令和2年度「専修学校による地域産業中核的人材養成事業」
スマートコントラクトを使用したシステム開発人材の育成

スマートコントラクト開発実践
指導マニュアル

学校法人 麻生塾 麻生情報ビジネス専門学校

令和2年度「専修学校による地域産業中核的人材養成事業」
スマートコントラクトを使用したシステム開発人材の育成

スマートコントラクト開発実践

スマートコントラクト開発実践

目次

環境構築

演習で必要な環境を整える

環境構築の章を参考に行ってください

1章
リレーショナルデータベースを用いたwebシステムの構築

1.1.1

Webアプリケーション

ネイティブアプリケーション
インターネットなどのネットワークから利用するアプリケーション

ネイティブアプリケーション
→端末にインストールして使うアプリケーション

- Webアプリケーションとは、インターネットなどのネットワークから利用するアプリケーション
- Webアプリケーションとよく対比されるものとして、ネイティブアプリケーションがある。（手元のPCやスマートフォンなどの端末にインストールして利用するアプリケーション）

この教材ではブロックチェーンを用いたWebアプリケーションを作成する。

1.1.2

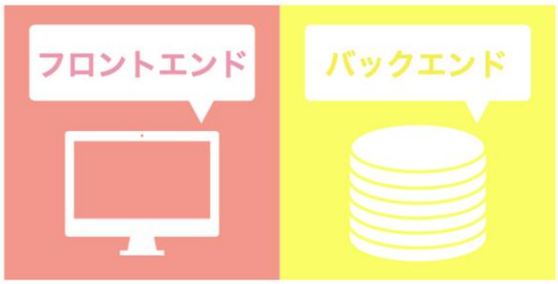
Webアプリケーション

Webアプリケーションの構成を細かく確認する

Webアプリケーションの構成を分割して捉える

1.1.2

Webアプリケーション



ユーザーから見える部分
（HTML, CSS, Javascript）

ユーザーから見えない部分
（Java, PHP, Ruby,）

7

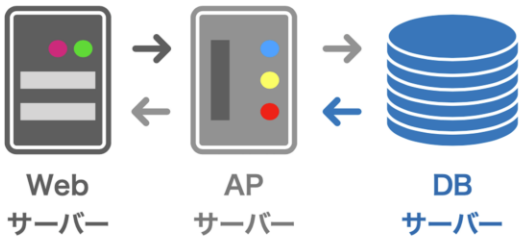
「フロントエンド」と「バックエンド」の2つに分割してWebアプリケーションを捉える

- フロントエンド：ユーザーがWebアプリケーションにアクセスした際に、ブラウザからユーザーに見える部分、直接操作できる部分
- バックエンド: ユーザーが見えない部分・直接操作できない部分(データを処理して結果を返す、入力された内容をデータベースに保存する)

1.1.2

Webアプリケーション

Webアプリケーションの3層構造



8

バックエンドをさらに3つに分割して捉えると「Webサーバー」「アプリケーションサーバー」「データベースサーバー」に分割することができる(これらの3つのサーバーは1つの物理的なサーバーの中で機能するあることもあれば、それぞれ別の物理サーバーで機能することもある)

- Webサーバー: ユーザーがブラウザからリクエストした処理が最初に届くサーバーであり、また最終的にユーザーにHTMLファイルなどを返す役割
- アプリケーションサーバー:Webサーバーからのリクエストを Java、Ruby、PHPなどのプログラミング言語で作成したプログラムにより実行して処理する。(HTMLファイルを動的に作成する、データベースにアクセスする)
- データベースサーバー : データの管理を行う。Webサイトに必要なユーザーデータや商品データを保存する。

1.1.2

Webアプリケーション

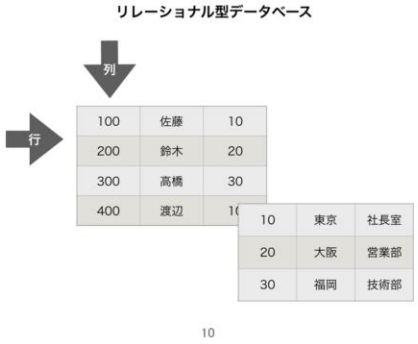
データベースサーバーに着目する

この教材では、DBとブロックチェーンの関連を扱うため、データベースに着目して話を進める

Webアプリケーション

リレーショナルデータベース(RDB)

行と列からなる表形式で表されたデータベース



10

リレーショナルデータベースとは、行と列によって構成された「表形式のテーブル」と呼ばれるデータの集合を、互いに関連付けて関係モデルを使ったデータベース

リレーショナルデータベースを制御するソフトは「リレーショナルデータベース管理システム」と呼ばれる

1.1.3

Webアプリケーション

MySQL

Webアプリケーションの大部分で使用されているオープンソースの
リレーショナルデータベース管理システム

→今回作成するシステムで利用する

Webアプリケーションの大部分で使用されているオープンソースのリレーショナルデータベース管理システム

Webアプリケーション

MySQLの環境構築

Webアプリケーションを作成する際の仮想マシン上のMySQL環境の構築も兼ねて、MySQLを使った演習

※仮想マシンの立ち上げが必要になります。

Webアプリケーション

SQL演習

作成したMySQLの環境を利用して、SQLの演習を行う。

この演習の目的は、以下を体感してもらうこと

- DBの直接操作は難しくないこと
- 権限があればDBを好きに触れること

1.1.3

Webアプリケーション

データベースの権限

- データベース上のデータを直接変更することは難しくない
- 権限さえあれば、データベースの操作を好きにできる

データベースサーバーの一部をブロックチェーンで置き換える

→システム全体の改ざん耐性が高まる

今回作成し、削除したデータベースはあくまで練習用のデータベース
→しかし、Webアプリケーションが運用されているデータベースでも同じように権限さえ持っていれば、今回のように好きにデータベースを操作できてしまう。

その権限が奪われてしまうことや管理者の悪意により、不正な操作を行うことができる
→この問題を管理者の存在しないシステムであるブロックチェーンを使うことで解決することができる。

1.2.1

トレーサビリティシステム

対象とする物品の流通履歴を確認できる状態

その製品が
いつ、どこで、だれによって作られたのか
を追跡可能にすることができる

15

トレーサビリティとは、トレースとアビリティを組み合わせた造語

その製品が「いつ、どこで、だれによって作られたのか」を追跡可能にすることができ、生産者、流通に関連する業者、消費者の全てがそれらの情報を確認することができる

製品に対する安全意識の高まりから業界問わず幅広い分野に浸透している。

1.2.2

トレーサビリティシステム

有機野菜のトレーサビリティシステム

宮崎県の綾町で生産されたブランド野菜の流通経路を記録する

16

- 宮崎県の綾町は、豊かな自然に恵まれた地域であり、美味しい野菜が育つ地域として注目されている
- ブランド野菜として高級スーパーで扱われることも多い綾町のレタスを使って、「この野菜は宮崎県綾町で生産されたものである」と証明したい

1.2.2

トレーサビリティシステム

有機野菜のトレーサビリティシステム

1.収穫したレタスをダンボールへ詰める

2.梱包の際に、LTEでインターネットに接続された「IoTセンサー」をダンボールに同封する

3.箱に伝わる振動や温度、照度などのデータをセンサーに蓄積し、ブロックチェーンに記録する

4.ダンボールに各生産物の情報がアップロードされたNFCタグを取り付ける

- 食卓に並ぶまでの流通経路の随所でセンサーを通してデータがブロックチェーンに書き込まれる
- ブロックチェーンには、野菜が生育した土壌や地域に関する情報、出荷時の荷姿なども記録することができる
- 記録された情報は専用のWebページから確認することができる

1.3.1

トレーサビリティシステムの構成/環境構築

今回作成するトレーサビリティシステム

→バッグのトレーサビリティシステム

1.3.1

トレーサビリティシステムの構成/環境構築

トレーサビリティシステムの構成

・ブロックチェーンを利用しない通常のトレーサビリティシステムを作成し、その後ブロックチェーンをデータベースサーバーの機能の一部として利用する形に改修する

・今回のシステムは全てローカル環境で完成する構成

1.3.2

トレーサビリティシステムの構成/環境構築

トレーサビリティシステムの構成

Apache　　PHP

Linux　　MySQL

- Google Chrome: 利用するツールの兼ね合いもあり、今回利用するブラウザはGoogle Chromeに指定します。
- Linux: Virtual Boxで仮想マシンを立ち上げます
- Apache: Webサーバーには Apacheを利用します
- PHP: サーバーサイドはPHPで実装します
- MySQL: データベースにはMySQLを利用します

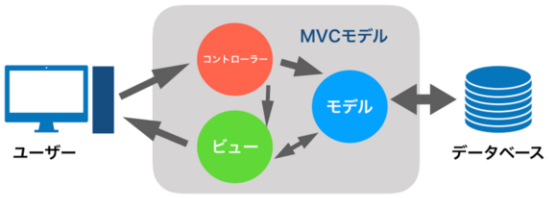
1.3.3

トレーサビリティシステムの構成/環境構築

Laravel

PHPのWebフレームワーク

LaravelではMVCモデルを採用



21

- サーバーサイドのフレームワークにPHPのWebフレームワークであるLaravelを使用する
- LaravelではMVCモデルを採用しています

1.3.4

トレーサビリティシステムの構成/環境構築

以下の環境の構築を行う

- Laravel
- Apache
- SSH

Laravel, Apache, SSHの環境構築(SSHは開発を便利に行うため)

トレーサビリティシステムの構成/環境構築

データベース作成

MySQLにログインし、
開発に必要なデータベースを作成する

MySQLで「traceability」と言う名のデータベースを作成する。(MySQL環境の構築はすでに終わっていることが前提)

1.3.4

トレーサビリティシステムの構成/環境構築

以下の環境の構築を行う

- Laravel
- Apache
- SSH

Laravel, Apache, SSHの環境構築(SSHは開発を便利に行うため)

トレーサビリティシステム作成

新規のLaravelプロジェクトを立ち上げる

- プロジェクトの立ち上げ
- 権限の設定
- 設定ファイルの書き換え
- 認証機能の作成

1.4.1

トレーサビリティシステムの構成/環境構築

以下の環境の構築を行う

- Laravel
- Apache
- SSH

Laravel, Apache, SSHの環境構築(SSHは開発を便利に行うため)

1.4.1

トレーサビリティシステム作成

ユーザー機能一覧

/	ホーム画面。「製品登録」と「流通経路確認」に遷移できる
/login	メールアドレスとパスワード入力してログインをする
/register	新規のユーザー登録をする画面。「氏名」「メールアドレス」「パスワードが必要」
/content/add_content	新規の商品登録画面。「ブランド名」「製品名」「出荷価格」「製品情報」「商品の画像」が必要
/content/add_content_success	商品の新規登録が成功した際の画面。「商品コード」と「QRコード」が表示される
/trace/content_detail	流通経路確認で商品コードを検索した際に表示される画面。「製品詳細」「流通経路」「コメントを追加」の項目あり
/user/info	ユーザー情報を確認する画面。「ユーザー情報」と「登録した製品一覧」が表示される
/user/info_edit	ユーザー情報を編集する画面。「名前」と「メールアドレス」を変更できる
/user/info_edit_success	ユーザー情報変更が成功した際に表示される画面

1.4.1

トレーサビリティシステムの構成/環境構築

以下の環境の構築を行う

- Laravel
- Apache
- SSH

Laravel, Apache, SSHの環境構築(SSHは開発を便利に行うため)

1.4.2

トレーサビリティシステム作成

マイグレーションファイルの作成

users	
id	bigInteger
name	string
email	string
email_verified_at	timestamp
password	string
role	string
rememberToken	string
created_at	timestamp
updated_at	timestamp

contents	
id	bigInteger
brand	string
name	string
price	string
information	string
user_id	bigInteger
identifier	string
image_path	string
created_at	timestamp
updated_at	timestamp
user_id	bigInteger

traces	
id	bigInteger
content_id	bigInteger
user_id	bigInteger
comment	text
latitude	string
longitude	string
created_at	timestamp
updated_at	timestamp
user_id	bigInteger
content_id	bigInteger
price	double

必要なテーブル、カラムをマイグレーションファイルから作成する

トレーサビリティシステム作成

演習
マイグレーションファイルの作成

演習形式で進める。解答は巻末に記述してある

トレーサビリティシステム作成

演習

Model/Controller/Routingの作成

- 教材に従ってPHPのコーディングを進めていく
- Laravelのドキュメントも参考に

1.4.4

トレーサビリティシステム作成

演習

管理者機能の作成

教材に従って管理者機能のコーディングを進めていく

1.4.4

トレーサビリティシステム作成

作成したシステムの確認

WebサーバーとVirtualBoxの設定を行い、
ホストマシンからシステムの動作を確認する

2章

Webアプリケーションとブロックチェーン

2.1.1

ブロックチェーン

ブロックチェーンの特徴覚えていますか?

2.1.1

ブロックチェーン

非中央集権

特別な権限を持った管理者が存在しない
→管理者権限による操作が存在しない

高い改ざん耐性

独自のデータ構造 & 大量のバックアップ
→データを完全に消すのは不可能

今回のシステムにとって重要なブロックチェーンの特徴を紹介する

非中央集権
- 特定の企業や団体が管理しているのではなく、その参加者により自律的にシステムが維持管理
- 保存したデータを管理者の独断により削除、編集されることや、管理者がシステムの運用をやめることでデータが失われる事態を防ぐ

高い改ざん耐性
- ブロックチェーンの独自のデータ構造(ブロックの連鎖構造やマイニングによるもの)
- 世界中の独立した大量のノードが記録を保存していること

2.1.2

ブロックチェーン

ブロックチェーンの種類

パブリックブロックチェーン
誰でも参加できるブロックチェーン

パーミッションドブロックチェーン
参加に制限があるブロックチェーン

パブリックブロックチェーン
- 「パブリック」という言葉の通り開かれたブロックチェーン
- ブロックチェーンのネットワークに参加するための許可や、使用するコンピュータの性能、OSなどの制約はない

パーミッションドブロックチェーン
- パブリックブロックチェーンと異なり、ネットワークへの参加に管理者の許可が必要となるブロックチェーン
- 企業や団体などの組織内や組織間で利用されることの多いブロックチェーン

ブロックチェーン

Bitcoin

「誰にも止められることなく通貨を取引すること」を

目的に作られたブロックチェーン

Ethereum

「管理者を必要としないシステム」を

実現するために作られたブロックチェーン

Bitcoin

- Bitcoinは2009年から運用が始まった「誰にも止められることなく通貨を取引すること」を目的に作られたブロックチェーン
- 「Bitcoin」という言葉は通貨システムのことであり、ブロックチェーンの名前でもある

Ethereum

- Ethereumは2015年にリリースされたブロックチェーンであり、「管理者を必要としないシステム」を実現するために生まれたブロックチェーン
- ブロックチェーン上でのプログラムの作成という面で最もメジャーなブロックチェーン

2.2.1

ブロックチェーン

スマートコントラクト

これまでは

→コンピュータによって自動的に行われる取引

最近は、

→ブロックチェーンが使われたシステム

- スマートコントラクトには厳密な定義はない。これまでは、人間が介在しない全ての商取引がスマートコントラクトと呼ばれることが多かった。
- ブロックチェーンが誕生してからは、ブロックチェーンを利用したシステムのことを指すようになった。

2.2.1

スマートコントラクトの利点と課題点

利点

管理者不在

データをブロックチェーンに保存することで、

権限に関連した問題によりデータが失われる可能性を小さくすることができる

- MySQLの演習でわかった通り、データベースには管理者が存在し、管理者はそのデータベースに対して、新規登録、編集、削除などあらゆる操作をすることができる。
- 管理者の存在しないブロックチェーンをデータベースとすることで、権限に関連した問題によりデータが失われる可能性を小さくすることができる。

2.2.1

スマートコントラクトの利点と課題点

利点

改ざん耐性が高い

Webアプリケーションにブロックチェーンを用いることで、
ブロックチェーンの改ざん耐性の高さを利用することができる

- Webアプリケーションに必要なコードや、取引の実行記録などを改ざんされることなく保存し続けることができる。
- アプリケーションサーバー内のプログラムの改ざんによる不正な取引や、データベースの内容の書き換えなどが発生するリスクを低下させることができる

2.2.2

スマートコントラクトの利点と課題点

課題点

非中央集権には限界がある

Webサーバーやアプリケーションサーバーの

全てをブロックチェーンで置き換えることは難しい

- ブロックチェーン自体には、管理者が存在していませんが、システム全体を見たときに、現在の技術ではWebサーバーやアプリケーションサーバーの全てをブロックチェーンで置き換えることは非常に難しい
- それらの部分に管理者が存在し、システム全体は中央集権的な仕組みになる

2.2.2

スマートコントラクトの利点と課題点

課題点

スケーラビリティ問題

一定時間に処理することの処理数に制限がある

- BitcoinやEthereumをはじめとしたパブリックブロックチェーンでは、一定時間に処理することの処理数に制限がある
- 自身でサーバーのスペックなどを選定することができないパブリックブロックチェーンでは、このような処理数の少なさも理解した上で利用する必要がある

2.2.2

スマートコントラクトの利点と課題点

課題点

プログラムの修正

一度ブロックチェーンに記録された内容は

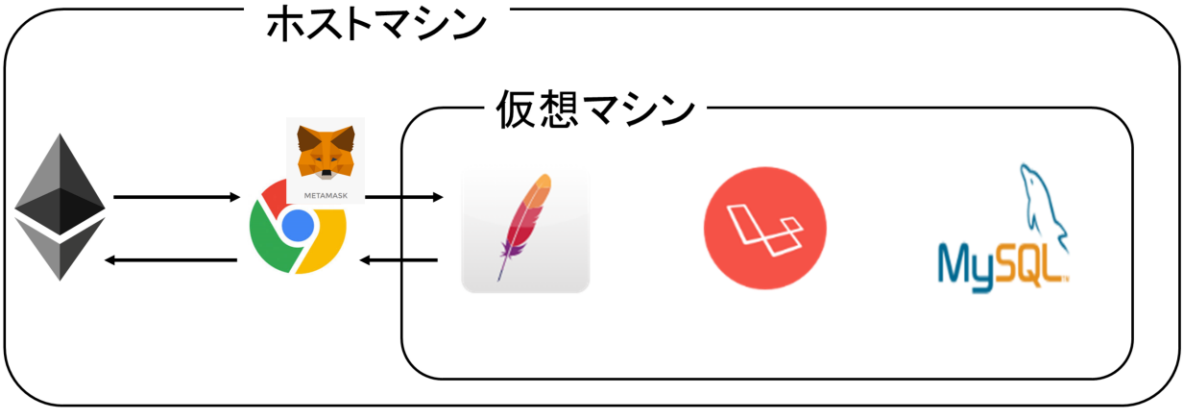
削除することや、変更することができない

- 一度ブロックチェーンに記録された内容は削除することや、変更することが基本的にはできない
- 作成したコードにバグがあり、自身の通貨が他者によって盗まれるなどの問題があった場合にも、このプログラムを改変することや、消してしまうことはできない
- あらかじめプログラムに特定の関数を停止する機能や、プログラム自体を利用できないようにする機能を持たせておく必要がある

2.3.1

スマートコントラクトの構成

スマーコントラクトの構成



今回のブロックチェーンを用いたトレーサビリティシステムの構成

2.3.2

スマートコントラクトの構成

秘密鍵の管理

仮想通貨を管理する秘密鍵の再発行は不可能
→誰がどのように管理するか

秘密鍵の管理の点からも構成を考える必要がある

- 通貨を利用する際や、コントラクトを利用する際には、自身の秘密鍵で署名を行う必要がある
- 秘密鍵は一切の個人情報に結びついておらず、紛失した際に鍵を再発行してくれる機関は存在しない
- ITの知識を持ち合わせていない利用者が適切に秘密鍵の管理ができるか?

ユーザーが鍵の管理を自身で行うか?サービスを提供する企業が管理するか?などの面からシステムの構成を考える必要がある

2.3.3

スマートコントラクトの構成

手数料の支払い

ブロックチェーンの利用には手数料が必要
→誰が支払うか

仮想通貨の保有率の低さを考慮する必要がある

- ブロックチェーンを利用するためには、トランザクション手数料と呼ばれる手数料を支払う必要がある
- トランザクション手数料はそれぞれのブロックチェーンの内部通貨で支払う必要がある
- ブロックチェーンを用いた素晴らしいサービスが発表されたとしても、大半の人は仮想通貨を保有していない

ユーザーが通貨を支払う?サービス提供者が代替して支払う?

3章

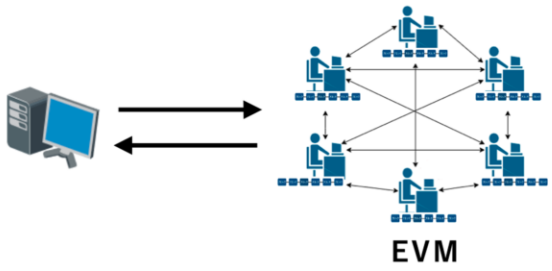
スマートコントラクトを用いたシステム構築

3.1.1

Ethereum

Ethereumを一つの仮想的なコンピュータとして捉える

→Ethereum Virtual Machine (EVM)



49

Ethereumはパブリックブロックチェーンの一つであり、その役割は管理者の存在しないコンピュータ(この特徴からワールドコンピュータとも呼ばれる)

Ethereumはブロックチェーンの一つであり、仮想マシンでもあるというと難しく聞こえるかもしれない
→今回の演習ではEthereumをコンピュータとして利用するのみ(Ethereumを少し特徴のある単なるコンピュータとして捉えるだけで十分)

3.1.2

Ethereum

Solidity

Ethereum上でコントラクトを作成する際の最もメジャーな言語

Remix

コントラクトを作成するための統合開発環境

- SolidityはEthereum上でコントラクトを作成するために作られた、オブジェクト指向型の言語
- コントラクトを作成するためにRemixという統合開発環境を利用する(今回はオンライン版を使用)

3.2.1

Ethereumの利用準備

Ethereumネットワークの種類

メインネットワーク
→ Ethereumの本番環境

テストネットワーク
→ Ethereumのテスト環境

プライベートネットワーク
→ Ethereumのローカル環境

51

- メインネットワーク：一般的にEthereumのネットワークといった時に指されるのが、このメインネットワーク
- テストネットワーク: インネットワークと同様に世界中に広がるネットワーク(テストネットワーク内で取引されるEtherは他の通貨に変換することができない)
- プライベートネットワーク: プライベートネットワークはEthereumのクライアントソフトを用いてローカル環境で作るネットワーク

3.2.1

Ethereumの利用準備

プライベートネットワークをローカル環境に構築する

3.2.2

Ethereumの利用準備

ウォレット

仮想通貨を保有するためアプリケーション

MetaMask

ブラウザの拡張として利用するEthereumウォレット

- 仮想通貨を保有するためにはウォレットというアプリケーションを利用する必要がある。
- ウォレットには様々な形式があり、スマートフォンのアプリ形式のもの、Webアプリケーションの形式で利用できるものなどがある。

※MetaMaskをGoogleChromeの拡張として利用できる状態にしてください。

3.2.3

Ethereumの利用

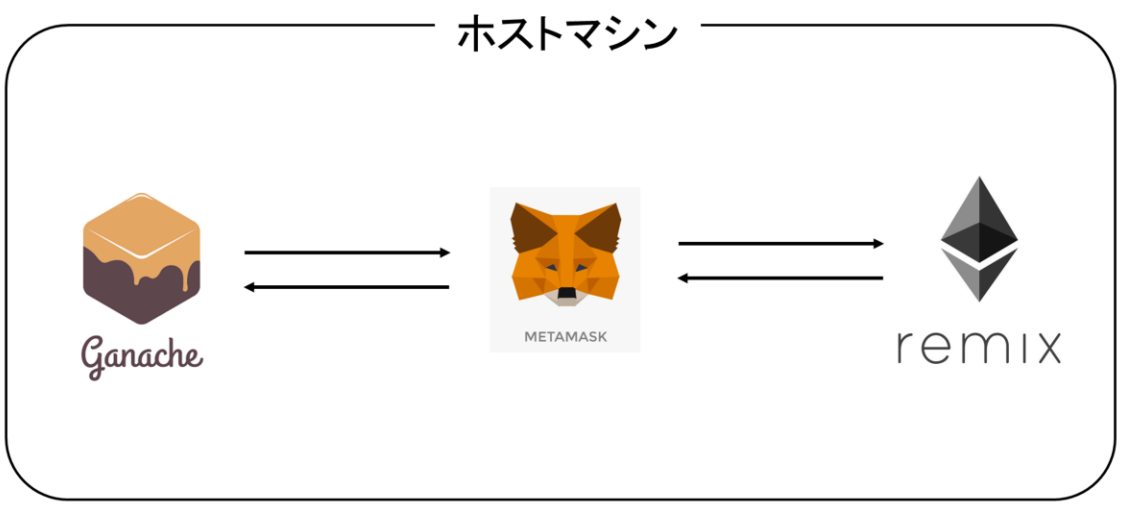
環境構築

Ganache

簡易的にEthereumのローカル環境を作るためのツール

3.2.3

Ethereumの利用



RemixからMetaMaskを経由して、Ganacheにコンロラクトをデプロイする

3.3.1

Ethereumの利用

流通履歴のバックアップをブロックチェーンに保存する

手順

・Ethereumにコントラクトを登録する
・ブラウザからEthereumを利用するためのコードの記述

3.3.1

Ethereumの利用

利用するコントラクト

- 流通履歴を保存する機能
- 流通履歴を検証する機能
- コントラクトのオーナーを登録する機能
- コントラクトのオーナーを削除する機能
- コントラクトを破棄する機能

ソースコードをもとに今回のコントラクトの機能について説明する。

3.3.1

Ethereumの利用

Constructor

コントラクトがブロックチェーンに登録される際に、一度だけ実行される関数

Modifier

関数を実行する前に必ず実行しておきたい処理がある時にこれを利用

selfdestruct

コントラクトを破棄する

- Solidityが元々持っている関数について説明する
- Constructor : コントラクトがブロックチェーンに登録される際に、一度だけ実行される関数
- Modifier : 関数修飾子と呼ばれ、ある関数を実行する前に必ず実行しておきたい処理がある時にこれを利用する。。このコントラクトでは、コントラクトのオーナーのみに実行させたい関数のために、modifierを作成しています。deleteOwner関数とkill関数に利用している。
- selfdestruct: コントラクトを破棄し、利用不可能にするための関数

3.3.1

Ethereumの利用

Web3.js

Web3.jsはローカルまたはリモートの

Ethereumとやりとりできるようにするライブラリ

Ethereumを利用するにあたって、web3.jsと言うjavascriptのライブラリを使用します。Web3.jsは、HTTPまたはIPC接続を使用して、ローカルまたはリモートのEthereumとやりとりできるようにするライブラリ

3.3.1

Ethereumの利用

トレーサビリティシステムにブロックチェーンを利用する

Ethereumへの履歴の登録と検証機能の作成を行う。また、演習の後半ではEthereumの送金機能も作成する。

3.3.1

Ethereumの利用

履歴登録機能

ブロックチェーンを流通履歴を登録する

3.3.2

Ethereumの利用

履歴参照機能

ブロックチェーン上の流通履歴を呼び出す

3.3.3

Ethereumの利用

仮想通貨支払い機能

Etherの送金を行う

Ethereumの利用

管理者機能

・コントラクトのオーナーを削除する

・コントラクト自体を破棄する

4章

スマートコントラクトを用いたシステムの実例

4.1.1

仮想通貨の種類

ブロックチェーンと仮想通貨

パブリックブロックチェーンでは、
システムへ貢献した人に対して仮想通貨で報酬が支払われる
→この時使われる通貨を内部通貨と呼ぶ

- パブリックブロックチェーンでは、管理者を存在させずに同等な権限を持った利用者のみで機能させるために、Proof of Work、Proof of Stakeといったコンセンサスアルゴリズムが用いられている。
- これらのアルゴリズムでは、システムへ貢献した人に対してそれぞれのシステムの内部で利用される仮想通貨で報酬が支払われる。
- ブロックチェーンのシステムの一部として利用されている仮想通貨は内部通貨または、ネイティブ通貨と呼ばれる。

仮想通貨の種類

仮想通貨の種類

内部通貨以外にも仮想通貨は存在する

→ブロックチェーン上のアプリケーションとして作られた通貨

Ethereumをはじめとしたブロックチェーン上にアプリケーションとしては発行される通貨が存在する。(コンピュータの上に様々なポイントサービスが構築されることに似ている)

4.1.2

仮想通貨の種類

ERC

Ethereum上にトークンを発行する際に使われるスマートコントラクトの規格

ERC-20

最もシンプルな形式の通貨を発行することができる

ERC-721

代替不可能な通貨を発行することができる

- ERCとはEthereum Request for Commentsの略
- ERCを利用することで、全な通貨をEthereum上で発行することができる。

- ERC-20: Ethereum上で通貨を発行するための規格であり、Token Standardと呼ばれる最もシンプルな形式の通貨を発行することができる
- ERC-721: Non Fungible Tokenと呼ばれる代替不可能トークンを作成するための規格

4.1.2

仮想通貨の種類

Webアプリケーションに利用する仮想通貨の種類

Ethereumでは

内部通貨であるEtherでもERCに準拠した通貨でも

システム作成の難易度は同じ

- 今回のシステムをオリジナルの仮想通貨を発行して、構築し直すことも可能
- 実際にシステム内部でしか利用できない通貨を発行しているサービスも多い

4.2.1

スマートコントラクトを用いたシステム

Cryptokitties

仮想的な猫を売買するゲーム

・それぞれの猫がERC-721に準じたトークンになっている

・猫の取引に関連する部分だけEthereumが利用されている

・ユーザーは自身のウォレットで通貨を管理する

- 仮想的な猫を育成し、それらをユーザー間で売買するゲーム
- それぞれの猫は代替不可能なトークンであるERC-721となっている
- 猫を売買する際に価値の指標として必要になるデータがトークンの情報として記録され、取引に関連する処理と、取引情報の記録はEthereum上で行われる
- 通常のWebアプリケーションと同様にWebサーバーやアプリケーションサーバー、データベースサーバーにはブロックチェーンは使われていない

4.2.2

スマートコントラクトを用いたシステム

Ether Delta

Ethereum上に構築される分散型取引所

・EtherやERCに準じたトークンの取引を行う

・トークンの取引部分だけEthereumが利用されている

・鍵の管理などは全てユーザーが行う

- EtherDeltaはEthereum上に構築される仮想通貨取引所
- Ethereumの上に存在しない通貨の取引は行うことができない

4.2.3

スマートコントラクトを用いたシステム

著作権の管理

音楽作品情報の存在証明

・コンテンツのハッシュ値などをブロックチェーンに記録する

・独自のブロックチェーンが利用される

- ブロックチェーン技術による音楽作品情報の存在証明
- 音楽作品の「存在証明」のために、音楽作品ごとに「デジタルコンテンツのハッシュ値」「創作者のID」「時刻証明情報」をセットにしてブロックチェーンに記録
- オリジナルのパーミッションドブロックチェーンが利用されている

スマートコントラクト開発実践

令和2年度「専修学校による地域産業中核的人材養成事業」
スマートコントラクトを使用したシステム開発人材の育成

スマートコントラクト開発実践指導マニュアル

令和3年2月

学校法人 麻生塾　麻生情報ビジネス専門学校
〒812-0016　福岡県福岡市博多区博多駅南２丁目１２-３２