## Invite d’exercice

**Écrivez l’architecture Solidity (noms des fonctions et leurs entrées et sorties) pour les contrats intelligents de votre projet final. Téléchargez-le dans le dépôt GitHub que vous avez créé au chapitre 1.**

## Concept d’exercice

Depuis le dernier exercice, vous avez appris :

* Contexte du réseau Ethereum
* Comment les comptes, les transactions et l’état sont stockés sur le réseau Ethereum
* Comment exécuter un client Ethereum
* Comment les contrats intelligents s’intègrent dans notre modèle mental blockchain
* Comment utiliser Truffle
* Fondamentaux de solidité
* Modèles de conception de contrats intelligents
* Pièges de sécurité et vecteurs d’attaque
* Options de vérification et de protection de votre application

Pour cet exercice, nous aimerions que vous preniez votre compréhension des contrats intelligents et que vous l’appliquiez à votre projet final.

Plus précisément, nous aimerions que vous réfléchissiez au **flux de travail unique pour le futur utilisateur de votre projet** que vous avez écrit dans le premier exercice. Pour ce flux de travail, écrivez les fonctions de contrat intelligent de base qui seront requises. Vous n’avez pas encore besoin d’écrire la logique de solidité !

## Parties d’exercice

**1. Dans un premier temps, écrivez simplement le nom de la fonction, les entrées dont la fonction a besoin et ce qu’elle pourrait retourner. Démarrez un nouveau fichier dans votre dépôt GitHub et commencez à esquisser les fonctions.**

Voici à quoi cela pourrait ressembler pour notre exemple de vote du premier exercice :

*1. Les utilisateurs devront s’inscrire d’une manière ou d’une autre sur le contrat*

function registerVoter(address \_voter) {

// registers voter

};

*2.Ils doivent identifier la campagne sur laquelle ils votent*

function registerVote(uint campaignID) {

// registers the vote of the voter

};

*3. Ils devront soumettre un vote pour cette campagne, mais*

*4. Ils ne peuvent pas voter deux fois pour une seule campagne.*

modifier onlyVoteOnce() {

// checks the vote hasn't voted before

\_

};

Encore une fois, vous n’avez pas encore besoin d’écrire la logique interne (bien que vous le puissiez). **Le but est de commencer à réfléchir aux comportements de l’utilisateur en termes d’actions discrètes et à ce que ces actions exigent**. Vous pouvez voir dans l’exemple ci-dessus comment nous commençons déjà à avoir une idée de certaines variables globales dont le contrat aura besoin pour exécuter ces fonctions. C’est le but de cet exercice, **commencer à comprendre les contours généraux de la façon dont un utilisateur interagira avec votre code de contrat intelligent.**

**2. Une fois que vous avez quatre à cinq fonctions esquissées, validez et poussez le changement vers votre dépôt GitHub final de projet existant.** Une fois que vous avez poussé vers le référentiel, partagez votre lien avec votre groupe d’étude ou vos sessions ou dans le Discord !

[**https://github.com/YOUR\_GITHUB\_USERNAME\_HERE/education-dao-final-project**](https://github.com/YOUR_GITHUB_USERNAME_HERE/blockchain-developer-bootcamp-final-project)

Bon croquis !