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

学校法人 麻生塾 麻生情報ビジネス専門学校

目次

環境構築

スマートコントラクト開発実践の演習を行う際に必要となるツールや環境の構築について説明します。

VirtualBox

VirtualBox とは使用している PC に仮想環境を構築し、他の OS をインストールすることができる仮想化ソフトです。VirtualBox を導入することにより、複数の OS を切り替えて使用することが可能になります。

インストール

1. VirtualBox の公式サイトである「https://www.virtualbox.org/」を開く
2. 「Download VirtualBox 6.0」をクリックする
3. ダウンロードを行った後は、VirtualBox のインストーラを開き、画面の指示に従い VirtualBox をインストールする

VirtualBox はバージョン 5. 0 以上で問題なく演習を行うことができると確認しています。また、VirtualBox を起動する際に、仮想マシンにメモリの割り当てを行います。その際に、メモリを 2GB 以上割り当てることを推奨します。

Visual Studio Code

Visual Studio Code は Microsoft が開発したソースコードエディタです。C#や VisualBasic といった Microsoft 社の開発言語はもちろん、HTML、CSS、JavaScript といったフロントエンドの言語や、Java や Python、SQL といった多くの言語をサポートしています。動作環境も Windows、Linux、macOS に対応しており、無料で利用することができます。この教材の演習では、Web アプリケーションを作成します。その際にソースコードの編集で利用します。

インストール

1. Visual Studio Code の公式サイトである「https://azure.microsoft.com/ja-jp/products/visual-studio-code/」を開く
2. ダウンロードを行った後は、Visual Studio Code のインストーラを開き、画面の指示に従い Visual Studio Code をインストールする

Ubuntu

Ubuntu は Linux のディストリビューションの一つです。仮想マシン内で立ち上げ、サーバーとして利用します。

インストール

1. Ubuntu のダウンロードページである、「https://jp.ubuntu.com/download」を開く
2. Ubuntu Desktop 18.04.3 LTS のダウンロードボタンをクリックする
3. Ubuntu のイメージのダウンロードが完了すると、VirtualBox で Ubuntu を起動する

これら以外に利用するツールは、演習などの際に適宜環境の作成方法について説明します。

1.　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.　Webアプリケーション

1.1.1.　Webアプリケーションとは

Webアプリケーションとは名前の通り、**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**のことを言います。Webアプリケーションはインターネット上にあるサーバー上で動作し、ネットワークを通じてChrome・FireFoxなど普段ホームページをみるのに使っているWebブラウザから利用することができます。例を挙げると、動画配信サービスであるYoutubeや、通販サイトであるAmazonなどがこれに該当します。

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**があります。ネイティブアプリケーションとは、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を指します。Webアプリケーションでは、プログラム本体はネットワーク上のサーバー内にあるのに対し、ネイティブアプリのプログラム本体が保存されるのは手元の端末内です。ただし、ネイティブアプリの中にもインターネットを経由してサーバーとの通信を必要とするアプリも存在しています。アプリ版のYoutubeなどがこれにあたります。

図 1.1 Web アプリとネイティブアプリ

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します。システムの機能や構成に関しては、以降の章で説明します。

1.1.2. Web アプリケーションの構成

ここでは一般的な Web アプリケーションの構成について説明します。はじめには、Web アプリケーションを「**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ていきます。

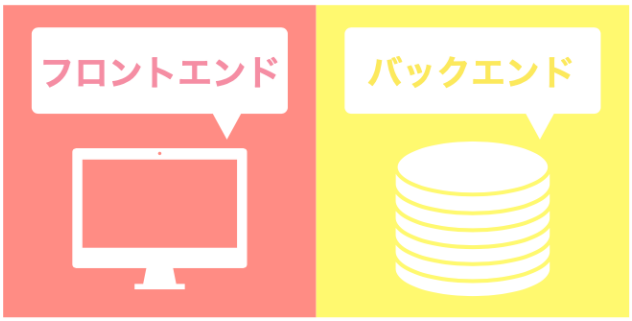

図 1.2 フロントエンドとバックエンド

フロントエンド

フロントエンドとは、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**です。YouTube を例にあげると、サイト自体のデザイン・見た目、動画の検索や再生・停止・コメントの追加といった部分がフロントエンドにあたります。開発する際には、HTML、CSS、JavaScript などを使用します。

バックエンド

フロントエンドが、ユーザーが見える部分・直接操作できる部分なのに対し、バックエンドはユーザーが見えない部分・直接操作できない部分をさします。具体的には、フロントエンドでユーザーが入力した内容に基づき、その**データを処理して結果を返す**ことや、入力された内容を**データベースに保存**するのがバックエンドの役割です。

Web アプリケーションの 3 層構造

Web アプリケーションの「バックエンド」についてさらに詳細に見ていきます。様々な方法はありますが、ここではバックエンドを 3 つの階層に分割します。3 つの階層をそれぞれ「**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**」と呼びます。アプリケーションの構成にもよりますが、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ります。3 つのサーバーについてそれぞれについて説明します。

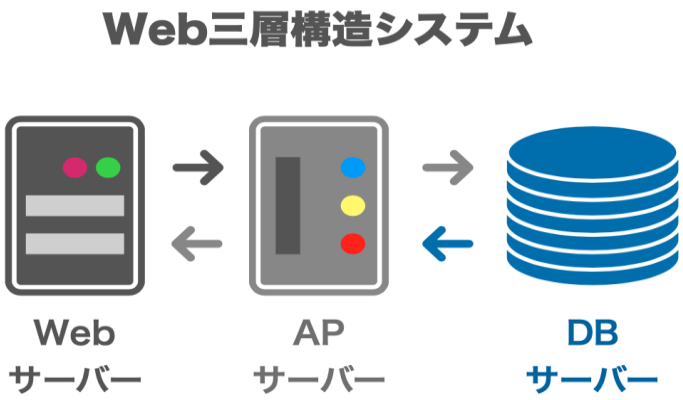

図 1.3Web 三層構造システム

Web サーバー

Web サーバーは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があります。

アプリケーションサーバー

アプリケーションサーバーでは、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します。処理の内容としては、必要な **HTML ファイルを動的に作成**し、Web サーバーへ渡すことや、データベースにアクセスしてユーザーから届いたデータを保存することなどがあります。

データベースサーバー

データベースサーバーは、データの管理を行っている部分であり、Web サイトに必要なユーザーデータや商品**データを保存**しています。データベースの種類の一つに「**リレーショナルデータベース**」というものがあります。これについては次のページで詳しく説明します。

1.1.3. リレーショナルデータベース

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**データベースです。頭文字を取って「**RDB**」と略されることもあります。最も普及しているデータベースシステムの 1 つであり、単にデータベースといった場合はリレーショナルデータベースを指すことが多いです。リレーショナルデータベースでは、表の行（横）を「レコード」、表の列（縦）を「フィールド」、表を「テーブル」と呼びます。テーブル同士の関係を設定し、関連付ける機能をリレーションと呼びます。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**」と呼ばれます。

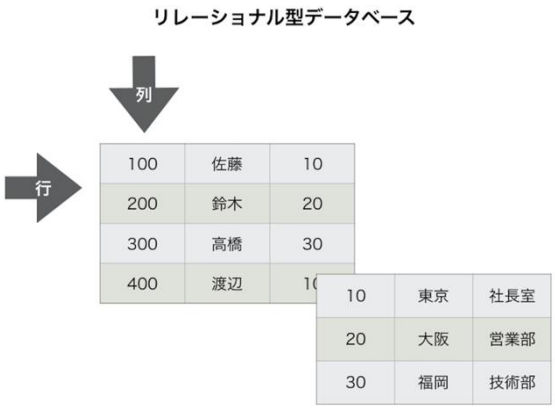

図 1.4 リレーショナル型データベース

MySQL

MySQL は、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です。今回作成するトレーサビリティシステムでは、データベースサーバーに MySQL を利用します。それを踏まえ、次のページで MySQL の環境構築と演習を行います。

MySQL 演習

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 を使った演習を行います。

MySQL の環境の構築

VirtualBox で Ubuntu を立ち上げ、ターミナルアプリを起動します。その後、以下のコマンドを順に実行し、MySQL の環境構築を行います。

MySQL サーバーをインストールします。

```
$ sudo apt install mysql-server
```

インストールした MySQL のバージョンが表示されれば問題ありません。

```
$ mysql --version
```

MySQL を立ち上げます。

```
$ sudo service mysql start
```

MySQL の root ユーザーの設定をします。パスワードの設定が必要になりますので、任意のパスワードを入力してください。

```
$ sudo mysql_secure_installation
```

MySQL に root ユーザーでログインします。先程設定したパスワードが必要になります。

```
$ sudo mysql -u root -p
```

MySQL と SQL に関する理解を深めるために以下の演習を行います。ターミナルアプリで MySQL にログインし、SQL を実行することで以下の作業を順に行ってください。演習の解答は巻末にあります。

1. **sample** という名前のデータベースを作成してください
2. **student** というユーザーを作成してください
3. 作成したユーザーに全てのデータベース、全ての命令の権限を与えてください
4. id (primary key)カラムと name カラムを持つ **users** テーブルを作成してください。
5. **users** テーブルにレコードを追加してください
6. 既存のカラムのフィールドを書き換えてください
7. 全てのレコードを削除してください
8. 作成したデータベースを削除してください

データベースの権限

ここまで SQL を使用した演習を行うと、以下のようなことがわかるのではないかと思います。

1. 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。

2. 権限さえ持っていればデータベース内のデータの書き換え、構造の変更、削除まであらゆることができる。

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です。しかし、実際に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することができます。このように既存のデータベースシステムは管理者により管理されており、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てしまいます。このような問題を**管理者の存在しないシステムであるブロックチェーン**を用いることで防ぐことが可能になります。

この教材で作成する Web アプリケーションは**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**ことでシステム全体の改ざん耐性を高めます。

1.2.　トレーサビリティシステム

1.2.1.　トレーサビリティ

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であり、日本語では「**追跡可能性**」と訳されます。具体的には、対象とする物品の**流通履歴を確認できる状態**を言います。この仕組みによりその製品が「**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ます。近年では製品の品質向上に加え、製品に対する安全意識の高まりからトレーサビリティの重要度が増しており、自動車や電子部品をはじめ、食品や医薬品など**業界問わず幅広い分野に浸透**しています。トレーサビリティが確立できていると、たとえば出荷後の製品に問題が発生した場合、部品の使用実績にさかのぼって調査して原因を究明し、同様の問題が発生する可能性がある製品を抽出することができます。

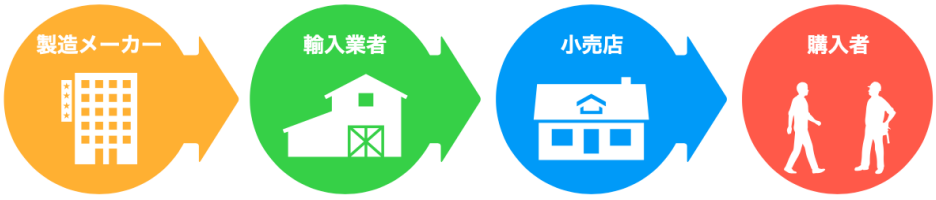

図 1.5 トレーサビリティシステム

1.2.2.　トレーサビリティシステムの実例

有機野菜のトレーサビリティシステム

この実証実験が行われた宮崎県の綾町は、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ます。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**」ということが、ブロックチェーンを通じて証明されています。以下に手法を示します。

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、ブロックチェーンの台帳に記録する
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

こうして出荷された生産物は、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ます**。同封された IoT センサーは様々な値を計測することができます。その中に照度を計測する機能もあります。これにより、購入した店舗までの経路で箱が開けられて、中身がすり替えられていたり、直射日光にさらされていたり、別のものと入れ替えられたりしていなかったかといったことも記録することができます。また、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**することができます。

記録された情報は段ボールに取り付けられた NFC タグから QR コードを読み取れるようになっており、専用の Web ページから書き込まれたこの野菜についての一連の情報を確認できるようになっています。

1.3.　トレーサビリティシステムの構成／環境構築

1.3.1. トレーサビリティシステムの概要

この教材では、簡易的なトレーサビリティシステムを作成します。このシステムで追跡するものは「**高級バッグ**」を想定します。まずは、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していきます。

図 1.6 トレーサビリティシステムの構成

1.3.2.　トレーサビリティシステムの構成

今回作成するトレーサビリティシステムは、全てローカル環境で完結させます。使用するツールは以下になります。

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。これから利用される場合には、以下のサイトより準備をしてください。

https://www.google.com/intl/ja_jp/chrome/

Virtual Box

Virtual Box で仮想マシンを立ち上げ、その上に 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を設置します。Ubuntu を立ち上げ、ターミナルアプリを開いた状態にしておいてください。

Apache

Web サーバーには Apache を利用します。Apache の環境構築については、これから行っていきます。

PHP

今回作成するアプリケーションのサーバーサイドは PHP で実装していきます。PHP の環境構築については、これから行っていきます。

MySQL

データベースには MySQL を利用します。ここまでの演習を通して、仮想マシン上で環境構築は終わっているものとします。

1.3.3.　フレームワーク

Laravel

今回のシステムでは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 を使用します。Laravel は 2011 年にリリースされた PHP のフレームワークです。PHP フレームワークの中では後発ながら、PHP の中でも代表的なフレームワークとなっています。Laravel は非常に柔軟で多機能なフレームワークであり、Web アプリケーションであれば全般的に向いています。多機能で学習コストが低いという観点から、少人数でスピード感を持って開発したいときなどにはそのメリットを享受できます。また、Laravel では MVC モデルを採用しています。MVC モデルとは、処理を **Model（データ処理）**、**View（画面表示）**、**Controller（全体の制御）**の各機能に分けて、パーツごとに開発を行なっていくものです。

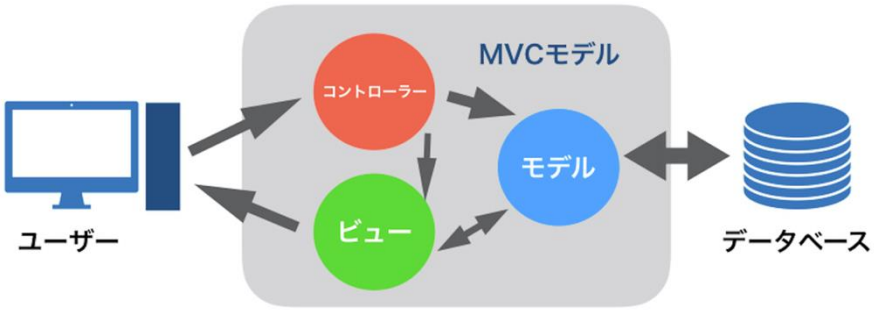

図 1.7 MVC モデル

1.3.4.　環境構築

仮想マシン内に開発環境を構築します。仮想マシン内のターミナルアプリから行ってください。

Laravel の環境構築

Laravel プロジェクトを立ち上げるための環境を作ります。以下のコマンドを順に実行してください。

必要な PHP 拡張パッケージをインストールします。

```
$ sudo apt install php-mbstring php-xml php-json
```

Composer というパッケージ管理ソフトをインストールします。

```
$ sudo apt install curl
$ sudo apt install7.4-cli
$ curl https://getcomposer.org/installer | php
$ sudo mv composer.phar /usr/local/bin/composer
```

Apache の環境構築

今回のシステムで利用する Web サーバーである Apache の設定を行います。以下のコマンドを実行し、Apache をインストールします。

```
$ sudo apt install apache2
```

SSH の環境構築

ここからは、ホストマシンから仮想マシンに SSH 接続して作業することを推奨します。SSH の設定方法について説明します。以下のコマンドを実行してください。

```
$ sudo apt install net-tools
$ sudo apt-get install openssh-server
```

仮想マシンの IP アドレスは以下のコマンドで確認することができます。

```
$ ifconfig
```

仮想マシンの IP アドレスを確認することができたら、ホストマシンから SSH 接続します。その際に必要となる、ユーザー名と、パスワードは仮想マシンを立ち上げる際に設定したものです。

1.3.5. データベース作成

MySQL でトレーサビリティシステム用のデータベースを作成します。以下の作業は、仮想マシン内のターミナルアプリまたは、仮想マシンに ssh 接続した状態で行ってください。以下の作業は MySQL の環境の構築ができていることを前提として行います。以下の作業を順に行ってください。

MySQL にログインします。

```
$ sudo mysql -u root
```

トレーサビリティシステム用のデータベースを作成します。名前は「traceability」とします。

```
> create database traceability;
```

1.4.　トレーサビリティシステム作成

1.4.1.　プロジェクト作成

プロジェクト名は「**traceability**」とします。以下のコマンドを実行してください。

```
$ cd /var/www/html

$ sudo chmod 777 /var/www/html

$ composer create-project --prefer-dist laravel/laravel traceability
```

これ以降の作業については、 traceability ディレクトリの直下で行ってください。

```
$ cd traceability
$ composer install
```

環境設定ファイルの編集

Laravel の設定ファイルはプロジェクトのルートディレクトリ直下の.env ファイルです。このファイルの内容を書き換えます。編集する部分はデータベースに関連する部分です。先程行ったデータベースの情報を .env ファイルに書き込んでください。以下に例を示します。

.env 修正箇所

```
DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306
```

```
DB_DATABASE=traceability

DB_USERNAME=student

DB_PASSWORD=password
```

認証機能の作成

　Laravel の laravel/ui パッケージは、認証に必要なルートとビューを簡単に、安全に作成するための、シンプルなコマンドを提供しています。今回はこれを利用して認証機能を作成します。

```
$ sudo apt install npm

$ npm install && npm run dev

$ composer require laravel/ui

$ php artisan ui vue --auth
```

　上記のコマンドで、メールアドレスとパスワードでのユーザー認証、ユーザーの新規作成、パスワードリセット機能などが完成します。

1.4.2.　マイグレーションファイル作成

今回作成するアプリケーションのデータベースの構造は、以下のようになっています。

システムの利用者についてのテーブル

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

商品についてのテーブル

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

流通経路についてのテーブル

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

Laravel でマイグレーションファイルは以下のコマンドで作成します。

```
$ php artisan make:migration マイグレーションファイル名
```

それぞれのテーブルについてマイグレーションファイルを作成していきます。ただし、認証機能の作成の際に、users テーブルを作成するためのマイグレーションファイルは自動的に作成されています。以下にマイグレーションファイル作成のためのコマンドの例を示します。名は createContentsTable とします。

```
$ php artisan make:migration createContentsTable
```

上記のコマンドにより、プロジェクトの database/migrations ディレクトリにマイグレーションファイルが作成されます。

今回作成した contents テーブルに関するマイグレーションファイル以外に、**users テーブルにカラムを追加するためのマイグレーションファイル**、**traces テーブルを作りカラムを設定するためのマイグレーションファイル**が必要になりますので作成してください。正しいマイグレーションファイルについては、巻末に載せてあります。

以上が完成しましたら、マイグレーションをデータベースに適応します。以下のコマンドを実行してください。

```
$ php artisan migrate
```

以上で、トレーサビリティシステムに必要となるデータベースは完成しました。

1.4.3.　利用者機能作成

以下に、トレーサビリティシステムの機能一覧を表示します。

/	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

Model の作成

モデルを作成するためのコマンドは以下です。

```
$ php artisan make:model モデル名
```

今回作成する必要となるモデルは以下の 4 つです。

- User モデル
- Admin モデル
- Content モデル
- Trace モデル

上記の 4 つのモデルをコマンドにより作成してください。以下に User モデルの作成例を示します。

```
$ php artisan make:model User
```

モデルはプロジェクトの App/Models ディレクトリ内に作成されます。作成したモデルファイルの中身はそれぞれ以下のようにしてください。

User.php

```
<?php

namespace App¥Models;

use Illuminate¥Contracts¥Auth¥MustVerifyEmail;
use Illuminate¥Database¥Eloquent¥Factories¥HasFactory;
use Illuminate¥Foundation¥Auth¥User as Authenticatable;
use Illuminate¥Notifications¥Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;
```

```
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function contents ()
    {
        return $this->hasMany('App¥Models¥Content');
    }

    public function trace ()
    {
        return $this->hasMany('App¥Models¥Trace');
    }
```

```
}
```

Admin.php

```
<?php

namespace App¥Models;

use Illuminate¥Contracts¥Auth¥MustVerifyEmail;
use Illuminate¥Database¥Eloquent¥Factories¥HasFactory;
use Illuminate¥Foundation¥Auth¥User as Authenticatable;
use Illuminate¥Notifications¥Notifiable;

class Admin extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
```

```
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

Content.php

```
<?php

namespace App¥Models;

use Illuminate¥Database¥Eloquent¥Factories¥HasFactory;
use Illuminate¥Database¥Eloquent¥Model;

class Content extends Model
{
    public function user ()
    {
        return $this->belongsTo('App¥Models¥user');
    }

    public function trace()
    {
        return $this->hasMany('App¥Models¥Trace');
    }

    public static function boot()
    {
        parent::boot();

        static::deleting(function($content){
            $content->traces()->delete();
        });
    }
```

```
    protected $table = 'contents';

    protected $fillable = ['brand', 'name', 'price', 'information'];

}
```

Trace.php

```
<?php

namespace App¥Models;

use Illuminate¥Database¥Eloquent¥Factories¥HasFactory;
use Illuminate¥Database¥Eloquent¥Model;

class Trace extends Model
{
    public function user()
    {
        return $this->belongsTo('App¥Models¥user');
    }

    public function content()
    {
        return $this->belongsTo('App¥Models¥Content');
    }

    protected $fillable = ['comment', 'latitude', 'longitude'];
}
```

Controller の作成

```
$ php artisan make:controller　コントローラー名
```

今回必要となるコントローラーは以下になります。

- User コントローラー
- Content コントローラー
- Trace コントローラー
- Admin コントローラー

以下に UserController の作成例を示します。

```
$ php artisan make:controller UserController
```

必要なコントローラーを上記のコマンドを例にして作成してください。コントローラーは App/Http/Controllers 内に作成されます。

UserController.php

```
<?php

namespace App¥Http¥Controllers;

use App¥Models¥Content;
use App¥Models¥User;
use Illuminate¥Http¥Request;
use Auth;
use Illuminate¥Support¥Facades¥Hash;
use Illuminate¥Support¥str;

class UserController extends Controller
{
    public function user_info()
    {
        $user = Auth::user();
        $contents = Content::where('user_id', $user->id)->orderBy('created_at', 'desc')->get();
        return view('user_info', compact('user', 'contents'));
    }
```

```
    public function user_info_edit()
    {
        $user = Auth::user();
        return view('user_info_edit', compact('user'));
    }

    public function user_updete(Request $request)
    {
        $user = Auth::user();
        $user->name = $request->name;
        $user->email = $request->email;
        $user->address = $request->address;
        $user->save();
        return redirect()->route('user_info_edit_success');
    }

    public function user_info_edit_success()
    {
        return view('user_info_edit_success');
    }

    public function user_logout()
    {
        Auth::logout();
        return redirect()->route('login');
    }
}
```

ContentController.php

```
<?php

namespace App¥Http¥Controllers;

use App¥Models¥Content;
use App¥Models¥User;
use Illuminate¥Http¥Request;
```

```
use Auth;
use Illuminate¥Support¥Facades¥Hash;
use Illuminate¥Support¥str;

class ContentsController extends Controller
{
    public function index()
    {
        $auth = Auth::user();
        return view('index', compact('auth'));
    }

    public function add_content(Request $request)
    {
        $content = $request->session()->get('content');
        return view('content.add_content', compact('content'));
    }

    public function add_content_store(Request $request)
    {
        $request->validate([
            'brand' => 'required',
            'name' => 'required|unique:contents',
            'price' => 'required',
            'information' => 'required',
        ],
        [
            'brand.required' => 'ブランド名は必須です',
            'name.required' => '製品名は必須です',
            'name.unique' => 'その製品名はすでに使われてます。別の製品名を入力してください',
            'price.required' => '出荷価格は必須です',
            'information.required' => '製品詳細は必須です',
        ]);

        $imgdata = $request->except('imagefile');
        $content_file = $request->file('imagefile');

        if($request->hasFile('imagefile') && $content_file->isValid()){
            $image_path = $content_file->store('public');
```

```
            $read_image_path = str_replace('public/', 'storage/', $image_path);
        }

        $content = new Content;
        $validated = $request->validate([
            'brand' => 'required',
            'name' => 'required|unique:contents',
            'price' => 'required',
            'information' => 'required',
        ],
        [
            'brand.required' => 'ブランド名は必須です',
            'name.required' => '製品名は必須です',
            'name.unique' => 'その製品名はすでに使われてます。別の製品名を入力してください',
            'price.required' => '出荷価格は必須です',
            'information.required' => '製品詳細は必須です',
        ]);
        $content->fill($validated);
        $content->user_id = $request->user()->id;
        $content->identifier = Str::random(10);
        $content->image_path = $read_image_path;
        $content -> save();
        $request->session()->put('content', $content);
        return redirect()->route('content.add_content_success');
    }

    public function add_content_success(Request $request)
    {
        $content = $request->session()->get('content');
        return view('index',compact('content'));
        // return view('content.add_content_success',compact('content'));
    }

    public function content_delete($id)
    {
        Content::where('id', $id)->delete();
        return redirect()->route('user_info');
    }
```

```
}
```

TraceController.php

```
<?php

namespace App¥Http¥Controllers;

use App¥Models¥Content;
use App¥Models¥Trace;
use Auth;
use Illuminate¥Http¥Request;
use Illuminate¥Support¥Carbon;
use Illuminate¥Support¥Facades¥DB;

class TraceController extends Controller
{
    public function content_detail(Request $request)
    {
        $request->validate([
            'search' => 'required|max:10',
        ]);

        $content = Content::where('identifier', $request->input('search'))->first();

        if($content === null) {
            return redirect()->route('index')->with('identifier_error', '入力した商品は登録されていません');
        } else {
            $user = Auth::user();
            $dateTime = Carbon::now();
            $traces = Trace::where('content_id', $content->id)->orderBy('created_at', 'desc')->get();
            return view ('trace.content_detail', compact('user', 'dateTime', 'content', 'traces'));
        }

    }

    public function content_detail_store(Request $request)
    {
        $request->validate([
```

```
            'comment' => 'required',
            'latitude' => 'required',
            'longitude' => 'required',
        ],
        [
            'comment.required' => 'コメントを入力してください',
            'latitude.required' => '緯度は必須です',
            'longitude.required' => '経度は必須です',
        ]);

        $trace = new Trace;
        $validated = $request->validate([
            'comment' => 'required',
            'latitude' => 'required',
            'longitude' => 'required',
        ],
        [
            'comment.required' => 'コメントを入力してください',
            'latitude.required' => '緯度は必須です',
            'longitude.required' => '経度は必須です',
        ]);
        $trace->fill($validated);
        $trace->user_id = $request->user_id;
        $trace->content_id = $request->content_id;
        $trace->save();
        return redirect()->route('content_detail');
    }

    public function comment_detail()
    {
        $auth = Auth::user();
        $dateTime = Carbon::now();
        return view ('trace.comment_detail', compact('auth', 'dateTime'));
    }

    public function comment_detail_update()
    {

    }
```

```
}
```

Routing の作成

web.php

```
<?php

use Illuminate¥Support¥Facades¥Route;
use App¥Http¥Controllers¥AdminController;
use App¥Http¥Controllers¥ContentsController;
use App¥Http¥Controllers¥UserController;
use App¥Http¥Controllers¥TraceController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/


Auth::routes();

//Admin 用
Route::group(['prefix' => 'admin', 'middleware' => 'admin.auth'], function(){
        Route::get('/', [AdminController::class, 'index'])->name('admin_index');
        Route::get('/users_list', [AdminController::class, 'users_list'])->name('users_list');
        Route::get('/users_list/{id}', [AdminController::class, 'users_show'])->name('users_show');
        Route::get('/contents_list', [AdminController::class, 'contents_list'])->name('contents_list');
        Route::get('/contents_list/{id}', [AdminController::class, 'contents_show'])->name('contents_show');
});
```

```
 //ログインユーザー用
 Route::group(['middleware' => 'auth'], function(){
    Route::get('/', [ContentsController::class, 'index'])->name('index');

    Route::prefix("content")->group(function(){
        Route::get('/add_content', [ContentsController::class, 'add_content'])->name('content.add_content');
        Route::get('/add_content_success', [ContentsController::class, 'add_content_success'])-
>name('content.add_content_success');
        Route::post('/store', [ContentsController::class, 'add_content_store'])->name('content_store');
        Route::delete('/{id}', [ContentsController::class, 'content_delete'])->name('content_delete');
    });
    Route::prefix("trace")->group(function(){
        Route::get('/content_detail', [TraceController::class, 'content_detail'])->name('content_detail');
        Route::post('/content_detail_sotre', [TraceController::class, 'content_detail_store'])-
>name('content_detail_store');
    });
    Route::prefix("user")->group(function(){
        Route::get('/info', [UserController::class, 'user_info'])->name('user_info');
        Route::get('/logout', [UserController::class, 'user_logout'])->name('user_logout');
        Route::get('/info_edit', [UserController::class, 'user_info_edit'])->name('user_info_edit');
        Route::get('/info_edit_success', [UserController::class, 'user_info_edit_success'])-
>name('user_info_edit_success');
        Route::post('/updete', [UserController::class, 'user_updete'])->name('user_updete');

    });
 });

 //非ログインユーザー用
 Route::get('/', [ContentsController::class, 'index'])->name('index');
 Route::get('trace/content_detail', [TraceController::class, 'content_detail'])->name('content_detail');
```

View の作成

システムの見た目である view ファイルを作成します。内容については巻末に示します。

1.4.4.　管理者機能作成

次に管理者の機能を作成します。機能一覧は以下になります。

/admin	管理者トップ画面
/admin/contents_list	登録商品一覧
/admin/contents_list/{id}	商品詳細
/admin/users_list	ユーザー一覧
/admin/users_list/{id}	ユーザー詳細

管理者は、user テーブルのの **role カラムが admin** となっているユーザーです。初期の管理者を作るために、seeder ファイルを作成します。以下のコマンドで必要な seeder を作成します。

```
$ php artisan make:seeder createAdminSeeder
```

プロジェクト内の database/seeders ディレクトリに createAdminSeeder.php ファイルができますので、中身を以下のようにしてください。

```
<?php

namespace Database¥Seeders;

use Carbon¥Carbon;
use Illuminate¥Database¥Seeder;
use Illuminate¥Support¥Facades¥DB;

class UserTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
```

```
     *
     * @return void
     */
    public function run()
    {
        DB::table('users')->insert([
            'name' => 'admin',
            'email' => 'admin@example.com',
            'password' => bcrypt('admin0000'),
            'role' => 'admin',
            'created_at' => Carbon::now(),
            'updated_at' => Carbon::now(),
        ]);
    }
}
```

Seeder の情報をデータベースに反映します。以下のコマンドを実行してください。

```
$ php artisan db:seed
```

管理者機能で必要な AdminController を作成します。

AdminController.php

```
<?php

namespace App¥Http¥Controllers;

use Illuminate¥Http¥Request;
use App¥Models¥User;
use App¥Models¥Content;
use App¥Models¥Trace;
```

```
use Auth;
use Illuminate¥Support¥Carbon;

class AdminController extends Controller
{
    public function index()
    {
        return view('admin.index');
    }

    public function users_list()
    {
        $users = User::orderBy('created_at', 'desc')->get();
        return view('admin.users_list', compact('users'));
    }

    public function users_show($id)
    {
        $user = User::find($id);
        return view('user_info', compact('user'));
    }

    public function contents_list()
    {
        $contents = Content::orderBy('created_at', 'desc')->get();
        return view('admin.contents_list', compact('contents'));
    }

    public function contents_show($id)
    {
        $content = Content::find($id);
        $dateTime = Carbon::now();
        $user = Auth::user();
        $traces = Trace::where('content_id', $content->id)->orderBy('created_at', 'desc')->get();
```

```
        return view('trace.content_detail', compact('traces', 'content', 'dateTime', 'user'));
    }
}
```

次に middleware の作成を行います。今回作成する middleware には、ユーザーが管理者であるかどうかで認可を行う機能を持たせます。

```
$ php artisan make:middleware AdminAuth
```

プロジェクトの app/Http/Middleware ディレクトリに、AdminAuth.php ファイルができていますので、以下のように書き換えてください。

```
<?php

namespace App¥Http¥Middleware;

use Closure;
use Illuminate¥Http¥Request;

class AdminAuth
{
    /**
     * Handle an incoming request.
     *
     * @param  ¥Illuminate¥Http¥Request  $request
     * @param  ¥Closure  $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        if(auth()->check() && auth()->user()->role == 'admin'){
            return $next($request);
```

```
        }

        return redirect('/');
    }
}
```

Web サーバーの設定

これで機能は完成しました。動作の確認のために、Web サーバーの設定を行います。Apache の設定ファイル内の DocumentRoot を、今回作成したプロジェクト内の public ディレクトリに設定してください。

また、VirtualBox の設定を変更する必要があります。仮想マシンのネットワーク設定から、「割り当て」の設定を「**ホストオンリーアダプター**」に設定してください。この状態で、仮想マシンの IP アドレスを ifconfig コマンドで確かめ、ホストマシンから、**http://IP アドレス**を開くと、作成したプロジェクトを見ることができます。環境にもよりますが、VirtualBox の設定を変更していないならば、**http://192.168.56.101** で作成したアプリを確認することができます。

2. Web アプリケーションと ブロックチェーン

2.1 ブロックチェーン

2.1.1 ブロックチェーンの特徴

ここからは、今回作成するシステムにとって重要なブロックチェーンの特徴を 2 つ紹介します。

非中央集権

ブロックチェーンは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されています。このように特別な権限を持った管理者が存在しない状態を**非中央集権的**な状態と呼びます。ブロックチェーンが非中央集権であることで、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**事態を防ぐことができます。

高い改ざん耐性

ブロックチェーンは既存のデータベースシステムに比べ、極めて高い改ざん耐性を持っています。この特徴は主にブロックチェーンの 2 つの特性によって生まれています。1 つ目はブロックチェーンの**独自のデータ構造**によるものです。2 つ目に世界中の**独立した大量のノードが記録を保存**していることが挙げられます。

2.1.2 ブロックチェーンの種類

現在運用されているブロックチェーンは世界中に無数にあり、それらのブロックチェーンにはそれぞれの作られた目的や特徴があります。これらのブロックチェーンは管理者の有無という点で「**パブリックブロックチェーン**」と「**パーミッションドブロックチェーン**」に分類することができます。

パブリックブロックチェーン

パブリックブロックチェーンとは「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**であり、誰でもブロックチェーンのネットワークに参加することができます。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 などの制約はありません。ネットワークに参加するとは、**ブロックチェーンを動かすための一員となる**ことができるということです。Bitcoin を例にすると、取引が正当なもであるか確認することや、取引の記録を保存することができます。

パーミッションドブロックチェーン

パーミッションドブロックチェーンは、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です。**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です。

ブロックチェーンの特徴として、「非中央集権」と「高い改ざん耐性」を上げましたが、これらの特徴はパブリックブロックチェーンのものです。パーミッションドブロックチェーンでは、これらの特徴についてはパブリックブロックチェーンに比べると劣る代わりに、処理の速度や量などの点で優れています。パブリックブロックチェーンとパーミッションドブロックチェーンでは特徴が大きく異なります。

2.1.3 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です。つまり、価値の移転に特化したブロックチェーンであると言うことができます。また、Bitcoin はその特徴から金(きん)に例えられ、**DigitalGold** とも呼ばれます。ブロックを作成し、新たな通貨を発行することがマイニング(採掘する)と呼ばれるのは、このような背景からです。また「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ります。

2.1.4 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です。Ethereum には内部通貨として Ether が存在し、Ethereum 上でシステムを動かす際の手数料を支払うために利用されます。Bitcoin と同様に Ether は通貨としての利用も可能です。また、Ethereum は、現在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であり、日本語の情報も多いことから、このテキストでは Ethereum を用いて、Web アプリケーションの開発を行います。
Ethereum の詳細については次の章で確認します。

2.1.5 スマートコントラクト

まずはスマートコントラクトの概要と特徴について説明します。一言にスマートコントラクトと言ってもその解釈は様々あり、例として以下が挙げられます。

- 賢い契約
- 契約の自動化
- プログラム化された契約
- 自動執行権のある契約
- スマートコントラクト・プラットフォーム上で動く契約

紹介したスマートコントラクトの解釈からわかるようにスマートコントラクトには厳密な定義はありません。これらの共通点としては**コンピュータによって自動的に取引が行われる**点です。これらをまとめると、**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ていることがわかるのではないかと思います。

しかし、馴染みのあるありふれた仕組みであるスマートコントラクトがここ数年で注目を浴びています。これは、ブロックチェーンの誕生と関係しています。ブロックチェーンにより、これまで行うことができたかった形式のスマートコントラクトが実現できるようになりました。同時にスマートコントラクトという言葉は、ブロックチェーンを利用したシステムを指すことが多くなっています。決してお金の関係する取引に関連するものだけを示すものではありません。この教材ではこれ以降**スマートコントラクトはブロックチェーンを利用したシステムのことを指す**こととします。

2.2 スマートコントラクトの利点と課題点

2.2.1 利点

管理者不在

スマートコントラクト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特徴の 1 つである**管理者を必要としない**という特徴を活かしたシステムを作成することができます。

ここでは、例として Web アプリケーションのデータベースサーバーに注目します。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ます。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ます。このような仕組みにより管理者の権限を奪われることや、管理者自身が不正をすることで、ユーザーが登録したデータが書き換えられることや消されてしまう可能性があります。また、書き込まれた内容に対する不正な検閲によりデータが削除される可能性もあります。

このような懸念を少なくするために、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して利用します。Web アプリケーションのデータをブロックチェーンに保存することで、**管理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**することができます。

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特徴の 1 つである**改ざん耐性の高さ**を利用することができます。 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ます。これにより、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ます。

2.2.2 課題点

非中央集権には限界がある

ブロックチェーンの特徴の 1 つに管理者を必要とせずにシステムが稼働する非中央集権性が挙げられます。しかし、多くのスマートコントラクトではこの特徴を活かすことができていません。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**現実があります。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っていると言えます。このため、 サービスの提供者がサービスを終えてしまうと、ブロックチェーン上にサービスに関連するプログラムやデータは残っているにも関わらず、サービスを利用することができなくなってしまいます。

スケーラビリティ問題

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**があります。具体的には Bitcoin では 1 秒間に最大 7 件、Ethereum では最大 15 件しか処理することができません。これに対して、大手のクレジットカード会社では 1 秒間に

数千件の処理を行うことができます。これと比較すると、いかに Bitcoin や Ethereum が一定時間に処理することのできるトランザクションの数が少ないかわかると思います。

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ります。

プログラムの修正

ブロックチェーンの特徴の 1 つとして**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**点が挙げられます。この点は大きな利点とも捉えることができ、また場合によっては難点として捉えることもできます。

ブロックチェーンに記録されたプログラムは誰にも内容を書き換えることはできません。つまり、攻撃などによりコードが改変されることや、消されてしまうことはありません。この点は正しいプログラムをデプロイした場合には、大きな利点として働きます。しかし、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ません。

このような場合に備えて、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ります。また、プログラムをアップデートする際にも事前に工夫が必要になります。

このように、スマートコントラクトの開発では、ブロックチェーンを利用するために生じるこれまでの開発と異なる特徴があります。ブロックチェーンの利点と難点に関する知識をしっかりと持った上で開発を行うことでより良いシステムを作ることができます。

2.3 スマートコントラクトの構成

2.3.1 スマートコントラクトの構成例

ここからは作成した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準備を進めていきます。以下にこれから作成するシステムの構成図を示します。

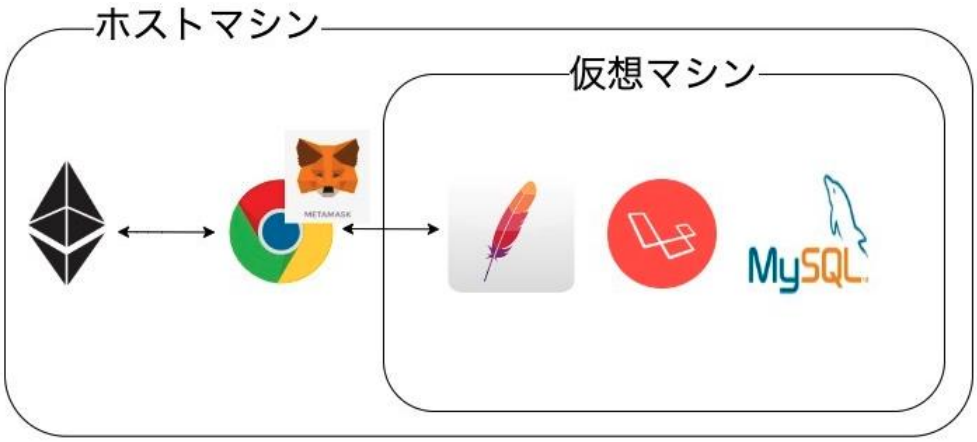

図 2.1　システムの構成図

これら以外にもスマートコントラクトの構成、つまりブロックチェーンの利用方法については考えることができますが、構成を考える際には、ブロックチェーンならではの特徴について考慮する必要があります。考慮する点として、「**鍵の管理**」と「**手数料**」について以降で説明します。

2.3.2 秘密鍵の管理

スマートコントラクトを利用するためには、利用するブロックチェーンでアカウントを作成しなければなりません。アカウントとは**ブロックチェーン上で仮想通貨を管理するための口座**のよう

な役割を果たすものです。それぞれのアカウントには秘密鍵が紐付き、そのアカウントの通貨を利用する、つまりトランザクションを発行する際には、この**秘密鍵で署名を行う必要**があります。この鍵の管理が一般の利用者にとって大きな課題となります。

課題となる理由について、銀行口座を例にして説明します。銀行口座のパスワードは比較的桁数が少ないため、暗記してしまっている人も多いのではないかと思います。また、仮にパスワードを忘れた際にも、銀行で口座を開設した際の個人情報などからパスワードを再発行してもらうことが可能です。これに対して、ブロックチェーンのアカウントに紐づく秘密鍵はアルファベットと数字が混じった文字列になっており、更に桁数も多く銀行のパスワードなどに比べると覚えることは現実的ではありません。

また、アカウントを作る際には銀行のような機関で作ってもらうのではなく、専用のアプリやコマンドラインから簡単に行うことができます。これらのアプリやソフトウェアはインターネットから簡単に入手することができ、専門の機関や企業などは必要ありません。また、この際に銀行口座のように名前、住所などを入力する必要はありません。つまり、ブロックチェーンの**アカウントは一切の個人情報に結びついていません**。また、ブロックチェーンは管理者が存在しない分散自律型の仕組みであることから、鍵を紛失した際に**鍵を再発行してくれる機関は存在しません**。つまり、アカウントにどれだけ大量の通貨を保有していたとしても、秘密鍵を紛失してしまうと、その通貨は二度と使うことができなくなります。IT の知識を持っている人でも秘密鍵の管理には細心の注意を払います。これに対して、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でしょうか。この点について開発者は理解しておく必要があります。

これに対して、ユーザーが秘密鍵を管理せず、サービスを提供する**企業が仮想通貨の保有を代替**することもできます。しかし、これには大量の通貨をサービスの提供者が保有する必要があり、サービスの提供者に取ってはリスクであるとも言えます。このような通貨の保有、鍵の管理という点を考慮してシステムの構成を考える必要があります。

2.3.3 手数料の支払い

ブロックチェーンを利用した Web アプリケーションを利用するためにはブロックチェーンに対して、トランザクションを発行する必要があります。トランザクションとは、ブロックチェーンに対する処理の要求のことです。これを発行する際には発行者が**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ります**。この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**があります。つまり、Bitcoin を用いたスマートコントラクトであれば bitcoin を、Ethereum を用いたブロックチェーンであれば Ether によって手数料を支払わなければなりません。

ここで一つ問題があります。それは、一般ユーザーの**仮想通貨の保有率**です。仮に、今日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ます。しかし、大半の人は仮想通貨を保有していないため、このサービスを利用することができません。このように一般のユーザーが仮想通貨を保有していないため、ブロックチェーンを用いたサービスを利用することができず、普及させる際のボトルネックになります。この問題の解決策はいくつか提案され、実現されつつあります。最もシンプルな方法としてはブロックチェーンを用いたサービスを提供する企業が**仮想通貨の支払いを肩代わりすることです。**企業がユーザーの代わりに通貨の保有やトランザクションの発行を代行することで、ユーザーが直接仮想通貨を保有する必要がなくなります。トランザクションの発行にかかった手数料は企業が負担する、またはかかった費用を法定通貨に換算して、企業がユーザーに請求するなどが考えられます。この方法ではユーザーが仮想通貨の購入や、鍵の管理などを一切行う必要がありません。その反面、特定の企業が大量の通貨を管理することになるため、仮想通貨取引所からの鍵の流出による通貨の盗難事件と同じような事件が発生するリスクがあります。

3.　スマートコントラクトを用いたシステム構築

3.1 Ethereum

3.1.1 Ethereum の構成

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です。この特徴から**ワールドコンピュータ**とも呼ばれます。その言葉の通り世界中の Ethereum ネットワークに参加するコンピュータにより、一つの**仮想的なコンピュータである Ethereum Virtual Machine (EVM)** が構成されています。

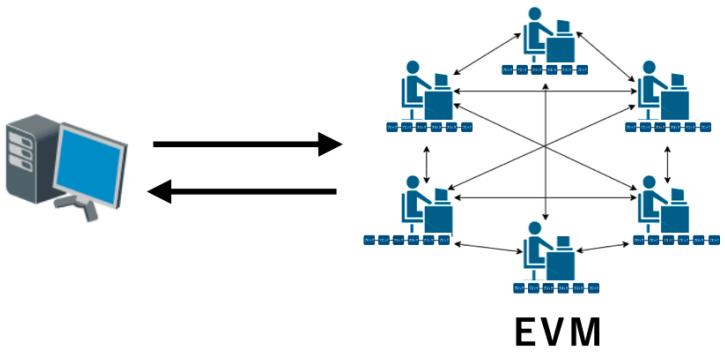

図 3.1 Ethereum の構成

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ませんが、今回の演習では Ethereum をコンピュータとして利用するのみです。つまり、

Ethereum の内部構造を詳細に知る必要はなく、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です。そのように捉えると、利用者が知っておく必要があるのは Ethereum の内部的な構造ではなく、Ethereum のインターフェースだけでよくなります。みなさんが利用しているスマートフォンを使うためには、スマートフォンの中身を知っておく必要がないのと同じです。

3.1.2 Solidity

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です。C++や Javascript の影響を受けて設計され、現在 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であると言えます。今回はこの Solidity を利用して、トレーサビリティシステムのデータベースの**検証機能**を作成していきます。

Remix

今回は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します。Remix ではコードの作成からコンパイル、ブロックチェーンへのデプロイといった一連の流れの全てを行うことができます。

Remix 公式サイト：https://github.com/ethereum/remix

Remix を利用する方法は 2 つあり、1 つはローカルに環境を構築する方法で、もう 1 つはオンラインで利用する方法です。今回はオンライン版を使用し、コントラクトの作成から、作成したコントラクトの利用までの流れを説明します。ここからは演習形式で作業を行っていきます。

3.2 Ethereum の利用準備

3.2.1 Ethereum ネットワークの種類

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です。価値を持った Ether が取引されています。

テストネットワーク

メ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です。メインネットワークとは、**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**点が異なります。Bitcoin や法定通貨をはじめとして、メインネットワークの Ether とも交換することができません。

プライベートネットワーク

メインネットワークとテストネットワークは世界中のコンピュータで構成されているのに対し、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ることで、**誰でも作成することのできるネットワーク**です。一台のコンピュータで構成してもいいですし、複数のコンピュータを繋げてネットワークを作成することもできます。

今回作成するトレーサビリティシステムでは、ローカル環境に Ethereum のプライベートネットワークを作成して利用します。

3.2.2 ウォレット

Ethereum を利用するためには手数料が必要になり、この**手数料は Ethereum の内部通貨である ETH で支払う必要**があります。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ります。ここで言うウォレットは、私たちが普段利用している財布(ウォレット)とは違い、実際に通貨が入っているわけではなく、通貨を利用するための秘密鍵を管理しています。

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ります。これらを仮想通貨の利用方法や利用頻度によって使い分けることで、より仮想通貨を安全に管理することができます。今回利用するウォレットはブラウザの拡張機能として利用することのできる **MetaMask** というウォレットです。事前準備として 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 環境構築

ここからはホストマシン上に Ethereum のプライベートネットワークを作成します。ローカル環境に Ethereum のプライベートネットワークを作成する方法はいくつかありますが、今回は **Ganache** というツールを利用して環境を構築します。

Ganache

Ganache は、Ethereum のローカル開発環境です。ローカルで開発用のチェーンを構築するだけでなく、GUI でブロックやトランザクションを参照して、アプリケーションの動作を確認することができます。

利用方法

1. 公式サイトにアクセスします。(https://www.trufflesuite.com/ganache)

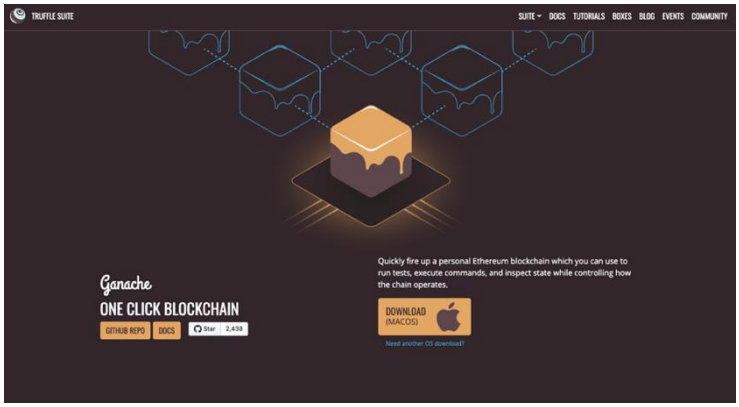

図 3.2 Ganache の公式サイト

2. 公式サイト画面内の DOWNLOAD ボタンをクリックする。
3. DOWNLOAD が終わると、 Ganache アプリを開きます。

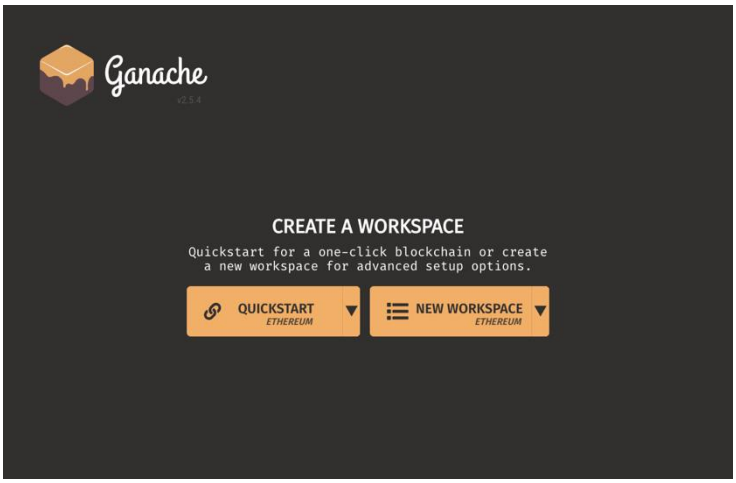

図 3.3 Ganache アプリのトップ画面

4. QUICKSTART クリックします。
5. 以下の画面が表示されれば、Ganache での環境構築は終了です

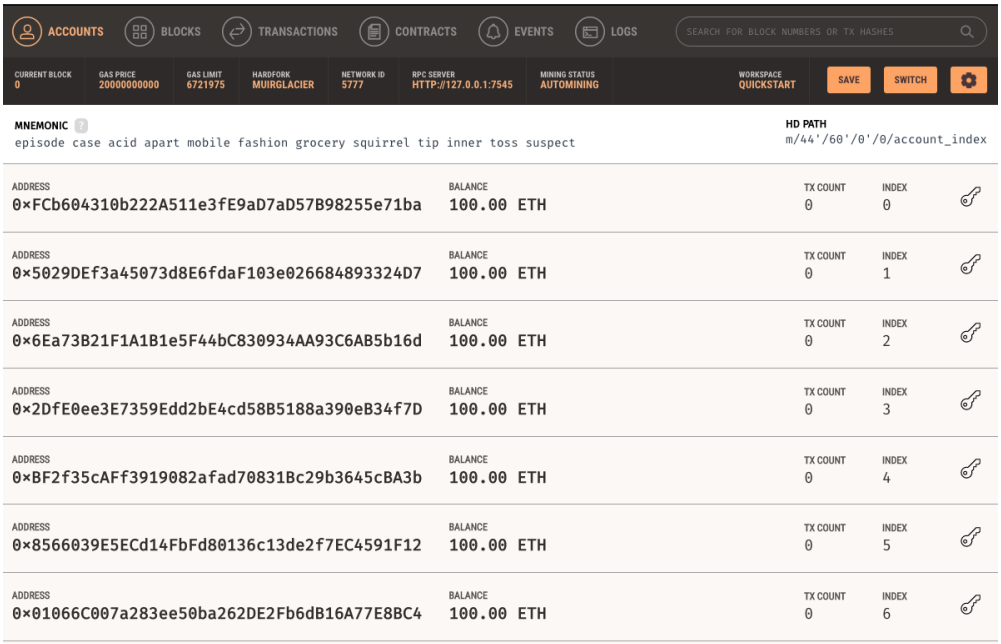

図 3.4 Ganache の設定完了

以上で、ローカル環境での Ethereum のプライベートネットワークの環境構築が終了しました。

MetaMask の設定

次に MetaMask の設定を変更し、作成したプライベートネットワークを見に行くようにします。まずは、Ganache 上から作成したプライベートネットワークの設定を確認します。Ganache のトップ画面の右上にある歯車のボタンをクリックし、設定画面を開きます。設定画面内に「SERVER」の設定項目がありますので、さらにそれを開きます。

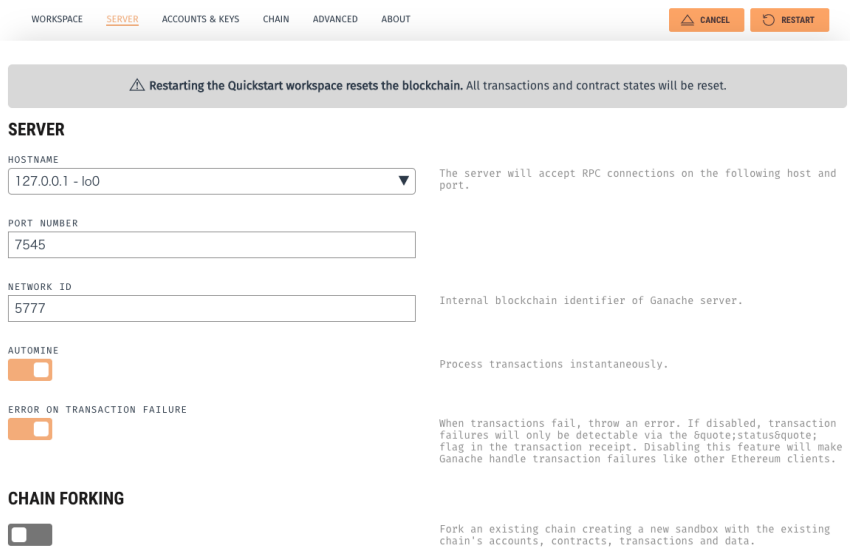

図 3.5 SERVER の設定画面

上記の画面の以下の情報を記録します。

HOSTNAME: 127.0.0.1

PORT NUMBER: 7545

NETWORK ID: 5777

次に MetaMask の設定に移ります。MetaMask のトップ画面を開き、右上のまるいボタンをクリックします。

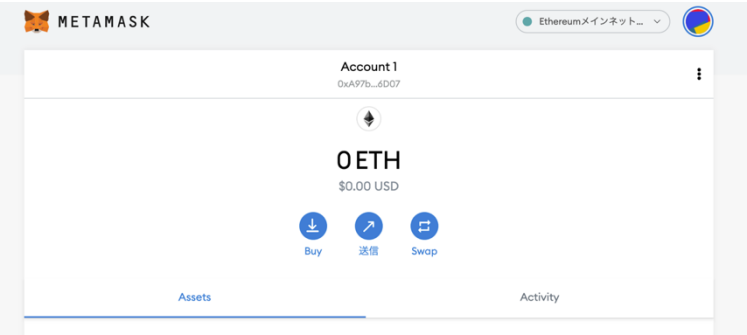

図 3.6 Metamask のトップ画面

設定ボタンをクリックします。

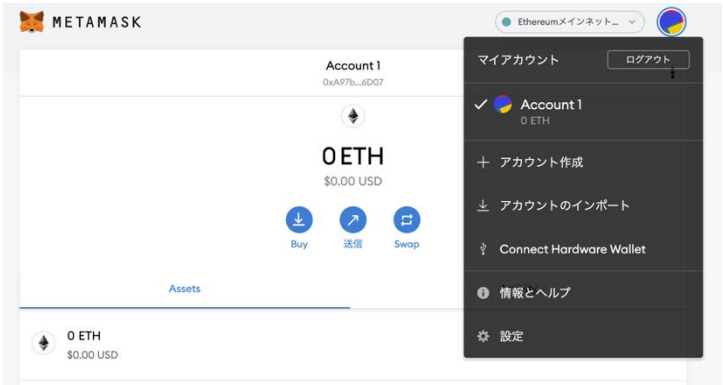

図 3.7 Metamask の設定

ネットワークの設定を開きます。

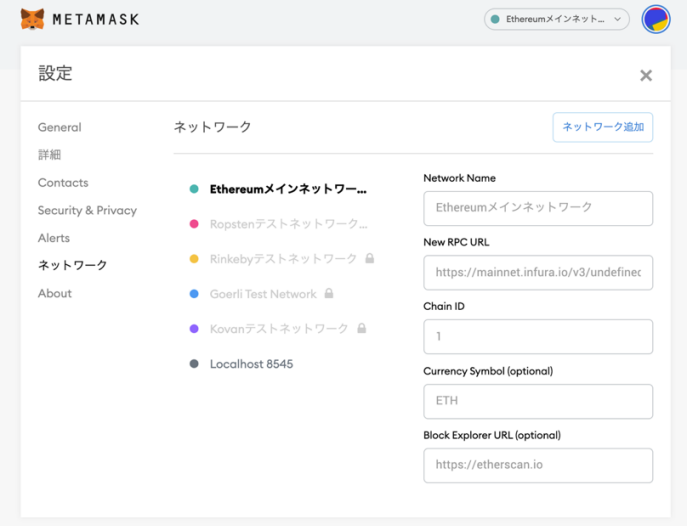

図 3.8 Metamask のネットワーク設定

ネットワーク設定画面で「ネットワーク追加ボタン」をクリックし、Ethereum のプライベートネットワークの情報を記入します。

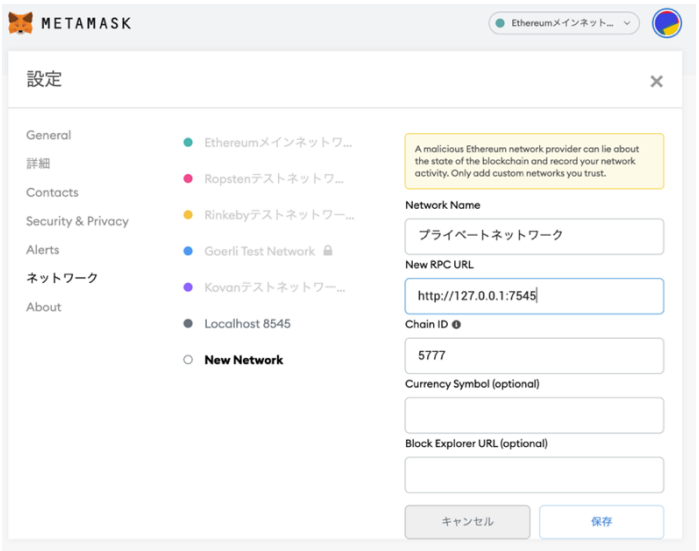

図 3.9 Metamask のネットワーク設定

保存ボタンを押します。以下のように ChainID が訂正される場合は、指示通りに訂正してください。

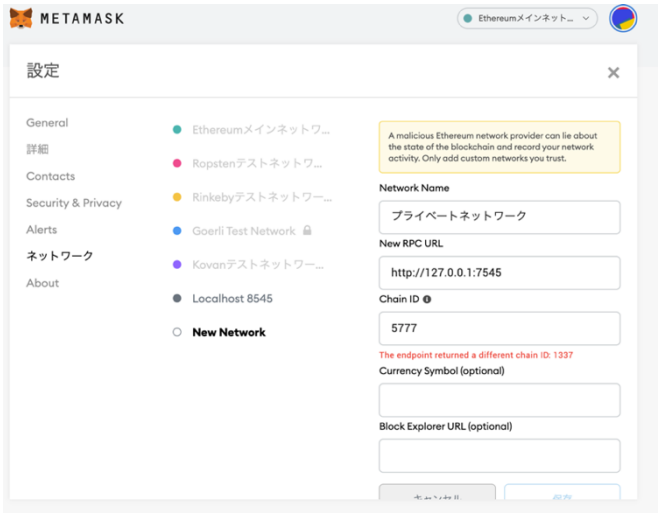

図 3.10 Metamask のネットワーク設定

以上が終わると、Metamask のトップ画面に戻り、画面右上のネットワーク選択部分から、作成したネットワークを選択します。

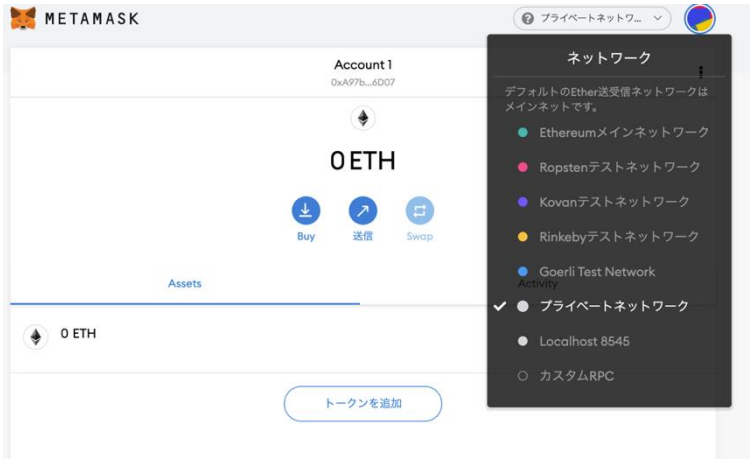

図 3.11 Metamask のネットワーク設定

次に、Ganache を起動した際に作成されている Ethereum アカウントを MetaMask 上から利用できるようにします。Ganache のトップ画面に戻り、それぞれのアカウントの最も右側にある鍵ボタンをクリックすると、下記のようなポップアップ画面が開きます。この画面の PRIVATE KEY をコピーします。

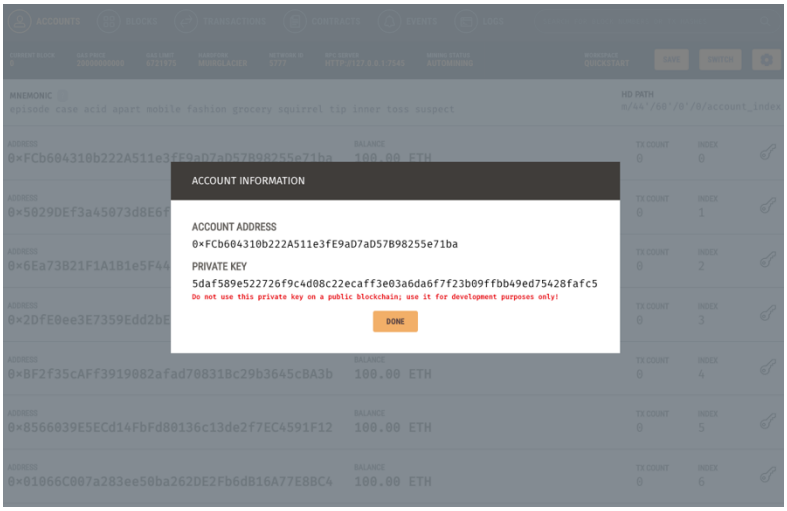

図 3.12 Metamask のネットワーク設定

MetaMask へ戻ります。画面右上の丸いアイコンをクリックし、「アカウントのインポート」を選択します。

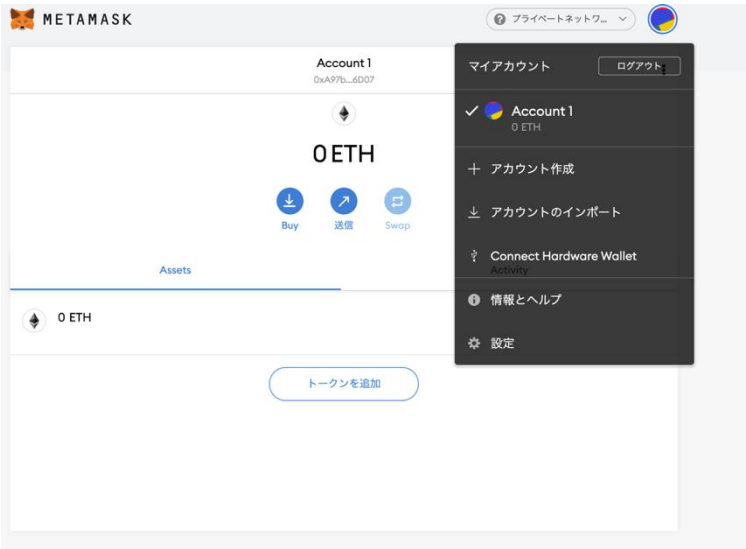

図 3.13 Metamask のネットワーク設定

先ほどコピーした秘密鍵を貼り付けます。

図 3.14 Metamask のネットワーク設定

トップ画面に戻り、以下のように通貨の保有額が 0ETH より多くなっていれば完了です。

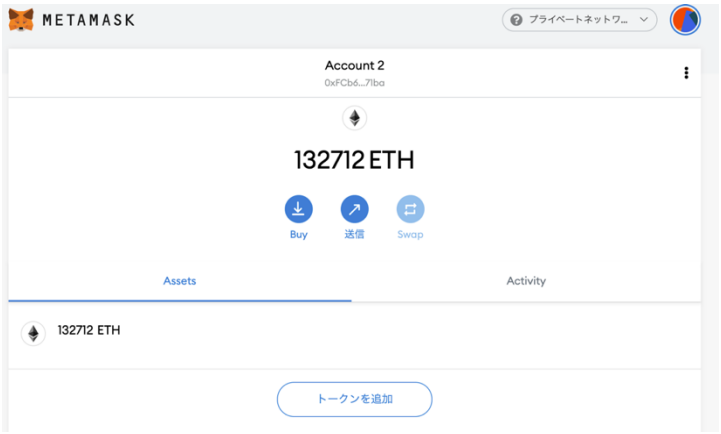

図 3.15 Metamask のネットワーク設定

以上で、ローカル環境でのスマートコントラクトの利用環境が整いました。次の章では、作成したトレーサビリティシステムに対して、ブロックチェーンを使った機能を追加していきます。

3.3 Ethereum の利用

3.3.1 流通履歴の Ethereum への登録

これまでは、流通履歴をデータベースのみに保存していましたが、ここでは流通履歴のバックアップをブロックチェーンに保存します。

ここからの作業は大きく 2 つに分けることができます。

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

Ethereum にコントラクトを登録する

今回利用するコントラクトを紹介します。

- 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

また、コントラクトが保有するデータについても紹介します。

- コントラクトを作成したオーナーのアドレス
- 流通データを保存する連想配列

以下が、実際に利用する Solidity で書かれたソースコードです。

```
pragma solidity ^0.6.12;


contract Trace {


    // 流通履歴の id と値を保存する連想配列
    mapping (string => string) traceHashes;

    //コントラクトの owner のアドレス
    address payable owner;

    // コンストラクタ(コントラクトが登録される際に一度だけ、実行される)
    constructor () public {

    // msg.sender はトランザクションの発行者のアドレスが入っている
        owner = msg.sender;
    }

    // owner のみ利用することのできる関数に利用する
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    // 流通履歴の登録
    function setTrace(string memory _id, string memory _traceDetail) public {
        traceHashes[_id] = _traceDetail;
    }

    // 流通履歴の検証
    function validateTrace(string memory _id) public view returns (string memory){
```

```
        return traceHashes[_id];
    }

    //  コントラクトのオーナーを消す
    function deleteOwner() public onlyOwner {
        owner = address(0x0);
    }

    // コントラクトを破棄する
    function kill() public onlyOwner {
        selfdestruct(owner);
    }
}
```

Solidity の関数について

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**です。このコントラクトでは、owner 変数にコントラクトをブロックチェーンに登録したアカウント、つまりコントラクトのオーナーのアドレスを記録します。

modifier

modifier は**関数修飾子**と呼ばれるものです。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します。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 関数に利用しています。

selfdestruct

コントラクトの最後の関数に kill 関数があります。この中で利用している selfdestruct 関数は solidity が元々持っている関数であり、**コントラクトを破棄**することができます。コントラクト自体に致命的なバグが合った際や、サービスを終了する際に利用します。

Solidity の変数には型が存在します。その中でも、address payable と言う特殊な型があります。これはその ethereum address にコントラクト内で ether の送金が発生する場合にこの型を利用します。今回は、送金は関連していないように思えますが、selfdestruct 関数では、コントラクトを破棄する際に、引数で指定されたアドレスにコントラクトが持っている通貨を送金する機能があります。このため、owner 変数は address payable 型となっています。

上記のコードを Remix でコンパイルし、Ganache で作成した Ethereum にデプロイしてください。その際のコントラクトのアドレスと ABI がコントラクトを呼び出す際に必要になります。また、コントラクトをデプロイする際には、Remix の **ENVIRONMENT の項目を Injected web3** に設定し、MetaMask を通じたデプロイを行うようにしてください。また、この状態で Metamask の設定をメインネットワークや、テストネットワークにすることで、コントラクトをデプロイするネットワークを変更することができます。

ブラウザから JavaScript を利用するためのコードの記述

今回、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です。

ここからは、 1 章で作成したコードにブロックチェーンを利用するための記述を加えていきます。resources/views/trace/content_detail.php 内の下部の script タグ内に以下のコードを記述してください。 abi 変数にはコントラクトの ABI を、 ContractAddress 変数には、コントラクトのアドレスを入れてください。

```
// コントラクトのインターフェイスの登録
let abi;

// コントラクトのアドレスの登録
let ContractAddress;
let ContractInstance;

// コントラクトをインスタンスとして、変数に持たせる
function initApp() {
    let myContract = web3.eth.contract(abi);
    ContractInstance = myContract.at(ContractAddress);
}

window.register = () => {
    let id = event.target.id;
    let value = [id, document.getElementById(`value_${id}`).value];
    ContractInstance.setTrace(id, document.getElementById(`value_${id}`).value, function (err, result) {
        if (!err) {
          console.log(result);
        } else {
```

```
            console.log(err);
        }
    });
};

window.validate = () => {
    let id = event.target.id;
    let message = '';
    ContractInstance.validateTrace(id, function (err, result) {
        if (!err) {
          if(result) {
              if(result == document.getElementById(`value_${id}`).value) {
                message = '正しい情報が登録されています';
              } else {
                message = '情報が一致しません。改ざんされています';
              }
          } else
          {
              message = 'ブロックチェーンに登録されていません。または、必要な情報が DB から削除された可能性があります';
          }
        } else {
            message = '通信失敗';
            console.log(err);
        }
        console.log(document.getElementById(`message_${id}`));
        document.getElementById(`message_${id}`).innerText = message;
    });
};

window.addEventListener('load', function () {

    if (typeof window.ethereum !== 'undefined' || typeof window.web3 !== "undefined") {

        let provider = window['ethereum'] || window.web3.currentProvider;
        web3 = new Web3(provider);
        ethereum.enable();

    } else {
        console.log("Metamask が認識されません");
```

```
    }

    initApp();

});
```

次に、 content_detail.php ファイル内の以下の部分を書き換えます。

```
@foreach($traces as $trace)
        <div class="content-history">

//ここから追記
        @if(Auth::user()->id == $trace->user->id)
          <button id='{{$trace->id}}' onclick='register();'>登録</button>
        @endif
        <div id='message_{{$trace->id}}'></div>
        <input id='value_{{$trace->id}}' type='hidden' value={{$trace->latitude . $trace->longitude . $trace->created_at}}></input>

//ここまで
```

ブロックチェーンには、データベースに記録されている**流通履歴の id を key** として、**流通履歴の情報である、緯度、経度、登録された時刻を文字列として結合したものを value** として保存しています。

3.3.2 Ethereum から流通履歴の参照

ここでは、ブロックチェーンに登録した流通履歴が削除、改ざんされていないかの検証機能を作成します。先ほどのコントラクトに検証機能は作成が終わっています。ここではデータベースに

記録されている情報と、ブロックチェーンに記録されている情報を比較し、改ざんが行われていないか検証します。content_detail.php の以下を変更してください。

```
 @foreach($traces as $trace)
        <div class="content-history">

 //ここから追記
        <button id='{{$trace->id}}' onclick='validate();'>検証</button>
 //ここまで

        @if(Auth::user()->id == $trace->user->id)
          <button id='{{$trace->id}}' onclick='register();'>登録</button>
        @endif
        <div id='message_{{$trace->id}}'></div>
        <input id='value_{{$trace->id}}' type='hidden' value={{$trace->latitude . $trace->longitude . $trace->created_at}}></input>
```

3.3.3 支払い機能

次に、仮想通貨での支払い機能を作成します。今回は、Ethereum アドレスを知っている任意の相手に報酬を支払うことができる仕組みとします。Web3.js の機能を利用すると仮想通貨の送金は非常に簡単に行うことができます。

以下のコードを参考に、送金機能を作成してください。必要に応じて content_detail.php ファイル内に input タグを作成する必要もあります。

```
function sendEther() {
    const sendButton = document.getElementById('Button');
    sendButton.addEventListener('click', function () {
        web3.eth.sendTransaction({
            from: web3.eth.accounts[0], // metamask 内の現在選択されているアドレス
            to: '0xee8d0f5b3ce42210ad154f4a45628db768b55012', //  input タグから送金先のアドレスを取得
```

```
            value: 10 // input タグから送金額を取得
        }, function (error, txHash) {
            if (!error) {
            } else {
            }
        })
    });
}
```

また、今回のコントラクトには、「**コントラクトの管理者を消す機能**」「**コントラクトを破棄する機能**」があります。今回のシステムの管理者機能として、ブラウザから利用にできるようにしてください。

4.　スマートコントラクトを用いたシステムの実例

4.1 仮想通貨の種類

4.1.1 ブロックチェーンと仮想通貨

2009 年に運用が始まった Bitcoin や、今回開発に利用した Ethereum などの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ます。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**が支払われます。このように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**と呼ばれます。Bitcoin というシステムの中の Bitcoin という通貨や、Ethereum というシステムの中の Ether という通貨がこれに該当します。

4.1.2 仮想通貨の種類

内部通貨以外にも世の中に流通している仮想通貨はあります。それらの通貨は、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**です。これは、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ます。仮想通貨を発行すると言うと、極めてハードルが高いことのように思えます。実際に、書き換えの効かないブロックチェーンの上で通貨を発行し、安全面などに問題があると、通貨としても信用を失う可能性もあり細心の

注意を払う必要もあり高い技術力が必要となります。このような理由から完全にオリジナルの仮想通貨をブロックチェーン上に発行することは避けた方がいいように思えます。そのため、Ethereum には**独自の通貨を安全に発行するためのルール**がいくつか存在しています。このルールに則っとることで安全な通貨を Ethereum 上で発行することができます。

ERC

ERC とは **Ethereum Request for Comments** の略で、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のことを指します。「ERC」の後の番号は Github で提案された順番を示しています。ERC-20 なら 20 番目、ERC-721 なら 721 番目に提案されたものであるということになります。

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ます。ERC-20 を利用して作成されたトークンを利用して作られているサービスはたくさんあります。

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です。代替不可能トークンについて説明するために、まず代替可能なトークンについて説明します。代替可能なトークンの最もわかりやすい例は、私たちが普段利用している通貨です。手元にある 1 万円札と、今誰かが持っている一万円札の価値は全く同じで交換可能です。つまり、この二つの一万円札は同一のものとして認識することができます。これに対して、代替不可能トークンとは、**量としては同じであるが同一のものとして認識することができない**ものが該当します。例えば、あなたが犬を飼っ

ているとします。また、あなたの友人も犬を飼っているとします。これらは、同じ犬一匹ですが、異なるものであり交換可能ではありません。犬だけでなく、あなたが持っているスマートフォンや、服、ペンなどあらゆるものが代替不可能です。このようなものをトークン化しているものが代替不可能トークンです。

今回の演習では、Ethereum とその内部通貨の Ether を利用しましたが、今回紹介したようなオリジナルの仮想通貨を利用して、Web アプリケーションを作成することもできます。

4.2 スマートコントラクトを用いたシステム

4.2.1 ゲーム

ブロックチェーンが利用された Web アプリケーション形式のゲームを紹介します。

Cryptokitties

https://www.cryptokitties.co/

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です。それぞれの猫は代替不可能なトークンである **ERC-721** となっています。具体的には、それぞれの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ています。また、それらの取引に関連する処理と、取引情報の記録は Ethereum 上で行われます。つまり、このゲームの全てがブロックチェーン上で完結しているわけではなく、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おらず、管理者は存在します。また、このサービスを利用して猫の取引を行うためにはユーザーが MetaMask などのウォレットを準備し、自身で Ether を購入する必要があります。

ブロックチェーンを利用したゲームは、これ以外にもたくさんあります。それらの多くが Cryptokitties のように、ゲーム内でキャラクターを育成しそれを仮想通貨により売買することができるものです。そこで得た仮想通貨は、仮想通貨取引所などで日本円などの法定通貨に変えることもできます。これらのゲームが流行っている最大の理由は、キャラクターの売買により利益をあげることができるためです。

4.2.2 分散型取引所

一般的に仮想通貨取引所は企業により運営されています。これに対して、ブロックチェーンを用いて**非中央集権的に運営される仮想通貨取引所**を分散型取引所と言います。

非中央集権的な仕組みのブロックチェーンに対して、仮想通貨を取引する中央集権的な取引所に大きなリスクがあることは、とても好ましい状態であるとは言えません。このような仕組みに対して、分散型取引所ではブロックチェーンを基盤とすることで、管理者が存在しない状態で取引を行うことができます。

Ether Delta

https://Etherdelta.com/

EtherDelta は Ethereum 上に構築される分散型取引所です。Ethereum 上にあるコントラクトの利用率で常に上位を保っていることからも、最も活発な取引所であると言うことできます。EtherDelta を利用するために必要になる手数料には「Ethereum ネットワークに対する手数料」と「プラットフォームに対する手数料」の 2 つがあり、EtherDelta は「プラットフォームに対する手数料」によりマネタイズを行っています。また、 EtherDelta は Ethereum 上に構築される取引所であるため、Ethereum の上に存在しない通貨の取引は行うことができません。具体的には、Ethereum の内部通貨である Ether や、Ethereum 上で作成される ERC などが対象となり、Bitcoin などのその他の仮想通貨や、法定通貨などは取引を行うことができません。

4.2.3 権利の管理

音楽情報の管理

日本音楽著作権協会（JASRAC）はブロックチェーン技術による**音楽作品情報の存在証明**に関する実証実験を行いました。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します。この実証実験の背景には、デジタルコンテンツで使われている音楽著作権の特定や管理が複雑化していることや、音楽著作物の演奏利用では、人手を用いた既存の著作権管理だけでは十分に管理できないことがあります。また、ブロックチェーンも活用することで、権利者に還元できる透明性の高い仕組みを構築することもできます。

解答

MySQL 演習

1. Sample という名前のデータベースを作成してください

```
$ CREATE database sample;
```

2. student というユーザーを作成してください

```
$ CREATE USER 'student'@'localhost' IDENTIFIED BY 'your_password';
```

3. 作成したユーザーに全てのデータベースに対する全ての命令の権限を与えてください

```
$ GRANT ALL ON *.* TO student@localhost;

$ FLUSH PRIVILEGES;
```

4. Users テーブルを作成してください

```
$ USE sample;

$ CREATE TABLE users(id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,name VARCHAR(50));
```

5. Users テーブルにレコードを追加してください

```
$ INSERT INTO users (name) VALUES ('tanaka');
```

6. 既存のカラムのフィールドを書き換えてください

```
$ UPDATE users SET name='yamada' WHERE id=1;
```

7. 先頭のレコードを削除してください

```
$ delete from users where id=1;
```

8. 作成したデータベースを削除してください

```
$ drop database sample;
```

Migration ファイル

Contents テーブル

```
public function up()
{
    Schema::create('contents', function (Blueprint $table) {
        $table->id();
        $table->string('brand');
        $table->string('name')->unique();
        $table->string('price');
        $table->string('information');
        $table->bigInteger('user_id')->unsigned();
        $table->string('identifier',10)->unique();
        $table->string('image_path');
        $table->timestamps();

        $table->foreign('user_id')->references('id')->on('users');
    });
}

 public function down()
{
    Schema::dropIfExists('contents');
}
```

Traces テーブル

```
 public function up()
{
    Schema::create('traces', function (Blueprint $table) {
        $table->id();
        $table->bigInteger('content_id')->unsigned();
        $table->bigInteger('user_id')->unsigned();
        $table->text('comment');
        $table->string('latitude');
```

```
            $table->string('longitude');
            $table->timestamps();

            $table->foreign('user_id')->references('id')->on('users');
            $table->foreign('content_id')->references('id')->on('contents')->onDelete('cascade');
        });
    }

    public function down()
    {
        Schema::dropIfExists('traces');
    }
```

Users テーブル

```
public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string("role");
        });
    }
```

View ファイル

content/add_content.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>製品登録</title>
    <link rel="stylesheet" href="./css/reset.css">
    <!-- UIkit CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css"/>
    <!-- UIkit JS -->
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/qrcode-generator/1.4.4/qrcode.min.js"></script>
    <link rel="stylesheet" href="/css/style.css">
    <link rel="stylesheet" href="/css/modal.css">
</head>
<body>
<nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
        <ul class="uk-navbar-nav">
            @can('isAdmin')
                <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
            @else
                <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
```

```
            @endcan
        </ul>
    </div>
    <div class="uk-navbar-right">
        <ul class="uk-navbar-nav">
            <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
            <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
        </ul>
    </div>
</nav>

<form class="h-adr" action="{{ route('content_store') }}" method="post" enctype="multipart/form-data">
    @csrf
    <div class="main-area">
        <div class="uk-card-default uk-card-body">
            <div class="uk-card-title uk-margin uk-margin-auto">
                @if(Session::has('success'))
                    {{ session('success') }}
                @endif
                <h3>製品登録</h3>
            </div>
            <div
                class="add-content-info uk-child-width-1-2@m uk-child-width-1-1@s uk-margin uk-margin-auto
uk-grid-collapse"
                uk-grid>
                <div>
                    <div class="uk-padding-small">
                        <div class="uk-form-stacked">
                            <div class="content-brand">
                                <label class="uk-form-label" for="form-stacked-text">ブランド名</label>
                                @if($errors->has('brand'))
                                    <div class="error-msg"> ! {{ $errors->first('brand') }}</div>
                                @endif
```

```
                    <div class="uk-form-controls">
                        <input name="brand" class="uk-input" type="text" required placeholder="ブランド名を入力...">
                    </div>
                </div>
                <div class="content-name">
                    <label class="uk-form-label" for="form-stacked-text">製品名</label>
                    @if($errors->has('name'))
                        <div class="error-msg"> ! {{ $errors->first('name') }}</div>
                    @endif
                    <div class="uk-form-controls">
                        <input name="name" class="uk-input" type="text" required placeholder="製品名を入力...">
                    </div>
                </div>
                <div class="content-price">
                    <label class="uk-form-label" for="form-stacked-text">出荷価格(ETH)</label>
                    @if($errors->has('price'))
                        <div class="error-msg"> ! {{ $errors->first('price') }}</div>
                    @endif
                    <div class="uk-form-controls">
                        <input name="price" class="uk-input" type="text" required placeholder="出荷価格を入力...">
                    </div>
                </div>
                <div class="content-info">
                    <label class="uk-form-label" for="form-stacked-text">製品情報</label>
                    @if($errors->has('information'))
                        <div class="error-msg"> ! {{ $errors->first('information') }}</div>
                    @endif
                    <div class="uk-form-controls">
                        <textarea name="information" class="uk-input" type="textarea" required
                                placeholder="製品情報を入力..."></textarea>
```

```
                        </div>
                    </div>
                </div>
            </div>
            <div>
                <div class="uk-padding-small">
                    <div class="content-image-area">
                        <div class="content-image uk-margin-auto uk-background-muted">
                            <div class="preview"></div>
                        </div>
                    </div>
                    <div class="uk-margin uk-text-center">
                        <div uk-form-custom>
                            <input type="file" name="imagefile" required>
                            <button class="uk-button uk-button-secondary" type="button" tabindex="-1">画像を選択</button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="uk-card-footer uk-margin uk-margin-auto uk-text-center">
            <div id="open">
                <button type="submit" class="uk-button uk-button-primary">登録</button>
            </div>
        </div>
    </div>
</div>
</form>
</body>
<script>
    document.addEventListener('DOMContentLoaded', function () {
```

```
        document.querySelector("[name='imagefile']").addEventListener('change', function (e) {
            let file = e.target.files[0],
                reader = new FileReader(),
                preview = document.querySelector(".preview"),
                t = this;

            reader.onload = (function (file) {
                return function (e) {
                    while (preview.firstChild) preview.removeChild(preview.firstChild);

                    let img = document.createElement("img");
                    img.setAttribute('src', e.target.result);
                    img.setAttribute('title', file.name);
                    preview.appendChild(img);
                };
            })(file);

            reader.readAsDataURL(file);
        });
    });

</script>
</html>
```

content/add_content_success.blade.php

```
<body>
<div class="main-area">
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
        <div class="uk-card-title uk-margin uk-margin-auto">
            <h3>商品登録情報</h3>
        </div>
```

```
        <div class="uk-text-center uk-h4">
            <div class="uk-inline">
                <span class="uk-text-success" uk-icon="icon: check"></span>
                <span class="uk-text-middle uk-text-success">商品登録完了しました</span>
            </div>
        </div>
        <div class="identifier-qr">
            <span class="identifier">製品コード：{{$content->identifier}}</span>
            <div id="qr_maker">
                <input class="url-input" type="hidden" id="url"
                    value="http://18.220.72.4/trace/content_detail?search={{$content->identifier}}">
                <div id="qrcode"></div>
                <button class="uk-button uk-button-default printing" type="button" onclick="funcprint();">印刷す
る</button>
            </div>
        </div>
        <div class="uk-card-footer uk-margin uk-text-center">
            <button class="uk-button uk-button-primary return" type="button"
                onclick="location.href='{{ route('index') }}'">戻る
            </button>
        </div>
    </div>
 </div>
 </body>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/qrcode-generator/1.4.4/qrcode.min.js"></script>
 <script>
    let qr = qrcode(4, 'L');
    qr.addData(document.getElementById('url').value);
    qr.make();
    document.getElementById('qrcode').innerHTML = qr.createImgTag();
 </script>
```

trace/comment_detail.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>コメント更新</title>
    <link rel="stylesheet" href="./css/reset.css">
    <!-- UIkit CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css"/>
    <!-- UIkit JS -->
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
    <link rel="stylesheet" href="./css/style.css">

</head>
<body>
<nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
        <ul class="uk-navbar-nav">
            @can('isAdmin')
                <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
            @else
                <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
            @endcan
        </ul>
    </div>
    <div class="uk-navbar-right">
        <ul class="uk-navbar-nav">
```

```
            <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
            <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
        </ul>
    </div>
</nav>
<div class="main-area">
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
        <div class="uk-card-title uk-margin uk-margin-auto">
            <h3>コメント（詳細）</h3>
        </div>
        <div class="uk-margin-auto">
            <form class="uk-grid-small" uk-grid action="" method="post" enctype="multipart/form-data">
                @csrf
                <div class=" uk-margin-small-bottom">
                    <div class="uk-grid-small" uk-grid>
                        <div>
                            <div class="uk-margin-small-bottom">
                                <div class="uk-inline">
                                    <span class="uk-form-icon" uk-icon="icon: calendar"></span>
                                    <input class="uk-input uk-form-blank uk-disabled" type="text"
                                        placeholder={{ $dateTime->format('Y/m/d-H:s') }}>
                                </div>
                            </div>
                            <div class="uk-margin-small-bottom uk-margin-remove-top">
                                <div class="uk-inline">
                                    <span class="uk-form-icon" uk-icon="icon: user"></span>
                                    <input class="uk-input uk-form-blank uk-disabled" type="text"
                                        placeholder={{ $auth->name }}>
                                </div>
                            </div>
                        </div>
                        <div>
                            <div class="uk-margin-small-bottom uk-margin-remove-top uk-width-1-1">
```

```
                        <div class="uk-inline">
                            <span class="uk-form-icon" uk-icon="icon: location"></span>
                            <input class="uk-input uk-form-blank" type="text" placeholder="場所（地図から選
択）">
                        </div>
                        <div class="select-map-area">
                            <div class="select-map">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            <div class="uk-width-2-3@m uk-width-1-1@s uk-margin-small-bottom">
                <fieldset class="uk-fieldset">
                    @if($errors->has('comment'))
                        <div class="error-msg"> ! {{ $errors->first('comment') }}</div>
                    @endif
                    <textarea class="uk-textarea uk-padding-small" type="textarea" name="comment" rows="5"
                            placeholder="コメントを入力"></textarea>
                </fieldset>
            </div>
            <div class="comment-btn-area uk-margin-auto uk-margin-auto-top uk-margin-small-bottom">
                <button class="uk-button uk-button-primary" type="button">送信</button>
            </div>
        </form>
    </div>
  </div>
 </div>
 </body>
 </html>
```

trace/content_detail.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>商品詳細</title>
    <link rel="stylesheet" href="./css/reset.css">
    <!-- UIkit CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css"/>
    <!-- UIkit JS -->
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/qrcode-generator/1.4.4/qrcode.min.js"></script>
    <link rel="stylesheet" href="/css/style.css">

</head>
<body>
<nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
        <ul class="uk-navbar-nav">
            @can('isAdmin')
                <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
            @else
                <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム
</a></li>
            @endcan
        </ul>
    </div>
    <div class="uk-navbar-right">
```

```
        <ul class="uk-navbar-nav">
            @auth
                <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
                <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
            @endauth
            @guest
                <li><a href="{{ route('login') }}">ログイン</a></li>
            @endguest
        </ul>
    </div>
 </nav>
 <div class="main-area">
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
        <div class="uk-card-title uk-margin uk-margin-auto">
            <h3>製品詳細</h3>
        </div>
        <div id="qr_maker">
            <input type="hidden" style="width: 50%;" id="url">
            <div id="qrcode"></div>
        </div>
        <div
            class="content-detail-info uk-child-width-1-2@m uk-child-width-1-1@s uk-margin uk-margin-auto uk-
grid-collapse"
            uk-grid>
            <div>
                <div class="content-name uk-padding-small">
                    <div class="content-brand">
                        <label class="uk-form-label">ブランド名</label>
                        <div class="uk-form-controls">
                            <h4>{{ $content->brand }}</h4>
                        </div>
                    </div>
                    <div class="content-name">
```

```
                    <label class="uk-form-label">製品名</label>
                    <div class="uk-form-controls">
                        <h4>{{ $content->name }}</h4>
                    </div>
                </div>
                <div class="content-name">
                    <label class="uk-form-label">出荷価格(ETH)</label>
                    <div class="uk-form-controls">
                        <h4>{{ $content->price }}ETH</h4>
                    </div>
                </div>
                <div class="uk-section-muted uk-section-small">
                    <h4>{{ $content->information }}</h4>
                </div>
            </div>
        </div>
        <div>
            <div class="uk-padding-small">
                <div class="content-image-area">
                    <div class="content-image uk-margin-auto uk-background-muted">
                        <img src="{{ asset($content->image_path) }}" alt="製品画像">
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
@auth
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
        <div class="uk-card-title">
            <h3>流通履歴</h3>
        </div>
        @foreach($traces as $trace)
```

```
<div class="content-history">
  <table class="uk-table uk-table-responsive uk-table-divider uk-table-justify">
    <thead>
    <tr>
      <th class="uk-width-small">日時</th>
      <th class="uk-width-small">場所</th>
      <th class="uk-width-small">ユーザー名</th>
      <th class="uk-table-expand">コメント</th>
    </tr>
    </thead>
    <tbody>
    <tr>
      <td>
        <div class="uk-visible@m">
          {{$trace->created_at}}
        </div>
        <div class="uk-inline uk-hidden@m">
          <span class="uk-form-icon" uk-icon="icon: calendar"></span>
          <p class="uk-input uk-form-blank uk-disabled">{{$trace->created_at}}</p>
        </div>
      </td>
      <td>
        <div class="uk-visible@m">
          緯度：{{$trace->latitude}}<br>
          経度：{{$trace->longitude}}
        </div>
        <div class="uk-inline uk-hidden@m">
          <span class="uk-form-icon" uk-icon="icon: location"></span>
          <p class="uk-input uk-form-blank uk-disabled">緯度{{$trace->latitude}}
            経度{{$trace->longitude}}</p>
        </div>
      </td>
      <td>
```

```
                        <div class="uk-visible@m">
                            {{$user->name}}
                        </div>
                        <div class="uk-inline uk-hidden@m">
                            <span class="uk-form-icon" uk-icon="icon: user"></span>
                            <p class="uk-input uk-form-blank uk-disabled">{{$user->name}}</p>
                        </div>
                    </td>
                    <td>
                        <div class="uk-visible@m">
                            {{$trace->comment}}
                        </div>
                        <div class="uk-inline uk-hidden@m">
                            <span class="uk-form-icon" uk-icon="icon: comment"></span>
                            <!--指定の文字数以上は「～～...」な風に隠せるといいな-->
                            <p class="uk-input uk-form-blank uk-disabled">{{$trace->comment}}</p>
                        </div>
                        <iframe frameborder="0" style="border:0; width: 100%; height: 100%"
                                src="//www.google.com/maps/embed/v1/place?key=AIzaSyCHHmpm5hgNJI-
K9s5Q92uvPrtWPpwJN3E&zoom=12&q={{$trace->latitude}},{{$trace->longitude}}"></iframe>
                    </td>
                </tr>
                </tbody>
            </table>
        </div>
    @endforeach
</div>
@cannot('isAdmin')
    <div class="add-comment uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
        <div class="uk-card-title">
            <h3>コメントを追加</h3>
        </div>
        <div class="uk-margin-auto">
```

```
                <form class="uk-grid-small" uk-grid action="{{ route('content_detail_store') }}"
method="post"
                      enctype="multipart/form-data">
                    @csrf
                    <div class="uk-width-2-3@m uk-width-1-1@s uk-margin-small-bottom">
                        <div class="uk-grid-small" uk-grid>
                            <div class="uk-margin-small-bottom">
                                <div class="uk-inline">
                                    <span class="uk-form-icon" uk-icon="icon: calendar"></span>
                                    <input class="uk-input uk-form-blank uk-disabled" type="text"
                                           placeholder={{ $dateTime->format('Y/m/d-H:s') }}>
                                </div>
                            </div>
                            <div class="uk-margin-small-bottom uk-margin-remove-top">
                                <div class="uk-inline">
                                    <span class="uk-form-icon" uk-icon="icon: location"></span>
                                    <input class="uk-input uk-form-blank uk-disabled" type="text"
                                           placeholder="場所（手動)">
                                    <input name='latitude' type="text" required placeholder="緯度">
                                    @if($errors->has('latitude'))
                                        <div class="error-msg"> ! {{ $errors->first('latitude') }}</div>
                                    @endif
                                    <input type="text" name="longitude" required placeholder="経度">
                                    @if($errors->has('longitude'))
                                        <div class="error-msg"> ! {{ $errors->first('longitude') }}</div>
                                    @endif
                                </div>
                            </div>
                            <div class="uk-margin-small-bottom uk-margin-remove-top">
                                <div class="uk-inline">
                                    <span class="uk-form-icon" uk-icon="icon: user"></span>
                                    <input class="uk-input uk-form-blank uk-disabled" type="text"
                                           placeholder={{ $user->name }}>
```

```
                        </div>
                    </div>
                </div>
                <fieldset class="uk-fieldset">
                    <input type="hidden" name="content_id" value="{{ $content->id }}">
                    <input type="hidden" name="user_id" value="{{ $user->id }}">
                    @if($errors->has('comment'))
                        <div class="error-msg"> ! {{ $errors->first('comment') }}</div>
                    @endif
                    <textarea class="uk-textarea uk-padding-small" type="textarea" name='comment'
rows="5"
                              required placeholder="コメントを入力"></textarea>
                </fieldset>
            </div>
            <div class="comment-btn-area uk-margin-auto uk-margin-auto-top uk-margin-small-
bottom">
                <button class="uk-button uk-button-primary" type="submit">送信</button>
            </div>
        </form>
    </div>
  </div>
@endcan
@endauth
</div>
</body>
<script>
    let url = location.href;
    document.getElementById('url').innerHTML = url;
    document.getElementById('url').value = url;

    let qr = qrcode(4, 'L');
    qr.addData(document.getElementById('url').value);
    qr.make();
```

```
    document.getElementById('qrcode').innerHTML = qr.createImgTag();

</script>
</html>
```

users/edit.blade.php

```
<!doctype html>
<html>
<head>
</head>
<body>
<div>
    <h1>
        ユーザー編集
    </h1>
    <a href={{route('users.index')}}>ユーザー一覧へ</a>
    <div>
        <form action='{{route('users.update')}}' method='post'>
            @csrf
            <label>名前</label>
            <input name="name" value={{$user->name}}><br>

            <label>メールアドレス</label>
            <input name="email" value={{$user->email}}><br>
            <button>送信</button>
        </form>
    </div>
</div>
</body>
</html>
```

users/edit.blade.php

```
<!doctype html>
<html>
<head>
</head>
<header>
    <form method="POST" action={{route('logout')}}>
        @csrf
        <button>logout</button>
    </form>
</header>
<body>
<div>
    <h1>
        ユーザー一覧
    </h1>
    @foreach($users as $user)
        <ul>
            <li>{{$user->name}}</li>
            <li>{{$user->email}}</li>
            <a href={{route('users.show', $user)}}>詳細</a>
            <a href={{route('users.edit', $user)}}>編集</a>
        </ul>
    @endforeach
</div>
<a href={{route('items.create')}}>商品新規登録</a>
<a href={{route('items.index')}}>商品一覧</a>
</body>
</html>
```

users/show.blade.php

```
<!doctype html>
<html>
<head>
```

```
</head>
<body>
<div>
   <h1>
      ユーザー詳細
   </h1>
   <a href={{route('users.index')}}>ユーザー一覧へ</a>
   <div>
      <ul>
         <li>
            {{$user->name}}
         </li>
         <li>
            {{$user->email}}
         </li>
      </ul>
   </div>
</div>
</body>
</html>
```

index.blade.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>トップページ</title>
  <link rel="stylesheet" href="./css/reset.css">
  <!-- UIkit CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
```

```
 <link rel="stylesheet" href="/css/style.css">
 <link rel="stylesheet" href="/css/modal.css">
 </head>
 <body>
  <!--ナビゲーション-->
  @auth
   <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
  @endauth
  @guest
   <nav class="guest-uk-navbar-container uk-navbar-transparent" uk-navbar>
  @endguest
    <div class="uk-navbar-left">
     <ul class="uk-navbar-nav">
      {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
       @can('isAdmin')
        <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @else
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
       @endcan
     </ul>
    </div>
    <div class="uk-navbar-right">
      <ul class="uk-navbar-nav">
        @auth
          <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
          <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
        @endauth
        @guest
          <li><a href="{{ route('login') }}">ログイン</a></li>
        @endguest
      </ul>
    </div>
  </nav>

  <!--main content-->
  <div class="main-area" id="main-area">
   @cannot('isAdmin')
    @auth
```

```
        <div class="add-comment-detail uk-child-width-1-2@m uk-child-width-1-1@s uk-text-center uk-grid-match" uk-grid>
          <div>
            <div class="uk-card-default uk-card-body">
              <p>新しい製品を登録する</p>
              <a href="{{ route('content.add_content') }}" title="製品を登録する" class="uk-button uk-button uk-button-primary">製品登録</a>
            </div>
          </div>
          <div>
            <div class="uk-card-default uk-card-body">
              <p>流通経路確認</p>
              <form class="uk-search uk-search-default uk-width-medium" method="get" action="{{ route('content_detail') }}">
                <button type="submit" class="uk-search-icon-flip uk-background-primary" uk-search-icon></button>
                <input class="uk-search-input" type="search" name="search" placeholder="製品コードを入力...">
              </form>
                <li class="error">{{ session('identifier_error') }}</li>
            </div>
          </div>
        </div>
      @endauth
      @guest
        <div>
          <div class="uk-card-default uk-card-body guest-main-area">
            <p>製品を見る</p>
            <form class="uk-search uk-search-default uk-width-medium" method="get" action="{{ route('content_detail') }}">
              <button type="submit" class="uk-search-icon-flip uk-background-primary" uk-search-icon></button>
              <input class="uk-search-input" type="search" name="search" required placeholder="製品コードを入力...">
            </form>
              <li class="error">{{ session('identifier_error') }}</li>
              @foreach ($errors->all() as $error)
                 <li>{{ $error }}</li>
              @endforeach
          </div>
        </div>
      @endguest
    @endcan
```

```
    @can('isAdmin')
      <p class="not-admin">管理者は利用出来ません</p>
    @endcan
  </div>
  @if (isset($content))
    <div class="popup" id="js-popup">
      <div class="popup-inner">
        <div class="close-btn" id="js-close-btn"><i class="fas fa-times"></i></div>
          @include('content.add_content_success', ['content' => $content])
      </div>
      <div class="black-background" id="js-black-bg"></div>
    </div>
  @endif
</body>
<script>
  window.onload = function() {
  var popup = document.getElementById('js-popup');
  if(!popup) return;
  popup.classList.add('is-show');

  var blackBg = document.getElementById('js-black-bg');
  var closeBtn = document.getElementById('js-close-btn');

  closePopUp(blackBg);
  closePopUp(closeBtn);

  function closePopUp(elem) {
    if(!elem) return;
    elem.addEventListener('click', function() {
      popup.classList.remove('is-show');
    })
  }
}
</script>
<script>
  function funcprint(){

    var mainArea = document.getElementById('main-area');
```

```
    mainArea.style.visibility = "hidden";



    mainArea.style.visibility = "visible";
  };
</script>
</html>
```

Admin/content_list.blade.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>contents list</title>
  <link rel="stylesheet" href="./css/reset.css">
  <link rel="stylesheet" href="/css/contents_list.css">
  <!-- UIkit CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <!--ナビゲーション-->
  <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
      <ul class="uk-navbar-nav">
        {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
        @can('isAdmin')
        <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @else
```

```
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @endcan
      </ul>
    </div>
    <div class="uk-navbar-right">
      <ul class="uk-navbar-nav">
        <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
        <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
      </ul>
    </div>
  </nav>
  <!--main content-->
  <div class="main-area">
   <div class="uk-card-default uk-card-body">
    <div class="uk-card-title">
      <h4>製品 一覧</h4>
    </div>
    @foreach($contents as $content)
     <div class="contents-list">
      <div class="id-img">
       <img src="{{ asset($content->image_path) }}" alt="製品画像"></td>
      </div>
      <div class="content">
          <h4>製品名<span class="mar-name"> | </span>{{$content->name}}</h4>
       <ul class="content-ul">
          <li>ID<span class="mar-03"> | </span>{{$content->id}}</li>
          <li>ブランド名 | {{$content->brand}}</li>
          <li>製品コード | {{$content->identifier}}</li>
          <li>登録日<span class="mar-01"> | </span>{{$content->created_at}}</li>
          <li class="a-button"><a href="/admin/contents_list/{{ $content->id }}" class="uk-button uk-button-
small uk-button-primary">詳細</a></li>
        </ul>
      </div>
     </div>
    @endforeach
   </div>
  </div>
 </body>
 </html>
```

Admin/index.blade.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>トップページ</title>
  <link rel="stylesheet" href="./css/reset.css">
  <!-- UIkit CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
<link rel="stylesheet" href="./css/style.css">
</head>
<body>
  <!--ナビゲーション-->
  <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
      <div class="uk-navbar-left">
        <ul class="uk-navbar-nav">
          {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
          <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        </ul>
      </div>
      <div class="uk-navbar-right">
          <ul class="uk-navbar-nav">
              <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
              <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
          </ul>
      </div>
  </nav>

  <!--main content-->
```

```
  <div class="main-area">
    <div class="add-comment-detail uk-child-width-1-2@m uk-child-width-1-1@s uk-text-center uk-grid-match" uk-grid>
      <div>
        <div class="uk-card-default uk-card-body">
          <p>ユーザーの一覧を見る</p>
          <a href="{{ route('users_list') }}" title="製品を登録する" class="uk-button uk-button uk-button-primary">一覧表</a>
        </div>
      </div>
      <div>
        <div class="uk-card-default uk-card-body">
          <p>製品の一覧を見る</p>
          <a href="{{ route('contents_list') }}" title="製品を登録する" class="uk-button uk-button uk-button-primary">一覧表</a>
        </div>
      </div>
    </div>
  </div>
 </body>
 </html>
```

Admin/users_list.blade.php

```
 <!DOCTYPE html>
 <html>
 <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>users list</title>
    <link rel="stylesheet" href="./css/reset.css">
    <link rel="stylesheet" href="/css/users_list.css">
    <!-- UIkit CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css"/>
    <!-- UIkit JS -->
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
    <link rel="stylesheet" href="/css/style.css">
```

```
</head>
<body>
<!--ナビゲーション-->
<nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
        <ul class="uk-navbar-nav">
            <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        </ul>
    </div>
    <div class="uk-navbar-right">
        <ul class="uk-navbar-nav">
            <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
            <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
        </ul>
    </div>
</nav>
<!--main content-->
<div class="main-area">
    <div class="uk-card-default uk-card-body">
        <div class="uk-card-title">
            <h4>ユーザー 一覧</h4>
        </div>
        <table class="list">
            <thead>
            <tr>
                <th>ID</th>
                <th>氏名</th>
                <th>メールアドレス</th>
                <th>詳細</th>
            </tr>
            </thead>
            <tbody>
            @foreach($users as $user)
                <tr>
                    <td class="id">{{$user->id}}</td>
                    <td class="name">{{$user->name}}</td>
                    <td class="email">{{$user->email}}</td>
                    <td class="detail"><a href="/admin/users_list/{{ $user->id }}"
                                          class="uk-button uk-button-small uk-button-primary">詳細</a></td>
```

```
            </tr>
            @endforeach
            </tbode>
        </table>
    </div>
</div>
</body>
</html>
```

user_info.blade.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>ユーザー情報</title>
  <link rel="stylesheet" href="./css/reset.css">
  <!-- UIkit CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
<link rel="stylesheet" href="/css/style.css">

</head>
<body>
  <!--ナビゲーション-->
  <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
    <div class="uk-navbar-left">
      <ul class="uk-navbar-nav">
        {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
        @can('isAdmin')
        <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
```

```
        @else
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @endcan
      </ul>
    </div>
    <div class="uk-navbar-right">
      <ul class="uk-navbar-nav">
        <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
        <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
      </ul>
    </div>
  </nav>
  <!-- off-canvas 単に便利だから入れているだけで不要なら消してください-->
  <div id="my-id" uk-offcanvas="overlay: true">
    <div class="uk-offcanvas-bar">
      <button class="uk-offcanvas-close" type="button" uk-close></button>
      <ul class="uk-nav uk-nav-default">
        <li>単に便利だから入れているだけで不要なら消してください</li>
        <li><a href="./add-content.html" teitle="製品登録">製品登録</a></li>
        <li><a href="./user-info.html" teitle="ユーザー情報">ユーザー情報</a></li>
        <li><a href="./css-check.html" title="CSS の当たり方をチェックする">style 確認用のページ</a></li>
        <li><a href="./partslist.html" title="使えそうなパーツ">パーツ確認用のページ</a></li>
      </ul>
    </div>
  </div>
  <!--main content-->
  <div class="main-area">
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
      <div class="uk-card-title uk-margin uk-margin-auto">
        <h3>ユーザー情報</h3>
      </div>
      <div class="uk-margin-auto">
        <div class="uk-grid-divider" uk-grid>
          <div class="uk-margin-auto-vertical uk-width-1-3">
            <label class="uk-inline">
              <span class="uk-icon" uk-icon="icon: user"></span>
              <span class="uk-margin-auto uk-visible@s">名前</span>
            </label>
          </div>
```

```
          <div class="uk-margin-auto-vertical uk-width-1-3">
            <h5 class="uk-text-muted">{{ $user->name }}</h5>
          </div>
        </div>
        <div class="uk-grid-divider" uk-grid>
          <div class="uk-margin-auto-vertical uk-width-1-3">
            <label class="uk-inline">
              <span class="uk-icon" uk-icon="icon: mail"></span>
              <span class="uk-margin-auto uk-visible@s">メールアドレス</span>
            </label>
          </div>
          <div class="uk-margin-auto-vertical uk-width-1-3">
            <h5 class="uk-text-muted">{{ $user->email }}</h5>
          </div>
        </div>
      </div>
      @cannot('isAdmin')
      <div class="uk-card-footer uk-margin uk-text-center">
        <button class="uk-button uk-button-primary" type="button"
onclick="location.href='{{ route('user_info_edit') }}'">ユーザー情報を編集する</button>
      </div>
      @endcan
    </div>
  </div>
  @cannot('isAdmin')
  <div class="main-area">
    <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
      <div class="uk-card-title uk-margin uk-margin-auto">
        <h3>{{$user->name}}さんが登録した製品一覧</h3>
      </div>
      <div class="uk-margin-auto">
        <div class="uk-grid-divider" uk-grid>
        @foreach($contents as $content)
          <div class="contents-list">
            <img src="{{ asset($content->image_path) }}" alt="製品画像">
            <div class="content">
              <ul class="content-ul">
                <li>ID<span class="mar-03"> | </span>{{$content->id}}</li>
                <li>ブランド名 | {{$content->brand}}</li>
```

```
                <li>製品名<span class="mar-01"> | </span>{{$content->name}}</li>
                <li>製品コード | {{$content->identifier}}</li>
                <li>登録日<span class="mar-01"> | </span>{{$content->created_at}}</li>
                <li class="info-delete"><a href="{{ route('content_detail') }}?search={{$content->identifier}}"
class="uk-button uk-button-small uk-button-primary">詳細</a></li>
                <li class="info-delete">
                <form action="{{route('content_delete', $content->id)}}" method="POST">
                   @csrf
                   @method('DELETE')
                   <input type="submit" class="uk-button uk-button-small uk-button-danger" value="削除">
                 </form>
                </li>
              </ul>
             </div>
            </div>
          @endforeach
          </div>
        </div>
       </div>
     </div>
  </div>
   @endcan
  </body>
  </html>
```

user_info_edit.blade.php

```
  <!DOCTYPE html>
  <html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>ユーザー情報編集</title>
    <link rel="stylesheet" href="./css/reset.css">
    <!-- UIkit CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
```

```
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
 <link rel="stylesheet" href="./css/style.css">

 </head>
 <body>
  <!--ナビゲーション-->
  <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
     <div class="uk-navbar-left">
      <ul class="uk-navbar-nav">
       {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
        @can('isAdmin')
        <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @else
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @endcan
      </ul>
     </div>
     <div class="uk-navbar-right">
        <ul class="uk-navbar-nav">
           <li><a href="{{ route('user_info') }}">ユーザー情報</a></li>
           <li><a href="{{ route('user_logout') }}">ログアウト</a></li>
        </ul>
     </div>
  </nav>
  <!--main content-->
   <div class="main-area">
     <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
      <div class="uk-card-title uk-margin uk-margin-auto">
       <h3>ユーザー情報編集</h3>
      </div>
      <form class="h-adr" action="{{ route('user_updete') }}" method="post" enctype="multipart/form-data">
      @csrf
       <div class="uk-margin-auto">
        <div class="uk-grid-divider" uk-grid>
         <div class="uk-margin-auto-vertical uk-width-1-3">
          <label class="uk-inline">
```

```
                <span class="uk-icon" uk-icon="icon: user"></span>
                <span class="uk-margin-auto uk-visible@s">名前</span>
              </label>
            </div>
            <div class="uk-margin-auto-vertical uk-width-2-3">
              <input class="uk-input" type="text" name="name" value="{{ $user->name }}" required placeholder="名前を入力...">
            </div>
          </div>
          <div class="uk-grid-divider" uk-grid>
            <div class="uk-margin-auto-vertical uk-width-1-3">
              <label class="uk-inline">
                <span class="uk-icon" uk-icon="icon: mail"></span>
                <span class="uk-margin-auto uk-visible@s">メールアドレス</span>
              </label>
            </div>
            <div class="uk-margin-auto-vertical uk-width-2-3">
              <input class="uk-input" type="text" name="email" value="{{ $user->email }}" required placeholder="メールアドレスを入力...">
            </div>
          </div>
        </div>
        <div class="uk-card-footer uk-margin uk-text-center">
          <button class="uk-button uk-button-primary" type="submit" href="{{ route('user_info_edit_success') }}">確定</button>
        </div>
      </form>
    </div>
  </div>
 </body>
 </html>
```

user_info_edit_success.blade.php

```
 <!DOCTYPE html>
 <html>
 <head>
```

```
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>登録完了</title>
  <link rel="stylesheet" href="./css/reset.css">
  <!-- UIkit CSS -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/css/uikit.min.css" />
  <!-- UIkit JS -->
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/uikit@3.5.8/dist/js/uikit-icons.min.js"></script>
 <link rel="stylesheet" href="./css/style.css">

 </head>
 <body>
  <!--ナビゲーション-->
  <nav class="uk-navbar-container uk-navbar-transparent" uk-navbar>
     <div class="uk-navbar-left">
      <ul class="uk-navbar-nav">
       {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
      </ul>
     </div>
     <div class="uk-navbar-right">
      <ul class="uk-navbar-nav">
       {{--  <li><button  class="uk-navbar-toggle" uk-navbar-toggle-icon uk-toggle="target: #my-
id"></button></li>  --}}
        @can('isAdmin')
        <li><a href="{{ route('admin_index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @else
        <li><a href="{{ route('index') }}" class="uk-navbar-item uk-logo">高級バッグ追跡システム</a></li>
        @endcan
      </ul>
     </div>
  </nav>
  <!--main content-->
  <div class="main-area">
   <div class="uk-card uk-card-body uk-card-default uk-margin uk-margin-auto">
     <div class="uk-card-title uk-margin uk-margin-auto">
```

```
        <h3>ユーザー情報編集</h3>
      </div>
      <div class="uk-text-center uk-h4">
        <div class="uk-inline">
          <span class="uk-text-success" uk-icon="icon: check"></span>
          <span class="uk-text-middle uk-text-success">ユーザー情報が変更されました</span>
        </div>
      </div>
      <div class="uk-card-footer uk-margin uk-text-center">
        <button class="uk-button uk-button-primary" type="button" onclick="location.href='{{ route('index') }}'">戻る
</button>
      </div>
    </div>
  </div>
 </body>
 </html>
```

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

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