

- ▶ Aller sur : <https://github.com/benjaminfontaine/codelab-ethereum>
- ▶ Lancer le téléchargement et l'installation des outils spécifiés dans le README.md



CODELAB

ETHEREUM





Benjamin Fontaine,
ingénieur fullstack,
passionné par la
blockchain et
investisseur Ethereum



Benjamin Fontaine,
ingénieur fullstack,
passionné par la
blockchain et
investisseur Ethereum





Benjamin Fontaine,
ingénieur fullstack,
passionné par la
blockchain et
investisseur Ethereum

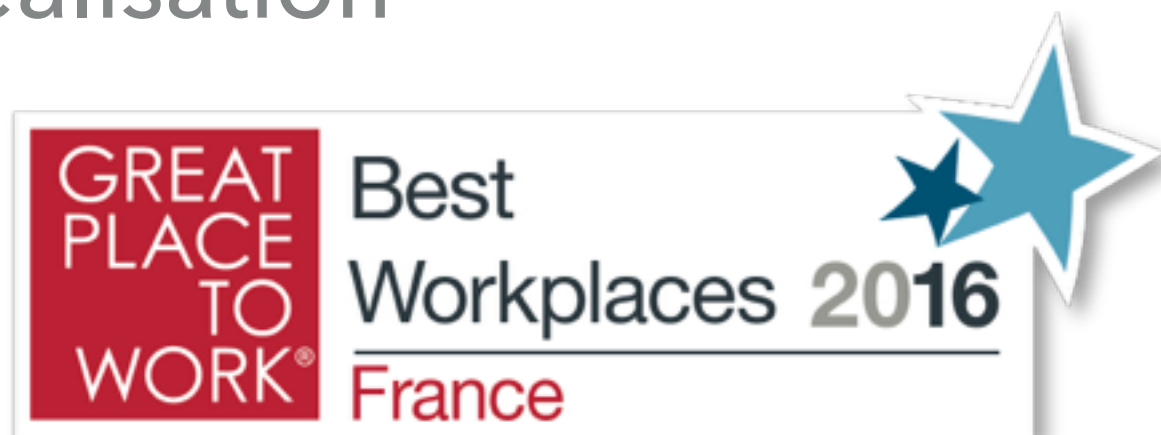
Conseil, formation et
réalisation





Benjamin Fontaine,
ingénieur fullstack,
passionné par la
blockchain et
investisseur Ethereum

Conseil, formation et
réalisation



C'EST QUOI ?

LA BLOCKCHAIN

LE BITCOIN, LA PREMIÈRE MONNAIE VIRTUELLE...



- ▶ Créée en 2009 par Satoshi Nakamoto



- ▶ Virtuelle

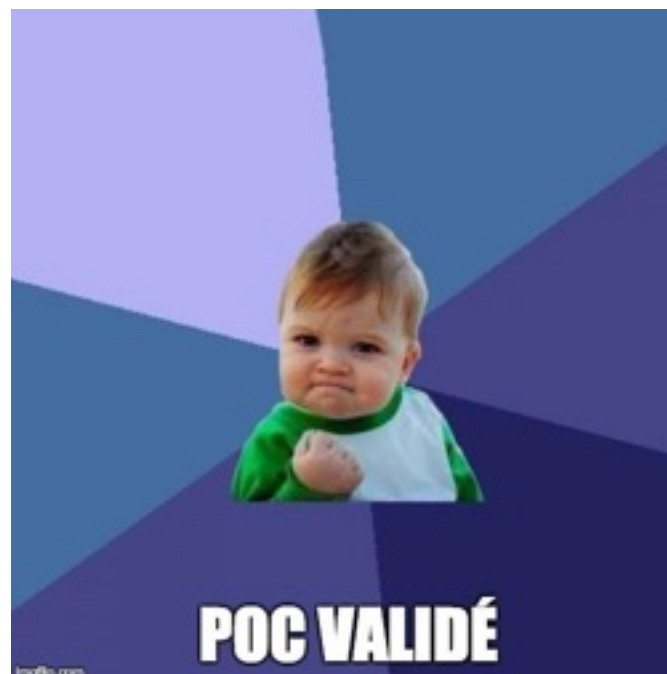
- ▶ Mais avec une vraie valeur marchande car acceptée chez de nombreux commerçants et tradée

MAIS QUI FONCTIONNE À MERVEILLE

MAIS QUI FONCTIONNE À MERVEILLE



MAIS QUI FONCTIONNE À MERVEILLE



ET LA BLOCKCHAIN DANS TOUT ÇA ?



- ▶ Répond aux problématiques suivantes :
 - ▶ Désintermédiation : Échange des informations directement entre utilisateurs
 - ▶ Traçabilité : Empêcher que l'on donne deux fois le même bitcoin
 - ▶ Consensus distribué : Garantir la monnaie sans autorité centralisée

ET LA BLOCKCHAIN DANS TOUT ÇA ?



- ▶ Répond aux problématiques suivantes :
 - ▶ Désintermédiation : Échange des informations directement entre utilisateurs
➡ **Echange peer-to-peer**
 - ▶ Traçabilité : Empêcher que l'on donne deux fois le même bitcoin
 - ▶ Consensus distribué : Garantir la monnaie sans autorité centralisée

ET LA BLOCKCHAIN DANS TOUT ÇA ?



- ▶ Répond aux problématiques suivantes :
 - ▶ Désintermédiation : Échange des informations directement entre utilisateurs
 - ➡ **Echange peer-to-peer**
 - ▶ Traçabilité : Empêcher que l'on donne deux fois le même bitcoin
 - ➡ **Garde les traces de transaction dans un registre**
 - ▶ Consensus distribué : Garantir la monnaie sans autorité centralisée

ET LA BLOCKCHAIN DANS TOUT ÇA ?



- ▶ Répond aux problématiques suivantes :
 - ▶ Désintermédiation : Échange des informations directement entre utilisateurs
 - ➡ **Echange peer-to-peer**
 - ▶ Traçabilité : Empêcher que l'on donne deux fois le même bitcoin
 - ➡ **Garde les traces de transaction dans un registre**
 - ▶ Consensus distribué : Garantir la monnaie sans autorité centralisée
 - ➡ **Ce registre est partagé et vérifiable par tous**

QU'EST-CE QUE LA BLOCKCHAIN ?

REGISTRE ou LEDGER

Ledger		
From	To	Amt
Bill	Alice	15
Jon	Ann	3
Bob	Ryan	30

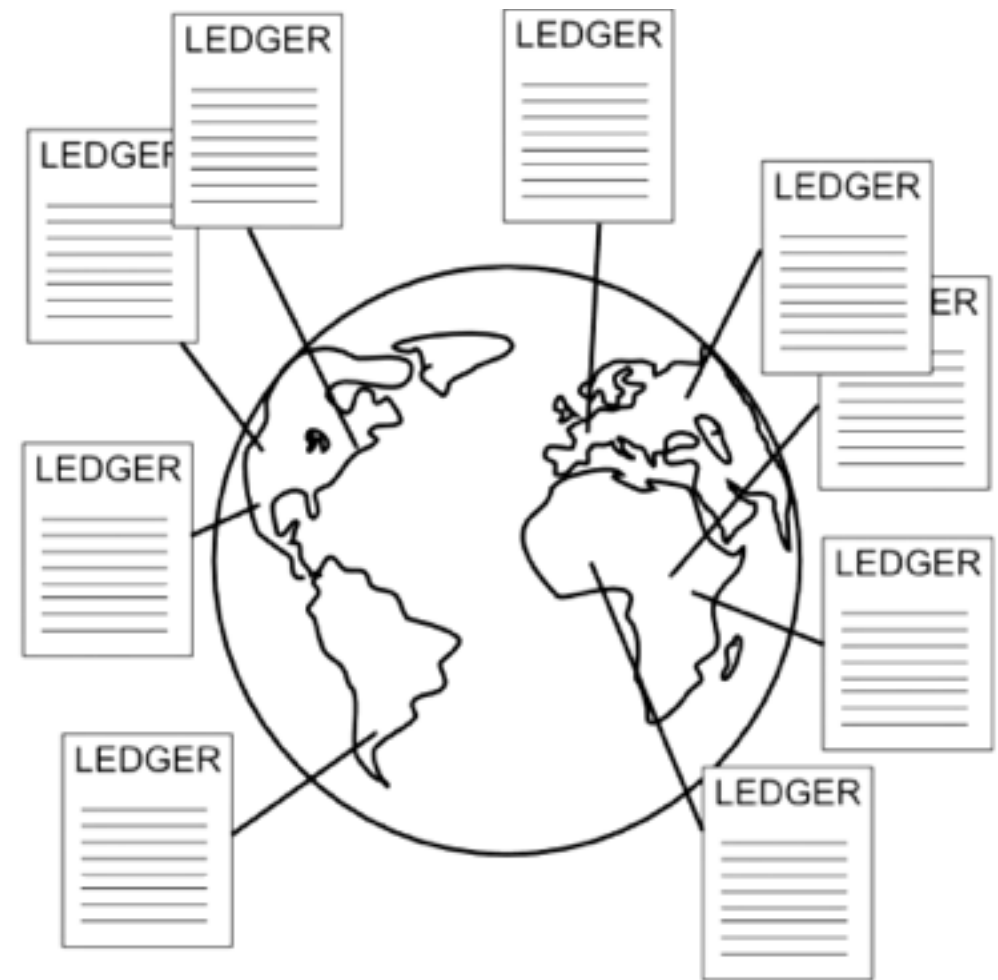
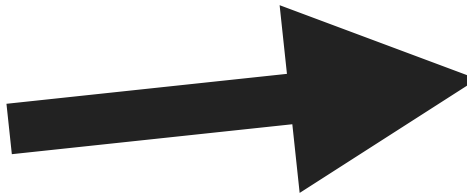
Unverified		
From	To	Amt
Alice	Bob	10

QU'EST-CE QUE LA BLOCKCHAIN ?

REGISTRE ou LEDGER

Ledger		
From	To	Amt
Bill	Alice	15
Jon	Ann	3
Bob	Ryan	30

Unverified		
From	To	Amt
Alice	Bob	10



Distribué et consultable par tous

QU'EST-CE QUE LA BLOCKCHAIN ?

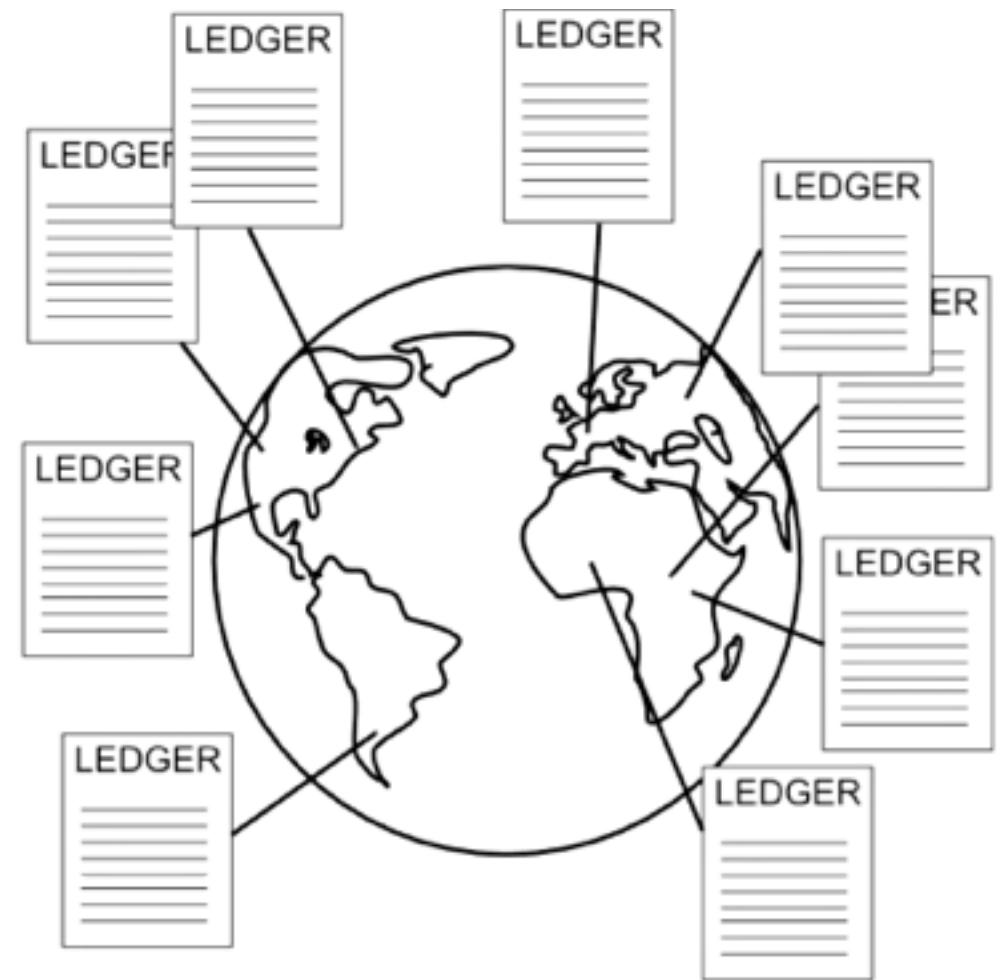
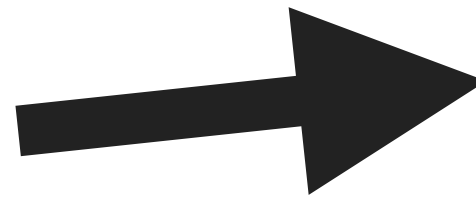
REGISTRE ou LEDGER

Ledger		
From	To	Amt
Bill	Alice	15
Jon	Ann	3
Bob	Ryan	30

Unverified		
From	To	Amt
Alice	Bob	10



**Données introduites
par consensus**



Distribué et consultable par tous

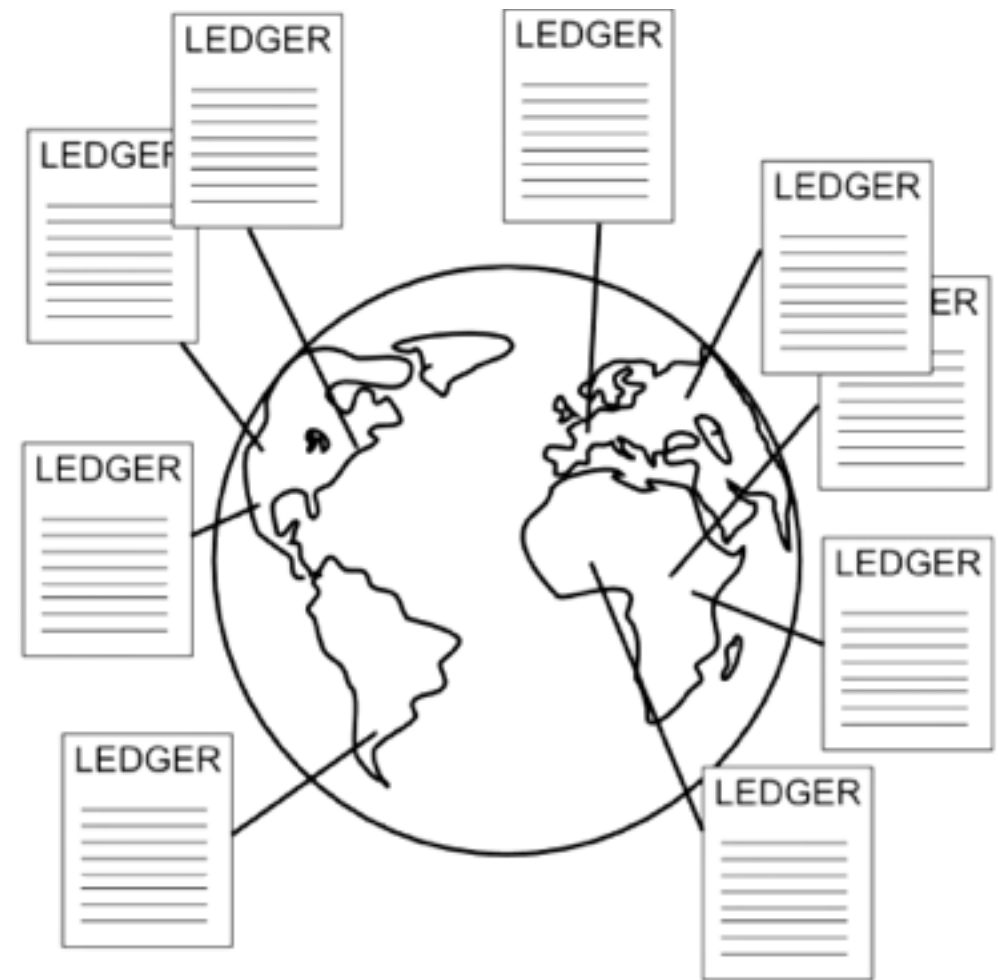
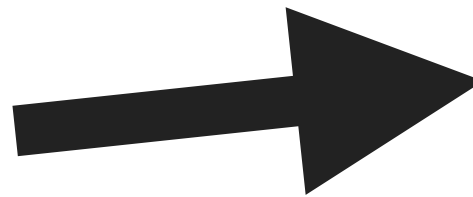
QU'EST-CE QUE LA BLOCKCHAIN ?

REGISTRE ou LEDGER

Ledger			Unverified		
From	To	Amt	From	To	Amt
Bill	Alice	15	Alice	Bob	10
Jon	Ann	3			
Bob	Ryan	30			



**Données introduites
par consensus**

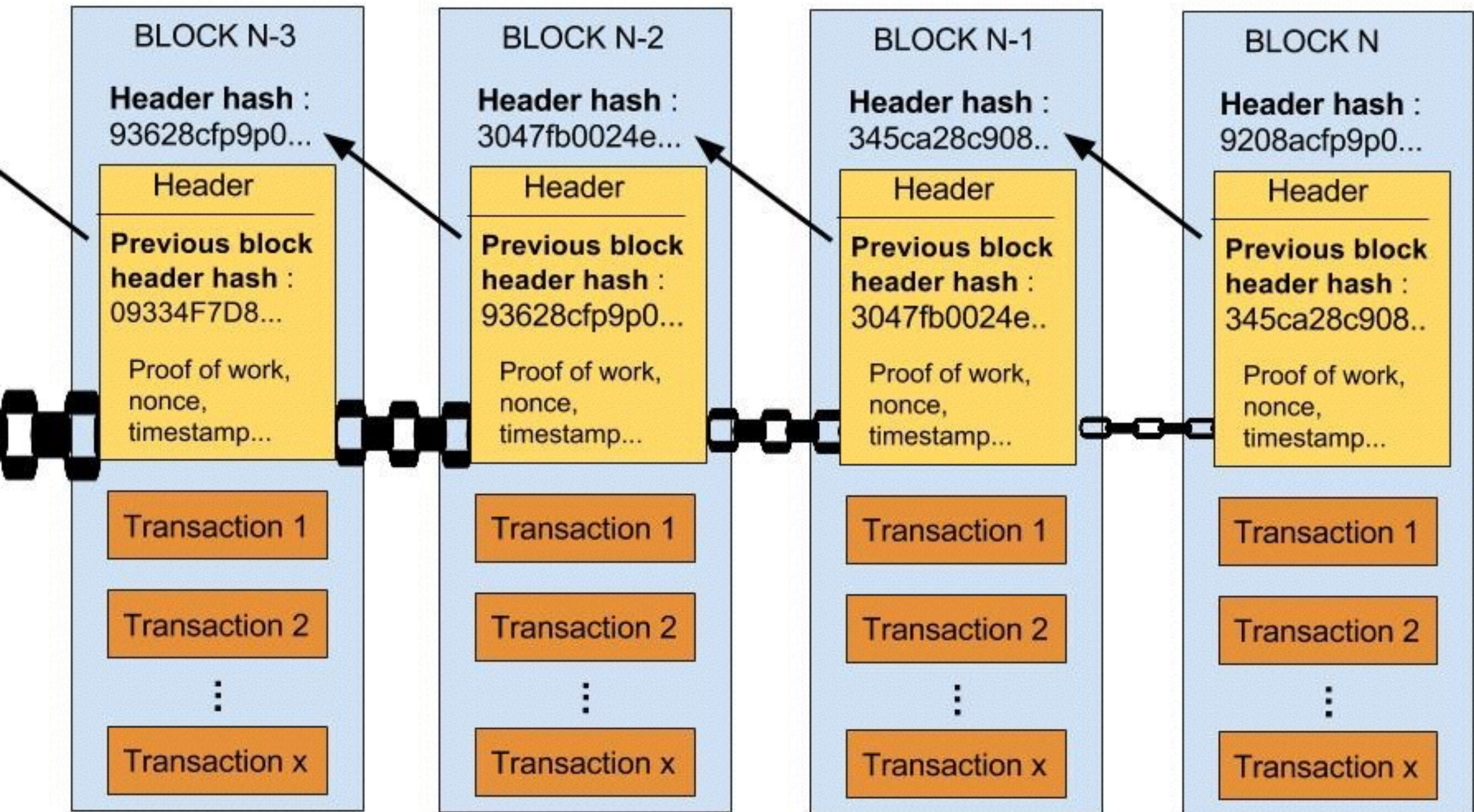


Distribué et consultable par tous



Sécurisé et fiable

STRUCTURE DU REGISTRE



ETHEREUM

POURQUOI ?

En 2014, un constat :

De + et + d'utilisations détournées du réseau bitcoin
(alter coins, dns, stockage de fichier) grâce à du
scripting et à des détournements du protocole

LES LIMITES DU SCRIPTING BITCOIN

Le langage de script du bitcoin trop rudimentaire (pas de boucles par exemple)

Pas de notion d'état intermédiaire sur les transactions (spend ou unspend)

Pas de possibilité de diviser le montant de la transaction (le script peut juste verser l'intégralité du montant sur une adresse donnée)

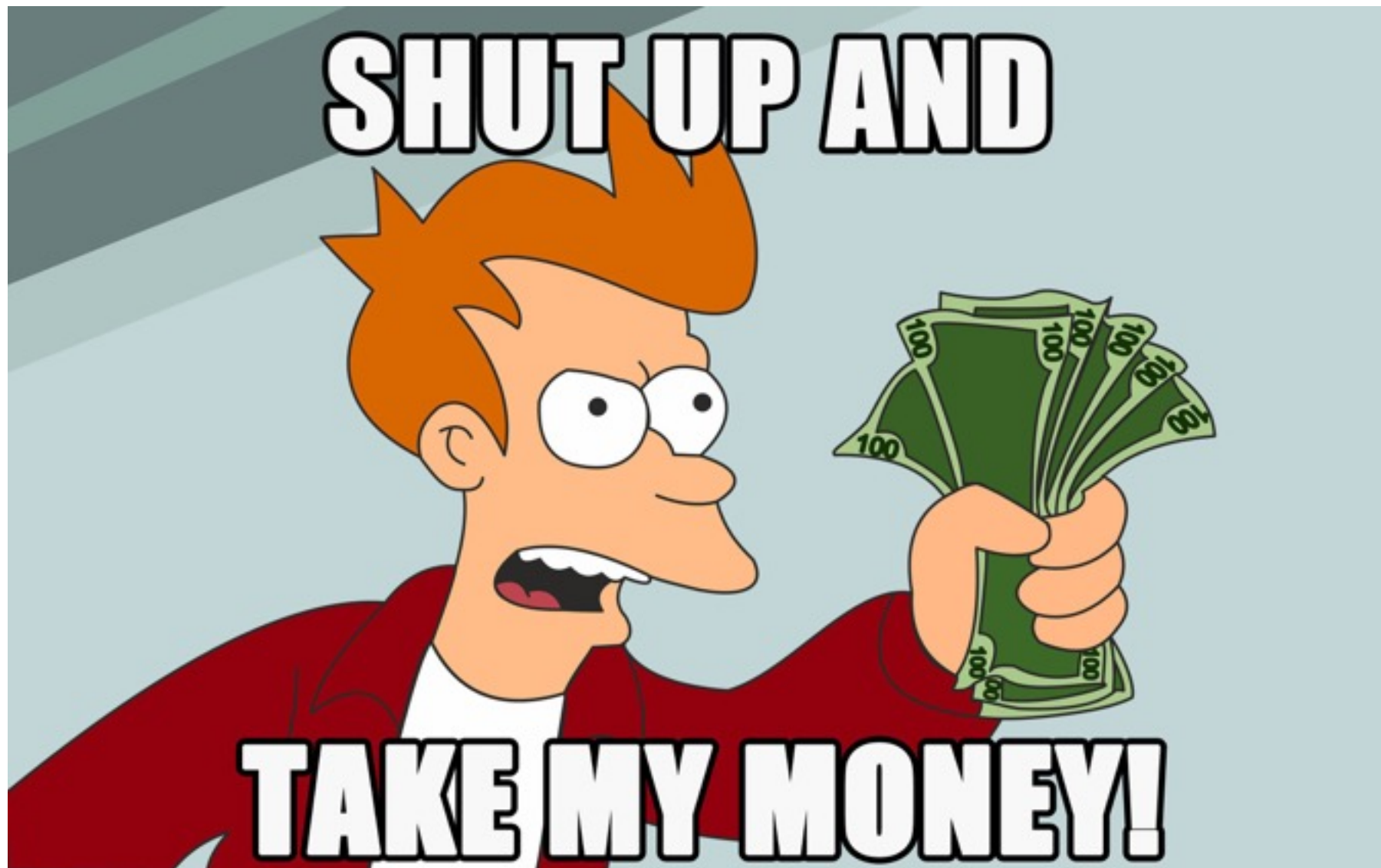
Pas d'accès à toutes les données du block (nonce, previous block hash)

Bref : un casse-tête pour les développeurs

LE CONCEPT D'ETHEREUM

une blockchain 2.0, améliorée de manière à pouvoir stocker du code dans la blockchain et qui ouvre une infinité de possibilité aux développeurs.

Un kickstarter plus tard...



Récolte 18 millions de dollars

LES CONTRATS INTELLIGENTS

Des programmes autonomes qui exécutent automatiquement les conditions et termes d'un contrat, sans permettre d'intervention humaine une fois créés

Capables d'interagir avec d'autres comptes utilisateurs et smart contracts

Code as Law

LES COMPTES ET LES CONTRACTS



- **Adresse** : 160 bits extraits de la clé publique
- **Nonce** : compteur de transactions
- **Solde** : nb d'ether possédé

LES COMPTES ET LES CONTRACTS



- **Adresse** : 160 bits extraits de la clé publique
- **Nonce** : compteur de transactions
- **Solde** : nb d'ether possédé



- **Adresse** : 160 bits extraits de la clé publique
- **Nonce** : compteur de transactions
- **Solde** : nb d'ether possédé
- **Code** : code du smart contract
- **Stockage** : espace de stockage dédié au contrat

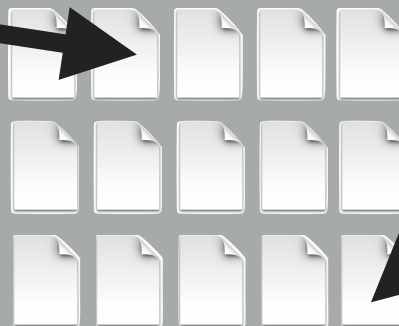
LES COMPTES ET LES CONTRACTS



- **Adresse** : 160 bits extraits de la clé publique
- **Nonce** : compteur de transactions
- **Solde** : nb d'éther possédé

BLOCK

Etat



Transactions

TX 1

TX 2

TX 3

TX N



- **Adresse** : 160 bits extraits de la clé publique
- **Nonce** : compteur de transactions
- **Solde** : nb d'éther possédé
- **Code** : code du smart contract
- **Stockage** : espace de stockage dédié au contrat

LES DIFFERENCES AVEC LA BLOCKCHAIN DU BITCOIN

- ▶ Chaque block contient un ensemble de transactions **et l'état de l'intégralité de la blockchain**
- ▶ Temps d'insertion d'un block réduit à 12 secondes
- ▶ Génération de monnaie constante et infinie
- ▶ Notion de comptes utilisateurs et de smart-contracts
- ▶ Cout de transaction calculé en fonction de la complexité
- ▶ Langages de développement très élaborés pour décrire ce smart contracts
- ▶ Décourage le mining centralisé et spécialisé

UN TIERS DE CONFIANCE DÉCENTRALISÉ



Source : <http://fr.artquid.com>

Tiers de confiance : banques, état, notaires, partenaires de paiement...

UN TIERS DE CONFIANCE DÉCENTRALISÉ



Source : <http://fr.artquid.com>

UN TIERS DE CONFIANCE DÉCENTRALISÉ

UN TIERS DE CONFIANCE DÉCENTRALISÉ



UN POTENTIEL DISRUPTIF ÉNORME

► Pourrait à terme remplacer :

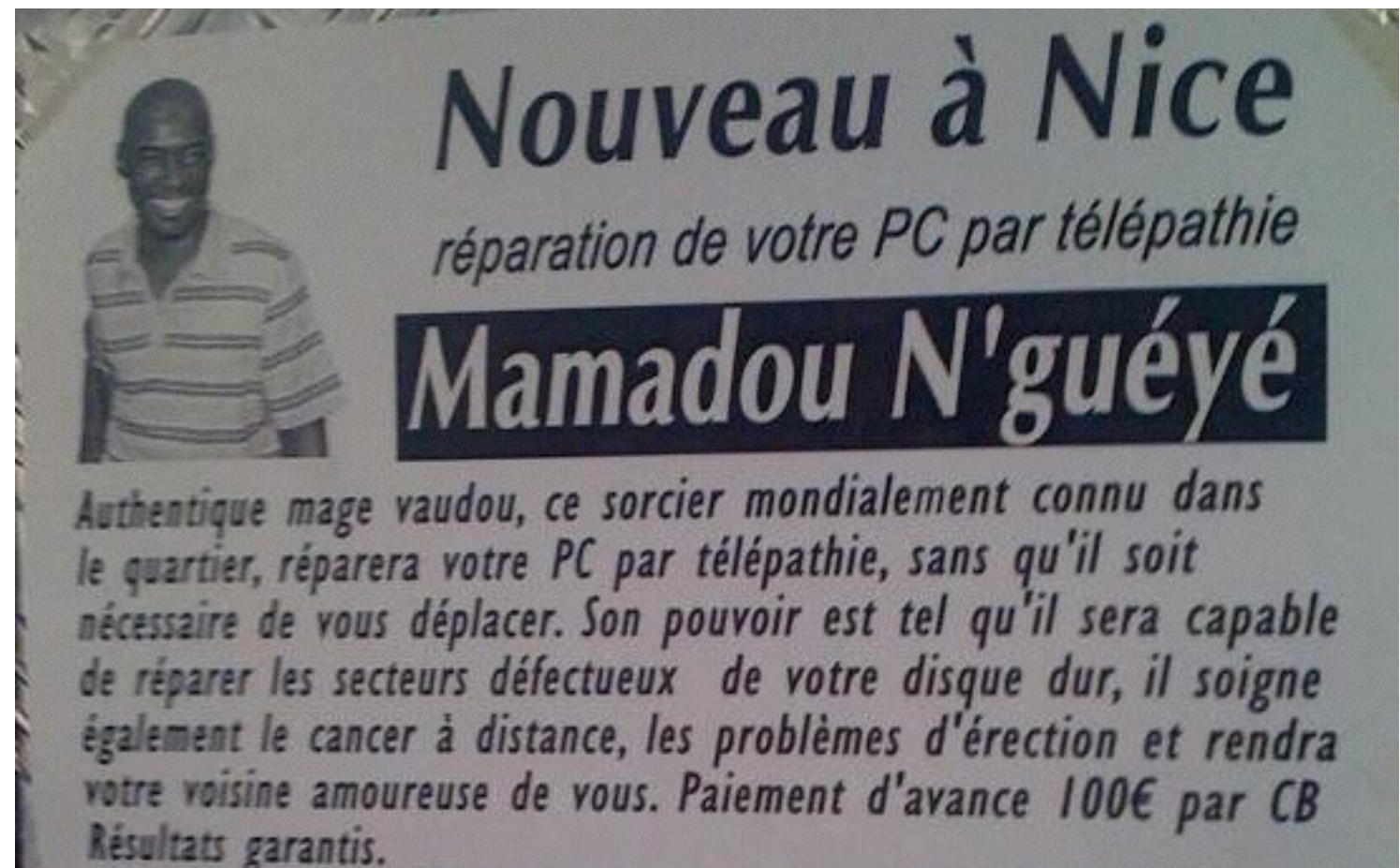
- **les banques** : les utilisateurs d'une blockchain mettraient leur argent dans un fond commun, et un ensemble de contrats régirait les dépôts, retraits et prêts;
- **les assurances** : des assurances peer-to-peer où les utilisateurs verseraient leurs cotisations dans un fond et décideraient entre eux des indemnisation, indemnisations automatiquement débloquées par des smart contracts
- **les notaires** : les actes de vente, les testaments et tout les documents certifiés pourraient être stockés sur une blockchain
- les services de paiement, les maisons de disque, uber et airbnb, les **bookmakers**

ET DE NOUVEAU USAGES

- ▶ Rendre disponibles des services bancaires n'importe où dans le (tiers) monde
- ▶ Révolutionner l'économie de partage (slock it, partage d'énergie)
- ▶ Des prédictions sur le futur
avec Augur

ET DE NOUVEAU USAGES

- ▶ Rendre disponibles des services bancaires n'importe où dans le (tiers) monde
- ▶ Révolutionner l'économie de partage (slock it, partage d'énergie)
- ▶ Des prédictions sur le futur avec Augur



Nouveau à Nice
réparation de votre PC par télépathie

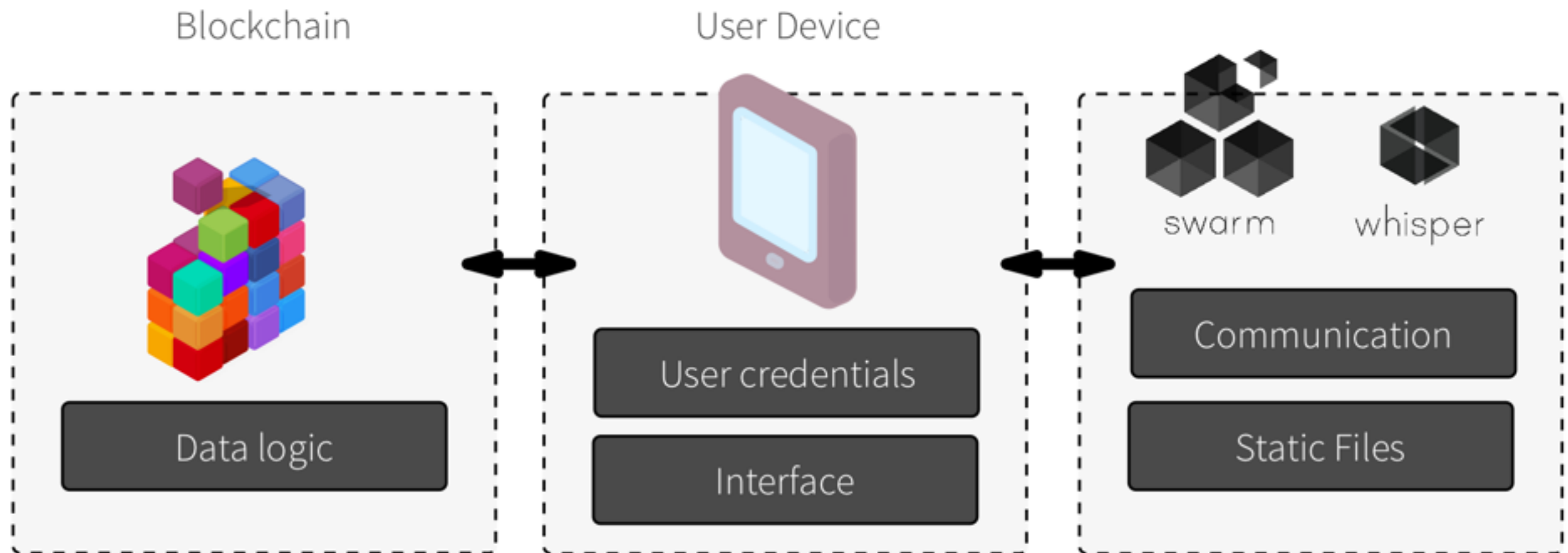
Mamadou N'guéyé

Authentique mage vaudou, ce sorcier mondialement connu dans le quartier, réparera votre PC par télépathie, sans qu'il soit nécessaire de vous déplacer. Son pouvoir est tel qu'il sera capable de réparer les secteurs défectueux de votre disque dur, il soigne également le cancer à distance, les problèmes d'érection et rendra votre voisine amoureuse de vous. Paiement d'avance 100€ par CB Résultats garantis.

ETHEREUM

NOTRE D-APP

ARCHI D'UNE APP ETHEREUM



Source : Ethereum blog (<https://blog.ethereum.org>)

NOTRE D-APPS: MON TIERCE



Notre appli Angular 2

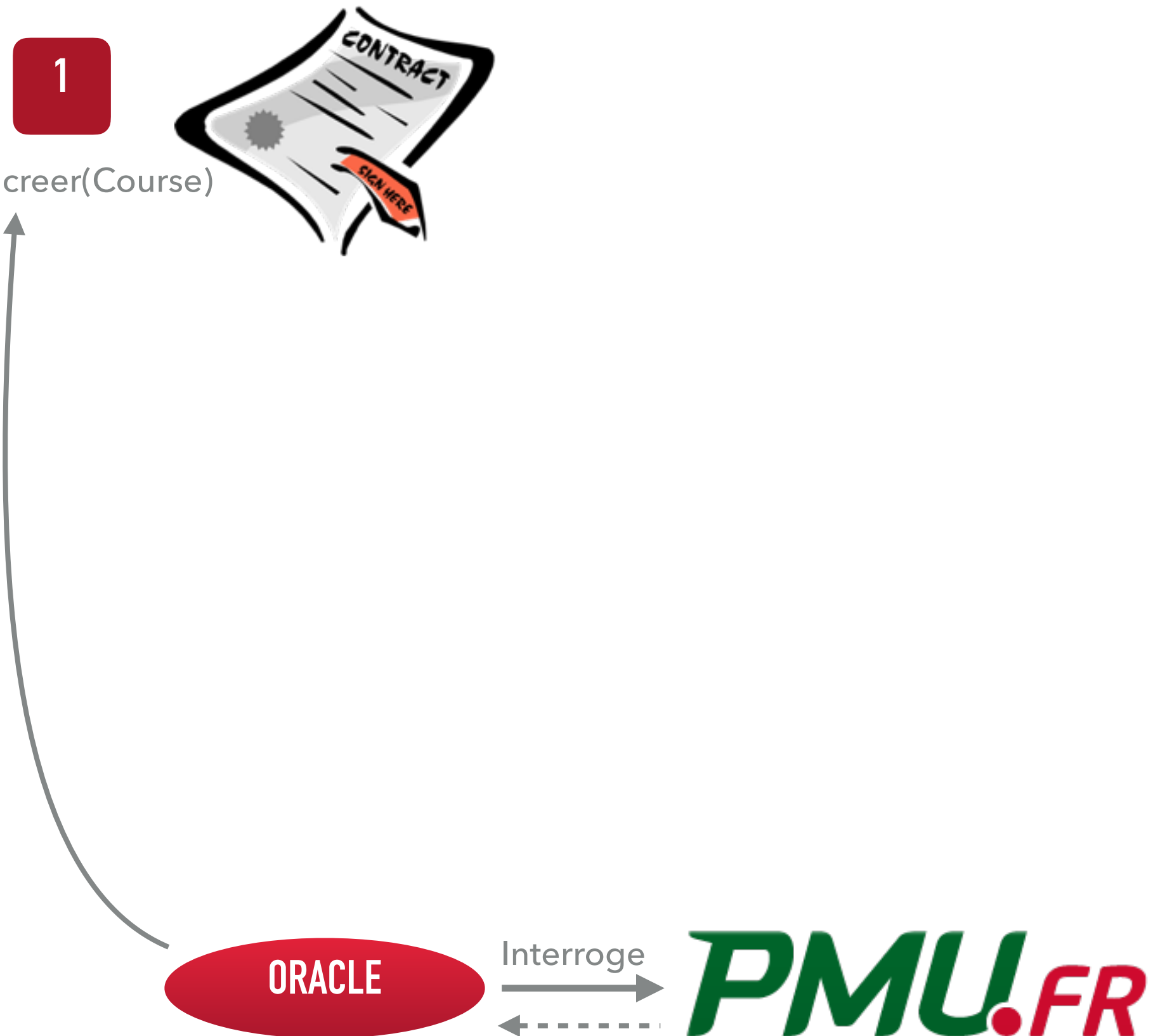


Interroge
→
←

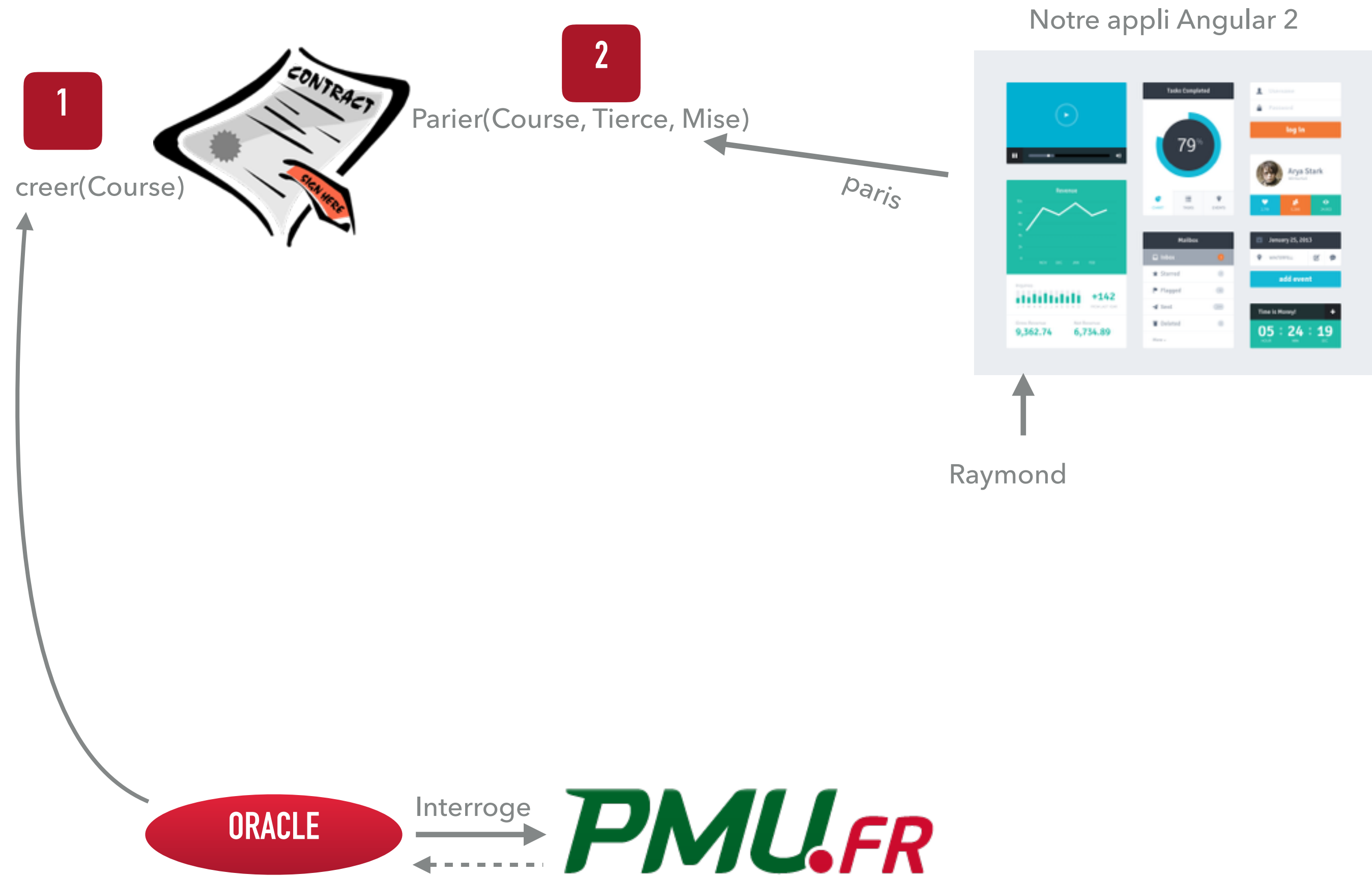
PMU.FR

NOTRE D-APPS: MON TIERCE

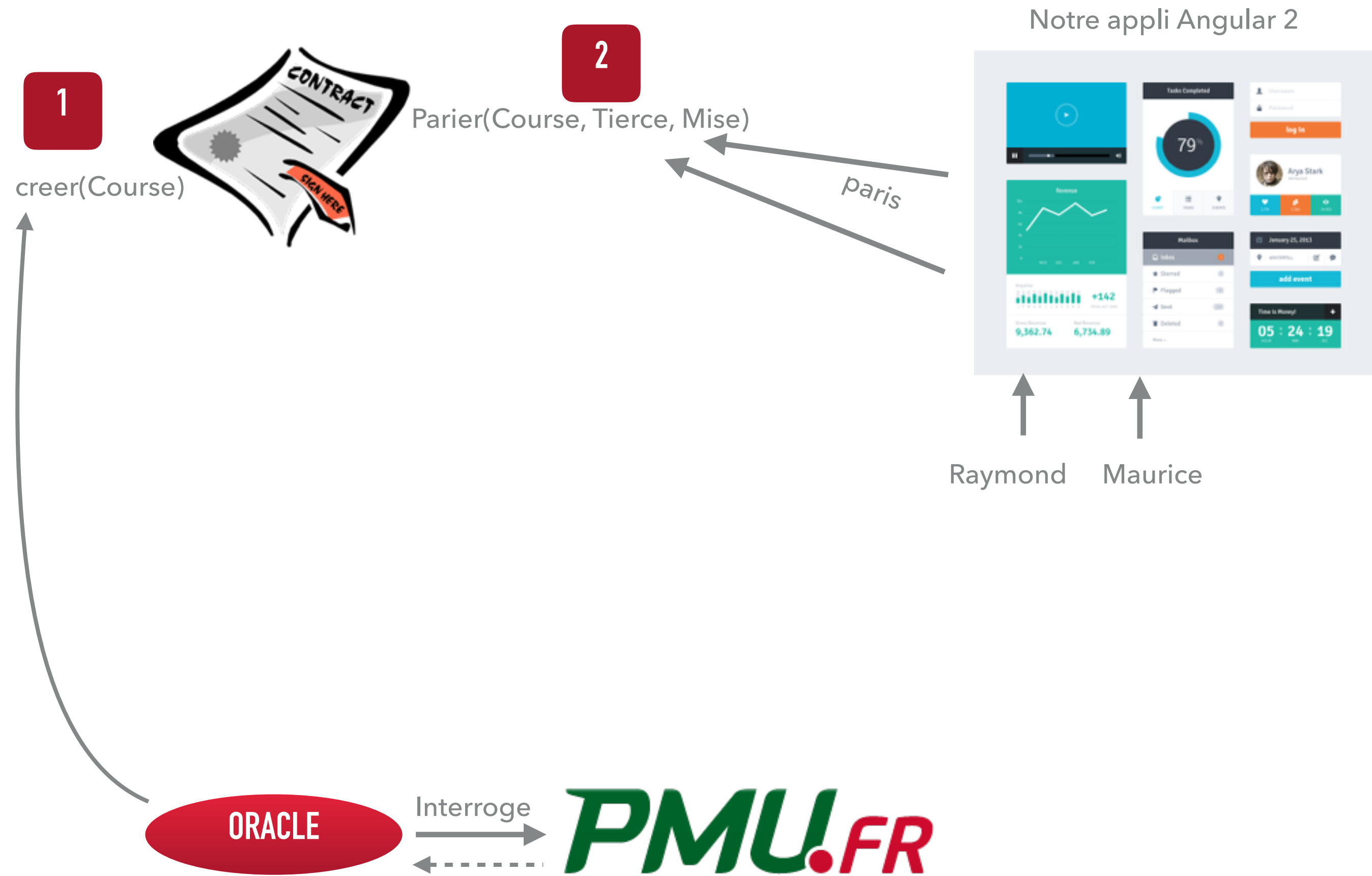
Notre appli Angular 2



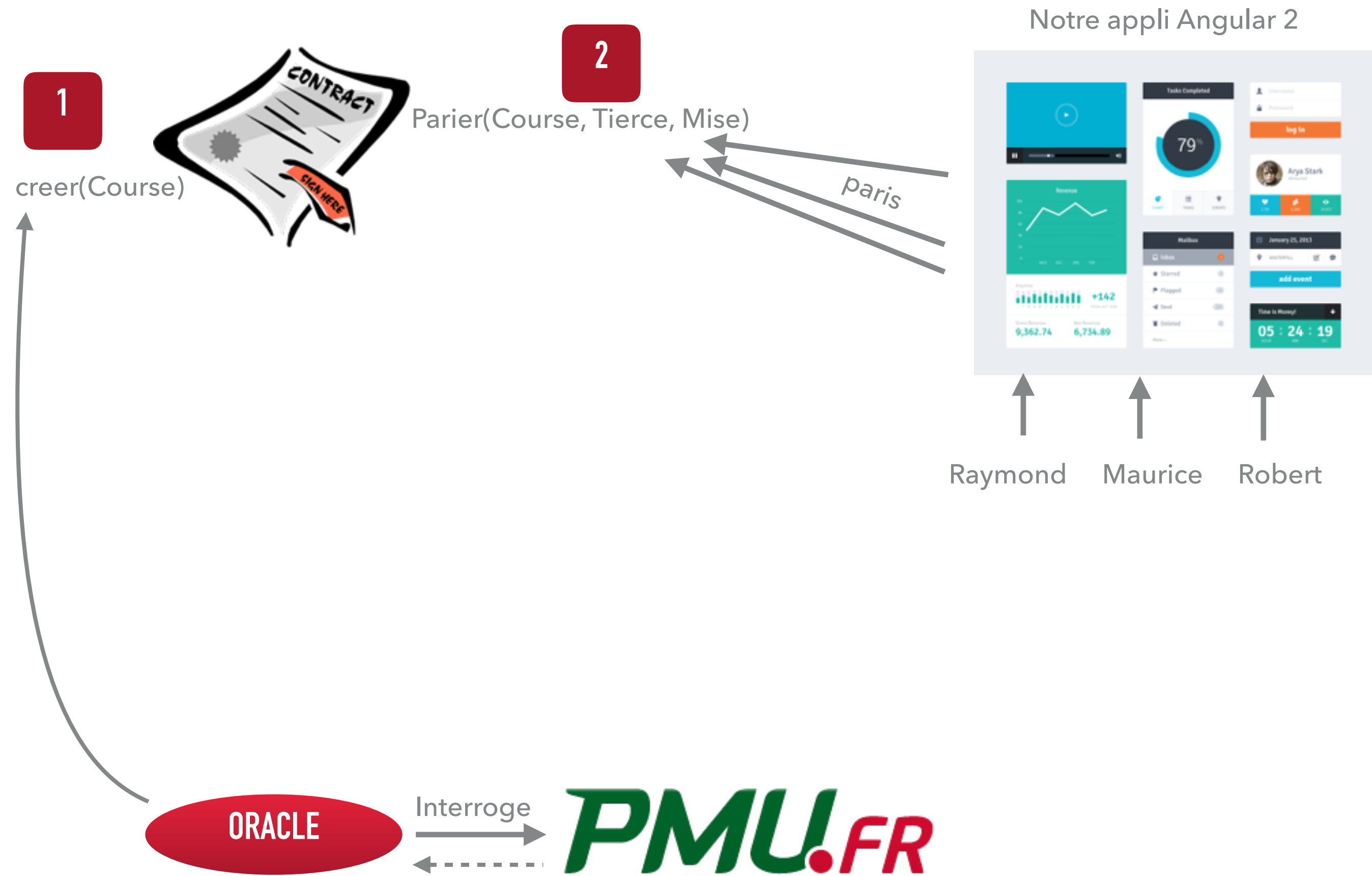
NOTRE D-APPS: MON TIERCE



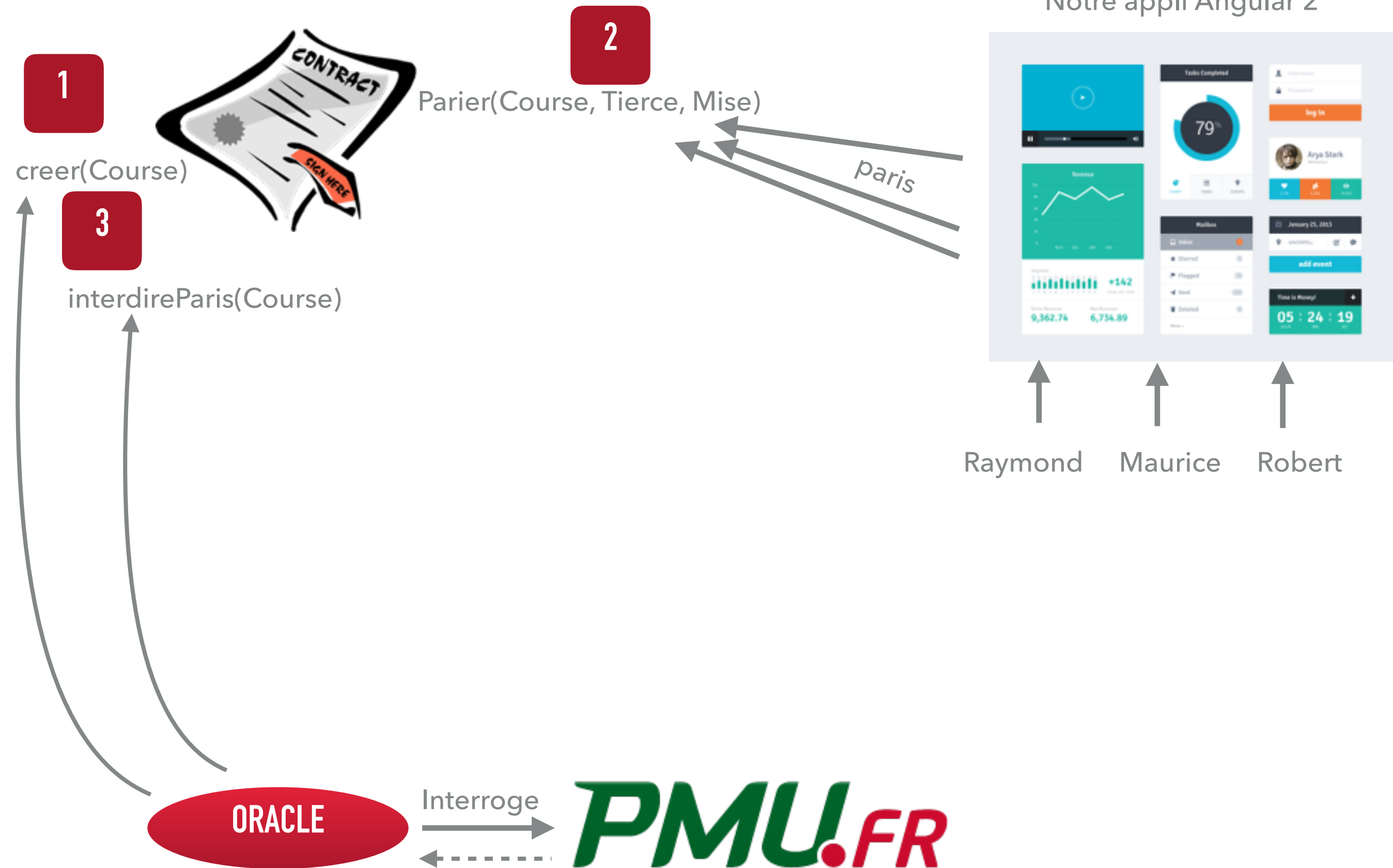
NOTRE D-APPS: MON TIERCE



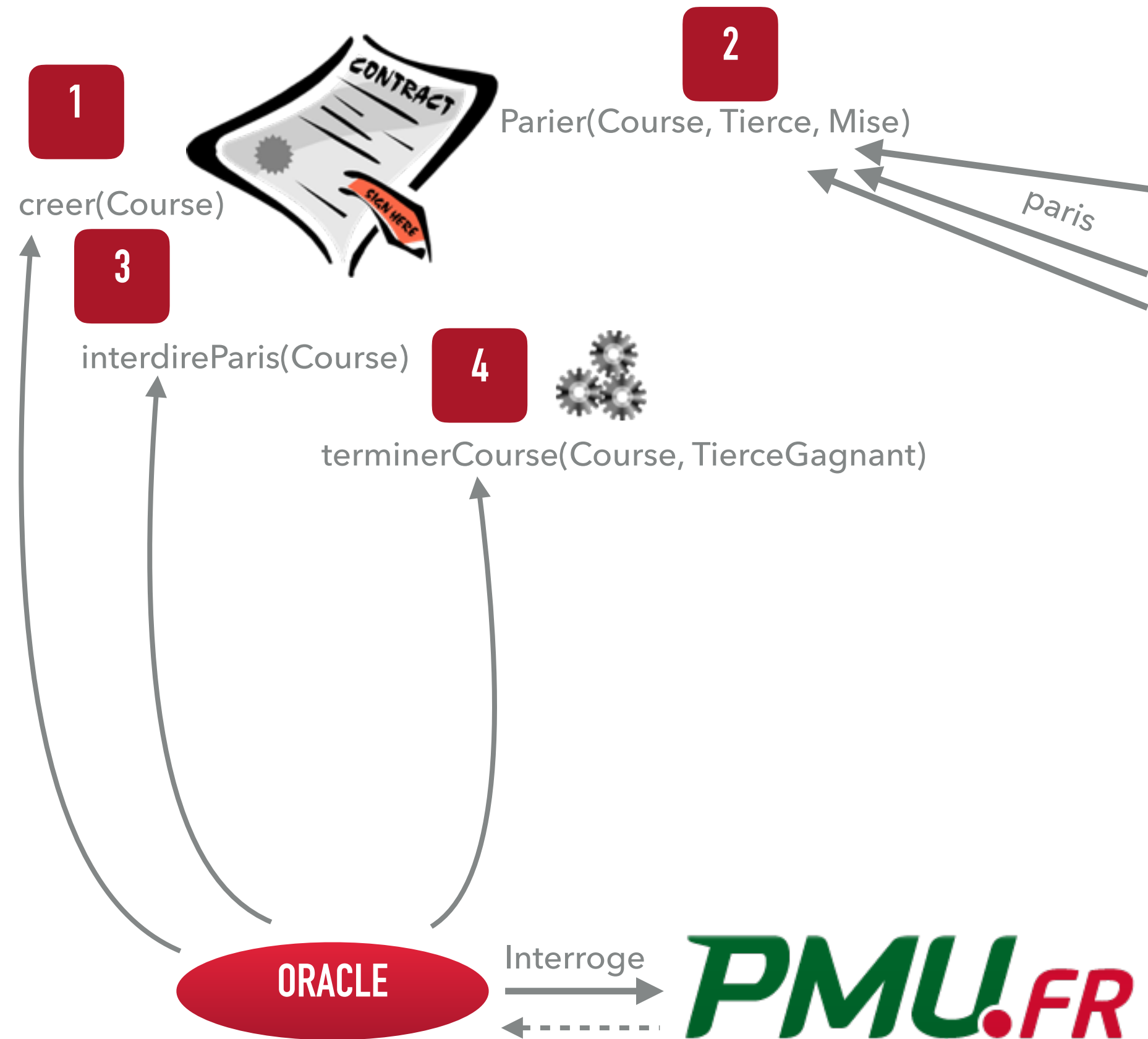
NOTRE D-APPS: MON TIERCE



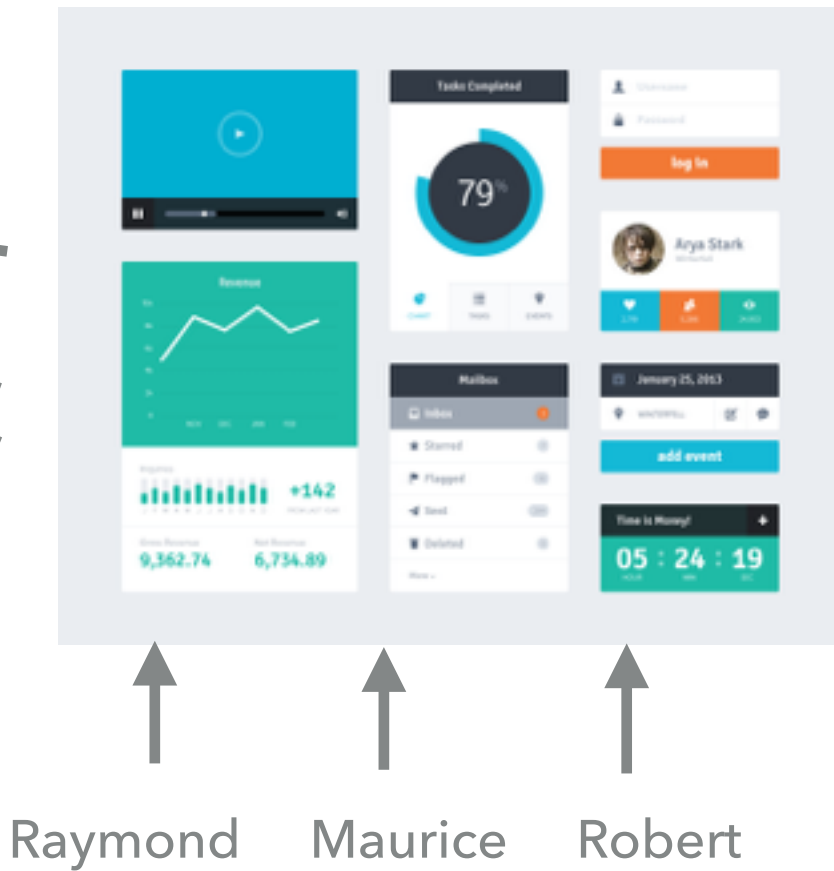
NOTRE D-APPS: MON TIERCE



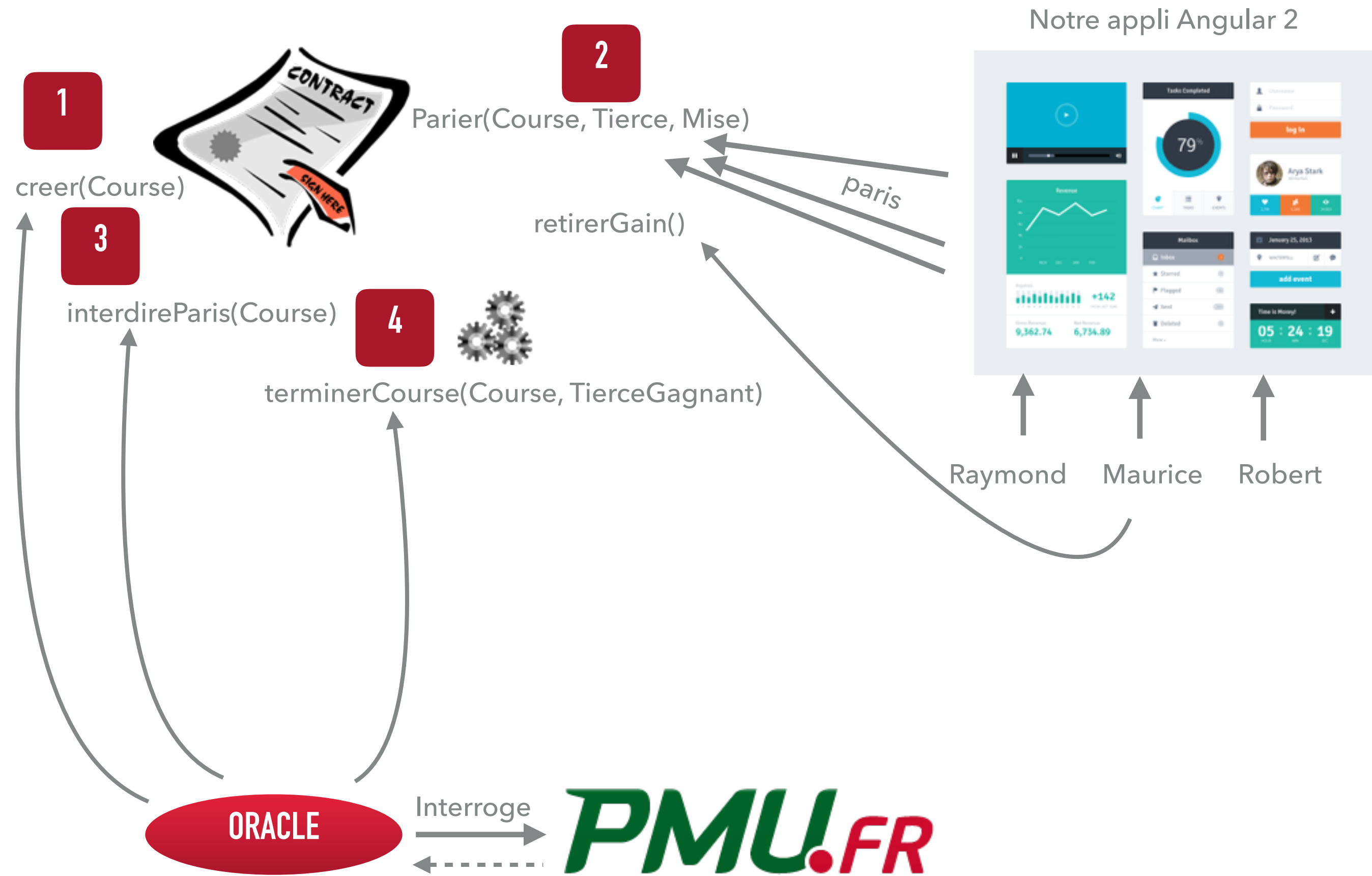
NOTRE D-APPS: MON TIERCE



Notre appli Angular 2



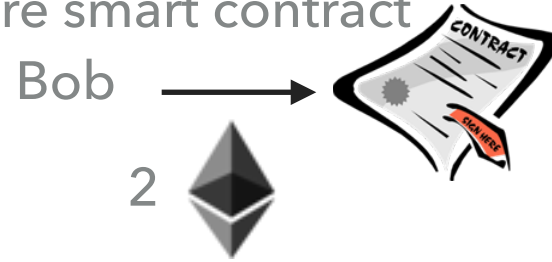
NOTRE D-APPS: MON TIERCE



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

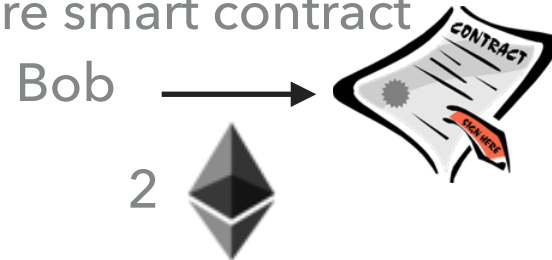
Bob veut faire un pari
notre smart contract



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract



2

Une transaction est créée

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

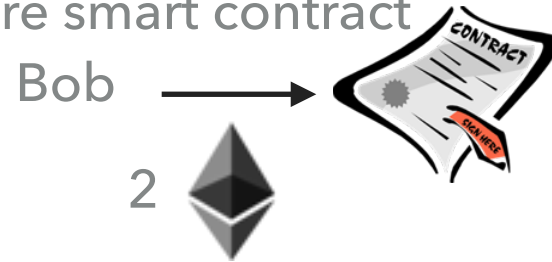
courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

TRANSACTIONS ET SMART-CONTRACT (1/2)

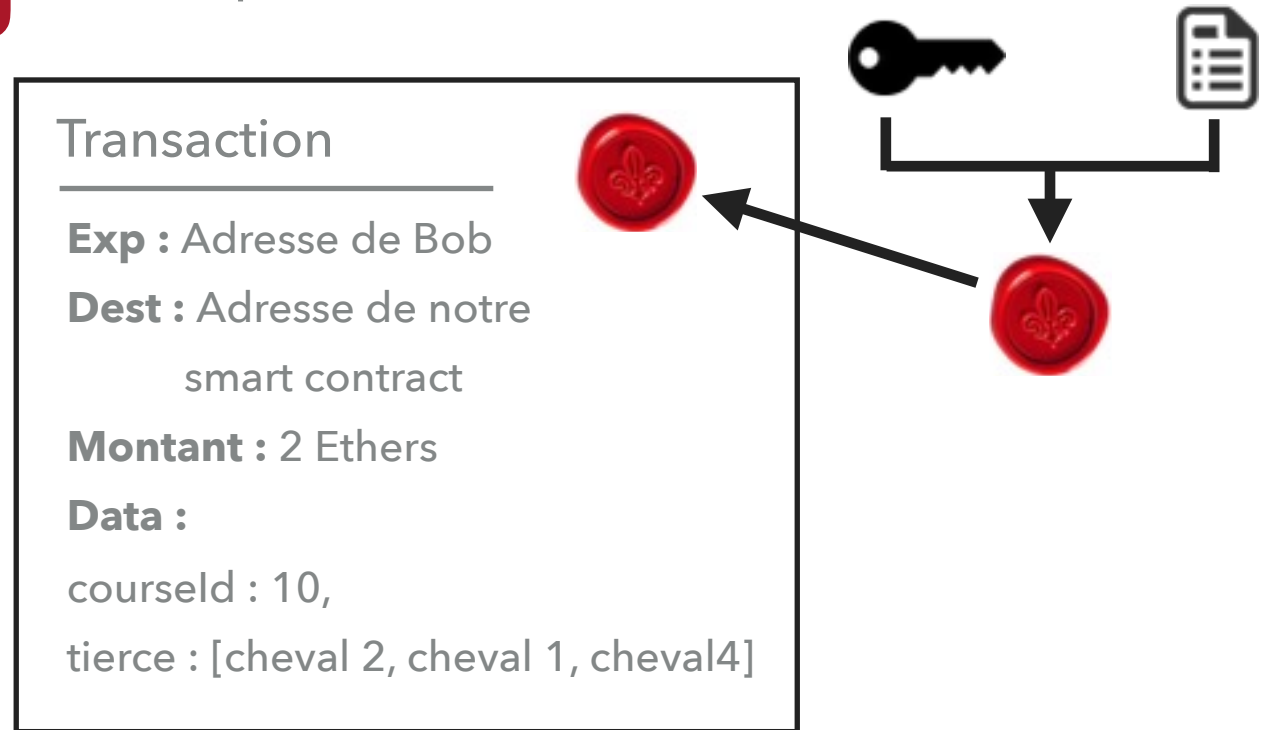
1

Bob veut faire un pari
notre smart contract



2

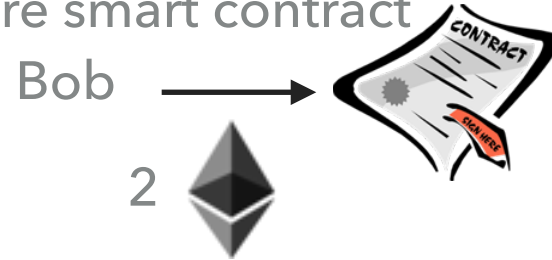
Une transaction est créée et signée en utilisant
la clé privée de Bob



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract



2

Une transaction est créée et signée en utilisant
la clé privée de Bob

MINEURS



Transaction

Exp : Adresse de Bob

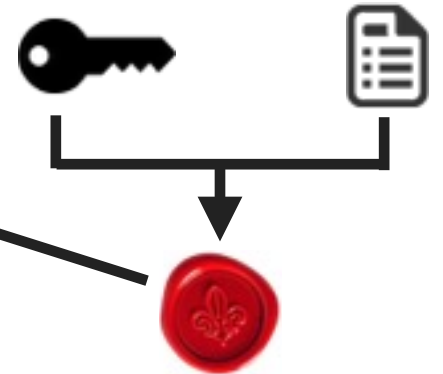
Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

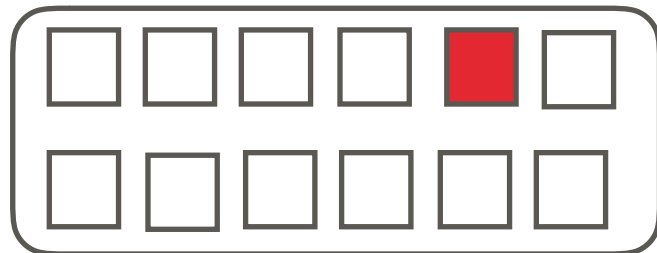
Bob veut faire un pari
notre smart contract

Bob

2



3



2

Une transaction est créée et signée en utilisant
la clé privée de Bob

MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

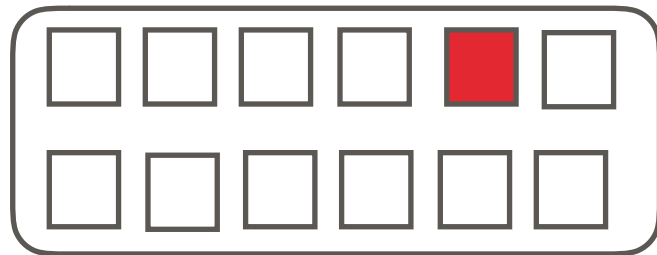
Bob veut faire un pari
notre smart contract

Bob

2



3



M1

M2

M3

M4

...

MX

2

Une transaction est créée et signée en utilisant
la clé privée de Bob

MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

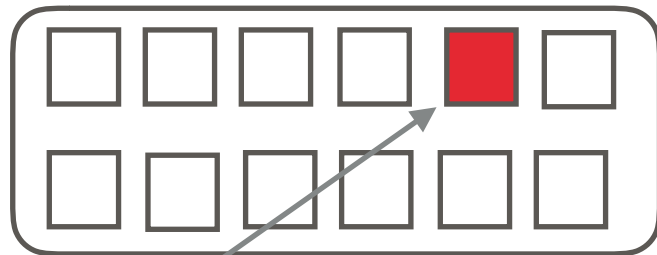
Bob veut faire un pari
notre smart contract

Bob

2



3



M1

M2

M3

M4

...

MX

2

Une transaction est créée et signée en utilisant
la clé privée de Bob

MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



Contrôle des transactions

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

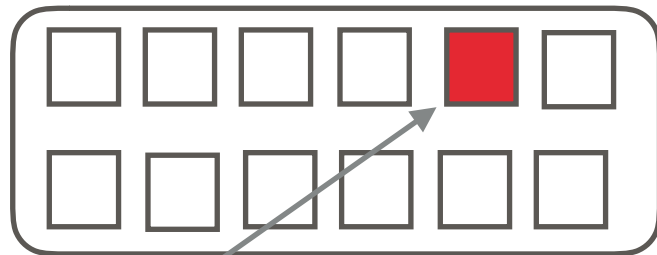
Bob veut faire un pari
notre smart contract

Bob

2



3



M1

M2

M3

M4

...

MX

MINEURS

M1

M2

M3

M4

MX

2

Une transaction est créée et signée en utilisant
la clé privée de Bob

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

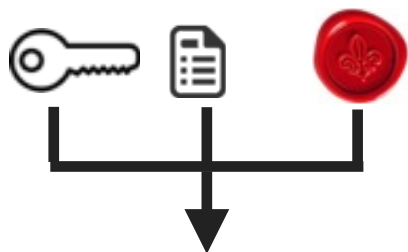
courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



Contrôle des transactions

1) ID de Bob



APPROVED

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

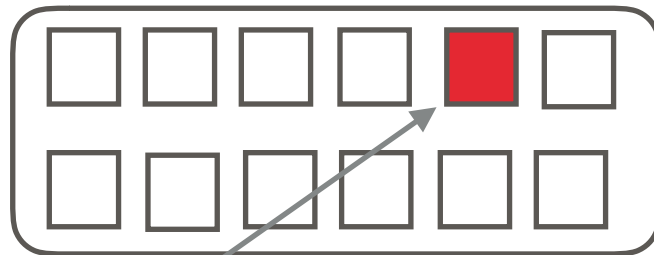
Bob veut faire un pari
notre smart contract

Bob

2



3



M1 M2 M3 M4 ... MX

2

Une transaction est créée et signée en utilisant
la clé privée de Bob

Transaction

Exp : Adresse de Bob

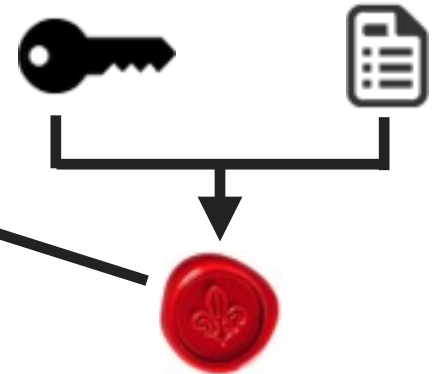
Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



MINEURS

M1

M2

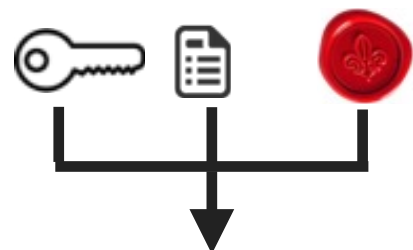
M3

M4

MX

Contrôle des transactions

1) ID de Bob



APPROVED

2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

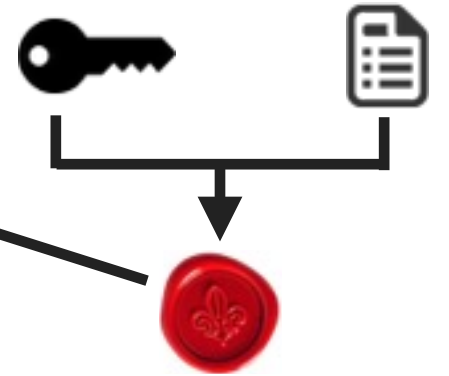
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

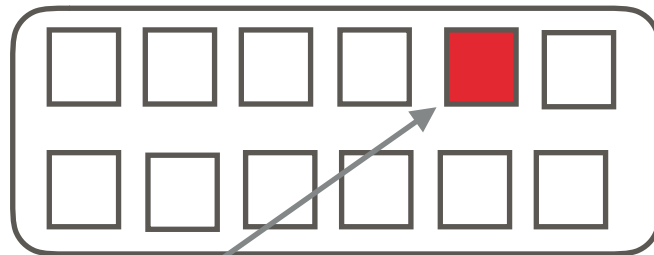
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

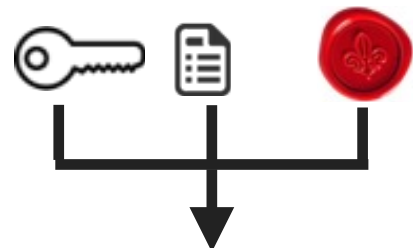
M4

...

MX

Contrôle des transactions

1) ID de Bob



APPROVED

2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

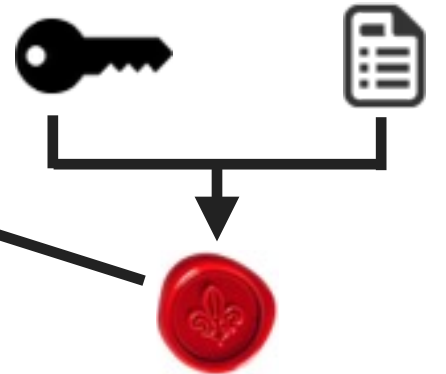
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

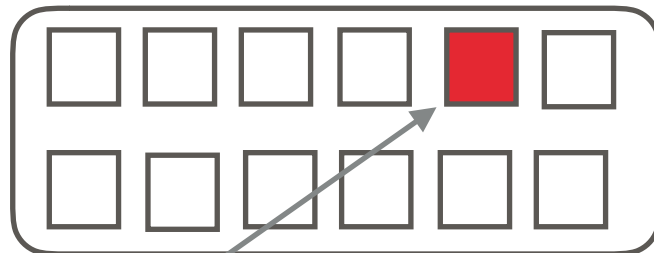
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

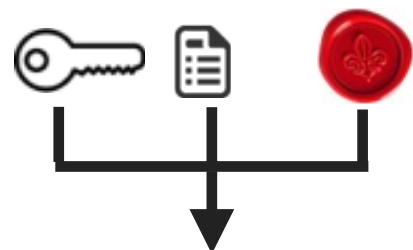
M4

...

MX

Contrôle des transactions

1) ID de Bob



2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

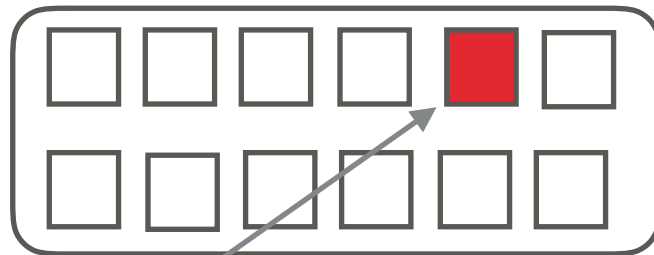
Bob veut faire un pari
notre smart contract

Bob

2



3



M1

M2

M3

M4

...

MX

MINEURS

M1

M2

M3

M4

MX

2

Une transaction est créée et signée en utilisant
la clé privée de Bob

Transaction

Exp : Adresse de Bob

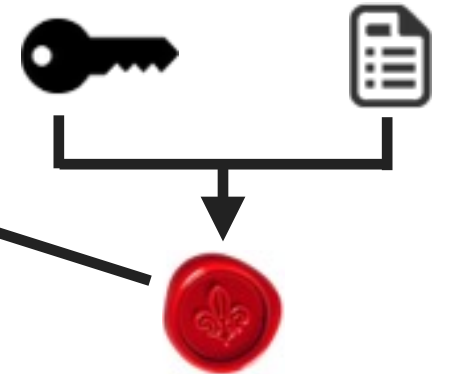
Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

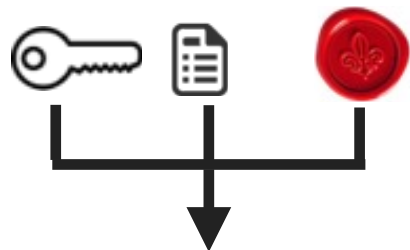
courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]



Contrôle des transactions

1) ID de Bob



2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

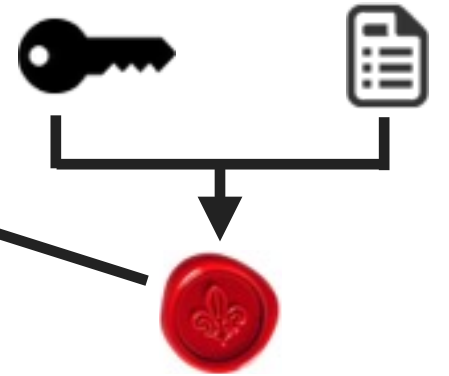
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

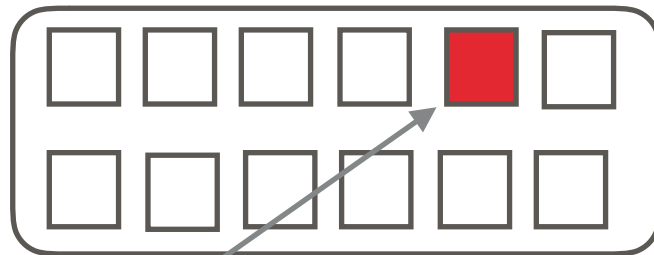
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

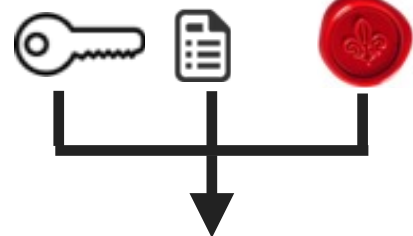
M4

...

MX

Contrôle des transactions

1) ID de Bob



APPROVED

2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

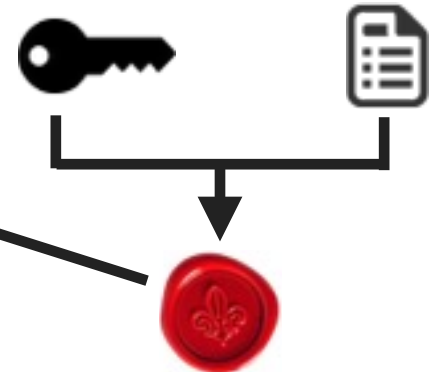
Bob

2

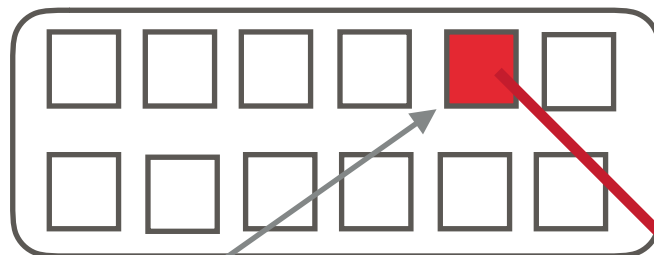


2

Une transaction est créée et signée en utilisant
la clé privée de Bob



3



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

M1

M2

M3

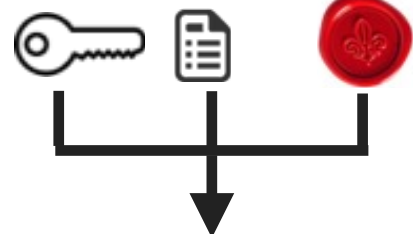
M4

...

MX

Contrôle des transactions

1) ID de Bob



2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

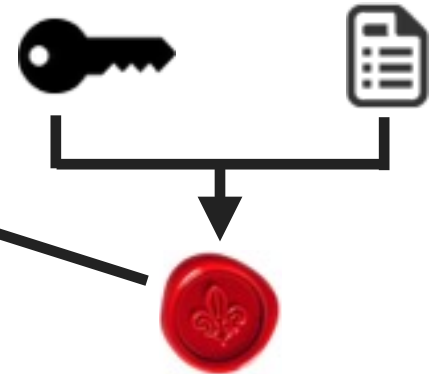
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

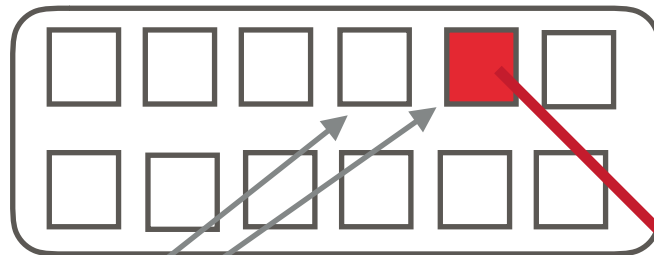
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

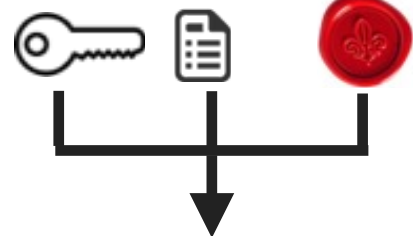
M4

...

MX

Contrôle des transactions

1) ID de Bob



2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

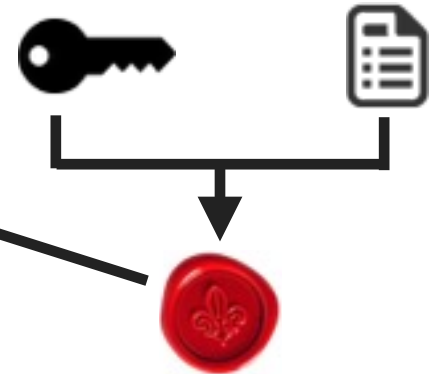
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

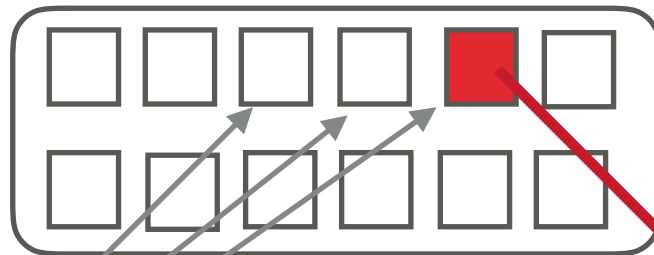
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

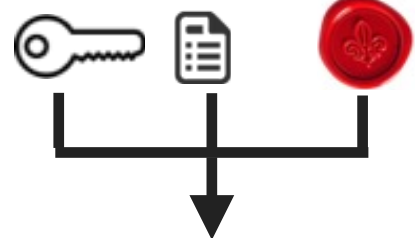
M4

...

MX

Contrôle des transactions

1) ID de Bob



APPROVED

2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT



TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

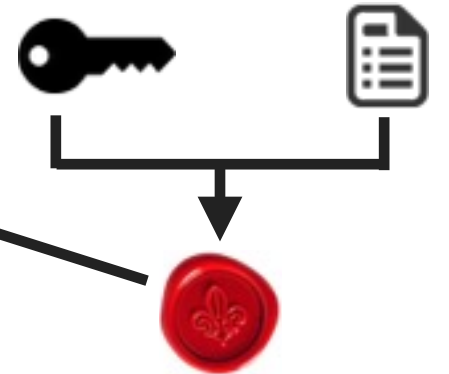
Bob

2

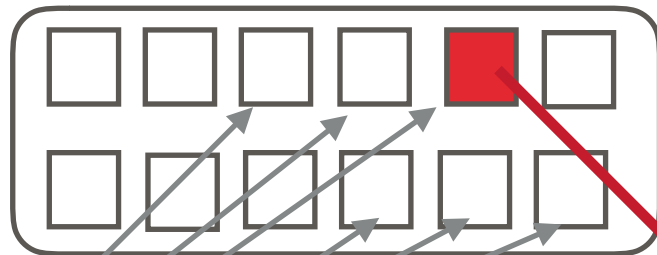


2

Une transaction est créée et signée en utilisant
la clé privée de Bob



3



MINEURS

M1

M2

M3

M4

MX

M1

M2

M3

M4

...

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

Montant : 2 Ethers

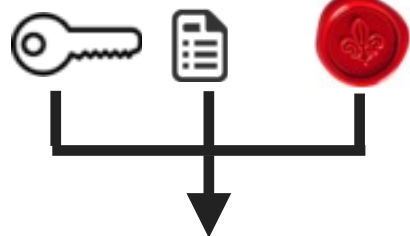
Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

Contrôle des transactions

1) ID de Bob



2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob -= 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT

ETAT

ETAT

ETAT

B2

ETAT

ETAT

ETAT

ETAT

TRANSACTIONS ET SMART-CONTRACT (1/2)

1

Bob veut faire un pari
notre smart contract

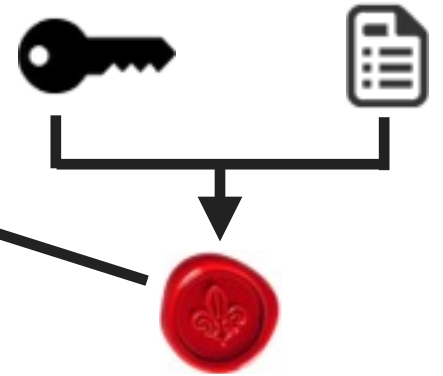
Bob

2



2

Une transaction est créée et signée en utilisant
la clé privée de Bob



MINEURS

M1

M2

M3

M4

MX

Transaction

Exp : Adresse de Bob

Dest : Adresse de notre
smart contract

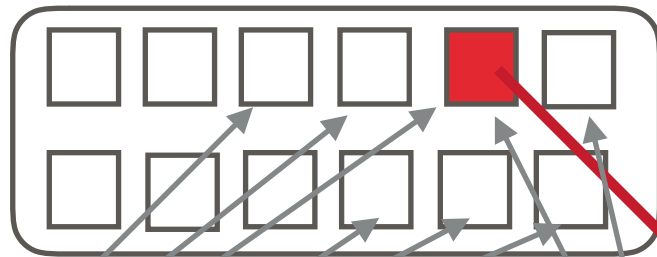
Montant : 2 Ethers

Data :

courseld : 10,

tierce : [cheval 2, cheval 1, cheval4]

3



M1

M2

M3

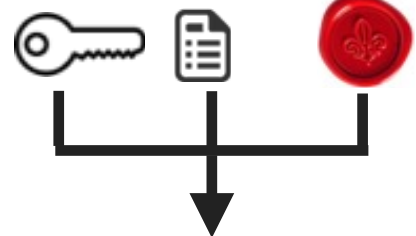
M4

...

MX

Contrôle des transactions

1) ID de Bob



APPROVED

2) Exécution de la transaction



Parier(Course,
Chevaux, Mise)



solde de Bob == 2 Ethers

storage contract : nouveau pari
2 ethers sur tierce

B1

ETAT

■

□

□

□

B2

ETAT

□

□

□

□

B3

ETAT

□

□

■

□

B4

ETAT

□

□

□

□

BX

ETAT

■

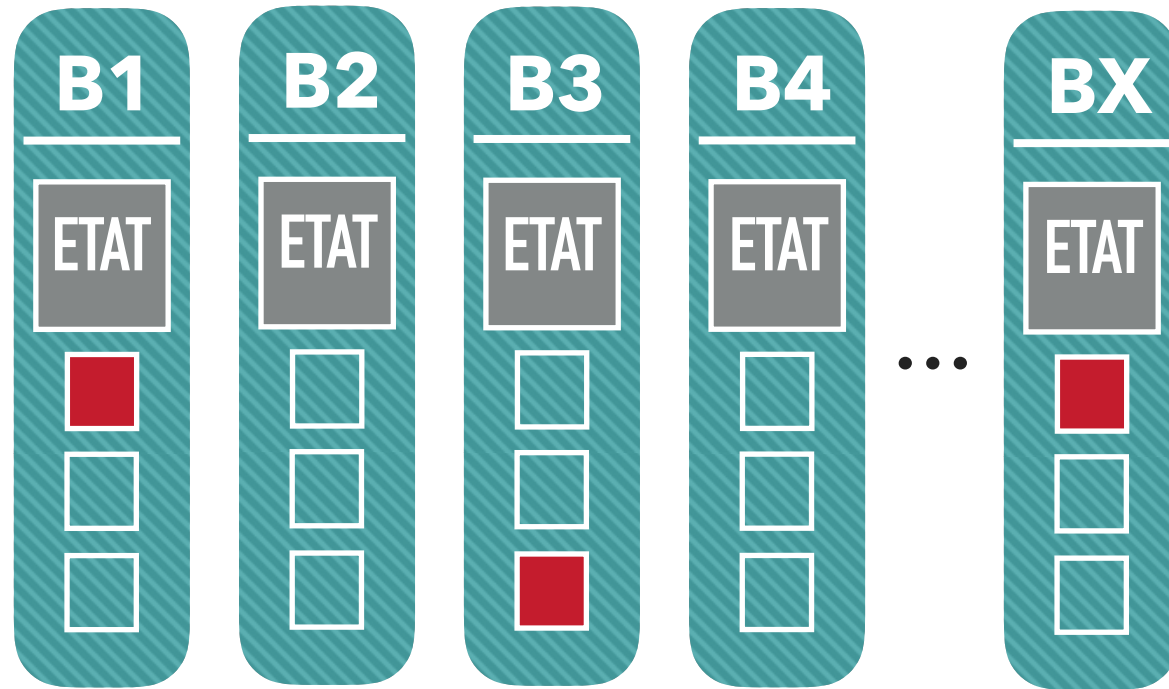
□

□

□

TRANSACTIONS ET SMART-CONTRACT (2/2)

4



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

Le mineur M4 y arrive en premier.



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

Le mineur M4 y arrive en premier.
Il gagne une récompense

Frais trs +
5 ETH



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

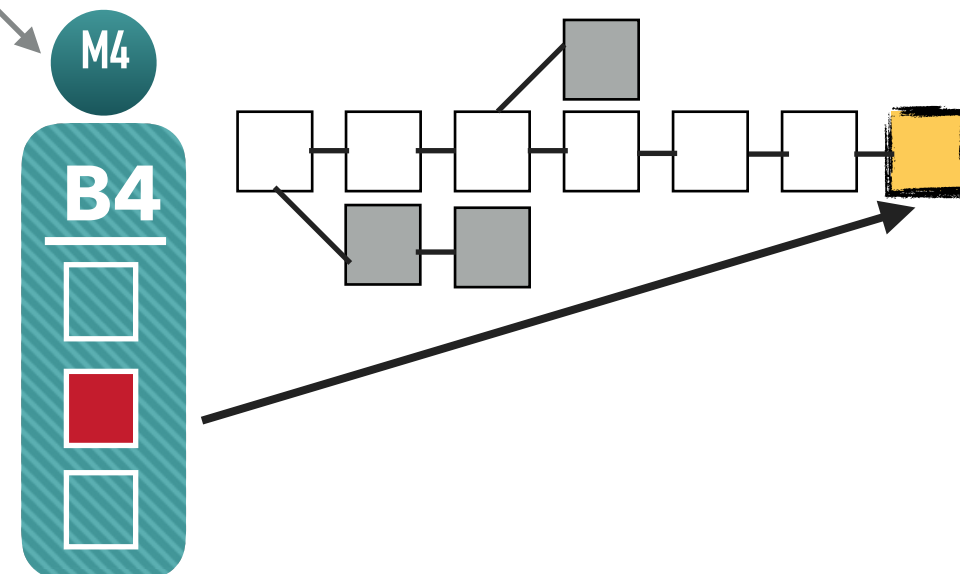
Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

Le mineur M4 y arrive en premier. Il gagne une récompense et ajoute son block à la fin de la blockchain.

Frais trs +
5 ETH



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



6

Le block est broadcasté et vérifié par tous les noeuds

Proof-of-work

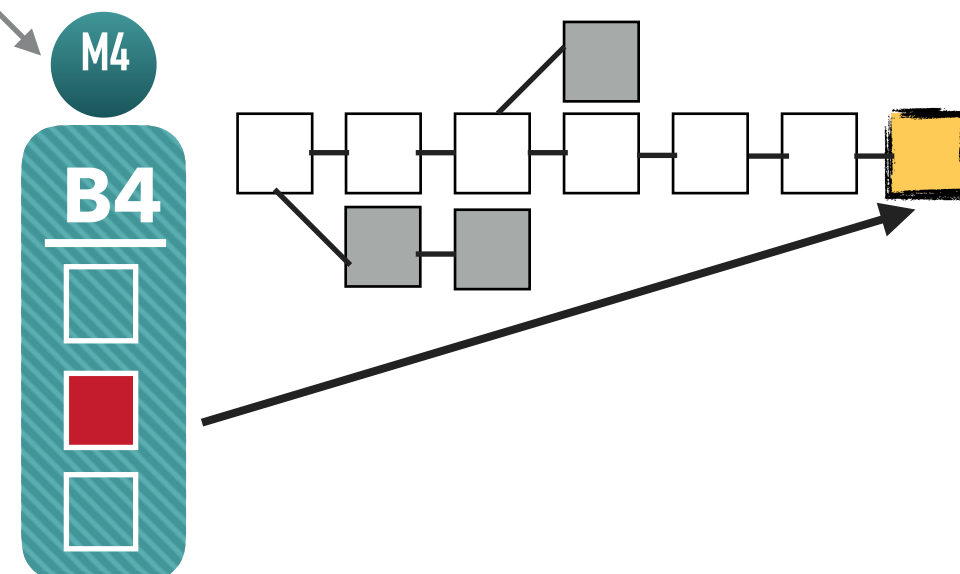
Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

Le mineur M4 y arrive en premier. Il gagne une récompense et ajoute son block à la fin de la blockchain.

Frais trs +
5 ETH



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

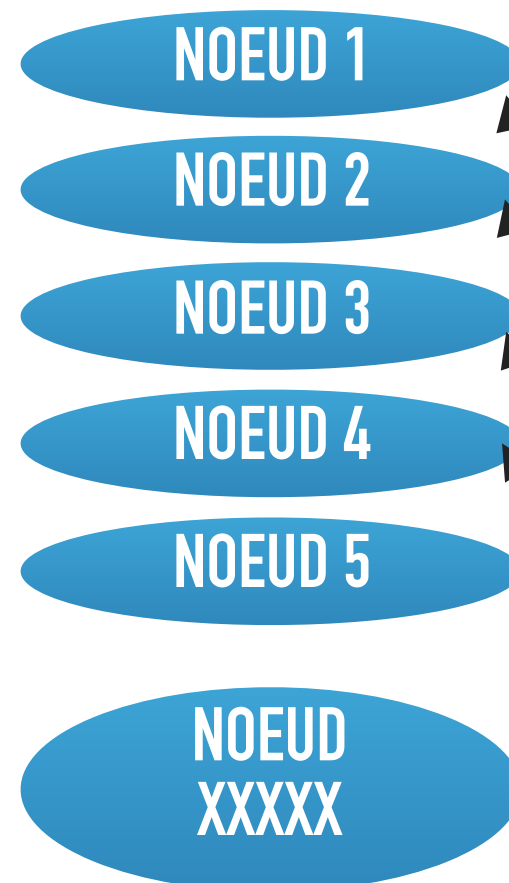
Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

6

Le block est broadcasté et vérifié par tous les noeuds

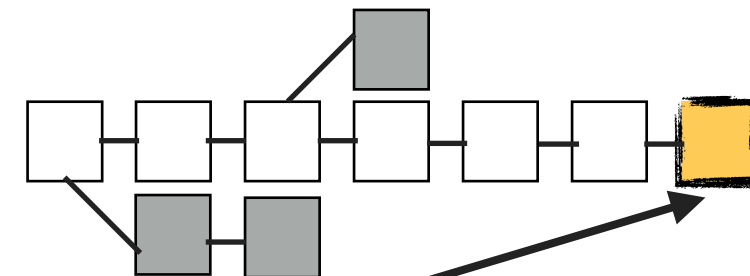


Frais trs +
5 ETH

M4

B4

Le mineur M4 y arrive en premier. Il gagne une récompense et ajoute son block à la fin de la blockchain.



TRANSACTIONS ET SMART-CONTRACT (2/2)

4



Proof-of-work

Les mineurs entrent en compétition dans l'espoir d'être les premiers à résoudre un algorithme cryptographique.



5

6

Le block est broadcasté et vérifié par tous les noeuds

Pour chaque trs du bloc

1) Contrôle ID

2) Exécution de la transaction

- Modification solde pour tr simple
- Exécution du code smart-contracts

NOEUD 1

NOEUD 2

NOEUD 3

NOEUD 4

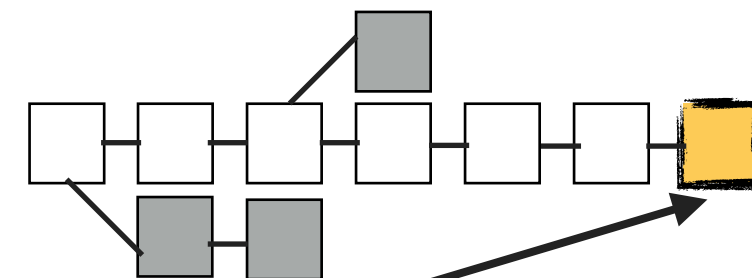
NOEUD 5

NOEUD XXXXX

Frais trs +
5 ETH

M4

B4



UNE DÉMO, UNE DÉMO !!!!

MURPHY'S LAW

**ANYTHING THAT CAN GO WRONG,
WILL GO WRONG.**

ETHEREUM

**MISE EN PLACE D'UN
ENVIRONNEMENT DE DÉVELOPPEMENT**

LE CHOIX DE LA BLOCKCHAIN

- ▶ 4 environnements possibles pour déployer vos smart contrats
 - ▶ **testrpc** : une blockchain de test en mémoire. Indispensable pour vos dévs mais à quelques comportements qui diffèrent avec une vraie blockchain ;
 - ▶ **une blockchain privée** : gratuite, et sans téléchargement interminable préalable => test en conditions réelles mais sur sa machine ;
 - ▶ **la blockchain de test ethereum AKA Ropsten** : c'est la blockchain de test officielle, avec pleins de vrais mineurs, mais l'éther y est gratuit (nécessite de télécharger l'intégralité de la blockchain de test Ethereum => 24h) => incontournable avant la vraie MEP, permet à des inconnus de jouer avec votre contrat
 - ▶ **la blockchain ethereum** : c'est la prod !!! Chaque opération de test vous coûtera du gaz et donc des sous (nécessite de télécharger l'intégralité de la blockchain Ethereum => 24h) => quand vous êtes super, super, super, super, super, super sûr de vouloir risquer votre argent et celui de vos utilisateurs dans votre contrat en béton armé.

Indispensable pour tester une blockchain privée, celle de test et celle de prod :

Metamask : client léger sous forme de plugin

Chrome

Geth : client en GO,

Eth : en C++,

Pyethapp : en python

- ▶ <https://live.ether.camp/> : IDE en ligne, que je n'ai pas essayé
- ▶ <https://ethereum.github.io/browser-solidity/>
- ▶ votre bon vieil IDE des familles

ETHEREUM

CODAGE DE SMART CONTRACT

LA VM ETHEREUM

STACK DE TAILLE ARBITRAIRE (PUSH, POP, SWAP, DUP)

ESPACE MEMOIRE VOLATILE DE TAILLE TEMPORAIRE (MLOAD, MSTORE)

ESPACE DE STOCKAGE ISOLE POUR CHAQUE CONTRACT DE TAILLE TEMPORAIRE (SLOAD, SSTORE)

ROM VIRTUELLE POUR LE CODE

INSTRUCTIONS POUR :

- OPERATIONS ARITHMETIQUES ET CRYPTOGRAPHIQUES,
- FLOW CONTROL,
- LECTURE D'INPUT,
- OUTPUT,
- CREER ET ENVOYER DES MESSAGES A D'AUTRES CONTRACTS,
- RECUPERER DES INFOS SUR LA BLOCKCHAIN (PREVIOUS HASH, TIMESTAMP, SENDER...)

CHACUNE DE CES INSTRUCTIONS COÛTE DU GAZ (ET, IN FINE, DE L'ETHER)

COUT EN GAZ

step	1	Default amount of gas to pay for an execution cycle.
stop	0	Nothing paid for the SUICIDE operation.
sha3	20	Paid for a SHA3 operation.
sload	20	Paid for a SLOAD operation.
sstore	100	Paid for a normal SSTORE operation (doubled or waived sometimes).
balance	20	Paid for a BALANCE operation
create	100	Paid for a CREATE operation
call	20	Paid for a CALL operation.
memory	1	Paid for every additional word when expanding memory
txdata	5	Paid for every byte of data or code for a transaction
transaction	500	Paid for every transaction

SOLIDITY : LE JAVASCRIPT DES BRAVES

- ▶ Typé statiquement
- ▶ Héritage
- ▶ Import de librairies
- ▶ Types complexes de type Struct

- ▶ Faites vous plaisir, pleins de langages sont supportés pour s'interfacer avec une D-app.
- ▶ Dans le TP, on l'a fait en Javascript, en Angular2, en s'appuyant sur ethereumJS

A VOUS DE JOUER

<https://github.com/benjaminfontaine/codelab-ethereum>



LES DANGERS DES SMARTS CONTRACTS

- ▶ UN BUG COUTE CHER (50 MILLIONS À DATE POUR LA PLUS GROSSE BÊTE)
- ▶ UN BUG EST EXTRÊMEMENT DIFFICILE À CORRIGER UNE FOIS DEPLOYÉ
- ▶ VOTRE SMART CONTRACT SERA ATTAQUÉ

AVANT LA MEP

- ▶ le contrat doit avoir été intégralement testé sur en local et sur le testnet
- ▶ toutes les failles de sécurité connues et recensées sur <https://github.com/ethereum/wiki/wiki/Safety> doivent avoir été éliminées (outil OYENTE)
- ▶ avoir prévu un moyen de corriger les bugs et de désactiver le contrat
- ▶ mettre en place des récompenses pour les rapporteurs de bug sur le testnet

PERMETTRE LE BUG FIX : LIBRARY PATTERN

- ▶ Par conception, un contrat ne peut pas être modifié, mais on peut tricher...
- ▶ Le storage coûte cher à migrer : stocker les données dans un contrat à part
- ▶ Ce pattern consiste à encapsuler sa logique dans une library, le contrat devenant une sorte d'interface déléguant les opérations à la librairie (<https://blog.colony.io/writing-upgradeable-contracts-in-solidity-6743f0eecc88#.yd9c22won>)

DES QUESTIONS ?



UNE FAILLE DE CONTRAT TORDUES : ENTREES CONCURRENTES

```
contract TokenWithInvariants {  
    mapping(address => uint) public balanceOf;  
    uint public totalSupply;
```

```
    modifier checkInvariants {  
        _  
        if (this.balance < totalSupply) throw;  
    }
```

```
    function deposit(uint amount) checkInvariants {  
        balanceOf[msg.sender] += amount;  
        totalSupply += amount;  
    }
```

```
    function transfer(address to, uint value) checkInvariants {  
        if (balanceOf[msg.sender] >= value) {  
            balanceOf[to] += value;  
            balanceOf[msg.sender] -= value;  
        }  
    }
```

```
    function withdraw() checkInvariants {  
        uint balance = balanceOf[msg.sender];  
        if (msg.sender.call.value(balance)()) {  
            totalSupply -= balance;  
            balanceOf[msg.sender] = 0;  
        }  
    }
```

```
}
```

UNE FAILLE DE CONTRAT TORDUES : ENTREES CONCURRENTES

```
contract TokenWithInvariants {  
    mapping(address => uint) public balanceOf;  
    uint public totalSupply;
```

```
    modifier checkInvariants {  
        _  
        if (this.balance < totalSupply) throw;  
    }
```

```
    function deposit(uint amount) checkInvariants {  
        balanceOf[msg.sender] += amount;  
        totalSupply += amount;  
    }
```

```
    function transfer(address to, uint value) checkInvariants {  
        if (balanceOf[msg.sender] >= value) {  
            balanceOf[to] += value;  
            balanceOf[msg.sender] -= value;  
        }  
    }
```

```
    function withdraw() checkInvariants {  
        uint balance = balanceOf[msg.sender];  
        if (msg.sender.call.value(balance)()) {  
            totalSupply -= balance;  
            balanceOf[msg.sender] = 0;  
        }  
    }  
}
```

UNE FAILLE DE CONTRAT TORDUES : ENTREES CONCURRENTES

```
contract TokenWithInvariants {  
    mapping(address => uint) public balanceOf;  
    uint public totalSupply;
```

```
    modifier checkInvariants {
```

```
        if (this.balance < totalSupply) throw;  
    }
```

```
    function deposit(uint amount) checkInvariants {  
        balanceOf[msg.sender] += amount;  
        totalSupply += amount;  
    }
```

Bonne pratique, permet d'éviter une attaque race-to-empty qui siphonne le compte.

Mais :

- 1) elle est trop permissive (devrait être ==)**
- 2) Les invariants ne sont vérifiés qu'en entrée et sortie de méthode**

```
    function transfer(address to, uint value) checkInvariants {  
        if (balanceOf[msg.sender] >= value) {  
            balanceOf[to] += value;  
            balanceOf[msg.sender] -= value;  
        }  
    }
```

```
    function withdraw() checkInvariants {  
        uint balance = balanceOf[msg.sender];  
        if (msg.sender.call.value(balance)()) {  
            totalSupply -= balance;  
            balanceOf[msg.sender] = 0;  
        }  
    }
```

ATTAQUE EN DEUX TEMPS

- Etape 1 -a : L'attaquant A va d'abord appeler récursivement la méthode `withdraw`, volant de l'éther à chaque itération. Mais l'invariant est censé annuler toute la transaction si la $\text{balance} < \text{totalSupply}$. C'est pourquoi il va tout rembourser d'un coup et transférer l'argent à un compte B.

totalSupply n'est modifié qu'à l'instruction d'après

	totalSupply	this.balance	balanceOf[A]	balanceOf[B]
A : appel 1 <code>contrat.withdraw()</code>	100	100	10	0
<code>balance = balanceOf[A];</code>	100	100	10	0
<code>A.call.value(balance)()</code>	100	100	10	0
A : appel 2 <code>contrat.withdraw()</code>	100	90	10	0
<code>balance = balanceOf[A];</code>	100	90	10	0
<code>A.call.value(balance)()</code>	100	90	10	0
A : appel 10 <code>contrat.withdraw()</code>	100	0	10	0
<code>balance = balanceOf[A];</code>	100	0	10	0
<code>A.call.value(balance)()</code>	100	0	10	0
A : appel 1 <code>contrat.send(100)</code>	100	0	10	0
A : appel 1 <code>contrat.transfert(A, B, 10)</code>	100	100	10	0
	100	100	0	10

ATTAQUE EN DEUX TEMPS

- Etape 1- b : Rien n'a changé à part que la balance de A se retrouve chez B. Sauf qu'ensuite les appels à la fonction `withdraw` vont alors se terminer, diminuant le `totalSupply` (l'éther que le contrat croit avoir) sans que ça gêne la validation de l'invariant.

balance est encore égal à 10
dans la méthode

	totalSupply	this.balance	balanceOf[A]	balanceOf[B]
A : appel 1 <code>contrat.send(100)</code>	100	0	10	0
A : appel 1 <code>contrat.transfert(A, B, 10)</code>	100	100	0	10
A : suite appel 1 <code>contrat.withdraw()</code> totalSupply -= balance;	100	100	0	10
balanceOf[msg.sender] = 0;	90	100	0	10
A : suite appel 2 <code>contrat.withdraw()</code> totalSupply -= balance;	80	100	0	10
balanceOf[msg.sender] = 0;	70	100	0	10
A : suite appel 10 <code>contrat.withdraw()</code> totalSupply -= balance;	0	100	0	10
balanceOf[msg.sender] = 0;				

ATTAQUE EN DEUX TEMPS

Etape 2 : Les appels aux méthodes ne sont plus protégés par l'invariant (totalSupply sera tout le temps inférieur à balance). L'attaquant B va donc pouvoir mener exactement la même attaque que A sauf que le contrat l'autorisera à garder la somme.

	totalSupply	this.balance	balanceOf[A]	balanceOf[B]
B : appel 1 contrat.withdraw() balance = balanceOf[B]; B.call.value(balance)()	0	100	0	10
B : appel 2 contrat.withdraw() balance = balanceOf[B]; B.call.value(balance)()	100	100	0	110
	90	100	0	110
	80	100	0	110
B : appel 10 contrat.withdraw() balance = balanceOf[B]; B.call.value(balance)()	70	100	0	110
	0	100	0	110