

Développeur blockchain

Détail de votre parcours

Prototyper, Tester, Déployer

Vos formateurs

Cyril Castagnet

Développeur blockchain

Formé au Développement Web,
Solidity et à la simulation
quantique.

Après avoir fondé une entreprise
de consulting sur le thème de la
blockchain, il a rejoint Alyra.



Vos formateurs

Benjamin Brucher

Alias "Ben BK" /Développeur blockchain

Formateur Web2.0 et Web3.0.

Youtubeur dans le développement
blockchain.



A qui s'adresse le programme ?

Développeur Web

Développeur logiciel

Chief technical officer

Ingénieur informatique

Celles et ceux désirant prendre en
main l'innovation

Prérequis nécessaires

Cette formation exigeante permet d'accompagner progressivement l'apprenant de la découverte d'une technologie et la conception de Smart Contract dans le but de déployer des solutions innovantes.

Il est nécessaire d'avoir de solides bases en programmation et de savoir réaliser une interface Web (JavaScript/Node.js, HTML/CSS, Git/GitHub).

Un week end pour travailler

Le contenu est débloqué le jeudi 9h, pour vous laisser les week end pour lire / voir / faire le cours et les exercices.

Séance de question réponse et cours

Le lundi après le talk, on reprends les questions importantes. Avant de reprendre le cours et aller plus loin les mardi mercredi et jeudi!

Talk Alyra & Talk Technique

Lundi et jeudi, vous serez amené à rencontrer des professionnels du secteur.

Nouveau contenu
chaque semaine

Sommaire du programme

1. Module introductif 2 semaines
2. Solidity & Outils 2 semaines
3. Tests unitaires 1 semaine
4. Sécurité & Optimisations 1 semaine
5. Création de DApps 2 semaines
6. La Finance décentralisée 1 semaine
7. Les NFTs 1 semaine
8. Projet final et soutenances 2 semaines

01

Module introductif

- Un aboutissement de recherche scientifique
- Blockchain et valeur
- Découverte d'Ethereum
- Les premières manipulations

2 semaines

- L'apprentissage d'un nouveau langage
- Vos premiers projets
- Découverte des outils de l'écosystème

2 semaines

02 Solidity & Outils

03

Tests unitaires

- Retour sur un essentiel du développement
- Une méthodologie claire
- Etre prêt à développer pour le monde professionnel

1 semaine

- Connaitre les failles
- Savoir s'en prémunir
- Faire du "beau" code
- Connaitre les normes

1 semaine

04 Sécurité & optimisations

05

Création de Dapps

- Savoir faire rapidement un front end
- Lier nos contrats à notre front
- Déployer en production

2 semaines

- Comprendre la révolution DeFi
- Maîtriser sa théorie complète
- Voir du code en production
- Intégrer sur des protocoles

1 semaine

06 Finance décentralisée

07

Les NFTs

- L'innovation derrière le NFT
- Maîtriser les technologies associées
- Produire vos premiers NFT grâce aux librairies
- Savoir réaliser un produit NFT complexe

1 semaine

- Formaliser votre projet
- Collaborer avec une équipe de consultant & expert DeFi
- Réaliser votre prototype de bout en bout
- Soutenir devant un jury de professionnel

2 semaines

08 Projet final et soutenances

Les 4 épreuves

1. Réalisation d'un Smart Contract de vote (4 pts - en solo)
2. Optimisation & Testing d'un Smart Contract (2 pts - en solo)
3. Création d'une DApps de voting (4 pts - en duo)
4. Un projet final d'envergure en équipe hybride (10 pts - en solo)

```

17 // Contract to store and redeem money.
18 contract Store {
19     struct Safe {
20         address owner;
21         uint amount;
22     }
23
24     Safe[] public safes;
25
26     /// @dev Store some ETH.
27     function store() public payable {
28         safes.push(Safe({owner: msg.sender, amount: msg.value}));
29     }
30
31     /// @dev Take back all the amount stored.
32     function take() public {
33         for (uint i; i<safes.length; ++i) {
34             Safe storage safe = safes[i];
35             if (safe.owner==msg.sender && safe.amount!=0) {
36                 payable(msg.sender).transfer(safe.amount);
37                 safe.amount=0;
38             }
39         }
40
41     }
42 }
43
44 /*** Exercice 2 ***/
45 // You can buy some object.
46 // Further purchases are discounted.
47 // You need to pay basePrice / (1 + objectBought), where objectBought is the number of object you previously bought.
48 contract DiscountedBuy {
49     uint public basePrice = 1 ether;
50     mapping (address => uint) public objectBought;
51
52     /// @dev Buy an object.
53     function buy() public payable {
54         require(msg.value * (1 + objectBought[msg.sender]) == basePrice);
55         objectBought[msg.sender]+=1;
56     }
57
58     /** @dev Return the price you'll need to pay.
59      * @return price The amount you need to pay in wei.

```



Chaque mardi,
 mercredi et jeudi
 lives de 18h à 20h30

Des questions ? Go Discord !

