

Mercredi 18 Décembre 2019,

TD8 Monnaies Numériques

Toutes les informations concernant les artistes, venues, concerts et tickets seront modélisés par des structures. Ensuite, pour pouvoir leur donner un id unique on va utiliser un mapping avec comme clé l'id et comme valeur la structure. L'id sera une variable globale que l'on initialisera à 1 et qui s'incrémentera à chaque fois que l'on voudra créer un nouvel objet (que ce soit artiste ou venue etc.).

1) Create/modify artist profile :

```
// Structure artiste, id artiste depuis la variable globale
uint256 public nextArtistNumber;
mapping(uint256 => Artist) public artistsRegister;

struct Artist {
    address payable owner;
    bytes32 name;
    uint256 artistCategory;
    uint256 totalTicketSold;
}

function createArtist(bytes32 artistName, uint256 artistCategory) public {
    Artist memory newArtist = Artist(msg.sender, artistName, artistCategory, 0);
    artistsRegister[nextArtistNumber] = newArtist;
    nextArtistNumber += 1;
}

function modifyArtist(uint256 artistId, bytes32 nameArtist, uint256 artistCategory, address payable newOwner) public {
    // Pour passer le try catch du test,
    // il faut verifier que la personne qui veut modifier les informations de l'artiste est bien l'artiste en personne
    require(artistsRegister[artistId].owner == msg.sender);
    artistsRegister[artistId].owner = newOwner;
    artistsRegister[artistId].name = nameArtist;
    artistsRegister[artistId].artistCategory = artistCategory;
}
```

La fonction createArtist permet de créer une nouvelle structure artiste en renseignant les champs nécessaires et d'affecter la valeur au mapping à l'aide de la clé id.

La fonction modifyArtist permet de modifier les informations de l'artiste, par contre pour modifier il faut être l'artiste lui-même, donc si ce n'est pas le propriétaire qui demande de modifier il ne passe pas le require.

Résultat :

```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test/01_creatingArtistProfile.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Creating artist profile
  ✓ Create an artist profile (448ms)
  ✓ Modifying an artist profile (694ms)

2 passing (2s)
```

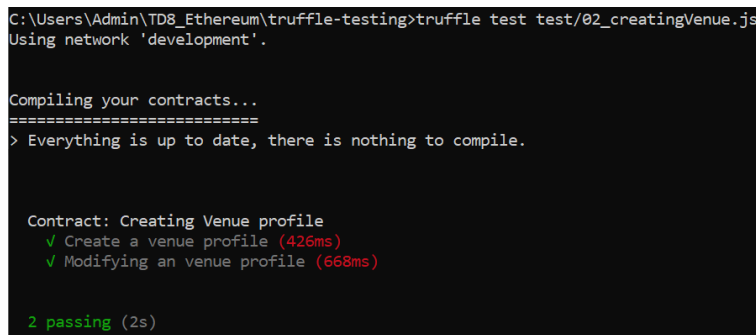
2) createVenue/modifyVenue :

```
function createVenue(bytes32 _name, uint256 _capacity, uint256 _standardComission) public {
    Venue memory newVenue = Venue(msg.sender, _name, _capacity, _standardComission);
    venuesRegister[nextVenueNumber] = newVenue;
    nextVenueNumber += 1;
}

function modifyVenue(uint256 venueId, bytes32 venueName, uint256 venueCapacity, uint256 venueComission, address payable owner) public {
    // Seul la personne qui a creer la venue peut la modifier, d'ou le require
    require(venuesRegister[venueId].owner == msg.sender);
    venuesRegister[venueId].name = venueName;
    venuesRegister[venueId].capacity = venueCapacity;
    venuesRegister[venueId].standardComission = venueComission;
    venuesRegister[venueId].owner = owner;
}
```

Même principe que pour l'artiste mais cette fois-ci avec les venues.

Résultat :



```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test/02_creatingVenue.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Creating Venue profile
  ✓ Create a venue profile (426ms)
  ✓ Modifying an venue profile (668ms)

2 passing (2s)
```

3) creatingConcert/validateConcert/emitTicket/useTicket :

Même principe qu'artiste et venue pour créer un concert. Ensuite, pour valider un concert, il faut qu'il soit validé par l'artiste et par la venue. Ensuite, pour émettre les tickets, il faut passer la condition suivante : seuls les artistes peuvent émettre les tickets ». Si c'est l'artiste qui veut émettre un ticket, alors on crée une nouvelle structure ticket avec les informations à renseigner. Ensuite, chaque ticket à un propriétaire, donc pour pouvoir utiliser un ticket on vérifie que le propriétaire du ticket et la personne qui en fait la demande est bien la même personne. On vérifie également que la venue et l'artiste a bien validé le concert et que la date du concert correspond.

```
function createConcert(uint256 _artistId, uint256 _venueId, uint256 _concertDate, uint256 _ticketPrice) public {
    Concert memory newConcert = Concert(_artistId, _venueId, _concertDate, _ticketPrice, 0, 0, false, false);
    concertsRegister[nextConcertNumber] = newConcert;
    validateConcert(nextConcertNumber);
    nextConcertNumber += 1;
}

function validateConcert(uint256 _concertId) public {
    // Tester si l'artiste du concert en question a bien émis la création de ce concert
    uint256 idArtist = concertsRegister[_concertId].artistId;
    uint256 idVenue = concertsRegister[_concertId].venueId;
    // On regarde si l'adresse de l'artiste est équivalente à l'adresse qui a émis la création du concert
    // Pareil avec la venue
    if (artistsRegister[idArtist].owner == msg.sender) concertsRegister[_concertId].validatedByArtist = true;
    if (venuesRegister[idVenue].owner == msg.sender) concertsRegister[_concertId].validatedByVenue = true;
}

function emitTicket(uint256 _concertId, address payable _ticketOwner) public {
    // Only artists can emit tickets
    uint256 artistIdConcert = concertsRegister[_concertId].artistId;
    address payable artistOwnerAddress = artistsRegister[artistIdConcert].owner;
    require(artistOwnerAddress == msg.sender);

    concertsRegister[_concertId].totalSoldTicket += 1;
    Ticket memory newTicket = Ticket(_ticketOwner, true, _concertId, 0, false, 0);
    ticketsRegister[nextTicketNumber] = newTicket;
    nextTicketNumber += 1;
}

function useTicket(uint256 _ticketId) public {
    // Vérifier que la personne qui veut utiliser le ticket est bien le propriétaire du ticket
    require(ticketsRegister[_ticketId].owner == msg.sender);

    // Vérifier que ce ticket vient bien d'un concert qui a été validé par l'artiste et la venue
    require(concertsRegister[ticketsRegister[_ticketId].concertId].validatedByArtist);
    require(concertsRegister[ticketsRegister[_ticketId].concertId].validatedByVenue);

    // Et finalement il faut vérifier si le ticket correspond bien aux dates du concert
    require(concertsRegister[ticketsRegister[_ticketId].concertId].concertDate < now + 60*60*24);

    // We used the ticket so we delete it now in the mapping
    delete ticketsRegister[_ticketId];
}
```

Résultat :

```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test\03_concertManagement.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Concert management functions
  ✓ Creating a concert (1218ms)
  ✓ Emitting tickets (1749ms)
  ✓ Using tickets (2080ms)

3 passing (7s)
```

4) buyTicket/transferTicket :

Pour acheter un ticket, il suffit d'incrémenter le nombre total de tickets achetés et la cagnotte accumulé niveau concert. Et côté ticket, il faut assigner le concert, le montant payé, le nouveau propriétaire du ticket et le rendre disponible à être utilisé.

Pour transférer un ticket, seul le propriétaire lui-même peut le faire.

```
function buyTicket(uint256 _concertId) public payable {
    // On fait monter le nombre de tickets vendus et increment de l'argent gagne pour le concert donnee
    concertsRegister[_concertId].totalSoldTicket +=1;
    concertsRegister[_concertId].totalMoneyCollected += msg.value;

    // On met le montant du ticket
    ticketsRegister[nextTicketNumber].concertId = _concertId;
    ticketsRegister[nextTicketNumber].amountPaid = msg.value;
    ticketsRegister[nextTicketNumber].owner = msg.sender;
    ticketsRegister[nextTicketNumber].isAvailable = true;
    nextTicketNumber +=1;
}

function transferTicket(uint256 _ticketId, address payable _newOwner) public {
    require(ticketsRegister[_ticketId].owner == msg.sender);
    ticketsRegister[_ticketId].owner = _newOwner;
}
```

Résultat :

```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test\04_TicketBuyingAndTransferring.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Concert management functions
  ✓ Buying tickets (584ms)
  ✓ Using bought tickets (586ms)
  ✓ Transferring tickets (583ms)

3 passing (4s)
```

5) cashOutConcert :

Quand on veut fermer un concert, il faut pouvoir redistribuer l'argent à l'artiste et à la venue. Il faut vérifier d'abord que c'est bien l'artiste du concert en question qui veut récupérer l'argent et que la date du concert est inférieure à celle actuelle. Si c'est le cas, on calcule l'argent pour la venue et pour l'artiste. Sur le nombre total d'argent dans la cagnotte, il y a un certain pourcentage qui revient à la venue (calculé avec la commission). Le pourcentage est la commission / 10 000. Pour l'artiste c'est tout l'argent qu'il reste. Ensuite, on transfère l'argent : de la forme « addressToTransferTo.transfer(argent) ».

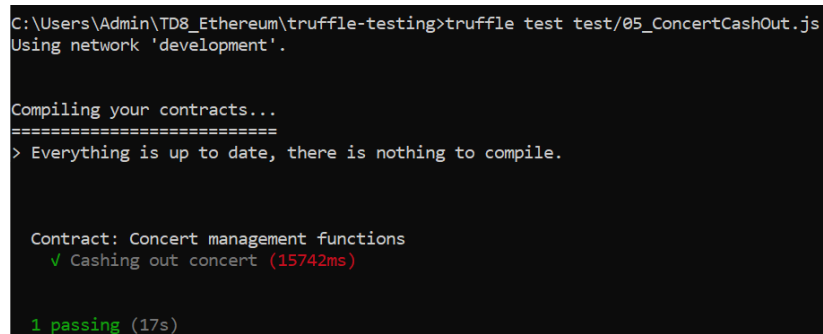
```
function cashOutConcert(uint256 _concertId, address payable _cashOutAddress) public {
    // On vérifie que c'est bien l'artiste du concert qui veut récupérer l'argent et que le concert a bien été effectué
    uint256 artistIdConcert = concertsRegister[_concertId].artistId;
    address payable addressOwnerArtist = artistsRegister[artistIdConcert].owner;
    require(addressOwnerArtist == msg.sender);
    require(concertsRegister[_concertId].concertDate < now);

    // Maintenant on récupère l'argent en utilisant la fonction .transfer (<address payable>.transfer(uint256 amount))
    uint256 totalMoney = concertsRegister[_concertId].totalMoneyCollected;
    uint256 venueIdConcert = concertsRegister[_concertId].venueId;
    uint256 commissionVenue = venuesRegister[venueIdConcert].standardCommission / 10000;
    // Venue get a certain percentage of the ticket price : commission / 10 000
    uint256 venueShare = totalMoney * commissionVenue;
    uint256 artistShare = totalMoney - venueShare;

    _cashOutAddress.transfer(artistShare);
    venuesRegister[concertsRegister[_concertId].venueId].owner.transfer(venueShare);

    // Le concert est terminé
    artistsRegister[concertsRegister[_concertId].artistId].totalTicketSold += concertsRegister[_concertId].totalSoldTicket;
    delete concertsRegister[_concertId];
}
```

Résultat :



```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test/05_ConcertCashOut.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Concert management functions
  ✓ Cashing out concert (15742ms)

1 passing (17s)
```

6) offerTicketForSale/buySecondHandTicket :

Pour pouvoir revendre un ticket, il faut vérifier que le prix demandé n'est pas supérieur au prix initial du concert sinon c'est une arnaque, que le ticket est disponible et valide. Ensuite, on peut émettre le ticket à vendre avec son prix.

Pour acheter un ticket qui est revendu par un propriétaire, il faut que le prix proposé corresponde au prix demandé par le vendeur et qu'il soit en vente. Ensuite, si c'est le cas on fait le transfert d'argent au propriétaire du ticket et on devient par la suite le nouveau propriétaire.

```
function offerTicketForSale(uint256 _ticketId, uint256 _salePrice) public {
    require(ticketsRegister[_ticketId].owner == msg.sender);
    // Il faut que le prix du ticket soit supérieur au prix de vente
    uint256 ticketIdFromConcertId = ticketsRegister[_ticketId].concertId;
    require(concertsRegister[ticketIdFromConcertId].ticketPrice > _salePrice);
    // Le ticket est-il disponible ou a-t-il déjà été utilisé
    require(ticketsRegister[_ticketId].isAvailable == true);

    ticketsRegister[_ticketId].isAvailableForSale = true;
    ticketsRegister[_ticketId].isAvailable = true;
    ticketsRegister[_ticketId].salePrice = _salePrice;
}

function buySecondHandTicket(uint256 _ticketId) public payable {
    require(msg.value >= ticketsRegister[_ticketId].salePrice);
    require(ticketsRegister[_ticketId].isAvailable == true);
    require(ticketsRegister[_ticketId].isAvailableForSale == true);

    // On fait le transfert d'argent au propriétaire du billet
    ticketsRegister[_ticketId].owner.transfer(msg.value);
    // On change le nom du ticket pour le nom du nouveau propriétaire
    ticketsRegister[_ticketId].owner = msg.sender;
}
```

Résultat :

```
C:\Users\Admin\TD8_Ethereum\truffle-testing>truffle test test/06_TicketSelling.js
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Concert management functions
  ✓ Putting up tickets to sell (646ms)
  ✓ Buying auctioned tickets (568ms)
  ✓ Using a ticket while it is on sale (503ms)

3 passing (6s)
```