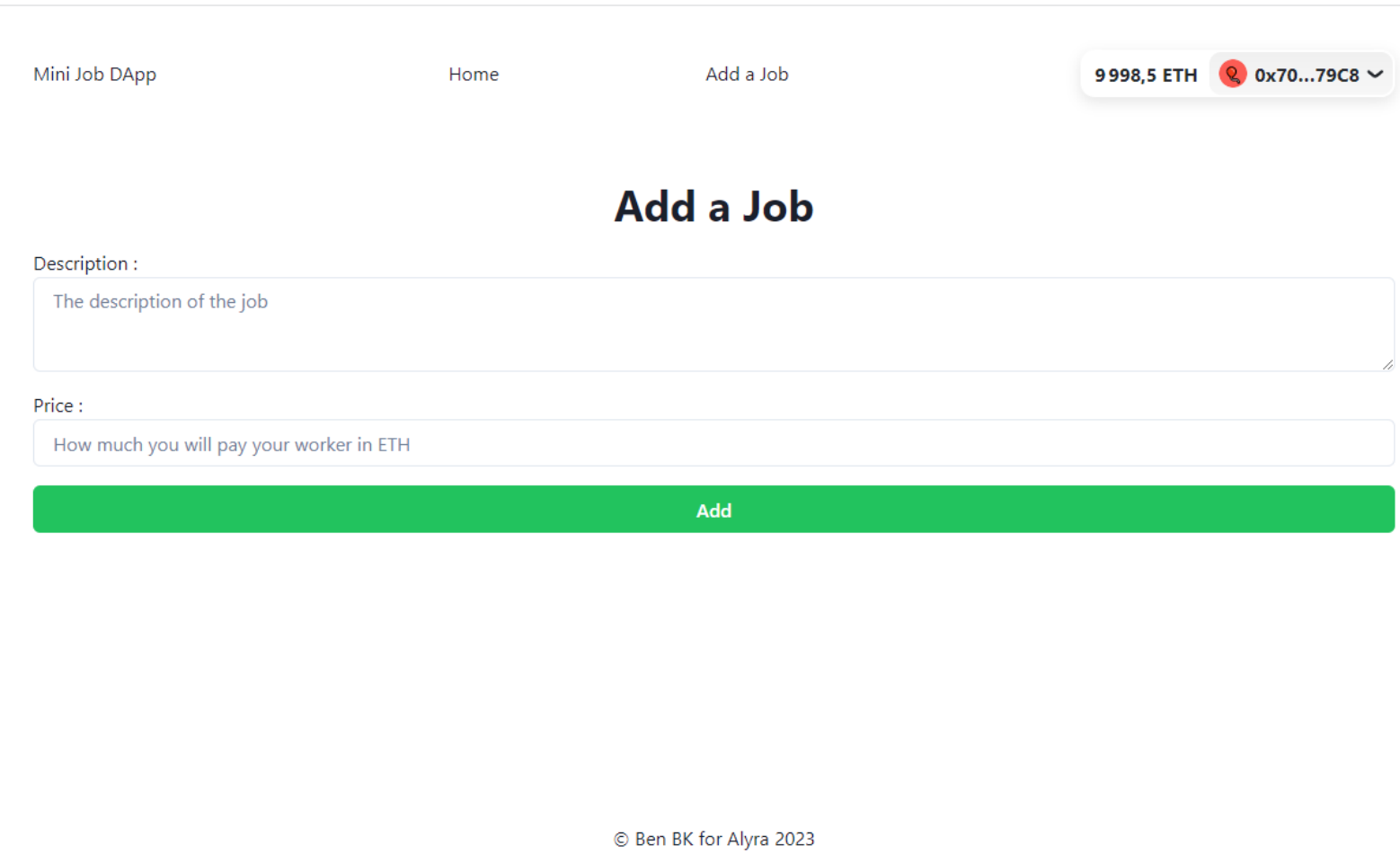
Mini Job Dapp

L'objectif est de créer une Dapp sur laquelle chacun sera libre de demander de l'aide, aider quelqu'un ou être aidé, et enfin, payer quelqu'un pour l'aide apportée ou être payé pour l'aide apportée.

Premièrement, l'utilisateur pourra créer un job :



Au niveau de l'UI, il y aura deux champs importants :

- un champs *description* du job
- un champs où l'utilisateur qui veut être aidé (qui créer le job) précise quel montant il donnera à la personne qui va l'aider (en ETH)

Au niveau technique, l'utilisateur qui créé le job donnera la « récompense » directement au smart contract. Lorsque la personne qui réservera le job aura terminé, l'utilisateur qui a créé le job déclenchera une fonction qui permettra de payer automatiquement le « travailleur ».

Plusieurs informations seront indispensables dans le smart contract pour chaque job :

- id
- auteur
- travailleur
- description
- prix
- si le job est terminé
- si le job est réservé

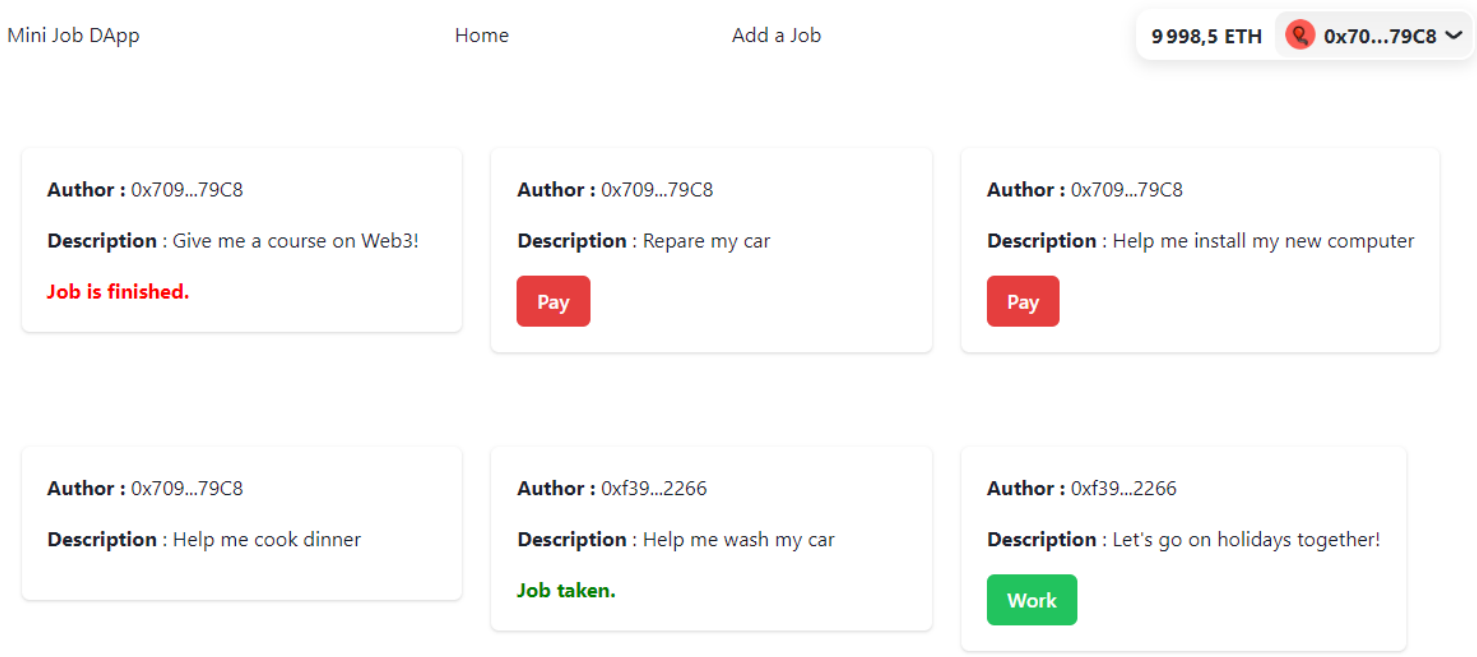
Voici le smart contract à compléter :

Untitled-1

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.17;

///@title A Mini Job Smart contract
///@author Ben BK
contract Jobs {

    struct Job {
        address author;
        address worker;
        string description;
        uint price;
        bool isFinished;
    }

    Job[] jobs;

    event jobAdded(address indexed author, string description, uint price, uint id, bool isFinished);
    event jobTaken(address indexed worker, uint id);
    event jobIsFinishedAndPaid(address indexed author, address indexed worker, uint id, uint pricePaid);

    ///@notice Allows to add a new job
    ///@param _description the complete description of the job
    function addJob(
        string calldata _description
    ) external payable {
    }

    ///@notice Allows to take the job
    ///@param _id the index of the job in the jobs array
    function takeJob(uint _id) external {
    }

    ///@notice Allows to end the job and to pay the worker
    ///@param _id the id of the job in the jobs array
    function setIsFinishedAndPay(uint _id) external {

    }
}
```

Comme vous pouvez le remarquer, on n'utilise pas de fonction en Solidity pour récupérer tous les jobs dans un tableau, en effet, cet aspect doit être géré d'une façon plus efficace grâce aux events !

Au niveau de la liste des jobs, voici le rendu attendu :



Les jobs auront plusieurs statuts possibles (dans l'ordre d'apparition sur l'image ci-dessus) :

- Le job est terminé. Il a été réservé par un travailleur et payé par le créateur du job. (1)
- Le job est réservé et le créateur doit payer le travailleur (2 et 3)
- Le job a été créé par son créateur (qui est ici la personne connectée), il ne peut donc pas prendre le job ni payer un travailleur pour le moment car le job n'a pas été réservé par un travailleur (4)
- Le job vient d'être pris par un travailleur (qui est différent du créateur) (5)
- Un job a été posté par une autre adresse que celle connectée et l'utilisateur connecté peut prendre le job. (6)

Si aucun job est présent sur la Dapp, il faut que le front le précise :