng phân bố ngang hàng peer-2-peer (p2p) bao gồm rất nhiều các nốt mạng liên kết trực tiếp với nhau, trong đó mỗi nốt mạng có vai trò và vị trí như nhau. Một ví dụ điển hình là Bittorrent, giao thức chia sẻ file qua Internet, hoặc Napster, ứng dụng chia sẻ nhạc trực tuyến những năm 1990s. Mạng ngang hàng cho phép hàng triệu người sử dụng có thể kết nối trực tiếp với nhau, hình thành một khối khổng lồ với khả năng tính toán và băng thông của tất cả mạng tham gia gộp lại. Khác với mạng ngang hàng, mạng tập trung là mạng trong đó các nốt mạng cùng liên kết đến một đơn vị trung tâm, có toàn quyền quản lý việc lưu trữ và trao đổi thông tin. Điều này có thể được nhìn thấy ở bất cứ một hệ thống giao dịch tài chính, bảo hiểm hay hợp đồng đầu tư nào. Tất cả thông tin sẽ do đơn vị trung tâm nắm giữ và xử lý, tạo ra một chi phí không nhỏ cho việc duy trì, kiểm soát thông tin. Trong mạng P2P, do không tồn tại một đơn vị trung tâm kiểm soát thông tin, độ tin cậy của mạng này phụ thuộc vào độ tin cậy của từng nốt mạng. Do đó đặt ra phải có một giải pháp mã hóa thông tin đảm bảo được tính bền vững của các giao dịch và quản lý dữ liệu ngay cả khi không có được sự trung thực của tất cả các nốt mạng. Blockchain được đề cập đến như một giải pháp như thế.



Hình 1- 3: Mạng phân bố P2P Network vs Centralized Network

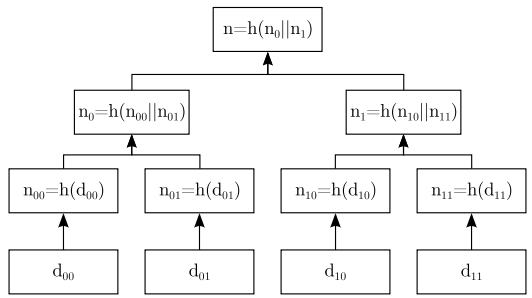
**Hàm hash:**

Hàm Hash là một hàm mã hóa dữ liệu với dung lượng bất kỳ để thu được một kiểu dữ liệu có độ dài nhất định. Dữ liệu sau khi bị mã hóa đảm bảo không thể tạo ngược lại dữ liệu ban đầu. Mỗi dữ liệu đầu vào khác nhau sẽ tạo ra một đoạn mã Hash khác nhau mà không rất khó đoán định. Kết quả Hash của cùng một dữ liệu ban đầu sẽ luôn cho ra cùng một đoạn mã Hash duy nhất. Hash Sha-256 là hàm mã hóa dữ liệu trong đó kết quả hash là một chuỗi số gồm 256 bits. Với tính chất như vậy, hàm hash được dùng để chứng minh tính nguyên vẹn của dữ liệu ban đầu. Sau khi dữ liệu được hash lại, thì việc thay đổi bất kỳ giá trị nào trong dữ liệu đó sẽ sinh ra một giá trị Hash mới. Việc kiểm tra kết quả của hàm hash hiện tại với giá trị hash ban đầu sẽ kết luận được dữ liệu có bị thay đổi không trong quá trình gửi và nhận thông tin.

**Cây Merkle:**

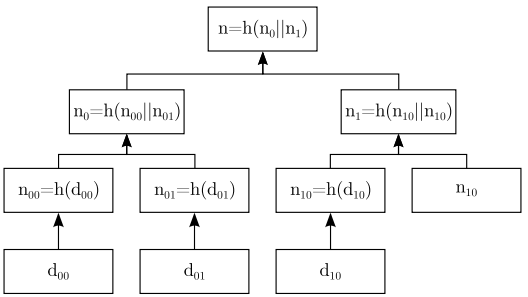
Trong blockchain, cây băm nhị phân được sử dụng để kiểm tra tính toàn vẹn của toàn bộ các giao dịch có trong block chỉ với một mã băm gốc. Thuật toán băm mật mã được sử dụng trên các cây merkle của bitcoin là SHA-256 được áp dụng hai lần. Dung lượng của nơi lưu trữ mã băm này là 32 byte và được lưu trữ trong tiêu đề block.

Cây nhị phân Merkle là cây băm nhị phân được dùng để xác minh tính toàn vẹn dữ liệu. Dữ liệu ở các lá trong cây Merkle được tính giá trị băm, trong khi các nút lá ở nhánh trên được tính giá trị băm bằng cách nối và tính giá trị băm của hai lá tương ứng bên dưới. Hình dưới đây mô tả cấu trúc cây Merkle có số lá chẵn.



Hình 1- 4: Cây Merkle với lá chẵn

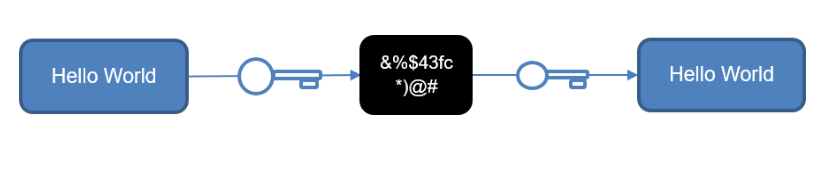
Trong trường hợp số nút lá là số lẻ thì nút lá cuối cùng sẽ được nhân đôi và tính giá trị băm.



Hình 1- 5: Cây Merke với lá số lẻ

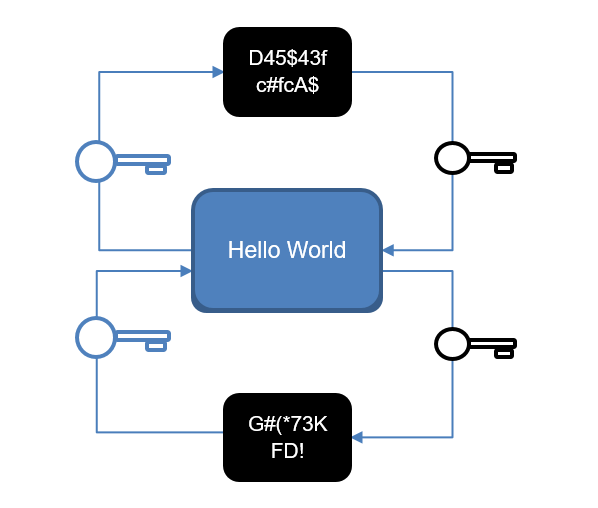
**Mã hóa đối xứng và bất đối xứng(Symmetric/Asymmetric Cyptography):**

Mã hóa đối xứng sử dụng một khóa duy nhất cho việc mã hóa và giải mã dữ liệu:



Hình 1- 6: Mã hóa đối xứng

Mã hóa bất đối xứng hay còn gọi là mã hóa công khai – bí mật, bao gồm một cặp khóa trong đó một khóa sẽ dùng để mã hóa và phải dùng khóa còn lại mới có thể giải mã; Cả hai khóa đều có thể dùng để mã hóa nhưng cần đến khóa còn lại để giải mã.



Hình 1- 7: Mã hóa bất đối xứng với khóa công khai và khóa bí mật

Giả sử Bob cần gửi một thông điệp cho Alice và trước đó, Bob và Alice đã trao đổi với nhau khóa công khai mình. Bob sẽ mã hóa thông điệp của mình sử dụng khóa công khai của Alice và gửi cho Alice, Alice nhận được thông điệp đã bị mã hóa đấy sẽ sử dụng khóa bí mật của mình để giải mã và đọc nội dung bên trong. Để gửi lại thông điệp cho Bob, Alice cũng làm tương tự, mã hóa bằng khóa công khai của Bob, và Bob sẽ sử dụng khóa bí mật của mình để giải mã. Tuy nhiên một nhân vật trung gian có thể thay đổi nội dung và mã hóa dữ liệu sử dụng một trong hai khóa công khai của hai người. Để chắc chắn rằng thông điệp mà Alice nhận được là thuộc về Bob và không phải do một nhân vật trung g mạo danh Bob gửi đến, thì sau khi mã hóa thông điệp của mình bằng khóa công khai của Alice, Bob sẽ Hash toàn bộ thông điệp đã được mã hóa đấy thành một chuỗi ký tự, sau đó sử dụng khóa bí mật của mình để mã hóa thêm lần nữa chuỗi ký tự đấy, kết quả là một chuỗi số được gọi là chữ ký điện tử của Bob; Bob sẽ đính kèm chữ ký điện tử này trong thông điệp đã được mã hóa bằng khóa công khai của Alice và gửi đi.

Alice sau khi nhận được thông điệp đã được mã hóa và chữ ký điện tử của Bob kèm theo, sẽ giải mã thông điệp bằng khóa bí mật của mình, đồng thời Alice giải mã chữ ký điện tử của Bob với khóa công khai của Bob để thu được chuỗi kỹ tự đã Hash. Sau đó Alice chỉ việc thực hiện lại việc Hash thông điệp đã được giải mã và so sánh nó với giá trị Hash từ chữ ký điện tử của Bob để chắc chắn kết quả thu được là trùng khớp. Do chỉ có Bob mới giữ khóa bí mật, nên có thể công nhận là chỉ Bob mới tạo ra chữ ký điện tử này, và rõ ràng thông điệp trên là do Bob gửi đến.

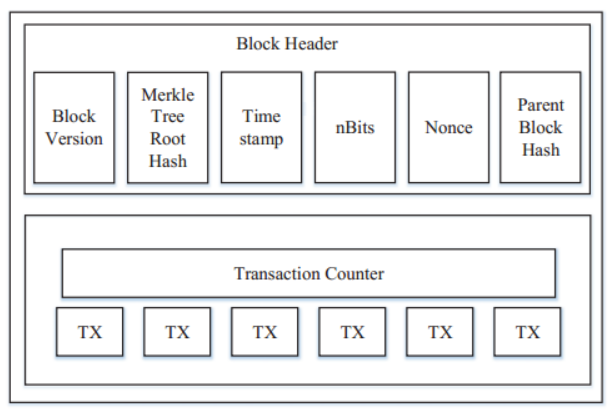
* + 1. Cấu trúc

**Blockchain** là cấu trúc chuỗi các block. Mỗi block trong blockchain được xác định bằng một mã băm do thuật toán băm mật mã SHA256 tạo ra trong tiêu đề block (block header). Mỗi block cũng tham chiếu đến một block trước đó gọi là block cha, thông qua một trường có tên “Block Hash trước” trong tiêu đề block. Nói cách khác, mỗi block đều chứa mã băm của block trước đó trong tiêu đề của chính nó. Chuỗi các mã băm liên kết từng block tới block trước tạo thành một chuỗi mắt xích đi ngược trở về cho tới block được tạo ra đầu tiên, gọi là block gốc (genesis block). Cấu trúc của một khối cũng phụ thuộc vào loại và thiết kế của blockchain, phần block header bao gồm:

* Block Version: Phiên bản của khối.
* Merkle Tree Root Hash: Giá trị băm của tất cả các giao dịch trong khối theo cây merkle.
* Timestamp: Nhãn thời gian tạo khối sử dụng định dạng UNIX.
* nBits: Ngưỡng mục tiêu của một mã băm khối hợp lệ.
* Nonce: Một giá trị có kích thước 4 byte, thường bắt đầu bằng 0 và thay đỗi trong mỗi lần tính mã băm, là đáp án để mã băm của block header thỏa mãn độ khó hiện thời.
* Parent Block Hash: Giá trị băm 256-bit của khối liền trước đó.

Phần block body bao gồm các giao dịch trong khối. Số lượng giao dịch tối đa trong một khối phụ thuộc vào kích thước khối và kích thước của mỗi giao dịch.

Mỗi block chỉ có một block trỏ đến, có thể có trường hợp rẽ nhánh, 2 block cùng trỏ đến 1 block trước nhưng đến một thời điểm có 1 nhánh dài hơn và khi đó nhánh ngắn hơn sẽ phải xóa bỏ và dữ liệu trên nhánh này phải rollback trở lại.



Hình 1- 8: Cấu trúc block

Chuỗi liên kết

Một chuỗi, hay một danh sách liên kết, được tạo ra khi mỗi đơn vị dữ liệu đồng thời lưu trữ một giá trị Hash của đơn vị dữ liệu trước đó. Chỉ cần có một thay đổi trong dữ liệu trước đó, thì giá trị hash sẽ bị thay đổi và mối liên kết đó sẽ bị phá vỡ. Một cấu trúc như vậy phù hợp cho việc lưu trữ và liên kết các khối dữ liệu không xuất hiện đồng thời mà xảy ra lần lượt từng thời điểm.



Hình 1- 9: Chuỗi liên kết trong blockchain

Cấu trúc của một đơn vị được mô tả như sau:

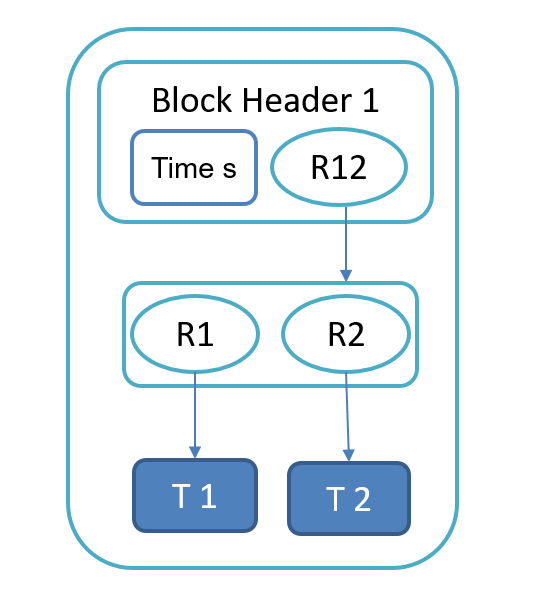
|  |
| --- |
| type Block struct {  Hash []byte  PrevBlockHash []byte  Data []byte  Timestamp int64  } |

Hình 1- 10: Cấu trúc một block trong code

Trên hình, mã Hash R3 có thể dùng để truy xuất ngược lại toàn bộ dữ liệu trong chuỗi đến giá trị đầu tiên. R3 được gọi là đỉnh của chuỗi. Kết hợp với mô hình cây Merkle, một chuỗi các giao dịch có thể được liên kết lại với nhau theo trình tự thời gian diễn ra. Ở hình 2, có thể coi các khối dữ liệu Tx là các giao dịch diễn ra đồng thời tại một thời điểm và tạo thành một cây Merkle.

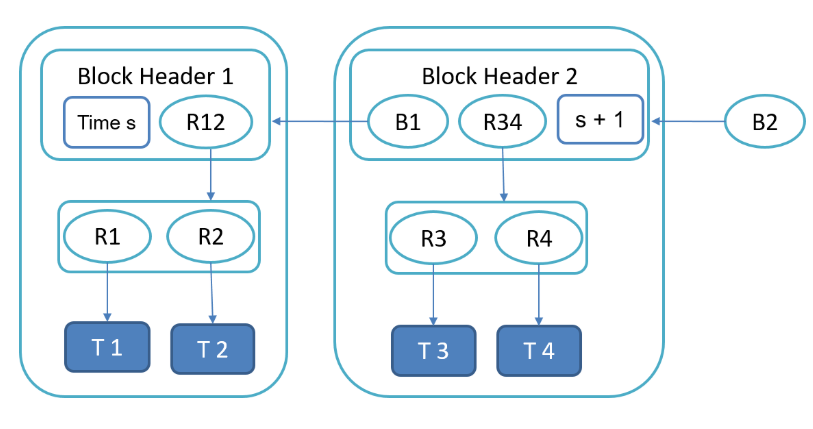


Hình 1- 11: Cây Merkle liên kết với nhau bằng mã Hash của cây trước đó



Hình 1- 12: Cấu trúc của một khối dữ liệu

Hình trên mô phỏng một khối dữ liệu bao gồm phần thân là một cây Merkle chứa liên kết đến các giao dịch diễn ra trong cùng một thời điểm và phần đỉnh chứa rễ của cây merkle cùng với hàm thời gian. Các khối tiếp kế tiếp theo trình tự thời gian khi được tạo ra sẽ gắn kèm với liên kết hash của khối trước đó. Toàn bộ cấu trúc hình thành nên một chuỗi khối hay còn gọi là blockchain.



Hình 1- 13: Mô phỏng một liên kết trong blockchain

**Bài toán tìm mã Hash phù hợp (proof-of-work hay Hash Puzzle):**

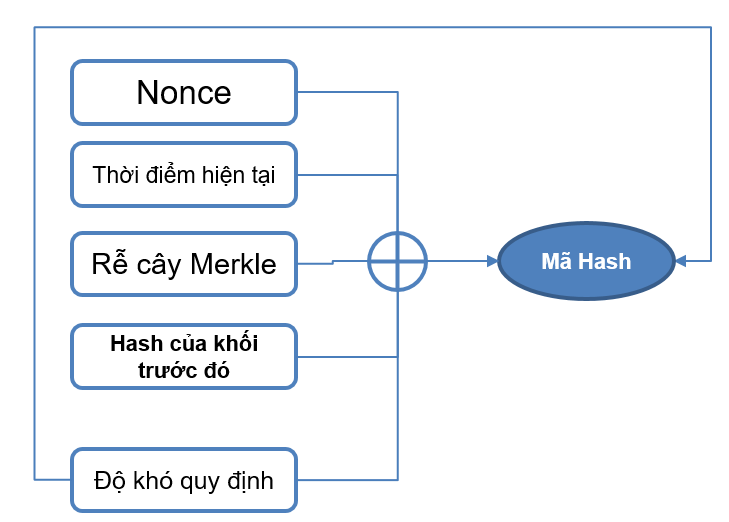
Khi mã hóa dữ liệu về một chuỗi ký tự hash, công đoạn này tiêu tốn một lượng thời gian tính toán của PC. Do đó hàm Hash có thể được một máy tính sử dụng để bắt một máy tính khác tính toán kết quả hash sao cho thỏa mãn một đề bài đặt ra.

Giả sử Bob quản lý một server hòm thư điện tử; Để tránh việc một địa chỉ gửi thư rác SPAM vào hòm thư của mọi người, Server của Bob áp đặt mọi hòm thư khi gửi email phải tìm ra một giá trị hash bất kỳ thỏa mãn điều kiện sau: Cho một số Nonce bất kỳ kết hợp với nội dung của email và tính toán mã Hash. Yêu cầu tìm ra số Nonce sao cho kết quả mã hóa Hash được một chuỗi ký tự có ít nhất 3 số 0 ở đầu. Bảng dưới mô tả quả trình tính toán của ví dụ trên:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nonce** | **Data** | **Data || Nonce** | **Hash Result** |
| 000 | HelloWord | HelloWorld 000 | 4EE4B774 |
| 001 | HelloWord | HelloWorld 001 | 3345B9A3 |
| 002 | HelloWord | HelloWorld 002 | 72040842 |
| **....** | **....** | **...** | **...** |
| 613 | HelloWord | HelloWord 613 | E8639001 |
| 614 | HelloWord | HelloWord 614 | 00068A3C |

Bảng 1- 1: Bảng Hash Puzzle

Hàm Hash đảm bảo rằng không có một giải thuật hay phương pháp nào khác để đi đến kết quả như yêu cầu nhanh hơn ngoại trừ việc phải tính toán toàn bộ các trường hợp có thể. Tùy độ khó của đề bài mà công đoạn này đòi hỏi khoảng thời gian khác nhau. Ở ví dụ trên, Server của Bob bắt buộc mọi tài khoản khi gửi email phải tính toán được mã Hash thỏa mãn đề bài và tìm được Nonce tương ứng với địa chỉ email nhận. Tiến trình này có thể mất vài chục giây, như vậy ý định SPAM thư cho một loạt tài khoản sẽ tiêu tốn một lượng lớn thời gian khiến cho công việc đấy không còn nhiều giá trị thực hiện. Blockchain ứng dụng một cách thức tương tự khi xây dựng các khối mới để gắn vào toàn bộ chuỗi. Khi giao dịch mới xuất hiện, một khối mới sẽ được xây dựng dựa trên giá trị hash của khối trước đó, rễ của cây Merkle liên kết đến các giao dịch mới và thời điểm ở hiện tại. Một khối mới chỉ được chấp nhận vào chuỗi khi mà kết quả mã Hash của nó thỏa mãn một lượng đủ lớn các bits 0 ở đầu. (Trong Bitcoin số lượng này là 32-72 số 0 đầu tiên trong số 256 kí tự của Sha-256)

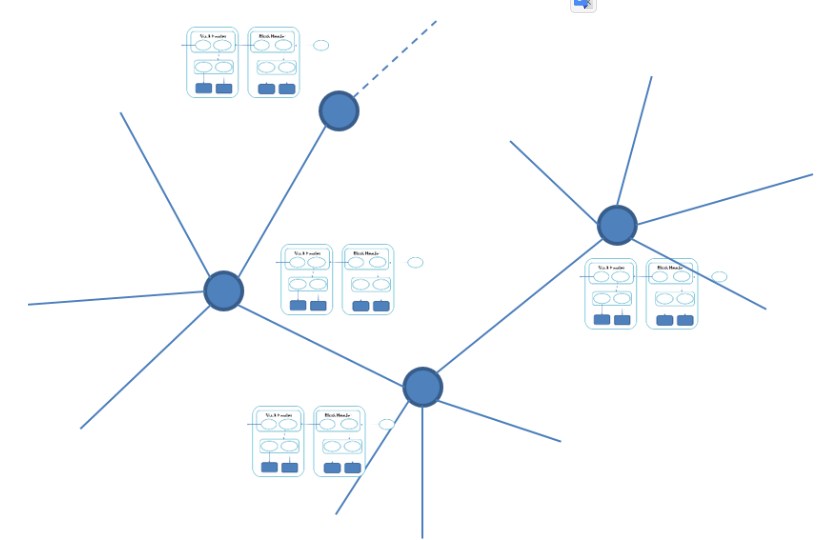


Hình 1- 14: Mô hình Proof-Of-Work

Tiến trình tìm mã hash này được gọi là proof-of-work[][1] và được các máy tính tham gia quá trình thực hiện gọi nôm na là “đào”/”cày” hash.

Tính bền vững của chuỗi liên kết

Trong mạng phân bố ngang hàng P2P, mỗi nốt mạng được coi là một máy tính hay một tài khoản cá nhân được thể hiện bằng một khóa công khai (public key) và giữ bí mật khóa cá nhân (private key). Mỗi nốt mạng không chia sẻ một chuỗi khối dữ liệu chung, mà mỗi nốt sở hữu một bản sao của toàn bộ chuỗi khối do toàn bộ mạng tạo ra. Khi một nốt mới tham gia vào mạng, nó sẽ phải đồng bộ dữ liệu về chuỗi khối với toàn bộ mạng bằng việc gửi và nhận từng khối từ các nốt mạng khác liên kết với nó.

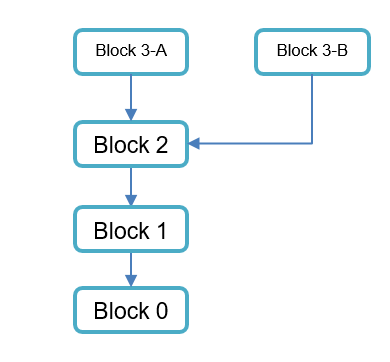


Hình 1- 15: Mạng phân bố Blockchain

Khi một khối mới được tạo ra từ một nốt trong mạng, nó sẽ truyền đi thông tin của khối mới cho tất cả các nốt mạng xung quanh. Khối mới phải thỏa mãn tiến trình proof-of-work do hệ thông quy định. Nhiệm vụ của các nốt xung quanh là kiểm chứng lại kết quả mã hóa Hash của khối mới và sau đó gắn khối mới vào bản sao chuỗi khối của mình nếu như kết quả hash thỏa mãn. Khi khối mới được các nốt mạng xung quanh chấp thuận, nốt đã tạo ra khối mới sẽ nhận được một khoản phí (thường là hoa hồng của giao dịch trong khối đó); Ngược lại khi khối mới bị phát hiện có xung đột với chuỗi khối của các nốt mạng còn lại, khối mới sẽ bị loại bỏ.Việc có một khoản thù lao như vậy khiến các nốt cùng phải chạy đua với nhau trong việc xây dựng khối mới và quản lý việc các khối do các nốt khác tạo ra phải phù hợp và không có xung đột với chuỗi hiện tại.

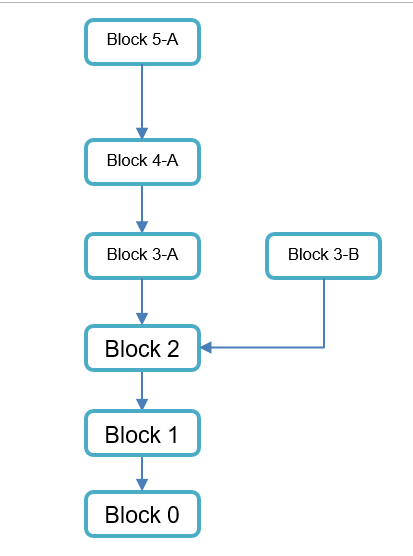
Xác định một chuỗi khối thống nhất

Do mỗi nốt mạng sở hữu một bản sao của chuỗi khối (hay còn gọi là cuốn sổ cái) về giao dịch của toàn bộ mạng, và cùng tham gia vào việc tạo thêm khối mới từ các giao dịch xảy ra, nên tại một thời điểm có thể xuất hiện cùng lúc hai khối mới cùng thỏa mãn độ khó của proof-of-work. Mạng blockchain sẽ chia thành hai phần với một số nốt mạng phát triển tiếp theo một nhánh trong khi một số nốt khác phát triển theo thánh còn lại.



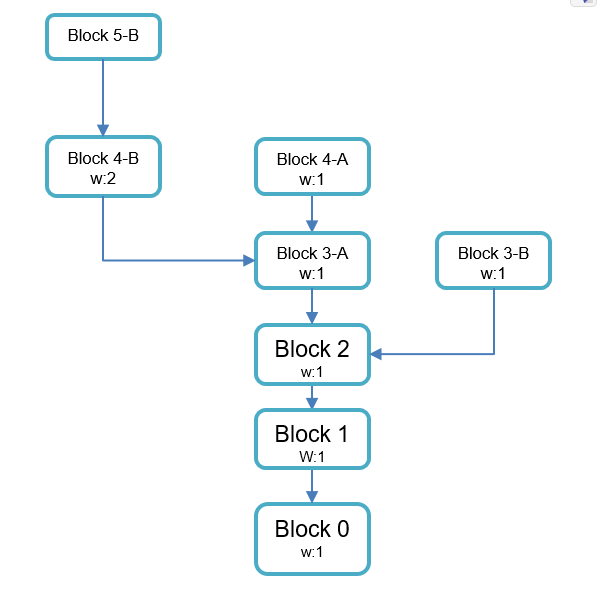
Hình 1- 16: Mạng blockchain với hai nhánh song song cùng tồn tại

Hình trên mô tả một blockchain với hai nhánh song song cùng tồn tại. Sẽ có một số nốt phát triển theo hướng 0 đến 3-A và đi tiếp, trong khi số khác đi theo của 3-B. Tại đây, giao dịch trong các khối 3-A và 3-B sẽ chưa được công nhận, do cả hai đều được coi là đỉnh của blockchain. Các nốt phát triển theo 3-A và 3-B tiếp tục tạo ra khối mới và nối vào với đỉnh của mình. Đến thời điểm tiếp theo, khi mà nhánh 3-A tạo được khối mới gắn tiếp vào trước và thông báo đến được hơn 51% nốt mạng, thì tất cả sẽ cùng chuyển sang đi theo nhánh 3-A để đi tiếp. 3-B sẽ trở thành nhánh cụt, và giao dịch nằm trong 3-B sẽ được các nốt mạng tính toán lại Hash để xây dựng khối mới.



Hình 1- 17: Blockchain công nhận nhánh dài hơn

Ở trên là cách thức Blockchain quy ước nhánh nhiều chuỗi khối nhất là nhánh có các giao dịch hợp lệ được chấp nhận. Ngoài ra cũng có trường hợp Blockchain sử dụng nhánh mà có tổng độ khó của thuật toán tìm Hash (proof-of-work) cao nhất làm nhánh hợp lệ. Ở hình dưới, giả sử tất cả các nhánh có độ khó là 1 (tương ứng 10 ký tự 0 ở đầu), riêng nhánh 4-B có độ khó bằng 2, tổng độ khó từ 0 đến 4-B sẽ cao nhất, và được các nốt chọn làm nhánh hợp lệ và đi tiếp.



Hình 1- 18: Blockchain công nhận nhánh có khối lượng tính toán lớn nhất

Chú ý là các khối 0 đến 2 nằm sâu trong cùng và gần như không còn thay đổi sau khi các khối tiếp theo được gắn tiếp vào, các khối 0 đến 2 sẽ trở thành cố định, các giao dịch trong khối này sẽ được công nhận là chính xác. Tất cả các nốt mạng khi tham gia vào Blockchain đều phải tuân thủ quy tắc này, bởi nếu không thì không được các nốt khác công nhận khối mới do nốt đó tạo ra, và như vậy giao dịch không được thực hiện cũng như không nhận được phí hoa hồng do quá trình “cày” mã Hash tạo ra.

* + 1. Đặc điểm và ứng dụng

Những ưu điểm chính của blockchain:

* Không thể làm giả, không thể phá hủy các chuỗi Blockchain: theo như lý thuyết thì chỉ có máy tính lượng tử mới có thể giải mã Blockchain và công nghệ Blockchain biến mất khi không có internet trên toàn cầu
* Bất biến: dữ liệu trong Blockchain không thể sửa (có thể sửa nhưng sẽ phải tốn rất nhiều tài nguyênđể lại dấu vết ) và sẽ lưu trữ mãi mãi
* Bảo mật: Các thông tin, dữ liệu trong Blockchain được phân tán và an toàn tuyệt đối
* Minh bạch: Ai cũng có thể theo dõi dữ liệu trong Blockchain đi từ địa chỉ này sang địa chỉ khác và có thể thống kê toàn bộ lịch sử trên địa chỉ đó
* Hợp đồng thông minh: là hợp đồng kỹ thuật số được nhúng vào đoạn code if-this-then-that (IFTTT), cho phép chúng tự thực thi mà không cần bên thứ ba

Nhược điểm của blockchain

Đối với hệ thống blockchain công cộng thì chi phí về cho việc xác thực giao dịch đang ngày càng tăng cao và công việc này tốn rất nhiều tài nguyên cho việc tính toán. Tính dư thừa dữ liệu do dữ liệu trong blockchain được sao chép và lưu trữ tại nhiều nơi, từ đó tốn nhiều không gian và tăng chi phí cho vấn đề lưu trữ dữ liệu. Với mô hình blockchain riêng tư và liên kết thì có thể hạn chế được vấn đề này khi thiết kế hệ thống.

Việc bảo trì hệ thống khó khăn hơn so với công nghệ lưu trữ tập trung. Với một cơ sở dữ liệu tập trung thì chỉ cần ghi dữ liệu ở một vài nơi, với blockchain thì cần ghi ở rất nhiều nơi thì cơ sở dữ liệu mới đảm bảo tính nhất quán. Việc thay đổi hay sửa chữa cơ sở dữ liệu trong một hệ thống tập trung cũng dễ dàng và nhanh chóng hơn. Trong blockchain cần phải có sự đồng thuận trong hệ thống mới có thể thay đổi cơ sở dữ liệu.

Việc bảo trì hệ thống khó khăn hơn so với công nghệ lưu trữ tập trung. Với một cơ sở dữ liệu tập trung thì chỉ cần ghi dữ liệu ở một vài nơi, với blockchain thì cần ghi ở rất nhiều nơi thì cơ sở dữ liệu mới đảm bảo tính nhất quán. Việc thay đổi hay sửa chữa cơ sở dữ liệu trong một hệ thống tập trung cũng dễ dàng và nhanh chóng hơn. Trong blockchain cần phải có sự đồng thuận trong hệ thống mới có thể thay đổi cơ sở dữ liệu.

Ứng dụng của blockchain vào thực tiễn

Công nghệ Blockchain có thể thay đổi nhiều hệ thống mà chúng ta gặp phải trong cuộc sống hàng ngày, chẳng hạn như dịch vụ tài chính, chính phủ và y tế. Với việc không đòi hỏi phải có sự tín thác và chi phí tốn kém để đảm bảo bảo mật, blockchain mang lại hiệu quả đáng kể. Hơn nữa, mạng lưới phi tập trung có thể được cấu hình để trở thành một cơ sở dữ liệu minh bạch, có thể được nhìn thấy bởi tất cả những người tham gia vào mạng lưới. Theo cách này, công nghệ blockchain cho phép tạo một bản ghi phân tán nhưng đảm bảo tính thống nhất. Điều này mang đến cơ hội cải thiện hiệu suất và bảo mật cho nhiều ngành công nghiệp và tổ chức (ví dụ: từ thiện, chuỗi cung ứng, chăm sóc sức khỏe,…).

* Từ thiện: Nhiều tổ chức từ thiện trên thế giới đang gặp các thách thức trong việc quản lý tài nguyên, minh bạch hoạt động và quản trị hiệu quả. Công nghệ blockchain có thể giúp các tổ chức này tối ưu hóa quá trình nhận và quản lý quỹ tiền. Chúng ta có một số ví dụ nổi bật về việc tích hợp công nghệ blockchain vào công tác từ thiện. Chẳng hạn, Quỹ từ thiện Blockchain (BCF) là một tổ chức phi lợi nhuận hoạt động hướng tới các mục tiêu phát triển bền vững nhằm chống lại nghèo đói và bất bình đẳng bằng cách thúc đẩy việc triển khai blockchain vào hoạt động từ thiện trên toàn thế giới.
* Chuỗi cung ứng: Hầu hết các mạng lưới chuỗi cung ứng đang gặp nhiều trở ngại liên quan đến tính minh bạch và hiệu quả. Hệ thống quản lý hiện tại vẫn phụ thuộc vào sự tín thác và sự tích hợp giữa công ty và các bên liên quan trong chuỗi cung ứng là còn rất xa vời. Công nghệ blockchain có thể được sử dụng để theo dõi toàn bộ quá trình tạo và phân phối vật liệu trong mạng lưới chuỗi cung ứng. Một cơ sở dữ liệu phân tán có thể phù hợp để ghi lại an toàn mọi dữ liệu liên quan, đảm bảo tính xác thực của các sản phẩm cũng như tính minh bạch trong thanh toán và vận chuyển.
* Chăm sóc sức khỏe: Tình trạng nghẽn cổ chai trong vận hành, lỗi dữ liệu và quan liêu là các vấn đề nổi cộm đối với lĩnh vực chăm sóc sức khỏe. Blockchain được ứng dụng vào một số trường hợp trong lĩnh vực chăm sóc sức khỏe, bao gồm công tác theo dõi thuốc thông qua chuỗi cung ứng và quản lý dữ liệu bệnh nhân. Hơn nữa, blockchain có thể mang lại lợi ích bảo mật đáng kể cho dữ liệu của các bệnh viện vì các tổ chức này thường bị tin tặc tấn công bởi giá trị cao và tính đáng tin cậy cao của các dữ liệu này. Các công ty đang khám phá việc sử dụng blockchain như một cách để lưu trữ hồ sơ sức khỏe kỹ thuật số. Các giải pháp này có thể giúp giảm chi phí tổng thể, đồng thời tăng cường tính riêng tư và chính xác của dữ liệu.
* Tiền bản quyền: Các nhạc sĩ, nhà làm game video và các nghệ sĩ nói chung thường xuyên phải đấu tranh để thu về tiền bản quyền họ xứng đáng được nhận do tình trạng vi phạm bản quyền kỹ thuật số, mối quan hệ không công bằng với các bên thứ ba hoặc đơn giản là do không được trả tiền bản quyền. Công nghệ blockchain có thể được sử dụng để tạo ra một nền tảng giúp những nghệ sĩ sáng tạo này có một hồ sơ bất biến và minh bạch ghi chép những đối tượng đang thuê, mua và/hoặc sử dụng nội dung sáng tạo của họ. Một nền tảng như vậy cũng có thể tạo điều kiện cho việc thanh toán thông qua các hợp đồng thông minh – về cơ bản, đây là các hợp đồng kỹ thuật số tự thực thi.
* Quản trị: Công nghệ Blockchain có tiềm năng cải thiện đáng kể công tác quản trị trong các lĩnh vực khác nhau. Bằng cách quản lý các mạng lưới và các hoạt động vận hành theo hướng dân chủ hóa, công bằng và an toàn hơn, các hệ thống dựa trên blockchain có thể được triển khai như một công cụ để loại bỏ tình trạng gian lận phiếu bầu và tăng niềm tin trong các cuộc bầu cử hoặc các quy trình lập hiến khác. Chúng cũng có thể được sử dụng như một vũ khí mạnh mẽ chống tham nhũng, tăng cường tính toàn vẹn của dữ liệu và khả năng truy xuất nguồn gốc trong nhiều tình huống khác nhau, từ thu thuế cho đến phân phối các khoản hỗ trợ tài chính.
* Giải pháp thanh toán: Khi nói đến việc việc gửi tiền trên toàn thế giới, công nghệ blockchain đã chứng minh tính hiệu quả của mình. Gửi tiền điện tử cho bạn bè, gia đình và những người khác trên khắp thế giới có chi phí rẻ hơn và nhanh hơn so với các phương thức thanh toán được cung cấp bởi các ngân hàng tập trung. Ngoài ra, các trang web và các Apps tập trung không cung cấp cho người dùng quyền kiểm soát dữ liệu của họ và thường không thưởng cho người dùng dựa theo giá trị thực mà họ mang lại cho nền tảng. Các ứng dụng phi tập trung dựa trên Blockchain (dApps) loại bỏ đối tượng trung gian, mang đến cho người dùng tiềm năng trong việc được giảm phí, nhận được ưu đãi tốt hơn và có được hiệu quả giao dịch cao hơn trong quá trình gửi và nhận tiền kỹ thuật số. Như Vitalik Buterin từng nói, các giải pháp blockchain cho phép mọi người làm việc trực tiếp với nhau, loại bỏ các khâu trung gian hoặc các hệ thống tập trung. “Trong khi hầu hết các công nghệ có xu hướng tự động, đưa người lao động ra rìa để làm các công việc tầm thường thì blockchain lại ngược lại. Thay vì khiến công việc xa khỏi người tài xế taxi, blockchain đưa công việc xa khỏi Uber và cho phép các tài xế taxi làm việc trực tiếp với khách hàng.”
* Internet vạn vật (IoT): Blockchain và Internet Vạn Vật (IoT) là một kết hợp tự nhiên. Blockchain là công nghệ phi tập trung và các mạng lưới IoT thường được sử dụng để thu thập dữ liệu từ các nguồn nằm rải rác. Blockchain cho phép các tổ chức giữ một sổ cái bất biến và minh bạch về các thiết bị IoT, dữ liệu thu thập và các tương tác giữa các thiết bị. Trong số các tính năng bảo mật và các ứng dụng tiền điện tử, blockchain cung cấp một nền tảng lý tưởng cho các giao dịch giữa máy với máy (M2M). Vì blockchain là một công nghệ dựa trên việc tạo điều kiện cho các giao dịch chính xác và an toàn, điều đó chỉ có ý nghĩa khi nó được tích hợp với IoT để đảm bảo tính trách nhiệm giải trình và tính chính xác và bảo mật dữ liệu. Đó là lý do tại sao nhiều công ty đã đặt rất nhiều tài nguyên vào mạng IoT được hỗ trợ bởi blockchain.
  + 1. Phân loại

Hệ thống Blockchain thường được phân chia thành 3 loại: Public, Private và Permissioned, trong đó:

Public

Đây là loại Blockchain mà bất kỳ ai cũng có quyền đọc và ghi dữ liệu. Quá trình xác thực giao dịch trên hệ thống Blockchain này đòi hỏi phải có hàng nghìn (thậm chí nhiều hơn) nút (Node) tham gia. Vì vậy, hệ thống Blockchain này được đánh giá khá an toàn (vì chi phí để thực hiện một vụ tấn công vào hệ thống là khá cao). Điển hình cho loại Blockchain này là các đồng tiền điện tử như Bitcoin, Ethereum…

**Private:**

Private Blockchain là một mạng do một nhóm tổ chức cùng nhau dựng lên một mạng phục vụ những nghiệp vụ kinh doanh nhất định trong nhóm tổ chức đó, mà họ không muốn bên ngoài can thiệp vào, ví dụ như một mạng phân phối xe hơi giữa các tổ chức nhà sản xuất, nhà phân phối, nhà bán lẻ.., mỗi người khi tham gia vào mạng sẽ được nhận một cặp public-private key, giúp xác định danh tính của họ, họ thuộc về tổ chức nào, quyền của họ đối với hệ thống. Người dùng chỉ được quyền đọc dữ liệu, không có quyền ghi (quyền này thuộc về một tổ chức bên thứ 3 với độ tin cậy tuyệt đối). Bên thứ 3 có toàn quyền quyết định mọi thay đổi trên Blockchain. Thời gian xác thực giao dịch đối với Private Blockchain khá nhanh (vì chỉ cần một lượng nhỏ thiết bị tham gia vào giao dịch). Ripple là một dạng Private Blockchain, hệ thống này chỉ cần 80% các nút hoạt động ổn định là có thể tiến hành giao dịch.

**Permissioned**

Hay còn gọi là Consortium, một dạng của Private nhưng bổ sung thêm một số tính năng nhất định, kết hợp giữa "niềm tin" khi tham gia vào Public và "niềm tin tuyệt đối" khi tham gia vào Private. Ví dụ: Các ngân hàng hay tổ chức tài chính liên doanh sẽ sử dụng Blockchain cho riêng mình.

**Tóm tắt sự khác nhau giữa Private Blockchain và Public Blockchain:**

|  |  |  |
| --- | --- | --- |
|  | **Public Blockchain** | **Private Blockchain** |
| Truy cập | Đọc/Ghi: Không giới hạn | Phân quyền đọc/ghi |
| Tốc độ | Chậm hơn | Nhanh hơn |
| Bảo mật | Proof-of-work/Proof-of-state | Pre-approved participants |
| Xác định danh tính | Ẩn danh | Xác định danh tính |
| Loại giao dịch | Giao dịch cơ bản | Tất cả các giao dịch |
| Chi phí khởi tạo | Rẻ hơn (Chỉ cần tham gia và xây dựng ứng dụng) | Đắt hơn (Tự xây dựng mạng lưới) |
| Chi phí giao dịch | Đắt hơn | Rẻ hơn |

Bảng 1- 2: Sự khác biệt giữa private blockchain và public blockchain

* 1. Giới hạn phạm vi bài toán

Với đặc điểm của các kì thi là bảo mật điểm của thí sinh chỉ có cá nhân thí sinh và những người có thẩm quyển mới xem được điểm vì vậy trong bài toán này sẽ sử dụng mạng private blockchain. Trong đó, **Hyperledger Fabric** là một mạng Private Blockchain điển hình, được sinh ra để phục vụ cho mục đích giao dịch riêng tư giữa các doanh nghiệp và có thể áp dụng trong bài toán này. Chi tiết về fabric sẽ được nêu rõ ở trang tiếp theo. Ứng dụng chỉ áp dụng được với những môn thi trắc nhiệm đối với những môn thi tự luận thì cần xem xét và phát triển thêm. Đồ án sẽ tập trung giải quyết với những bài thi trắc nhiệm như toán, lý , hóa, tiếng anh…và có thể cho phép một lượng lớn các thí sinh thi cùng lúc với các biện pháp cân bằng tải và các biện pháp phòng tránh sự cố sẽ được đề cập đến ở chương sau.

Chương 1 đã trình bày các đặc điểm cụ thể của bài toán ứng dụng Blockchain trong thi cử. cũng như các định nghĩa khái niệm, giải thích về công nghệ Blockchain. Bên cạnh đó, phạm vi bài toán cũng được giới hạn và nêu rõ, là cơ sở cần thiết cho việc xác định các phương pháp thích hợp để giải quyết một cách thích hợp nhất.

# CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

Chương này sẽ xây dựng mô hình hệ thống thi cử, giới thiệu các thành phần của hệ thống, các giai đoạn xử lý và các công nghệ, biện pháp giúp ổn định, tránh sự cố cho hệ thống đối với kì thì có lượng thí sinh lớn. Mô hình mạng lưới blockchain và framework blockchain được sử dụng để xây dựng hệ thống.

* 1. Tổng quan về hệ thống

Như đã được nêu ở trên, hệ thống tổ chức thi sẽ đầy đủ các quy trình như một kì thi, thí sinh làm bài thi trắc nhiệm trên giao diện web, và điểm sẽ được lưu trên mạng private blockchain.

Api

Mạng riêng không truy cập được từ bên ngoài

Backend: Python

* DB: Lưu đề thi
* Code: Gửi đề thi, và nhận kết quả làm bài, sau đó chấm điểm

Frontend-Vuejs:

* Login
* Xác thực 2 bước dùng OTP (do giám thị nhập)
* Cơ chế chỉ login được 1 lần trên một máy
* Giao diện làm bài thi trắc nghiệm

Mạng blockchain

* Lưu thông tin thí sinh dự thi
* Cập nhật điểm sinh viên
* Tra cứu thông tin sinh viên

Hình 2- 1: Thiết kế chi tiết hệ thống tổ chức thi

Chi tiết về phần giao diện làm bài thi:

* Xác thực: Gồm phần login , xác thực 2 bước với otp do giám thị nhập, và tính năng login một lần trên một trình duyệt của một máy tính nếu tài khoản được đăng nhập ở nơi khác thì thì tài khoản ở trong phòng thi sẽ bị thoát ra và không đăng nhập lại nữa. Phần này đảm bảo tính an toàn tránh tình trạng làm hộ bài thi.
* Phần bài làm: làm bài theo hình thức trắc nhiệm với thời gian quy định, khi hết thời gian thì bài làm tự động được nộp, gần hết thời gian sẽ có những thông báo. Nếu thí sinh làm xong bài thi trước thời gian thì có thể nộp bài trước.

Chi tiết phần backend:

* Cơ sở dữ liệu: Gồm thông tin đăng nhập. và đề thi phân biệt theo mã đề.
* Chấm điểm: Nhận kết quả làm bài của thí sinh và chấm điểm sau đó gửi kết quả lên mạng blockchain.

Chi tiết phần mạng blocchain:

* Mạng private blochain: sử dụng platfrom hyperledger fabric sẽ được trình bày ở phần tiếp theo.
* Cơ sở dữ liệu: Lưu thông tin của sinh viên như họ tên, mã dự thi, điểm.
* Tra cứu kết quả: Trả về kết quả thi thông qua api.

Trong mô hình trên thì phần giao diện làm bài và phần backend sẽ nằm trên một mạng riêng để tránh truy cập từ bên ngoài và chỉ có backend có thể truy cập ra được bên ngoài để gửi điểm lên mạng blochain. Quá trình trao nhận dữ liệu phải sử dụng phương thức https để đảm bảo dữ liệu trên đường truyền.

Vì lượng thí sinh lớn và tính chất quan trọng của kì thi nên cần đảm bảo tính ổn định của hệ thống có thể chịu được nhiều request cùng lúc bằng cách sử dụng các phương pháp cân bằng tải. Trong đồ án này sẽ sử dụng công nghệ kubernetes để cân bẳng tải cụ thể mô hình như sau:

* Server frontend sẽ được deploy lên nhiều server và load balancer để giảm tải requests đến từng server tương tự với phần backend. Trong demo thì mỗi phần được deploy lên 2 server.
* Phần cơ sở dữ liệu sẽ dùng mongodb replication để phòng tránh việc nếu một server db mất lập tức có server khác lên thay đảm bảo tính ổn định của hệ thống. Trong demo thì db sẽ được replicate ra 3 server

Chi tiết từng kỹ thuật sử dụng trong từng thành phần nêu trên sẽ được trình bày cụ thể trong các phần tiếp theo.

* 1. Giới thiệu về Hyperledger Fabric

1. Giới thiệu về Hyperledger

Hyperledger là một dự án mã nguồn mở, nó xây dựng một hệ sinh thái các giải pháp và người dùng trên nền tảng công nghệ blockchain nhằm giải quyết các vấn đề trong ngành công nghiệp.

Hyperledger thuộc tổ chức Linux Foundation. NodeJs, Alljoyn, Dronecode là một số dự án nổi tiếng của Linux Foundation. Mục đích của Linux Foundation là tạo ra một cộng đồng các nhà phát triển làm việc trên các dự án nguồn mở, nhằm duy trì sự phát triển của các dự án, trong đó, mã nguồn dự án luôn được nâng cấp, sửa đổi và phân phối lại. Tư tưởng của Hyperledger là thế giới sẽ gồm nhiều kênh thanh toán (private chain) riêng biệt với các thị trường khác nhau. Mỗi doanh nghiệp có những đặc trưng riêng, nên các ứng dụng cho các doanh nghiệp sẽ cần phát triển với các quy tắc được cá nhân hóa. Không giống như ethereum có xu hướng buộc các nhà phát triển xây dựng các ứng dụng của họ xung quanh một bộ giao thức định sẵn. Dự án Hyperledger bắt đầu với một số ít các nhà phát triển vào cuối năm 2015. Những nhà phát triển này đến từ nhiều lĩnh vực khác nhau như khoa học dữ liệu, sản xuất, ngân hàng, v.v. và họ có một mục tiêu chung, đó là làm cho blockchain trở thành công nghệ dễ tiếp cận hơn với các nhà phát triển, các doanh nghiệp. Dự án được bắt đầu với các thử nghiệm tương tác giữa ứng dụng và một mạng blockchain an toàn.

**Cách hoạt động của Hyperledger**

Trên mạng Hyperledger Các peer liên kết trực tiếp với nhau và chỉ có sổ cái của riêng họ được cập nhật về thỏa thuận giao dịch. Các bên giúp thực hiện giao dịch chỉ được biết một lượng thông tin đủ để họ cần để chuyển tiếp và cho phép giao dịch trên mạng. Giả sử Alice và Bob thực hiện giao dịch đặc biệt của họ trên mạng Hyperledger, cô sẽ tìm kiếm Bob thông qua một ứng dụng truy vấn danh sách các thành viên tham gia vào mạng. Sau khi đã được xác thực, hai peer sẽ được kết nối và kết quả được trả về. Trong thỏa thuận hai bên này, cả hai kết quả trả về phải giống nhau để giao dịch có thể được xác nhận. Trong các giao dịch khác với nhiều bên, nhiều quy tắc hơn có thể được áp dụng.

**Những đặc điểm của Hyperledger**

Trong kiến ​​trúc này, những đăc điểm đáng chú ý nhất được thể hiện trong các peer của mạng. Các peer đã được chia thành ba vai trò riêng biệt, đó là:

* **Endorser**: Các endorser là những peer thực thi các giao dịch trong chaincode container và đề xuất giao dịch lên mạng dựa trên kết quả của hợp đồng thông minh. Tất cả các endoser peer phải được cài đặt chaincode.
* **Committer**: Đây là những peer không nhất thiết phải cài đặt chaincode, chúng lưu trữ sổ cái đầy đủ (full ledger). Sự khác biệt chính giữa committer peer và endoser peer là việc committer peer không thể gọi chaincode hoặc chạy các hàm trong hợp đồng thông minh.
* **Consenters**: Các nút này chịu trách nhiệm điều hành sự đồng thuận của mạng.Consenters có trách nhiệm xác nhận các giao dịch và quyết định các giao dịch sẽ được đưa vào sổ cái.

1. Hyperledger Fabric

**Hyperedger Fabric** là một trong 5 Framework về Blockchain nằm trong chiến lược **Hyperledger Umbrella**của**Linux Foundation** gồm: Hyperledger Indy, Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth, Hyperledger Burror. Điều đặc biệt là Hyperledger Fabric được contributed bởi công ty lớn như **IBM**. Hyperledger Fabric có **modularity** (tính mô đun) khá cao nên nó cho phép các Doanh nghiệp dễ dàng **plug and play** để xây dựng một ứng dụng Private Blockchain phù hợp các yêu cầu nghiệp vụ của mình .

Hyperledger Fabric là một triển khai của công nghệ sổ cái phân tán (DLT) mang lại sự bảo mật, khả năng mở rộng, bảo mật và hiệu suất mạng sẵn sàng cho doanh nghiệp, trong một kiến ​​trúc blockchain mô-đun. Hyperledger Fabric cung cấp các chức năng mạng blockchain sau:

* Quản lý danh tính: Để kích hoạt các mạng được phép, Hyperledger Fabric cung cấp dịch vụ nhận dạng thành viên để quản lý ID người dùng và xác thực tất cả người tham gia trên mạng. Danh sách kiểm soát truy cập có thể được sử dụng để cung cấp các quyền bổ sung thông qua ủy quyền cho các hoạt động mạng cụ thể. Ví dụ: ID người dùng cụ thể có thể được phép gọi chaincode, nhưng bị chặn triển khai một chaincode mới.
* Quyền riêng tư và bảo mật: Hyperledger Fabric cho phép cạnh tranh lợi ích kinh doanh và bất kỳ nhóm nào yêu cầu giao dịch riêng tư, bí mật, cùng tồn tại trên cùng một mạng được phép. Các kênh riêng là các đường dẫn nhắn tin và bị hạn chế, có thể được sử dụng để cung cấp quyền riêng tư và bảo mật giao dịch cho các tập hợp con cụ thể của các thành viên mạng. Tất cả dữ liệu, bao gồm thông tin giao dịch, thành viên và kênh, trên một kênh là vô hình và không thể được truy cập với bất kỳ thàng viên nào không được cấp quyền.
* Xử lý hiệu quả: Hyperledger Fabric cấp vai vai trò trong mạng theo loại nút. Để cung cấp đồng thời và song song cho mạng, việc thực hiện giao dịch được tách biệt khỏi lệnh giao dịch và cam kết. Thực hiện các giao dịch trước khi đặt hàng chúng cho phép mỗi nút ngang hàng xử lý đồng thời nhiều giao dịch. Việc thực hiện đồng thời này làm tăng hiệu quả xử lý trên mỗi thiết bị ngang hàng và đẩy nhanh việc phân phối các giao dịch đến dịch vụ đặt hàng.
* Chức năng chaincode: Các ứng dụng Chaincode mã hóa logic được gọi bởi các loại giao dịch cụ thể trên kênh. Ví dụ, Chaincode xác định các tham số để thay đổi quyền sở hữu tài sản, đảm bảo rằng tất cả các giao dịch chuyển quyền sở hữu phải tuân theo cùng các quy tắc và yêu cầu.
* Thiết kế mô-dun: Hyperledger Fabric thực hiện một kiến ​​trúc mô đun để cung cấp lựa chọn chức năng cho các nhà thiết kế mạng. Các thuật toán cụ thể để nhận dạng, đồng thuận và mã hóa, ví dụ, có thể được cắm vào bất kỳ mạng Hyperledger Fabric nào (HF). Kết quả là một blockchain phổ quát.

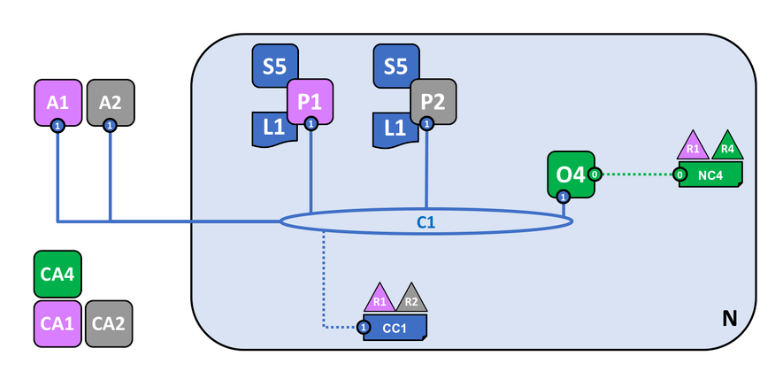
**Các khái niệm cơ bản**

* Identity: Mỗi tác nhân trong HF bao gồm peers, orderer, client, admin,.. đều có một identity. Các tác nhân này sẽ sử dụng identity của mình để tương tác với mạng, identity đó được cấp dưới dạng một **X.509 digital certificate**. Các identity rất quan trọng vì nó còn giúp hệ thống xác định tác nhân có thể thực hiện những hành động nào, có quyền truy cập vào những tài nguyên nào của network. Một **Public key Infrastructue** (PKI) là tập hợp các internet technologies cung cấp sự liên lạc an toàn trong network. Trong ngữ cảnh Hyperledger Fabric network thì PKI là tập hợp các **Certificate Authorities** của các tổ chức mà ở bài trước được minh họa là các node CA trong kiến trúc mạng. Các Certificate Authorities này sẽ cấp cho mỗi tác nhân muốn tham gia vào vào mạng một identity.
* Membership: Membership Service Provider (MSP) của một tổ chức tham gia - xác định CA nào được ủy quyền cấp identity hợp lệ cho các thành viên của tổ chức đó. MSP không chỉ đơn giản là liệt kê ai là người tham gia network hoặc thành viên của channel. MSP có thể xác định các vai trò cụ thể mà một tác nhân có thể đóng trong phạm vi tổ chức mà MSP đại diện (ví dụ: quản trị viên hoặc là thành viên của nhóm tổ chức con) và là cơ sở để xác định đặc quyền truy cập trong phạm vi network và channel (ví dụ: quản trị viên, quyền đọc, quyền ghi). Ngoài ra, MSP có thể cho phép xác định danh sách các identity đã bị thu hồi ( Certificate Revocation List ) xem hình đầu tiên.
* Peer: Một Blochain network bao gồm chủ yếu các peer. Peer là thành tố cơ bản của network vì nó lưu trữ bản sao của Smart Contract (Chaincode ) và bản sao của Ledger. Các peer có thể được tạo, start, stop, tái cấu hình, thậm chí là xóa.
* Ledger: Ở cấp độ đơn giản nhất, một blockchain bất biến ghi lại các giao dịch cập nhật kết quả thực hiện giao dịch trong một Ledger. Smart contract có thể truy cập theo 2 phần riêng biệt, một là lịch sử của tất cả các giao dịch và hai là trạng thái hiện tại của Ledger (kết quả sau khi thực hiện tất cả các giao dịch).
* The odering service: Hầu hết các mạng Public Blockchain, chẳng hạn như Ethereum và Bitcoin, bất kỳ node nào cũng có thể tham gia vào quá trình đồng thuận ( consensus). Hyperledger Fabric thì khác, nó có một loại node được gọi là một orderer (hay còn được gọi "ordering node" ), thực hiện nhiệm vụ "consensus", có thể là chỉ có 1 ordering node trong một network, hoặc có thể có nhiều node tạo nên 1 ordering service. Bởi vì thiết kế Hyperledger Fabric là dựa trên các thuật toán "deterministic consensus ", nên bất kỳ block nào đã được các peer validates và được tạo bởi ordering service thì đều được đảm bảo là chính xác. Ledger sẽ không thể xảy ra tình trạng rẽ nhánh như các blockchain khác. Ngoài vai trò trên, ordering service còn lưu trữ các thông tin khác như tổ chức nào được phép tạo channel, ai có thể thay đổi các cấu hình channel, và tất cả hành động thay đổi cấu hình đó đều phải đi qua orderer.

**Kiến trúc của một mạng Hyperledger Fabric**

Phần này sẽ phác thảo các tính năng thiết kế chính trong Hyperledger Fabric để thực hiện một giải pháp blockchain doanh nghiệp toàn diện nhưng có thể tùy chỉnh:

* Assets: Tài sản có thể từ hữu hình (bất động sản và phần cứng) đến vô hình (hợp đồng và sở hữu trí tuệ). Hyperledger Fabric cung cấp khả năng sửa đổi tài sản bằng cách sử dụng các giao dịch chuỗi. Các tài sản được thể hiện trong Hyperledger Fabric dưới dạng tập hợp các cặp khóa-giá trị, với các thay đổi trạng thái được ghi lại dưới dạng giao dịch trên sổ cái Kênh. Các tài sản có thể được biểu diễn dưới dạng nhị phân và / hoặc JSON.
* Chaincode: là phần mềm xác định tài sản và hướng dẫn giao dịch để sửa đổi tài sản. Nói cách khác, nó lập ra logic kinh doanh. Chaincode thực thi các quy tắc để đọc hoặc thay đổi các cặp giá trị khóa hoặc thông tin cơ sở dữ liệu và các trạng thái khác. Các hàm Chaincode thực thi đối với cơ sở dữ của sổ cái và được bắt đầu thông qua một đề xuất giao dịch. Kết quả thực thi chuỗi mã có thể được gửi tới mạng và áp dụng cho sổ cái trên tất cả các đồng nghiệp (peers).
* Tính năng sổ cái (ledger): Sổ cái là bản ghi trình tự, chống giả mạo của tất cả các chuyển trạng thái trong vải. Sổ cái bao gồm một blockchain (‘chuỗi’) để lưu trữ bản ghi tuần tự, bất biến trong các khối, cũng như cơ sở dữ liệu trạng thái để duy trì trạng thái fabric hiện tại. Có một sổ cái cho mỗi kênh. Mỗi đồng nghiệp (peer) duy trì một bản sao của sổ cái cho mỗi kênh mà họ là thành viên.
* Quyền riêng tư thông qua các kênh (Channels): Hyperledger Fabric sử dụng một sổ cái bất biến trên cơ sở mỗi kênh, cũng như các chuỗi mã có sửa đổi trạng thái hiện tại của tài sản (ví dụ: cập nhật các cặp giá trị khóa). Một sổ cái tồn tại trong phạm vi của một kênh - nó có thể được chia sẻ trên toàn bộ mạng (giả sử mọi người tham gia đang hoạt động trên một kênh chung) - hoặc có thể được tư nhân hóa để chỉ bao gồm một nhóm người tham gia cụ thể.
* Dịch vụ bảo mật và thành viên: Hyperledger Fabric củng cố một mạng lưới giao dịch nơi tất cả những người tham gia đều có danh tính. Cơ sở hạ tầng khóa công khai được sử dụng để tạo chứng chỉ mật mã được gắn với các tổ chức, thành phần mạng và người dùng cuối hoặc ứng dụng khách. Do đó, kiểm soát truy cập dữ liệu có thể được thao tác và chi phối trên mạng rộng hơn và trên các cấp kênh. Điều này đã cho phép khái niệm của Hyperledger Fabric, cùng với sự tồn tại và khả năng của các kênh, giúp giải quyết các tình huống trong đó quyền riêng tư và bảo mật là mối quan tâm hàng đầu.
* Đồng thuận (Consensus): Trong công nghệ sổ cái phân tán, sự đồng thuận gần đây đã trở thành đồng nghĩa với một thuật toán cụ thể, trong một chức năng duy nhất. Tuy nhiên, sự đồng thuận bao gồm nhiều hơn là chỉ đồng ý với thứ tự giao dịch và sự khác biệt này được nêu bật trong Hyperledger Fabric thông qua vai trò cơ bản của nó trong toàn bộ dòng giao dịch, từ đề xuất và chứng thực, đến đặt hàng, xác nhận và cam kết. Tóm lại, sự đồng thuận được định nghĩa là xác minh toàn vòng tròn về tính đúng đắn của một tập hợp các giao dịch bao gồm một khối. Sự đồng thuận cuối cùng đạt được khi thứ tự và kết quả của một giao dịch khối khối đã đáp ứng các kiểm tra tiêu chí chính sách rõ ràng. Các kiểm tra và số dư này diễn ra trong vòng đời của một giao dịch, và bao gồm việc sử dụng các chính sách chứng thực để ra lệnh cho các thành viên cụ thể phải xác nhận một loại giao dịch nhất định, cũng như các chuỗi hệ thống để đảm bảo rằng các chính sách này được thi hành và duy trì. Trước khi cam kết, các đồng nghiệp sẽ sử dụng các chuỗi hệ thống này để đảm bảo rằng có đủ chứng thực và chúng có nguồn gốc từ các thực thể phù hợp. Ngoài ra, kiểm tra phiên bản sẽ diễn ra trong đó trạng thái hiện tại của sổ cái được thỏa thuận hoặc đồng ý, trước khi bất kỳ khối nào có chứa các giao dịch được thêm vào sổ cái. Kiểm tra cuối cùng này cung cấp bảo vệ chống lại các hoạt động chi tiêu gấp đôi và các mối đe dọa khác có thể ảnh hưởng đến tính toàn vẹn dữ liệu và cho phép các chức năng được thực thi đối với các biến không tĩnh.



Hình 2- 2: Kiến trúc đơn giản của một mạng Hyperledger Fabric

N: (Netword) Mạng

NC: Network Configuration (Cấu hình của mạng)

C: Channel (Kênh), tập hợp các tổ chức có vai trò nhất định trong cùng một quy trình kinh doanh. Ví dụ, trong một channel về mua bán xe hơi sẽ gồm có 2 tổ chức là: Nhà sản xuất xe hơi, Nhà phân phối xe hơi.

CC: Channel Configuration (Cấu hình của kênh)

R: Organization (Tổ chức)

O: Orderer Node: Nếu như trong Publuc Blockchain, tất cả các node của mạng đều tham gia vào quá trình đồng thuận, thì ở Hyperledger Fabric chỉ có Orderer tham gia vào quá trình đó.

P: Peer, là điểm tương tác giữa các thành viên trong tổ chức tương ứng với kênh, mọi hành động của người dùng đều phải đi qua peer.

S: Smart Contract (Chaincode) được cài đặt trên kênh, định nghĩa rõ các struct, các hành động mà người dùng có thể thực hiện để tương tác trạng thái của struct được lưu trong sổ cái.

L: Ledger ( Sổ cái ), lưu trữ trạng thái của các đối tượng.

CA: Certificate Authority, phát hành identity cho người dùng hoặc node của tổ chức tương ứng. Ví dụ, người dùng A là thành viên của Tổ chức R1, khi muốn tham gia vào mạng thì sẽ gửi yêu cầu đến CA1, sau đó CA1 sẽ tạo ra một identity gồm private-key, public-key và các đặc tính liên quan khác, sau đó trả về cho người dùng A, từ đó về sau A dùng identity đó để thực hiện các tương tác với mạng, mạng sẽ tự động biết đó là người dùng A đến từ tổ chức R1.

A: Application, ứng dụng hay giao diện (web, mobile app ) giúp người dùng tương tác với hệ thống dễ dàng hơn.

Tất cả cả thành phần ở trên hoặc là chạy trên docker hoặc là ta có thể thấy được định nghĩa của chúng trong code, nên tạm thời ta coi chúng là thành phần vật lý của mạng

Transaction trong Hyperledger Fabric

Giải định rằng chúng ta đã hoàn thiện xong phần network, trong network đó có 1 channel gồm Orderer và 2 tổ chức A, B. Mỗi client A, B đã được đăng ký một identity với Certificate Authority (CA) của tổ chức tương ứng. Chaincode đã được install trên các peer và được instantiate trên channel. Chaincode chứa logic để thực hiện một transaction mua-bán ô tô. Endorsement Policy (Chính sách chứng thực) cũng đã được đặt ra cho chaincode này, với yêu cầu là 1 transaction phải được endorsing (chứng thực ) bởi cả 2 tổ chức ("AND ('Org1MSP.peer', 'Org2MSP.peer')" ).

Client A

SDK

Peer

Peer

Hình 2- 3: Sơ đồ bắt đầu một transaction

Client A gửi một request muốn mua ô tô. Request này được target đến peerA, peerB, là các peer của 2 tổ chức A, B. Một Transaction Proposal (đề xuất giao dịch ) sẽ được xây dựng từ request ở trên. Transaction Proposal là một yêu cầu để invoke (gọi ) một function nào đó trong chaincode với các tham số đầu vào nhất định, function này có thể là write hoặc read Ledger. SDK servers sẽ có nhiệm vụ đóng gói Transaction Proposal thành một "Properly Architected Format" (bộ đệm giao thức trên gRPC) và lấy thông-tin-số (hiểu đơn giản đây là khi client dùng identity của mình để tạo request thì hệ thống sẽ phân tích và lấy được thông-tin-số từ identity đó) của client để tạo signature (chữ ký ) cho Transaction Proposal. Các Endorsing peers sẽ kiểm tra chữ ký và thực thi transaction với bản sao Ledger được lưu trên nó. Các Endorsing Peers sẽ lấy đầu vào của Transaction Proposal làm đối số cho function trong chaincode mà được gọi đến. Chaincode sẽ được thực thi đối với bản sao Ledger lưu tại mỗi peer để tạo ra một Transaction Value bao gồm giá trị trả về sau khi thực thi, cặp key-value cho một đối tượng cần được tạo hoặc cập nhật. Tại thời điểm này, Ledger sẽ bị "đóng băng" lại. Transaction Value ở trên, cùng với chữ ký của Endorsing Peer đã tạo ra nó sẽ được gửi trở lại đến SDK dưới dạng một "proposal response"

Application sẽ kiểm tra chữ ký của các Endorsing Peers và so sánh các Proposal Response xem chúng có giống nhau không. Nếu đây chỉ là một hành động truy vấn thì Application sẽ không submit transaction đến ordering service. Nếu đây là một transaction cập nhật Ledger, Application sẽ kiểm tra xem Endorsement Policy đã chỉ định trước đó ("AND ('Org1MSP.peer', 'Org2.peer')" ) có được thực hiện hay không trước khi submit lên Ordering Service (tức là cả peerA và peerB đều đã endorsing chưa). Ngay cả khi Application "không kiểm tra Proposal Response" hoặc chuyển trực tiếp transaction chưa được chứng thực đến Ordering Service, thì Endorsement Policy vẫn sẽ được thực thi bởi các Peer và ở giai đoạn commit validation.

Application sẽ "broadcast" Transaction Proposal và Transaction Response trong một "Transaction Message" đến Ordering Service. Transaction sẽ chứa cặp key-value cho đối tượng cần cập nhật, các chữ ký của các Endorsing Peer. Ordering Service sẽ không cần phải kiểm tra nội dung của transaction, nó chỉ cần nhận tất cả các transaction và sắp xếp nó theo thứ tự thời gian.

Các blocks chứa các transactions sẽ được chuyển đến tất cả các Peers trên Channel. Các transaction trong block sẽ được kiểm tra để đảm bảo rằng Endorsement Policy đã được thực hiện và Ledger được "đóng băng" trong lúc tạo cặp key-value. Từng transaction trong block sẽ được gắn tag là hợp lệ hoặc không hợp lệ.

* 1. Các công nghệ khác được dùng trong hệ thống

Blockchain có tính phân tán vì vậy để có thể demo được trên một máy tính thì ta cần sử dụng thêm một số công nghệ và ở trong phạm vi đồ án này sẽ sử dụng docker.

Về phần hệ thống để vận hành tốt tránh các rủi ro như sập hệ thống, mất db do một số lí do nào đó thì trong đồ án này sẽ dùng đến kubernetes và mongo replication. Chi tiết về từng phần sẽ được trình bầy ở phần dưới đây.

* + 1. Docker

Docker là 1 công cụ hỗ trợ việc tạo môi trường ảo (container) trên linux nhanh gọn và đơn giản.

Docker được thiết kể để tạo mới, chạy các tứng dụng bằng cách sử dụng các container. Các container cho phép đóng gói các ứng dụng với các thư viện, service, database... tất cả sẽ được đóng gói lại trong một container duy nhất. Nhờ vậỵ các developer sẽ chỉ cần tập trung vào việc viết code mà ko phải quan tâm đến việc cài cắm các service hay các ứng dụng thứ 3.

**Dokerfile**

Dockerfile chứa tập hợp các lệnh để docker có thể đọc hiểu và thực hiện để đóng gói thành một image theo yêu cầu của người phát triển. Có thể hiểu Dockerfile là một file text chứa tất cả các câu lệnh để chỉ dẫn cho docker xây dựng images. Bằng cách sử dụng câu lệnh *docker build* docker sẽ tự động xây dựng theo các bước mà developer viết trên dockerfile. Format của Dockerfile :

|  |
| --- |
| # Comment  INSTRUCTION arguments |

Hình 2- 4: Format của dockerfile

 Ý nghĩa của các chỉ dẫn trong Dockerfile:

* FROM : Là base image để chúng ta tiến hành build một image mới trên image. Vd: FROM ruby || ubuntu || mysql
* MAINTAINER : thông tin của người phát triển Dockerfile
* RUN : Sử dụng khi muốn thực thi các command trong quá trình xây dựng image
* COPY : Copy một file từ host machine tới docker container
* WORKDIR : image sẽ trỏ tới directory khai báo, nếu không có thì docker sẽ tự tạo mới
* ENV : Định nghĩa các biến môi trường
* CMD : Sử dụng khi muốn thực thi các command trong quá trình build container mới từ image

Sau đây là Dockerfile xây dựng với base image: ubuntu kết hợp với nginx

|  |
| --- |
| CMD echo "Hello ubuntu"  # Update the repository  RUN apt-get update  # Install necessary tools  RUN apt-get install -y nano wget dialog net-tools vim git  # Download and Install Nginx  RUN apt-get install -y nginx  # Remove file index docker  RUN rm -v /var/www/html/index.nginx-debian.html  ADD index.nginx-debian.html /var/www/html  EXPOSE 80  CMD [ "nginx", "-g", "daemon off;" ] |

Hình 2- 5: Dockerfile xây dựng với base image ubuntu kết hợp với nginx

Sử dụng các câu lệnh *docker build -t "myubuntu:1.0"* để build image với tên là myUbuntu. Sau khi build xong image ta sẽ thực hiện tạo container từ image myubuntu vừa tạo được bằng câu lệnh sau*: docker run -p 8080:80 -it myubuntu:1.0 /bin/bash.* Truy cập vào địa chỉ localhost:8080 sẽ nhận được giao diện mặc định của nginx.

**Dockercompose**

Docker compose là một công cụ dùng để quản lý và liên kết nhiều containers , mỗi container chạy 1 service riêng biệt nhưng phục vụ chung một ứng dụng. Docker-compose có định dạng giống file yaml (key - value), chứa các chỉ dẫn để khởi động, liên kết các container với nhau, sử dụng docker-compose cũng đơn giản chỉ với một vài câu lệnh đơn giản là có thể hiểu được. Ba bước cơ bản để sử dụng compose file:

1. Xây dựng Dockerfile cho từng ứng dụng riêng biệt.
2. Định nghĩa các service trong file docker-compose.yml để liên kết và chạy các ứng dụng cùng một thời điểm nhưng tách biệt môi trường.
3. Sự dụng 2 câu lệnh sau để build và start docker-compose: *docker-compose build* và *docker-compose up*

Ví dụ sử dụng dockercompose để liên kết 3 containers, mỗi container chạy 1 service gồm có nginx, rails, mysql server. Dockecompose file:

|  |
| --- |
| version: "2"  services:  # web:  # build: ./nginx  # depends\_on:  # - app  # links:  # - app  # ports:  # - 8080:80  app:  build: .  command: bundle exec rails s -p 3000 -b '0.0.0.0'  volumes:  - .:/noteapp  ports:  - "3000:3000"  depends\_on:  - db  links:  - db  db:  image: mysql:5.7  environment:  MYSQL\_ROOT\_PASSWORD: 123456  ports:  - "3307:3306" |

Hình 2- 6: Docker compose file

Giải thích về file docker compose:

* version: để khai báo phiên bản định dạng mà mình thiết kế file docker-compose, ví thế tùy từng phiên bản mà các câu lệnh chỉ dẫn khác nhau.
* services: chứa các container sẽ được khởi tạo và chạy
* build: địa chỉ của dockerfile
* command: khai báo lệnh sẽ được chạy khi start container
* ports: port của hostmachine : port của container

Trong phần demo thì docker thay thế cho các server vật lí, và được sử dụng rất phổ biến.

* + 1. Mongodb

1. Khái niệm

MongoDB là một chương trình cơ sở dữ liệu mã nguồn mở được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp và đa dạng và không cố định (hay còn gọi là Big Data).

1. Lợi thế của mongodb so với các cơ sở dữ liệu dạng quan hệ

Mongodb có những lợi thế sau:

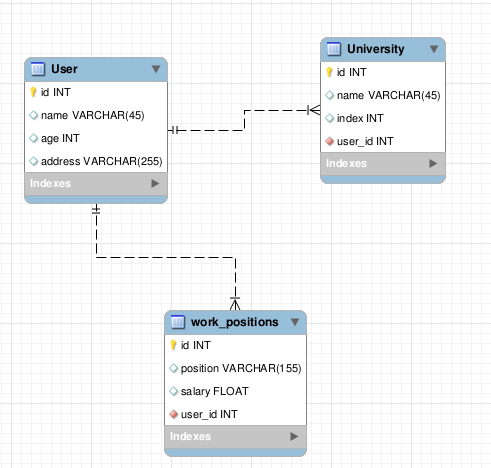
* Ít Schema hơn: MongoDB là một cơ sở dữ liệu dựa trên Document, trong đó một Collection giữ các Document khác nhau. Số trường, nội dung và kích cỡ của Document này có thể khác với Document khác.
* Cấu trúc của một đối tượng là rõ ràng.
* Không có các Join phức tạp.
* Khả năng truy vấn sâu hơn. MongoDB hỗ trợ các truy vấn động trên các Document bởi sử dụng một ngôn ngữ truy vấn dựa trên Document mà mạnh mẽ như SQL.
* MongoDB dễ dàng để mở rộng.
* Việc chuyển đổi/ánh xạ của các đối tượng ứng dụng đến các đối tượng cơ sở dữ liệu là không cần thiết.
* Sử dụng bộ nhớ nội tại để lưu giữ phần công việc, giúp truy cập dữ liệu nhanh hơn.

1. Một số đặc điểm của mongodb

* Kho lưu định hướng Document: Dữ liệu được lưu trong các tài liệu kiểu JSON.
* Lập chỉ mục trên bất kỳ thuộc tính nào.
* Các truy vấn đa dạng.
* Cập nhật nhanh hơn.

1. So sánh mongodb với cơ sở dữ liệu quan hệ

So sánh mongodb và Mysql. Thiết kế database trong Mysql:



Hình 2- 7: Thiết kế database trong Mysql

Trong khi trong thiết kế Schema của MongoDB sẽ chỉ có một Collection:

|  |
| --- |
| {  \_id: user\_id,  name: user\_name,  email: user\_email,  age: age,  address: user\_address,  universities: [  {name: university\_name, index: university\_index}  ],  work\_positions: [  {position: work\_position, salary: user\_salary}  ]  } |

Hình 2- 8: Cơ sở dữ liệu trong mongodb

Sau đây là bảng so sánh cơ sở dữ liệu mysql và mongodb:

|  |  |  |
| --- | --- | --- |
| **Operation** | **Mysql** | **MongoDB** |
| Create | Insert | Insert |
| Read | Select | Find |
| Update | Update | Update |
| Delete | Delete | Remove |
| Table | Table | Collection |
| Row | Row | Document |

Bảng 2- 1: So sánh mongodb và mysql

Lưu ý khi sử dụng mongodb:

* MongoDB sinh ra để tăng tốc độ truy xuất dữ liệu, phù hợp cho các ứng dụng cần tốc độ phản hồi nhanh(realtime như facebook chẳng hạn). Còn các tác nghiệp cần tính toàn vẹn dữ liệu(trong banking) thì Nosql sẽ ko bao giờ là 1 giải pháp cả mà người ta sẽ dùng Mysql.
* MongoDB không có tính ràng buộc, một điều tồi tệ trong Database vì vậy sẽ rất cần sự cẩn thận khi thao tác trên các collection có quan hệ dữ liệu với nhau.
* MongoDB đẩy trách nghiệm thao tác Database cho tầng ứng dụng nên sẽ tốn tài nguyên (tài nguyên bây giờ không còn là vấn đề quá lớn nữa).
* MongoDb có thể mở trộng theo chiều ngang (scale out) phương pháp tăng cường khả năng lưu trữ và xử lý là dùng nhiều máy tính phân tán. MongoDb còn có thể mở rộng theo chiều dọc (scale up) tăng cấu hình server.
* MongoDb có thể dùng nhiều máy tính phân tán để lưu trữ dữ liệu nên chi phí sẽ rẻ hơn Mysql.

1. **Một số câu lệnh thao tác trong mongodb**

Hiện thị tất cả các databases: *show dbs*

Tạo database*: use ‘database\_name’* . Nếu database chưa tồn tại thì sẽ được tạo mới.

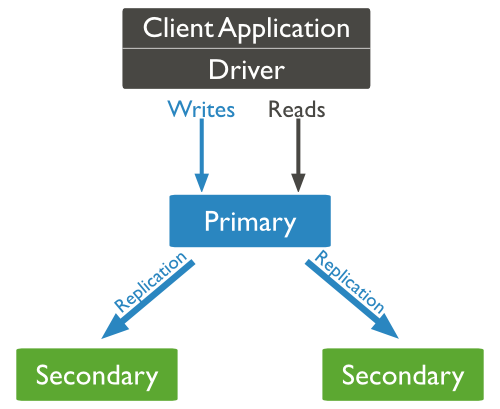
Tạo một collection và insert dữ liệu: *db.collection\_name.insert({'key\_name': 'value'})*

Get tất cả object của một collection: *db.collection\_name.find({})*

Drop một collection: *db.collection\_name.drop()*

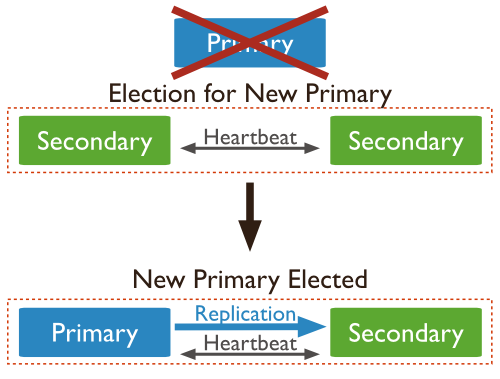
1. Replication trong mongodb

Replication có ý nghĩa là “nhân bản”, là có một phiên bản giống hệt phiên bản đang tồn tại, đang sử dụng.Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi cơ sở dữ liệu toàn vẹn, không bị mất mát trước những sự cố ngoài dự đoán là rất cao. Vì vậy, người ta nghĩ ra khái niệm “nhân bản”, tạo một phiên bản cơ sở dữ liệu giống hệt cơ sở dữ liệu đang tồn tại, và lưu trữ ở một nơi khác, đề phòng có sự cố. Phiên bản cơ sở dữ liệu phục vụ ứng dụng được lưu trữ trên server master. Phiên bản cơ sở dữ liệu “nhân bản” được lưu trữ trên server slave. Quá trình nhân bản từ master sang slave gọi là replication.Khi có một thay đổi trên cơ sở dữ liệu master, master sẽ đồng bộ đến các máy slave. MongoDB hỗ trợ replication giữa các DB cực kỳ tốt và dễ dàng thực hiện.



Hình 2- 9: Replication trong mongodb

Replication trong MongoDB theo cơ chế **Asynchronous Replication**, có nghĩa là một transaction gửi đến DB chính (Master) được thực hiện và xác nhận hoàn thành với client ngay khi hoàn thành mà không cần quan tâm đến việc bản sao lưu của transaction này đã được hoàn thành ở server dự phòng (slave) hay chưa. Mặc nhiên là khả năng **Automatic Failover** được MongoDB tự động hỗ trợ, có nghĩa là máy DB Master down thì MongoDB tự động chọn 1 trong các máy Slave làm Master và khi máy DB vừa down hoạt động lại thì MongoDB cũng tự kết nối nó vào hệ thống (lúc này sẽ là một máy Slave) một cách hoàn toàn tự động mà DBA không cần phải làm gì.



Hình 2- 10: Cơ chế automatic failover trong mongo

Ngoải ra thì MongoDB cũng hỗ trợ chế độ Read Preference có nghĩa là cho phép Client đọc vào các máy Slave để cân bằng tải cho máy Master.

Read Preference Mode trong Mongo có những lựa chọn sau:

* **primary**: Mode mặc định khi ta thiết lập, với mode này operations sẽ mặc định đọc giữ liệu từ DB Master (máy primary), nếu máy Master down thì chương trình sẽ báo lỗi.
* **primaryPreferred**: Cũng giống mode primary  nhưng khi máy Master down thì operations sẽ đọc từ máy Slave (secondary)
* **secondary**: Tất cả operations sẽ đọc giữ liệu từ các máy Slave (secondary)
* **secondaryPreferred**: Cũng giống mode secondary nhưng khi các Slave down hết thì operations sẽ đọc giữ liệu từ máy Master.
* **nearest** Operations sẽ đọc từ máy gần nhất (tức là hệ thống sẽ đọc từ máy có đỗ trệ mạng ít nhất (least network latency) không phân biệt nó là Master hay Slave.

Trong phạm vi đồ án thì mongodb sẽ được sử dụng để lưu thông tin đăng nhập và đề thi cho thí sinh, sử dụng mogoreplication để đảm bảo db luôn hoạt động không xảy ra tình trạng chết server db.

* + 1. Kubernetes

Kubernetes là một Platform dùng để tự động hoá việc quản lý, scaling và triển khai ứng dụng dưới dạng container hoá, Kubernetes (K8s) còn gọi là Container orchestration engine. Trong những năm gần đây, nhiều ứng dụng đã thực hiện container hoá bằng cách sử dụng docker. Trên thực tế nhiều ứng dụng sử dụng docker chạy trên môi trường production ngày càng tăng lên. Trên môi trường production, Vì việc cấu trúc hệ thống chạy bằng container chỉ sử dụng docker là rất khó khăn. Cho nên việc sử dụng một Platform gọi là Container orchestration engine như là k8s thì khá phổ biến hiện nay. Ngoài K8s thì còn có docker swarm. Kubernetes là một hệ thống mã nguồn mở để tự động triển khai, scaling, quản lý các container. Kubernetes được xây dựng bởi Google dựa trên kinh nghiệm quản lý sử dụng các container trong khi triển khai một hệ thống quản lý gọi là Borg ( nhiều lúc gọi là Omega). Nó được phát hành vào tháng 6 năm 2014, và thời điểm ra mắt phiên bản 1.0 vào tháng 7/2015 thì nó được chuyển giao cho Cloud Native Computing Foundation (CNCF). Trong bối cảnh có thể nói là K8s đã trở thành Defacto standard thì có sự đối ứng phù hợp của những nhà cung cấp dịch vụ Cloud. Trong khoảng tháng 11/2014 Google Cloud Platform (GCP) đã bắt đầu cung cấp dịch vụ Google Container Engine (GKE, sau này là Google Kubernetes Engine). Để sử dụng k8s đã được quản lý, GKE là một lựa chọn, tuy nhiên tháng 2/2017 Microsoft Azure cũng đã release Azure Container Service (AKS), Tháng 11/2017 Amazon Web Service (AWS) cũng cho ra đời Amazon Elastic Container Service for Kubernetes(Amazon EKS)

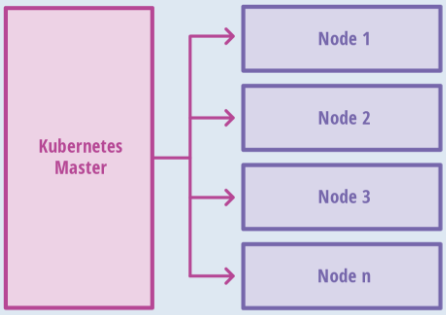
Kubernetes  ở cấp độ cơ bản, là một hệ thống quản lý các ứng dụng được chứa qua một cluster của các node. Theo nhiều cách, Kubernetes được thiết kế để giải quyết sự ngắt kết nối giữa cách mà cơ sở hạ tầng hiện đại, nhóm được thiết kế và một số giả thuyết mà hầu hết các ứng dụng và dịch vụ có về môi trường của chúng. Hầu hết các công nghệ phân cụm đều cố gắng cung cấp một nền tảng thống nhất để triển khai ứng dụng. Người dùng không cần phải quan tâm nhiều đến nơi công việc được lên kế hoạch. Đơn vị công việc được trình bày cho người dùng ở cấp độ "dịch vụ" và có thể được thực hiện bởi bất kỳ node thành viên nào.

Tuy nhiên, trong nhiều trường hợp, điều quan trọng là cơ sở hạ tầng cơ bản trông như thế nào. Khi Định tỉ lệ một ứng dụng, quản trị viên quan tâm rằng các trường hợp khác nhau của một dịch vụ không được gán cho cùng một máy chủ. Mặt khác, nhiều ứng dụng phân phối được xây dựng với việc chia tỷ lệ trong trí nhớ thực sự được tạo thành từ các dịch vụ thành phần nhỏ hơn. Các dịch vụ này phải được lên lịch trên cùng một máy chủ như các thành phần liên quan nếu chúng được cấu hình theo một cách bình thường. Điều này trở nên quan trọng hơn khi chúng dựa vào các điều kiện mạng cụ thể để truyền đạt một cách thích hợp. Mặc dù có thể với hầu hết các phần mềm phân cụm để thực hiện các loại quyết định lên lịch này, hoạt động ở cấp dịch vụ cá nhân không phải là lý tưởng. Các ứng dụng bao gồm các dịch vụ khác nhau vẫn nên được quản lý như một ứng dụng duy nhất trong hầu hết các trường hợp. Kubernetes cung cấp một lớp trên cơ sở hạ tầng để cho phép loại quản lý này.

**Các khái niệm trong kubernetes**

**Kubernetes Cluster**

Một cluster là một tập các máy vật lý hay máy ảo được sử dụng bởi Kubernetes để chạy các ứng dụng. Kubernetes Cluster bao gồm Nodes(master) và Nodes(worker).



Hình 2- 11: Mô hình node trong kubernetes

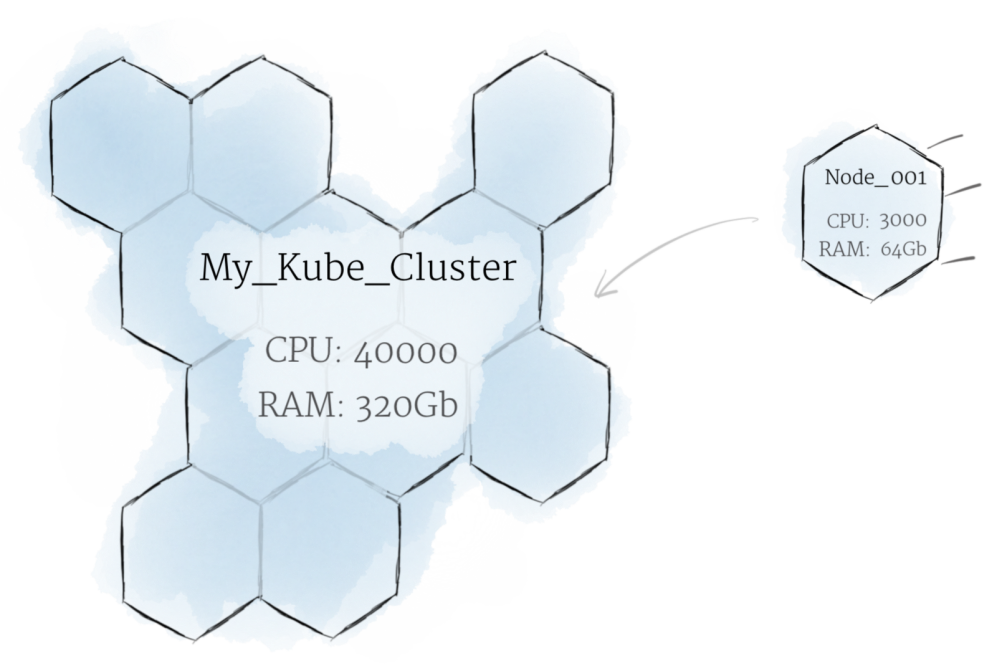
**Nodes**

Một node là một máy ảo hoặc máy vật lý chạy Kubernetes. Nodes hay còn gọi là docker host



Hình 2- 12: Mô tả các node trong kubernetes

Khi làm việc chỉ với một máy, sẽ gặp các vấn đề như không đủ tài nguyên, không thể backup khi sự cố xảy ra, vv. Ta nghĩ đến vấn đề join các máy tính riêng lẽ tạo thành 1 cụm, ta có Kubernetes Cluster như hình dưới đây:



Hình 2- 13: Mô hình pod trong kubernetes

Pods

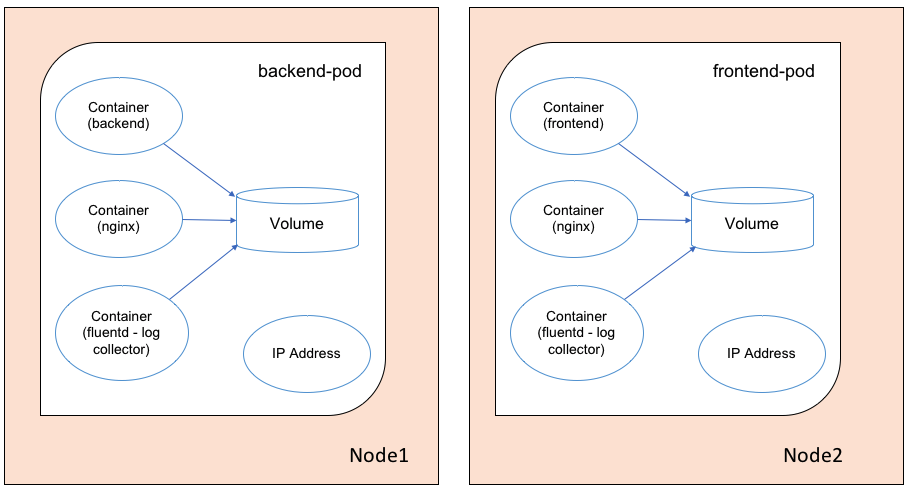
Pod là 1 nhóm (1 trở lên) các container thực hiện một mục đích nào đó, như là chạy software nào đó. Nhóm này chia sẻ không gian lưu trữ, địa chỉ IP với nhau. Pod thì được tạo ra hoặc xóa tùy thuộc vào yêu cầu của dự án. Đơn vị nhỏ nhất của ứng dụng chạy Kubernetes đó là container, nhưng đơn vị quản lý cơ bản nhất thì là Pods. Ví dụ: Có một ứng dụng sau

* backend: có 3 container: backend (chứa code php backend), nginx(cấu hình dành cho backend), fluentd (tool gửi log)
* frontend: có 2 container: frontend (chứa code php frontend), nginx (cấu hình dành cho frontend), fluentd (tool gửi log)

Thiết kế theo Kubernetes chia làm 2 pods:

* backend-pod
* frontend-pod

Cụ thể như hình:

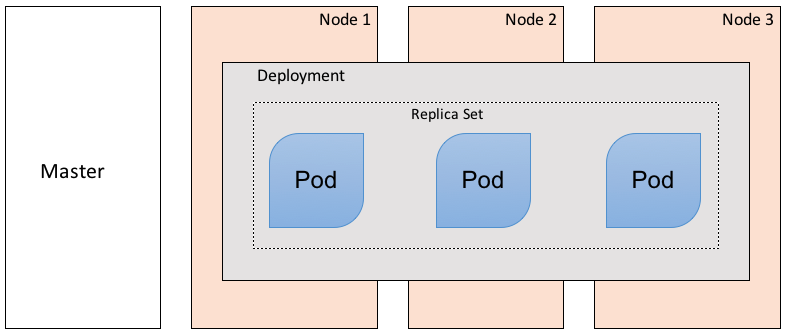


Hình 2- 14: pods trong node kubernetes

Pods có thể nằm trên cùng 1 node vì chúng là 2 ứng dụng độc lập có ip riêng chỉ cần chỉ định vùng lưu trữ cho mỗi Pods thì không ảnh hưởng đến nhau.

Deployment

Deployment trong Kubernetes dùng để phân phối các pods cho các nodes cụ thể, mang tính dự phòng. Ví dụ như hình trên, nếu lỡ Node 1 bị sập (chết server) thì ứng dụng backend sẽ mất. Replica Set đảm nhận vai trò tạo ra số lượng Pods giống nhau dựa vào yêu cầu và luôn luôn duy trì số lượng đó. Giả sử quy định replicas: 3 thì nó sẽ tạo ra 3 Pods giống nhau, giả sử 1 pod ở Node (server) nào đó bị sự cố, nó sẽ tạo ra 1 Pod mới để duy trì số lượng là 3 như yêu cầu lúc đầu. Vậy Deployment là việc định nghĩa policy update/phân phối của Replica Set. Nghĩa là sao, là nó sẽ quy định Pods gồm những container nào, các Pods được tạo ra sẽ phân phối đến các Nodes nào trong Kubernetes.



Hình 2- 15: Mô hình deployment trong kubernetes

Tổng kết lại:

* Kubernetes Cluster là 1 cụm các Nodes: Master Nodes và Worker Nodes
* Nodes là một máy vật lý hoặc máy ảo, mỗi Node bao gồm 1 hoặc nhiều Pods
* Pods là đơn vị quản lý container, mỗi Pod có thể có 1 hoặc nhiều container.
* Replica Set Dùng để tạp ra nhiều Pods giống nhau
* Deployment dùng để quản lý việc tạo Pod và phân bổ các Pod vào các Node quy định.

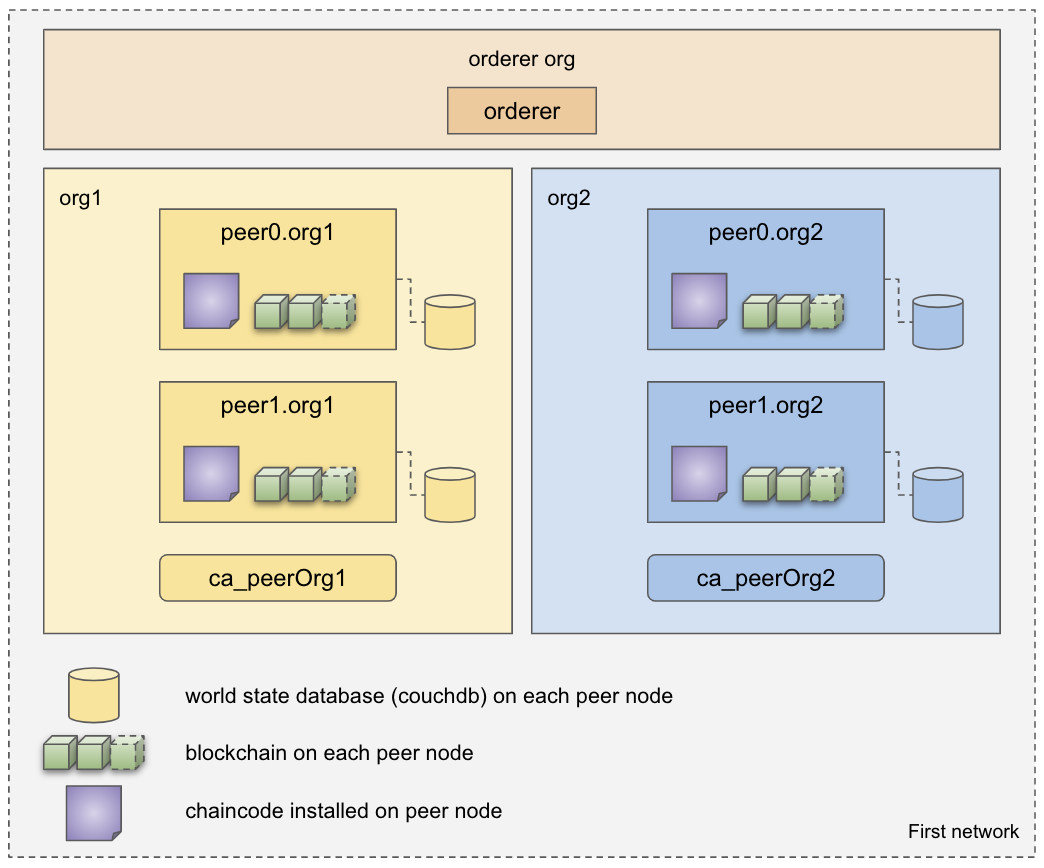
Trong phạm vi đồ án kubernetes sẽ được sử dụng để đảm bảo hệ thống thi luôn ở trạng thái ổn định nhất phòng tránh được các trường hợp chết server.

* 1. Xây dựng mạng lưới fabric blockchain

Phần cơ sở hạ tầng sẽ bao gồm:

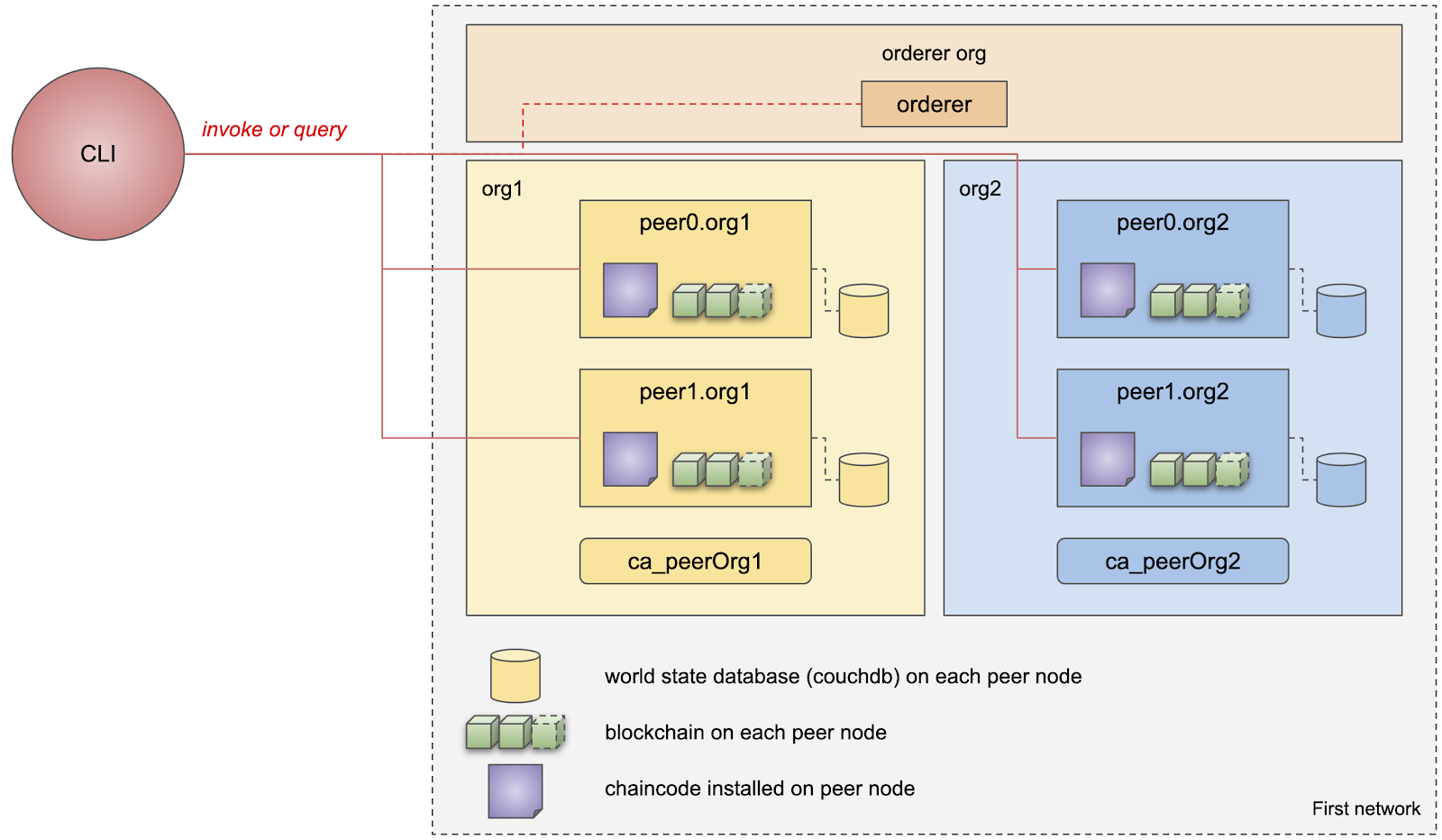
* Hai tổ chức, Org1 và Org2, mỗi tổ chức có hai nút (peer0 và peer1). Do đó, trong First Network, chúng ta có tổng cộng bốn nút.
* Một tổ chức orderer và một nút orderer sử dụng SOLO làm phương thức orderer.
* Mỗi nút ngang hàng chạy một cơ sở dữ liệu CouchDB để lưu trữ thông tin trong sổ cái (có thể thay bằng LevelDB)
* Mỗi Org1 và Org2 có thể đi kèm với Tổ chức phát hành chứng chỉ (CA).
* Giao diện dòng lệnh (CLI) để máy client có thể tương tác với mạng Fabric.

Tất cả các thành phần được triển khai dưới dạng các Container và chạy trên một máy chủ.



Hình 2- 16: Mô hình mạng fabric

Xây dựng các phương thức hoạt động trên chaincode:



Hình 2- 17: Mô hình client tương tác với mạng blockchain

Các phương thức hoạt động:

* Truy vấn tất cả bản ghi thí sinh trên sổ cái
* Truy vấn thí sinh theo ID
* Thêm mới một bản ghi vào sổ cái
* Truy vấn tất cả bản ghi thí sinh trên sổ cái để xem bản ghi vừa được thêm mới
* Thay đổi điểm của một thí sinh bất kỳ nào đó
* Truy vấn lại sổ cái và xem sự thay đổi

# CHƯƠNG 3. LẬP TRÌNH VÀ ĐÁNH GIÁ HỆ THỐNG

Trong chương này, đồ án sẽ trình bày các bước xây dựng hệ thống, việc thực nghiệm từng giai đoạn và toàn bộ ứng dụng tổ chức thi từ đó rút ra các nhận xét, đánh giá. Trình bày cách làm và thiết kế chi tiết cho từng phần.

* 1. Lập trình

Hyperledger Fabric đi kèm với rất nhiều ví dụ trên mạng, do chính Hyperledger phát triển , trong đó Fabcar là một trong số đó, nó được dùng để hướng dẫn trong tài liệu chính thức của Hyperledger Fabric. Ứng dụng Fabcar gồm có 2 phần chính: Chaincode được triển khai và thực thi bên trong mạng fabric và ứng dụng client để người dùng tương tác với chaincode được triển khai trên mạng fabric này. Phần demo sẽ dựa trên ứng dụng fabcar này và chạy trên mạng firtnetwork có sẵn.

**Mô tả hệ thống**

Hệ thống sẽ gồm 3 phần riêng biệt backend, frontend, mạng fabric blockchain. Hai thành phần backend và frontend sẽ được đặt vào trong một mạng riêng để đảm bảo từ bên ngoài không truy cập vào được. Mỗi thành phần trong 2 thành phần này sẽ được triển khai trên nhiều server để cân bằng tải đảm bảo tính ổn định của hệ thống( sử dụng công nghệ kubernets) trong demo thì sẽ được triển khai dưới dạng docker. Phần mạng fabric blockchain sẽ xây dựng dựa trên mạng fadcar có sẵn và cần xây dựng thêm các api để thực hiện các hành động như thêm sửa xóa thí sinh. Hệ thống sẽ hoạt động theo các bước như sau:

Bước 1: Khi thí sinh đăng nhập vào hệ thống, backend sẽ lấy phần username của thí sinh đó và query trên mạng blockchain để lấy thêm các thông tin chi tiết trả về và hiển thị những thông tin cần thiết nên giao diện. Sau đó sử dụng thông tin mã đề để query đề thi trong mongo db.

Bước 2: Thí sinh đọc lại thông tin được trả về và xác nhận chuyển sang bước tiếp theo, nếu có sai sót thì báo lại với giám thị

Bước 3: Thí sinh đọc hướng dẫn làm bài và chuyển qua bước tiếp theo

Bước 4: Sau khi đã xác nhận hết thông tin thí sinh báo giám thị nhập mã otp để làm bài thi

Bước 5: Thí sinh chọn môn và làm bài

Bước 6: Thí sinh làm bài khi là song thì kick vào nút nộp bài, hết thời gian hệ thống sẽ tự động nộp bài

Bước 7: Backend nhận phần làm bài và chấm điểm sau đó sẽ gọi api update lên mạng blockchain

* + 1. Lập trình mạng blockchain

Cơ sở dữ liệu

Thiết kế cơ sở dữ liệu để lưu trên mạng blockchain, cơ sở dữ liệu sẽ có format như sau:

|  |
| --- |
| {  HoTen: Nguyễn Minh Vũ  DiaChi: Hải Dương  MaDuThi: B15DCAT195  Cmnd: 142882857  MaDe: k8sa  } |

Hình 3- 1: Định dạng cơ sở dữ liệu trong mạng blockchain

Tương tác với chaincode

Một chaincode sẽ phải implement interface bao gồm 2 funtion: *Init* và *Invoke*

Hàm *Init* sẽ được gọi mỗi khi chaincode được ***instantiate*** hoặc ***upgrade*** trong channel. Nó có dạng như sau:

|  |
| --- |
| func(t\*SimpleChaincode)Init(stubshim.ChaincodeStubInterface)pb.Response{}shim.ChaincodeStubInterface) pb.Response {} |

Hình 3- 2: Hàm init trong chaincode

Hàm *Invoke* sẽ được gọi mỗi khi ta muốn query dữ liệu hoặc tạo transaction trong Fabric.

|  |
| --- |
| func (t \*SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {} |

Hình 3- 3: Hàm invoke trong chaicode

Fabric không có phân biệt call (query) hay send (transaction) như trong Ethereum, mà tất cả đều là Invoke.

Chain code sẽ có thêm các hàm:

* *queryAll():* Trả về tất cả các bản ghi trong mạng blockchain
* *queryOnce()*: Trả về một bản ghi theo id
* *createdContestant()*: Thêm thí sinh vào mạng blockchain
* *changePoin*(): Thay đổi điểm của thí sinh theo id

Sau đây là các file js để tương tác với chaincode:

Hai file query.js và invoke.js thực hiện gọi các hàm trong chaincode nhằm truy vấn hoặc thay đổi trạng thái của sổ cái trong kênh. File query.js có hai hàm được định nghĩa trong chaincode để truy vấn dữ liệu từ sổ cái : *queryAll()* và *queryOnce()*. Logic của file query.js:

* Required package fabric-network.
* Lấy thông tin chi tiết về việc triển khai First Network. Nó chứa thông tin về điểm truy cập của Fabric CA (nằm ở file first-network/connection-org1.json)
* Kiểm tra xem user1 đã được đăng ký chưa (trong folfer wallet). Nếu chưa, không cần thực hiện thêm hành động nào.
* Tạo gateway để kết nối với nút đã được cấu hình.
* Sử dụng gateway để chỉ định kênh (mychannel) và chaincode (fabcar) cần kết nối tới.
* Sử dụng API *\_evaluateTransaction()\_* với tham số cần truy vấn.
* Nhận kết quả trả về

|  |
| --- |
| // queryOnce transaction - requires 1 argument, ex: (queryOnce, B15DCAT195)  // queryAll transaction - requires no arguments, ex: (queryAll)  const result = await contract.evaluateTransaction(‘queryAll’);  console.log(`Transaction has been evaluated, result is: ${result.toString()}`); |

Hình 3- 4: Hàm lấy thông tin của một thí sinh

File *invoke.js có* hai hàm được định nghĩa trong chaincode để thay đổi trạng thái của sổ cái trong kênh: *createdContestant*() và *changePoin* (). File invoke.js được viết để có thể gọi 2 hàm này. Logic của *invoke.js*:

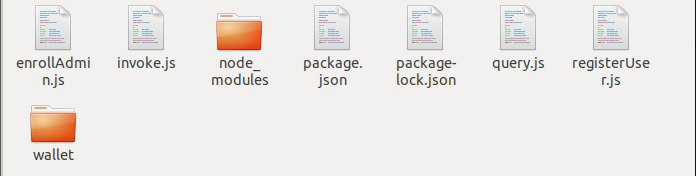
* Required package fabric-network.
* Lấy thông tin chi tiết về việc triển khai First Network. Nó chứa thông tin về điểm truy cập của Fabric CA (nằm ở file first-network/connection-org1.json)
* Kiểm tra xem user1 đã được đăng ký chưa (trong folfer wallet). Nếu chưa, không cần thực hiện thêm hành động nào.
* Tạo gateway để kết nối với nút đã được cấu hình.
* Sử dụng gateway để chỉ định kênh (mychannel) và chaincode (fabcar) cần kết nối tới.
* Sử dụng API *\_submitTransaction()\_* với tham số cần truy vấn.
* Ngắt kết nối gateway sau khi giao dịch được xử lý.

Gọi hàm *creatContestant* để tạo bản ghi mới:

|  |
| --- |
| // createContestant transaction - requires 5 argument, ex: ('Nguyễn Minh Vũ', 'Hải Dương', 'B15CQAT195', '142882857', 'k8s01')  // changePoin transaction - requires 2 args , ex: ('changePoin', 'B15CQAT195', '9')  await contract.submitTransaction('createContestant', 'Nguyễn Minh Vũ', 'Hải Dương', 'B15CQAT195', '142882857', 'k8s01');  await contract.submitTransaction('changePoin''B15CQAT195', '9'); |

Hình 3- 5: Hàm thêm mới thí sinh và thay đổi thông tin thí sinh

Những file và thư mục cần thiết trong mạng fabric blockchain



Hình 3- 6: Những file và thư mục cần thiết trong mạng fabricblockchain

Thư mục node\_moduled cài đặt từ file pakage.json chứa những thư viện cần thiết để chạy các file js trong mạng. File package-lock.json chứa phiên bản các thư viện cần được cài. File enrollAdmin.js để đang kí admin vào trong mạng thông tin sau khi đăng kí được lưu ở thư mục wallet. File registerUse để đang kí user vào trong mạng thông tin cũng được lưu trong thư mục wallet

Xây dựng bộ Api để tương tác với mạng blockchain

* Lấy thông tin một thí sinh:

Url: <https://domain/api/query/id> trong đó id là mã dự thị của thí sinh

Method: Get

Header: Authorization: ‘mã bí mật được cấp’

Respone: "{"CMT":"142887999","DiaChi":"Hải Phòng","Diem":"0","HoTen":"Mai ThịLan","MaDe":"MV02","MaDuThi":"B15DCAT196","NgaySinh":"20/4/1997"}

* Lấy danh sách thí sinh:

Url: https://domain/api/queryall

Method: Get

Header: Authorization: ‘mã bí mật được cấp

Respone: Danh sách thí sinh với thông tin như trên

* Thêm thí sinh:

Url: https://domain/api/create

Method: POST

Param: {"{“CMT”:"142887999","DiaChi":"Hải Phòng","Diem":"0","HoTen":"Mai ThịLan","MaDe":"MV02","MaDuThi":"B15DCAT196","NgaySinh":"20/4/1997"}

Header: Authorization: ‘mã bí mật được cấp’

Respone: {status: True, mss: ‘Thêm thành công’}

* Cập nhật điểm cho thí sinh:

Url: https://domain/api/create

Method: POST

Param: {“MaDuThi”: “B15DCAT195”, “Diem”: “9”}

Header: Authorization: ‘mã bí mật được cấp’

Respone: {status: True, mss: ‘Cập nhật thành công’}

* + 1. Xây dựng api backend

1. **Thiết kết cơ sở dữ liệu**

Cơ sở dữ liệu được dùng: Mongodb

Mô hình cơ sở dữ liệu gồm hai bảng là user (chứa thông tin đăng nhập của thí sinh và

exam (chứa thông tin đề thi):

|  |
| --- |
| {  Username: B15DCAT195  Password: 20fb05713e46ca7ed1b1e3675f35a52b  Secret\_key: BHGFSEAYDHETDJSY  Create\_time: 157337509  Token: udfadfafa-adfasdf-fasdfasdf-ddwdw (Token được cấp sau khi đăng nhập)  Token\_otp: jfdqwr-rrwxgawrf-fawfv (Token opt được cấp sau khi nhập đúng mã otp)  } |

Hình 3- 7: Mô tả cơ sở dữ liệu của thí sinh

Giải thích các tham số:

* Username: Tên đăng nhập là mã dự thi
* Password: Mật khẩu đã mã hóa md5 sẽ được cấp khi vào phòng thi
* Secret\_key: Secret key để tạo mã otp
* Token: mã được cấp khi đăng nhập thành công để có thể gọi được api
* Token\_otp: mã được cấp khi nhập đúng otp dùng cùng với Token để có thể gọi api

|  |
| --- |
| {  Id: k8sas  Exam: [{Question: The apartment on the lower floors cost less because thay are more exposed …… dust and the noise of traffic, AnsA: to, AnsB: without, AnsC: from, AnsD: againts, TrueAns: to},{…},..] } |

Hình 3- 8: Mô tả cơ sở dữ liệu đề thi

Giải thích tham số:

* Id: mã đề, sau khi đăng nhập thành công thì hệ thống sẽ lấy thông tin mã đề của thí sinh trên mạng blockchain và truy vấn đề thi cho thí sinh
* Exam: Đề thi

1. **Xây dựng api backend**

Api được viết bằng python gồm có các api sau

* Đăng nhập

Url: https://domainbackend/v1/login

Method: POST

Param: { username: B15DCAT195, password: ptit2019, step\_1 : yes}

Respone: {status: yes, token: dfasdfas-dffffff0-sssssfff-adddd}

* Xác thực otp

Url: https://domainbackend/v1/login

Method: POST

Param: {otp:251098, token: dfasdfas-dffffff0-sssssfff-adddd, step\_2 : yes}

Respone: {status: yes, token\_otp: czxczzdf-sdfasdf-bbxcvnfdgdfg}

* Lấy đề thi:

Url: https://domainbackend/v1/get\_exam

Method: GET

Header: { token: dfasdfas-dffffff0-sssssfff-adddd, token\_otp: czxczzdf-sdfasdf-bbxcvnfdgdfg }

Respone: {status: yes, exam: Đề thi như đã phân tích ở phần cơ sở dữ liệu}

* Nộp bài:

Url: https://domainbackend/v1/receve

Method: GET

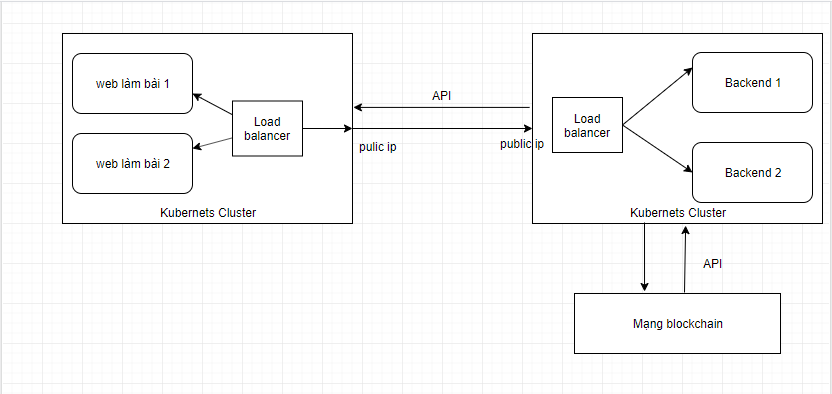
Header: { token: dfasdfas-dffffff0-sssssfff-adddd, token\_otp: czxczzdf-sdfasdf-bbxcvnfdgdfg }

Param: {answer: {1:to, 2:from …}}

Respone: {status: yes, mssage: Đã nộp bài thành công}

* + 1. **Xây dựng phần hệ thống**

Trong phạm vi đề mô hệ thống sẽ được thiết kế như sau:



Hình 3- 9: Mô hình hệ thống sử dụng kubernetes

Phần frontend sẽ được đóng gói thành docker image sau đó dùng kubernetes đẩy lên 2 server, trên thực tế với lượng thí sinh lớn sẽ cần nhiều hơn. Phần cân bằng tải sẽ do thuật toán của kubernetes đảm nhận và chúng ta không cần quan tâm tới. Vd khi có request gửi tới web thì kubernetes sẽ điều hướng requet đó tới một trong hai web như hình trên và sự điều hướng này là lần lượt. Quá trình xây dựng như sau.

**Phần backend:**

Viết dockerfile:

|  |
| --- |
| FROM python:3.7-alpine  #RUN echo "deb http://ftp.us.debian.org/debian/ jessie main non-free contrib" > /etc/apt/sources.list && \  # echo "deb http://security.debian.org/ jessie/updates main non-free contrib" >> /etc/apt/sources.list  #  #RUN apt-get update && \  # apt-get clean && \  # rm -rf /var/lib/apt/lists/\*  # Cài code API  RUN mkdir /code  RUN mkdir /code/api  COPY api/ /code/api/  COPY api/requirements.txt /code/api/  WORKDIR /code/api  #RUN mkdir /code  #RUN mkdir /code/api  #COPY api/ /code/api/  RUN pip install --trusted-host pypi.python.org -r requirements.txt  #RUN apt-get update && apt-get install net-tools && apt-get install -y iputils-ping  RUN pip install -r requirements.txt  CMD ["python", "run.py"] |

Hình 3- 10: Dockerfile cho code backend

Viết file docker-compoe.yml để build thành image:

|  |
| --- |
| version: '3'  services:  api:  container\_name: api\_flask  build:  context: ../  dockerfile: docker/Dockerfile  ports:  - "0.0.0.0:8000:8000"  environment:  FLASK\_ENV: development  restart: unless-stopped |

Hình 3- 11: Dockercompose file cho backend

Sau khi build thành image thì sẽ viết file config cho kubernetes để đẩy lên sever:

Tạo resource để thêm 1 pod: Tên file : backend-pod.yaml

|  |
| --- |
| apiVersion: v1  kind: Pod # 1  metadata:  name: server # 2  labels:  app: server  spec: # 3  containers:  - image: vunm/docker\_api:01 # 4  name: server # 5  ports:  - containerPort: 8000 |

Hình 3- 12: File config kubernetes pod một cho backend

Tạo resource để thêm 1 pod nữa: Tên file : backend-pod2.yaml

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: server2 # => Thay đổi  labels:  app: server  spec:  containers:  - image: vunm/docker\_api:01  imagePullPolicy: Never  name: server2  ports:  - containerPort: 8000 |

Hình 3- 13: File config kubernetes pod hai cho backend

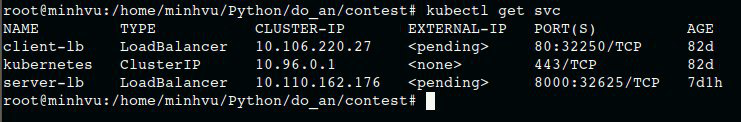
Sau khi tạo xong file thì có thể chạy lệnh *kubectl create* để chạy hai pod trên

Cuối cùng là tạo file service để quy định xem service sẽ kết nốt tới pod nào:

|  |
| --- |
| apiVersion: v1  kind: Service # 1  metadata:  name: server-lb  spec:  type: LoadBalancer # 2  ports:  - port: 8000 # 3  protocol: TCP # 4  targetPort: 8000 # 5  selector: # 6  app: server |

Hình 3- 14: File service kubernetes cho backend

Chạy lệnh *kubectl create –f* để tạo service kết quả thu được như hình dưới đây:



Hình 3- 15: Kết quả tạo service backend

**Phần frontend**

Phần tạo service tương tự như backend nhưng ở phần frontend cần cấu hình thêm nginx để có thể chạy được thành web. Sau đây là file config nginx.conf

|  |
| --- |
| server {  listen 8080;  root /code;  index index.html;  location / {  include /etc/nginx/mime.types;  try\_files $uri $uri/ @rewrites;  }  location @rewrites {  include /etc/nginx/mime.types;  rewrite ^(.+)$ /index.html last;  }  location ~ \.css {  include /etc/nginx/mime.types;  add\_header Content-Type text/css;  }  location ~ \.js {  include /etc/nginx/mime.types;  add\_header Content-Type application/x-javascript;  }  location ~\* \.(jpg|jpeg|png|gif|ico|css|js)$ {  include /etc/nginx/mime.types;  expires 1M;  }  } |

Hình 3- 16: File config nginx.conf cho frontend

Giải thích phần cấu hính nginx trên: nginx sẽ lắng nghe ở cổng 8080 sau đó trỏ vào file index.html và trả về giao diện cho người dùng. Location cho phép chúng ta định nghĩa 1 nhóm các thiết lập được áp dụng cho 1 vị trí cụ thể trên website (thể hiện qua URL của website đó). Ở đây có tác dụng load các file js, css và file ảnh.

Tiếp theo là các file docker file và docker compose dùng cho frontend:

|  |
| --- |
| FROM nginx:latest  RUN mkdir /code  WORKDIR /code  COPY dist /code  ADD nginx.conf /etc/nginx/conf.d/default.conf  EXPOSE 8080  CMD ["nginx", "-g", "daemon off;"] |

Hình 3- 17: File dockerfile cho frontend

File này có tác dụng thay cấu hình file nginx đã nêu ở trên vào file config mặc định của nginx.

|  |
| --- |
| version: '3'  services:  app:  build: ./  container\_name: manager\_app  volumes:  - ./nginx.conf:/etc/nginx/conf.d/default.conf  ports:  - "127.0.0.1:8080:8080" |

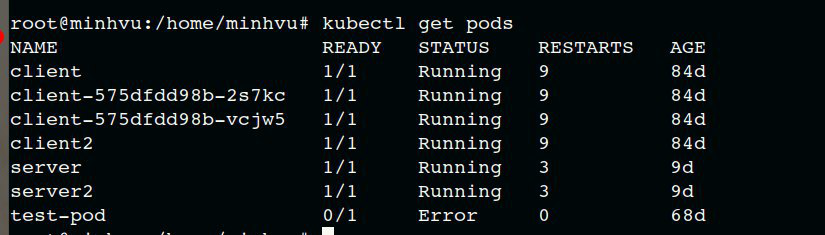
Hình 3- 18: File dockercompose của frontend

Trang web sẽ lắng nghe ở cổng 8080.

Phần tạo service bằng kubernetes sẽ tương tự như phần backend nên sẽ không đề cập lại.

Kết quả sau khi tạo như sau:

Các pods sau khi tạo như sau:

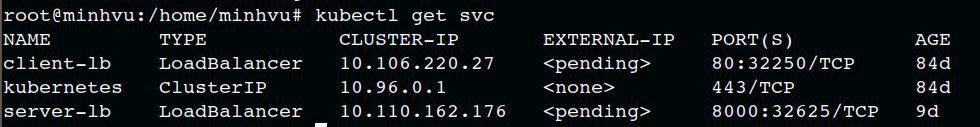


Hình 3- 19: Các pods của frontend và backend sau khi tạo

Giải thích các tham số:

* NAME : Tên của pod (2 pods fronend là clinet và clinet2, 2 pod backend là server và server2)
* READY : Trạng thái sẵn sàng
* STATUS: Trạng thái hiện tại (Running là đang chạy, Error là đang lỗi không chạy được)
* RESTARTS: Số lần restarts
* AGE : Ngày được tạo

Các service sau khi tạo:



Hình 3- 20: Service sau khi tạo bằng kubernetes

Giải thích các tham số:

* NAME : Tên service (ở đây frontend là client-lb, backend là server-lb)
* TYPE: Loại service (ở đây là loadbalancer)
* CLUSTER\_IP: ip của cluter
* ETERNAL-IP : ip ánh xạ ra ngoài mạng
* PORT : port ánh xạ ra ngoài mạng
* AGE: ngày được tạo

Hình 10: Hai service sau khi tạo

**Phần cơ sở dữ liệu backend**

Cơ sở dữ liệu là một phần rất quan trọng vì vậy cần đảm bảo tính ổn định tránh mất kết nỗi hoặc mất cơ sở dữ liệu. Như đã trình bày ở chương 2 thì phần cơ sở dữ liệu sẽ dùng mongodb và dùng mongo replication để đảm bảo dữ liệu luôn ổn định. Trong phần demo này sẽ tạo 3 cơ sở dữ liệu trên 3 docker, 1 cơ sở dữ liệu làm master còn 2 cái còn lại sẽ làm slave. Khi master mất thì lập tức một trong 2 docker còn lại sẽ lên làm master thay thế. Trên thực tế thì có thể làm nhiều hơn. Sau đây là quá trình xây dựng.

Dùng mongodump để lấy file cơ sở dữ liệu đã chuẩn bị. Viết dockerfile:

|  |
| --- |
| FROM mongo:4.0-xenial  RUN mkdir /code  RUN mkdir /code/api  COPY mongo/contest /code/api/  WORKDIR /code/api |

Hình 3- 21: Dockerfile cho mongodb

File này sẽ cop file cơ sở dữ liệu đã được dump ở bước trên vào trong docker.

Viết file docker-compose:

|  |
| --- |
| version: "3"  services:  mongo1:  hostname: mongo1  container\_name: localmongo1  build:  context: ../  dockerfile: mongo/Dockerfile  ports:  - "0.0.0.0:27011:27017"  restart: unless-stopped  entrypoint: [ "/usr/bin/mongod", "--bind\_ip\_all", "--replSet", "rs0" ]  mongo2:  hostname: mongo2  container\_name: localmongo2  image: mongo:4.0-xenial  expose:  - 27017  ports:  - 27012:27017  restart: unless-stopped  entrypoint: [ "/usr/bin/mongod", "--bind\_ip\_all", "--replSet", "rs0" ]  mongo3:  hostname: mongo3  container\_name: localmongo3  image: mongo:4.0-xenial  expose:  - 27017  ports:  - 27013:27017  restart: unless-stopped  entrypoint: [ "/usr/bin/mongod", "--bind\_ip\_all", "--replSet", "rs0" ] |

Hình 3- 22: Docker compose file cho mongodb

Localmongo1, localmongo2, localmongo3: Lần lượt là 3 docker mongo được tạo ra.

Chọn một docker làm master. Dùng lệnh docker exec để chui vào mongo đó sau đó dùng mongorestore file đã được cop vào ở bước trên, chạy lệnh slave 2 docker còn lại ta sẽ thành công. Kết quả thu được như hình sau:



Hình 3- 23: Kết quả replication mongodb

## **Đánh giá**

* + 1. Hệ thống

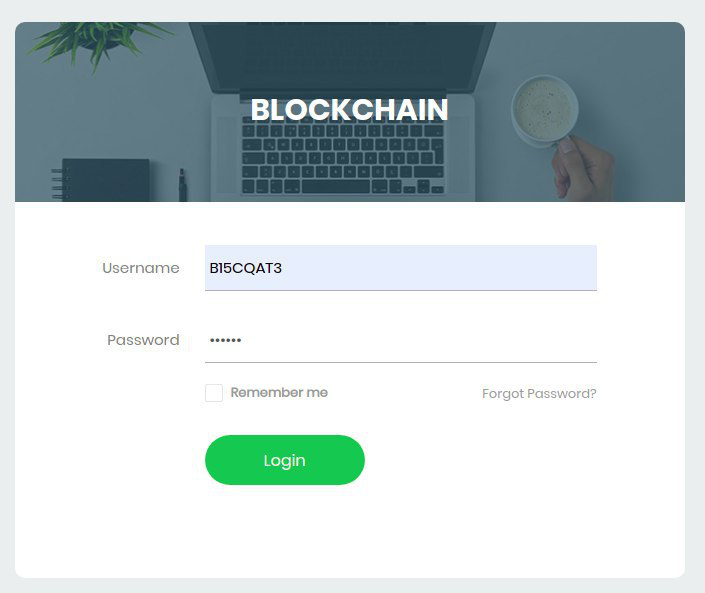
Với hệ thống được thiết kế như trên thì có thể phục vụ một lượng thí sinh tham gia lớn. Các giao tiếp thông qua api đảm bảo tốc độ nhanh ổn định. Trên thực tế có thể tăng các pod để đảm bảo được lượng thí sinh lớn hơn nữa.

Hệ thống có thể ảnh hưởng bởi các tác nhân vật lí bên ngoài như hỏng hóc đường mạng hoặc server.

* + 1. Giao diện chương trình

Để minh họa cho quá trình làm bài thi em đã xây dựng một trang web đơn giản. Trang web cho phép thí sinh làm bài thi trên hình thức trắc nhiệm. Thí sinh làm bài trong thời gian cho phép, hết thời gian thì hệ thống tự động nộp bài. Các bước sẽ được thực hiện theo kịch bản sau:

Bước 1: Đăng nhập

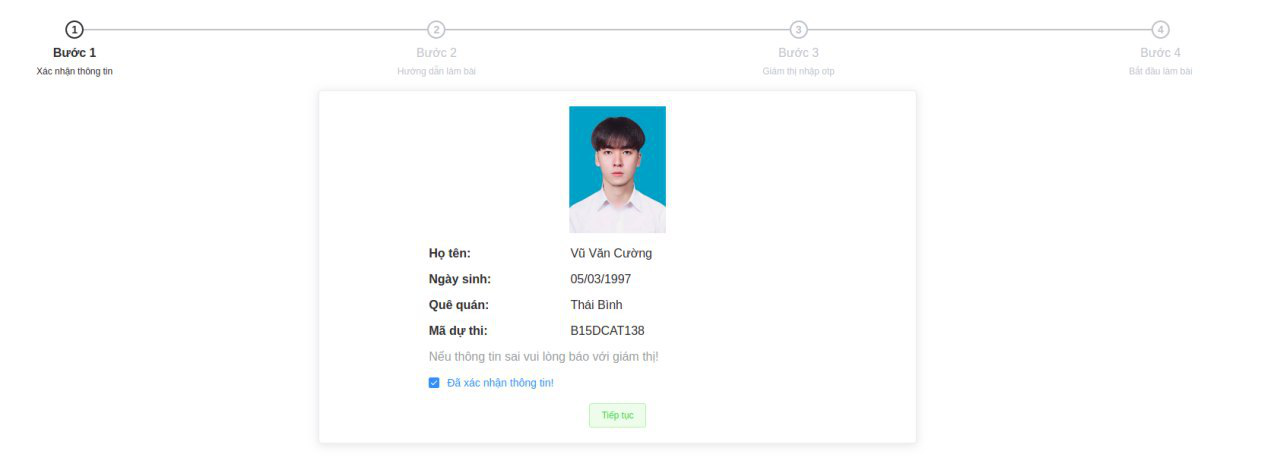


Hình 3- 24: Giao diện phần đăng nhập

Hình 9: Giao diện đăng nhập

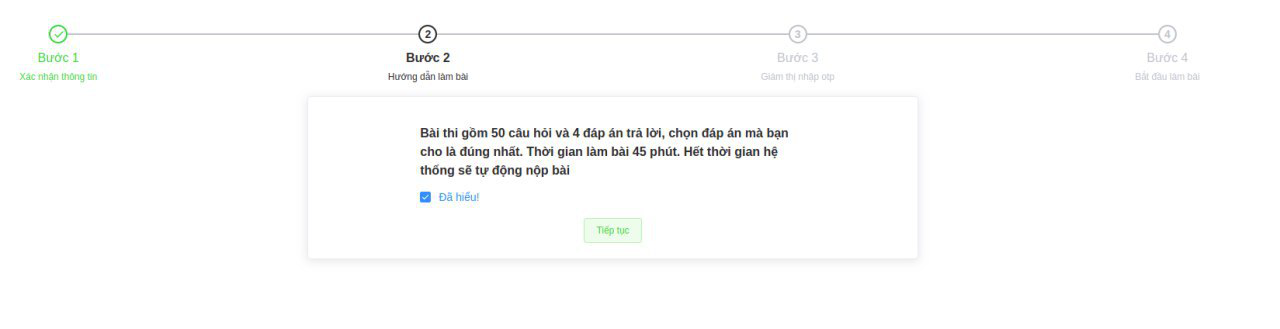
Bước 2: Xác thực thông tin gồm có các giai đoạn sau:

* Xác nhận lại thông tin thí sinh nếu có sai sót báo lại cho giám thị



Hình 3- 25: Giao diện phần xác thực thông tin

* Đọc hướng dẫn làm bài thi



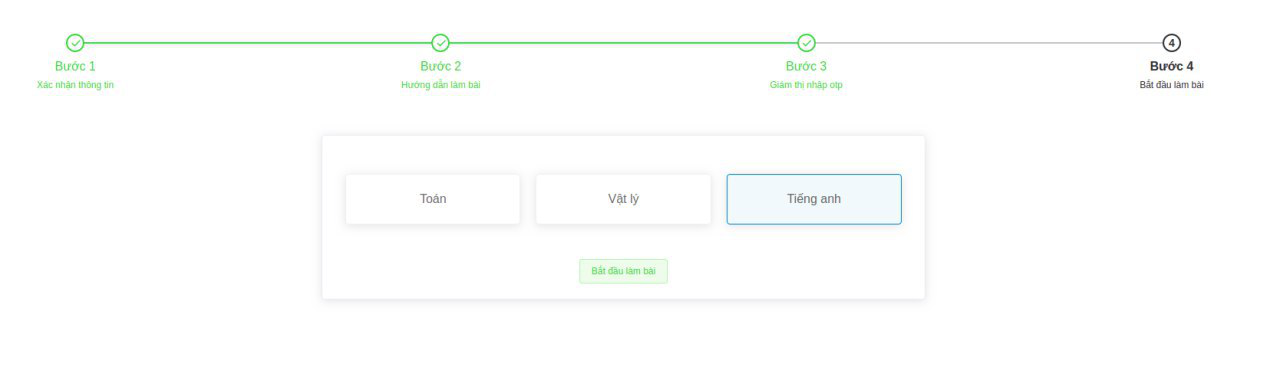
Hình 3- 26: Giao diện hướng dẫn làm bài thi

* Phần nhập mã otp của giám thị



Hình 3- 27: Giao diện xác thực hai bước bằng otp

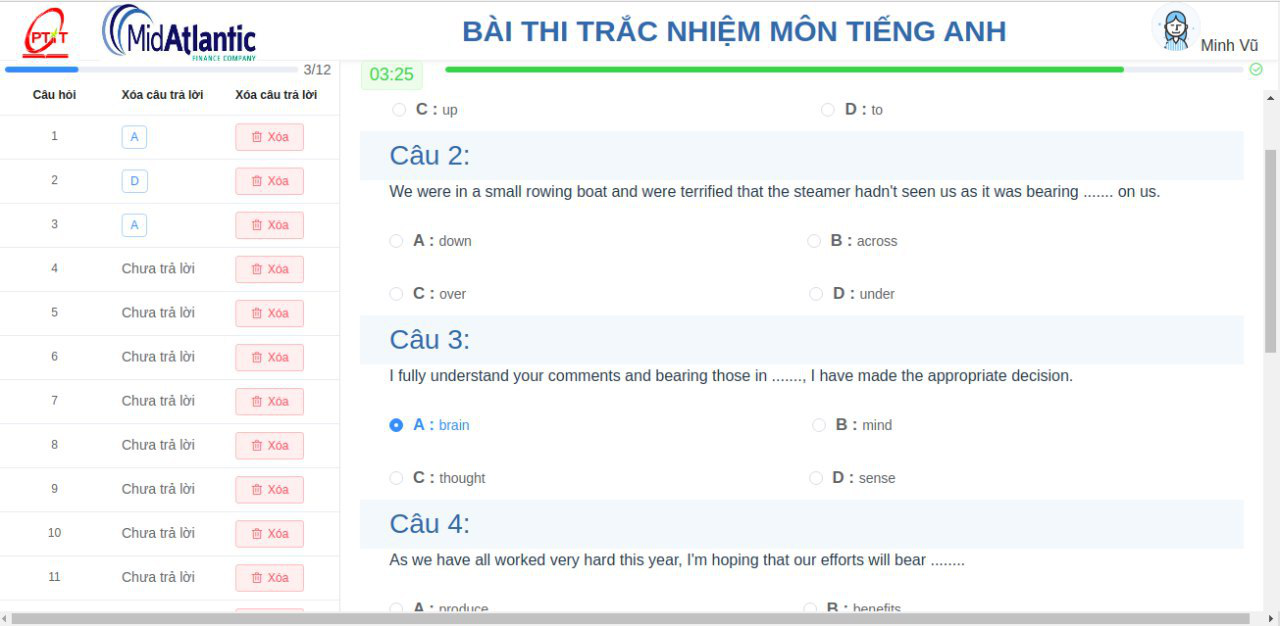
* Chọn môn và bắt đầu làm bài



Hình 3- 28: Giao diện chọn môn thi

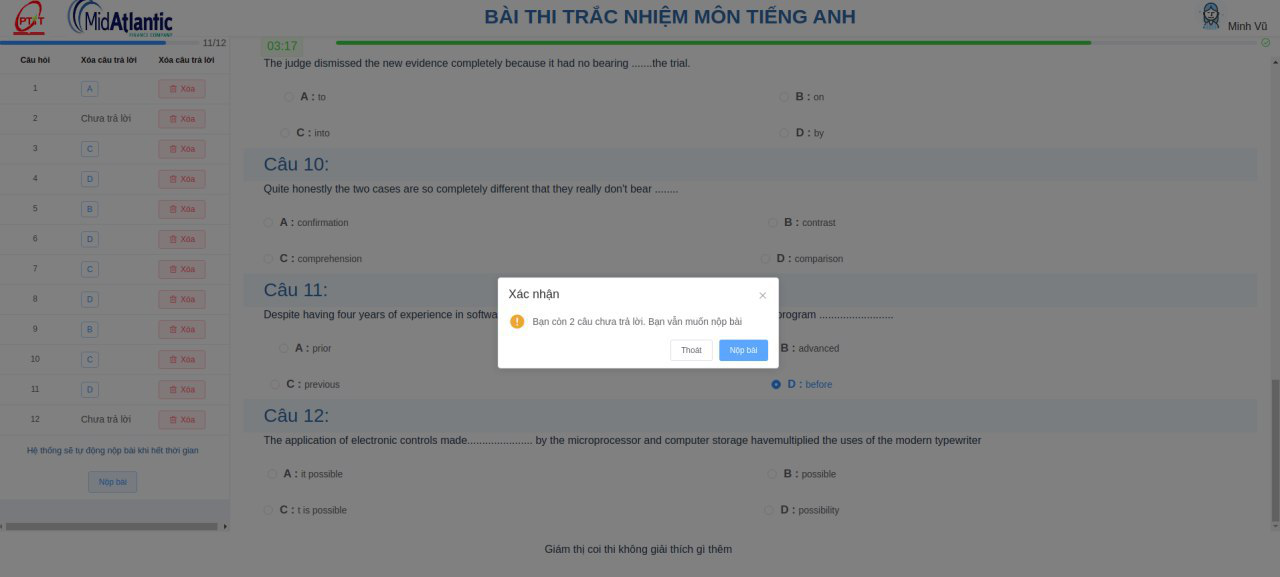
Hình 11: Giao diện chọn môn thi

Bước 3: Bắt đầu làm bài



Hình 3- 29: Giao diện làm bài thi

Bước 4: Nộp bài, nếu hết thời gian hệ thống sẽ tự động nộp bài



Hình 3- 30: Giao diện nộp bài thi

Web được viết bằng ngôn ngữ vuejs

Backend được viết bằng python có sử dụng framework flask.

Chương 3 đã trình bày toàn bộ quá trình xây dựng hệ thống cũng như kết quả thu được. Việc đánh giá và nhận xét các kết quả đó đã cho thấy các ưu và nhược điểm của hệ thống, các thiếu sót cần khác phục và khả năng mở rộng trong tương lai.

* + 1. Đánh giá độ an toàn

1. **Độ an toàn mạng fabric blockchain lưu thông tin thí sinh**

Với đặc điểm bất biến của dữ liệu trên mạng blockchain nói chung và các đặc điểm riêng của mạng private blockchain như:

* Có cá nhân tổ chức điều hành quản lí
* Chỉ có cá nhân sử dụng blockchain mới có thể truy cập và sửa đổi
* Không công khai dữ liệu

Thì dữ liệu về điểm và thông tin thí sinh sẽ được đảm bảo cả về tính bí mật lẫn tính toàn vẹn, khả năng bị sửa thông tin là điều rất khó có thể xảy ra. Vì vậy với phương pháp dùng mạng private blockchain để lưu điểm và các thông tin cá nhân của các thí sinh là lựa chọn hợp lí

1. **Độ an toàn hệ thống làm bài thi**

Với cơ chế otp xác thực 2 bước phần nhập otp của giám thị và cơ chế chỉ login được một lần trên một trình duyệt có thể đảm bảo được độ an toàn về quá trình làm bài, đảm bảo định dang được máy làm bài không xảy ra tình trạng làm bài hộ. Cộng thêm cơ chế https thì có thể đảm bảo tính bí mật và toàn vẹn của dữ liệu trên đường truyền.

# KẾT LUẬN

Sau khi tìm hiểu và nghiên cứu về bài toán ứng dụng blockchain trong thi cử thì đồ án đã đạt được những kết quả sau

Giới thiệu về bài toán ứng dụng blockchain trong thi cử về tính ứng dụng trong thực tế, các đặc điểm cụ thể và mô hình xây dựng, tổ chức.

Tìm hiểu về fabric blockchain để xây dựng một mạng lưới private blockchain lưu thông tin và điểm của thí sinh

Tìm hiểu về các kiến trúc và công nghệ như docker và kubernetes giúp ổn định hệ thống.

Giới thiệu về các framework viết api và web: python, vuejs, nodejs.

Kết hợp các giai đoạn thành một hệ thống thi đầy đủ các quy trình như một kì thi thật.

Tuy vậy, đồ án còn có các hạn chế như:

Hệ thống chị áp dụng được với môn thi trắc nhiệm như toán, lý hóa, đối với môn tự luận thì vẫn phải chấm điểm bằng tay.

# 

# TÀI LIỆU THAM KHẢO

**Tài liệu tiếng Việt:**

1. M N. Bình and N. Đ. Thiện (2013), “Giáo trình cơ sở mật mã học,” Học viện công nghệ bưu chính viễn thông, Hà Nội.
2. Đ. M. Tuấn (2016), “Hệ mật mã khóa công khai dựa trên đường cong Elliptic -Một số ứng dụng,” *Epsil. Mag.*, vol. 9, pp. 17–35.

**Tài liệu tiếng Anh:**

1. M. Gupta, *Blockchain for Dummies, IBM Limited Edition*. John Wiley & Sons, Inc., 2017..
2. L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
3. K. Okupski, “Bitcoin Developer Reference,” *Tech. Univ. Eindhoven, Netherlands*, no. July, 2016.
4. IBM, “Maersk and IBM Introduce TradeLens Blockchain Shipping Solution,” 2018. [Online]. Available: http://newsroom.ibm.com/2018-08-09-Maersk-and-IBM-Introduce-TradeLens-Blockchain-Shipping-Solution
5. Leanne Kemp, “Everledger’s Pioneering Blockchain Work for Diamonds,” *https://www.ibm.com/blogs/think/2018/05/everledger/*, 2018. .
6. D. T. T. Anh, M. Zhang, B. C. Ooi, and G. Chen, “Untangling Blockchain: A Data Processing View of Blockchain Systems,” *IEEE Trans. Knowl. Data Eng.*, vol. 4347, no. c, pp. 1–20, 2018.
7. Cachin Christian (2016). Architecture of the hyperledger blockchain fabric. *IBM Res*.

**Danh mục Website tham khảo:**

1. <https://hyperledger-fabric.readthedocs.io/en/release-1.4/tutorials.html>
2. <https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim>
3. <https://github.com/hyperledger/fabric-gateway-java>
4. <https://github.com/hyperledger/fabric/releases/tag/v1.4.0>
5. <https://github.com/hyperledger/fabric-samples>
6. <https://101blockchains.com/hyperledger-fabric-tutorial/>
7. <https://medium.com/coinmonks/hyperledger-fabric-by-example-part-1-758547e9fb04>
8. <https://medium.com/@mycoralhealth/start-your-own-hyperledger-blockchain-the-easy-way-5758cb4ed2d1>
9. <https://medium.com/coinmonks/hyperledger/home>
10. <https://techblog.vn/xay-dung-blockchain-don-gian-voi-golang-p1-cau-truc-co-ban>
11. <https://viblo.asia/p/bai-1-blockchain-la-gi-bitcoin-la-gi-phan-biet-private-blockchain-voi-public-blockchain-gDVK2GgmZLj>
12. <https://viblo.asia/p/mot-ung-dung-tren-hyperledger-fabric-se-trong-nhu-the-nao-ORNZqGxN50n>