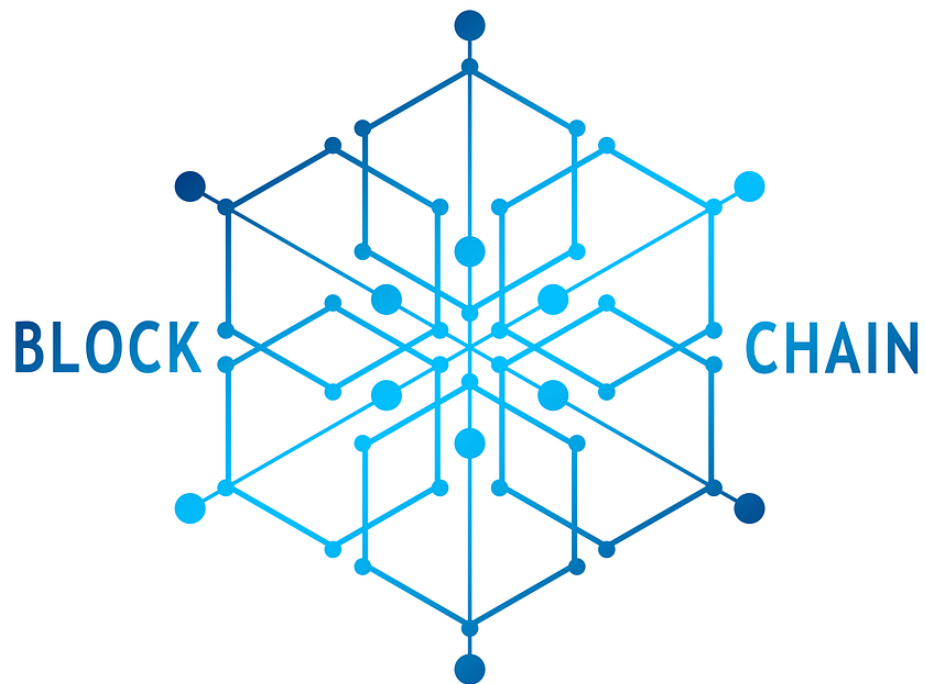

Projet IOT & Blockchain

Sujet spécial en informatique II



Sommaire

Objectif globale du projet	3
Architecture globale du projet	3
Coté Serveur	4
Mise en place serveur	4
BDD Personnes et vidéos	5
Algorithme de reconnaissance d'image	5
Algorithme de sauvegarde de vidéo	6
Coté Client	6
Flux Vidéo	6
Interface de consentement	6
Mise en place de la Blockchain	7

I. Objectif globale du projet

L'objectif du projet est de créer une gestion de consentement privée et sécurisée suite à une reconnaissance faciale en utilisant la blockchain.

Une fois une personne reconnue dans un flux vidéo, une demande de consentement contenant ses informations personnelles lui sera envoyée en lui indiquant qu'elle a été reconnue. Cette demande ainsi que la réponse du consentement de la personne sera stockée dans la blockchain afin de la rendre immuable. De cette manière, si une personne accepte ou refuse de donner son consentement sa réponse ne pourra pas être modifiée, falsifiée par un intermédiaire.

II. Architecture globale du projet

L'architecture du projet prendra la forme d'une architecture client-serveur.

Le serveur contiendra l'algorithme de reconnaissance faciale qui fonctionnera en continue tout au long de la journée. Une base de données y sera aussi présente afin de sauvegarder les vidéos des caméras. L'algorithme de reconnaissance faciale essayera de reconnaître les personnes du flux vidéo reçu par rapport à la base de données de personnes qu'il possède. Une fois reconnues, un e-mail leur sera envoyé avec leurs informations pour leur demander leur consentement. Cette demande ainsi que la réponse de la personne seront stockées dans la blockchain.

Le côté client sera simplement représenté par la caméra qui enverra constamment des flux vidéos au serveur pour les faire traiter par l'algorithme de reconnaissance faciale. Une interface de consentement permettra de gérer les demandes et les réponses des personnes concernées.

La blockchain permet simplement de stocker la liste des personnes ayant donné ou non leur consentement suite à une reconnaissance faciale. Ainsi, les données sauvegardées dans la blockchain seront immuables.

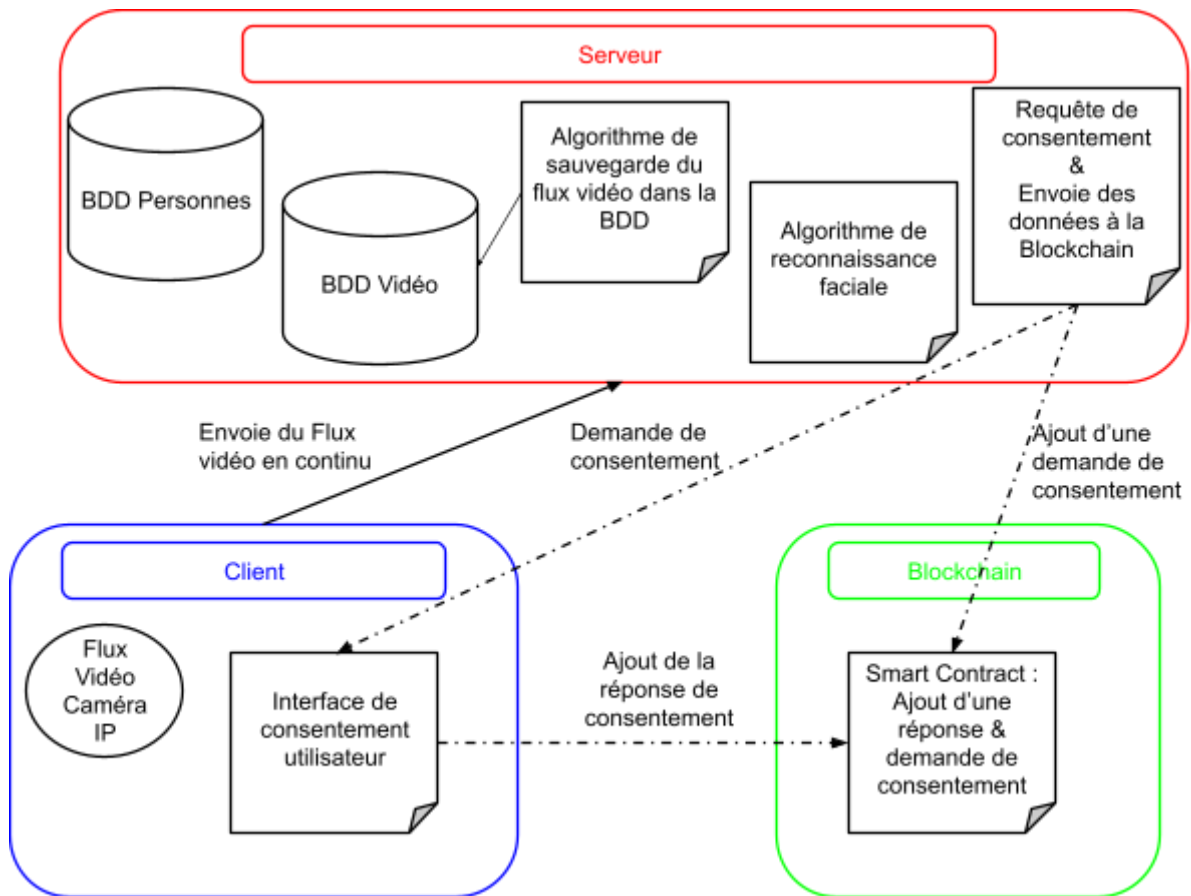


Schéma représentant l'architecture du projet

III. Coté Serveur

A. Mise en place serveur

Le serveur héberge les différentes bases de données (personnes ainsi que la sauvegarde des flux entrant) ainsi que le programme de reconnaissance faciale et enfin interagir avec le client pour les décisions de consentement. Il s'agira d'un serveur Apache avec MySQL installé dessus.

Ressources utilisé :



B. Base de données Personnes et vidéos

Ces bases de données sont prévues pour fonctionner en MySQL. Une d'entre elles sera dédiée à la gestion des utilisateurs : différents attributs permettent de stocker les informations des utilisateurs tels que leur numéro d'identification, leur nom, leur prénom, leur adresse mail ainsi que le chemin de l'image du visage qui lui est associé (voir figure). Une autre base de données permet de stocker les vidéos où les personnes ont été reconnues.

Nous avons utilisé Wamp pour héberger notre base de données MySQL en local et ainsi pouvoir faire fonctionner notre application en local.

		idPersonne	photo	nom	prenom	email
<input type="checkbox"/>	Éditer Copier Supprimer	3	../faces/person1.PNG	Emilie	Dupont	e.dupont@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	4	../faces/person2.PNG	Duffet	Chloe	c.duffet@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	5	../faces/person3.PNG	Poirier	Eloise	e.poirier@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	6	../faces/person4.PNG	Benoit	Franck	f.benoit@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	7	../faces/person5.PNG	Marleau	Louis	l.marleau@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	8	../faces/person6.PNG	Longpre	Henri	h.longpre@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	9	../faces/person7.PNG	Dastous	Emmanuelle	e.dastous@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	10	../faces/person8.PNG	Phaneuf	Alexandre	a.phaneuf@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	11	../faces/person9.PNG	Turgeon	Nathalie	n.turgeon@gmail.com
<input type="checkbox"/>	Éditer Copier Supprimer	12	../faces/person10.PNG	Sacre	Gustave	g.sacre@gmail.com

Capture d'écran du contenu de la table Personne de la base de donnée

Pour remplir cette base de données, les utilisateurs doivent se connecter à la page d'inscription dédiée à renseigner leurs informations personnelles.

Nous avons choisi de concevoir cette page en utilisant le langage PHP, plus particulièrement la version 7.4 qui introduit de nombreux correctifs de sécurité et qui a un support de mises à jour de sécurité jusqu'à fin 2022 (le plus récent à ce jour).

Également, le langage PHP nous permet de communiquer avec la base de données en toute facilité et sécurité grâce à l'extension intégrée PDO (PHP Data Object). En effet, en plus d'être invisible pour le côté client (le côté client ne voit jamais le code PHP, donc le client n'a pas accès au code source et autres données tels que le chemin d'accès à la base de données par exemple).

Finalement, nous avons bien pris soin d'utiliser le concept de requêtes préparées afin d'éviter certaines failles de sécurité les plus courantes tel que l'injection SQL.

Le même principe s'applique pour le téléversement de l'image choisie par l'utilisateur. Le code a été testé au mieux pour éviter de téléverser des images ne respectant pas les pré requis fixés par l'algorithme (extensions, noms erronés..etc).

Pour aller plus loin, on pourrait ajouter à cette base de données d'autres informations comme la localisation des caméras, la date de validité du consentement (afin de le renouveler par exemple).

Add your data in our database

Nom

ALHABAJ

Prenom

Mahmod

Email address

mhabaj99@hotmail.Com

We'll never share your email with anyone else.



Choisir un fichier | photoProfilMah.png

Save My Data

Page d'ajout des informations personnelles

Personnes
idPersonnes INT
photo VARCHAR(45)
nom VARCHAR(45)
prenom VARCHAR(45)
email VARCHAR(45)
Indexes

Videos
idVideos INT
video VARCHAR(45)
date DATETIME
Indexes

MCD de la base de données

Ressources utilisé :



C. Algorithme de reconnaissance faciale

Présentation :

Python est le langage utilisé. C'est un langage documenté et très largement utilisé dans le domaine de l'intelligence artificielle. Les bibliothèques sont facilement disponibles et leur utilisation est abordable. C'est pour cela que nous avons choisi de l'utiliser.

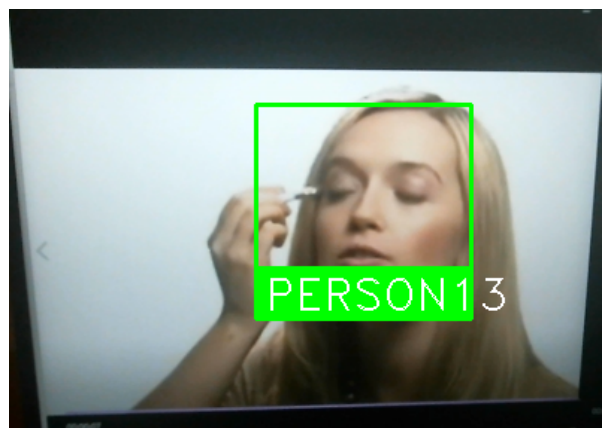
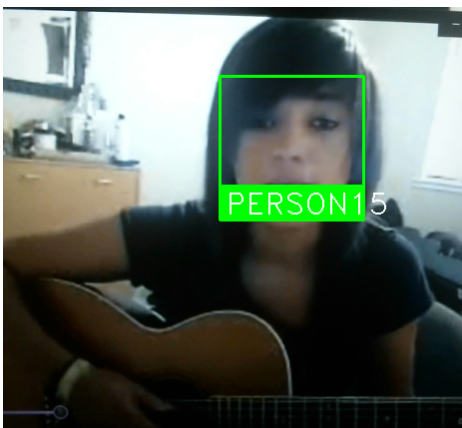
La bibliothèque [Face_recognition](#) créée par [Adam Geitgey](#) permet de pouvoir faire du face matching de manière simplifiée. Nous avons utilisé cette bibliothèque pour plusieurs raisons:

- facilité d'utilisation et compréhension
- rapidité de mise en place
- bonne fiabilité des résultats dans notre contexte.

En effet l'objectif de ce projet **n'est pas** de développer le meilleur programme de reconnaissance de personnes mais de produire une solution qui gère la gestion des consentements à l'aide d'une Blockchain. Cependant cette partie de la solution se trouve être assez robuste, fiable quand les personnes ne s'éloignent pas trop des images fournies en entraînement.

Pour cela, lors du lancement du programme, celui-ci sera obligé d'apprendre (phase apprentissage systématique) les personnes qu'il doit reconnaître. Durant l'exécution, le programme va envoyer le signalement que la personne reconnue dès qu'il rencontre cette personne. Nous avons mis un temps réglable, durant cet intervalle, une personne ne sera pas identifiée deux fois pour éviter le spamming. Ce choix a été fait car, dans le sujet il est décrit que la personne doit recevoir une notification à chaque fois qu'elle est reconnue. Pour des problèmes de saturation de bande passante, d'envoi de notification nous avons choisi de notifier la personne uniquement une fois par intervalle de temps. En effet l'algorithme va reconnaître le visage de la personne à chaque frame ou il est visible, ce qui fait pas loin de 30 notifications par secondes.

Ci-dessous un exemple de l'utilisation du programme en utilisant une webcam usb afin de pouvoir filmer l'écran de l'ordinateur.



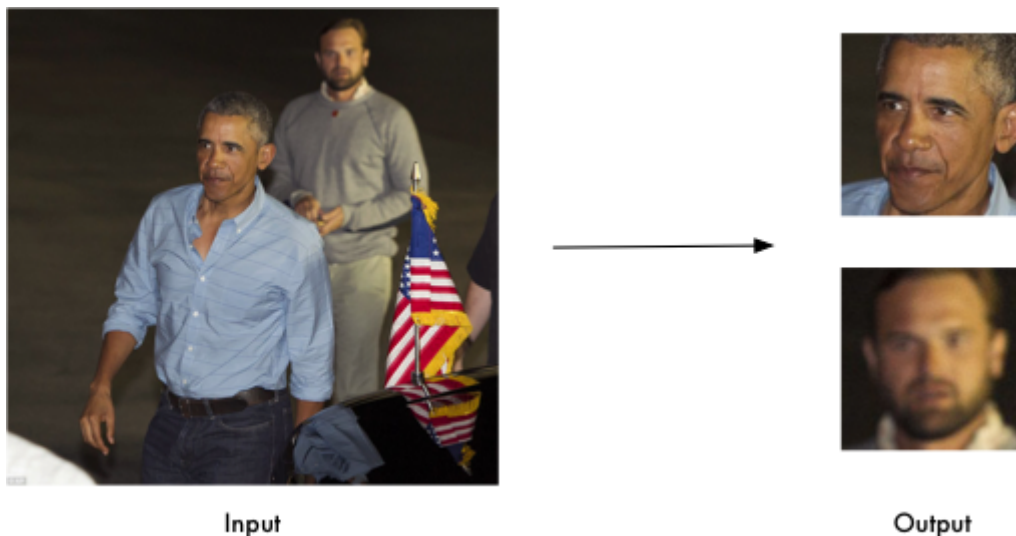
Principe de fonctionnement:

Tout est concentré dans le fichier python FRCBigData.

En premier lieu, le programme va énumérer toutes les photos de profil des personnes qui se sont déjà inscrites dans le système. Ensuite il va encoder chaque image pour en extraire les caractéristiques du visage.

A l'aide de la documentation, nous pouvons dégager deux étapes principales pour l'encodage : Recherche du visage dans la photo de profil et caractérisation.

L'image d'origine va passer en premier lieu par la méthode suivante pour extraire les visages de l'image : `face_recognition.face_locations(image)`. Cela va permettre d'isoler



l'information essentielle de l'image : le(s) visage(s). Dans notre cas, nous supposons que la personne sur la photo de profil est seule. Donc une photo de profil sera associée à une et une seule personne.

Dans un second temps, le programme va identifier les personnes en associant à chacun les caractéristiques suivantes : Menton, nez, yeux, bouche. Les données récoltées sur ces différentes parties du visages seront notre base pour reconnaître la personne.



Input



Output

Lors de l'apprentissage, chaque image ainsi traitée va être associée à une personne. Ensuite, dès qu'une personne aura ses caractéristiques de visage proche de la personne déjà étiquetée, le programme déduit que cette personne est passée devant la caméra à un instant t.

Quand la personne est reconnue, le programme va récupérer ses informations contenues dans la base de données et va lancer la procédure de demande de consentement par l'envoi d'un mail détaillée dans le paragraphe suivant.

Le code tournera sur un container docker contenant les dépendances suivantes : python, face_recognition. Cet algorithme a une fiabilité de plus de 99% et est issu du deep learning.

Ressources utilisé :



Suite à la reconnaissance de la personne grâce à l'algorithme utilisé, le logiciel lui enverra un mail. Ce mail sert simplement à lui indiquer qu'elle a été reconnue dans un certain lieu ainsi qu'à lui demander son consentement de ses données. Le mail prend la forme d'un message contenant un lien qui redirige l'utilisateur vers la page web de l'interface de demande de consentement (voir capture d'écran suivantes).

projetiot197@gmail.com

À n.maggouh ▾

Bonjour Maggouh Naoufal,

Vous avez été reconnu(e) près de : Batiment Alphonse Desjardin.

Pour le bon fonctionnement de l'application, veuillez nous donner votre consentement en suivant ce lien: [ici](#).

Merci.

Cordialement

Exemple de mail reçu par un utilisateur

Pour programmer l'envoi du mail, nous avons utilisé le protocole SMTP. SMTP est un protocole de communication assez répandu qui est généralement utilisé pour l'envoi de courriel électronique. Nous avons choisi de l'utiliser car nous avons déjà effectué des travaux ainsi que des projets avec. Cela nous a permis d'éviter de passer par la phase d'apprentissage du fonctionnement de celui-ci.

Nous avons donc créé deux adresses email pour le projet. Une qui enverra le mail et la seconde qui représentera l'adresse mail des personnes reconnues. Dans le cadre de notre projet toutes les personnes qui seront reconnues auront la même adresse email.

La fonction qui envoie le mail est divisée en plusieurs parties principales:

- L'identification des paramètres d'envoi de mail (expéditeur, destinataire, contenu du mail, objet, ...)
- L'encodage du message en HTML pour que le lien contenu dans le message puisse être reconnu comme un lien cliquable par l'utilisateur
- La connexion au serveur SMTP de google ("smtp.gmail.com" sur le port 587, puisque nous utilisons une adresse mail google pour l'expéditeur du mail)
- L'ajout d'une sécurisation TLS (Transport Layer Security) pour rendre la communication et l'envoi du mail plus sécurisés
- La connexion du compte qui enverra le mail (avec l'identifiant et le mot de passe du compte)
- L'envoi du mail
- La déconnexion au serveur smtp de google

D. Algorithme de sauvegarde de vidéo

Pour cet algorithme nous utiliserons Python ainsi que la bibliothèque MySQL Connector afin de communiquer avec notre base de données Vidéo. Le code tournera potentiellement à l'aide d'un serveur sur un container docker contenant les dépendances Python et MySQL.

Le flux vidéo reçu sera sauvegardé en format .avi. Ces fichiers vidéos auront pour nom la date heure à laquelle ils ont été pris et seront présents sur le serveur comme indiqué précédemment.

Finalement, pour communiquer avec la base de données, nous avons décidé d'utiliser la bibliothèque mysql.connector.

Ressources utilisé :



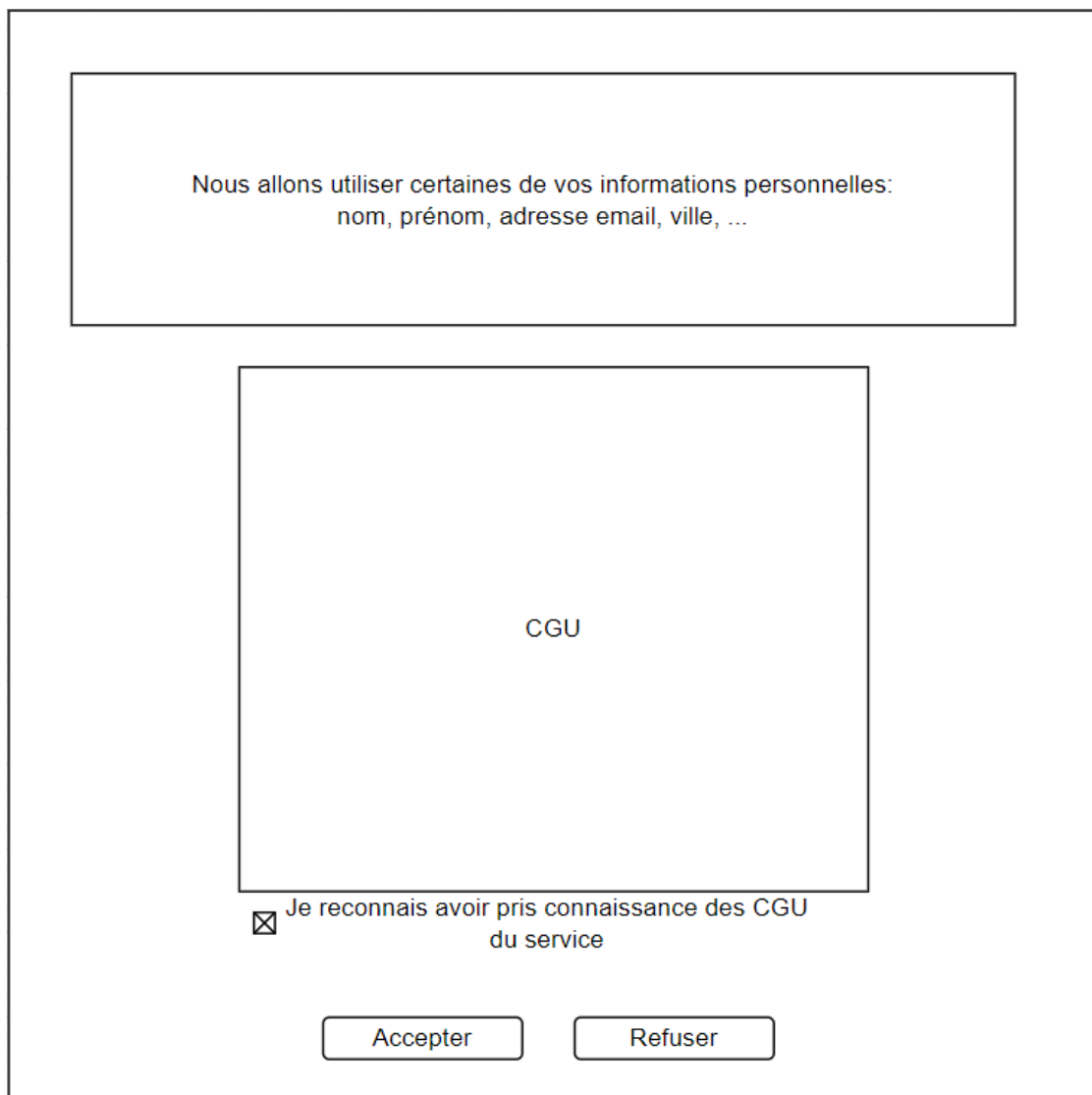
IV. Coté Client

A. Flux Vidéo

Le flux vidéo sera fourni par des caméras connectées au système. Pour le projet, nous allons simuler ces flux par des vidéos préenregistrés depuis le dataset mentionné précédemment. En pratique, ce flux vidéo sera enregistré dans la base de données côté serveur une fois transféré, puis sera passé au programme de reconnaissance faciale.

B. Interface de consentement

Pour créer cette interface web, nous avons créé une maquette dans laquelle nous avons choisi d'afficher les informations de l'utilisateur qui seront sauvegardées: le nom, le prénom, l'adresse e-mail ainsi que l'emplacement dans lequel la personne a été reconnue. Une maquette est présentée ci-dessous.




The image shows a wireframe of a consent interface. It consists of a large outer rectangle containing several elements:

- A top rectangular box with the text: "Nous allons utiliser certaines de vos informations personnelles: nom, prénom, adresse email, ville, ...".
- A large central rectangular box labeled "CGU" (Conditions Générales d'Utilisation).
- Below the "CGU" box, a checkbox followed by the text: "Je reconnais avoir pris connaissance des CGU du service".
- At the bottom, two buttons: "Accepter" and "Refuser".

Maquette de l'interface d'acceptation du consentement

Finalement, l'interface web de consentement contiendra en plus la photo de la personne reconnue ainsi que la date et l'heure où la personne a été reconnue. Nous avons ajouté deux boutons radios liés pour laisser le choix de la réponse à l'utilisateur. Les deux boutons sont liés donc l'utilisateur ne peut pas sélectionner les deux boutons à la fois. En appuyant sur le bouton "Save my consent", le résultat est accepté et si l'utilisateur accepte de donner son consentement, ses données seront envoyées et stockées dans la blockchain. L'ensemble des données qui sont sauvegardées sont stockées pour une durée de 30 jours maximum. A la fin de ces 30 jours, un nouveau mail de demande de consentement sera envoyé à l'utilisateur afin qu'il puisse révoquer ou continuer de partager ses données. Ce mail sera identique au premier mail reçu lors de la première demande de consentement.

Consent Form



Nom	Maggouh
Prenom	Naoufal
Adresse Email	n.maggouh@gmail.com
Emplacement	Batiment Alphonse Desjardin
Date	Sun Apr 17 2022 23:27:53 GMT-0400 (heure d'été de l'Est)

☒ I give my consent ☐ I don't give my consent

Save my consent

Capture d'écran de l'interface web d'acceptation du consentement

Il est tout de même important de notifier que l'emplacement affiché est aléatoire dans le cadre de notre projet. En effet, le projet ne permet pas de détecter l'emplacement des personnes

reconnues. Pour simuler l'emplacement nous avons choisi d'ajouter une liste de lieux et à chaque personne reconnue, un lieu aléatoire sera choisi par ceux présents dans la liste. Ce lieu sera alors considéré comme le lieu de reconnaissance de la personne et sera affiché dans le mail ainsi que dans l'interface web de demande de consentement.

Pour créer cette interface web, nous avons décidé d'utiliser HTML5 puisque c'est simplement la version d'HTML la plus récente et donc la meilleure version. En plus de cela, elle a une très grande compatibilité. En effet, elle est compatible avec tous les navigateurs (ou au moins une très grande majorité) ce qui nous permettra de mettre en place notre projet dans presque n'importe quel système.

Ensuite, nous utilisons du JavaScript pour nous permettre de faire communiquer le côté client et le côté serveur. Il sert principalement à faire passer les données entre l'algorithme de reconnaissance facial aux pages webs ainsi que des pages web à la blockchain. Nous utilisons le JavaScript car c'est un langage que deux des quatre membres de l'équipe ont déjà utilisé et il nous permet de gagner du temps dans l'avancement du projet.

Enfin, nous avons ajouté bootstrap à la liste des outils que l'on utilise. En effet, bootstrap permet de faire facilement des pages web qui sont esthétiques et très ergonomiques. C'est une bibliothèque qui fournit des composants déjà fait et très utiles dans la mise en page de pages web.

Ressources utilisé :



V. Mise en place de la Blockchain

A. Blockchain Public

Il faut savoir qu'il existe différents types de Blockchain : privé, public, fédéré ou hybride. Ce qui diffère entre toutes ces blockchains est le consortium défini pour valider les blocks. Dans le cadre de ce projet, nous avons choisi la blockchain publique pour plusieurs raisons.

La première est la sécurité, plus la décentralisation et la participation active sont importantes, plus une Blockchain sera sécurisée. Autrement dit, plus il y aura de nœuds dans le réseau, plus il sera difficile pour les hackers d'attaquer le réseau.

La deuxième est la transparence, cela promet un large éventail de cas d'utilisation, du vote aux transactions financières. De plus, n'importe qui peut vérifier la validité des transactions et des données enregistrées.

B. Description générale

Dans cette partie du projet, il s'agira de développer une API capable de communiquer avec un smart contrat déposé au préalable sur la Blockchain. Cette API permet lorsqu'elle est enquêtée d'effectuer des transactions avec la Blockchain (d'écrire des données et lire des données présent sur le smart contrat déployé préalablement). Ces données seront les informations des personnes reconnues suite à la reconnaissance faciale ainsi que leur récompense quant à la demande de consentement.

Dans l'optique d'atteindre cet objectif nous travaillerons d'abord sur le bon fonctionnement du projet sur une blockchain local créé par Truffle puis nous passerons à une blockchain publique : Rinkeby qui est une blockchain ethereum de test entièrement gratuite.

Pour développer le smart contrat nous utiliserons le langage Solidity qui est le langage compris par la blockchain ethereum. L'API quant à elle sera développée en Javascript. Nous utiliserons la bibliothèque web3 pour interagir avec un nœud ethereum local ou distant en utilisant HTTP, IPC ou WebSocket (nous utiliserons quant à nous WebSocket).

Ressources utilisé :



Ganache



SOLIDITY

C. Déploiement en local :

Dans un premier temps utiliserons Truffle qui est un Framework capable de créer une architecture de projet permettant de créer/déployer des smart contracts sur une blockchain Ethereum locale créée par Truffle. Ganache est une interface graphique permettant de visualiser les transactions effectuées sur la Blockchain Ethereum créée localement (voir le minage des blocs, constater le bon déploiement des smart contracts, voir les transactions effectuées avec l'API,...).

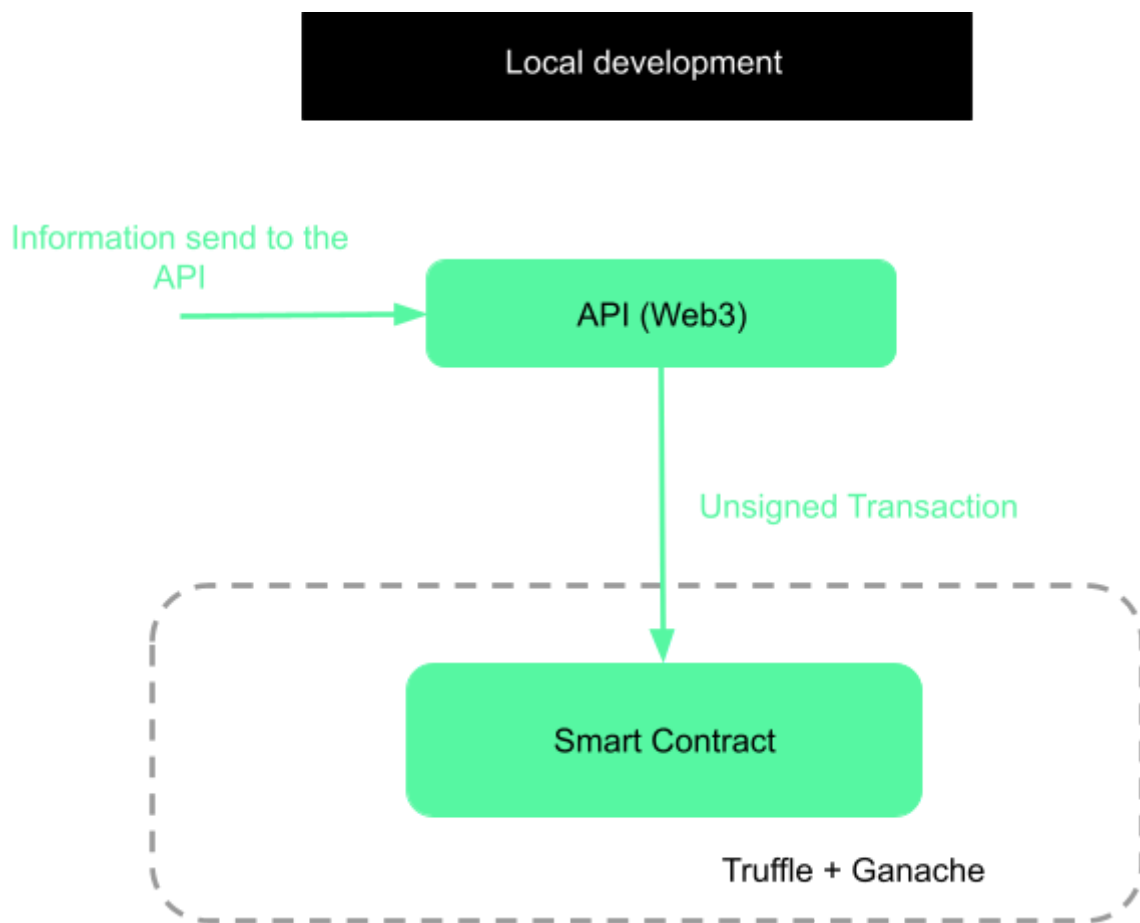


Schéma explicatif

D. Déploiement sur une blockchain public :

Enfin par la suite, nous utiliserons Rinkeby qui est une blockchain Ethereum de test créée pour les développeurs et qui a la particularité d'être gratuite (inutile de payer des Éthers pour déployer des smart contracts dessus).

A la différence du fonctionnement en local de notre API, il est nécessaire ici d'ajouter quelques fonctionnalités supplémentaires. Premièrement, afin qu'une transaction puisse être minée par la blockchain, il est nécessaire que celle-ci soit

signée avec la clef privée de l'utilisateur (connu de lui seul) afin de prouver qu'il est bien détenteurs des fonds qu'ils souhaitent dépenser. Une fois la transaction signée, il est nécessaire que notre API se connecte à un nœud de la blockchain (principe du réseau peer-to-peer) afin que la transaction puisse être minée. Deux options s'offre à nous : faire partie du réseau peer-to-peer et devenir un noeud de la blockchain ou utiliser l'API *INFURA* qui nous permet de se connecter à un noeud de la blockchain de notre choix (on optera pour cette option qui est la plus légère).

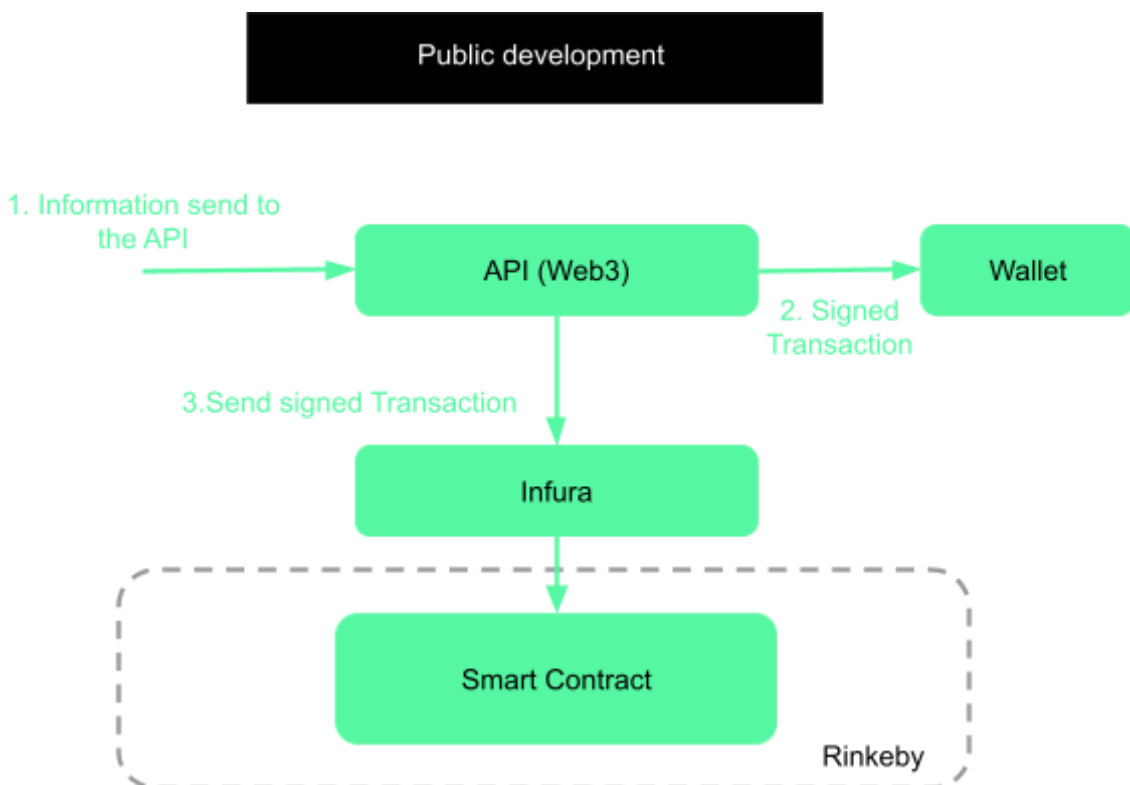


Schéma explicatif

E. Diagramme de package UML :

Package	Description
contracts	Contient l'ensemble des smart contrats déployés sur la Blockchain. Ces smart contracts contiendront les informations des personnes reconnues ainsi que leur réponse à la demande de consentement.
migrations	Contient l'ensemble fichier permet d'effectuer la migration/déploiement des contrats écrits. À chaque contrat est associé un fichier migration.
test	Contient l'ensemble des fichiers permettant de tester le bon fonctionnement des smart contracts.
buid/contract	Contient l'ensemble build des smart contracts.

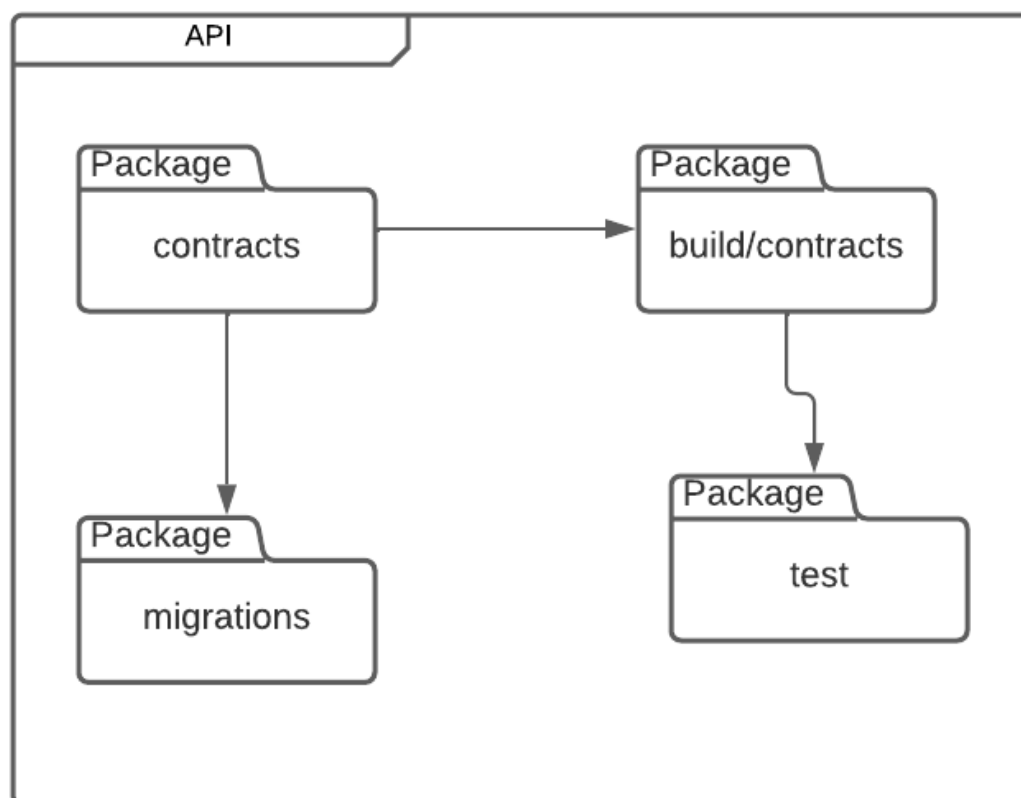


Diagramme de déploiement

F. Installation de la solution :

Tout d'abord ouvrez un terminal puis rendez vous dans le dossier du projet. Avant de lancer l'application il est nécessaire d'avoir préalablement installé le gestionnaire de

dépendance node (<https://nodejs.org/en/download/>). Puis lancer la commande **`npm install`** afin d'installer toutes les dépendances du projet. Vous pourrez ensuite lancer l'API à l'aide de la commande **`npm run build`**.

G. Fonctionnalités de notre API :

Route	Body	Réponse
POST Request http://localhost:8080	<pre>{ "idPerson": 1, "answer" : false }</pre>	<p><u>Cas Favorable:</u> L'utilisateur (id:1, consentement:false) a été ajouté avec succès</p> <p><u>Cas Non Favorable:</u> L'erreur est affiché</p>
GET Request http://localhost:8080?idPerson=1	<p>none</p>	<p><u>Cas Favorable:</u> id = 1, consentement = false</p> <p><u>Cas Non Favorable:</u> Cet individu n'existe pas dans la base de donnée</p>

A l'aide de ce tableau récapitulatif, on peut facilement observer que notre API fournit deux fonctionnalités. Une requête de type POST qui permet d'ajouter une personne sur la blockchain et une requête de type GET permettant de récupérer les informations sur une personne et donc son consentement. On peut remarquer que cette requête GET prend en query l'id de la personne dont on souhaite obtenir les informations.

H. Respect des données personnels stocké sur la Blockchain :

```
2  pragma solidity ^0.8.0;
3
4  contract Consentement {
5      address owner;
6      struct UserInfo{
7          bool answer;
8      }
9
10     event Return (UserInfo person);
11     mapping(uint256 => UserInfo) AllUsers;
12
13     //constructor sets the creator of the contract to the owner variable
14     constructor() {
15         owner = msg.sender;
16     }
17
18     //modifier checks that the caller of the function is the owner
19     modifier onlyOwner() {
20         require(msg.sender == owner, "Not Owner");
21         _;
22     }
23
24     //Set a new person on the blockchain
25     function SetUserInfo(uint256 id, bool answer) public onlyOwner{
26         UserInfo memory person = UserInfo(answer);
27         AllUsers[id] = person;
28     }
29
30     //Get a new person on the blockchain
31     function getUserInfo(uint256 id) public returns (UserInfo memory){
32         emit Return (AllUsers[id]);
33         return AllUsers[id];
34     }
35 }
```

Smart Contract déployé sur la Blockchain

Dans le smart contract on peut voir la présence d'une structure. Cette structure contiendra la réponse de consentement de l'utilisateur. Une des évolutions possible du projet pourrait donc être d'ajouter d'autres informations à cette structure : date de réponse de l'utilisateur, date d'inscription dans la blockchain ... La présence de cette structure dans le smart contrat permet donc une meilleure évolutivité de notre projet ainsi qu'une meilleure scalabilité.

```
2      pragma solidity ^0.8.0;
3
4      contract Consentement {
5          address owner;
6          struct UserInfo{
7              bool answer;
8          }
9      }
```

Ensuite on peut voir ci-dessous les deux fonctions présent dans le smart contract:

- **setUserInfo** : stocke l'id de la personne et la réponse de consentement
- **getUserInfo** : récupère pour un id donné la valeur du consentement de la personne

```
24      //Set a new person on the blockchain
25      function SetUserInfo(uint256 id, bool answer) public onlyOwner{
26          UserInfo memory person = UserInfo(answer);
27          AllUsers[id] = person;
28      }
29
30      //Get a new person on the blockchain
31      function getUserInfo(uint256 id) public returns (UserInfo memory){
32          emit Return (AllUsers[id]);
33          return AllUsers[id];
34      }
35  }
```

L'intérêt de ne stocker que l'id et la réponse de consentement associée permet de respecter le vie privé et ne pas directement stocker les données personnelles sur la blockchain. Seule l'entreprise possédant la base donnée connaîtra l'identité de la personne derrière l'id donnée.

Github du projet:

[mhabaj/BigBrother \(github.com\)](https://github.com/mhabaj/BigBrother)