**LỜI CẢM ƠN**

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của gia đình, thầy cô và bạn bè.

Con xin cám ơn Cha Mẹ vì luôn luôn tin tưởng, ủng hộ trong suốt thời gian qua. Cám ơn những người thân luôn bên cạnh, giúp đỡ và hỗ trợ hết mình cho em giúp em đã được những kết quả tốt nhất.

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô ở Khoa Công nghệ thông tin & Truyền thông – Trường Đại học Cần Thơ đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Trong học kỳ này nếu không có những lời hướng dẫn, dạy bảo của các thầy cô thì em nghĩ bài niên luận này của em rất khó có thể hoàn thiện được.

Em xin gởi lời cảm ơn chân thành và sự tri ân sâu sắc đối với thầy Hồ Văn Tú đã nhiệt tình hướng dẫn em hoàn thành tốt niên luận, thầy đã tận tình chỉ bảo, hướng dẫn em, giúp em hoàn thành niên luận của mình.

Trong quá trình thực hiện, khó tránh khỏi sai sót, rất mong các thầy cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy cô để em tích lũy thêm được nhiều kinh nghiệm và phát triển tốt hơn trong tương lai.

Em xin chân thành cảm ơn!

Cần Thơ, ngày 15 tháng 11 năm 2018

Sinh viên thực hiện

**LÝ NHẬT TÂM**

1. MỤC LỤC

[MỤC LỤC i](#_Toc530039542)

[DANH MỤC HÌNH iii](#_Toc530039543)

[DANH MỤC BẢNG iv](#_Toc530039544)

[DANH MỤC KÝ HIỆU CHỮ VIẾT TẮT v](#_Toc530039545)

[TÓM TẮT vi](#_Toc530039546)

[CHƯƠNG 1. TỔNG QUAN 1](#_Toc530039547)

[1.1. ĐẶT VẤN ĐỀ 1](#_Toc530039548)

[1.2. MỤC TIÊU ĐỀ TÀI 1](#_Toc530039549)

[1.3. PHẠM VI ĐỀ TÀI 1](#_Toc530039550)

[1.3.1. Phạm vi chức năng 1](#_Toc530039551)

[1.3.2. Phạm vi người dùng 1](#_Toc530039552)

[1.4. NỘI DUNG CHÍNH CỦA NIÊN LUẬN 2](#_Toc530039553)

[1.4.1. Tổng quan 2](#_Toc530039554)

[1.4.2. Cơ sở lý thuyết 2](#_Toc530039555)

[1.4.3. Nội dung và kết quả nghiên cứu 2](#_Toc530039556)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc530039557)

[2.1. BLOCKCHAIN 3](#_Toc530039558)

[2.1.1. Cơ chế đồng thuận trong blockchain: 3](#_Toc530039559)

[2.1.2. Đặc tính của Blockchain: 4](#_Toc530039560)

[2.2. BOOTSTRAP: 4](#_Toc530039561)

[2.3. ASP.NET CORE 2.0 5](#_Toc530039562)

[2.3.1. Khái niệm: 5](#_Toc530039563)

[2.3.2. Cấu trúc Project 6](#_Toc530039564)

[2.4. MVC 7](#_Toc530039565)

[2.5. CODE FIRST MIGRATION 8](#_Toc530039566)

[2.6. REPOSITORY PATTERN 8](#_Toc530039567)

[2.7. GENERIC REPOSITORY PATTERN 9](#_Toc530039568)

[2.8. DEPENDENCY INJECTION 9](#_Toc530039569)

[CHƯƠNG 3. NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU 11](#_Toc530039570)

[3.1. Các thành phần của Blockchain: 11](#_Toc530039571)

[3.2. Cài đặt 11](#_Toc530039572)

[3.2.1. Class block 11](#_Toc530039573)

[3.2.2. Class Blockchain: 12](#_Toc530039574)

[3.2.3. Class Transaction: 15](#_Toc530039575)

[3.2.4. Class Program: 15](#_Toc530039576)

[3.3. Kết quả 16](#_Toc530039577)

[CHƯƠNG 4. KẾT LUẬN VÀ ĐỀ NGHỊ 18](#_Toc530039578)

[4.1. KẾT QUẢ ĐẠT ĐƯỢC 18](#_Toc530039579)

[4.2. HƯỚNG PHÁT TRIỂN 18](#_Toc530039580)

[CHƯƠNG 5. TÀI LIỆU THAM KHẢO 19](#_Toc530039581)

1. DANH MỤC HÌNH

[Hình 1. Mô hình MVC 7](#_Toc530039314)

1. DANH MỤC BẢNG

[Bảng 1. Cấu trúc project 6](#_Toc530039391)

1. DANH MỤC KÝ HIỆU CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| **Ký hiệu/chữ viết tắt** | **Diễn giải** |
| ASP | Active Server Pages |
| POW | Proof Of Work |
| POS | Proof Of Stake |
| DI | Dependency Injection |
| P2P | Peer-to-peer |
| SHA | Secure Hash Algorithm |
| MVC | Model-View-Controller |

1. TÓM TẮT

Blockchain là chủ đề đang vô cùng nóng trên toàn cầu hiện nay. Nó cùng với [Bitcoin](https://quantrimang.com/s/?q=bitcoin) và tiền kỹ thuật số trở thành đề tài bàn luận trên rất nhiều mặt báo và trong những cuộc trò chuyện của mọi người. Tuy nhiên, khi nói về blockchain vẫn còn nhiều tranh cãi. Có người lo lắng rằng Bitcoin có thể chỉ là bong bóng, nhiều người cho rằng công nghệ phía sau nó là một sự đột phá, và công nghệ ấy sẽ tiếp tục con đường của mình cho đến khi được chấp nhận và tích hợp với Internet.

[Blockchain](https://bitcoinvietnamnews.com/blockchain-la-gi) là gì ? Blockchain là một công nghệ cho phép truyền tải dữ liệu một cách an toàn dựa vào hệ thống mã hoá vô cùng phức tạp,  tương tự cuốn sổ cái kế toán của một công ty, nơi mà tiền mặt được giám sát chặt chẽ. Trong trường hợp này Blockchain là một cuốn sổ cái kế toán hoạt động trong lĩnh vực kỹ thuật số.

Blockchain sở hữu tính năng vô cùng đặc biệt đó là việc truyền tải dữ liệu không đòi hỏi một trung gian để xác nhận thông tin. Hệ thống Blockchain tồn tại rất nhiều nút độc lập có khả năng xác thực thông tin mà không đòi hỏi “dấu hiệu của niềm tin”. Thông tin trong Blockchain không thể bị thay đổi và chỉ được bổ sung thêm khi có sự đồng thuận của tất cả các nút trong hệ thống. Đây là một hệ thống bảo mật an toàn cao trước khả năng bị đánh cắp dữ liệu. Ngay cả khi một phần của hệ thống Blockchain sụp đổ, những máy tính và các nút khác sẽ tiếp tục bảo vệ thông tin và giữ cho mạng lưới tiếp tục hoạt động.

Công nghệ Blockchain có thể nói là sự kết hợp giữa 3 loại công nghệ bên dưới:

Mật mã học: Sử dụng public key và hàm hash function để đảm bảo tính minh bạch, toàn vẹn và riêng tư.

Mạng ngang hàng: Mỗi một nút trong mạng được xem như một client và cũng là server để lưu trữ bản sao ứng dụng.

Lý thuyết trò chơi: Tất cả các nút tham gia vào hệ thống đều phải tuân thủ luật chơi đồng thuận (PoW, PoS…) và được thúc đẩy bởi động lực kinh tế.

Hệ thống sử dụng .NET CORE 2.0(C# và ASP.NET) để vận hành, đảm bảo độ chính xác, tốc độ thực thi của dữ liệu và độ an toàn, đảm bảo được yêu cầu bảo mật của Blockchain.

**Từ khóa**: Blockchain, bảo mật, .NET CORE 2.0, PoW, PoS

* 1. TỔNG QUAN
     1. ĐẶT VẤN ĐỀ

Trong giai đoạn hiện nay và trong tương lai việc lưu trữ và bảo mật dữ liệu đang là đề tài cấp bách nhất, thông tin dữ liệu có thể bị thay đổi và đánh cắp bất cứ lúc nào, cơ chế bảo mật hiện nay còn yếu kém, phải thông một trung gian để lưu trữ thông tin. Vì vậy sẽ mất rất nhiều thời gian và hiệu quả của công việc.

Do đó, với đề tài “**NGHIÊN CỨU XÂY DỰNG HỆ THỐNG BLOCKCHAIN TRÊN NỀN ASP.NET CORE**”, tôi mong muốn xây dựng một ứng dụng website áp dụng cơ chế bảo mật của Blockchain để bảo mật dữ liệu và thực hiện các giao dịch một cách tối ưu nhất.

* + 1. MỤC TIÊU ĐỀ TÀI

Với mong muốn giúp các bạn hiểu được tầm quan trọng của việc bảo mật dữ liệu và thông tin của các giao dịch, không để một ai có thể tự ý chỉnh sửa hay thay đổi dữ liệu và các giao dịch một cách bất hợp pháp, thì ứng dụng web có mục đích chính là khắc phục được cơ chế có thể thay đổi được thông tin, để giúp toàn vẹn dữ liệu và chống lại sự thay đổi dữ liệu.

* + 1. PHẠM VI ĐỀ TÀI
       1. Phạm vi chức năng

Người dùng: có thể thêm giao dịch mới.

* + - 1. Phạm vi người dùng

Không hạn chế số lần giao dịch

Phạm vi kỹ thuật số lượng lỗi phải ở mức độ thấp nhất, đảm bảo độ chính xác của dữ liệu và mức độ an toàn, đảm bảo được yêu cầu theo cơ chế làm việc của Blockchain

Sử dụng Visual Studio hỗ trợ lập trình ASP.NET CORE.

Thực hiện cơ chế làm việc của Blockchain bằng cơ chế Proof Of Work (POW): Bằng chứng công việc.

Hệ quản trị cơ sở dữ liệu SQL Server để lưu và quản lý dữ liệu.

Ứng dụng Web có thể chạy trên các trình duyệt thông dụng: Google Chrome, Firefox, Opera,…

* + 1. NỘI DUNG CHÍNH CỦA NIÊN LUẬN
       1. Tổng quan

Giới thiệu tổng quan, mục tiêu và phạm vi đề tài.

* + - 1. Cơ sở lý thuyết

Tìm hiểu những cơ sở lý thuyết phục vụ cho việc giải quyết vấn đề đã đặt ra và áp dụng lý thuyết vào thực nghiệm.

* + - 1. Nội dung và kết quả nghiên cứu

Tiến hành phân tích, code thực nghiệm và hoàn thiện hệ thống. Từ đó đưa ra kết luận và các phương hướng phát triển hệ thống trong tương lai.

* 1. CƠ SỞ LÝ THUYẾT
     1. BLOCKCHAIN
        1. Cơ chế đồng thuận trong blockchain:

**Proof Of Work (POW: Bằng chứng công việc.):**

+ Các node tham gia hệ thống blockchain được gọi là các **miner** (thợ đào) cùng thực hiện tính toán để tìm ra hash cho khối block thoả mãn một điều kiện nhất định (điều kiện càng khó thì khối lượng tính toán càng nhiều). Ai tính toán ra kết quả đầu tiên sẽ là được phần thưởng, là một số lượng coin.

+ Để được nhận phần thưởng, các node phải cố gắng trở thành người hoàn thành tính toán đầu tiên, cũng có nghĩa là cần đẩy nhanh tốc độ và khối lượng tính toán. Dẫn đến khối lượng tính toán của toàn hệ thống tăng, tương ứng với nó là mức độ ngốn năng lượng.

+ Độ khó tính toán tăng, mất nhiều thời gian để tính toán hơn, block chain ngày càng trở nên tốn kém về cả thời gian và tiền bạc.

**Proof of Stake (PoS):**

**+ PoS** đưa ra một phương pháp đồng thuận mới trong việc đóng gói block vào chain.  
Trong các node tham gia, một node sẽ được **lựa chọn ngẫu nhiên** để trở thành người kiểm định và đóng gói block.

+ Node tham gia được gọi là các **validator** (người kiểm định).  
Điều kiện để tham gia là các node này cần phải bỏ một khoản tiền vào mạng lưới, mà anh Vitalik Buterin gọi là **bet** (đặt cược). Ta có thể hiểu nôm na là đặt cọc tiền để chứng minh danh tính bản thân.

+ Khoản tiền này được gọi bằng từ chuyên môn là **Stake**.

+ Khoản tiền này sẽ bị hệ thống thực hiện lock. Và sẽ được unlock sau khi node rút khỏi việc tham gia validator **một thời gian** (Chú ý việc unlock sẽ không được thực hiện ngay lập tức).

+ Validator được chọn sẽ thực hiện validate block. Thuật ngữ chuyên môn cho hành động này là **forge** hoặc **mint** (phân biệt với mine trong PoW).

+ Nếu block hợp lệ và ghi được vào chain, validator này sẽ được phần thưởng chính là từ transaction fee.

* + - 1. Đặc tính 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 còn Internet trên toàn cầu.

Bất biến: dữ liệu trong Blockchain không thể sửa (có thể sửa nhưng sẽ để 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 Blockchain đi từ địa chỉ này tới 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.

* + 1. BOOTSTRAP:

Bootstrap là một framework CSS được Twitter phát triển. Nó là một tập hợp các bộ chọn, thuộc tính và giá trị có sẵn để giúp web designer tránh việc lặp đi lặp lại trong quá trình tạo ra các class CSS và những đoạn mã HTML giống nhau trong dự án web của mình.

Bootstrap là bao gồm các HTML templates, CSS templates và Javascript tạo ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Trong bootstrap có thêm các plugin Javascript trong nó. Giúp cho việc thiết kế reponsive của bạn dễ dàng hơn và nhanh chóng hơn.

Những điểm thuận lợi khi bạn sử dụng bootstrap:

Rất dễ để sử dụng: Nó đơn giản vì nó được base trên HTML, CSS và Javascript chỉ cẩn có kiến thức cơ bản về 3 cái đó là có thể sử dụng bootstrap tốt.

Tính năng Responsive: Bootstraps xây dựng sẵn reponsive css trên các thiết bị phones, tablets, và desktops.

Mobile: Trong Bootstrap 3 mobile-first styles là một phần của core framework.

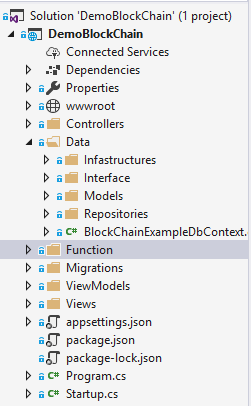
Tương thích với trình duyệt: Nó tương thích với tất cả các trình duyệt (Chrome, Firefox, Internet Explorer, Safari, and Opera).

* + 1. ASP.NET CORE 2.0

Khái niệm:ASP.NET Core là một open-source mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối đám mây, giống như web apps, IoT và backend cho mobile. Ứng dụng ASP.NET Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. Nó được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise. Nó bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy bạn giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bạn.

Bạn có thể phát triển và chạy những ứng dụng ASP.NET Core đa nền tảng trên Windows, Mac và Linux. Đồng thời nó đã trở thành một mã nguồn mở. Đây là một thay đổi rất lớn và theo mình là quan trọng nhất của ASP.NET Core. Điều mà trước đây khó có một lập trình viên nào có thể nghĩ đến. Có lẽ đó cũng là một xu thế mà các ngôn ngữ lập trình hiện nay đang hướng tới.

* + - 1. Cấu trúc Project

Dependencies: chứa các thư viện của Microsoft, các gói Nuget khi cài đặt vào chương trình

Properties: Chứa các thuộc tính của project.

Wwwroot: Chứa các thư viện css, javascripts, jquery..., các file hình ảnh

Controller: nơi chứa các lớp điều khiển cho dự án, chính là Controller trong 3 thành phần MVC (model-view-controller).

Data: tầng triển khai các class Model, Repository, Generic Repository pattern và Dependency Injection.

Function: Thư mục chứa các hàm chức năng tự định nghĩa.

Migrations: Lưu trữ các phiên khi thay đổi lớp Model.

ViewModels: thư mục chứa các class chứa dữ liệu cho view hiển thị.

Views: chứa các .cshtml để hiển thị giao diện, cũng là 1 trong 3 thành phần MVC.

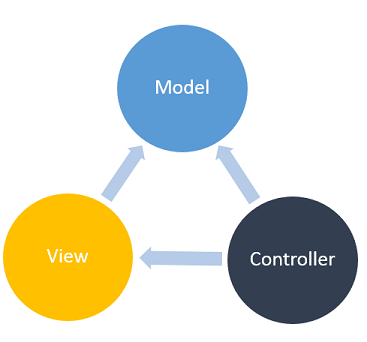
Package.json và Package-lock.json: file quản lý các package đã cài.

Startup.cs: Nơi định nghĩa các yêu cầu của services khi ứng dụng được cấu hình

Cấu trúc project

* + 1. MVC

Mô hình MVC là một chuẩn mô hình và đóng vai trò quan trọng trong quá trình xây dựng – phát triển – vận hành và bảo trì một hệ thống hay một ứng dụng – phần mềm. Nó tạo ra một mô hình 3 lớp Model – View – Controller tách biệt và tương tác nhau, giúp các chuyên gia có thể dễ dàng dựa vào mô hình để trao đổi và xử lý những nghiệp vụ một cách nhanh chóng. Đây là một mô hình đã xuất hiện từ những năm 70 của thế kỷ 20 tại phòng thí nghiệm Xerox PARC ở Palo Alto, nó không phụ thuộc vào môi trường, nền tảng xây dựng hay ngôn ngữ phát triển. Chúng ta có thể áp dụng mô hình MVC vào các dự án trong môi trường Windows, Linux… và sử dụng bất kỳ ngôn ngữ nào như PHP, ASP, JSP…



Mô hình MVC

Mô hình MVC được chia làm 3 lớp xử lý gồm Model – View – Controller :

Model : là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (mysql, mssql… ); nó sẽ bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu…

View : là nới chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.

Controller : là nới tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

* + 1. CODE FIRST MIGRATION

Code First Migration đề cập đến vấn đề bạn gặp phải khi cập nhật CSDL như thêm cột, xóa cột, tạo bảng mới và cập nhật vào CSDL đang có khi bạn sử dụng Code First trong Entity Framework. Việc cập nhật sẽ được thực hiện tự động mỗi khi có thay đổi trong các model mà bạn đã khai báo.

Ưu điểm:

+ Rất phổ biến (vì các lập trình viên thường không thích thiết kế DB, nhưng thích thiết kế class).

+ Kiểm soát hoàn toàn code model, thêm xóa sửa thuộc tính vô cùng dễ dàng.

+ Có thể version control Database.

Nhược điểm:

+ Các thay đổi cấu trúc trực tiếp trên DB sẽ mất.

+ Khó kiểm soát những column sẽ tạo trên Db.

+ Hơi khó khi kết hợp với Db có sẵn.

* + 1. REPOSITORY PATTERN

**Repository Pattern** đơn giản là một lớp trung gian giữa tầng business và tầng truy xuất dữ liệu. Nếu bạn sử dụng Entity Framework trong ASP.NET Core MVC hay trong ASP.NET MVC thì business là tầng xử lý nghiệp vụ của dự án, còn Data chính là tầng dùng để chứa các lớp DbContext và các class entity. Ở trong ASP.NET thì **repository** sẽ được sử dụng để nhằm điều chỉnh dữ liệu trước khi dữ liệu được truyền lên business logic hoặc truyền xuống tầng data. Mục đích tạo ra lớp này để cách ly với việc tiếp cận data sao cho những thay đổi không ảnh hưởng trực tiếp đến lớp logic business.

Lợi ích:

+ Dễ dàng cho unit test.

+ Kiến trúc linh hoạt.

+ Hạn chế code dư thừa.

+ Chia tách giúp quản lý và làm dự án hiệu quả hơn.

+ Nếu có thay đổi tầng business hoặc data thì không cần thay đổi Repository.

+ Các logic về business logic và xử lý dữ liệu được tập trung lại.

* + 1. GENERIC REPOSITORY PATTERN

Nếu như bạn có một dự án và bạn nhận thấy mỗi repository đều thực hiện những hành động tương tự hoặc gần tương tự nhau, nếu bạn tạo liên tiếp nhiều repository gần giống nhau đó sẽ là một sự lãng phí lớn. Thay vào đó. bạn chỉ cần tạo một và chỉ một repository cho việc thao tác với toàn bộ các class entity là đủ. Và đó chính là Generic Repository Pattern.

Lợi ích:

+ Giảm thiểu được sự trùng lập các đoạn code của bạn.

+ Đảm bảo các coder dùng chung một mẫu.

+ Dễ dàng cho việc bảo trì và kiểm thử.

+ Giảm thiểu các lỗi có thể xảy ra.

* + 1. DEPENDENCY INJECTION

Dependency Injection (DI) là 1 kỹ thuật để chống lại sự phụ thuộc giữa các đối tượng, các thành phần, và các lớp bằng cách nghịch đảo luồng điểu khiển của ứng dụng.

Có 3 dạng Dependency Injection:

+ Constructor Injection: Các dependency sẽ được container truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất.

+ Setter Injection: Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter.

+ Interface Injection: Class cần inject sẽ implement 1 interface. Interface này chứa 1 hàm tên Inject. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm Inject của interface đó. Đây là cách rườm rà và ít được sử dụng nhất.

Ưu điểm:

+ Giảm sự kết dính giữa các module.

+ Code dễ bảo trì, dễ thay thế module..

+ Rất dễ test và viết [Unit Test](https://toidicodedao.com/2015/08/25/tutorial-viet-unit-test-trong-c-voi-nunit/).

+ Dễ dàng thấy quan hệ giữa các module (Vì các dependecy đều được inject vào constructor).

Khuyết điểm:

+ Khái niệm DI khá “khó tiêu”, các developer mới sẽ gặp khó khăn khi học.

+ Sử dụng interface nên đôi khi sẽ khó debug, do không biết chính xác module nào được gọi.

+ Các object được khởi tạo toàn bộ ngay từ đầu, có thể làm giảm performance.

+ Làm tăng độ phức tạp của code.

* 1. NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU
     1. Các thành phần của Blockchain:

+ Block: bao gồm nhiều transaction và các giá trị thuộc tính như:

+ Index(chỉ số của 1 block).

+ TimeStamp(thời gian tạo ra 1 block).

+ Hash(chuỗi được băm bởi thuật toán mã hóa SHA256).

+ PreviousHash(chỉ số hash của block trước đó).

+ Số Nonce(một số được sử dụng để tìm một hash hợp lệ).

+ Transaction: là một đơn vị của blockchain, chứa các giá trị, dữ liệu thuộc tính cần lưu trữ trong Block.

+ Difficulty: số ký tự đầu của chuỗi Hash cần tìm ra.

+ Cơ chế POW(Bằng chứng công việc) và cơ chế POS.

+ Thuật toán SHA256: thuật giải băm mật.

* + 1. Cài đặt
       1. Class block

public class Block

{

public int Index { get; set; }

public DateTime TimeStamp { get; set; }

public string PreviousHash { get; set; }

public string Hash { get; set; }

public IList<Transaction> Transactions { get; set; }

public int Nonce { get; set; } = 0;

public Block(DateTime timeStamp, string previousHash, IList<Transaction> transactions)

{

Index = 0;

TimeStamp = timeStamp;

PreviousHash = previousHash;

Transactions = transactions;

}

public string CalculateHash()

{

SHA256 sha256 = SHA256.Create();

byte[] inputBytes = Encoding.ASCII.GetBytes($"{TimeStamp}-{PreviousHash ?? ""}-{JsonConvert.SerializeObject(Transactions)}-{Nonce}");

byte[] outputBytes = sha256.ComputeHash(inputBytes);

return Convert.ToBase64String(outputBytes);

}

public void Mine(int difficulty)

{

var leadingZeros = new string('0', difficulty);

while (this.Hash == null || this.Hash.Substring(0, difficulty) != leadingZeros)

{

this.Nonce++;

this.Hash = this.CalculateHash();

}

Console.WriteLine("\nBlock has been mined: " + this.Hash + "\n");

}

}

* + - 1. Class Blockchain:

public class Blockchain

{

IList<Transaction> PendingTransactions = new List<Transaction>();

public IList<Block> Chain { set; get; }

public int Difficulty { set; get; } = 2;

public int Reward = 1; //1 cryptocurrency

public Blockchain()

{

InitializeChain();

AddGenesisBlock();

}

public void InitializeChain()

{

Chain = new List<Block>();

}

public Block CreateGenesisBlock()

{

Block block = new Block(DateTime.Now, null, PendingTransactions);

block.Mine(Difficulty);

PendingTransactions = new List<Transaction>();

return block;

}

public void AddGenesisBlock()

{

Chain.Add(CreateGenesisBlock());

}

public Block GetLatestBlock()

{

return Chain[Chain.Count - 1];

}

public void CreateTransaction(Transaction transaction)

{

PendingTransactions.Add(transaction);

}

public void ProcessPendingTransactions(string minerAddress)

{

Block block = new Block(DateTime.Now, GetLatestBlock().Hash, PendingTransactions);

AddBlock(block);

PendingTransactions = new List<Transaction>();

CreateTransaction(new Transaction(null, minerAddress, Reward));

}

public void AddBlock(Block block)

{

Block latestBlock = GetLatestBlock();

block.Index = latestBlock.Index + 1;

block.PreviousHash = latestBlock.Hash;

block.Mine(this.Difficulty);

Chain.Add(block);

}

public bool IsValid()

{

for (int i = 1; i < Chain.Count; i++)

{

Block currentBlock = Chain[i];

Block previousBlock = Chain[i - 1];

if (currentBlock.Hash != currentBlock.CalculateHash())

{

return false;

}

if (currentBlock.PreviousHash != previousBlock.Hash)

{

return false;

}

}

return true;

}

public int GetBalance(string address)

{

int balance = 0;

for (int i = 0; i < Chain.Count; i++)

{

for (int j = 0; j < Chain[i].Transactions.Count; j++)

{

var transaction = Chain[i].Transactions[j];

if (transaction.FromAddress == address)

{

balance -= transaction.Amount;

}

if (transaction.ToAddress == address)

{

balance += transaction.Amount;

}

}

}

return balance;

}

}

* + - 1. Class Transaction:

public class Transaction

{

public string FromAddress { get; set; }

public string ToAddress { get; set; }

public int Amount { get; set; }

public Transaction(string fromAddress, string toAddress, int amount)

{

FromAddress = fromAddress;

ToAddress = toAddress;

Amount = amount;

}

}

* + - 1. Class Program:

class Program

{

static void Main(string[] args)

{

var startTime = DateTime.Now;

Blockchain phillyCoin = new Blockchain();

Console.WriteLine(JsonConvert.SerializeObject(phillyCoin, Formatting.Indented));

phillyCoin.CreateTransaction(new Transaction("MaHesh", "Henry", 5));

phillyCoin.CreateTransaction(new Transaction("MaHesh321", "Henry123", 10));

phillyCoin.CreateTransaction(new Transaction("Henry", "MaHesh", 10));

phillyCoin.ProcessPendingTransactions("Bill");

var endTime = DateTime.Now;

Console.WriteLine($"Duration: {endTime - startTime}");

Console.WriteLine("=========================");

Console.WriteLine($"Henry' balance: {phillyCoin.GetBalance("Henry")}");

Console.WriteLine($"MaHesh' balance: {phillyCoin.GetBalance("MaHesh")}");

Console.WriteLine($"Bill' balance: {phillyCoin.GetBalance("Bill")}");

Console.WriteLine("=========================");

Console.WriteLine($"phillyCoin");

Console.WriteLine(JsonConvert.SerializeObject(phillyCoin, Formatting.Indented));

Console.ReadKey();

}

}

* + 1. Kết quả

Block has been mined: 009O2vNUNioK2rJgjP2ZdTP8mtzsndo7+RAFJny7Hhg=

{

"Reward": 1,

"Chain": [

{

"Index": 0,

"TimeStamp": "2018-11-14T21:16:49.262704+07:00",

"PreviousHash": null,

"Hash": "009O2vNUNioK2rJgjP2ZdTP8mtzsndo7+RAFJny7Hhg=",

"Transactions": [],

"Nonce": 15135

}

],

"Difficulty": 2

}

Block has been mined: 00poPYZSMoWc2Zox/+JJa0JhBGOEMEr/lTvvDu+lEXE=

Duration: 00:00:07.9051936

=========================

Henry' balance: -5

MaHesh' balance: 5

Bill' balance: 0

=========================

phillyCoin

{

"Reward": 1,

"Chain": [

{

"Index": 0,

"TimeStamp": "2018-11-14T21:16:49.262704+07:00",

"PreviousHash": null,

"Hash": "009O2vNUNioK2rJgjP2ZdTP8mtzsndo7+RAFJny7Hhg=",

"Transactions": [],

"Nonce": 15135

},

{

"Index": 1,

"TimeStamp": "2018-11-14T21:16:56.1005757+07:00",

"PreviousHash": "009O2vNUNioK2rJgjP2ZdTP8mtzsndo7+RAFJny7Hhg=",

"Hash": "00poPYZSMoWc2Zox/+JJa0JhBGOEMEr/lTvvDu+lEXE=",

"Transactions": [

{

"FromAddress": "MaHesh",

"ToAddress": "Henry",

"Amount": 5

},

{

"FromAddress": "MaHesh321",

"ToAddress": "Henry123",

"Amount": 10

},

{

"FromAddress": "Henry",

"ToAddress": "MaHesh",

"Amount": 10

}

],

"Nonce": 10663

}

],

"Difficulty": 2

}

* 1. KẾT LUẬN VÀ ĐỀ NGHỊ
     1. KẾT QUẢ ĐẠT ĐƯỢC

Sau thời gian nghiên cứu và triển khai, ứng dụng website đã đạt được những vấn đề đặt ra:

Bảo mật theo cơ chế POW.

Không thể thay đổi dữ liệu một cách bất hợp pháp.

* + 1. HƯỚNG PHÁT TRIỂN

Hoàn thiện hơn các chức năng đã đạt được

Xây dựng giao diện đồ họa trong thời gian sắp tới

Phát triển ứng dụng ở quy mô lớn hơn

Áp dụng cơ chế bảo mật của Blockchain vào các lĩnh vực khác

* 1. TÀI LIỆU THAM KHẢO

[1] <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1>. Truy cập 11/2018.

[2] <https://bitcoinvietnamnews.com/blockchain-la-gi>. Truy cập 11/2018.

[3] <https://quantrimang.com/blockchain-la-gi-bong-bong-hay-cuoc-cach-mang-thuc-su-sau-internet-143099>. Truy cập 11/2018.

[4] <https://vi.wikipedia.org/wiki/Blockchain>. Truy cập 11/2018.

[5] https://www.c-sharpcorner.com/article/blockchain-basics-building-a-blockchain-in-net-core/. Truy cập 13/11/2018.

[6] https://1upnote.me/post/2018/06/pow-vs-pos/. Truy cập 10/2018.

[7] <https://github.com/ProgrammingBlockchain> . Truy cập 09/2018.

[8] https://www.youtube.com/channel/UC6dbBV2jyaOV-3b3Tboul9w/videos. Truy cập 11/2018.

[9] <https://getbootstrap.com/>. Truy cập 09/2018.

[10] https://www.youtube.com/watch?v=735K9rl42pI. Truy cập 10/2018.

[11] <http://nguyenit.net/asp-net-core/generic-repository-pattern-la-gi-trong-asp-net-core/>. Truy cập 10//2018.

[12] <http://nguyenit.net/asp-net-core/repository-pattern-la-gi-su-dung-trong-asp-net-core/>. Truy cập 10/2018.

[13] <https://code-maze.com/async-generic-repository-pattern/>. Truy cập 09/2018.

[14] https://toidicodedao.com/dependency-injection-va-inversion-of-control-phan-1-dinh-nghia/. Truy cập 11/2018.

[15] https://www.tutorialspoint.com/asp.net\_core/. Truy cập 10/2018.