Plane Insuance

Sommaire

1. Description du projet
2. Description technique de méthodes Solidity
   1. Mint function
   2. GetEthLastPrice function
3. Description technique de méthodes Javascript
   1. Appel d’API
   2. Appel à des méthodes du Smart Contract
4. Description du projet

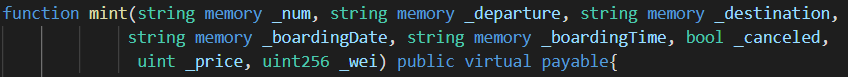
Une application décentralisée vivant dans la BlockChain Ethereum.

L’application permets à des voyageurs d’acheter des billets d’avions avec leur portefeuilles Ethereum. Ces billets d’avions sont stockés dans leur wallet, sur la blockchain, sous forme de NFT pour contenir des informations.  
Si jamais leur vol est annulé, ils peuvent demander à être rembourser à tout moment, à la hauteur de 70% du prix d’achat de leur billet.  
  
L’avantage de cette application d’assurance décentralisée, est qu’elle permet de s’affranchir d’un tiers de confiance, l’assureur. Le remboursement se fait automatiquement, tout comme le code du contrat la définit. Aucun conseiller d’assurance ne peut vous empêcher d’être rembourser, vous êtes sûr que le contrat d’assurance pour lequel vous avez souscrit, sera honorer tel quel.

1. Description technique de méthodes Solidity
   1. Mint function

Une image contenant texte

Description générée automatiquement



On définit une fonction ‘mint’, avec pour paramètres, ce qui est entre parenthèses. Le mot clé ‘memory’ définit une instanciation mémoire dans le smart contract des types non-primitifs.

‘Public’ est la visibilité de la fonction, ‘virtual’ est son modificateur, ‘payable’ est pour décrire qu’elle permet de faire une transaction appliquant une rémunération.



‘Require’ définit un test, dans ce cas-là, on test si ‘msg.value’ représentant la quantité de Wei envoyé au smart contract, a une valeur supérieure au paramètre ‘\_wei’, représentant le prix en Wei du token. Si le test est faux, alors le message qui suit est affiché.

‘msg.value’ est une variable récupéré par le smart contract, msg étant le wallet interagissant avec le smart contract, et value est la quantité de gas envoyé par ce même wallet.



‘\_tokenIdCounter’ est une variable définit dans le constructeur du smart contract, on incrémente ce compteur. Puis on définit une variable récupérant la valeur actuelle du compteur. Cette valeur sera utilisée tout le long de la fonction.



‘\_safeMint’ est une fonction héritée du standard ERC721 implémenté par le smart contract, cette fonction permets d’attribuer l’identifiant de token ‘actualIdCounter’, au wallet interagissant avec le contrat via ‘msg.sender’.

Une image contenant texte

Description générée automatiquement

Création d’un object de type <Struct>BoardingPassAttributes en lui passant en valeur les paramètres de la fonction mint.



Mapping de l’id du token acheté, à l’addresse du sender. Permet de stocker au sein du smart contract, qui détient tel token. Puis, mapping des attributes du token acheté, à son identifiant, pour stocker les informations du token.

* 1. GetLastEthPrice

Une image contenant texte

Description générée automatiquement

Fonction qui permet de récupérer le prix actuel de l’Ethereum, en dollar.

Elle le récupère via l’objet ‘priceFeed’, instancié dans le constructeur du Smart Contract.



AggregatorV3Interface étant une interface obtenue via ChainLink, l’adresse en paramètre corresponds à un nœud de données ChainLink, envoyant le prix actuel de l’Ethereum en dollar, sur Rinkeby.

1. Description technique de méthodes JavaScript
   1. Appel d’API

Une image contenant texte

Description générée automatiquement

L’appel d’API permets d’obtenir les données exposées par l’API.

L’API fournit des données via des endPoints, ça permet d’exposer seulement les données que l’on souhaite, et d’assurer la sécurité en filtrant les accès à la base de données. Ici, l’API appelé expose les données de tous les vols disponibles. Les données sont renvoyées en format Json.

* 1. Appel à des méthodes du Smart Contract

Une image contenant texte

Description générée automatiquement

‘mintNFT’ est une fonction JavaScript qui va permettre d’appeler les méthodes du Smart Contract, ont récupère les infos du Smart Contract via la méthode ‘getContract()’, puis on appelle la fonction ‘mint()’ en lui passant en paramètre les informations requises.



Cette ligne permet de passer en paramètre, la valeur à payer pour mint le NFT.