# Implémentation de la solution

# <u>Chapitre 6</u>: Présentation des applications existantes

#### 6.1 The Blockchain Insurance Industry Initiative (B3i):

Construit sur l'infrastructure existante fournie par la société de technologie blockchain R3<sup>27</sup>, qui repose elle-même sur la plateforme open source Distributed Ledger Technologie (DLT) Corda<sup>28</sup>. Le logiciel B3i permet aux compagnies d'assurance, aux courtiers et aux réassureurs d'établir des accords contraignants visibles par toutes les parties. Il permet également à chaque partie d'entamer un dialogue et de négocier le prix de la réassurance, donnant à chacune la possibilité de décider quelles informations elles rendent publiques à toutes les parties avant la décision du contrat final. Contrairement à une blockchain traditionnelle, il n'y a pas de cryptomonnaie liée au DLT de B3i, et au lieu d'un énorme réseau d'ordinateurs validant chaque transaction - les seules parties impliquées sont celles qui transfèrent activement ou prennent des risques. Selon B3i, les avantages de ceci sont la «certitude du contrat», les «coûts administratifs» et le risque d'erreurs dans le processus de transfert des risques. "Il s'agit d'une orchestration DLT de l'interaction entre les cédantes, qui sont les principaux assureurs cédant leurs risques à un panel de réassureurs via des courtiers", a ajouté Carolin. «Il enregistre chaque étape de la négociation jusqu'à la consolidation des contrats.»

Le principal client de B3i sera les compagnies d'assurance, mais la société souhaite également la commercialiser auprès d'autres acteurs du secteur, comme les start-ups insurtech, qui, selon elle, seraient en mesure de commercialiser leurs propres produits sur la base de la pile technologique de B3i. Carolin a déclaré que le produit Cat XoL est une «mise en œuvre» de la vision de l'entreprise qui montre ce qui est possible, mais pas la vision elle-même. «Nous ne sommes pas une entreprise de produits, nous construisons un système d'exploitation d'assurance conçu de manière à devenir un catalyseur pour permettre à d'autres personnes de s'appuyer sur notre pile technologique», a-t-il ajouté. «C'est une preuve pour nous de montrer que nous sommes capables de fournir ce dont nous avons besoin.» Les plates-formes existantes, notamment Synergy2 d'Eurobase et WebXL d'Effisoft, offrent déjà aux courtiers et aux réassureurs des outils pour suivre leurs activités de transfert de risques. Mais Carolin a déclaré que B3i n'avait pas l'intention de concurrencer des produits comme ceux-ci, mais après avoir prouvé les capacités de sa plate-forme DLT, il espère qu'ils deviendront des clients et s'appuieront sur elle pour améliorer le processus de réassurance.<sup>29</sup>

<sup>&</sup>lt;sup>27</sup> https://www.r3.com/

<sup>&</sup>lt;sup>28</sup> https://www.corda.net/

<sup>&</sup>lt;sup>29</sup> https://www.nsinsurance.com/news/b3i-launches-blockchain-based-technology-to-slash-reinsurance-admin-fees/

#### 6.2 <u>Fizzy</u>:

Fizzy est une police d'assurance retard de vol entièrement automatisée qui s'exécute sur la blockchain Ethereum et permet aux clients d'être indemnisés dès leur arrivée à destination. Le processus est entièrement automatisé, avec un contrat intelligent déterminant si les clients sont éligibles à l'indemnisation. Cela signifie qu'aucune action n'est requise de la part des clients éligibles pour réclamer leur indemnité.

On peut soutenir que Fizzy aurait pu se lancer en utilisant des technologies existantes, telles qu'une base de données centrale fonctionnant en combinaison avec une série d'API. Une telle implémentation fonctionnerait avec la plupart des aspects de la disposition centrée sur le client de Fizzy, tels que les réclamations et les paiements automatiques. Cependant, il manquerait l'occasion de renforcer la confiance dans la relation entre l'assureur et le preneur d'assurance en faisant en sorte que le contrat intelligent, au lieu de l'assureur, décide si le client est indemnisé.

Lorsque les clients paient en Ether, le contrat intelligent décide non seulement qu'un client est indemnisé, mais il effectue également le paiement sur le portefeuille Ether du client. Cela rendra Fizzy encore plus rapide et ajoutera davantage de confiance à la transaction.

# Chapitre 7 : Présentation de la solution

Ce référentiel contient le code d'un Smart contrat pour les assurances non vie, codé à l'aide de **Truffle** et **Solidity**. La blockchain backend est une configuration de réseau Ethereum et les interactions sont rendues possibles par la bibliothèque javascript **Web3**.

#### 7.1 Quels problèmes résolvons-nous?

Les agriculteurs sont souvent confrontés aux problèmes de dommages aux cultures en raison de causes naturelles telles que les inondations ou les sécheresses. Les agriculteurs assurés ont du mal à obtenir leurs montants de couverture auprès des compagnies d'assurance. De plus, les compagnies d'assurances jouent le rôle d'intermédiaire dans tout ce processus. Pour supprimer l'intermédiaire et régler immédiatement les réclamations d'assurance de l'utilisateur, j'ai eu l'inspiration de développer cette plateforme décentralisée.

#### 7.2 Notre solution au problème:

Agriculchain est un DApp qui donne le pouvoir de réclamer légitimement l'assurance aux agriculteurs. Les utilisateurs peuvent créer leur propre politique en spécifiant le type de culture, la superficie et l'emplacement de leur champ. Ils peuvent opter pour une assurance contre les inondations ou la sécheresse. Ils paient la prime requise au contrat intelligent. En cas de sinistre spécifié, les utilisateurs peuvent accéder à la page de réclamation, saisir leur identifiant de stratégie ainsi que la date du sinistre et lancer la transaction à partir de la même adresse Ethereum avec laquelle la stratégie a été créée. Si la réclamation est correcte, un paiement égal au montant de la couverture est initié à leur adresse. Tout cela est géré par le contrat intelligent qui vérifie si l'affirmation est vraie ou non en analysant les niveaux de précipitations à cette date via l'API fournie via le marché Honeycomb et les nœuds oracle hébergés par le réseau Chainlink. Afin de calculer le montant de la couverture, certains paramètres ont été définis dans le contrat. Selon les niveaux d'eau de la région, un paiement de 50% ou 100% est effectué en fonction de l'ampleur des dommages survenus. Les frais de paiement et de prime varient également selon le type de culture sélectionnée et augmentent linéairement avec la superficie du champ. Les utilisateurs peuvent afficher le statut des politiques ainsi que le temps jusqu'à leur validité.

## **Chapitre 8** : Implémentation

Nous configurons notre environnement de développement pour commencer à coder. Nous installerons toutes les dépendances dont nous avons besoin pour construire notre projet.

## 8.1 Prérequis

8.1.1 <u>Node Js</u> :

La première dépendance dont vous aurez besoin est Node.js, qui vous donnera Node Package Manager (NPM). Nous utiliserons NPM pour installer d'autres dépendances dans ce tutoriel. Vous pouvez vérifier si vous avez déjà installé Node en exécutant cette commande à partir de votre terminal: **\$ node -v** pour voir la version.

On peut l'installer directement depuis son site https://nodejs.org/en/

#### 8.1.2 <u>Ganache</u> :

La dépendance suivante est une blockchain de développement, qui peut être utilisée pour imiter le comportement d'une blockchain de production. Nous utiliserons Ganache comme chaîne de développement pour notre travail. Nous pouvons l'utiliser pour développer des smart contracts, créer des applications et exécuter des tests. Trouvez la dernière version de votre système d'exploitation sur <u>https://www.trufflesuite.com/ganache</u>.



Figure 8 Ganache

#### 8.1.3 <u>Truffle Framework</u> :

La prochaine dépendance est le Truffle Framework, qui nous donne une suite d'outils pour développer des applications blockchain. Cela nous permettra de développer des smart contrats, d'écrire des tests contre eux et de les déployer sur la blockchain.

Installez Truffle à partir de la ligne de commande avec NPM comme ceci (REMARQUE: vous devez utiliser cette version exacte de Truffle pour suivre ce guide): **\$ npm install -g truffle@5.0.2** 



#### Figure 9 Truffle

#### 8.1.4 <u>Metamask Ethereum Wallet</u> :

Maintenant, installons le portefeuille Metamask Ethereum afin de transformer notre navigateur Web en navigateur de blockchain. Votre navigateur Web actuel ne le prend probablement pas en charge de manière native, nous avons donc besoin de l'extension Metamask pour Mozilla (ou Google chrome) pour vous connecter à la blockchain.

Recherchez-le simplement dans la boutique en ligne de Mozilla. Lors de l'installation, assurezvous que vous l'avez activé dans votre liste d'extensions Mozilla (il doit être "vérifié"). Lorsqu'il est actif, vous verrez une icône de renard dans le coin supérieur droit de votre navigateur.



#### Figure 10 Metamask

#### 8.1.5 <u>Template</u> :

Au lieu de commencer le code à partir de zéro, je vais utiliser une template existante pour m'aider à démarrer. La template est DRIZZLE, on peut cloner ce modèle depuis github comme ceci:

#### \$ git clone <u>https://github.com/truffle-box/drizzle-box.git</u>

Cette boîte contient tout ce dont vous avez besoin pour commencer à utiliser des contrats intelligents à partir d'une application react avec Drizzle. Il comprend des composants drizzle, drizzle-react et drizzle-react-components pour vous donner un aperçu complet des capacités de Drizzle.

## 8.2 Configuration du projet

Maintenant que nous avons installé toutes les dépendances dont nous avons besoin, construisons notre application blockchain!

8.2.1 <u>Configuration de Ganache</u> :

Après avoir installé Ganache, ouvrez-le. Une fois qu'il est chargé, vous avez une blockchain en cours d'exécution sur votre ordinateur!

- Conquer sur new workspace

  Constant

  Constant
- Cliquer sur new workspace

#### Figure 11 Ganache start.

• Mettre un nom et sélectionner le fichier truffle-config.js dans le projet

WORKSPACE	
NORKSPACE NAME	A friendly name for this workspace.
TRUFFLE PROJECTS C:\testing\test2\Insurance-Ethereum\truffle-config.js	Link Truffle projects to this workspace by adding their truffle- config.js or truffle.js file to this workspace. This will show useful contract and event data to better understand what's going on under the hood.
ADD PROJECT REMOVE PROJECT	

Figure 12 Ganache workspace

ACCOUNTS ( BLOCKS ( TRANSACTIONS ) CONTRAIN	CTS () EVENTS () LOGS I ANNO AN			
CURRENT BLOCK GAS PRICE GAS LIMIT HABDFORK NETWORK ID RPC SEA 8 20000000000 6721975 PETERSBURG 5777 HTTP:	VER MINING STATUS WORKSPACE (127.0.0.1:7545 AUTOMINING INSURANCE-ETHEREUM		SWITCH	0
MNEMONIC bronze obscure rigid universe lady token fiction illness ch	imney heavy wet kind	HD PATH m/44'/60'/0	'/0/account	_index
ADDRESS	BALANCE	TX COUNT	INDEX	S
0×2bF6f0Dd7e844d751e94CEfD5441214EDd2a90D6	99.94 ETH	8	0	
ADDRESS	BALANCE	TX COUNT	INDEX	S
0×21481C0669b75a6b91c8ffc7f329205e0b1C94e8	100.00 ETH	O	1	
ADDRESS	BALANCE	TX COUNT.	INDEX	J
0×13d438555560BE16c0D9b8207337aE8b12EA1234	100.00 ETH	O	2	
ADDRESS	BALANCE	TX COUNT	INDEX	S
0×E6EfC49E6427D295aD3a29a3C5471a743A7986CA	100.00 ETH	Ø	3	
ADDRESS	BALANCE	TX COUNT	INDEX	F
0×A9817da11A065984b09dCC05E8d427dB2fEec0A8	100.00 ETH	Ø	4	
ADDRESS	BALANCE	TX COUNT	INDEX	F
0×FD5f6c4A6Ccd136D3501eef73d71C82c0304342a	100.00 ETH	Ø	5	

Nous avons 10 comptes fournis par Ganache, chacun pré-extrait avec 100 faux Ether (Cet Ether ne vaut pas de l'argent réel).

Figure 13 Ganache Accounts

kongeneration.			
0+25+01-0127-03-607-510	ACCOUNT INFORMATION	1	
0.214815865967448691	ACCOUNT ADDRESS 0×2bF6f0Dd7e844d751e94CEfD5441214EDd2a90D6	-	
01130/458555569881508	PRIVATE KEY c43e37a0c7a73149bd4f6cc3a2b650ac5df04b26828ddd962dab9ebadd2ae9e1 Do not use this private key on a public blockchain; use it for development purposes only!	-	
ONTAR FRAVERS 170235a0,	DONE		
1-89517de11-085984based			
D=TDSF8c446ccd118018881	ex+173471182283043-22 (0.8.00 #16)		

Chaque compte possède une paire de clés publique et privée unique. Vous pouvez voir l'adresse de chaque compte sur l'écran principal ici. Les comptes ressemblent beaucoup à des "noms d'utilisateur" sur la blockchain et qu'ils représentent chaque utilisateur qui peut publier sur notre réseau social.

Figure 14 Account information

#### 8.2.2 <u>Configuration de Metamask</u> :

Ensuite, nous importons nos comptes de Ganache dans Metamask.

• Pour commencer nous allons connecter Metamask au réseau de Ganache HTTP://127.0.0.1:7545



Figure 15 Réseau d'ethereum

• Ensuite Nous allons importer 2 comptes sur Metamask

Pour commencer on clique sur importer un compte



Figure 16 Comptes metamask

-	• .	,	,	1 1/	• • •		1	1 11		\ <i>T</i> / 1	. 1*		•
Hncu	110 1	raciur	oror		nr1170	0 0111	annacha	la collar	CI11 CI	Vlotomodz	of clic	11101 1011	· importor
EMISU	ILC I	UUUI		IA UIU		c sur	yanacılı.		SULL	VICTATITASN.		iuci sui	ппполіст
					· · · · · ·	• • •••	B					1	

METAMASK		( ● HTTP://127.0.0.1:7545 ∨ )
	Créer Importer Connecter	
	Les comptes importés ne seront pas associés avec voi phrase Seed que vous avez créé au départ dans MetaMask. Obtenir plus d'information sur les comptes importés <u>ici</u>	tre
	Selectionner le type Clé privée	-
	Collez votre clé privée ici:	
		4
	Annuler	

Figure 17 Importation de la clé privéé

	₩ TOOLIII	Deposer
Account 7	Historique	
Détails	Au	cune transaction
0x0b83cb7E 🖪		

#### Cliquer sur détails

Figure 18 Account



Ensuite cliquer sur account 7 pour changer le nom

Figure 19 Edit account



Figure 20 Compte créé

#### 8.2.3 <u>Configuration</u> :

Créons maintenant un nouveau projet pour tout notre code d'application. Au lieu de le faire à partir de zéro, je vais utiliser un modèle que j'ai créé pour nous aider à démarrer rapidement. Vous pouvez cloner ce modèle depuis github comme ceci:

#### \$ git clone https://github.com/truffle-box/webpack-box.git

Maintenant, vous pouvez entrer dans le dossier du projet nouvellement créé comme ceci:

**\$ Cd social-network** (sur PowerShell ou invite de commande)

On ouvre le projet dans l'éditeur de texte de votre choix et recherchez le fichier truffle-config.js. C'est là que nous allons stocker tous les paramètres de notre projet Truffle.

Il est connecté à ganache par le port 7545



Figure 21 Truffle Config

Voici une liste complète de tous les fichiers de notre projet:

- Répertoire des migrations: c'est là que seront les fichiers de migration qui nous permettront de mettre de nouveaux contrats intelligents sur la blockchain.
- Répertoire node\_modules: c'est là que toutes nos dépendances sont installées pour le projet.
- Répertoire app: il s'agit du dossier principal de notre site Web côté client
- Répertoire ./app/src: c'est ici que nous développerons tous les composants React.js qui alimentent notre site Web côté client.
- Répertoire contracts: c'est le dossier où nous développerons le code source de nos contrats intelligents avec Solidity.

Ensuite, regardons le fichier package.json.

Ce fichier contient toutes les dépendances du projet dont nous avons besoin pour créer l'application. J'ai inclus toutes les dépendances dont nous avons besoin dans ce modèle de kit de démarrage.

Allons de l'avant et installons ces dépendances comme ceci:

#### **\$ npm install**

Z Administrateur: Windows PowerShell -	٥	×
PS C:\testimg\test2\Insururance-Ethereum> cd .\app\		
PS C:\testimg\test2\Insururance-Ethereum\app> npm install		
npm rotice created a lockfile as package-lock.json. You should commit this file.		
npm 🚧 👯 tsutils@3.17.1 requires a peer of typescript@>=2.8.0    >= 3.2.0-dev    >= 3.3.0-dev    >= 3.4.0-dev    >= 3.5.0-dev    >= 3.6.0-dev    >= 3.6.0-beta    >= 3.7.0-dev    >= 3.7.0-de	7.0-be	eta
but none is installed. You must install peer dependencies yourself.		
npm WARN SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\jest-haste-map\node_modules\fsevents):		
npm MARN SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})		
npm WARN SKIPPING OPTIONAL DEPENDENCY: fsevents@2.0.7 (node_modules\fsevents):		
npm MARN SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.0.7: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})		
npm <mark>WARN</mark> SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\chokidar\node_modules\fsevents):		
npm 🔐 ARM SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})		
removed 12 packages and audited 928183 packages in 66.474s		
29 packages are looking for funding		
run `npm fund` for details		

Figure 22 Npm installation

### 8.3 Smart contract

Create New Policy		and and	Sec. 10. 1
Crop Type:			
Rabi			
O Kharif			
Area:			
320			
Policy For:			
O Flood			
Drought			
Location:			
New York		X 7.35	
Premium to Pay (in wei) :			
640			
	CREATE		19

#### Figure 23 Créer une nouvelle politique

La figure ci-dessus montre comment créez une nouvelle police en payant le montant de prime requis

Create New Policy	MetaMask Notification - 🗆 🗙	
cheate frem i ency	Ropsten Test Network	
Сгор Туре:	OWNER →	
Rabi	CONTRACT INTERACTION	
O Kharif	<b>\$</b> 0	
Area:		
320	DETAILS DATA	10
Policy For:	GAS FEE \$0.01	
O Flood	No Conversion Rate Available	
Drought	AMOUNT + GAS FEE	A CONTRACTOR OF A CONTRACTOR A
Location:	TOTAL \$0.01 No Conversion Rate Available	1 CUT
New York		
Premium to Pay (in wei) :		
640	Reject	
	*	

Figure 24 Payez la prime via le portefeuille Ethereum de Metamask.

Den Ado Contra	Claim Insurance	
	Date of Flood/Drought:	
	07-Nov-2019	
	Current Metamask ETH Address:	
the first state	0x4E04768CDD20e35EE87e6a89fC5B920f9492ffC5	
	(Login with same Address with which Policy Created)	
	Your Policy ID:	
2	3	٥
		CLAIM
A Part Bas	·	
0/00		

**Figure 25** Réclamez une assurance en cas de tragédies comme les inondations ou la sécheresse.

view your Policies				
nter Policy ID:				
ц				
- VIEW-				
User Address	Area	Сгор	Туре	Valid Till
0x4E04768CDD20e35EE87e6a89fC5B920f9492ffC5	543	Kharif	Flood	06-June-2020

Figure 26 Afficher les détails de votre politique