

DOSSIER DE PROJET

Vanessa Bouvier

TITRE PROFESSIONNEL DEVELOPPEUSE WEB & WEB MOBILE

FORMATION WEBFORCE3

SOMMAIRE

Compétences couvertes par le projet	p. 3
Résumé du projet	p. 4
Présentation de l'entreprise et de l'équipe	p. 5
Technologies utilisées	p. 6
Explication de la Blockchain	p. 7
Hachage des données	p. 9
Projet Blockchain Key Proof	p. 11
Version 1 : Insertion en BDD et dans le Registre	p. 12
Version 2 : Insertion dans le Registre & interface	p. 18
Version 3 : Insertion en BDD sans rechargeement de la page	p. 21
Tests de la Version 1	p. 25
Vulnérabilités de sécurité	p. 26
Recherche en anglais	p. 29
Synthèse et conclusion	p. 32

ANNEXES

ANNEXE 1 : Code de la page de connexion (Version 1)
ANNEXE 2 et 2 BIS : Code du formulaire et PHP (Version 1)
ANNEXE : CSS (Version 2)
ANNEXE 4 : Visuel du formulaire (Version 2)
ANNEXE 5 : Code de la page de connexion (Version 3)
ANNEXE 6 : insertion.php (Version 3)
ANNEXE 7 : Explication de l'écriture dans le Registre
ANNEXE 8 : Cinq premiers résultats de la recherche en anglais

COMPÉTENCES COUVERTES PAR LE PROJET

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique
- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web

COMPÉTENCES TRANSVERSALES

- Utiliser l'anglais dans son activité professionnelle en développement web
- Actualiser ses compétences en développement web

RÉSUMÉ DU PROJET

J'effectue un stage de 6 mois en tant que Développeuse Fullstack dans la société SPURO, spécialisée dans la Blockchain.

Une des missions principales de mon stage concerne la signature électronique BKP : Blockchain Key Proof. Cette application existait déjà mais je devais la modifier et l'améliorer pour proposer plus de fonctionnalités aux clients.

En effet, un utilisateur peut s'enregistrer à distance sur l'application Spuro et créer sa propre signature électronique, qui sera inscrite dans la Blockchain Spuro. Ensuite ce même utilisateur pourra signer électroniquement et envoyer un document en utilisant les informations qu'il a entrées lors de son enregistrement. Ceci permet d'avoir la garantie de l'intégrité du document, puisqu'il est possible de comparer la signature du document et celle enregistrée dans la Blockchain.

Mes missions :

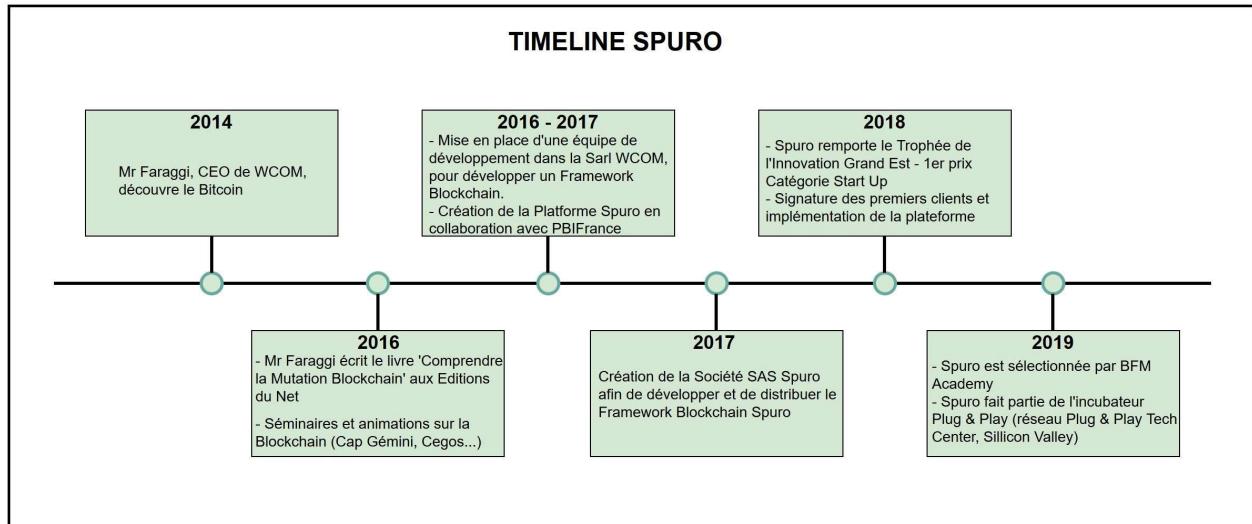
- Créer un formulaire d'enregistrement destiné à une Autorité d'Enregistrement (mairie, service RH d'une entreprise...), afin de vérifier de visu l'identité de la personne.
- Créer une Base de Données pour enregistrer les personnes inscrites
- Créer 3 versions du projet :
 - 1-** Un formulaire d'enregistrement avec insertion des données de l'utilisateur en Base de Données et enregistrement de l'empreinte numérique dans la Blockchain.
 - 2-** Un formulaire d'enregistrement renvoyant vers une page PHP qui effectue l'enregistrement de l'empreinte numérique dans la Blockchain et affiche le résultat sur une autre page que celle du formulaire.
 - 3-** Un formulaire d'enregistrement avec insertion des données de l'utilisateur en Base de Données, sans rechargement de la page du formulaire.

Pour réaliser ce projet j'ai dû me familiariser avec le code source de la Blockchain Spuro et avec les spécificités liées au principe de la Blockchain, ainsi que me documenter et maîtriser la fonction de hachage et les algorithmes cryptographiques qui lui sont liés.

PRÉSENTATION DE L'ENTREPRISE

SPURO

Cette société, où j'effectue un stage de 6 mois, est spécialisée dans la Blockchain.



PRODUITS :

- Framework Spuro

Spuro a développé un Framework Blockchain qui permet aux entreprises de déployer rapidement une blockchain, ainsi que de la personnaliser.

- Ledger Of Proof

Ledger Of Proof permet de gérer la preuve de la propriété et la traçabilité des informations sans avoir besoin d'un certificat. Il s'agit d'un réseau de partage de preuve de données.

- Vote Électronique

BVS (Blockchain Vote Secure) permet d'effectuer des votes électroniques en ligne.

- Signature Électronique

L'application BKP ('Blockchain Key Proof') effectue les fonctionnalités d'une signature électronique, elle permet de signer numériquement puis d'envoyer des documents.

EQUIPE :

Benjamin Faraggi : Fondateur / CEO

Isabelle Matejicek : Responsable Marketing et Communication

Jacques Hellard : Directeur Technique

Justine Jourdain : Stagiaire Fullstack WebForce3 (6 mois)

Vanessa Bouvier : Stagiaire Fullstack WebForce3 (6 mois)

La création de la Blockchain et des autres produits Spuro a été effectuée par un Développeur extérieur à l'entreprise.

TECHNOLOGIES UTILISÉES

Langages de programmation :

Php 7 (POO)

Html 5

Javascript

Framework CSS :

Bootstrap

Bibliothèque JS :

JQuery

Serveur :

LARAGON (HTTP Apache)

Autres Technologies:

SQL

AJAX

Spuro Platform Blockchain

Hash / Cryptographie

Registre (Ledger)

EXPLICATION DE LA BLOCKCHAIN

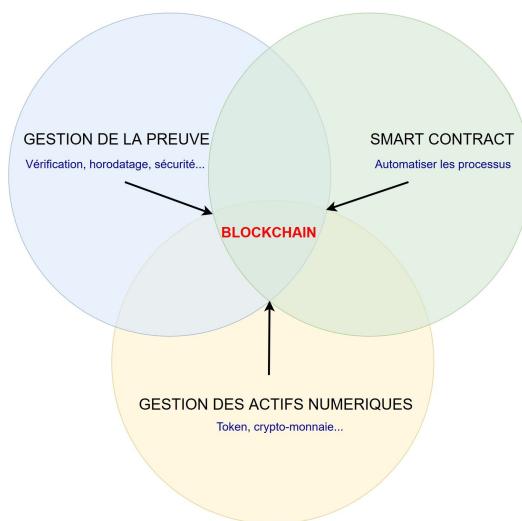
DEFINITION

La Blockchain est une technologie de stockage et de transmission de données qui fonctionne de manière décentralisée (pas d'organe central de contrôle).

Les informations échangées entre les utilisateurs du réseau sont regroupées par blocs horodatés et irréversiblement liés les uns aux autres, formant une chaîne: la blockchain. Les écritures enregistrées sur ce bloc et sur tous les précédents sont inaltérables et infalsifiables.

DOMAINES

De nombreuses Blockchains existent, avec des fonctions variées. Les Blockchains les plus connues concernent la gestion des actifs numériques et plus précisément les crypto-monnaies. Mais en réalité **il existe trois domaines distincts :**



SPECIFICITES DE LA BLOCKCHAIN

CRYPTOGRAPHIE / HASH : Les Blockchains s'appuient en grande partie sur la cryptographie pour assurer la sécurité de leurs données. La fonction cryptographique extrêmement importante est celle du hachage, l'objectif étant de garantir l'intégrité des documents.

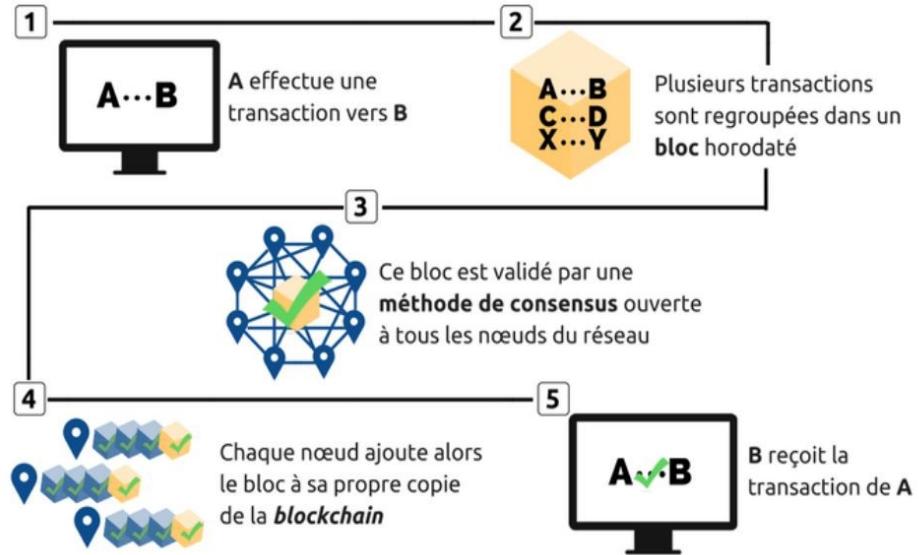
DLT : Distributed Ledger Technology ou Technologie des Registres Distribués, est un système numérique décentralisé qui enregistre des transactions d'actifs et leurs détails dans plusieurs emplacements à la fois (dans un Registre ou Ledger).

NODE : ou Nœud, ordinateur relié au réseau blockchain et utilisant un programme relayant les transactions. Les nœuds conservent une copie du registre blockchain.

LANGAGES UTILISES : JS, Java, Python, Go, Php, C# ou C++

FONCTIONNEMENT

EXEMPLE D'ENREGISTREMENT D'UNE TRANSACTION SUR UNE BLOCKCHAIN

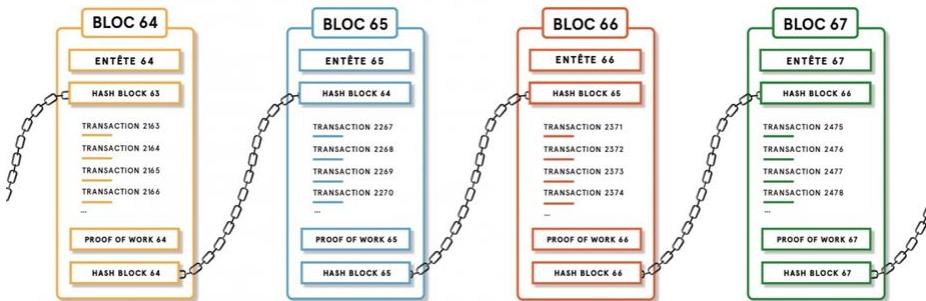


Source : Office parlementaire des choix scientifiques et technologiques

Les transactions effectuées entre les utilisateurs du réseau sont regroupées par blocs.

Une fois le bloc validé, il est horodaté et ajouté à la chaîne de blocs. La transaction est alors visible pour le récepteur ainsi que l'ensemble du réseau.

REGISTRE BLOCKCHAIN

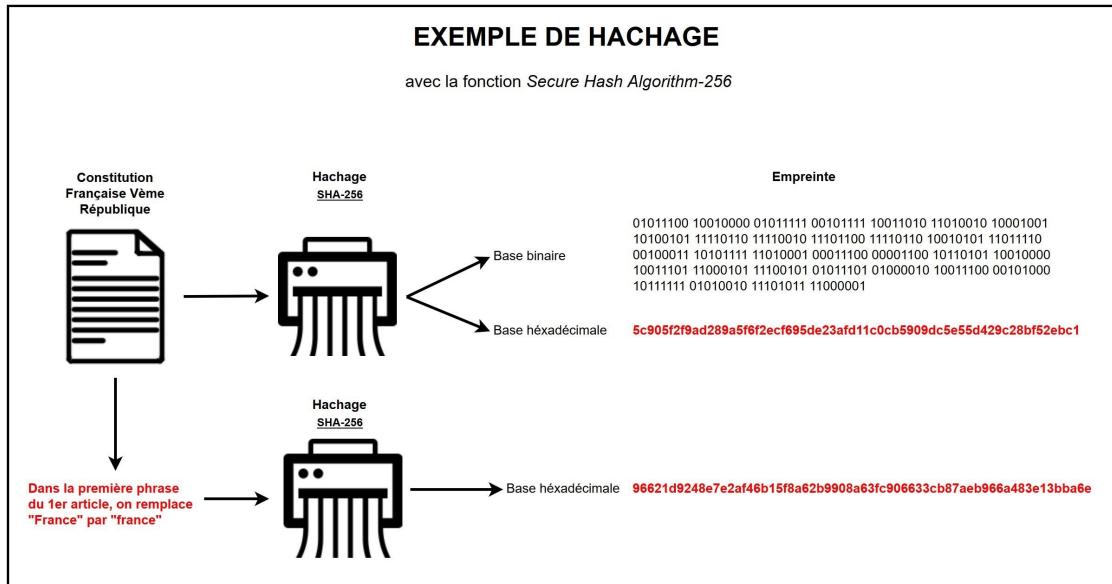


Les blocs sont liés les uns aux autres et gardent une trace des transactions précédentes, les rendant ainsi inaltérables et infalsifiables.

HACHAGE DES DONNEES

Une fonction de hachage est une fonction mathématique qui permet de convertir une valeur numérique d'une certaine taille dans une valeur numérique d'une autre taille. La fonction de hachage prend en entrée une chaîne de bits de taille arbitraire et produit en sortie une chaîne de bits de taille fixée et réduite : **l'empreinte (ou hash ou condensé)**.

Dans le cadre des Technologies de Registre Décentralisés, **la fonction de hachage effectuée par l'algorithme a pour fonction principale de comparer des groupes de données et de déterminer par comparaison si leur contenu a été altéré.** La majorité des algorithmes de signature numérique vérifient que la valeur de hachage du fichier n'a pas changé. A cause des caractéristiques des fonctions de hachage cryptographiques, la vérification de la valeur de hachage est considérée comme la preuve que le fichier lui-même est authentique.



Description de la fonction

```
hash ( string $algo , string $data [, bool $raw_output = FALSE ] ) : string
```

\$algo = nom de l'algorithme de hachage voulu : ‘md5’, ‘sha1’, ‘sha256’, ‘sha384’, ‘X-11’ ...

\$data = données à hacher

\$raw_output = FALSE : retourne une chaîne de caractères contenant l'empreinte numérique calculée en chiffres hexadécimaux et en minuscules.

\$raw_output = TRUE : la sortie se fait en données brutes binaires

Par défaut \$raw_output est paramétré à FALSE.

Pour effectuer le hachage il est également nécessaire de convertir tous les caractères alphabétiques en minuscules. En effet, les fonctions de hachage en md5, en sha256 ... sont **sensibles à la casse**. Il faut donc utiliser la fonction :

```
strtolower ( string $string ) : string
```

\$string : chaîne d'entrée

La chaîne est retournée en minuscule.

Il peut être également nécessaire d'encoder les caractères en utf8 car le moindre changement de caractère, oubli d'accent... changera le hash.

ALGORITHME SHA-256

SHA (Secure Hash Algorithm), désigne une fonction de hachage cryptographique conçue par l'Agence Nationale de Sécurité américaine (NSA). Il existe plusieurs versions : SHA-0, SHA-256, SHA-512.... Ces algorithmes de hachage sont utilisés par des autorités de certification pour la signature de certificats électroniques.

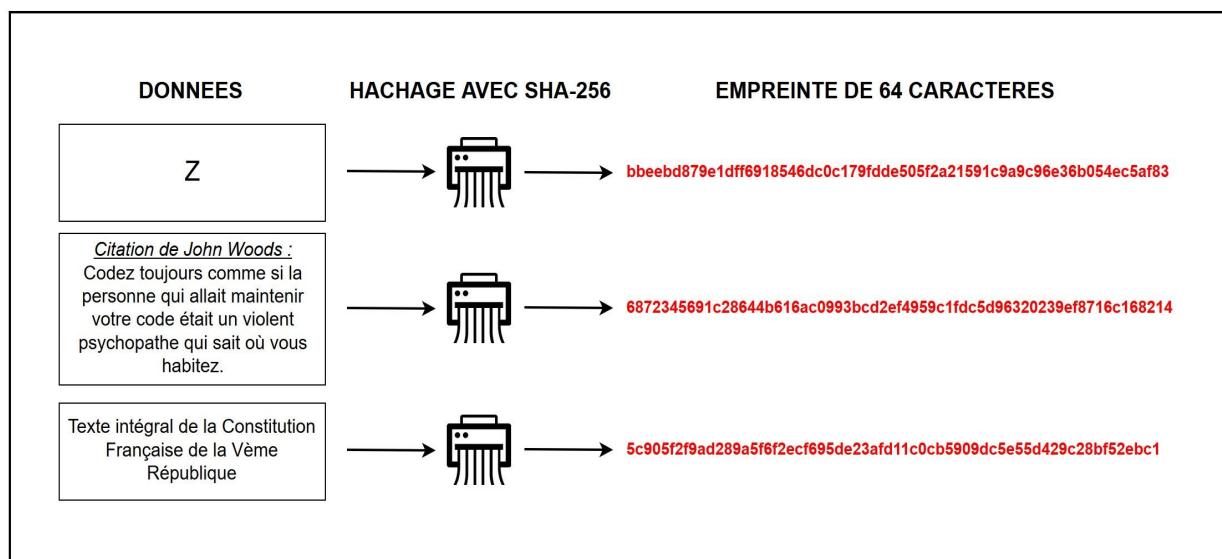
Le SHA-0, qui est l'algorithme d'origine, est dorénavant obsolète. Il a été créé au début des années 1990 et a été décliné et amélioré au fil des ans pour proposer des algorithmes plus performants et répondre aux nouvelles failles et aux nouveaux éléments de vulnérabilité des ordinateurs.

Dans la Blockchain, l'algorithme majoritairement utilisé est le '**SHA-256**', et l'empreinte numérique est calculée en **hexadécimal** (64 caractères) et non en binaire (256 caractères).

Un même algorithme produira en sortie une valeur de hachage qui sera toujours de la même longueur, quelque soit la longueur des données en entrée.

Notation hexadécimale complète de SHA-256 : 256 Bits = 32 Octets = 64 caractères

Exemple avec des sorties en hexadécimal :



PROJET BLOCKCHAIN KEY PROOF

INTRODUCTION

L’application BKP possède les fonctionnalités de la signature électronique sans utiliser de certificat numérique. La signature électronique permet, pour un document numérique, de garantir l’identité du signataire ainsi que la non-répudiation et l’intégrité du document signé.

L’application effectue le hash de certaines informations de l’utilisateur, dont la « clé de preuve », qui est une suite de caractères, possédée uniquement par le propriétaire. La clé de preuve est unique et ne peut pas être récupérée si elle est perdue. Elle sert à garantir l’intégrité de la signature électronique.

MISSIONS

1 – Création d’une interface d’enregistrement

Le principe est qu’une personne référente enregistre elle-même tous les utilisateurs et crée une signature électronique unique pour ces derniers. Par sécurité l’identité de la personne doit être vérifiée de visu. Lors de cet enregistrement dans les locaux de l’Autorité de Vérification (entreprise, mairie, assurance...), les informations personnelles de l’utilisateur sont enregistrées.

Un message avec le hash effectué doit apparaître après validation du formulaire.

2 – Création d’une Base De Données

Création d’une base de données avec une table permettant l’enregistrement des informations de chaque personne pour éviter de créer des doublons et garder une trace des personnes enregistrées.

3 – Création de différentes versions

- Un formulaire d’enregistrement avec insertion des données de l’utilisateur en Base de Données et enregistrement de l’empreinte numérique dans la Blockchain.
- Un formulaire d’enregistrement renvoyant vers une page PHP qui effectue l’enregistrement de l’empreinte numérique dans la Blockchain et affiche le résultat sur une autre page que celle du formulaire.
- Un formulaire d’enregistrement avec insertion des données de l’utilisateur en Base de Données, sans rechargement de la page du formulaire.

VERSION 1

Création d'un formulaire d'enregistrement avec insertion des données de l'utilisateur en BDD et enregistrement de l'empreinte dans la Blockchain.

SPECIFICITES FONCTIONNELLES

Le principe est qu'une personne référente enregistre elle-même tous les utilisateurs et crée une signature électronique unique pour ces derniers. Par sécurité l'identité de la personne doit être vérifiée de visu. Lors de cet enregistrement dans les locaux de l'Autorité de Vérification (entreprise, mairie, assurance...), les informations personnelles de l'utilisateur sont enregistrées et conservées dans une BDD.

Un message avec le hash de l'utilisateur doit apparaître après validation du formulaire. Ceci est fondamental car ce hash représente l'empreinte électronique de l'utilisateur. Celui-ci peut donc la conserver. Cette empreinte sera vérifiée pour garantir l'intégrité des documents signés.

Cette application est destinée à être utilisée sur ordinateur ou éventuellement sur tablette.

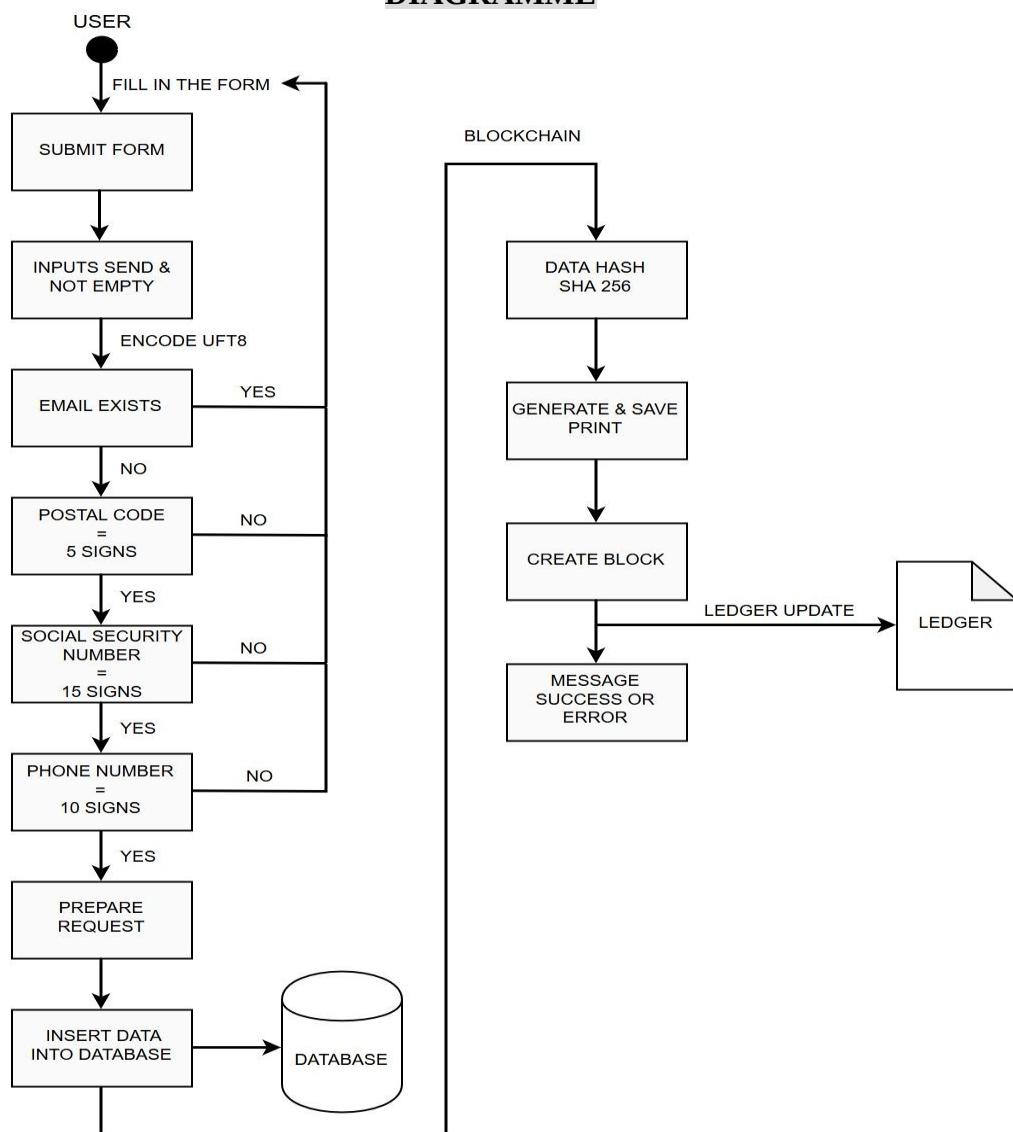
MAQUETTE DU PROJET A REALISER

ENREGISTRER UN UTILISATEUR		
Nom	Prénom	
<input type="text"/>	<input type="text"/>	
Date de naissance	N° Sécurité Sociale	N° Téléphone
<input type="text"/>	<input type="text"/>	<input type="text"/>
Adresse		
<input type="text"/>		
Code Postal	Ville	
<input type="text"/>	<input type="text"/>	
Email	Clé de preuve	
<input type="text"/>	<input type="text"/>	
Commentaires		
<input type="text"/>		
ENVOYER		

SPECIFICITES TECHNIQUES

- 11 champs, tous obligatoires, doivent être créés, dont 4 doivent être vérifiés.
- 1 champ doit être vérifié pour ne pas enregistrer un utilisateur déjà présent dans la BDD. Le Dirigeant de Spuro m'a demandé de vérifier si l'email existe déjà, car cette application est principalement destinée à une utilisation en interne (salariés d'une entreprise...).
- 3 des champs doivent être vérifiés, pour garantir la fiabilité des informations : 5 caractères pour le Code Postal, 15 caractères pour le n° de Sécurité Sociale, et 10 caractères pour le n° de téléphone.
- La clé de preuve doit être conservée uniquement par l'utilisateur, elle ne doit pas être enregistrée dans la BDD, par contre elle est essentielle pour créer l'empreinte.
- Liste des champs : nom, prénom, date de naissance, n° de Sécurité Sociale, n° de téléphone, adresse, code postal, ville, email, clé de preuve, commentaires.

DIAGRAMME



REALISATION DU PROJET

1- BASE DE DONNEES

- J'ai crée une Base de Données que j'ai appelée « bkp » et une table « utilisateurs ».
- Un ID se rajoute pour chaque nouvel utilisateur, en auto-incrémentation.
- Concernant les champs « secu », « code_postal » et « telephone » j'ai choisi le type VARCHAR et non INT, pour pouvoir contrôler et vérifier le nombre de caractères entrés dans les inputs et ainsi éviter les erreurs.
- Pour le champ « date_de_naissance », j'ai choisi le type DATE car l'enregistrement dans la BDD est fait au format JJ/MM/AAAA.
J'ai appelé ce champ « date_de_naissance » et non « date » car ce mot est utilisé dans le langage SQL (type de données que j'ai d'ailleurs utilisé), ce qui peut provoquer des erreurs avec d'autres systèmes de BDD comme Oracle.
- Tous les autres champs sont en type VARCHAR.

bkp utilisateurs	
id : int(11)	
nom : varchar(50)	
prenom : varchar(50)	
date_de_naissance : date	
secu : varchar(15)	
email : varchar(255)	
adresse : varchar(255)	
code_postal : varchar(5)	
ville : varchar(50)	
telephone : varchar(10)	
texte : varchar(255)	

2- CONNEXION A LA BDD

J'ai crée une page « connexion.php » avec tous les paramètres nécessaires pour effectuer la connexion à la BDD ci-dessus.

[Code de la page de connexion en ANNEXE 1](#)

3- FORMULAIRE

FRONT

Le formulaire est mis en forme avec le Framework Css BOOTSTRAP, ainsi qu'avec un peu de css.

J'ai utilisé la class « container » de Bootstrap pour centrer et rendre responsive le formulaire. Pour la mise en forme de celui-ci, j'ai utilisé la class «form-row» de Bootstrap pour aligner des inputs sur la même ligne.

Puis j'ai défini la largeur des colonnes : « form-group col-md-8 », « form-group col-md-6 » ou «form-group col-md-4 ».

« md » correspondant au breakpoint MEDIUM de Bootstrap c'est-à-dire une taille d'écran $\geq 768\text{px}$ (taille tablette).

La page du formulaire a nécessité peu de css. Les couleurs utilisées sont celles du logo de l'entreprise Spuro. [Code du formulaire en ANNEXE 2](#)

Voici le visuel :

The screenshot shows a user registration form titled "ENREGISTRER UN UTILISATEUR". The form consists of several input fields arranged in a grid-like structure. At the bottom right is a large black button labeled "ENVOYER".

BACK

Dans le formulaire j'ai défini la méthode :

```
<!-- FORMULAIRE -->
<form action="" method="post"> <!-- Valeur vide car même page -->
```

La méthode POST transmet les informations du formulaire de manière masquée mais non cryptée.

Concernant le code PHP, j'ai tout d'abord effectué =>

```
// Paramètres du framework
require_once 'inc/init.php';

// Connexion à la BDD
require_once 'connexion.php';
```

J'ai requis les paramètres du Framework pour pouvoir utiliser les fonctions déjà existantes, dont j'ai besoin pour l'écriture dans le Registre.

Je préfère utiliser « require_once » plutôt que « include » ... pour 2 raisons :

- PHP vérifie si le fichier a déjà été inclus, et si c'est le cas, ne l'inclut pas une deuxième fois.
- « require_once », à l'inverse de « include_once », va générer une erreur et stopper l'exécution du script PHP

INSERTION DANS LA BDD

J'ai fait une vérification des champs nécessaires avant de préparer la requête et de l'exécuter.

```
// Vérifie si l'adresse email n'est pas déjà enregistrée
$requete = $db->prepare("SELECT COUNT(id) AS nb_email FROM utilisateurs WHERE email = :email");
$requete->bindValue(':email', $email, PDO::PARAM_STR);
$requete->execute();

// Récupère le résultat de la requête ci-dessus
$resultat = $requete->fetch();

// Si l'adresse email n'existe pas...
if($resultat['nb_email'] == 0) {

    // ... On vérifie que le code postal fait 5 caractères
    if(strlen($_POST['code_postal']) == 5){

        // ... On vérifie que le n° de sécu fait 15 caractères
        if(strlen($_POST['securite']) == 15){

            // ... On vérifie que le n° de téléphone fait 10 caractères
            if(strlen($_POST['telephone']) == 10){

                // ... et on prépare la requête
                $requete = $db->prepare(
                    "INSERT INTO utilisateurs (nom, prenom, date_de_naissance, securite, email, adresse,
                    code_postal, ville, telephone, texte)
                    VALUES (:nom, :prenom, :date_de_naissance, :securite, :email, :adresse, :code_postal,
                    :ville, :telephone, :texte)"
                );
            }
        }
    }
}
```

Code du formulaire en ANNEXE 2bis

INSERTION DANS LE REGISTRE

La Blockchain Spuro ayant été créée par un Développeur extérieur, j'ai dû prendre en compte le code déjà créé. Pour pouvoir réaliser ce projet j'ai donc dû utiliser des fonctions déjà existantes :

- **generatePrint()** = générer l'empreinte
- **savePrint()** = sauvegarder l'empreinte
- **generateBlock()** = générer le Block
- **handleUpdate()** = mettre à jour le Registre
- **trim_utf8_encode()** = encoder les valeurs des inputs en utf8

J'ai dû effectuer des recherches pour comprendre la fonction **trim_utf8_encode()**

```
function trim_utf8_encode($s) {
    if (mb_detect_encoding($s, 'UTF-8', true) == 'UTF-8') return trim(addslashes($s));
    else return trim(addslashes(utf8_encode($s)));
}
```

J'ai trouvé les réponses sur www.php.net où se situe la documentation officielle de PHP.

mb_detect_encoding() : détecte l'encodage utilisé par la chaîne str et retourne l'encodage détecté ou FALSE si l'encodage ne peut être détecté.

trim(addslashes(\$value)) : ajoute des antislashes dans une chaîne et retourne la chaîne str, après avoir échappé les caractères qui doivent l'être (‘ « \ NUL).

Ici dans la fonction **trim_utf8_encode()** on demande si l'encodage est en UFT-8 auquel cas on retourne la chaîne en ajoutant des antislashes dans la chaîne de caractères.

Par contre si l'encodage n'est pas en UFT-8 on retourne la chaîne de caractère encodée en UFT-8 avec des antislashes.

Cette fonction est nécessaire car à cause des caractères spéciaux présents dans la langue française le moindre changement de caractère ou oubli d'accent changera le hash.

Pour écrire dans le Registre, voici le code que j'ai créé :

```
/*
 * Blockchain ---
 */

// Informations de base
$platformName = 'SpuroBKP';
$time = $fw->ledger->getTime();
$apiData = [
    'prenom' => urlencode($prenom),
    'nom' => urlencode($nom),
];

// Hash des informations voulues
$hash = hash('sha256', strtolower( $nom . $prenom . urlencode($date_de_naissance) . $securite . $email . $cle));
$hashHash = hash('sha256', $hash); // Signature électronique (hash du hash)

// Stocker des informations dans un tableau
$apiData = [
    'prenom' => urlencode($prenom),
    'nom' => urlencode($nom),
    'user_hash' => $hashHash
];

// Encode le tableau ci-dessus en JSON
$jsonData = json_encode($apiData);

// Stocke le hash, le timestamp et des données dans un tableau
$document = [
    'hash' => $hash,
    'date' => $time,
    'data' => $jsonData
];

// Génère un print et le sauvegarde
$print = $fw->ledger->generatePrint($hash, $time, $platformName);
$fw->ledger->savePrint($print, $document);

// Création du block et mise à jour du registre
$fw->ledger->saveBlock();
$fw->ledger->handleUpdate();

/*
 * Fin ---
 */
```

VERSION 2

Création d'un formulaire d'enregistrement avec écriture de l'empreinte dans la Blockchain. Le code du formulaire et le code PHP sont séparés sur deux pages différentes.

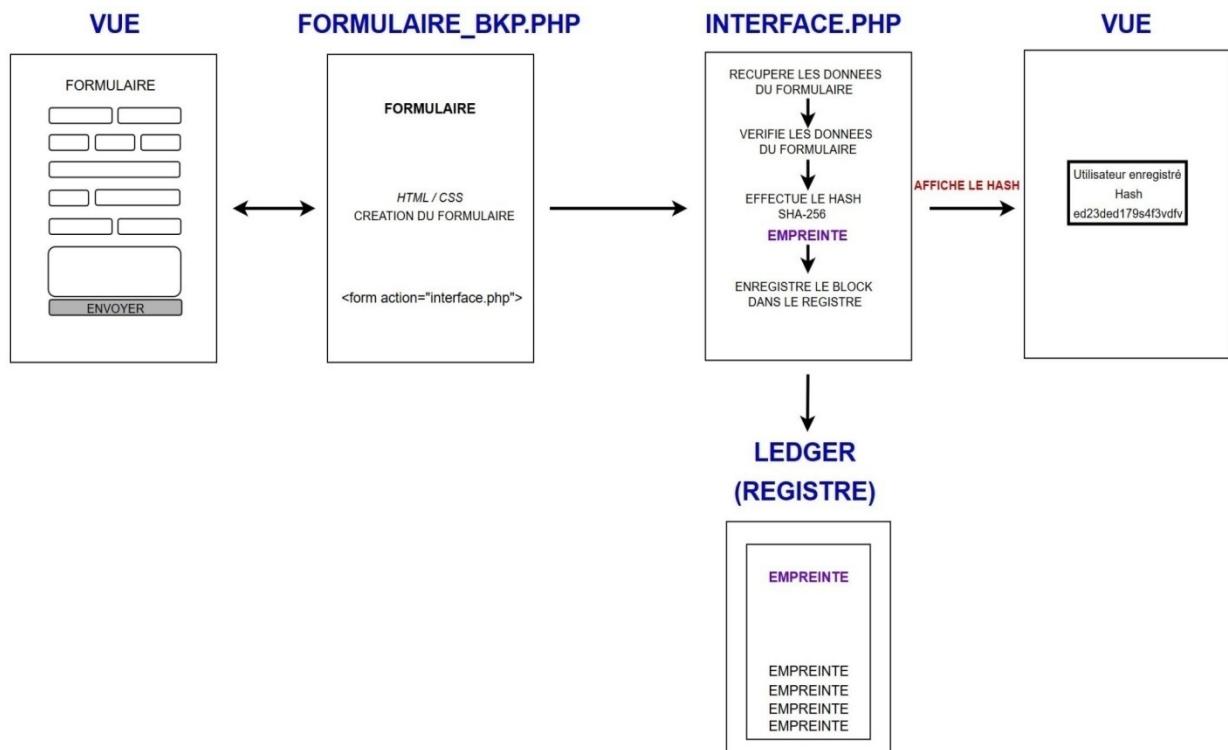
SPECIFICITES FONCTIONNELLES

Cette version est basée sur la première, sauf qu'il n'y a pas d'insertion dans une BDD.

Le formulaire d'enregistrement doit renvoyer vers une page PHP qui effectue l'écriture de l'empreinte numérique de l'utilisateur dans la Blockchain et affiche le résultat sur une autre page que celle du formulaire.

Cette application est destinée à être installée sur ordinateur.

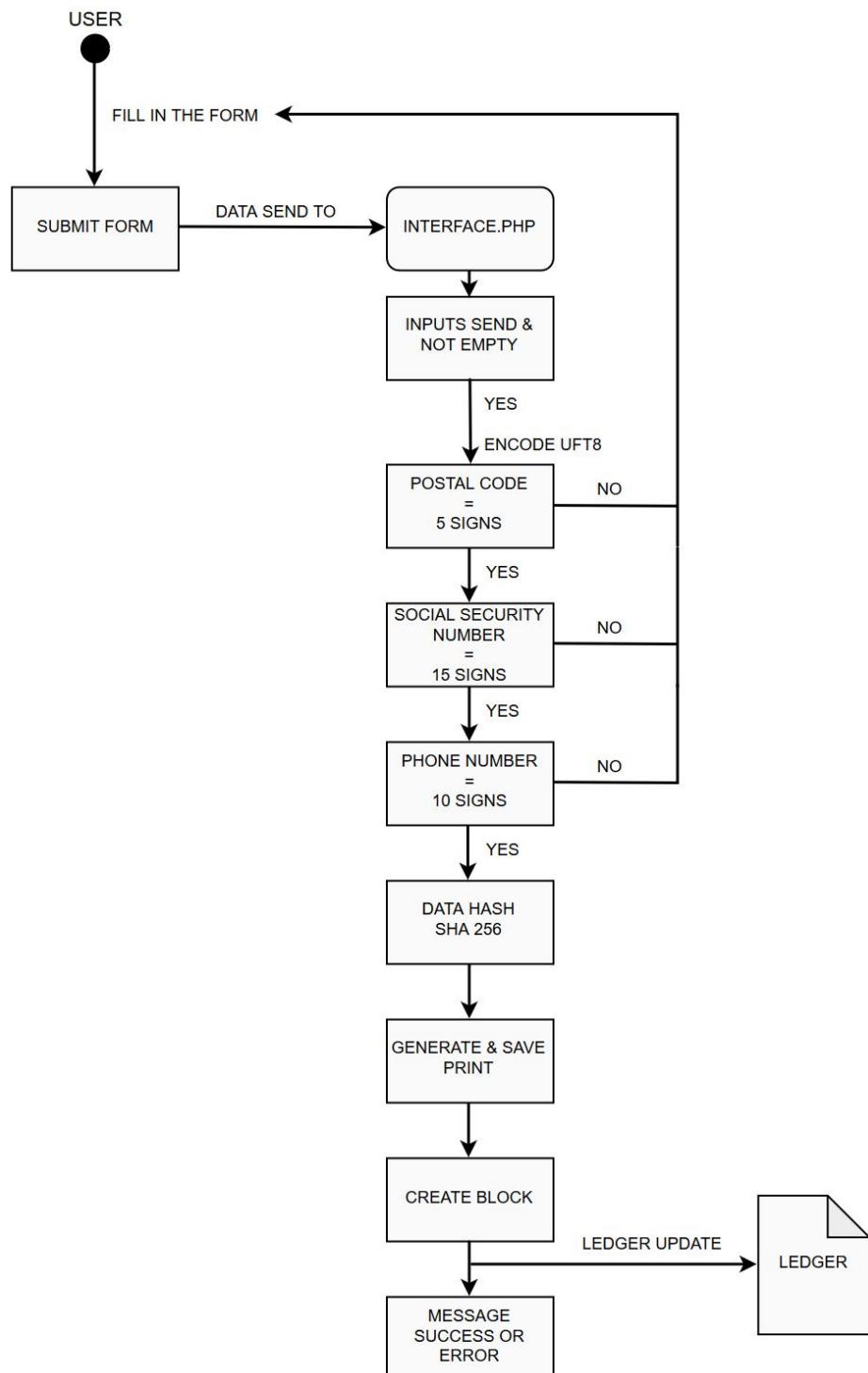
MAQUETTE DU PROJET A REALISER



SPECIFICITES TECHNIQUES

Les spécificités techniques sont identiques à la VERSION 1 (champs, vérification des champs...) sauf qu'il n'y pas d'insertion en BDD et que le code du formulaire et le code PHP doivent s'effectuer sur 2 pages différentes.

DIAGRAMME



Code du CSS en ANNEXE 3 et Visuel du formulaire en ANNEXE 4

Affichage du résultat sur une nouvelle page :



Mouse hover :



Côté BACK j'ai donc mis « action= “interface.php” » sur la balise <form> pour renvoyer sur la page interface.php, qui effectue la vérification des champs, le hachage des données , l’écriture dans le Registre puis affiche le message de réussite avec le hash de l’utilisateur.

La vérification des champs est identique à la Version 1.

VERSION 3

Création d'un formulaire d'enregistrement avec insertion des données de l'utilisateur en BDD sans recharger la page.

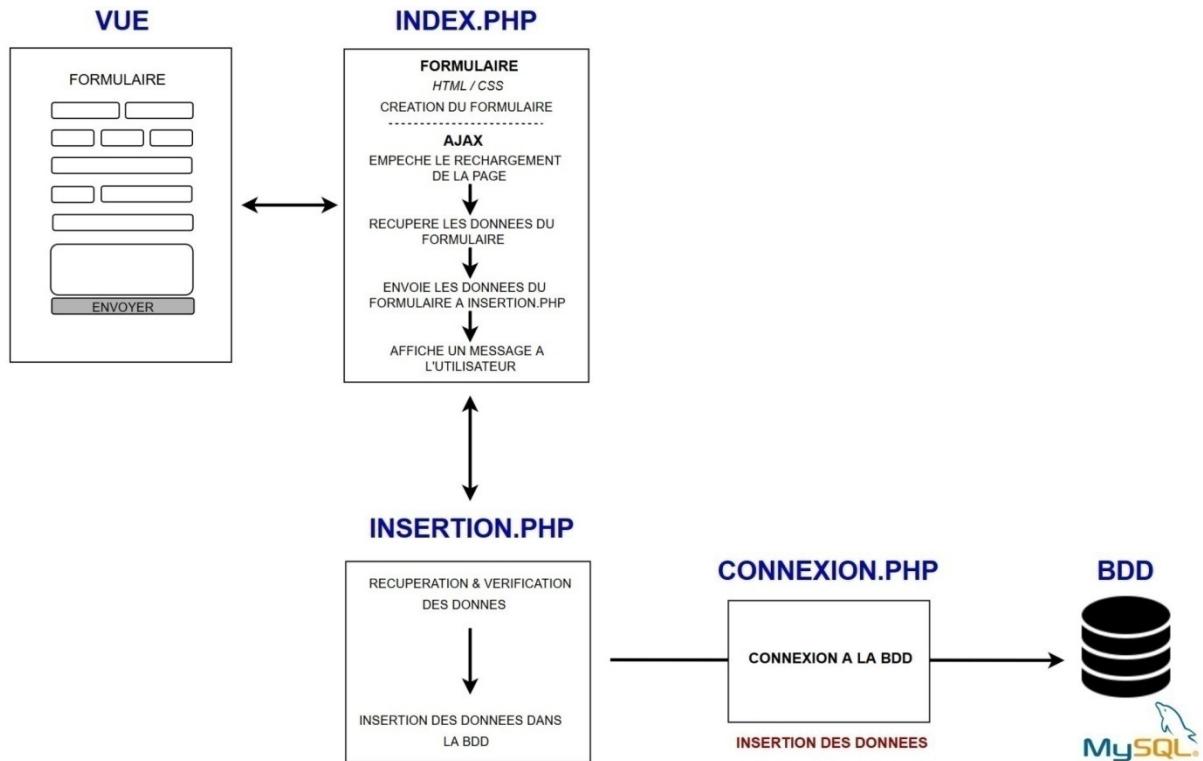
SPECIFICITES FONCTIONNELLES

Le principe est qu'une personne référente enregistre elle-même tous les utilisateurs et crée une signature électronique unique pour ces derniers. Par sécurité l'identité de la personne doit être vérifiée de visu. Lors de cet enregistrement dans les locaux de l'Autorité de Vérification (entreprise, mairie, assurance...), les informations personnelles de l'utilisateur sont enregistrées et conservées dans une BDD.

Ici, contrairement aux deux autres versions, il n'y a pas d'enregistrement dans la Blockchain. Il s'agit juste d'un formulaire pour enregistrer et sauvegarder les informations d'un utilisateur.

Cette application est destinée à être utilisée sur ordinateur.

MAQUETTE DU PROJET A REALISER

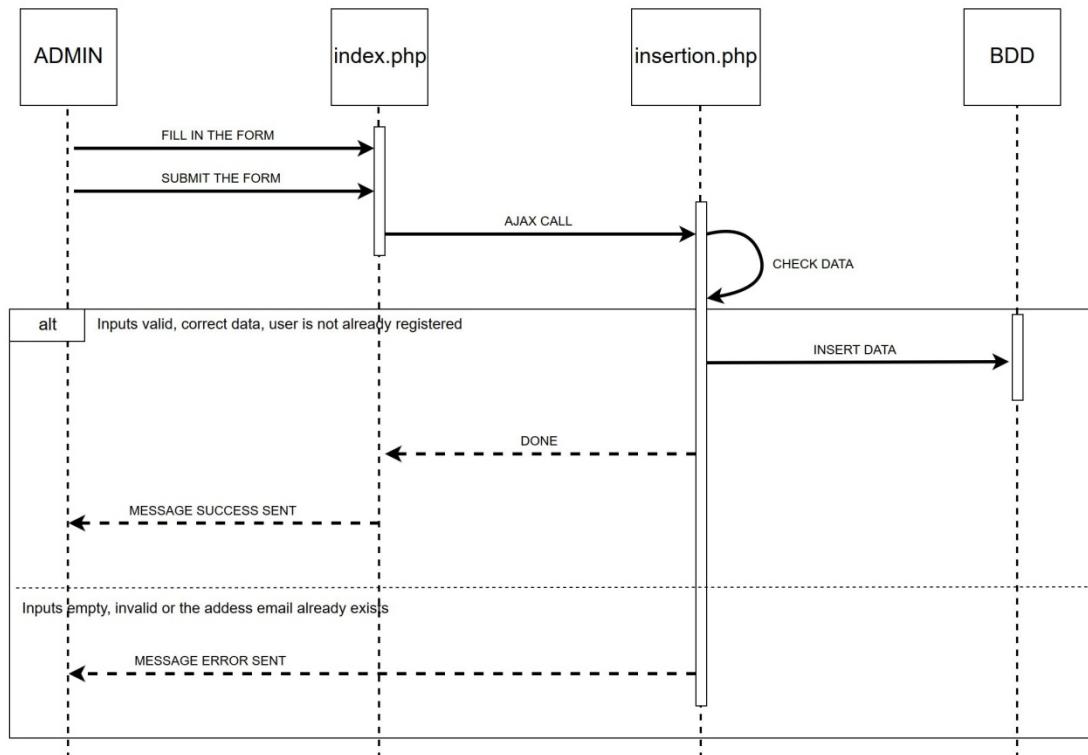


SPECIFICITES TECHNIQUES

Les spécificités techniques concernant les champs du formulaire sont identiques à la VERSION 1 sauf pour le champ « clé de preuve » qui n'est pas nécessaire puisque la clé de preuve doit être conservée uniquement par l'utilisateur.

En outre la page du formulaire ne doit pas se recharger lors de l'insertion des données.

DIAGRAMME DE SEQUENCE



REALISATION DU PROJET

1- BASE DE DONNEES

- J'ai créé une Base de Données que j'ai appelée « formulaire » et une table « enregister ».
 - Les types des champs sont identiques à la BDD de la VERSION 1.
 - Cette BDD pourra être développée avec la possibilité de mettre les utilisateurs dans différentes catégories : par exemple « Particulier », « Professionnel », « Employés »... Ceci peut être nécessaire pour certains clients qui souhaitent proposer la signature électronique à leurs salariés et à des personnes extérieures.
- J'ai donc développé une BDD de test pour proposer aux d'autres fonctionnalités aux clients, et leur donner une idée des possibilités de développement de l'application.

formulaire enregister	
id :	int(11)
nom :	varchar(55)
prenom :	varchar(55)
date_de_naissance :	date
secu :	varchar(15)
telephone :	varchar(10)
adresse :	varchar(255)
code_postal :	varchar(5)
ville :	varchar(55)
email :	varchar(255)
texte :	varchar(255)



2- CONNEXION A LA BDD

J'ai créé une page « connexion.php » avec tous les paramètres nécessaires pour effectuer la connexion à la BDD ci-dessus.

[Code de la page de connexion en ANNEXE 5](#)

3- FORMULAIRE ET SCRIPT AJAX

L'insertion et l'affichage des messages doivent se faire sans recharger la page, j'ai donc décidé d'utiliser de l'Ajax.

L'Ajax (Asynchronous Javascript and XML) est une combinaison de plusieurs technologies qui permet de récupérer des données sans à avoir à recharger la page.

J'ai décidé d'utiliser la bibliothèque JS Jquery car celle-ci simplifie l'utilisation de l'ajax.

RECHERCHES

J'ai dû effectuer de nombreuses recherches pour pouvoir utiliser Ajax, ne l'ayant pas appris lors de ma formation.

J'ai effectué des recherches sur le site de JQuery : <https://jquery.com>

Mais ce site étant parfois compliqué à comprendre, je me suis aussi basée sur :

https://sutterlity.gitbooks.io/apprendre-jquery/content/presentation_de_jquery.html

Il s'agit d'un manuel d'utilisation très complet et beaucoup plus simple, publié avec GitBook.

ECRITURE DU SCRIPT

Suite à mes recherches j'ai créé une page index.php où j'ai codé le formulaire et ajouté le script ajax. J'ai ajouté l'id #form sur le formulaire. L'ajax récupère l'id et empêche la page de se recharger. Ensuite j'ai utilisé la méthode `$.post()` cette requête permettant d'envoyer des données vers le serveur avec une requête HTTP POST. L'Ajax récupère la valeur des inputs et les envoie à la page insertion.php qui effectue la vérification des champs, prépare la requête et effectue l'insertion en BDD. [Code de la page insertion.php en ANNEXE 6](#)

Une fois la méthode terminée j'ai créé une fonction qui permet d'afficher soit un message d'erreur soit un message de réussite. J'ai utilisé `setTimeout()` pour afficher le message d'erreur puis qu'il disparaisse au bout de 10 secondes.

SCRIPT AJAX

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>

<script>

    // SCRIPT AJAX POUR EMPECHER LE RECHARGEMENT DE LA PAGE

    // On va chercher l'ID du formulaire
    $("#form").ready(function(){

        // Si le formulaire est validé (click sur le bouton)
        $("#submit").click(function(e){
            e.preventDefault(); // on empêche le bouton d'envoyer le formulaire

            // La méthode .post() permet d'envoyer des données vers le serveur avec une requête HTTP POST
            $.post(
                'insertion.php', // URL du script PHP qui traite l'insertion en BDD
                {
                    // Récupération de la valeur des inputs à passer à insertion.php
                    nom : $("#nom").val(),
                    prenom : $("#prenom").val(),
                    date_de_naissance : $("#date_de_naissance").val(),
                    secu : $("#secu").val(),
                    telephone : $("#telephone").val(),
                    adresse : $("#adresse").val(),
                    code_postal : $("#code_postal").val(),
                    ville : $("#ville").val(),
                    email : $("#email").val(),
                    texte : $("#texte").val()
                },
                // Indique une fonction à exécuter lorsque la méthode est terminée
                function(data){
                    // "data" contient les données résultant de la demande
                    if(data == 'Error'){
                        // ERROR : message si erreur
                        $("#resultat").html("<div>Erreur lors de la connexion</div>");
                        $('#resultat').fadeIn().html(data);
                        setTimeout(function() {
                            $('#resultat').fadeOut("slow");
                        }, 10000 );
                    }
                    else{
                        // SUCCESS : message si réussite
                        $("#resultat").html("<div>L'utilisateur a été enregistré avec succès !</div>");
                        $('#resultat').fadeIn().html(data);
                        setTimeout(function() {
                            $('#resultat').fadeOut("slow");
                        }, 10000 );
                    }
                },
                'text' // 
            );
        });
    });

</script>
```

TESTS DE LA VERSION 1

Quand on enregistre un utilisateur, les messages d'erreur doivent s'afficher si les champs ne sont pas correctement remplis :

Le N° de Sécurité Sociale doit faire 15 caractères !

ENREGISTRER UN UTILISATEUR

Nom _____ Prenom _____

Si tous les champs sont correctement remplis, un message de réussite doit s'afficher ainsi que le hash de l'utilisateur :

L'utilisateur est bien inscrit dans la Base De Données !

Votre hash : 89a8a5d1c532a5d2688161c7f4a2ce9768095549f6001554ebbef1e0eed49e36

ENREGISTRER UN UTILISATEUR

Nom _____ Prenom _____

L'utilisateur doit être enregistré dans la BDD :

✓ Affichage des lignes 0 - 0 (total de 1, traitement en 0,0025 seconde(s).)

SELECT * FROM `utilisateurs`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtre les lignes : Chercher dans cette tab

Options ↪ ↮ id nom prenom date_de_naissance secu email adresse code_postal ville telephone texte

Éditer Copier Supprimer 1 POTTER HARRY 2019-10-25 11111111111111 hp@test.fr rue des elfes 75000 Paris 0123456789 Hogwarts

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

	1	POTTER	HARRY	2019-10-25	11111111111111	hp@test.fr	rue des elfes	75000	Paris	0123456789	Hogwarts		

Le Block est enregistré dans le Registre :

```
438385d1aeb60e7366b68e807f2d14995e0623147bd9ae27d84e48e4a9d57c76 1572511290 data:{"prenom":"HARRY", "nom":"POTTER", "user_hash":"89a8a5d1c532a5d2688161c7f4a2ce9768095549f6001554ebbef1e0eed49e36"}  
70aecb2ec9a906e8ddd4678ddb44c0964c279d00ede387929c2365046a9222f 1572511290  
15c167af5eaf9a341516af07a62ab2adc1f030aab967f63bce625723a03aa5b9 1572511290
```

A la deuxième ligne, le hash de l'utilisateur (« user_hash ») correspond à celui communiqué lors du message de confirmation ci-dessus (2^{ème} photo , bandeau bleu) :

89a8a5d1c532a5d2688161c7f4a2ce9768095549f6001554ebbef1e0eed49e36

[Explication de l'écriture dans le Registre en ANNEXE 7](#)

VULNÉRABILITÉS DE SÉCURITÉ

GENERALITES

La veille concernant la sécurité est indispensable dans le secteur de la Blockchain, principalement à cause de l'utilisation fondamentale d'algorithmes de hachage cryptographique.

En effet certains algorithmes sont obsolètes ou peuvent le devenir rapidement à cause d'attaques et de l'apparition de nouveaux algorithmes plus sûrs.

ALGORITHMES PRINCIPAUX

ALGORITHME	DÉFINITION	SORTIE	STATUT
SHA-0	SHA-0 (1993) est le premier procédé d'une série créée par la NSA. Il n'a presque pas été utilisé puisque dès 1995 SHA-1 était publié pour corriger un défaut affectant la sécurité du chiffrement.	160 BITS	OBSOLÈTE Attaques de collision avérées
SHA-1	SHA-1 (1995) est la version légèrement modifiée de SHA-0. Il existe des attaques de collisions beaucoup plus faciles à réaliser que l'attaque des anniversaires, et réalisable par des attaquants disposant de moyens importants. SHA-1 a été très utilisé mais, à cause de ces attaques, il est recommandé d'utiliser SHA-256. De plus depuis 2017 les navigateurs de Microsoft, Google et Mozilla n'acceptent plus les certificats SHA-1.	160 BITS	OBSOLÈTE Attaques de collision avérées
MD5	MD5 est toujours très utilisée bien qu'il est recommandé d'utiliser des algorithmes plus robustes (SHA-256...) car des suites de collisions ont été découvertes. MD5 reste encore très utilisée comme outil de vérification et pour calculer l'empreinte d'un mot de passe avec la présence d'un sel permettant de ralentir une attaque par force brute. Au lieu de stocker les mots de passe dans un fichier, ce sont leurs empreintes MD5 qui sont enregistrées de sorte que quelqu'un qui lirait ce fichier ne pourrait pas découvrir les mots de passe.	128 BITS	UTILISE AVEC PRUDENCE (ajout de sel) Attaques de collision avérées

VERSIONS SHA-2

SHA-2 est une famille de fonctions de hachage cryptographique publiée en 2002. Les fonctions produisent des hashes de tailles différentes, désignées par le suffixe en bits. Elles utilisent des algorithmes très similaires inspirés de SHA-1. SHA256 et SHA512 sont deux grands standards utilisés actuellement car aucune attaque n'a encore révélé de failles de sécurité. Les attaques connues sur SHA-1 n'ont pas pu être réalisées sur SHA-2. **SHA-256 est l'algorithme le plus utilisé car sa vitesse de calcul est plus rapide que celles de SHA-384 et de SHA-512 étant donné que sa taille de sortie est plus courte.**

SHA-256	Fonction de hachage la plus utilisée dans le domaine de la Blockchain (Bitcoin...).	256 BITS	UTILISE STANDARD
SHA-384	Plus sécurisé que SHA-256, pourrait la remplacer en cas de faille de sécurité.	384 BITS	UTILISE
SHA-512	Plus sécurisé que SHA-256 et SHA-384, pourrait les remplacer en cas de faille de sécurité.	512 BITS	UTILISE STANDARD

PROPRIÉTÉS DES ALGORITHMES DE HACHAGE ET LES TYPES D'ATTAQUES QUI LEURS SONT LIÉES

PROPRIÉTÉS	ATTAQUES	SCHÉMA																
<p>FONCTION A SENS UNIQUE : Une fonction de hachage cryptographique est une fonction à sens unique. Le calcul de la fonction de hachage à partir d'une valeur d'entrée vers une valeur de sortie est facile et rapide alors qu'au contraire il est infaisable de trouver la valeur d'entrée à partir de la valeur de sortie.</p> <p>Les fonctions qui n'ont pas cette propriété sont vulnérables aux attaques de pré-image.</p>	<p>ATTAQUE DE PRE-IMAGE : - Attaque de pré-image. Il s'agit d'une attaque sur une fonction de hachage cryptographique dont le but est d'essayer de trouver une donnée d'entrée à partir d'une valeur spécifique de hachage. Par contraste avec une attaque de collision, dans l'attaque de pré-image la valeur du hash est spécifiée.</p> <p>- Attaque de seconde pré-image. Cette attaque consiste à essayer de trouver une seconde entrée qui a la même valeur de hachage qu'une entrée spécifique.</p>	<p>ATTAQUE DE PRE-IMAGE</p> <p>ENTREE ? → SORTIE VALEUR DU HASH d3885c74639d8c01 d10a3d26ea97d501 c746354f5aae2e761 0e3b0d7fe86e1d6</p> <p>ATTAQUE DE SECONDE PRE-IMAGE</p> <p>ENTREE VALEUR D'ENTREE N°1 → SORTIE VALEUR DU HASH "SPURO" → d3885c74639d8c01 d10a3d26ea97d501 c746354f5aae2e761 0e3b0d7fe86e1d6</p> <p>ENTREE VALEUR D'ENTREE N°2 → ?</p>																
<p>RÉSISTANCE AUX COLLISIONS : Deux valeurs d'entrées différentes ne peuvent avoir le même résultat de sortie. Pour obtenir cette propriété, il faut une valeur de hachage au moins deux fois plus longue que celle requise pour obtenir la résistance à la pré-image.</p> <p>- La résistance à la collision implique la résistance à la seconde pré-image, mais ne garantit pas la résistance à la pré-image.</p> <p>- La résistance aux collisions ne signifie pas qu'il n'y a pas de collisions, mais seulement que les collisions sont longues et difficiles à trouver.</p>	<p>COLLISION : Une attaque de collision sur un hash cryptographique est le fait d'essayer de trouver deux entrées produisant la même valeur de hash. Une attaque de collisions est plus facile à réaliser qu'une attaque de pré-image., car elle n'est pas limitée par une valeur spécifique (l'entrée ou la sortie peuvent être utilisées pour créer une collision).</p> <p>SUITE DE COLLISIONS : Cette attaque est liée au caractère itéré de la fonction et est très difficile à éviter. Dès que des collisions sont trouvées, on peut fabriquer un nombre exponentiellement grand de données différentes avec la même empreinte.</p>	<p>ATTAQUE DE COLLISION</p> <p>ENTREE VALEUR D'ENTREE N°1 → SORTIE VALEUR DU HASH Smiley face → d3885c74639d8c01 d10a3d26ea97d501 c746354f5aae2e761 0e3b0d7fe86e1d6 ✓</p> <p>ENTREE VALEUR D'ENTREE N°2 → SORTIE VALEUR DU HASH Skull and crossbones → d3885c74639d8c01 d10a3d26ea97d501 c746354f5aae2e761 0e3b0d7fe86e1d6 ✗</p> <p>DONNEES DIFFERENTES HASHES IDENTIQUES</p>																
<p>- Si une autre méthode est plus facile que cette attaque par force brute pour trouver deux sorties identiques, la fonction de hachage est considérée comme inadéquate pour la cryptographique.</p> <p>- L'attaque de l'anniversaire est l'attaque générique de référence pour déterminer la force d'un algorithme de hachage.</p>	<p>ATTAQUE D'ANNIVERSAIRE : Cette attaque démontre la limite de la résistance aux collisions et est basée sur le paradoxe d'anniversaire : la probabilité que dans un groupe de 'n' individus choisis aléatoirement, certains vont avoir la même date de naissance. Concernant les attaques de fonctions de hachage, cela signifie qu'on a au minimum 50 % de chance de casser la résistance à la collision.</p>	<p>ATTAQUE D'ANNIVERSAIRE</p> <table border="1"> <thead> <tr> <th>Taille de la sortie en bits</th> <th>Nombre d'entrées à essayer</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>13</td> </tr> <tr> <td>16</td> <td>213</td> </tr> <tr> <td>32</td> <td>54562</td> </tr> <tr> <td>64</td> <td>$9,6 \times 10^9$</td> </tr> <tr> <td>128</td> <td>$1,5 \times 10^{19}$</td> </tr> <tr> <td>160</td> <td>$1,0 \times 10^{24}$</td> </tr> <tr> <td>256</td> <td>$2,8 \times 10^{38}$</td> </tr> </tbody> </table> <p>Nombre d'entrées à essayer avant de trouver une collision avec une probabilité de 50 % minimum</p>	Taille de la sortie en bits	Nombre d'entrées à essayer	8	13	16	213	32	54562	64	$9,6 \times 10^9$	128	$1,5 \times 10^{19}$	160	$1,0 \times 10^{24}$	256	$2,8 \times 10^{38}$
Taille de la sortie en bits	Nombre d'entrées à essayer																	
8	13																	
16	213																	
32	54562																	
64	$9,6 \times 10^9$																	
128	$1,5 \times 10^{19}$																	
160	$1,0 \times 10^{24}$																	
256	$2,8 \times 10^{38}$																	

AUTRES PROPRIÉTÉS DES FONCTIONS DE HACHAGE

RÉSISTANCE AUX ATTAQUES :

La seule façon de trouver la valeur d'entrée d'une empreinte étant d'utiliser la méthode de « force brute » qui consiste à essayer toutes les valeurs d'entrée possibles jusqu'à trouver la bonne.

RÉSULTAT DE SORTIE CONSTANT :

Pour être considérée comme sécurisée, une fonction de hachage cryptographique doit toujours obtenir le même résultat quelque soit le nombre de fois que la même entrée est hachée : si les données d'entrée sont identiques, alors le résultat doit toujours être le même.

VITESSE DE CALCUL :

Le temps de calcul de la fonction est lié à sa taille de sortie. Une taille de sortie plus grande signifie plus de sécurité mais aussi plus de temps de calcul (donc un coût de calcul plus important), ainsi qu'un poids de sortie plus élevé.

SALAGE

Le sel est un nombre ou une chaîne de caractères totalement aléatoire (nonce). Le sel peut être ajouté avant ou après les données qui seront ensuite hachées (le résultat étant différent en fonction de la position du sel).

Le salage peut être effectué sur le résultat d'une concaténation de données avec une autre chaîne de caractères aléatoire. Les deux fonctions de hachage (celle qu'on utilise pour générer la chaîne aléatoire et celle qu'on applique au résultat) peuvent être différentes (par ex. SHA-256 et MD5). Cela permet de renforcer la sécurité de la fonction de hachage.

Sans salage, il est possible d'utiliser des tables de hachage pour attaquer les fonctions. L'ajout d'un sel rend l'utilisation de ces tables caduque, et le cassage doit faire appel à des attaques par force brute : cette méthode, consistant à tester toutes les valeurs possibles, prend beaucoup de temps et nécessite beaucoup de moyens.

CONCLUSION

Les fonctions de hachages cryptographiques sont conçues pour être mathématiquement **très difficiles ou impossibles** à inverser directement c'est-à-dire **techniquement impossible en pratique et en un temps raisonnable**. Les attaquants tenteront donc généralement des attaques par force brute.

Afin d'augmenter les temps de cassage, les fonctions de hachage cryptographiques sont souvent conçues pour présenter une charge de calcul suffisamment faible pour ne pas être gênante en usage normal mais suffisamment importante pour compliquer les attaques par force brute.

Actuellement on considère que pour obtenir **un niveau de sécurité correct il faut choisir une taille de sortie de 256 bits minimum.**

RECHERCHE EN ANGLAIS

INTRODUCTION

Pour le projet j'ai dû me familiariser avec plusieurs concepts surtout au niveau de la sécurité. La difficulté principale concernant les failles de sécurité repose sur l'utilisation de fonctions de hachage et donc d'algorithmes potentiellement faillibles. **En effet les Blockchains, quelque soit leur domaine, utilisent des fonctions de hachage.**

J'ai donc effectué des recherches sur les failles de sécurité liées aux fonctions de hachage. Une des recherches que j'ai effectuée sur des sites anglophones concerne les différents types d'attaques sur les algorithmes de hachage.

RECHERCHE ET RESULTATS

J'ai effectué la recherche suivante :



Cinq premiers résultats de la recherche en ANNEXE 8

CHOIX DU RESULTAT

Le premier résultat est un article de Toshendra Sharma sur le site Records Keeper, une entreprise qui propose une plateforme de stockage reposant sur une Blockchain.

Toshendra Sharma est le créateur et le CEO de Records Kepper, ainsi que le Directeur Exécutif de BLOCKCHAIN COUNCIL, un organisme regroupant des experts de la Blockchain. Il est également Consultant et formateur Blockchain et possède un Master of Technology (Engineering) en Cybersécurité.

J'ai donc choisi ce résultat car les autres sont : le site de wikipedia, un article de blog, un site regroupant des publications scientifiques très techniques, ou bien des publications anciennes (antérieures à 2016).

ETUDE DU TEXTE

Cet article est très intéressant car il explique ce qu'est le hachage et l'attaque de fonction de hachage. Il aborde également les différents types d'attaques tout en les expliquant, ce qui m'a permis de faire le tableau avec les types d'attaques en page 29.

L'auteur parle aussi de la sécurité en général des fonctions de hachage et à quel point elles sont sûres. Il confirme également que la fonction de hachage que j'ai utilisée pour mon projet, c'est-à-dire SHA-256, est résistante aux collisions pour l'instant.

TRADUCTION DU TEXTE

<p>Blockchain technology is one of the most innovative discoveries in recent years. One of its core principles is the hash function.</p>	<p>La technologie Blockchain est une des découvertes les plus innovantes de ces dernières années. Un de ses principes fondamentaux est la fonction de hachage.</p>
<p><u>What is Hashing?</u></p> <p>Hashing is, simply put, taking an input string of any length and giving out an output of a fixed length. Cryptographic hashing refers a special class of hash functions with set properties. To be considered secure, a cryptographic hash function needs to include properties such as always getting a consistent result irrespective of how many times you parse through an input, quick computation, and pre-image resistant among others.</p> <p>In case of cryptocurrencies such as bitcoin, the transactions are taken as an input and run through a hashing algorithm (Bitcoin uses SHA-256) which gives an output of a fixed length.</p> <p>Each input has its own unique hash. For examples, take inputs A and B where H(A) and H(B) are their respective hashes. It is infeasible for H(A) to be equal to H(B). Infeasible but unfortunately, not impossible.</p>	<p><u>Qu'est-ce que le hachage ?</u></p> <p>Le hachage est, dit simplement, le fait de prendre le string d'une entrée de n'importe quelle taille et de restituer une sortie de taille fixe.</p> <p>Le hachage cryptographique fait référence à une classe spéciale de fonctions de hachage avec des propriétés prédefinies. Pour être considérée comme sécurisée, une fonction de hachage cryptographique doit inclure des propriétés telles que le fait de toujours obtenir un résultat constant quelque soit le nombre de fois que vous analysez une même entrée, la rapidité de calcul, et une résistance aux attaques de pré-image, entre autres.</p> <p>Dans le cas des cryptomonnaies comme le Bitcoin, les transactions sont prises en entrée et passées à travers un algorithme de hachage (le Bitcoin utilise le SHA-256), qui donne une sortie de taille fixe.</p> <p>Chaque entrée a son propre hash unique. Par exemple, prenons les entrées A et B où H(A) et H(B) sont leurs hashs respectifs. Il est infaisable pour H(A) d'être égal à H(B). Infaisable mais malheureusement pas impossible.</p>
<p><u>What is a Hash Function Attack?</u></p> <p>A hash function attack is an attempt to find two input strings of a hash function that produce the same hash result. Because hash functions have infinite input length and a predefined output length, there is inevitably going to be the possibility of two different inputs that produce the same output hash.</p> <p>A hash collision occurs when two separate inputs produce the same hash output. This can be exploited by an application that compares two hashes together (such as password hashes, file integrity checks). However, the odds of a collision are extremely low, especially for functions with a large output size such as lengthy and widespread document formats or protocols but as available computational power increases, the ability to attack hash functions becomes more feasible.</p>	<p><u>Qu'est-ce qu'une attaque de fonction de hachage ?</u></p> <p>Une attaque d'une fonction de hachage est une tentative de trouver deux strings d'entrée d'une fonction de hachage qui produiront le même résultat de hash. Parce que les fonctions de hachage ont des longueurs d'entrée infinies et une longueur de sortie prédefinie, il y a inévitablement la possibilité que deux entrées différentes produisent le même hash en sortie.</p> <p>Ceci peut être exploité par une application qui compare ensemble deux hashs (tels que les hashs de mots de passe, les vérifications d'intégrité des fichiers).</p> <p>Néanmoins les probabilités d'une collision sont extrêmement basses, surtout pour les fonctions avec une grande taille de sortie comme les très longs et très répandus formats de document ou de protocoles, mais en même temps que la puissance disponible de calcul augmente, la capacité d'attaquer les fonctions de hachage devient plus faisable.</p>
<p><u>How Does a Hash Function Attack Occur?</u></p> <p>There are several ways a hash collision could be exploited. There are mainly three types of hash function attacks:</p>	<p><u>Comment une attaque sur une fonction de hachage se produit ?</u></p> <p>Il y a différentes façons d'exploiter une collision de hash. Il y a trois types d'attaques principales des fonctions de hachage :</p>

<p><u>Collision attack:</u> A collision attack on a cryptographic hash tries to find two inputs producing the same hash value. The attacker does not have control over the content of the message, but they are arbitrarily chosen by the algorithm. In this case, $H(A)$ is equal to $H(B)$.</p>	<p><u>Attaque de Collision :</u> Une attaque de collision sur un hash cryptographique essaye de trouver deux entrées produisant la même valeur de hash. L'attaquant n'a pas le contrôle sur le contenu du message, mais celui-ci est choisi arbitrairement par l'algorithme. Dans ce cas $H(A)$ est égal à $H(B)$.</p>
<p><u>Pre-image attack:</u> In contrast to a collision attack, in a pre-image attack the hash value is specified.</p>	<p><u>Attaque de pré-image :</u> Par contraste avec une attaque de collision, dans l'attaque de pré-image la valeur du hash est spécifiée.</p>
<p><u>Birthday attack:</u> The birthday attack is based on the birthday paradox, i.e., the probability that in a set of n randomly chosen people, some pair of them will have the same birthday. Applied to hash function attacks, this means you have a 50% chance to break the collision resistance.</p>	<p><u>Attaque d'anniversaire :</u> L'attaque d'anniversaire est basée sur le paradoxe d'anniversaire, c'est-à-dire la probabilité que dans un groupe de 'n' personnes choisies aléatoirement, quelques paires d'entre elles vont avoir la même date de naissance. Appliquée sur les attaques de fonctions de hachage, cela signifie que l'on a 50 % de chance de casser la résistance à la collision.</p>
<p><u>How Secure are Hash Functions?</u></p> <p>No hash function is collision free, but it usually takes extremely long to find a collision.</p> <p>Even if a hash function has never been broken, a successful attack against a weakened variant may undermine the experts' confidence and lead to its abandonment. In the past, weaknesses had been found in several then-popular hash functions, including SHA-0, RIPEMD, and MD5. These weaknesses called into question the security of stronger algorithms derived from the weak hash functions such as the SHA-1, RIPEMD-128, and RIPEMD-160.</p> <p>Also, there are applications of cryptographic hash functions that do not rely on collision resistance. This means that collision attacks do not affect their security. For example, HMACs are not vulnerable. For the hash attack to be successful, the attacker must be in control of the input to the hash function.</p>	<p><u>A quel point les fonctions de hachage sont-elles sûres ?</u></p> <p>Aucune fonction de hachage n'est sans risque de collision, mais habituellement cela prend très longtemps pour en trouver une.</p> <p>Même si une fonction de hachage n'a jamais été cassée, une attaque réussie contre une variante plus faible peut saper la confiance des experts et mener à son abandon. Dans le passé des failles ont été trouvées dans plusieurs fonctions de hachage populaires à l'époque, incluant SHA-0, RIPEMD et MD5. Ces failles ont remis en question la sécurité d'algorithmes plus résistants mais dérivés de fonctions de hachage faibles comme SHA-1, RIPEMD-128 et RIPEMD-160.</p> <p>En outre, il y a des applications de fonctions de hachage cryptographique qui ne reposent pas sur les résistances aux collisions. Cela signifie que les attaques de collision n'affectent pas leur sécurité. Par exemple les HMACs* ne sont pas vulnérables. Pour que les attaques de hachage soient réussies, l'attaquant doit avoir le contrôle de l'entrée de la fonction de hachage.</p>
<p><u>Are hash function attacks something to worry about?</u></p> <p>The truth is that it depends on the hash function. Even MD5 and SHA-1 are not completely collision resistant but stronger functions such as SHA-256 appear to be safe for now.</p>	<p><u>Est-ce qu'il faut s'inquiéter des attaques sur les fonctions de hachage ?</u></p> <p>La vérité est que cela dépend de la fonction de hachage. Même MD5 et SHA-1 ne sont pas complètement résistants aux collisions mais des fonctions plus résistantes comme SHA-256 apparaissent sûres pour l'instant.</p>

* **HMAC** : type de code d'authentification de message, calculé en utilisant une fonction de hachage cryptographique, en combinaison avec une clé secrète.

SYNTHESE ET CONCLUSION

Ce projet est appelé à évoluer en fonction des demandes des clients, la Blockchain Spuro étant une Blockchain Framework.

Pour ce projet j'ai dû effectuer de nombreuses recherches, la Blockchain étant un domaine que je connaissais déjà mais pas suffisamment. En outre j'ai dû utiliser des technologies pour la première fois (Ajax) et améliorer mes compétences sur d'autres (fonction de hachage).

Une des difficultés principales que j'ai rencontrée concernant la recherche sur la Blockchain est la difficulté de trouver les informations que je souhaitais. En effet les Blochains peuvent être écrites dans différents langages de programmation (PHP, JS, C#...) et beaucoup de résultats de recherches concernent les cryptomonnaies (Bitcoin..) ou la Blockchain Ethereum.

J'ai également réalisé que le domaine de la Blockchain impose une veille de sécurité régulière pour garantir l'intégrité de celle-ci (attaques de fonctions de hachage).

ANNEXES

ANNEXE 1

PAGE DE CONNEXION VERSION 1

```
<?php

//PARAMETRES DE CONNEXION A LA BDD
define('HOST', 'localhost');
define('BDD', 'bkp');
define('USER', 'root');
define('PASSWD', '');

try{
    $db = new PDO('mysql:host=' . HOST . ';dbname=' . BDD, USER, PASSWD, array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::MYSQL_ATTR_INIT_COMMAND =>"SET NAMES 'utf8'"
    ));

}catch(PDOException $error){
    throw new PDOException("La connexion à la base de données n'a pas fonctionné");
}

?>
```

ANNEXE 2

CODE DU FORMULAIRE VERSION 1

```
<!-- FORMULAIRE -->
<form action="" method="post"> <!-- Valeur vide ca

    <div class="form-row"> <!-- form-row-->
        <div class="form-group col-md-6">
            <label for="nom">Nom</label>
            <input type="text" name="nom" id="nom"
        </div>

        <div class="form-group col-md-6">
            <label for="prenom">Prénom</label>
            <input type="text" name="prenom" id="p
        </div>
    </div> <!-- Fin form-row-->

    <div class="form-row"> <!-- form-row-->
        <div class="form-group col-md-4">
            <label for="date_de_naissance">Date de
            <input type="date" name="date_de_naiss
        </div>

        <div class="form-group col-md-4">
            <label for="secu">N° Sécurité Sociale<
            <input type="text" name="secu" id="sec
        </div>

        <div class="form-group col-md-4">
            <label for="telephone">N° téléphone</l
            <input type="text" name="telephone" id
        </div>
    </div> <!-- Fin form-row-->

    <div class="form-group">
        <label for="adresse">Adresse</label>
```

ANNEXE 2bis

CODE PHP VERSION 1

```
// Formulaire envoyé
if(isset($p['submit'])){

    // Vérification des champs (envoyés et non vides)
    if(isset($p['nom']) && !empty($p['nom']))
        && isset($p['prenom']) && !empty($p['prenom'])
        && isset($p['date_de_naissance']) && !empty($p['date_de_naissance'])
        && isset($p['secu']) && !empty($p['secu'])
        && isset($p['email']) && !empty($p['email'])
        && isset($p['adresse']) && !empty($p['adresse'])
        && isset($p['code_postal']) && !empty($p['code_postal'])
        && isset($p['ville']) && !empty($p['ville'])
        && isset($p['telephone']) && !empty($p['telephone'])
        && isset($p['texte']) && !empty($p['texte'])
        && isset($p['cle']) && !empty($p['cle'])

    }

    // Encodage utf8
    $nom = trim_utf8_encode($p['nom']);
    $prenom = trim_utf8_encode($p['prenom']);
    $date_de_naissance = trim_utf8_encode($p['date_de_naissance']);
    $secu = trim_utf8_encode($p['secu']);
    $email = trim_utf8_encode($p['email']);
    $adresse = trim_utf8_encode($p['adresse']);
    $code_postal = trim_utf8_encode($p['code_postal']);
    $ville = trim_utf8_encode($p['ville']);
    $telephone = trim_utf8_encode($p['telephone']);
    $texte = trim_utf8_encode($p['texte']);
    $cle = trim_utf8_encode($p['cle']);
```

ANNEXE 3

CSS VERSION 2

```
/* CSS INTERFACE.PHP */
#interface{
    max-height: 60vh;
    margin-top: 15%;
    border: 3px solid black;
    background-color: #bcddec;
    box-shadow: 0 5px 10px 1px rgba(0, 0, 0, 0.6);
}

.message{
    font-size: 1.1em;
}

#interface:hover {
    background-color: rgba(197, 208, 255, 0.438);
    transform: scale(1.05);
    -webkit-transform: scale(1.05);
    -moz-transform: scale(1.05);
    -o-transform: scale(1.05);
    -ms-transform: scale(1.05);
}

#interface:hover .resultat{
    background-color: rgb(255, 255, 255);
    border: 2px dotted black;
    padding: 0.1em;
}
```

```
/* MEDIA QUERIES */

/* max 991.99px */
@media screen and (max-width: 991.99px){
    .message{
        font-size: 0.8em;
    }
    #interface{
        min-width: 80vw;
    }
}

/* 992px à 1199.99px */
@media screen and (min-width: 992px) and (max-width: 1199.99px){
    .message{
        font-size: 1em;
    }
}

/* 1200px à 1799.99px */
@media screen and (min-width: 1200px) and (max-width: 1799.99px){
    #interface{
        max-height: 50vh;
        max-width: 90vw;
    }
    .row{
        height: 50vh;
        margin-top: 25vh;
        margin-bottom: 25vh;
    }
    .message{
        font-size: 1.1em;
    }
}
```

ANNEXE 4

VISUEL FORMULAIRE VERSION 2

ENREGISTRER UN UTILISATEUR

TOUS LES CHAMPS SONT OBLIGATOIRES

Nom	Prenom	
Date de naissance N° Sécurité Sociale N° téléphone		
jj / mm / aaaa		
Adresse		
Code Postal	Ville	
Email	Clé de preuve	
Commentaires		
ENVOYER		

ANNEXE 5

CODE PAGE CONNEXION VERSION 3

```
<?php

//PARAMETRES DE CONNEXION A LA BDD
define('HOST', 'localhost');
define('BDD', 'formulaire');
define('USER', 'root');
define('PASSWD', '');

try{
    $db = new PDO('mysql:host=' . HOST . ';dbname=' . BDD, USER, PASSWD, array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::MYSQL_ATTR_INIT_COMMAND =>"SET NAMES 'utf8'"
    ));
}

}catch(PDOException $error){
    throw new PDOException("La connexion à la base de données n'a pas fonctionné");
}

?>
```

ANNEXE 6

INSERTION.PH VERSION 3

```
// Connexion à la BDD
require_once 'connexion.php';

// Vérification des champs (envoyés et non vides)
if(isset($_POST['nom']) && !empty($_POST['nom'])
    && isset($_POST['prenom']) && !empty($_POST['prenom'])
    && isset($_POST['date_de_naissance']) && !empty($_POST['date_de_naissance'])
    && isset($_POST['secu']) && !empty($_POST['secu'])
    && isset($_POST['telephone']) && !empty($_POST['telephone'])
    && isset($_POST['adresse']) && !empty($_POST['adresse'])
    && isset($_POST['code_postal']) && !empty($_POST['code_postal'])
    && isset($_POST['ville']) && !empty($_POST['ville'])
    && isset($_POST['email']) && !empty($_POST['email'])
    && isset($_POST['texte']) && !empty($_POST['texte']))
){

    // Récupération des données dans des variables
    $nom = $_POST['nom'];
    $prenom = $_POST['prenom'];
    $date_de_naissance = $_POST['date_de_naissance'];
    $secu = $_POST['secu'];
    $telephone = $_POST['telephone'];
    $adresse = $_POST['adresse'];
    $code_postal = $_POST['code_postal'];
    $ville = $_POST['ville'];
    $email = $_POST['email'];
    $texte = $_POST['texte'];

    // Vérifie si l'adresse email n'est pas déjà enregistrée
    $requete = $db->prepare("SELECT COUNT(id) AS nb_email FROM enregistrer WHERE email = :email");
    $requete->bindValue(':email', $email, PDO::PARAM_STR);
    $requete->execute();
}
```

```
// Récupère le résultat de la requête ci-dessus
$resultat = $requete->fetch();

// Si l'adresse email n'existe pas...
if($resultat['nb_email'] == 0) {

    // ... On vérifie que le code postal fait 5 caractères
    if(strlen($_POST['code_postal']) == 5){

        // ... On vérifie que le n° de sécu fait 15 caractères
        if(strlen($_POST['secu']) == 15){

            // ... On vérifie que le n° de téléphone fait 10 caractères
            if(strlen($_POST['telephone']) == 10){

                // ... et on prépare la requête
                $requete = $db->prepare(
                    "INSERT INTO enregistrer (nom, prenom, date_de_naissance, secu, telephone, adresse, code_postal,
                     ville, email, texte) VALUES (:nom, :prenom, :date_de_naissance, :secu, :telephone, :adresse,
                     :code_postal, :ville, :email, :texte)
                ");

                $requete->bindValue(':nom', $nom, PDO::PARAM_STR);
                $requete->bindValue(':prenom', $prenom, PDO::PARAM_STR);
                $requete->bindValue(':date_de_naissance', $date_de_naissance, PDO::PARAM_STR);
                $requete->bindValue(':secu', $secu, PDO::PARAM_STR);
                $requete->bindValue(':telephone', $telephone, PDO::PARAM_STR);
                $requete->bindValue(':adresse', $adresse, PDO::PARAM_STR);
                $requete->bindValue(':code_postal', $code_postal, PDO::PARAM_STR);
                $requete->bindValue(':ville', $ville, PDO::PARAM_STR);
                $requete->bindValue(':email', $email, PDO::PARAM_STR);
                $requete->bindValue(':texte', $texte, PDO::PARAM_STR);
            }
        }
    }
}
```

```
    $requete->execute();

    echo '<div class="alert alert-success">L\'utilisateur est bien inscrit dans la Base De Données !</div>';

}
else{
    echo '<div class="alert alert-danger">Le N° de téléphone doit faire 10 caractères !</div>';
}
}
else{
    echo '<div class="alert alert-danger">Le N° de Sécurité Sociale doit faire 15 caractères !</div>';
}
else{
    echo '<div class="alert alert-danger">Le Code Postal n\'est pas au bon format !</div>';
}
else {
    echo '<div class="alert alert-danger">Cette adresse email existe déjà !</div>';
}
else{
    echo '<div class="alert alert-danger">L\'enregistrement a échoué !</div>';
}
```

ANNEXE 7

EXPLICATION DE L'ECRITURE DANS LE REGISTRE

Le Registre est hébergé sur chaque ordinateur faisant partie du réseau de la Blockchain.
Il est mis à jour en temps réel (REGISTRE GLOBAL). Mais concernant la Blockchain Spuro il y a également un Registre Local qui me permet d'effectuer des tests sur serveur local.

```
b3035dc54c4a1cedec708d6f8c17bb6d3fd99ba119b8cb3bd9608c8c47fb7672 1571398652 data:{ "prenom": "X", "nom": "X",  
"user_hash": "dbb77b6e593549d34e38f7b43558bc67b986d1fd6aeedb10dce65816d96d2e88"}  
06cdffd13a8028933fbdf86175b913fb100d38245d39561613307feee9dd7c1d 1571398652  
733453c6144b78deba215f7e32ec1b1bdb3e4ae1e9b44d8515bafe223e697333 1571398652  
ceb35704a6dcf9068a7d1837d34734102f1813061634c8321b954fe3f3c874d5 1571398943 data:{ "prenom": "W", "nom": "W",  
"user_hash": "d8b705604618848eef3754f6f2fa7bedb3af5ad4ab0105b7306733bc5a7e39b62"}  
4c43bec19e8032bfce4ade8aa8e869de2940e7ceccf0df66779e7f07ea36c662 1571398943  
3529cc931c6b1d389fa247596971878e4241c0f62032fab8c8bb3143d59b700a 1571398943  
5353577c9af1f887284f00f10d296c3af089d8989f7cd582eeaf43aaaf914aed2 1571645600 data:{ "prenom": "000", "nom": "000",  
"user_hash": "73b547d3b20273581fb16a944f0da9fd159565eb8d40e591b3da365630d5593a"}  
270dc10e3beb227b5ec38691c3e00756158e1dc4a8d5d02168a57b9023f8a74a 1571645600  
a79860b41699b7ae1fa30725c3045aa51837e9169e5dc1d268a7d7585813aa57 1571645600  
ebf46d1b616a48a9f2e0fabbeae44e42275c09dd8f348a8b261b78fe89ba264 1571901321 data:{ "prenom": "XYZ", "nom": "XYZ",  
"user_hash": "28da960d76bcd57a331fd033ff378aec9bdf5e93dad6481fd38742cce8e67d56"}  
a3e31b42e4457e03ccac6873c854220b914d6bc5aa133666c88a19dd5bb79baf 1571901321  
1bbc8f8564fde3d0ba5145b02a82ad115a4ef9d6e143f887b9ca5fb99977f4 1571901321  
ec8e69136875540eb8832e300c80e1a53bb3a911c196cba0e62eb5cf927c3120 1572127870 data:{ "prenom": "azerty", "nom": "azerty",  
"user_hash": "967b8db5af3360eee958cc1770b489314ff804263250ab2fb69745d39fe5a402"}  
8dfc7bd089fa25dc059273861c2b31a603064a8f723e8151331108c0c3ac09cb 1572127870  
565a72f611b2192376310b5b2225cdc67b876dd6e5089ad946e95909e10ab060 1572127870  
7802b9b99f21329090903dfc6f8e146e35d6f513d38847154100320042ca7b1e 1572129639 data:{ "prenom": "zefzefze", "nom": "szdzdfzef",  
"user_hash": "15b5508720d8c2f7e26ca529950da4ff1d43379485ee756d6fd3652e3122a76"}  
e47bfd7fdecbb52b4f2d62df706a9598d9ff176d7ffa479da90e18374693ab74f 1572129639  
9a081094711fc226b74995010a06bf242d48505db04142648bde5bb038f93bd3 1572129639  
438385d1aeb60e7366b68e807f2d14995e0623147bd9ae27d84e48e4a9d57c76 1572511290 data:{ "prenom": "HARRY", "nom": "POTTER",  
"user_hash": "89a8a5d1c532a5d2688161c7f4a2ce9768095549f6001554ebbef1edeed49e36"}  
70aecb2ec9a906e8ddd4678ddb4f4c0964c279d00ede387929c2365046a9222f 1572511290  
15c167af5eaf9a341516af07a62ab2adc1f030aab967f63bce625723a03aa5b9 1572511290
```

Un Block fait 4 lignes

- **En jaune** : le hash de toutes les données du formulaire
- **En rouge** : l'horodatage
- **En violet** : le Nom et le Prénom de l'utilisateur sont en clairs pour pouvoir retrouver la signature électronique correspondant à l'identité d'un utilisateur
- **En vert** : le hash de l'utilisateur, sa signature électronique (certaines données du formulaire dont la clé de preuve)
- **En orange** : le hash du Block
- **En bleu** : le hash de tous les Blocks du Registre

ANNEXE 8

CINQ PREMIERS RESULTATS DE LA RECHERCHE EN ANGLAIS

Hash Function Attacks | Records Keeper

<https://www.recordskeeper.co> › blog › hash-function-at... ▾ Traduire cette page

16 févr. 2018 - There are mainly three **types of hash function attacks**: Collision **attack**: A collision **attack** on a cryptographic **hash** tries to find two inputs producing the same **hash** value. The attacker does not have control over the content of the message, but they are arbitrarily chosen by the **algorithm**.

Attaque de collisions — Wikipédia

<https://fr.wikipedia.org> › wiki › Attaque_de_collisions ▾

En cryptographie, une attaque de collisions est une attaque sur une fonction de hachage ... Il existe deux **types** principaux d'attaques de collisions : ... de l'article de Wikipédia en anglais intitulé « Collision attack » (voir la liste des auteurs). ... Dengguo Feng, Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, ...

Collision attack - Wikipedia

<https://en.wikipedia.org> › wiki › Collision_attack ▾ Traduire cette page

In cryptography, a collision **attack** on a cryptographic hash tries to find two inputs producing the same hash value, i.e. a hash collision. This is in contrast to a preimage **attack** where a specific target hash value is specified. There are roughly two **types** of collision **attacks**: ... A hash of n bits can be broken in $2^{n/2}$ time (evaluations of the **hash function**). Classical collision attack · Chosen-prefix collision ... · Attack scenarios

Lessons From The History Of Attacks On Secure Hash Functions

<https://electriccoin.co> › blog › lessons-from-the-history-... ▾ Traduire cette page

24 mai 2019 - I summarize here the history of **attacks** on secure **hash functions** in order ... digital signature **type**, today, quantum computer, asymmetric crypto ...

Hashing Algorithm - an overview | ScienceDirect Topics

<https://www.sciencedirect.com> › computer-science › ha.... - Traduire cette page

SHA-1: This is the second version of the Secure **Hash Algorithm** standard, SHA-0 Birthday **attacks** are based on a unique problem with hashing algorithms based There are multiple **types** of hashing algorithms, but the most common are ...