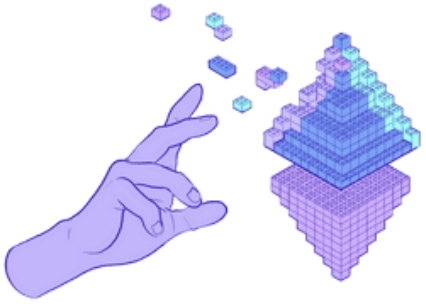
Documentation Technique

THE INTERNATIONAL INSTITUTE OF
SUPINFO
INFORMATION TECHNOLOGY

Introduction



Contexte :

L'équipe de **ImmoDApp** a pour but d'élaborer un smart contrat pour la vente de biens immobiliers.

Dans le cadre de ce projet, nous devions créer des smart contrats sur Ethereum avec son déploiement ainsi qu'une interaction avec un site web.

Dans ce projet, nous appellerons " token " les biens immobiliers gérés par le contrat intelligent.

Tout au long de ce document nous documenterons nos choix, leurs implémentations, les contraintes et détails auxquelles nous avons pu faire face.

Description du projet :

Le site **ImmoDApp** développais par nos soins a pour but d'interagir avec le ou les contrats.

La connexion à notre site et l'interaction se fera grâce au portefeuille Ethereum Metamask.

Sur la page d'accueil, on peut voir tous les token créés sur smart contract.

A partir de l'interface web, il peut être possible pour l'utilisateur connecté à Metamask de créer un token.

Un token a différents attributs :

- Un nom
- Une adresse
- Un prix (dans Ether)
- Une ou plusieurs images

Depuis l'interface web, le propriétaire du jeton peut le vendre ou en arrêter la vente.

Il est possible, depuis le site web, d'acheter les jetons mis en vente par d'autres propriétaires. Lorsque le jeton est vendu, il est transféré à son nouveau propriétaire et l'argent de la vente est reversé au vendeur.

Lors de la vente d'un jeton, une commission de 10 % est versée au contrat. Le propriétaire du contrat peut retirer les fonds des commissions à tout moment.

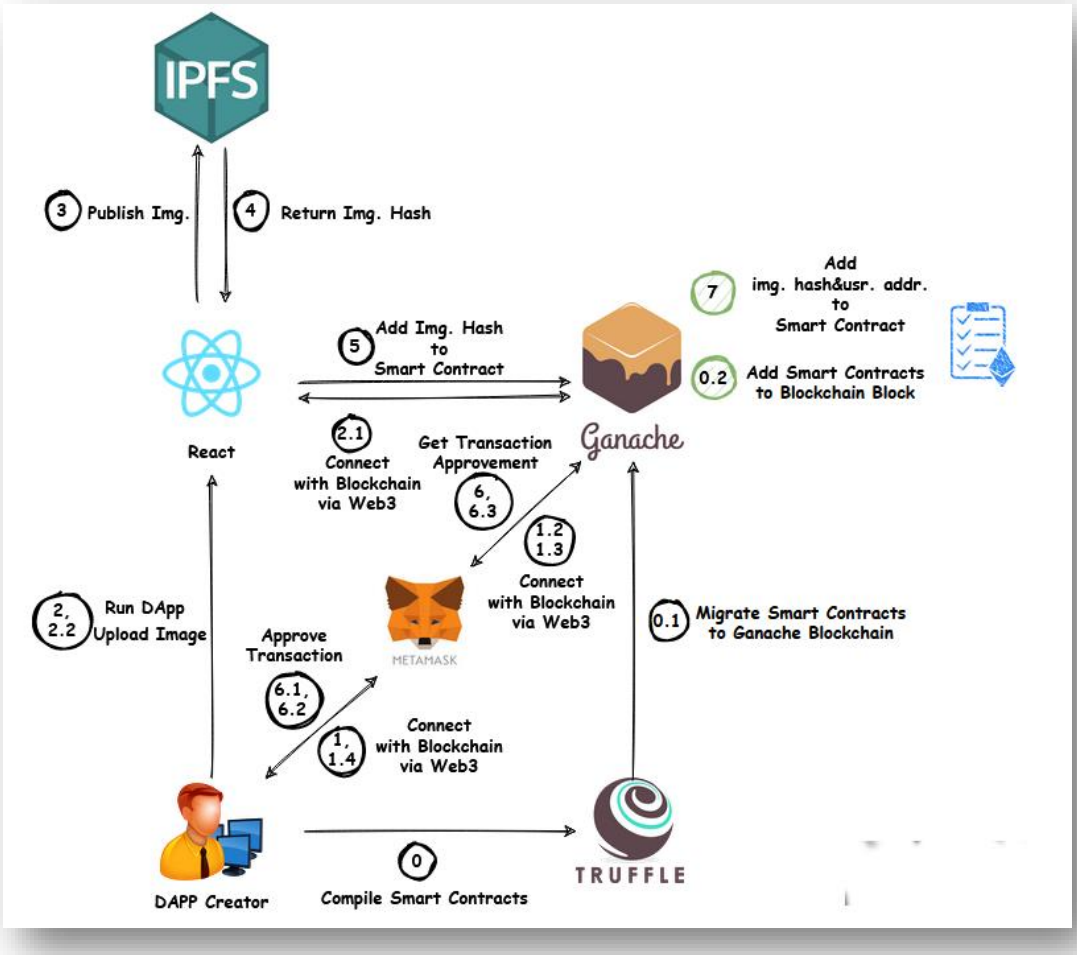
Expression des besoins du projet :

Le but premier du projet est que chaque utilisateur de la plateforme puisse acheter ou vendre son bien immobilier. Il pourra acheter son bien via son portefeuille Metamask qui est sous forme d'application.

Les objectifs du projet sont les suivants :

- Créer des token
- Voir les token
- Acheter un token
- Une commission est versée au contrat Smart lorsqu'un token est vendu
- Le propriétaire du contrat peut retirer les commissions
- Connection Metamask
- Mettre un token en vente
- Retirer un token de la vente

Architecture globale :



Technologie et BlockChain

JavaScript/ React

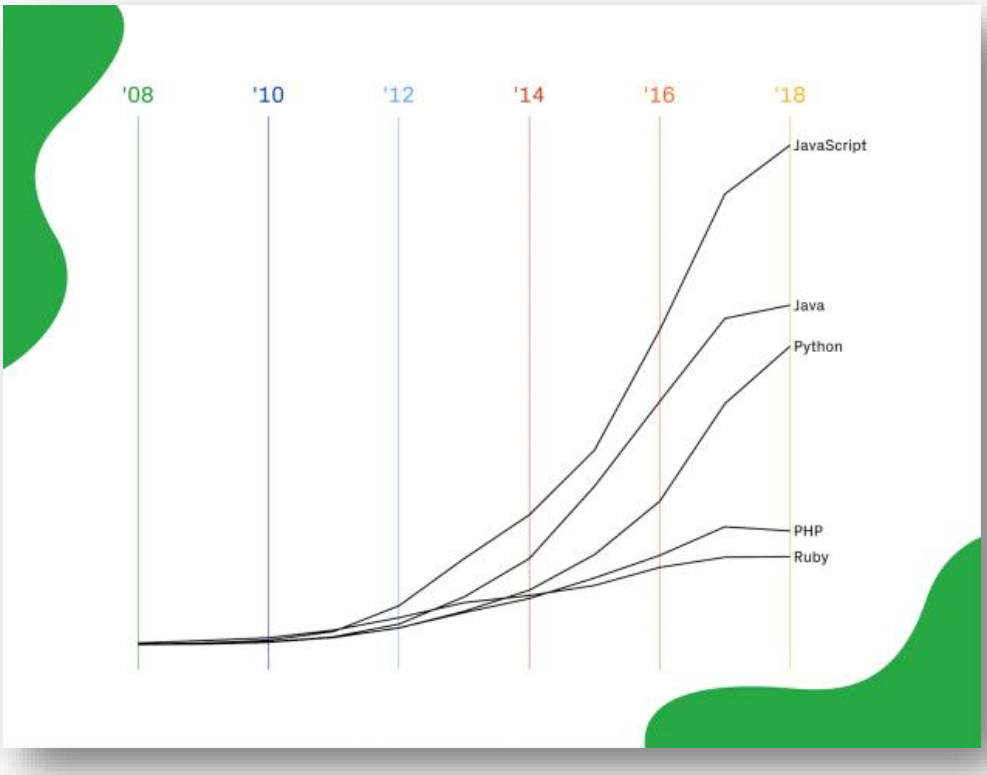
Après mure réflexion, nous nous sommes convenu d'utiliser des langages orienté WEB, principalement le **JavaScript/React** pour le développement.

Pourquoi me direz-vous, je vais détailler le pourquoi du comment nous avons choisis cette solution.

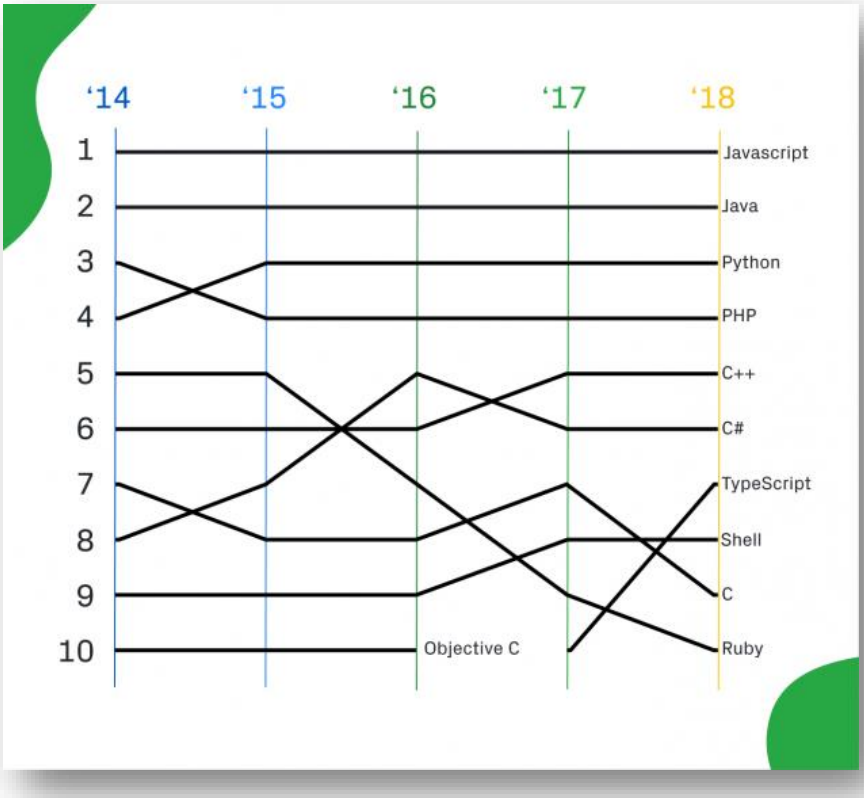
Premièrement, la popularité du langage est indéniable. Comme chaque année, GitHub publie son rapport annuel dévoilant ce que la communauté de 31 millions de développeurs a réalisé sur GitHub cette année.

GitHub revient sur les langages de programmation les mieux représentés cette année ainsi que les grandes tendances.

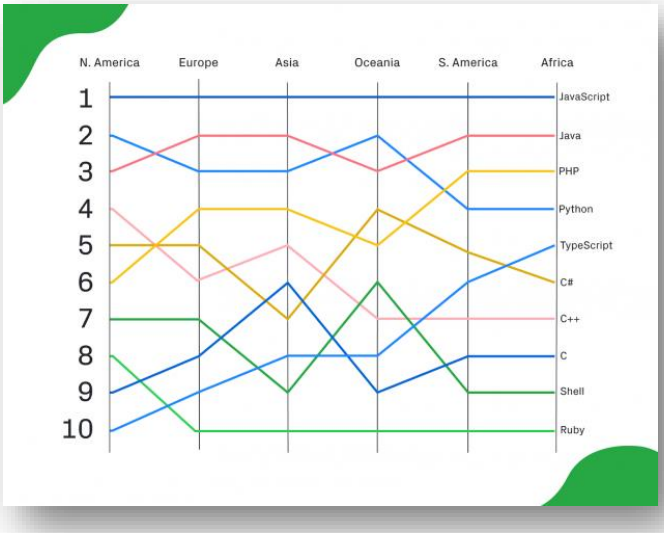
Les principaux langages de programmation en fonction des repositories créés de 2008 - 2018



Principaux langages de programmation par nombre de contributeurs au 30 septembre 2018



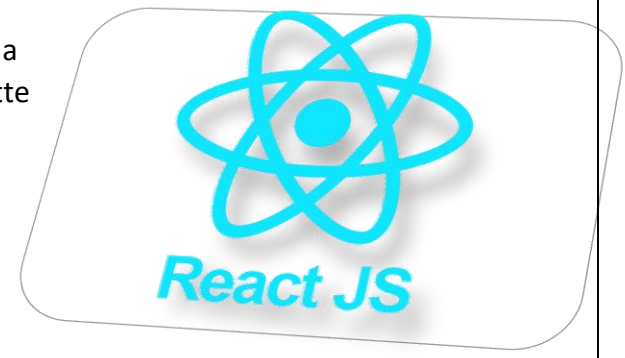
Les langages de programmation les plus utilisés par région en fonction de leur nombre de contributeurs au 30 septembre 2018



Plus populaire implique plus grosse communauté, plus de ressources exploitable, librairies, d'aide, moins chère…

Un dernier point important qui nous a fait privilégier JavaScript à un autre est que notre application entière est en javascript, ce qui permettra de la rendre compatible avec n'importe quel support numérique disposant d'un navigateur web.

Bien qu'il soit critiqué (souvent à tort), le javascript nous a semblé être le langage universel le plus adéquat pour cette solution.

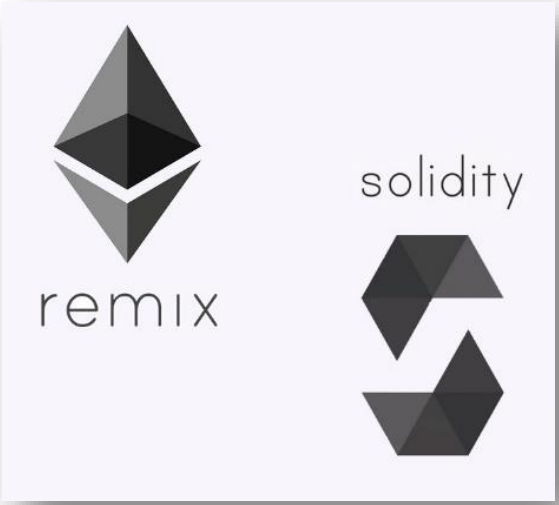


Ayant conscience que ce langage n'est pas exempt de défaut, beaucoup lui reproche sa vitesse d'exécution, que ce ne soit pas un vrai langage de programmation etc…

Nous avons jugé ces remarques pas assez pertinentes vis-à-vis aux avantages qu'il procure.

Certes, il a un indice de temps d'exécution 3x supérieur au C++, mais nous considérons que les machines ont évoluées. Et dans notre, nous n'exécutons que TRES PEU de code et à une fréquence négligeable. De plus les normes ECMASCRIPT qui régissent le javascript ne font qu'évoluer le rapprochant de plus en plus d'un langage de programmation dit « objet ».

Solidity



Solidity est un langage de programmation de type statique conçu pour développer des contrats intelligents qui s'exécutent sur l'EVM (Ethereum Virtual Machine). Solidity est compilé en bytecode lui-même exécutable sur l'EVM.

Grâce à Solidity, les développeurs sont en mesure d'écrire des applications implémentant une logique commerciale s'exécutant de manière autonome au travers des contrats intelligents, laissant une trace de transactions non répudiables et faisant autorité.

Mise en place de immoDApp:

Installation :

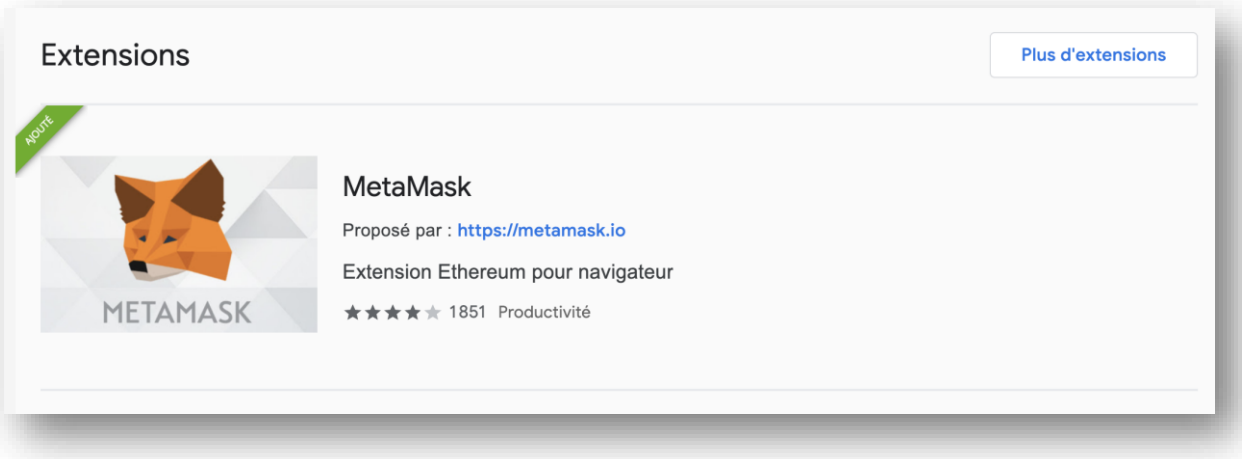
- Installation de **NodeJS** (https://nodejs.org/en/download/)

- Installer **Truffle** (Framework pour les Smarts Contracts Ethereum)

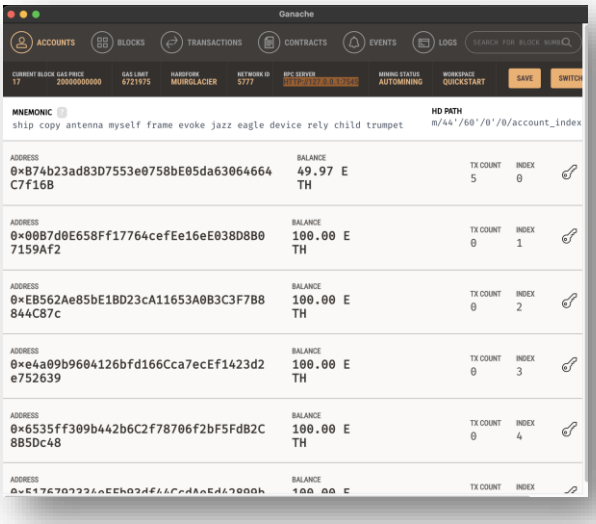
  - npm install --g truffle@5.1.39

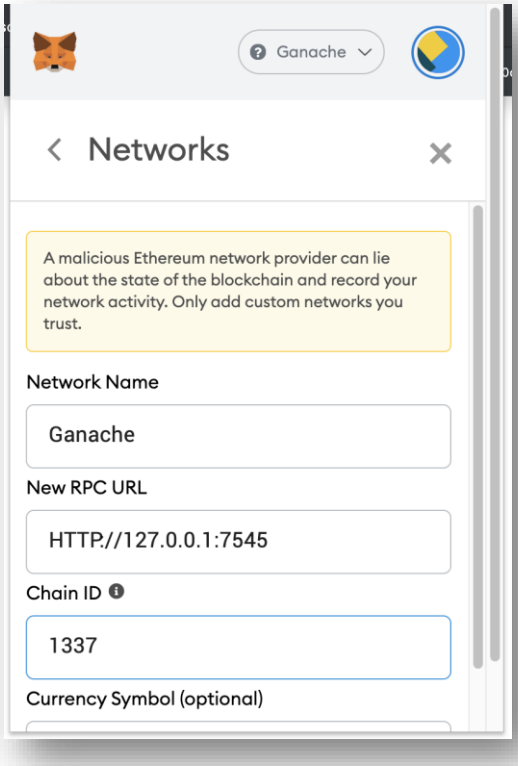
- Installer **Ganache** (Ganache est utilisé pour mettre en place une BlockChain Ethereum personnelle pour tester vos contrats Solidity. Il fournit plus de fonctionnalités par rapport à Remix.) (Https://www.trufflesuite.com/ganache)

- Installer **Metamask** (extension navigateur)



- Ouvrir **Ganache** (dans QuickStart) et Configurer **Ganache** sur **Metamask**





- Commande à exécuter : npm install

Démarrer le serveur :

- Réinitialise les migrations : truffle migrate –reset
- Démarre le serveur : npm run start
- Pour accéder à l'application, il faudra se rendre au : http://localhost:3000/