



Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

HumanTech
Technology for
Human Wellbeing Institute

TIC – Filière Informatique

Projet de diplôme

2017-2018

HumanRec

Identification et reconnaissance d'individus

Nicolas Fuchs

Mandant : **HumanTech**

Superviseurs : **Elena Mugellini**
Omar Abou Khaled
Julien Tscherrig

Expert : **Robert Van Kommer**

Fribourg, juillet 2018

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland



Nicolas Fuchs est un étudiant de 23 ans en Bachelor en informatique à la Haute école d'ingénierie et d'architecture de Fribourg, membre de La Haute Ecole spécialisée de Suisse occidentale.

Il possède une maturité gymnasiale bilingue français/allemand avec les options spécifique arts visuels et complémentaire chimie.

Table des matières

1	Introduction	6
1.1	Motivation	6
1.2	Contexte	6
1.3	Objectifs principaux	7
1.4	Objectifs secondaires	7
1.5	Contraintes technologiques	7
1.6	Structure du rapport	8
2	Analyse	8
2.1	Besoins du projet	8
2.2	Etat de l'art	10
2.3	Analyse des Frameworks frontend/backend	12
2.4	Tests du prototype du projet de semestre	13
2.5	Synthèse	15
3	Conception	16
3.1	Architecture générale du système	16
3.2	Architecture détaillée du système	17
3.3	Diagramme de classe	18
3.4	Objets du système	19
3.5	Définition des interfaces utilisateur	21
3.6	Synthèse	28
4	Implémentation	29
4.1	Documentation de l'API REST	29
4.2	Récupération d'un flux vidéo	32
4.3	Envoi d'un flux vidéo	34
4.4	Caractéristiques des caméras	35
4.5	Service de détection et reconnaissance faciale	36
4.6	Fonctionnement du prototype	38
4.7	Proposition d'amélioration de la reconnaissance	39
4.8	Binding entre les différentes pages	40
4.9	Aperçus du prototype	41
5	Tests	44
5.1	Tests des conditions de simulation	45
5.2	Test du scénario de simulation	46

5.3	Tests utilisateurs.....	49
5.4	Tests de compatibilité	49
5.5	Conditions d'amélioration	50
5.6	Synthèse	51
6	Conclusion	51
6.1	Discussions des résultats	51
6.2	Problèmes rencontrés	51
6.3	Synthèse	52
6.4	Possibilité future.....	52
7	Déclaration d'honneur.....	52
8	Licences et versions de logiciels / bibliothèques	53
9	Bibliographie	54
10	Remerciements	55
11	Annexes.....	55
11.1	Planning.....	55
11.2	Cahier des charges.....	55
11.3	Autres versions des maquettes	56
11.4	Données de compte Amazon.....	57

1 Introduction

Dans cette section, la motivation du projet HumanRec est présenté, précédé de son contexte applicatif. Par la suite, les objectifs principaux sont listés, accompagnés de leur explication respective, ainsi que les objectifs secondaires. Finalement, le sujet de la contrainte technologique est abordé suivi de la structure de ce rapport.

1.1 Motivation

La motivation globale de ce projet est d'être capable de détecter dans un espace public ou privé, des déplacements suspects d'individus. Un exemple typique d'espace public dans lequel la sécurité doit être prise très au sérieux, c'est l'aéroport. C'est un endroit public où il y a un nombre impressionnant de va-et-viens et donc de dangers potentiels. Un système de surveillance trouve son utilité dans un tel environnement afin de garantir au maximum la sécurité de chacun. Le danger ne peut pas être écarté tant qu'il n'a pas été détecté. Dans le but d'obtenir ces informations, il est essentiel de savoir où et quand un individu est passé à tel ou tel endroit et d'en ressortir un historique de positions. Il est alors possible dans une extension de projet, d'exploiter cet historique pour en ressortir un niveau de danger accompagné d'un système de notifications et d'alarmes par exemple. Le projet est donc principalement axé sur la conception et implémentation d'un prototype offrant le suivi d'individus par des capteurs visuels. L'aspect juridique ne sera pas abordé dans ce document. Il est cependant extrêmement important lors du déploiement d'une telle application, de se renseigner sur les lois en vigueur sur le territoire du client.

1.2 Contexte

Etant de plus en plus présentes dans les espaces publiques, la détection et la reconnaissance faciale sont presque devenues habituelles de nos jours. On peut trouver la détection faciale par exemple dans les appareils photos modernes. En effet, ceux-ci affichent automatiquement un cadre autour des visages capturés par l'appareil. On peut trouver la reconnaissance faciale dans des smartphones (Iphone par exemple) pour déverrouiller le téléphone. Elle est également présente dans des domaines plus complexes tels que le système de reconnaissance DeepFace [1] de Facebook lors de l'ajout de photos avec des personnes non-taguées ou encore les systèmes de reconnaissance utilisés à des fins de surveillance par les aéroports ou les services de police. On se rend alors bien compte que les domaines de détection et reconnaissance du visage servent à plusieurs secteurs d'application. Le projet HumanRec se situe plutôt dans le secteur de la surveillance. Le prototype final doit permettre de suivre, grâce à une ou plusieurs caméras, un individu sélectionné par l'utilisateur. Ce système pourrait être utilisé par des services de sécurité.

Dans un des projets du semestre d'automne de cette même année, deux étudiants, Alexandre Dessonnaz et Lucas Alborghetti, ont dû comparer deux services de détection et reconnaissance faciale, Amazon Rekognition et Microsoft Cognitive Services, imposés par les contraintes du projet. Ils ont ensuite développé une application Java FX permettant d'identifier un visage et de le reconnaître depuis une vidéo sélectionnée par l'utilisateur ou la webcam (interne et externe). Ce projet de Bachelor s'inscrit dans la suite de ce projet de semestre. Il s'agit de choisir le plus adapté des deux services analysés et de l'utiliser dans l'implémentation de ce prototype de suivi d'individus. Ce projet se distingue du projet précédent par le fait que des caméras sont les sources des vidéos analysées et également par le fait qu'il y en a plusieurs (deux au minimum dans notre cas). Les langages de programmation ont également changé puisqu'un prototype WEB est le résultat souhaité. Le prototype

doit être développé au sein de la Haute Ecole d'ingénierie et d'architecture de Fribourg sous la supervision d'Elena Mugellini, Omar Abou Khaled et Julien Tscherrig et l'expertise de Robert Van Kommer.

1.3 Objectifs principaux

Les objectifs principaux sont les étapes essentielles au bon déroulement de ce projet.

- **Conception et implémentation d'un prototype web permettant le suivi d'un individu au travers d'une interface visuelle**
Le prototype web développé doit fournir à l'utilisateur une interface graphique qui permet de suivre une personne sur plusieurs caméras.
- **Gestion des caméras à partir de cette même interface visuelle**
Le prototype web développé doit permettre d'ajouter ou de supprimer une caméra du système.

1.4 Objectifs secondaires

Les objectifs secondaires sont des activités optionnelles, c'est-à-dire des fonctionnalités à développer si le temps à disposition le permet. Ces activités ne changent pas l'aspect fonctionnel du système.

- **Optimisation des appels au service de détection faciale**
Le projet de semestre en amont de ce projet de Bachelor inclut un mode automatique de détection faciale. Celle-ci est lancée toutes les 3 secondes. Le service de détection étant payant, une optimisation des appels à ce service est nécessaire.
- **Inclusion des fonctionnalités du prototype du projet de semestre**
Certaines fonctionnalités comme le clic sur une détection de visage sont directement reprises dans les fonctionnalités principales du programme. Par contre, l'utilisation de la webcam ou d'une vidéo préalablement enregistrée n'est pas incluse dans le scope du projet. On pourrait alors réécrire le code des fonctionnalités désirées pour qu'elles soient compatibles avec le web.
- **Détection d'attributs de personne**
Il pourrait s'avérer utile lors du tracking d'un individu de détecter la couleur de ses vêtements par exemple pour aider à l'identifier. On pourrait imaginer dans un scénario spécifique que nous avons à disposition uniquement la description vestimentaire d'une personne et que l'on aimerait la suivre sur les caméras. Cette fonctionnalité pourrait s'avérer extrêmement utile dans le cas où le visage de la personne à tracker n'est plus visible par la caméra.

1.5 Contraintes technologiques

Puisque ce projet de Bachelor est une suite du projet de semestre, obligatoirement l'un des deux services de détection et d'identification faciale du projet de semestre doit être choisi. Ces services ont leurs propres contraintes d'utilisation.

Une caméra et une webcam sont mises à disposition pour capturer des flux vidéo. C'est également une forme de contrainte du fait du nombre et des caractéristiques des appareils. La caméra fournie est une

caméra IP d'intérieur. Un scénario précis doit être défini pour tester les performances du système développé. Le lieu de déroulement du scénario est forcément à l'intérieur d'un bâtiment puisque la caméra IP n'est pas conçue pour l'extérieur.

1.6 Structure du rapport

Ce document est divisé en cinq chapitres principaux : analyse, conception, implémentation, tests et conclusion. L'analyse est axée sur les besoins du projet dans un premier temps puis les choix technologiques dans un deuxième temps. Ensuite, la conception du système est principalement expliquée au travers de schémas mais aussi par des wireframes. Après, l'implémentation contient les parties significatives de code. Les tests de conditions de simulation ainsi que les tests utilisateurs définissent les limites d'utilisation du prototype. Les problèmes rencontrés sont exposés accompagnés de leur solution, si existante, pour les résoudre. Finalement, ce document se termine par une conclusion qui contient une comparaison entre les objectifs du projet et l'état de sa réalisation, une conclusion personnelle et des perspectives de réalisation.

Le prochain chapitre aborde l'analyse des besoins du projet ainsi que l'analyse technologique.

2 Analyse

Cette section explique les besoins du projet avec un schéma et un diagramme de cas d'utilisation (Use Case). Ensuite, l'état de l'art expose l'état actuel du domaine dans lequel s'inscrit le projet. La dernière partie est axée autour des technologies choisies pour la détection et reconnaissance de visage, la partie frontend et la partie backend.

2.1 Besoins du projet

Au niveau de la partie affichage du prototype, une ou plusieurs pages web sont requises. Celles-ci doivent interagir avec une API REST implémentée sur un serveur sur lequel sont branchés les périphériques de capture d'images. Le serveur s'occupe aussi de toute la logique de détection et reconnaissance faciale. Cette logique est possible grâce à l'API que l'un des deux services analysés expose. On peut observer dans la Figure 1 ci-dessous comment les parties mentionnées ci-dessus se relient.

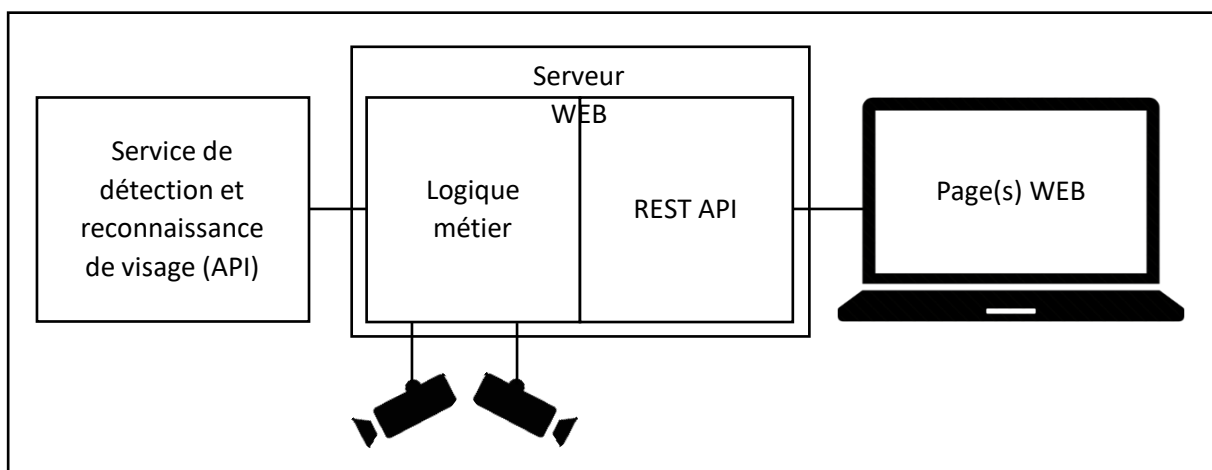


Figure 1 Schéma simplifié des composants du système

Concernant les fonctionnalités du programme, on aimerait pouvoir gérer les personnes à tracker. On doit donc pouvoir ajouter ou supprimer un visage avec ses informations associées dans la base de données du service de détection et reconnaissance faciale. On peut sélectionner un visage contenu dans la base de données ou en sélectionner un depuis la vidéo prise par une des caméras. Celles-ci sont configurables au travers d'une interface pour ajouter ou supprimer une caméra ainsi que changer l'ordre des caméras au niveau visuel. On doit pouvoir loguer le parcours d'un individu en précisant l'heure à laquelle il a été capturé par la caméra ainsi que l'identifiant de la caméra.

Les diagrammes de cas d'utilisation des Figures 2 et 3 permettent de mieux cerner les relations entre les actions que l'utilisateur peut faire avec le système. La version récente du diagramme de cas d'utilisation est celle de la Figure 3. La Figure 2 représente un diagramme créé en début de projet.

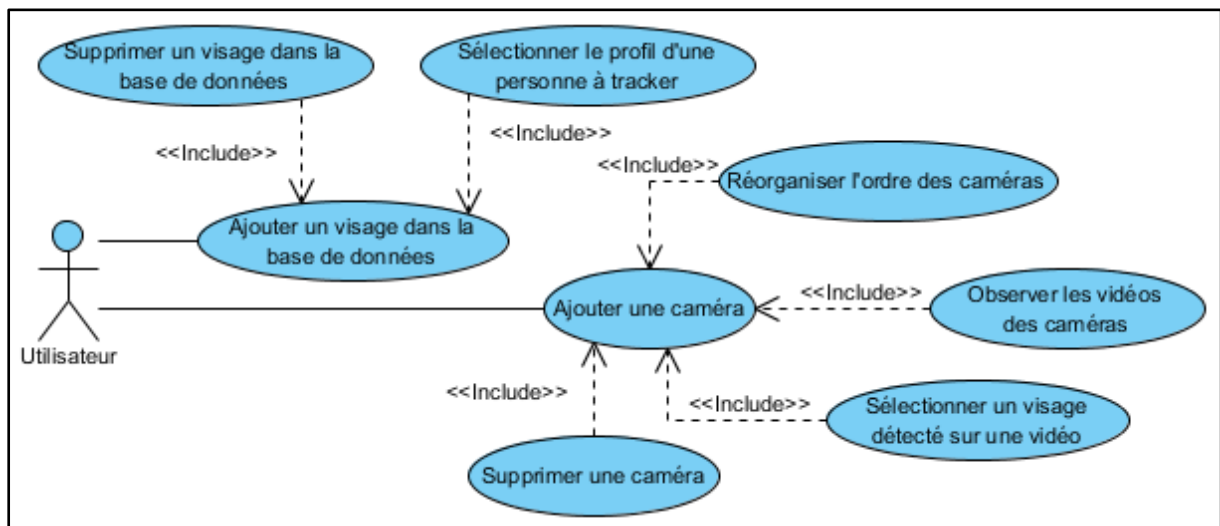


Figure 2 Diagramme de cas d'utilisation version 1

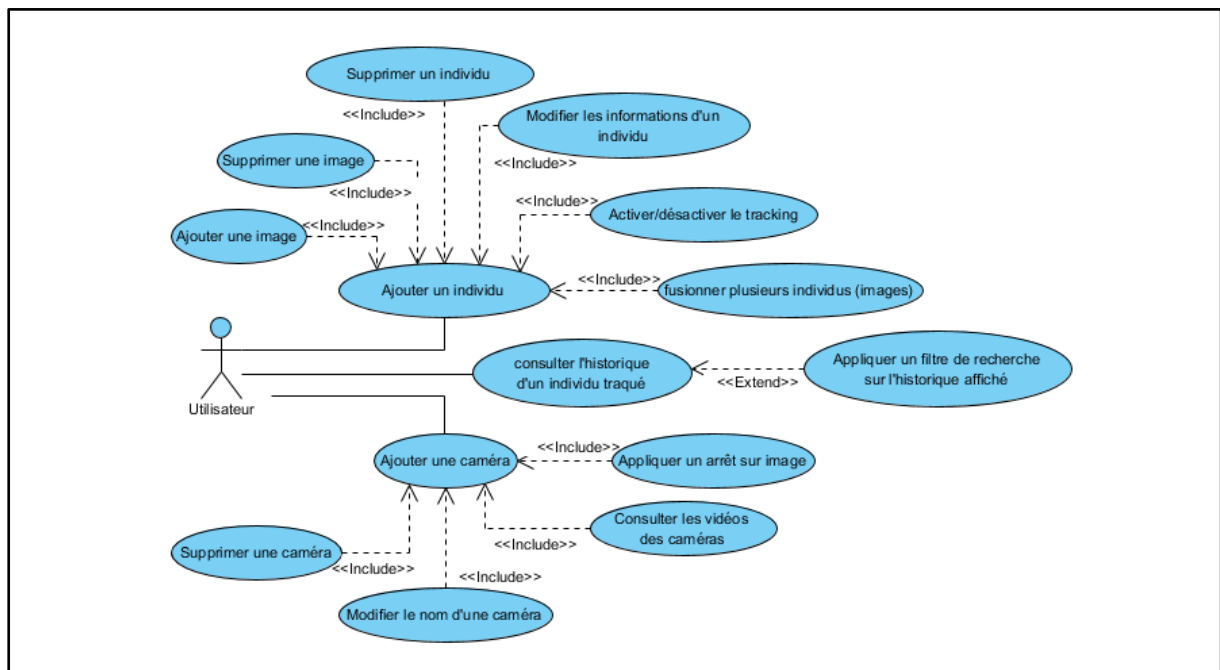


Figure 3 Diagramme de cas d'utilisation version 2

On se rend bien compte qu'il y a quelques différences au niveau de la terminologie ainsi que du nombre de fonctionnalités offertes. Les fonctionnalités de base comme la gestion des individus et des caméras sont restées les mêmes.

2.2 Etat de l'art

L'identification et reconnaissance faciale est très répandue et avancée dans les pays asiatiques, plus spécifiquement dans certains endroits de la Chine ainsi qu'aux Etats-Unis. Lorsque l'on désire passer un séjour aux Etats-Unis, chaque détenteur d'un visa doit obligatoirement passer devant une caméra pour être pris en photo. Ces photos sont par la suite utilisées comme entrées de programmes de reconnaissance faciale [2]. Un tel système développé par Hitachi Kokusai Electric permet d'identifier le visage d'une personne dans une base de données en comptant 36 millions [3]. La surveillance fonctionne en temps réel. L'utilisateur peut sélectionner une photo déjà enregistrée ou un visage détecté sur une des caméras installées et le système s'occupe de récupérer toutes les images provenant de toutes les autres caméras sur lesquelles cette personne apparaît. L'utilisateur peut ensuite visionner la vidéo à partir de laquelle chaque image a été extraite. Suivre une personne est également offert sur ce système. Bien que sa performance soit incroyable, le système Hitachi comporte tout de même des contraintes et des failles. La taille des visages doit être au minimum de 40 pixels en hauteur et en largeur pour que la détection fonctionne. Cette contrainte est présente dans presque tous les systèmes de détection faciale actuels avec des tailles minimales différentes. Si la caméra n'a pas la possibilité de prendre des photos plus ou moins de face (30 degrés d'écart maximum), la détection ne fonctionne pas bien. Finalement, du fait que certaines caméras sont positionnées à l'extérieur, les variations de luminosité et de temps rajoutent de la difficulté au système. La solution proposée est donc extrêmement rapide lorsque les bonnes conditions sont réunies.

[4]Le service de reconnaissance faciale par vidéo d'Amazon, Amazon Rekognition Video avec Amazon Kinesis Video Streams, a lui été testé dernièrement dans la ville américaine d'Orlando dans l'état de Floride aux Etats-Unis. Sept officiers de police se sont portés volontaires pour tester le fonctionnement de la reconnaissance faciale en pleine rue. Trois caméras ont été posées dans un domaine public afin de mener des tests. La surveillance publique est un sujet très controversé. Aucun résultat de test de cette technologie n'a été dévoilé par le département de police d'Orlando¹.



Figure 4 La police d'Orlando a utilisé le service d'Amazon

2.2.1 Produits existants

Une analyse des solutions actuellement existantes permet de découvrir des fonctionnalités par exemple auxquelles on n'aurait pas pensé en premier lieu ou bien des idées de design. Si un logiciel est open source, on peut même s'inspirer du code mis à disposition pour implémenter notre propre système. Trois entreprises citées ci-dessous offrent des logiciels de surveillance sur le marché.

1




https://mediaassets.abcactionnews.com/photo/2018/05/25/City%20of%20Orlando%20Police%20Department_1527232768970.jpg_87998586_ver1.0_640_480.jpg

NEC Corporation of America par exemple vend une suite de logiciels, NeoFace® Face Recognition Suite, composée des solutions suivantes : NeoFace® Smart ID, NeoFace® Reveal, NeoFace® Match et NeoFace® Watch.

Gemalto propose également un système de reconnaissance faciale nommé Live Face Identification System (LFIS).

Panasonic commercialise aussi plusieurs logiciels incluant ceux de base ainsi que des extensions. Pour qu'un système de surveillance soit opérationnel, plusieurs logiciels doivent être couplés.

Les solutions choisies dans le tableau ci-dessous sont celles qui ressemblent le plus au système que l'on désire développer dans le cadre de ce projet de Bachelor.

	 NeoFace® Watch [5]	 LFIS [6]	 WV-ASM200 [7]
Gestion en temps réel	✓	✓	✓
Gestion du playback	✓	Non-spécifié	✓
Nombre de caméras gérées	Plusieurs milliers	+1000	256 directes 6400 sur enregistreurs 256 sur encodeurs
Alertes contre menaces	✓	✓	✓
Catégorisation des individus	✓	✓	✗
Carte du système	✗	✗	✓
Taille minimale des visages	24 pixels entre les yeux	24 pixels entre les yeux Recommandé : 48 pixels	Non-spécifié
Résolution des caméras prise en charge	Non-spécifié	720p - 4K	HD
Open source	✗	✗	✗
Multi-plateforme	Non-spécifié (web-based) Alertes sur mobiles	Non-spécifié (web-based) Alertes sur mobiles	Windows (XP non supporté)
Liste de surveillance	✓	✓	✓
Suivi d'une personne sur plusieurs caméras	✓	Non-spécifié	✓

Cette liste de produits existants n'est évidemment pas exhaustive. Il existe encore plein d'autres solutions qui n'ont pas été explorées dans le cadre de ce projet en raison de la contrainte de temps.

2.2.2 Amazon Rekognition vs Microsoft Cognitive Services

Les deux services de détection et de reconnaissance faciale analysés dans le projet de semestre proposent tous deux une offre attractive pour le développeur. Cependant, il faut choisir le service le plus adapté pour la suite du projet. Des critères significatifs ont été retenus de l'analyse des services pour établir un scorecard, tableau de scores comparant plusieurs technologies, présent sur la page suivante. Toutes les informations de ce scorecard sont donc définies par rapport au chapitre Analyse du rapport de Lucas Alborghetti et Alexandre Dessonnaz.

Critère	Importance (%)	Amazon Rekognition	Microsoft Cognitive Services
Tarification	20	3	2
Rapidité	35	3	5
Précision	45	5	3

Score	100	3.9	3.5
-------	-----	-----	-----

Détail du calcul pour Amazon Rekognition : $(20*3 + 35*3 + 45*5) / 100 = 3.9$

Détail du calcul pour Microsoft Cognitive Services : $(20*2 + 35*5 + 45*3) / 100 = 3.5$

Au niveau de la tarification, les deux services proposent pendant les 12 premiers mois une offre gratuite incluant les fonctionnalités utiles à ce projet. En ce qui concerne l'importance entre la rapidité et la précision, la différence est infime. En effet, le prototype développé doit permettre d'analyser des vidéos en "temps réel" donc la rapidité est un critère important, mais sachant que le traitement de détection et reconnaissance n'est pas instantané avec la solution d'Amazon ou Microsoft, l'importance du critère de précision est supérieur.




Il est aussi intéressant de relever qu'Amazon propose aussi une reconnaissance faciale par vidéo. Cette fonctionnalité est également accessible à partir de la première année d'utilisation grâce à l'offre gratuite. Cette option n'a pas été gardée pour la suite du projet. La raison est la suivante : le prototype développé pourrait utiliser une librairie comme OpenFace par exemple qui fonctionne uniquement localement sur la machine et par conséquent, les images à comparer ne seraient pas utilisées par un service externe au programme ce qui assure la protection des données.

2.3 Analyse des Frameworks frontend/backend

L'analyse technologique inclut l'analyse de Frameworks potentiels pour la partie backend du système à développer ainsi que pour la partie frontend. De plus, le prototype du projet de semestre a été mis en place pour tester son fonctionnement. Ces tests figurent aussi dans l'analyse.

2.3.1 Technologie frontend

Du côté client, aucun gros Framework n'est utilisé comme Angular, React.js ou Vue.js. Par contre, un Framework est utilisé pour optimiser le temps passé pour obtenir un design visuellement acceptable. Le tableau ci-dessous compare les Frameworks Bootstrap [8], Materialize [9] et Foundation [10].

Critère	Importance (%)	 Bootstrap ²	 Materialize ³	 Foundation ⁴
Popularité	30	10	4	7
Connaissances	30	6	5	1
Simplicité	10	8	8	8
Documentation	30	8	7	8
Score	100	8	5.6	5.6

Détail du calcul pour Bootstrap : $(30*10 + 30*6 + 10*8 + 30*8) / 100 = 8$

Détail du calcul pour Materialize : $(30*4 + 30*5 + 10*8 + 30*7) / 100 = 5.6$

Détail du calcul pour Foundation : $(30*7 + 30*1 + 10*8 + 30*8) / 100 = 5.6$

² <https://img.stackshare.io/service/1101/C9QJ7V3X.png>




³ <https://img.stackshare.io/service/2015/GN559Lfb.png>

⁴ <https://img.stackshare.io/service/1105/glke9rjv.png>

En regardant ce tableau de scores, on en conclut que Bootstrap est le vainqueur. Cette solution se distingue des deux autres surtout par sa popularité ainsi que les connaissances acquises de ce Framework.

2.3.2 Technologie backend

La technologie employée sur le serveur (backend) doit pouvoir exécuter du traitement d'image. Un des objectifs secondaires est d'optimiser le nombre d'appels au service de reconnaissance faciale. La librairie très populaire et open source *OpenCV* permet de réaliser cela. Le code du backend doit pouvoir appeler des fonctions de la librairie *OpenCV*.

Critère	Importance (%)	[11]  python™	[12]  node JS	[13]  C++
Compatibilité au traitement d'image	40	10	9	10
Connaissances	30	5	8	6
Documentation	20	7	7	7
Simplicité	10	7	8	5
Score	100	7.6	8.2	7.7

Détail du calcul pour Python : $(40 \times 10 + 30 \times 5 + 20 \times 7 + 10 \times 7) / 100 = 7.6$

Détail du calcul pour Node.js : $(40 \times 9 + 30 \times 8 + 20 \times 7 + 10 \times 8) / 100 = 8.2$

Détail du calcul pour C++ : $(40 \times 10 + 30 \times 6 + 20 \times 7 + 10 \times 5) / 100 = 7.7$

Le Framework vainqueur est Node.js surtout grâce au critère de connaissances du Framework. C'est une solution également très intéressante au niveau des nombreux modules de traitement d'images (OpenCV, Rekognition, ...) installables au travers du manager de paquets npm (Node Package Manager).

2.4 Tests du prototype du projet de semestre

Les étudiants Alexandre Dessonnaz et Lucas Alborghetti ont du, pour leur projet de semestre 5, comparer deux frameworks imposés d'identification et de reconnaissance faciale : Amazon Rekognition et Microsoft Cognitives Services. Les caractéristiques comparées sont les suivantes : la gestion d'un compte et sa mise en place, les coûts d'utilisation, l'identification et la reconnaissance de personne. A chaque étudiant a été attribué un service. Une fois que chacun s'était suffisamment familiarisé avec son environnement et sa technologie, un programme modulable, générique et commun a été mis en place. L'application développée en javaFX devait identifier et reconnaître une personne sur une vidéo enregistrée ou diffusée en direct. Cette application a du être conçue, implémentée et testée. La conclusion de leur comparaison souligne que les deux frameworks ont leurs propres forces et faiblesses. L'un n'est pas meilleur que l'autre mais leur contexte d'utilisation est différent. Cette analyse provient de leur documentation.

Le prototype du projet de semestre a pu être testé avec le logiciel Eclipse et certaines fonctionnalités ne marchent pas. La webcam intégrée ainsi qu'une webcam externe branchée avec un port USB

fonctionnent et transmettent correctement leur flux vidéo. Je peux par la suite mettre la vidéo en mode pause. Les boutons "Previous" et "Next" fonctionnent aussi pour passer d'un frame capturé à un autre. Cependant, lorsque le check box de l'auto refresh est désélectionné, les frames ne sont plus capturés. J'ai été un peu surpris par ce comportement, j'avais pensé que l'affichage n'était pas mis à jour mais que les images étaient toujours capturées par intervalle régulier. La méthode de reconnaissance faciale marche aussi très bien. C'est le service Amazon Rekognition qui a été utilisé.

On peut observer ci-dessous sur les Figures 5, 6 et 7 que le fonctionnement général du prototype a pu être mis en place.

```
org.bytedeco.javacv.FrameGrabber$Exception: read() Error: Could not read frame in start().
    at org.bytedeco.javacv.OpenCVFrameGrabber.start(OpenCVFrameGrabber.java:211)
    at main.java.model.VideoGrabber.run(VideoGrabber.java:83)
    at java.lang.Thread.run(Unknown Source)
```

Figure 5 Erreur générée par l'objet OpenCVFrameGrabber

Comme indiqué dans le rapport du projet de semestre, le bouton "browse" provoque une erreur (Figure 5) sur Windows causée par l'objet OpenCVFrameGrabber qui n'arrive pas à obtenir le premier frame de la vidéo sélectionnée.

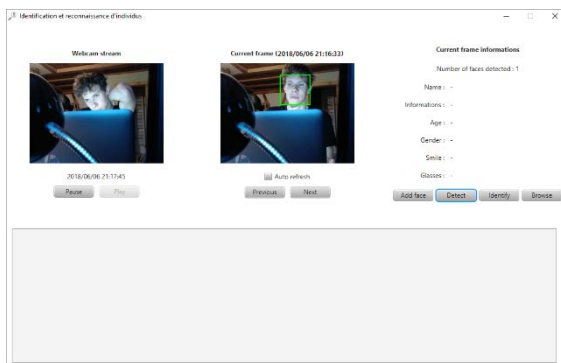


Figure 6 La détection ne passe bien

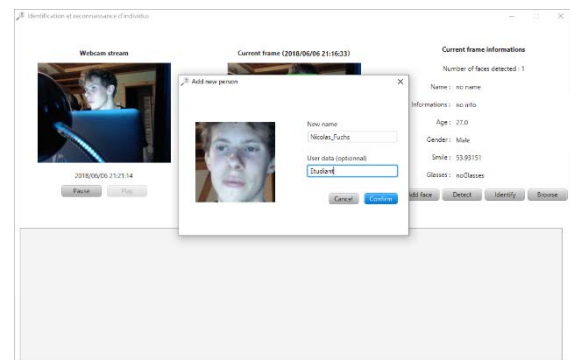


Figure 7 L'ajout d'une personne fonctionne

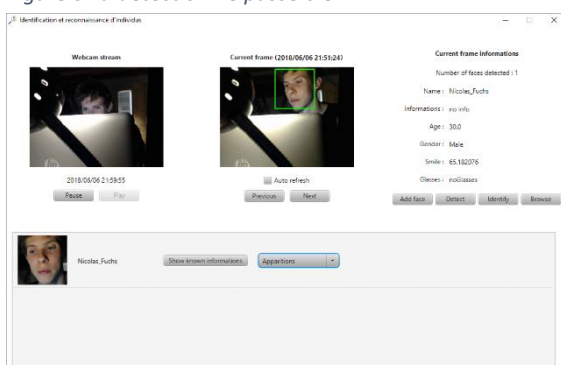


Figure 8 La reconnaissance marche également

Peu de problèmes ont été rencontrés pour faire tourner le prototype sur ma machine. Ils sont décrits dans le sous-chapitre 6.2. Lors du fonctionnement du programme, un bug lié à la reconnaissance faciale est survenu. Après la détection du visage, une reconnaissance faciale a été demandée en cliquant sur le bouton "identify" et une erreur survient dans Eclipse affirmant que l'image en paramètre envoyée au service d'identification ne contient pas de visage. Le problème a été contourné en approchant légèrement la tête de la caméra. Une hypothèse de cause de l'erreur pourrait être que le cadre de détection du visage était trop grand par rapport à la taille réelle du visage. Le système de reconnaissance a eu de la difficulté à récupérer les points importants du visage probablement aussi en raison du manque de lumière lors du test.

2.5 Synthèse

En conclusion de ce chapitre d'analyse, on se rend compte qu'il existe déjà pleins de systèmes très performants mettant en œuvre une surveillance par détection et reconnaissance faciale. Ces systèmes utilisent leurs propres algorithmes de machine learning incluant entre autres des réseaux de neurones. Ils sont donc très efficaces et placent la barre très haute pour les concurrents dans ce domaine.

Un travail sur ce sujet a déjà été réalisé pour un projet de semestre par les étudiants Lucas Alborghetti et Alexandre Dessonnaz. Ne partant pas de zéro, certains aspects de ce travail de semestre peuvent être réutilisés :

- **Design visuel du prototype javaFX**
La disposition des différents composants de l'interface visuelle du prototype javaFX peut donner un squelette approximatif pour le futur prototype web
- **Gestion et paramètres de compte Amazon/Microsoft**
La documentation à propos de la création d'un compte pour l'accès aux services d'Amazon ou Microsoft ainsi que son paramétrage a déjà été très utile pour faire fonctionner le prototype déjà existant.
- **Code java**
Le code java en lui-même ne peut pas être repris tel quel car le langage de programmation java est bien différent des langages de programmation pour le web comme JavaScript par exemple mais la logique qui se cache derrière reste semblable. C'est donc cette logique qui va être principalement reprise.
- **Analyse des services de détection et reconnaissance faciale**
L'analyse complète d'Amazon Rekognition et de Microsoft Cognitive Services sert de base au développement de ce projet de Bachelor. Elle donne les informations nécessaires pour choisir le service le plus adapté.

Au sujet des technologies utilisées pour l'implémentation du prototype web, du côté frontend, le Framework Bootstrap est choisi tandis que du côté backend, la technologie Node.js s'est démarquée des autres options.

Concernant le choix du service de détection et reconnaissance faciale, je me suis tourné vers celui qui a le meilleur taux de réussite, prenant en compte le fait qu'obtenir un prototype qui fonctionne en temps réel absolu est difficilement concevable, en raison du temps de latence des services imposés (entre 1 et 3 secondes globalement).

Le chapitre suivant se consacre à l'architecture du système par le biais de schémas puis le design des interfaces utilisateurs est présenté.

3 Conception

Ce chapitre de conception va aborder les sous-chapitres suivants : Architecture générale du système, architecture détaillée du système, diagramme de classe et pour finir, la définition des interfaces utilisateur. Cette phase permet de se faire une idée précise de comment va être le prototype final.

3.1 Architecture générale du système

Un schéma d'architecture générale permet de savoir en un coup d'œil rapide de quoi est composé le système sans entrer dans les détails. Comme une grande partie des applications WEB, l'architecture du système est de type trois tiers. L'ensemble du système est décomposé en trois grandes parties d'importance équivalente : La couche de présentation, affichant les données à l'utilisateur, la couche de traitement de données qui s'occupe de la partie logique la plus importante dans notre cas et la couche d'accès aux données, chargée de la persistance et récupération des données sur une base de données ou de simples fichiers.

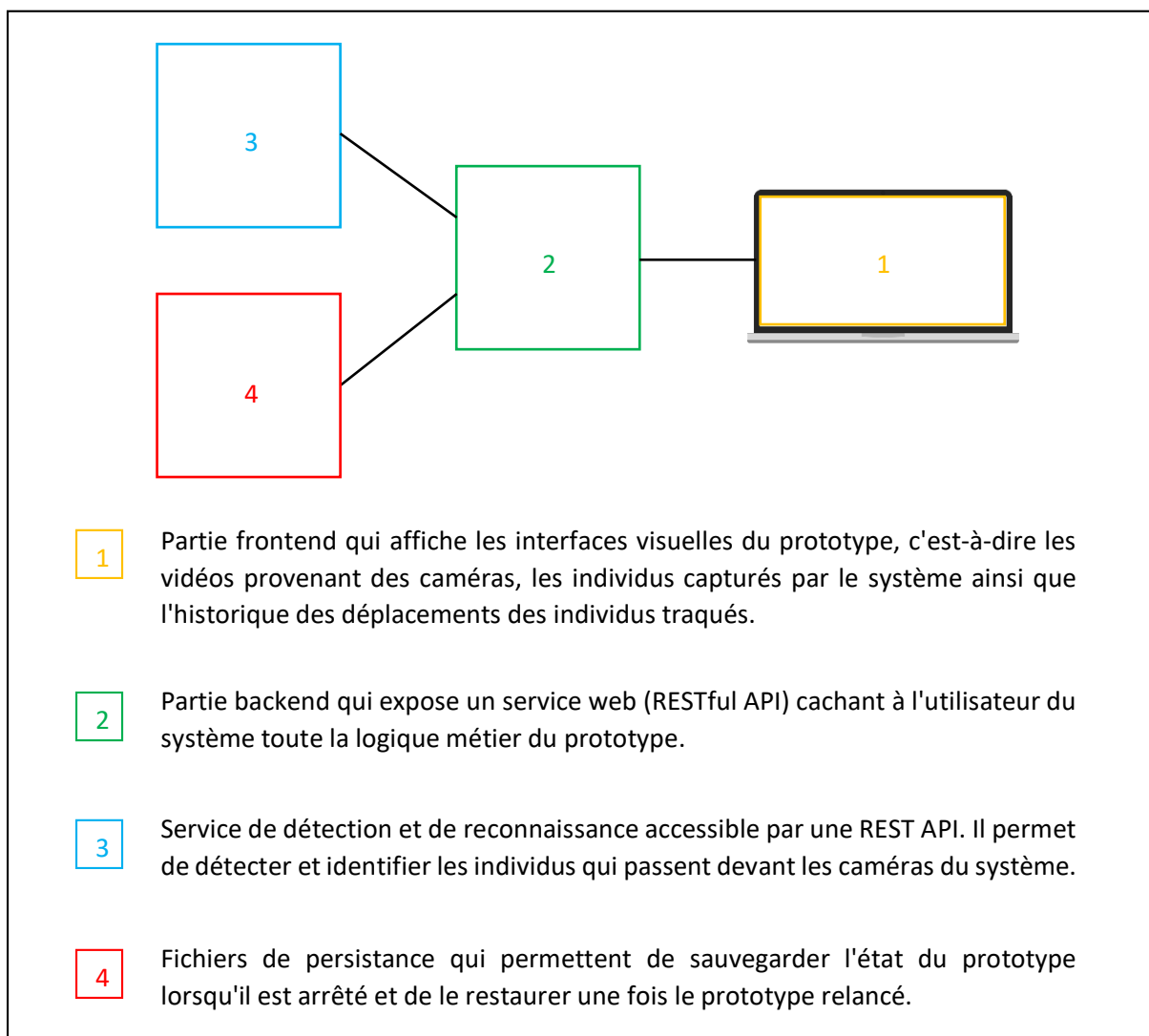


Figure 9 Schéma d'architecture générale

La couche de présentation se situe du côté client (1), la couche de traitement de données se situe principalement du côté serveur (2) mais aussi indirectement dans le service de reconnaissance faciale (3) puis la couche d'accès aux données se situe sur les fichiers de persistance (4).

3.2 Architecture détaillée du système

Un diagramme de déploiement permet de représenter les liens entre les différents composants du système ainsi que les nœuds (ici équivalent à machines) sur lesquels ils sont installés/déployés, et comment ils communiquent entre eux. Il permet également de connaître les technologies utilisées pour chaque composant au travers des artefacts.

Le premier diagramme de déploiement de la Figure 10 ci-dessous représente un déploiement maximal.

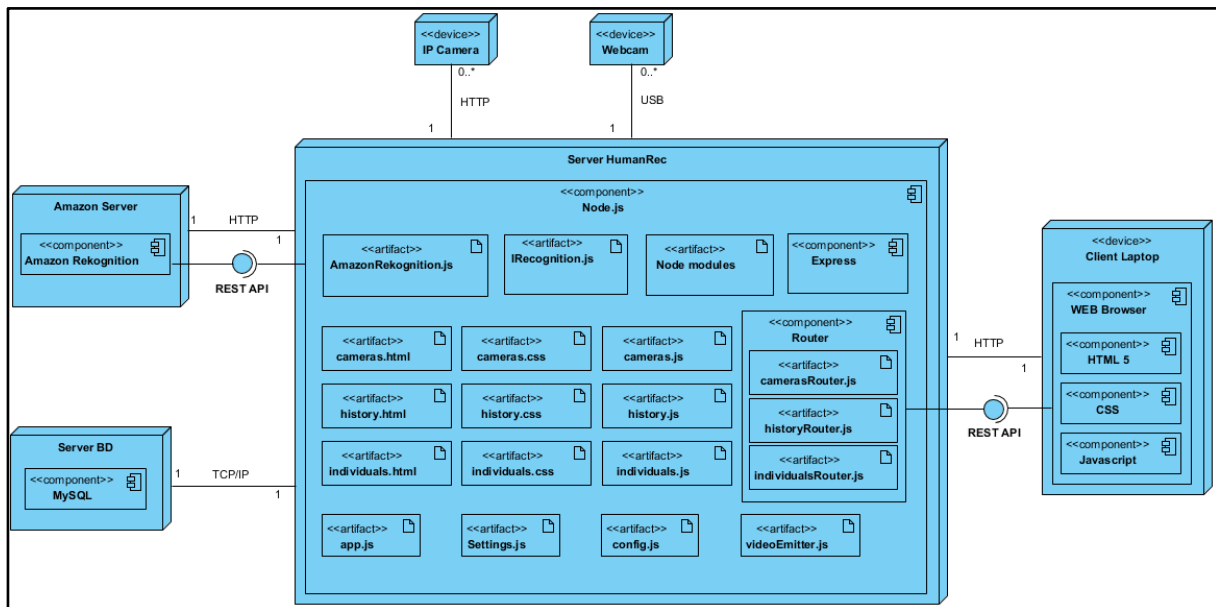


Figure 10 Diagramme de déploiement maximal

On retrouve donc la machine du client, un serveur dédié et un serveur de base de données, ici contenant une base de données MySQL. Cette configuration est celle d'une application déployée dans une entreprise voulant gérer une quantité importante de données. Le nombre d'individus connus par le système se compterait en milliers.

Sachant que le prototype développé dans le cadre de ce projet n'est pas déployé sur un serveur à part entière, le diagramme de déploiement ressemble plutôt à celui de la figure 11 ci-dessous.

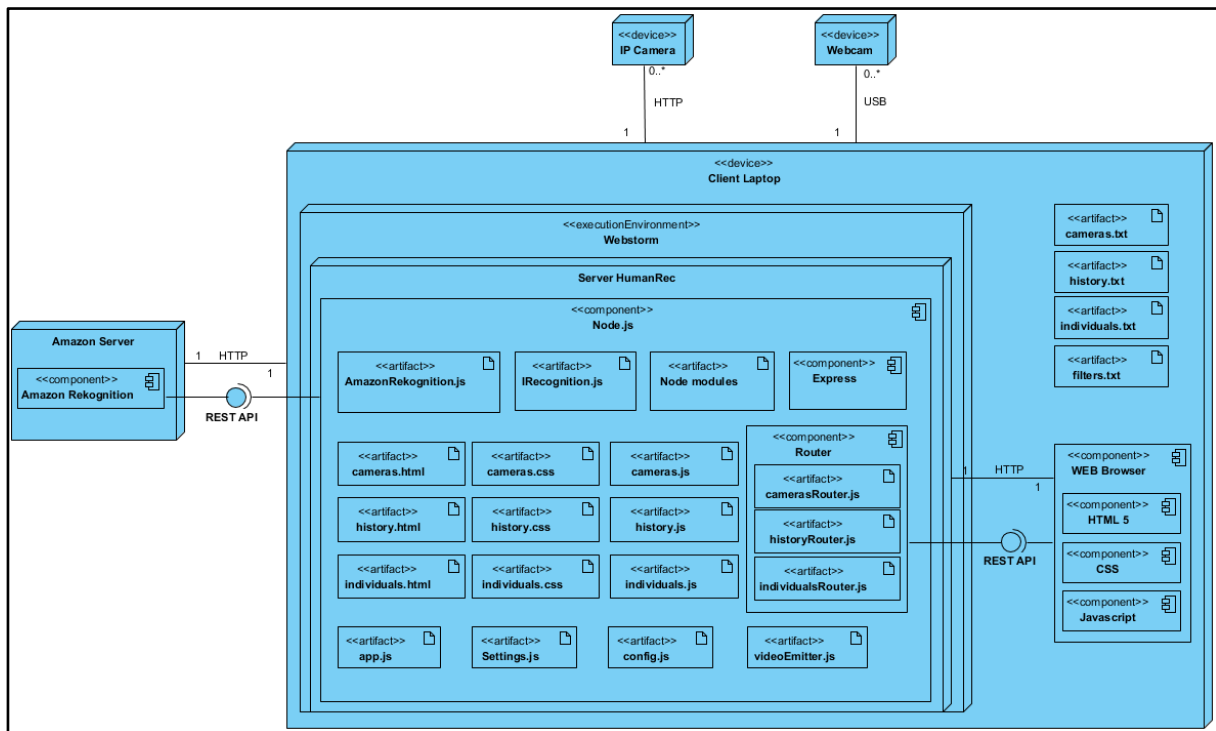


Figure 11 Diagramme de déploiement du prototype actuel

On remarque que la machine du client contient le serveur de l'application ainsi que les fichiers de persistance. Une base de données n'a pas été utilisée pour les raisons suivantes : la mise en place avec des fichiers textes est beaucoup plus rapide et la quantité de données traitées est faible en comparaison avec une utilisation en entreprise.

3.3 Diagramme de classe

Deux classes et une interface sont présentes dans le projet. La classe AmazonRekognition est la plus importante. C'est cette classe qui est chargée de faire les appels au service d'Amazon. Elle implémente l'interface IRecognition pour assurer la compatibilité du programme s'il on souhaite utiliser un autre service de reconnaissance faciale tel que Microsoft Cognitives Services ou encore un système développé par la Haute Ecole d'ingénierie et d'architecture de Fribourg. Une classe également disponible est la class Cam. Chaque fois qu'une caméra est créée, elle est représentée par un objet Cam. Cependant, l'utilisation d'une classe n'était pas forcément obligatoire, un objet Cam aurait pu être réalisé avec un objet JavaScript. La notion de classe en JavaScript a été introduite avec la sortie du standard ECMAScript 6 [14].

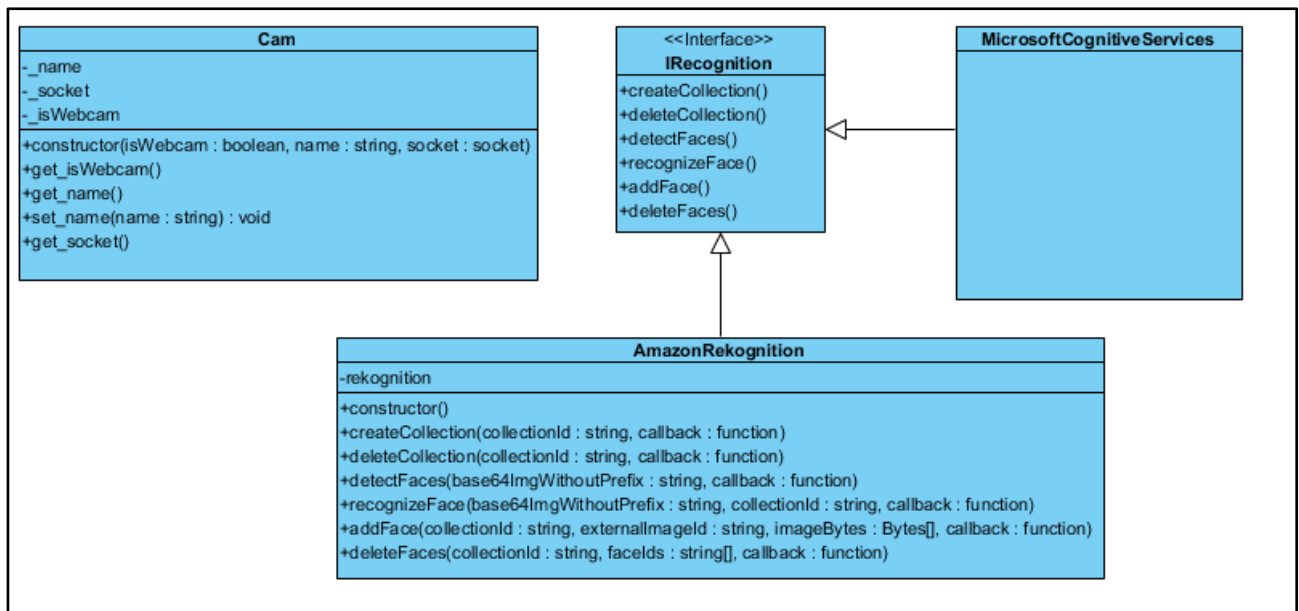


Figure 12 Diagramme de classe

3.4 Objets du système

Les objets du système sont les suivants : Cameras, Filters, History et Individuals.

Ce sont ces objets de type JSON qui sont persistés sur les fichiers textes. Une fois que le prototype est arrêté puis relancé, la restauration de l'état du système est possible grâce à ces objets de persistance. Ci-dessous, on aperçoit la structure de chaque objet avec en dessous un exemple et à côté les explications de chaque champ.

<pre> [{ "isWebcam": boolean, "cameraName": string, "cameraNum": string, "IPAddr": string }] [{ "isWebcam": false, "cameraName": "camHall", "cameraNum": "", "IPAddr": "http://169.254.194.26/mjpg/video.mjpg" }] </pre>	<p>isWebcam Définit si la caméra est une webcam ou une caméra IP.</p> <p>cameraName Nom unique de la caméra choisi par l'utilisateur.</p> <p>cameraNum Si la caméra est une webcam, alors ce champ a pour valeur l'index du device, connu par l'OS et utilisé par la librairie OpenCV. Sinon, ce champ est à vide.</p> <p>IPAddr Si la caméra est une caméra IP, alors ce champ a pour valeur l'url d'accès à la caméra. Sinon, ce champ est à vide.</p>
--	--

Figure 13 cameras.txt

<pre> { "similarity" : number, "cameras" : [string], "from" : Date, "to" : Date, "identifiant" : string, "tracking" : [string] } { "similarity" : 70, "cameras" : ["camHall", "camRoom"], "from" : "2018-07-10T08:27:00.000Z", "to" : "2018-07-10T08:30:00.000Z", "identifiant" : "Bastien_Monney", "tracking" : ["ID_sf5rsf15-4u12-5qj3-sdf65sf24", "Bastien_Monney", "Nicolas_Fuchs"] } </pre>	<p>similarity Option qui permet de filtrer l'historique d'un individu selon le pourcentage de similarité entre le visage capturé de la caméra et les métriques du visage trouvé sur Amazon.</p> <p>cameras Option qui permet de filtrer l'historique d'un individu selon les caméras souhaitées.</p> <p>from Option qui permet de filtrer l'historique d'un individu selon une date de début.</p> <p>to Option qui permet de filtrer l'historique d'un individu selon une date de fin.</p> <p>Identifiant Caractérise l'individu dont l'historique est affiché.</p> <p>Tracking Caractérise tous les individus dont le tracking a été activé. Au niveau de l'interface, cela correspond aux options des individus dont l'historique peut être affiché.</p>
---	--

Figure 14 filters.txt

<pre> { "{identifiant}" : [{ "fullImage" : { "type" : "Buffer", "data" : [number] }, "detectedFace" : { "type" : "Buffer", "data" : [number] }, "similarity" : number, "camera" : string, "date" : Date }] } { "Nicolas_Fuchs" : [{ "fullImage" : { "type" : "Buffer", "data" : [201, 56, 3, 123, ...] }, "detectedFace" : { "type" : "Buffer", "data" : [78, 2, 45, 236, ...] }, "similarity" : 95.4632628, "camera" : "camHall", "date" : "2018-07-10T09:32:25.000Z" }] } </pre>	<p>{identifiant} Identifiant de l'individu qui a été détecté. Cet identifiant est de type string.</p> <p>fullImage Image capturée par la caméra dans laquelle le visage (detectedFace) a été détecté. Ce champ contient un objet JSON avec à l'intérieur les champs type dont la valeur constante est "Buffer" et data, un tableau de bytes de l'image encodée en base64.</p> <p>detectedFace Image du visage détecté, ce champ a le même format que le champ fullImage.</p> <p>similarity pourcentage de similarité entre le visage détecté et l'individu trouvé dans la base de données d'Amazon.</p> <p>camera Nom de la caméra qui a capturé l'individu.</p> <p>date Date à laquelle l'individu a été capturé.</p>
---	--

Figure 15 historics.txt

<pre> { "{identifiant}" : { "age" : number, "description" : string, "comment" : string, "known" : string, "tracking" : string, "images" : [{ "base64Img" : string, "faceId" : string }] } } </pre>	{identifiant}	Identifiant de l'individu qui a été détecté. Cet identifiant est de type string. Il prend deux formes différentes : soit c'est un identifiant unique généré par le module uuidv4 car le système ne connaît pas encore cet individu, soit c'est le nom sans espace entré par l'utilisateur.
	age	Age de l'individu estimé par l'utilisateur.
<pre> { "ID_sf5rsf15-4u12-5qj3-sdf65sf24" : { "age" : 23, "description" : "The stealing student", "comment" : "-", "known" : "no", "tracking" : "yes", "images" : [{ "base64Img" : </pre>	description	Description de l'individu entrée par l'utilisateur.
<pre> "data:image/png;base64,/9j/4AA..., 442-4fe0-4ds31e" }] } } </pre>	comment	Commentaire à propos de l'individu entré par l'utilisateur.
<pre> "base64Img" : </pre>	known	Si l'identifiant commence par "ID_", l'individu est inconnu. Sinon, il est considéré comme connu.
<pre> "faceId" : "5cbfb-5 442-4fe0-4ds31e" }] } } </pre>	tracking	Définit si le tracking a été activé sur cet individu ou non.
	Images	Images de l'individu. La valeur de ce champ est un objet JSON qui contient les champs base64Img qui est l'image encodée en base64 et faceId qui est l'identifiant des métriques du visage stockées sur Amazon.

Figure 16 individuals.txt

3.5 Définition des interfaces utilisateur

Les interfaces ont été créées à partir du site <https://heiafr.mybalsamiq.com>. Cet outil a permis de créer des wireframes de haute-fidélité pour le projet HumanRec.

Plusieurs versions des maquettes existent, seule la première et la dernière version sont présentées dans ce chapitre. Les autres versions sont disponibles dans les annexes de ce document.

3.5.1 Version 1

La figure 17 de la page suivante représente la page WEB générale. Elle est décomposée en deux parties : La partie supérieure qui s'occupe d'afficher les vidéos et images des caméras et la partie inférieure qui s'occupe de l'affichage de l'historique accompagnée la personne à traquée ainsi qu'un bouton pour ajouter un individu au système.

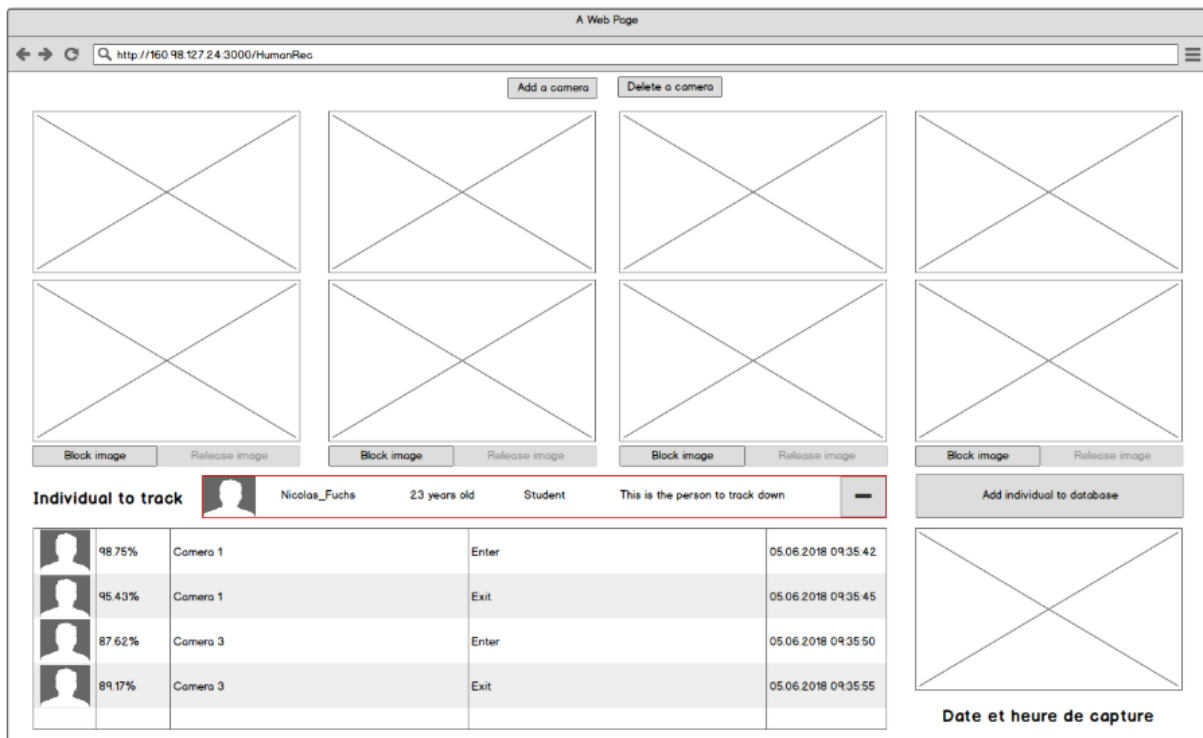


Figure 17 Interface générale

La Figure 18 ci-dessous nous donne une idée de la procédure d'ajout d'une caméra.

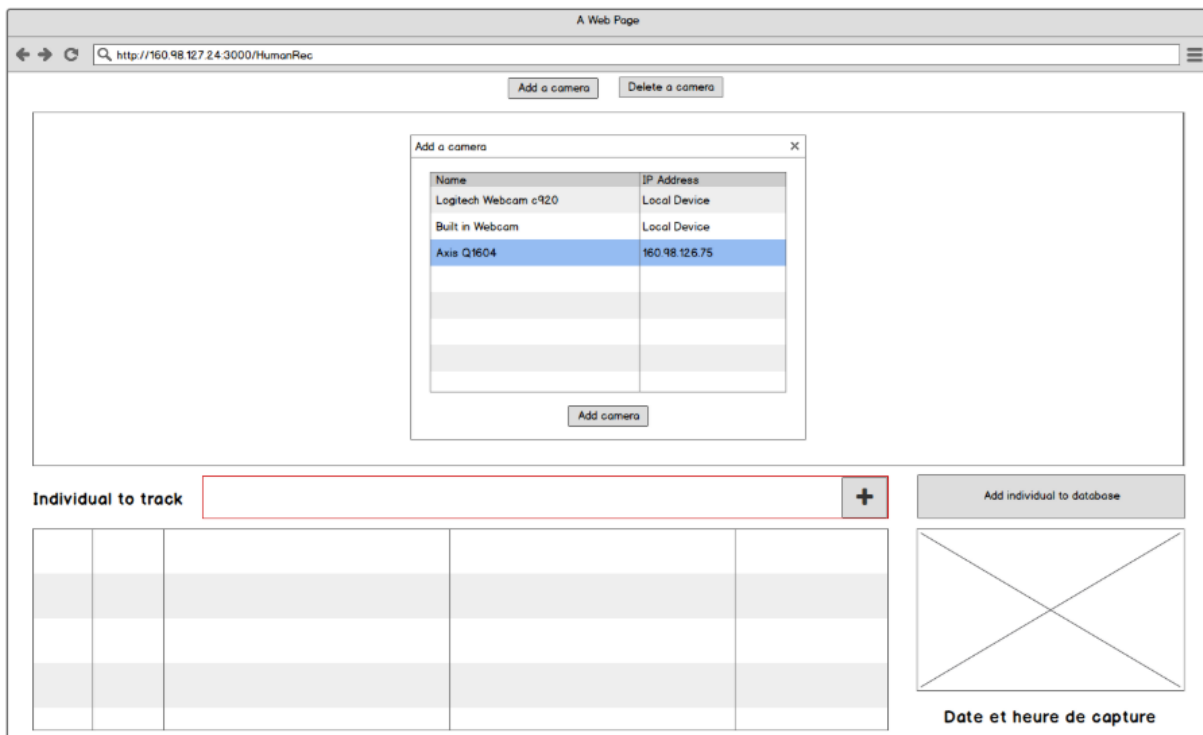


Figure 18 Ajout d'une caméra

L'interface visuelle de la Figure 19 ci-dessous montre le mode de suppression de caméra(s). Le bouton "Delete a camera" ne change pas de texte lorsque l'on entre ou sort du mode. Son effet semble ambigu.

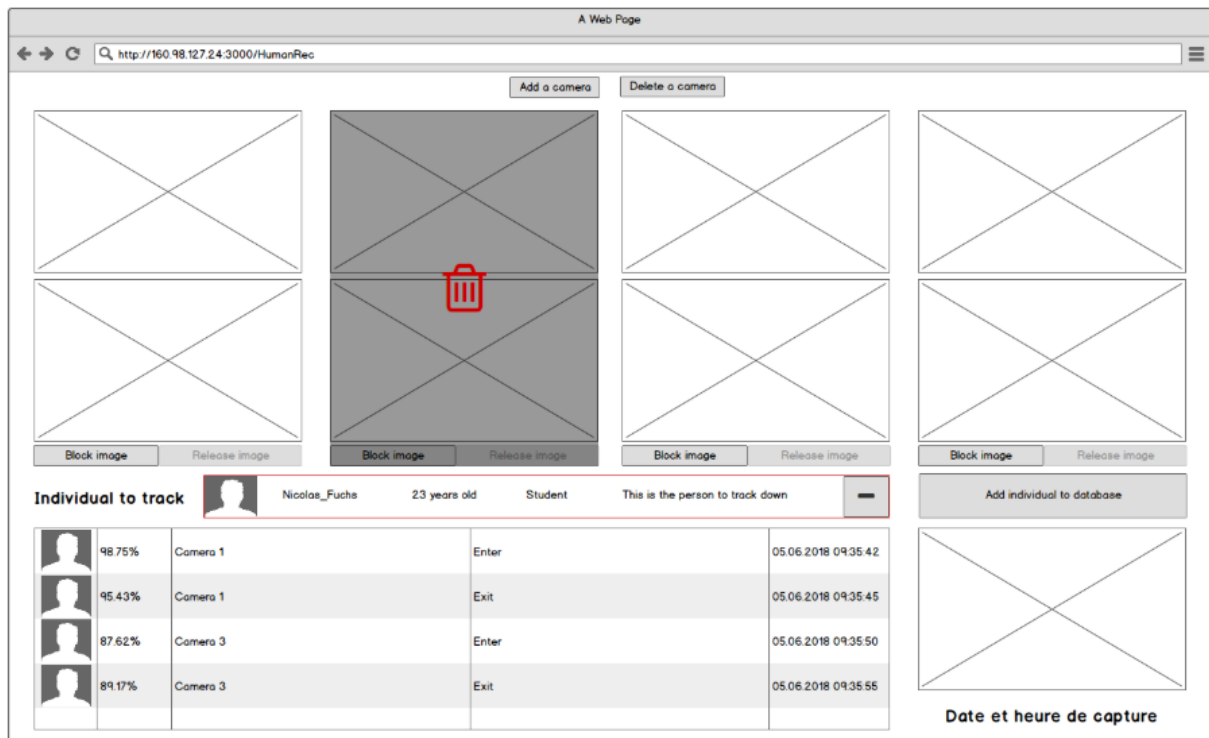


Figure 19 Suppression d'une caméra

La Figure 20 ci-dessous explique visuellement comment ajouter un individu au système. On ne peut ajouter qu'une seule image par individu créé.

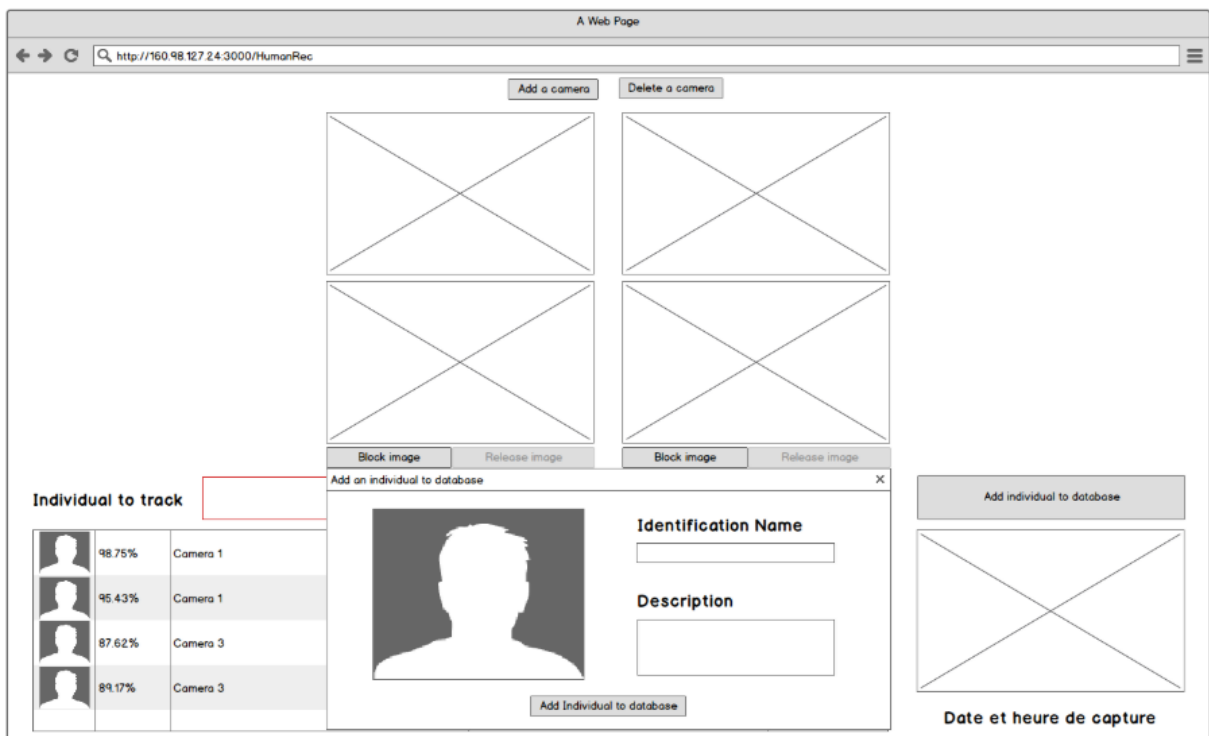


Figure 20 Ajout d'un individu au system

Cette dernière interface utilisateur de l'illustration ci-dessous explicite en partie la manière dont on choisit l'individu à traquer. En cliquant sur le plus, une table avec les individus existants apparaît.

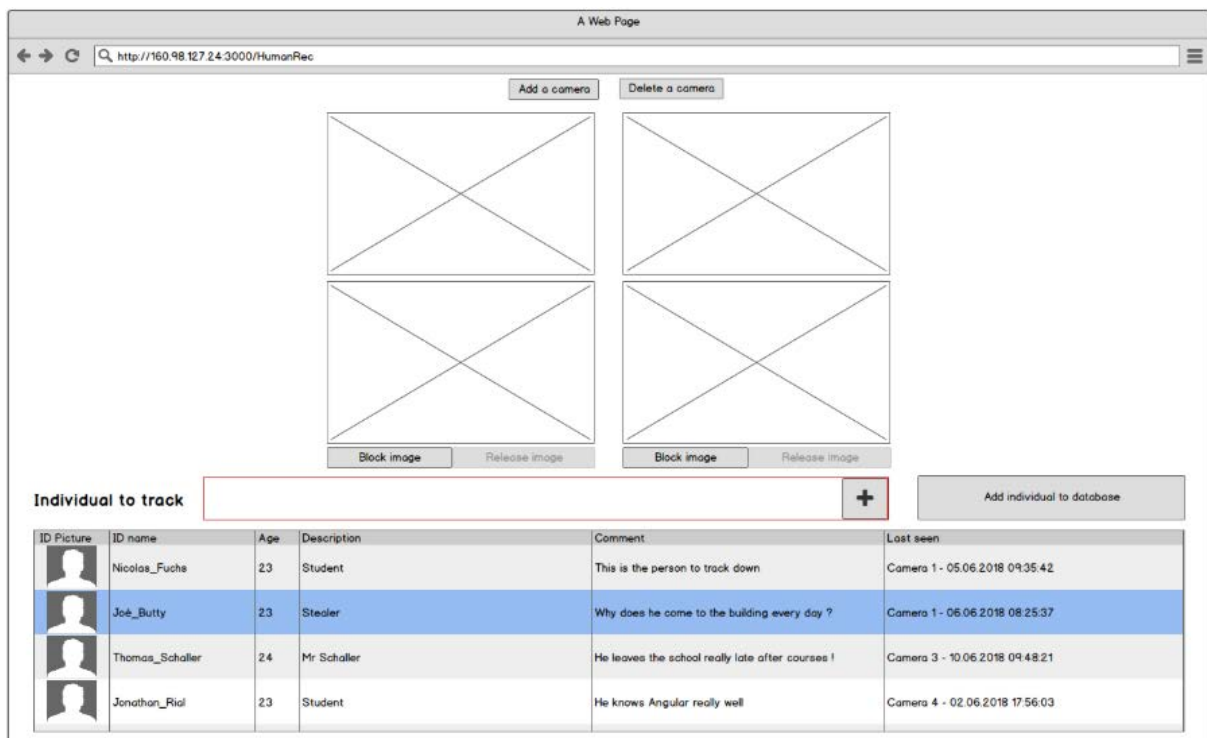


Figure 21 Sélection d'un individu à traquer

La première remarque globale pour cette première version des wireframes était que toutes les informations du prototype étaient accumulées dans une seule interface (page web). Je n'avais pas pensé au fait que l'on pouvait avoir à disposition des moniteurs supplémentaires pour éviter une interface trop chargée et inutilisable. C'est pourquoi dans la version 3 des maquettes, trois pages différentes sont visibles. Il manquait également l'option pour fusionner plusieurs personnes détectées. En effet, si le système ne reconnaît pas la même personne sur deux images qui représentent le même individu, l'utilisateur du système doit être en mesure de fusionner les deux entités créées dans la base de données de reconnaissance et dans la persistance locale. La notion de tracking n'était pas très claire non plus. Lorsque l'on clique sur le plus pour ajouter une personne à traquer, on ne sait pas si on joue sur l'affichage ou l'activation du tracking sur la personne choisie. De plus, une contrainte qui avait été définie par moi-même lors de la réalisation de cette version des wireframes était que l'utilisateur ne pouvait pas ajouter plus de 4 caméras. Cette contrainte a du finalement être abandonnée afin de pouvoir ajouter autant de caméras que l'on souhaite. Finalement, l'édition d'une personne déjà enregistrée n'était pas prévue. Il a donc fallu passer par plusieurs itérations pour parvenir aux maquettes qui correspondent à l'application voulue. Ce sont donc ces dernières maquettes (version 3) qui sont exposées dans la prochaine section.

3.5.2 Version 3

La version 3 des wireframes est la dernière version et en conséquence la plus proche de l'implémentation.

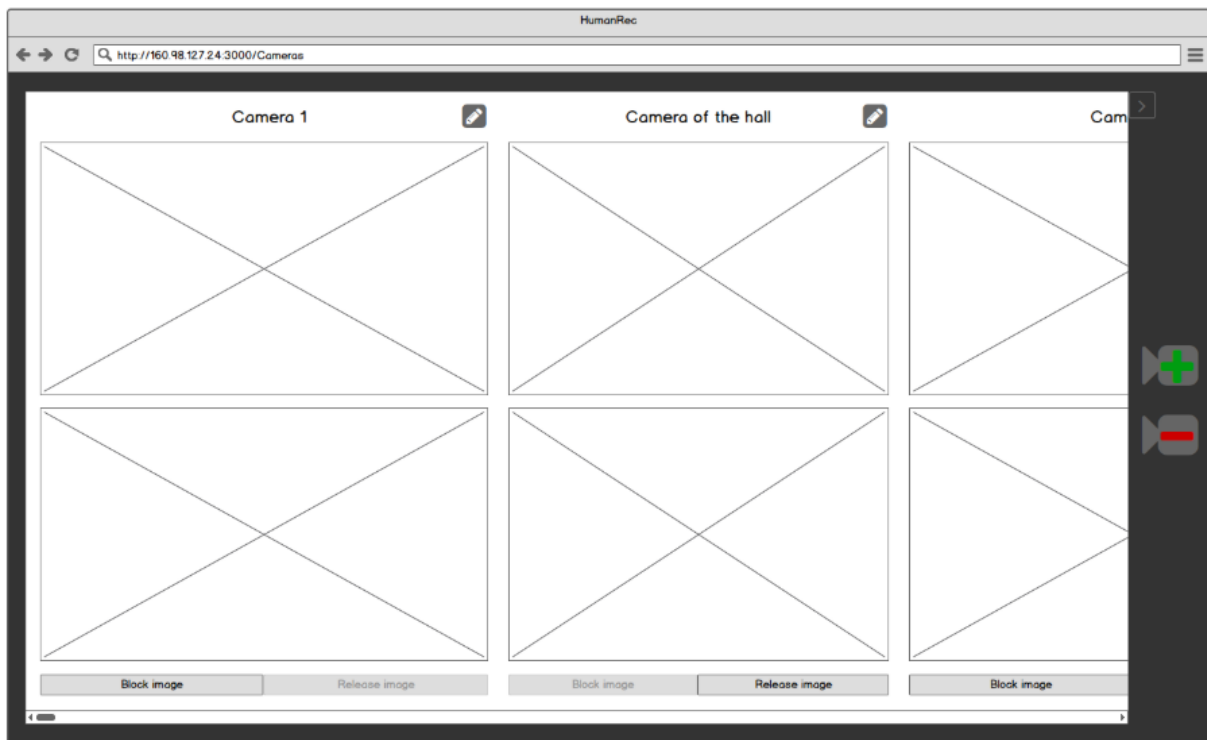


Figure 22 Cameras - Interface générale

La Figure 22 ci-dessus montre l'interface de base de la page cameras. On peut y modifier le nom des caméras, mettre une image sur pause, ajouter ou supprimer des caméras.

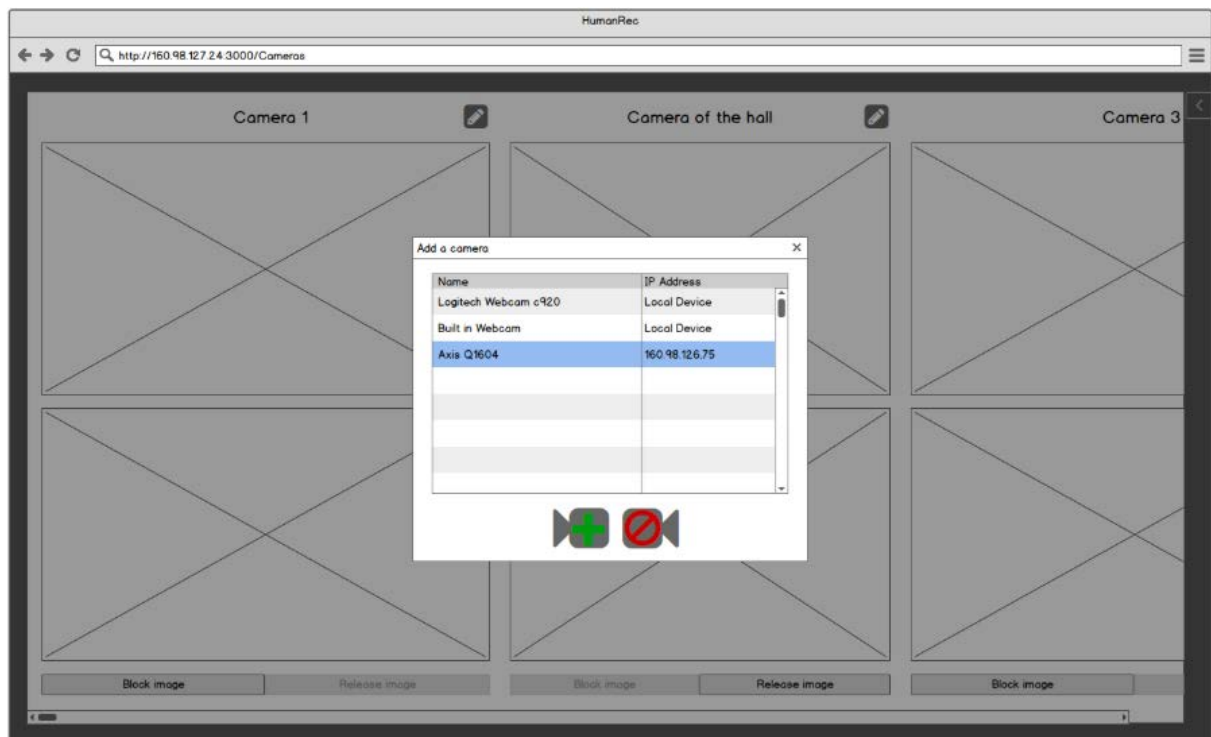


Figure 23 Cameras - Ajout d'une caméra

La Figure 23 ci-dessus représente la fenêtre d'ajout d'une caméra. L'implémentation n'est pas conforme par rapport à cette fenêtre. Le lien entre l'index du périphérique et son nom n'a pas pu être récupéré. A la place, on a un champ texte pour une url ou un input type nombre pour un index.

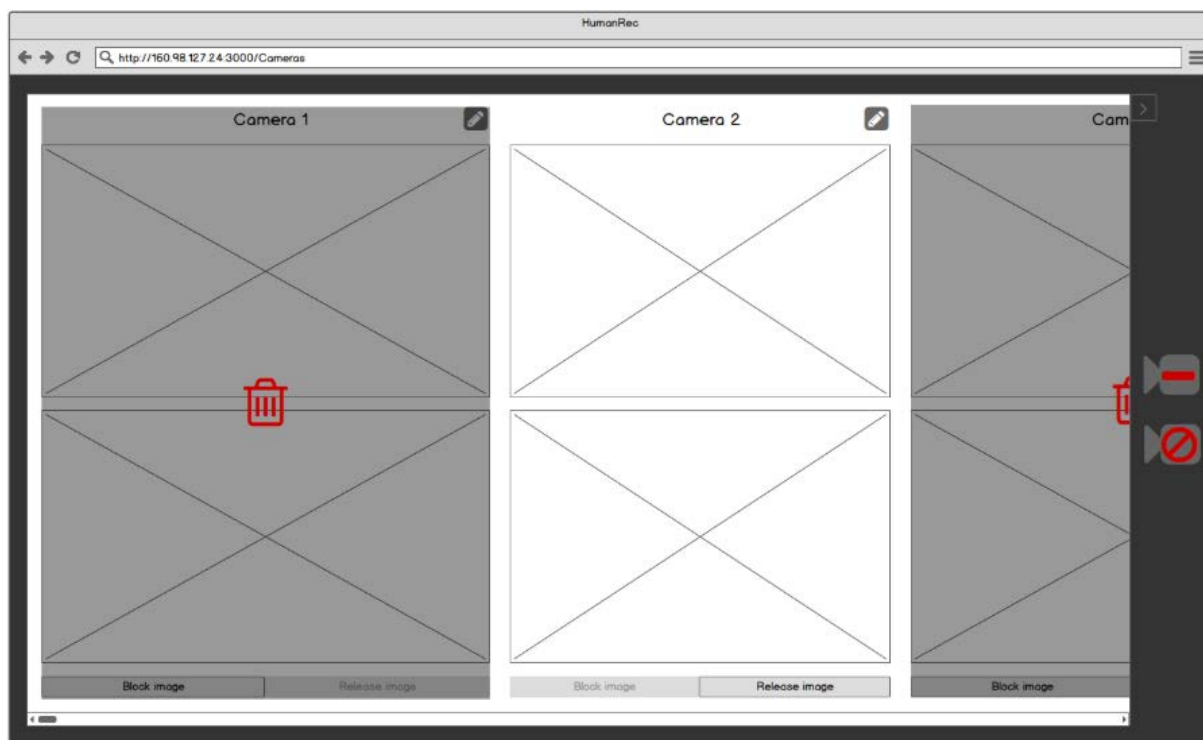


Figure 24 Cameras - Suppression de caméra(s)

Dans la Figure 24 ci-dessus, l'utilisateur se trouve dans le mode de suppression de caméras. Il peut sélectionner et désélectionner les caméras qu'il souhaite et finalement valider la suppression en cliquant sur le moins ou annuler la suppression en appuyant sur le bouton du dessous. Dans tous les cas, le mode de suppression est quitté.

Detected face	Similarity	Camera	State	Date
	98.75%	Camera 1	Enter	05.06.2018 09:35:42
	95.43%	Camera 1	Exit	05.06.2018 09:35:45
	87.62%	Camera 3	Enter	05.06.2018 09:35:50
	89.17%	Camera 3	Exit	05.06.2018 09:35:55
	75%	Camera of the hall	Enter	05.06.2018 10:02:34
	86%	Camera of the hall	In	05.06.2018 10:02:37
	99.5%	Camera of the hall	In	05.06.2018 10:02:40
	98.22%	Camera of the hall	Exit	05.06.2018 10:02:43

Figure 25 History - Interface générale

L'interface de la Figure 25 de la page précédente est divisée en deux parties. Celle du haut contient les filtres que l'on peut appliquer sur un historique. L'utilisateur peut sélectionner l'individu dont il veut voir l'historique. Celui-ci est représenté sous la forme d'une table contenant le visage détecté, la similitude du visage par rapport au modèle stocké chez Amazon, le nom de la caméra ayant capturé l'individu et la date et l'heure de capture. Lorsque l'on clique sur une entrée de cette table, l'image complète est affichée sur l'espace à droite.

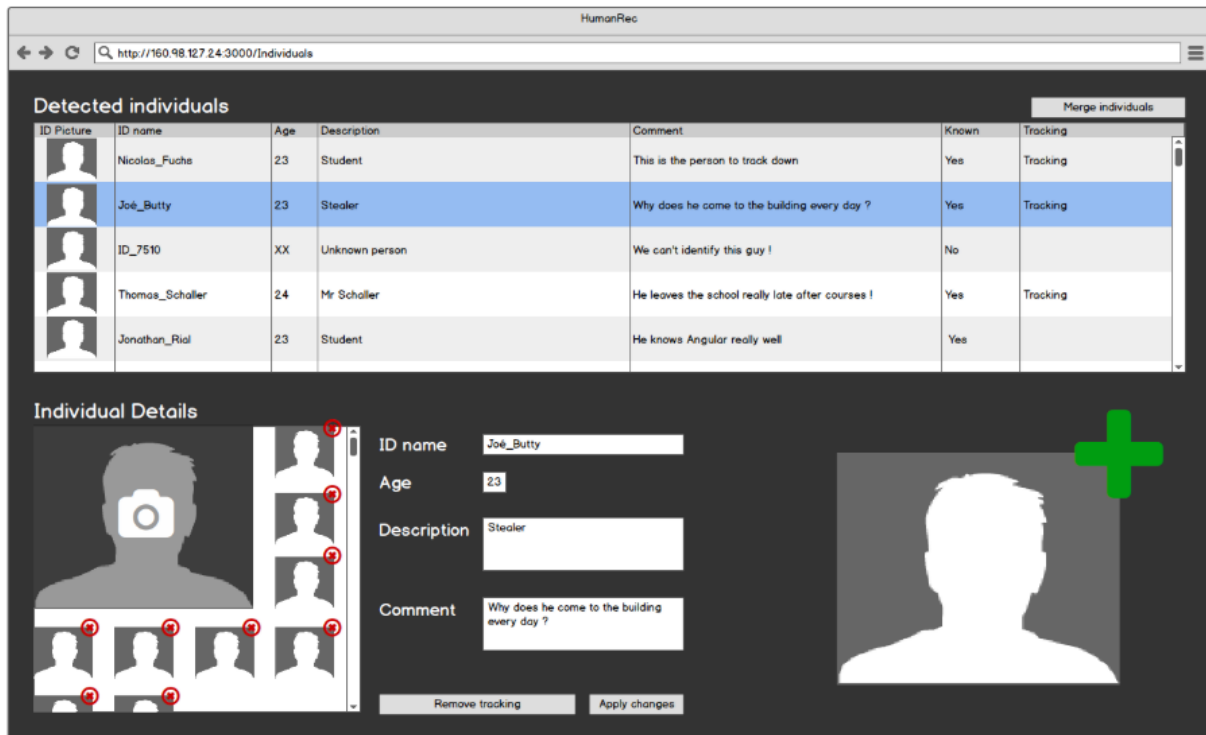


Figure 26 Individuals - Interface générale

La Figure 26 ci-dessus expose la page réservée aux individus. C'est dans cette interface que les personnes détectées par le système sont gérées. On y découvre une table avec tous les individus enregistrés, un espace en bas à gauche chargé d'afficher les détails d'un individu sélectionné dans la table, et en bas à droite, c'est un bouton qui rend possible l'ajout d'une personne à partir d'images locales. Ce qui diffère avec l'interface implémentée, ce sont les boutons d'interactions d'ajout et de suppression d'images. Le reste de la vue est fidèle. Un bouton qui cache une fonctionnalité primordiale, pourtant pas imaginée au commencement du projet, est positionné juste au dessus de la table à droite. Le texte du bouton est suffisamment explicite pour connaître les raisons de son utilisation. L'utilisateur fera l'usage de cette fonctionnalité dans le but de fusionner les images de plusieurs individus. Puisque le système ajoute les personnes non reconnues automatiquement, plusieurs individus pourraient s'avérer être la même personne physique. La solution, c'est de les fusionner ensemble.

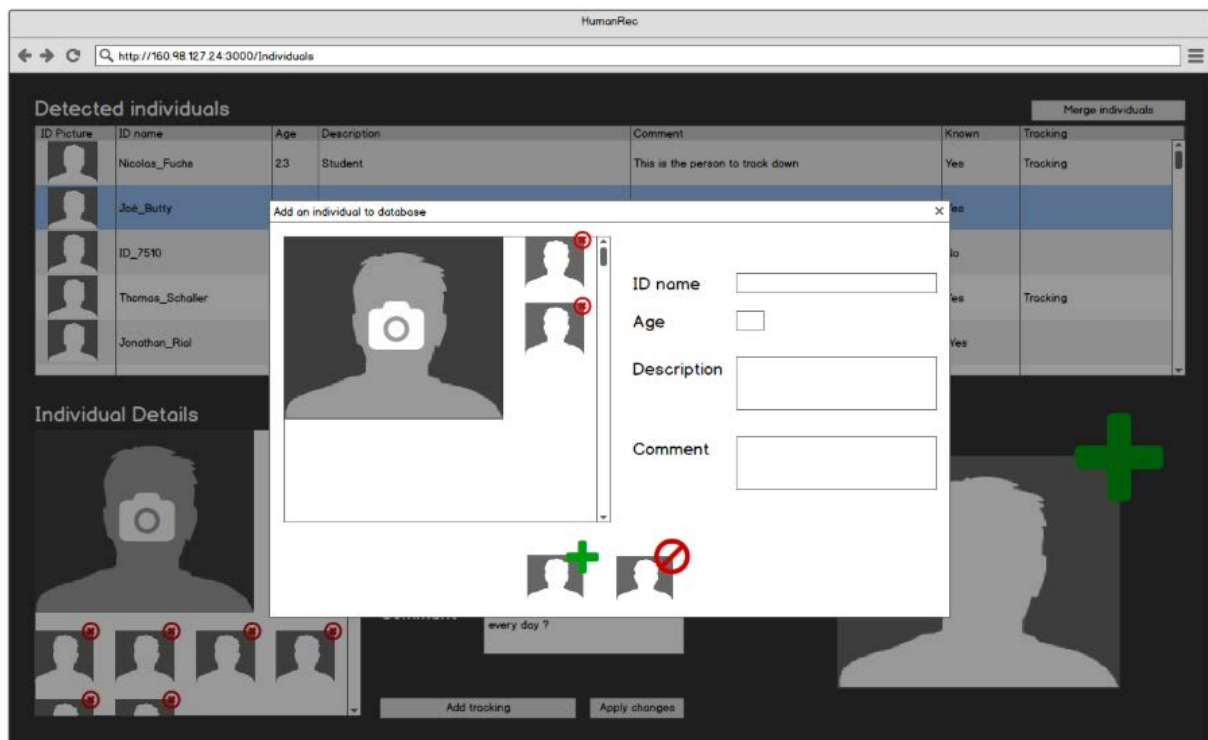


Figure 27 Individuals - Ajout manuel d'un individu au système

La fenêtre d'ajout d'un individu est la même que la section pour l'affichage des données d'une personne.

Dans cette dernière version des maquettes, toutes les informations affichées à l'utilisateur ont été séparées dans trois pages différentes (1 page par moniteur est idéal). La page 'Cameras' est dédiée à la gestion des caméras (ajout, retrait et changement de nom). La page 'History' sert à afficher un historique de captures d'un individu préalablement sélectionné. Cet historique peut être filtré au travers du pourcentage de similitude des visages entre celui qui a été détecté et celui qui se trouve chez Amazon, les caméras ayant capturés l'individu traqué ainsi que la date de début et de fin de capture. Finalement, la page 'Individuals' s'occupe de la gestion des individus détectés par le système. Ces individus peuvent être connus ou inconnus. La gestion des individus inclut l'ajout ou le retrait d'images, le changement de l'identifiant ou d'informations secondaires (âge, description, commentaire), ainsi que l'activation/désactivation du tracking. Un individu peut être ajouté de manière automatique lors de sa détection par le système ou de manière manuelle avec des images et informations entrées par l'utilisateur.

3.6 Synthèse

Tous les sous-chapitres de la conception mettent en avant le caractère catégorisé du projet. La séparation des domaines est claire : la gestion des caméras, la gestion de l'historique et la gestion des individus. Des informations sont échangées entre les pages par l'API REST entre autre et un stockage mis à disposition par les navigateurs récents, nommé localStorage. Ce dernier concept est abordé dans le sous-chapitre *Binding entre les pages* du prochain chapitre, l'implémentation du prototype.

4 Implémentation

Ce chapitre traite dans un premier temps de la documentation de l'API REST, suivie de l'envoi des frames pour l'affichage des images dynamiques (vidéo "live") et les images statiques (images analysées toutes les 3 secondes). Ensuite, l'utilisation du service de détection et de reconnaissance faciale, en l'occurrence Amazon Rekognition, est dévoilée. Par la suite, une proposition d'amélioration de la reconnaissance faciale est expliquée.

L'implémentation de ce projet a été menée à l'aide de l'IDE Webstorm de la suite JetBrains. Une partie du code a pu être repris d'un projet déjà existant [15].

4.1 Documentation de l'API REST

Dans ce sous-chapitre, une documentation de l'API REST humanrec accessible par des requêtes http est disponible. Ce service web a été développé en nodejs avec express, un framework web répandu pour la création efficace d'une API fiable. L'API humanRec n'est disponible que localement à l'adresse <http://localhost:3000>.

Cameras

GET	/cameras/getCameras
-----	---------------------

Retrieves all cameras of the system

POST	/cameras/addCamera	Body { isWebcam: boolean, //obligatoire cameraName: string, //obligatoire cameraNum: number, //obligatoire si IPAddr IPAddr: string //obligatoire si cameraNum }
------	--------------------	---

Adds a camera to the system

PUT	/cameras/updateCamera	Body { oldCameraName: string, //obligatoire newCameraName: string, //obligatoire }
-----	-----------------------	---

Updates a camera of the system (name)

DELETE	/cameras/removeCameras	Body { cameras : [cameraNames] //obligatoire }
--------	------------------------	--

Removes camera(s) from the system

GET	/cameras/createCameraSocket?cameraNum=&rate=
-----	--

Creates a server socket to send frames data from camera

POST	/cameras/createIndividualsSocket
------	----------------------------------

Creates the server socket to send individuals data

History

GET	/history/getTracking	
Retrieves tracking individuals options		
GET	/history/getFullImage?identifier=&similarity=&date=	
Retrieves the full image in which the face has been detected		
PUT	/history/updateFilters	Body <pre>{ similarity: number, //optionnel cameras: [cameraNames], //optionnel from: date, //optionnel to: date, //optionnel identifier: string, //optionnel addTrackingIdentifier: [individualIdentifiers], //optionnel removeTrackingIdentifier: [individualIdentifiers] //optionnel }</pre>
Updates filters		
GET	/history/getFilters	
Retrieves all filters		
GET	/history/getHistory?identifier=	
Retrieves history of an individual		
PUT	/history/updateHistoryIdentifier	Body <pre>{ oldIdentifier: string, //obligatoire newIdentifier: string //obligatoire }</pre>
Updates an identifier in history		
PUT	/history/updateHistoryCamera	Body <pre>{ oldCameraName: string, //obligatoire newCameraName: string //obligatoire }</pre>
Updates a camera name in history		
PUT	/history/mergeHistories	Body <pre>{ targetIdentifier: string, //obligatoire identifiersToMerge: [individualIdentifiers] //obligatoire }</pre>
Merges histories of several individuals		
DELETE	/history/removeHistory	Body <pre>{ identifier : string //obligatoire }</pre>
Removes history of an individual		

Individuals

GET	/individuals/getOneIndividual?identifier=	
Retrieves one specific individual		
GET	/individuals/getAllIndividuals	
Retrieves all individuals		
POST	/individuals/addIndividual	Body <pre>{ identifier: string, //required age: number, //required description: string, //required comment:string, //required known: string, //required tracking: string, //required images: [base64Images] //optional }</pre>
Adds a new individual to the system		
PUT	/individuals/updateIndividual	Body <pre>{ oldIdentifier: string, //required newIdentifier: string, //required newAge: number, //required newDescription: string, //required newComment: string, //required tracking: string //optional }</pre>
Updates an individual in the system		
DELETE	/individuals/removeIndividual	Body <pre>{ identifier : string //required }</pre>
Removes an individual from the system		
PUT	/individuals/mergeImages	Body <pre>{ targetIdentifier: string, //required identifiers: [IndividualIdentifiers], //required }</pre>
Merges several individuals images		
PUT	/individuals/addImages	Body <pre>[base64Images] //required</pre>
Adds individual image(s) to the system		
PUT	/individuals/removeImage	Body <pre>{ identifier: string[base64Images], //required image: base64Image //required }</pre>
Adds individual image(s) to the system		

4.2 Récupération d'un flux vidéo

Diverses sources vidéos ont été mises à disposition pour mener à bien ce projet : une caméra IP et une webcam. Dans le but d'afficher dans une page web ce que ces appareils capturent, il est essentiel dans un premier temps de récupérer leur flux respectif. La méthode de récupération diffère selon le type de device. Le premier sous-chapitre se consacre donc à la webcam, tandis que le second sous-chapitre se concentre sur la caméra IP.

4.2.1 Récupération d'un flux d'une webcam

D'abord, la webcam doit être branchée à un port usb de la machine du serveur, dans le cas du prototype actuel ma propre machine. La librairie *OpenCV* est capable, en recevant en paramètre l'index du périphérique, de capturer des images venant de la source correspondante. *OpenCV* doit être installé au préalable également sur la machine du serveur et par dessus vient s'ajouter un module (*OpenCV*) installable avec npm (Node Package Manager) (module *opencv*), qui s'occupe du linkage entre node et opencv. On peut voir ci-dessous dans la Figure 28 le code chargé de récupérer les images d'une webcam.

```
// Camera properties
const camWidth = 1280;
const camHeight = 720;

const camera = new cv.VideoCapture(parseInt(cameraNum));
camera.setWidth(camWidth);
camera.setHeight(camHeight);
setInterval(function() {
  camera.read(function(err, im) {
    if (err) {
      throw err;
    } else if (!im.empty()) {
      if (parseInt(camInterval)/1000 >= 1) {
        im.save('D:\\WebstormProjects\\HumanRec\\routes\\capturedFrames\\' + new Date().toISOString().replace(/:/g, '_') +
          cameraNum.replace(/:/g, '_') + '.jpeg');
        detectAndRecognize(socket, im, cameraNum, true);
      } else {
        socket.emit('frame', {buffer: im.toBuffer()});
      }
    }
  });
}, camInterval);
```

Figure 28 Récupération d'un flux d'une webcam

Tel que présenté dans les maquettes, afin de satisfaire au mieux les besoins de l'utilisateur, une table avec les noms des caméras IP ainsi que les webcams serait idéale. Il existe plusieurs façons d'obtenir les noms des caméras disponibles. Cependant, aucun moyen exploitable n'a été trouvé ni pour obtenir l'URL de flux mjpg pour les caméras IP ni pour lier l'index du périphérique à son nom. En ce qui concerne le linkage entre le nom d'un périphérique et son index, une méthode a pu être trouvée uniquement de manière visuelle. Sur le système d'exploitation Windows 10, l'information a pu être trouvée grâce au gestionnaire de périphériques. La Figure 29 de la page suivante montre l'endroit précis où se trouve l'index. En terme de programmation, l'information est introuvable. Le rectangle rouge désigne l'index qui est directement utilisable dans *OpenCV*. ⁵Un module développé en C++ est par contre disponible et son auteur affirme être capable de fournir l'accès à l'information du lien entre l'index et le nom du

⁵ <https://github.com/studiosi/OpenCVDeviceEnumerator>

périphérique. Ce module n'a pas été testé pour des raisons de manque de temps mais mérite une attention particulière pour d'éventuelles améliorations futures.

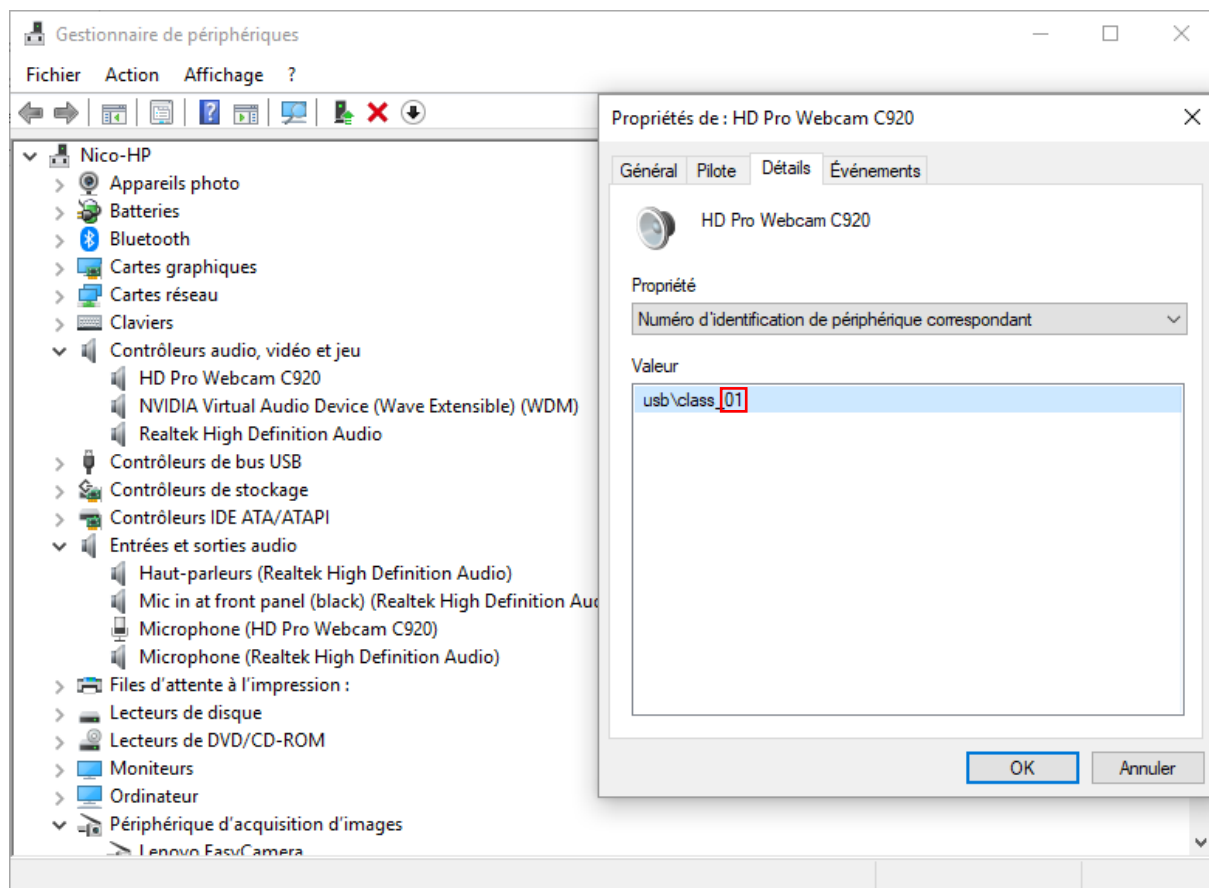


Figure 29 Index du périphérique HD Pro Webcam C920

4.2.2 Récupération d'un flux d'une caméra IP

En ce qui concerne le flux vidéo, on y a accès directement au travers de l'URL fournie par la caméra comme par exemple avec la caméra AXIS Q1604, l'URL d'accès est <http://IPAddress/mjpg/video.mjpg>. Il suffit juste de spécifier cette adresse dans l'attribut source d'une balise image d'une page html et la vidéo est retransmise en direct.

Des recherches ont été menées pour découvrir s'il existe un meilleur format d'image ou protocole pour récupérer un flux vidéo, tel que le protocole *RTSP* mais aucune recherche n'a abouti à un résultat concluant pour une inclusion dans une page web. Un flux vidéo a pu être visionné dans le lecteur VLC à l'aide du protocole *RTSP* mais aucun plugin VLC n'est actuellement compatible avec les navigateurs modernes.

Concernant la capture de frames (images extraites d'un flux vidéo), l'utilisation d'un module npm a été requise. Le module *mjpeg-camera* a simplement besoin d'un nom de caméra quelconque, un nom d'utilisateur et mot de passe pour se connecter à la caméra ainsi que l'URL d'accès. Le module s'occupe ensuite de récupérer une image avec la méthode 'getScreenShot'. Le format de l'image récupéré est différent de celui retourné par *OpenCV*, c'est pourquoi l'image est sauvee dans un fichier jpeg puis lue par *OpenCV*. Le nom du fichier est déterminé en fonction du nom de la caméra, de la date et de l'heure. On peut voir sur la page suivante sur la Figure 30 le code correspondant :

```
const IPCam = new MjpegCam({
  name: 'IPCam',
  user: 'root',
  password: 'humanrec',
  url: cameraNum,
  motion: true
});

setInterval(function() {
  IPCam.getScreenshot(function(err, im) {
    if (err) {
      throw err;
    }
    const tmpImgName = 'D:\\WebstormProjects\\HumanRec\\routes\\capturedFrames\\' + new Date().toISOString().replace(/:/|\./g,
      '_') + cameraNum.replace(/:|=|\?|\V|\./g, '_') + '.jpeg';
    fs.writeFileSync(tmpImgName, im);
    cv.imread(tmpImgName, function(err, im) {
      if (err) {
        throw err;
      } else if (!im.empty()) {
        fs.unlinkSync(tmpImgName);
        detectAndRecognize(socket, im, cameraNum, false);
      }
    });
  });
}, camInterval);
```

Figure 30 Récupération d'une image d'une caméra IP

4.3 Envoi d'un flux vidéo

Une fois que le flux est récupéré, il est alors possible d'envoyer ces données à la page `cameras.html` afin d'afficher ce que les caméras capturent. Contrairement à la récupération d'un flux, le code d'envoi d'un "flux" est semblable. Le premier sous-chapitre se focalisera sur la webcam, alors que le second se penchera sur la caméra IP.

La page `cameras.html` contient une ligne avec les vidéos "live" et la ligne juste en-dessous est constituée des images prises des vidéos respectives. Que l'on ait affaire à une caméra IP ou une webcam, l'envoi entre le serveur qui récupère les flux et le client qui les affiche est exécuté via les sockets pour transporter les données. Pour une webcam, deux sockets seront mis en place. Un pour la partie dynamique (vidéo) et l'autre pour la partie statique (image). Pour une caméra IP, seule la partie statique est demandée puisque la partie dynamique a accès à la vidéo par l'url. Un port ainsi qu'un intervalle de temps est défini dans la requête de création du socket. La difficulté dans le choix de l'intervalle de temps est de choisir un intervalle pas trop long pour empêcher que les individus puissent passer devant les caméras sans être détectés tout en sachant qu'il faut éviter un intervalle trop court sans quoi le buffer du socket se remplira trop rapidement et des bugs d'affichage apparaîtront. En fonction de ces contraintes, le temps d'intervalle de capture d'images a été fixé à 3 secondes.

Le code pour envoyer les données dans le socket est le suivant :

```
socket.emit('frame', {buffer: im.toBuffer()});
```

Figure 31 Envoi d'une image dans un socket

4.4 Caractéristiques des caméras

4.4.1 AXIS Q1604

Le modèle de la caméra IP prêtée est une AXIS Q1604. Ce produit a été commercialisé en Octobre 2011. Il est supporté encore jusqu'au 31 Mai 2021. Un produit de remplacement est déjà recommandé, la caméra IP AXIS P1364.

La caméra AXIS Q1604 est une caméra IP d'intérieur de résolution 1280 x 960 (ratio 4:3). La compression vidéo est possible avec les formats suivants : H.264 (MPEG-4), profiles main et baseline, Motion JPEG. Le streaming vidéo est possible avec les formats suivants : H.264 et Motion JPEG. Le frame rate est de 25/30 fps (Frames Per Second). Ce modèle⁶ offre la possibilité d'affiner le focus à distance. Par contre, on ne peut pas modifier l'inclinaison (tilt) ou la rotation de la caméra depuis un ordinateur. Deux modes sont disponibles : diurne et nocturne.



Figure 32 AXIS Q1604

Toutes ces informations proviennent des datasheets complètes disponibles aux adresses suivantes :

AXIS Q1604 https://www.axis.com/files/datasheet/ds_q1604_1493438_en_1508.pdf

AXIS P1364 https://www.axis.com/files/datasheet/ds_p1364_t10061529_en_1712.pdf

4.4.2 Logitech HD Pro Webcam C920



Figure 33 Logitech HD Pro Webcam C920

⁷Le modèle de la webcam prêtée est une Logitech HD Pro Webcam C920. Ce produit a été commercialisé en janvier 2012 et il est toujours d'actualité. Sa résolution est du full HD (1920 x 1080). Son frame rate s'élève à 30 fps. Le format de compression vidéo est du H.264. Aucun contrôle à distance n'est disponible pour ce périphérique. Le seul mode disponible est le mode diurne.

Toutes ces informations proviennent de la datasheet complète disponible à l'adresse suivante :

<http://static.highspeedbackbone.net/pdf/Logitech%20C920%20HD%20Pro%20Webcam%20Data%20Sheet.pdf>

4.4.3 Autres webcams

D'autres webcams telles que celle intégrée à l'ordinateur portable, Lenovo EasyCamera, ou encore une autre webcam personnelle, HD Webcam C270, se sont avérées utiles.

⁶ Source de l'image : <https://i2.cdscdn.com/pdt2/7/8/5/1/700x700/axi7331021034785/rw/axis-q1604-network-camera.jpg>

⁷ Source de l'image : <https://ae01.alicdn.com/kf/HTB1XMxRQVXXXXXfPXXq6xXFXXE/Logitech-HD-Pro-Webcam-C920e-Appels-Vid-o-grand-cran-et-D-enregistrement-1080-p-Cam.jpg>

4.5 Service de détection et reconnaissance faciale

Comme indiqué dans le chapitre d'analyse de ce rapport, le service de détection et reconnaissance faciale choisi est celui d'Amazon principalement en raison de son taux de succès de reconnaissance faciale supérieur à celui de Microsoft Cognitive Services. Cette comparaison a été documentée dans le rapport des étudiants Alexandre Dessonnaz et Lucas Alborghetti. Ma décision s'est donc basée sur les constatations de mes collègues.

La procédure de création des comptes nécessaires à l'utilisation des services Amazon, bien expliquée et détaillée dans leur rapport, a été suivie et rapidement finalisée. Les données d'accès à ces comptes ont été ajoutées dans les annexes pour permettre une éventuelle reprise de projet.

Le service d'identification et reconnaissance faciale se nomme Amazon Rekognition.

L'identification et reconnaissance peuvent être lancées sur des images comme sur des vidéos, enregistrées ou en streaming. Pour des raisons de coûts et budget, il a été décidé que l'analyse devait être effectuée sur des images.

Rekognition a défini un certain nombre de contraintes dans le but de garantir des résultats exploitables. Les contraintes susceptibles de poser problème sont celles citées par la suite. L'identification des visages se passe de manière optimale lorsque la taille du visage le plus petit parmi ceux présents dans l'image est supérieure à ~5% de la plus petite dimension de l'image capturée. Par exemple, si la dimension de l'image capturée mesure 1920 pixels de largeur et 1080 pixels de hauteur, chaque visage présent dans l'image doit avoir des dimensions supérieures à 40 pixels selon la documentation. Au niveau de la reconnaissance, la taille minimale du visage est de 80 pixels. La taille maximale des images transmises par tableau de bytes pour une analyse est de 5Mo. En revanche, si les images sont fournies à partir d'Amazon S3, un espace de stockage de fichiers, la limite de taille d'une image est montée à 15Mo. La limite d'un nombre de transactions par seconde (TPS) varie selon les régions. Celle de l'Oregon, région choisie pour ce projet, est autorisée à lancer 50 TPS [15].

Même si toutes les contraintes d'Amazon sont respectées, les résultats ne sont pas garantis. En effet, le contexte de mise en place des caméras joue un rôle important sur l'efficacité du système. Les individus qui passent devant la caméra ont eux aussi un fort impact sur la réussite ou l'échec de leur identification/reconnaissance. Il suffit d'avoir le visage caché en l'inclinant par exemple, volontairement ou non, pour éviter que la caméra nous reconnaisse. La position et le nombre de caméras sont en mesure de contrer partiellement ce problème. Ils sont essentiels et doivent par conséquent être déterminés de manière stratégique. Une solution serait par exemple de forcer l'individu à regarder la caméra, de manière directe ou indirecte. En ayant une gestion des accès par empreinte faciale par exemple, la personne qui désire franchir une porte est obligée de se mettre de face devant la caméra. Il serait également envisageable d'inciter fortement une personne à regarder en direction d'une caméra dissimulée derrière un capteur d'attention comme un panneau publicitaire. La pose du visage est un paramètre que le système peut difficilement modifier.

La luminosité est un autre facteur auquel il faut faire attention. Les visages présents sur des images brûlées ou des images trop sombres sont difficilement identifiables. L'idéal est donc de placer les caméras dans un espace éclairé de manière adéquate et constante.

Le code de la Figure 34 sur la page suivante montre les fonctions d'Amazon Rekognition utilisées par le programme. Les méthodes `createCollection` et `deleteCollection` sont rarement utilisées.

```

const settings = require('../Settings.js');
const IRecognition = require('./IRecognition.js');
const AWS = require('aws-sdk');

class AmazonRekognition extends IRecognition {
  constructor() {
    super();
    this.rekognition = new AWS.Rekognition({
      region: settings.region,
      accessKeyId: settings.accessKeyId,
      secretAccessKey: settings.secretAccessKey
    });
  }

  createCollection(collectionId, callback) {
    const params = { CollectionId: collectionId };
    this.rekognition.createCollection(params, function(err, data) {
      if (err) console.log('createCollection issue');
      else callback(data);
    });
  }

  deleteCollection(collectionId, callback) {
    const params = { CollectionId: collectionId };
    this.rekognition.deleteCollection(params, function(err, data) {
      if (err) console.log('deleteCollection issue');
      else callback(data);
    });
  }

  detectFaces(base64ImgWithoutPrefix, callback) {
    const bytes = new Buffer(base64ImgWithoutPrefix, 'base64');
    const params = { Image: { Bytes: bytes } };
    this.rekognition.detectFaces(params, function(err, data) {
      if (err) console.log('detectFaces issue');
      else callback(data);
    });
  }

  recognizeFace(base64ImgWithoutPrefix, collectionId, callback) {
    const bytes = new Buffer(base64ImgWithoutPrefix, 'base64');
    const params = { CollectionId: collectionId, Image: { Bytes: bytes },
      FaceMatchThreshold: parseFloat(settings.FaceMatchThreshold), MaxFaces: 1 };
    this.rekognition.searchFacesByImage(params, function(err, data) {
      if (err) console.log('searchFacesByImage issue');
      else callback(data);
    });
  }

  addFace(collectionId, externalImageId, imageBytes, callback) {
    const params = { CollectionId: collectionId, DetectionAttributes: [],
      ExternalImageId: externalImageId, Image: { Bytes: imageBytes } };
    this.rekognition.indexFaces(params, function(err, data) {
      if (err) console.log('IndexFaces issue');
      else if (data.FaceRecords.length !== 0) callback(data);
    });
  }

  deleteFaces(collectionId, faceIds, callback) {
    const params = { CollectionId: collectionId, FaceIds: faceIds };
    this.rekognition.deleteFaces(params, function(err) {
      if (err) console.log('deleteFaces issue');
      else callback();
    });
  }
}

module.exports = AmazonRekognition;

```

Figure 34 AmazonRekognition.js

4.6 Fonctionnement du prototype

La première étape, c'est la récupération des vidéos et des images des caméras/webcams. Les caméras IP sont employables uniquement si le client se trouve sur le même sous-réseau. Un switch a été mis à disposition pour relier indirectement ma machine à la caméra. Ensuite, une fonction d'identification et de reconnaissance faciale est lancée sur chaque image capturée espacée par un intervalle de temps de 3 secondes. Si l'individu n'est pas reconnu par le service, il est alors ajouté automatiquement dans le système. Cela implique de stocker les métriques de son visage sur Amazon et de l'ajouter à l'annuaire local des individus. Un ID unique lui est attribué à l'aide du module uuidv4 installable avec npm (Node package Manager). La version 4 est l'une des multiples versions disponibles pour ce module. Cette version génère des identifiants uniques de manière aléatoire. Comme le nom de la personne n'est pas connu lorsqu'elle entre dans le champ de la caméra, un identifiant unique fait office de nom. Que ce soit un ID généré aléatoirement ou entré par l'utilisateur, il ne doit pas contenir d'espaces. La conséquence de ce mode automatique est que chaque fois qu'un inconnu passe devant une caméra, il est ajouté au système. Si l'individu est reconnu par le système, une "entrée historique" contenant le visage détecté, l'image complète capturée par la caméra, la similarité, la caméra en question et la date de capture est ajoutée à l'historique des déplacements, uniquement dans le cas où le tracking est activé. Dans le cas contraire, aucun traitement supplémentaire n'est exécuté.

Malheureusement, le service d'Amazon n'est pas fiable à 100% bien évidemment et donc il arrive souvent qu'une même personne soit considérée comme deux différentes par une caméra pour plusieurs raisons possibles. Les deux explications les plus communes sont la détection d'un visage très petit ainsi que la position du visage qui n'est pas propice à la reconnaissance. Ce dernier aspect est discuté dans le sous-chapitre des tests de conditions de simulation.

Une fois que deux individus de la même personne ont été créés dans le système, il est possible de corriger l'erreur en fusionnant les deux individus créés. Les images de tous les individus sélectionnés sont ajoutées au dernier individu sélectionné. Toutes les informations par contre des individus à fusionner sont perdues. Il en a été décidé ainsi lors d'une séance hebdomadaire.

Dans l'idéal, il faut mettre en place des caméras sous différents angles et aux bons endroits pour assurer au minimum la capture d'un visage exploitable.

Dans la Figure 35 ci-dessous, un diagramme d'activités permet de mieux visualiser les étapes de la logique métier.

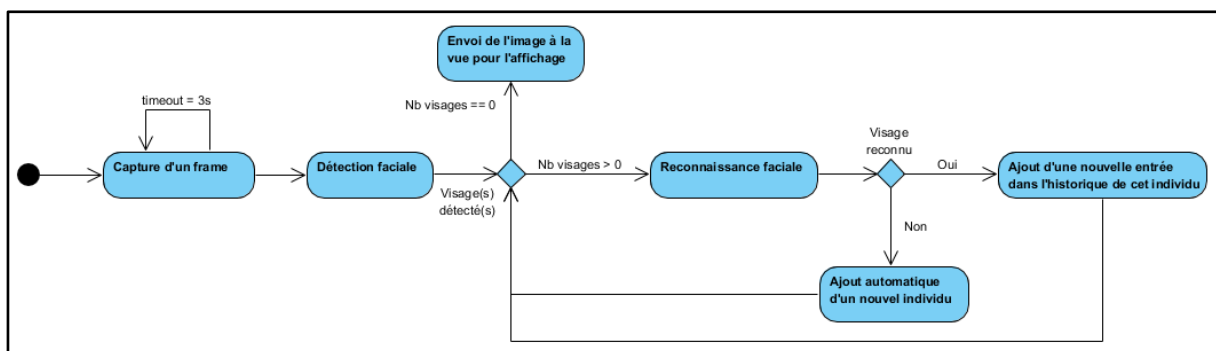


Figure 35 Diagramme d'activités

4.7 Proposition d'amélioration de la reconnaissance

Suite à un entretien avec l'expert Robert Van Kommer, un intérêt s'est manifesté autour d'une réflexion sur un éventuel algorithme pour améliorer le travail de reconnaissance.

En sachant à quel endroit se trouve quelle caméra, le système aurait la possibilité de calculer le temps parcouru pour une certaine distance par un individu. Il pourrait obtenir une vitesse sur le parcours et retourner la probabilité que l'individu soit le même sur les deux caméras. Si un individu est détecté par une caméra à un point X et détecté à nouveau 2 secondes plus tard à point Y distant de 100 mètres du point X, c'est très certainement pas la même personne. Seulement, si une caméra est ajoutée, les distances entre cette caméra et toutes les autres doivent être calculées à moins d'avoir un plan à l'échelle et de placer les caméras aux bons endroits. Dans tous les cas, les distances sont calculées en 2D et non en 3D. Cela implique qu'aucune relation n'existe entre deux caméras placées à deux étages différents. Ce "problème" de distance est similaire au travail de Martin Vetterli, un professeur de l'EPFL, qui a pris part à la conception avec LCAV (Laboratory for Audiovisual Communications) d'un système de calculs de distances euclidiennes entre plusieurs points à l'intérieur d'une boîte. Ce projet se nomme EDMBox (Euclidean Distance Matrices). Pour plus d'informations, le projet github est disponible à l'adresse suivante : <https://github.com/LCAV/edmbox>.

Si cette optimisation devait être réalisée, l'idée serait d'utiliser un plan d'architecte⁸ de l'établissement pour chaque étage, de permettre à l'utilisateur de sélectionner une distance pour représenter à choix 1 mètre, 10 mètres ou 100 mètres, et ensuite d'avoir la possibilité d'ajouter ou de supprimer une caméra sur le plan. Les distances seraient ensuite obtenues par des calculs. Le design de cette fonctionnalité ressemblerait à la Figure 36 ci-dessous.

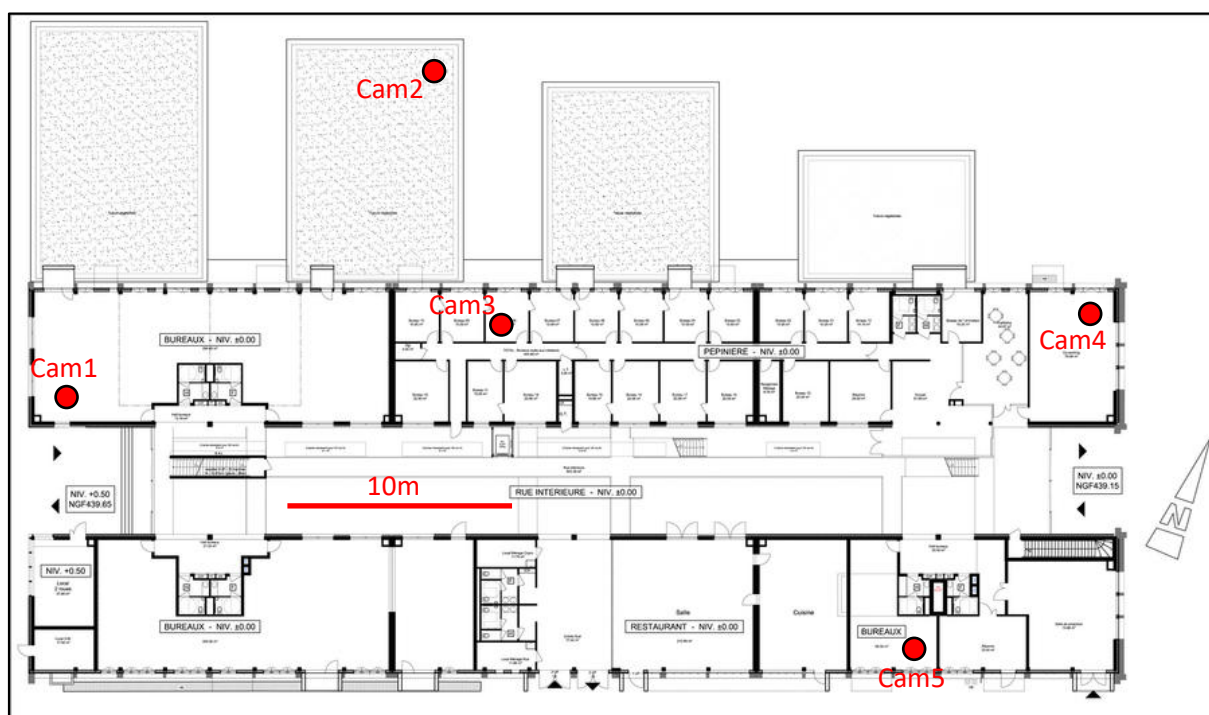


Figure 36 Proposition d'une interface pour le calcul de distance

⁸ http://www.ville-crangevrier.fr/var/ville_crangevrier/storage/images/mediatheque/les-papeteries/images/plan-rdc/370435-1-fre-FR/Plan-RDC_lightbox.jpg

4.8 Binding entre les différentes pages

Afin d'assurer la cohérence et la mise-à-jour des états des pages html (cameras, history et individuals), une solution disponible dans tous les navigateurs actuels a été grandement utile. Cette solution se nomme localStorage. Grâce à un système similaire au format json, qui est d'ailleurs utilisé en interne, on peut attribuer à une clé une nouvelle valeur. Ce changement de valeur est détectable au sein de tous les scripts JavaScript des pages et onglets ouverts dans un même navigateur (Chrome dans mon cas). Sur la Figure 37 ci-dessous, on apprend comment changer la valeur attribuée à une clé et sur la Figure 38 en-dessous, c'est le code qui permet de récupérer l'événement et d'appliquer un traitement en conséquence. Plus concrètement, le linkage est surtout sollicité lors de l'activation/désactivation du tracking, lors d'un changement des informations d'un individu traqué et lors du changement de nom d'une caméra.

```
if (isTrackingRequired) {  
    localStorage.setItem(targetIdentifier, 'merge_' + JSON.stringify(identifiers));  
}  
localStorage.setItem(targetIdentifier, $('#miniSide .miniPic').first().css('background-image').replace(/"/g, ""));
```

Figure 37 Utilisation de la méthode setItem de localStorage

```
$(window).bind('storage', function(e) {  
    if (e.originalEvent.key === 'camerasSelectionUpdate') {  
        ...  
    } else {  
        if (e.originalEvent.newValue !== null) {  
            ...  
        } else {  
            ...  
        }  
    }  
});
```

Figure 38 Utilisation d'un listener de changement de valeur dans localStorage

Il existe également une autre technologie de stockage dans le navigateur très proche de localStorage. Il s'agit de sessionStorage. Cette solution est légèrement différente de localStorage au niveau de la portée et de la durée de persistance des données. Comme son nom l'indique, les données stockées avec sessionStorage sont détruites une fois que la session est terminée. De plus, ce que contient sessionStorage n'est pas partagé avec les autres onglets [16].

4.9 Aperçus du prototype

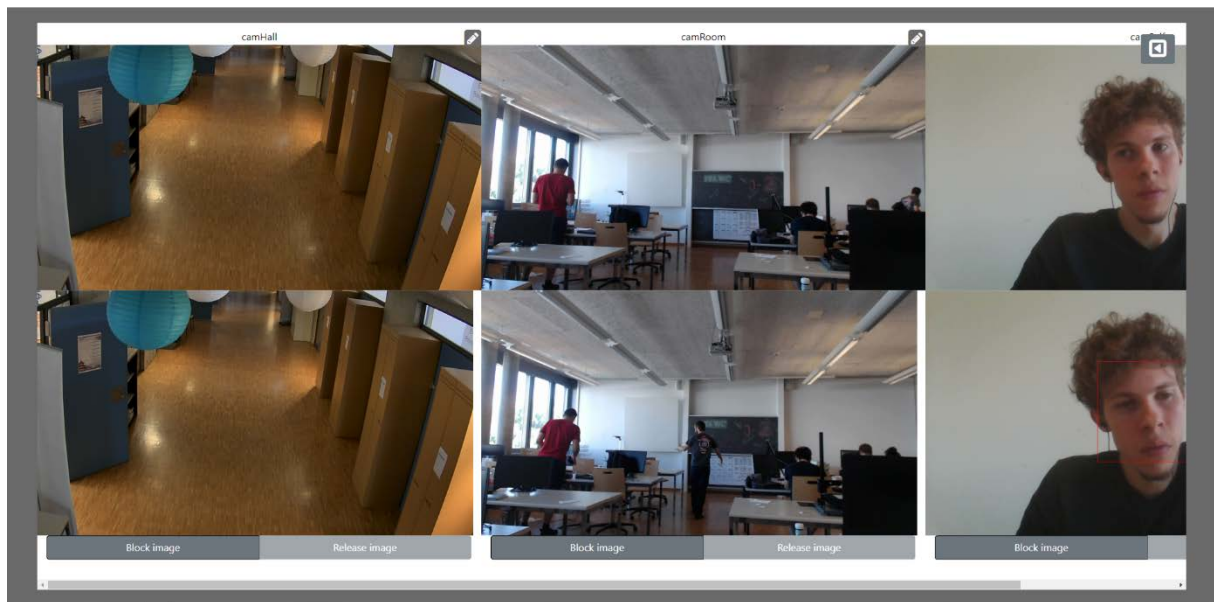


Figure 39 Cameras

La Figure 39 ci-dessus affiche la page cameras qui contient les flux vidéos des caméras.

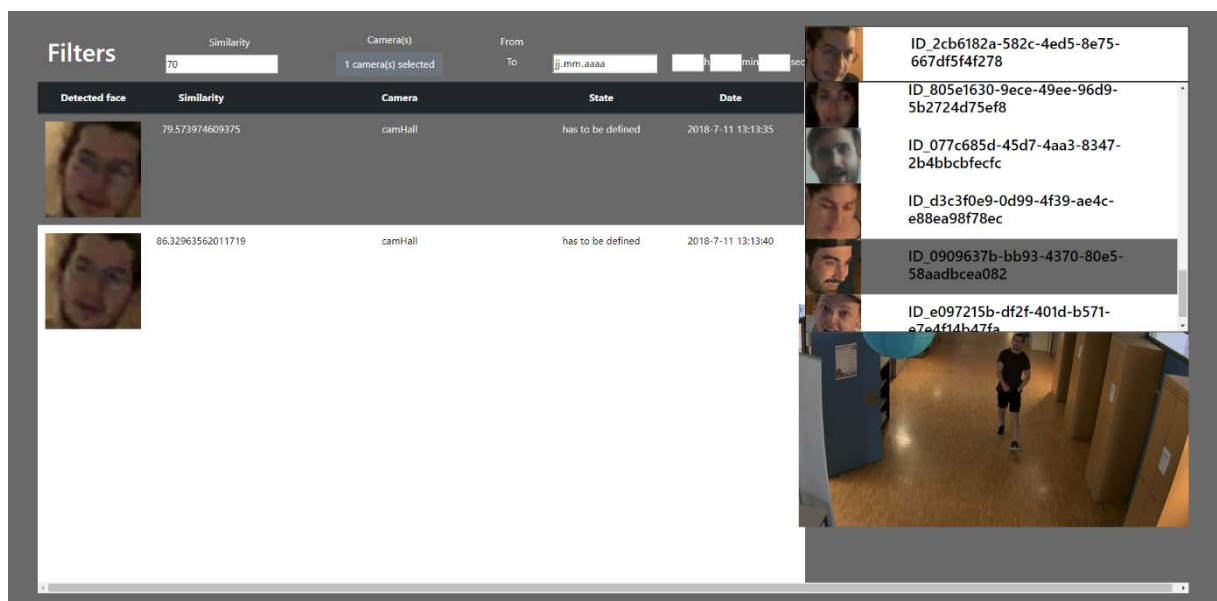


Figure 40 History - Sélection d'un individu pour l'affichage de son historique

Dans la Figure 40 ci-dessus, on peut voir les différents individus dont le tracking a été activé. Il n'est donc possible de visionner l'historique de quelqu'un que si son tracking est activé. Il serait préférable dans des améliorations futures de rendre indépendant l'affichage de l'historique de l'activation/désactivation du tracking.

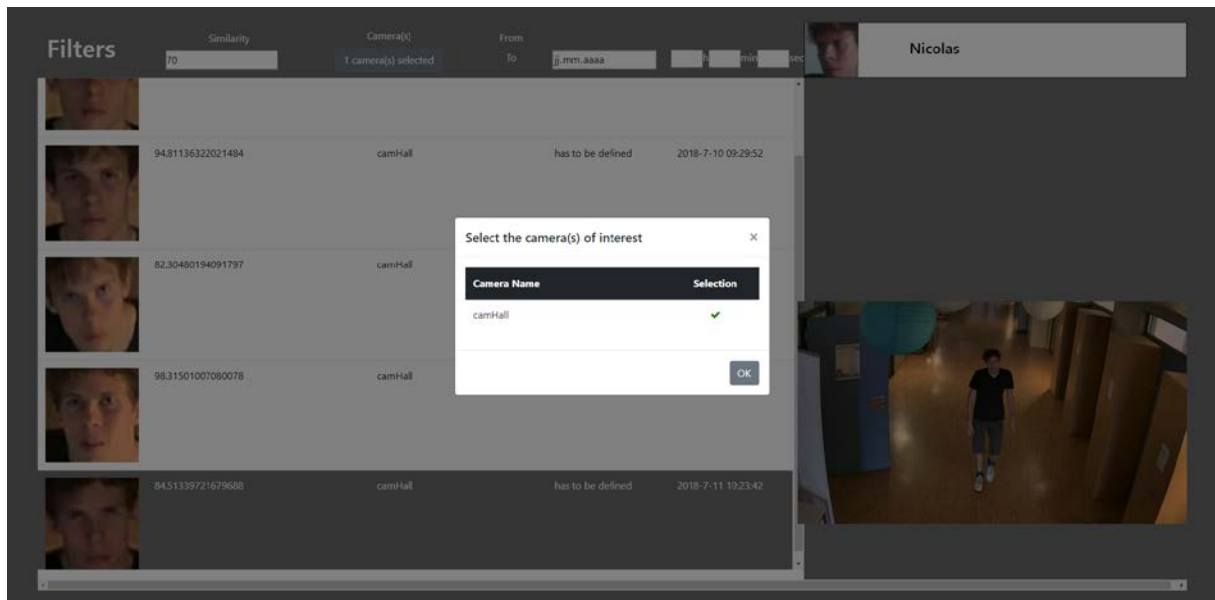


Figure 41 History - Sélection des caméras pour filtrer un historique

Dans l'illustration au-dessus, l'utilisateur peut sélectionner les caméras qui l'intéressent. La modification du nom d'une caméra est également mise à jour dans cette table de sélection. La cohérence est respectée.

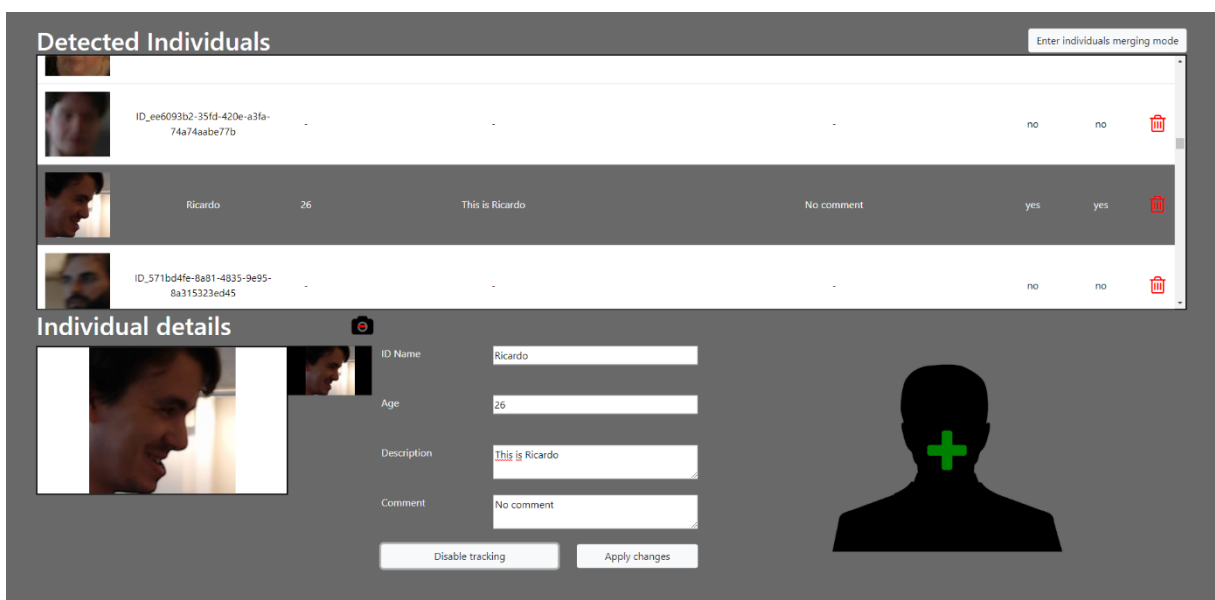


Figure 42 Individuals - Affichage d'un individu connu

Sur la Figure 42 ci-dessus, on aperçoit l'interface générale de la partie des individus. Lorsque l'on clique sur une entrée, toutes les images sont affichées en dessous ainsi que les informations associées à l'individu. L'âge est une donnée non dynamique, la date de naissance en revanche est une donnée dynamique. C'est un changement qui serait judicieux d'apporter au prototype.

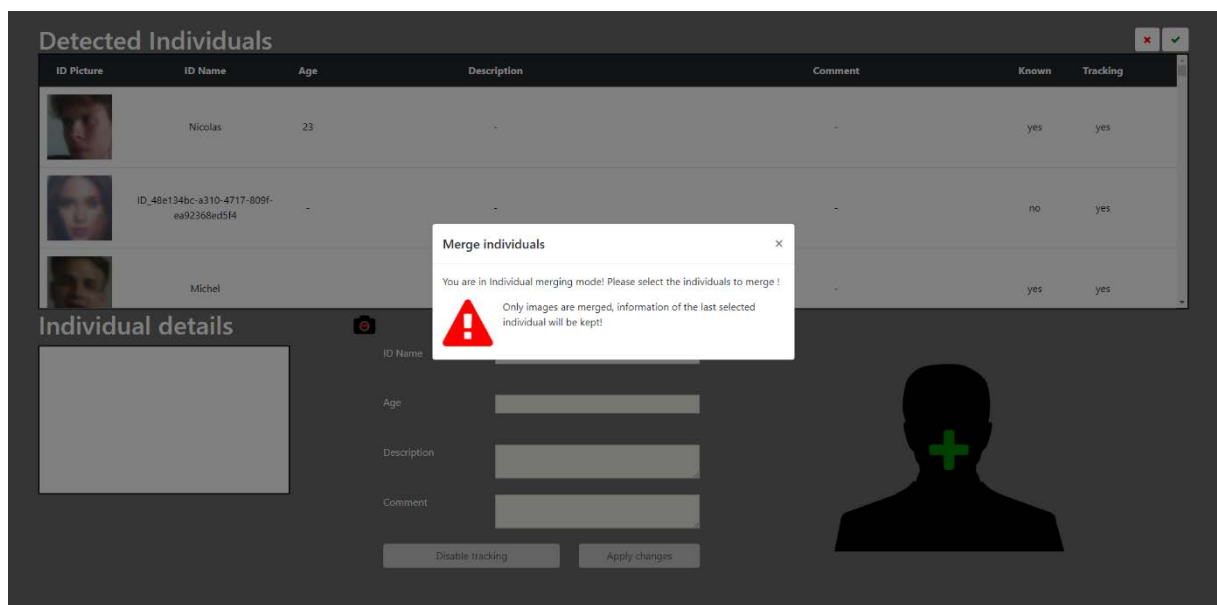


Figure 43 Individuals - Entrée dans le mode de fusion

Lorsque l'utilisateur entre dans le mode de fusion d'individus, il en est informé grâce à un modal (Figure 43). Un avertissement prévient que seules les images et les informations de la dernière personne sélectionnée sont gardées. Pour sortir de ce mode de fusion, il suffit de valider la fusion ou de l'annuler. Cette opération est applicable sur plus de deux individus.

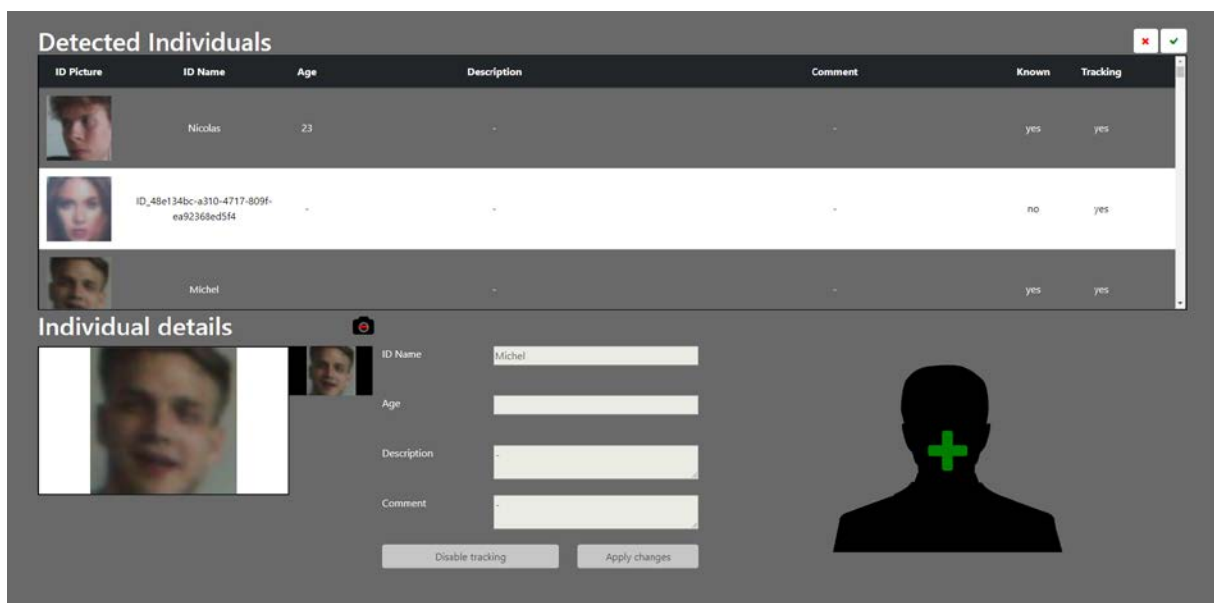


Figure 44 Individuals - Sélection de plusieurs individus pour la fusion

On peut observer dans l'illustration ci-dessus la possibilité de sélectionner plusieurs individus pour les fusionner. Pour rappel, ce sont les images de la dernière personne sélectionnée qui sont affichées ainsi que les informations textuelles.

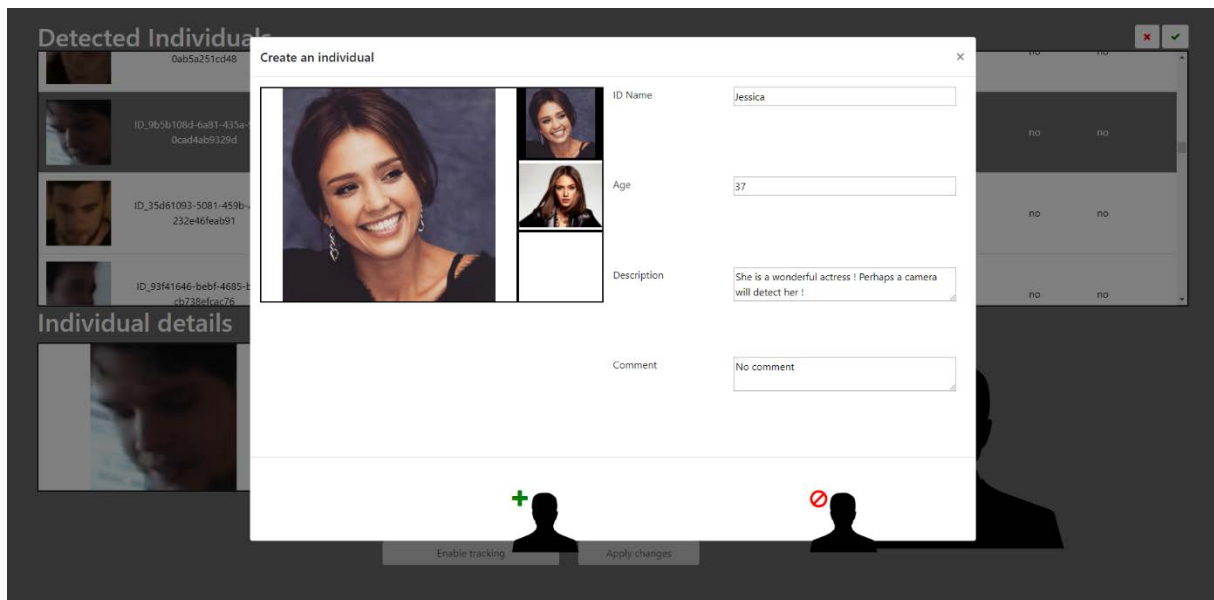


Figure 45 Individuals - Création d'un individu

La création manuelle d'un individu est similaire au traitement automatique d'ajout d'individu par le système, comme le démontre l'interface de la Figure 45 ci-dessus. L'utilisateur peut ajouter plusieurs images mais n'a pas la possibilité par contre de les supprimer. Il faut pour cela annuler l'ajout et recommencer la procédure. C'est une amélioration à apporter. Tant que l'identifiant ne commence pas par "ID_", l'individu est considéré comme connu. Le tracking est désactivé par défaut. C'est la première image qui fait office de photo de profil. Tant que cette photo n'est pas supprimée, la "photo d'identité" reste inchangée. Si l'utilisateur clique sur une photo miniature, celle-ci est affichée en grand.

Le prochain chapitre se consacre aux tests.

5 Tests

Les tests prennent part au projet lors de la conception d'un prototype avec des tests utilisateurs. C'est le cas avec les évaluations par exemple du client. Plus l'approche UCD (User Centered Design) est importante, plus les rencontres avec les intéressés sont régulières. Les tests sont également utiles pour mettre en évidence les limites du projet. D'autres encore prouvent que le prototype fonctionne de manière adéquate. Ce chapitre commence d'abord par les tests des conditions de simulation. Le thème abordé par la suite est le test du scénario de simulation, puis viennent les tests utilisateurs. Des tests de compatibilités ont été faits pour déterminer sur quel navigateur il est possible de faire tourner le prototype, suivis d'une liste des améliorations de conditions pour obtenir de meilleurs résultats. Enfin, une synthèse résume l'ensemble des tests de ce projet.

5.1 Tests des conditions de simulation

5.1.1 Luminosité

La variation de la lumière du jour a été simulé avec un changement de la luminosité de l'image. Le résultat est qu'un visage peut être détecté avec une luminosité entre -0.8 et 0.7 dans le cas d'une situation idéale (pas de flou et image très similaire).

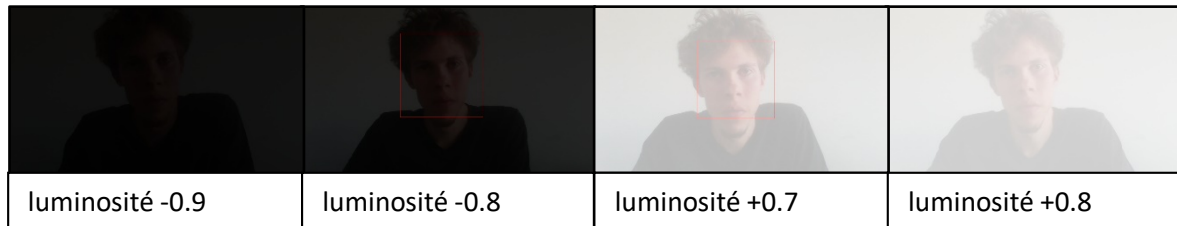


Figure 46 Tableau de détection sur luminosité variable

Pour la reconnaissance de visage, l'individu enregistré dans le système a approximativement la même image mais avec une luminosité à 0 (luminosité de base, normale). L'individu a pu être reconnu avec une luminosité entre -0.7 et 0.6 et un visage assez proche de la webcam (moins de 50 centimètres).

Une image est pire dans le cas d'une exposition brûlée ou à contre-jour car les traits du visage sont effacés contrairement au cas où la luminosité a été baissée dans sa globalité.

5.1.2 Nombre de personnes

Amazon Rekognition permet de détecter au maximum 100 visages sur une image. On voit sur la Figure 47 ci-dessous que tous les visages ont pu être détectés grâce au fait qu'ils étaient statiques. Dans la deuxième capture (Figure 48), les mêmes personnes sont en mouvement et donc le taux d'échec est bien plus élevé.

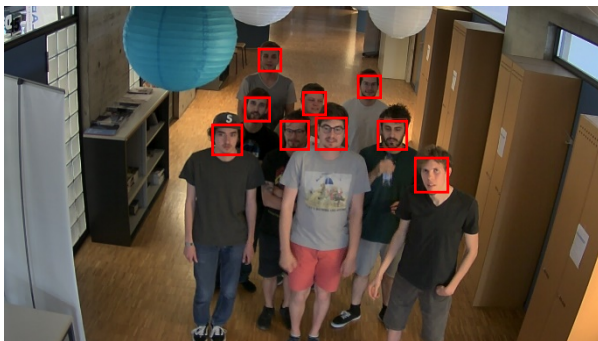


Figure 47 Foule statique

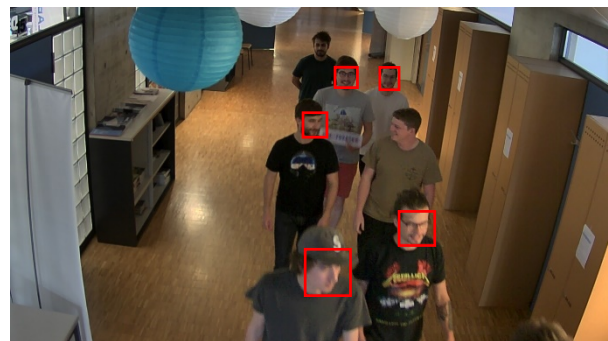


Figure 48 Foule en mouvement

5.1.3 Angles

Amazon Rekognition détecte des visages allant de face jusqu'au profil. Par contre, à partir d'une image de face, on peut lui attribuer des images avec un visage tourné jusqu'à 45 degrés. Si on veut avoir une chance de reconnaître un visage de profil, il est obligatoire d'avoir dans la collection d'Amazon à peu près le même visage de profil. Au niveau du tilt, la détection fonctionne dans des positions proches de 90 degrés.

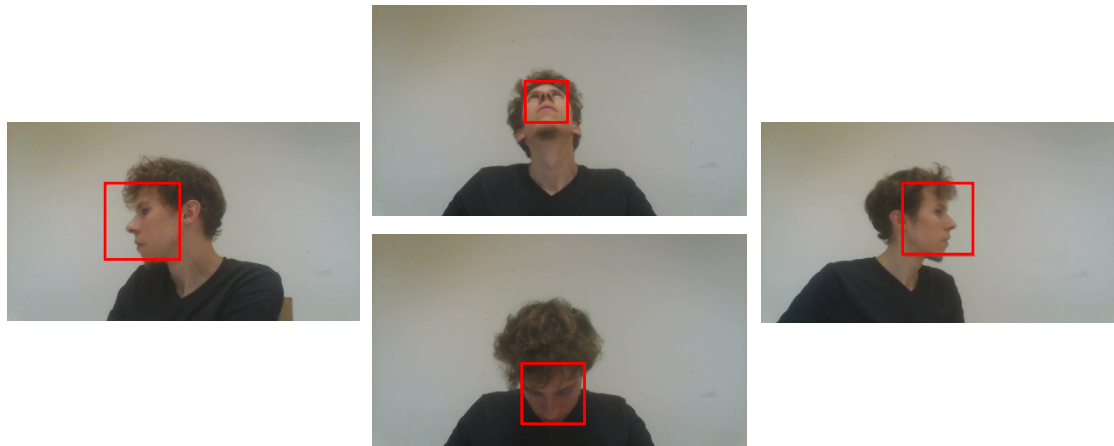


Figure 49 Tests des positions extrêmes du visage pour la détection faciale

5.2 Test du scénario de simulation

Le scénario de simulation est le suivant : Le suivi des individus au sein de la HEIA-FR se fait dans un couloir à l'intérieur du bâtiment D avec en plus une salle de classe dans laquelle se trouve le système (ordinateur portable). La caméra IP a été fixée au plafond au milieu du couloir en direction de la porte automatique. Les webcams sont obligatoirement branchées au port USB de mon ordinateur. La position de la webcam est placée au fond de la rangée centrale de la salle de classe sur un moniteur en direction du tableau noir.

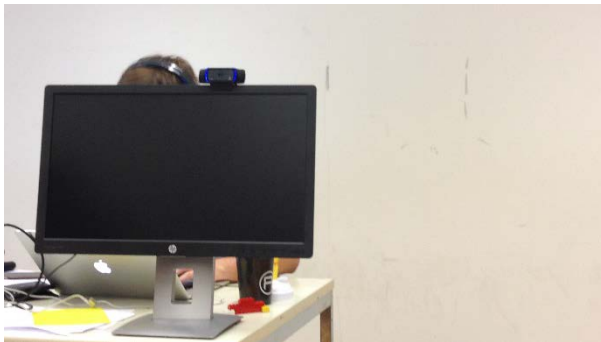


Figure 51 Webcam



Figure 50 Caméra IP

Avantages de la disposition des caméras

Elles sont placées devant un long couloir ce qui laisse le temps aux caméras de capturer les images. La capture est lancée toutes les 3 secondes. La caméra IP est placée dans un lieu avec beaucoup de trafic. Cela permet de tester dans un environnement réaliste le fonctionnement du prototype. Plus la caméra est haute, plus les personnes sont visibles et ne se cachent pas les unes les autres.

Désavantages de la disposition des caméras




La caméra IP est posée sous le plafond à une hauteur d'environ 3 mètres. Son champ de vision s'étend jusque devant la porte automatique. Puisque la caméra a une vision assez lointaine (une trentaine de mètres), lorsqu'une personne entre dans le champ depuis le fond, son visage est rapidement détecté. La taille du visage détecté validant la contrainte des 5% fixée par Amazon, elle n'est cependant pas très grande. Elle est estimée en moyenne entre 20 et 30 pixels. Le rectangle de détection est carré à un ou













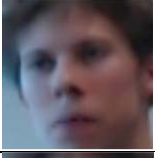







deux pixels près. Citée déjà précédemment dans ce document, la taille minimale d'une image en entrée de la reconnaissance faciale est de 80 pixels x 80 pixels. La taille du visage doit alors être agrandie pour respecter la contrainte ce qui provoque une pixellisation. Cette mesure a été prise pour garantir qu'un individu ne passe pas inaperçu lors d'un passage devant la caméra. Dans le cas où une personne court, aucun résultat n'est garanti en raison du temps des 3 secondes d'intervalle de capture d'images ainsi que le flou du visage provoqué par la course.

Le premier test est le suivant : je suis seul et je pars depuis la porte automatique pour aller jusqu'à mon bureau. La démarche n'est ni rapide ni lente. Aucune lunettes ni chapeau ne sont portés. Cette séquence est reproduite cinq fois pour ressortir les résultats sous la forme d'un tableau inclus dans la page suivante.



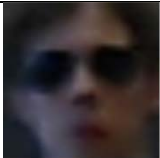

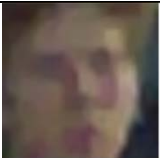

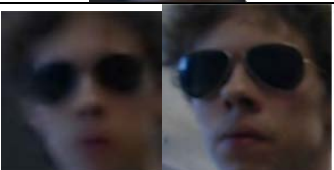

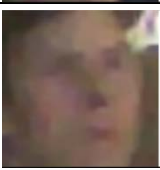

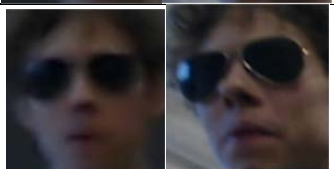



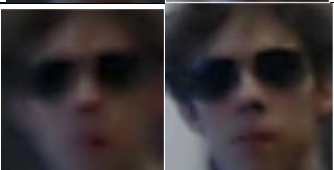

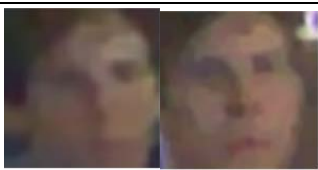



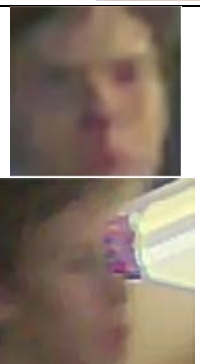


Cette image ci-contre est celle de l'individu Nicolas que l'on essaie d'identifier avec les images capturées par les caméras.

-  individu Nicolas identifié
-  aucun visage détecté / individu non reconnu
-  autre individu identifié avec numéro de l'essai de l'image

N° d'essai	Logitech HD PRO C920		AXIS Q1604	
	Visage	Reconnu	Visage	Reconnu
1				
2				
3				
4				
5				

Les résultats dans le tableau ci-dessus nous montre très clairement que la webcam a un taux de réussite de 100% si l'objectif est de capturer l'individu au moins une fois. En général, plus l'image est floue, plus elle a été prise de loin. Concernant la caméra AXIS, la position de cette caméra n'est pas optimale. Un visage a été détecté mais n'a pas été reconnu par le système au deuxième essai. C'est seulement au quatrième essai en regardant l'objectif que la caméra a pu me reconnaître.

Dans le second test, j'ai essayé de supprimer les erreurs survenues lors du premier test en modifiant la position de la caméra IP et en portant des lunettes de soleil devant la webcam uniquement pour augmenter la difficulté. Les déplacements sont les mêmes que dans le premier test.

N° d'essai	Logitech HD PRO C920		AXIS Q1604	
	Visage	Reconnu	Visage	Reconnu
1				
2		 1		
3		 1		
4				
5				 

Les résultats sont différents de ceux du test précédent. On voit au premier essai qu'aucune des deux images n'est reconnue. Le tracking a été activé sur ces deux "nouveaux" individus. On voit par la suite que pour la webcam dans les essais n° 2 et 3, les quatre images ont reconnu l'individu détecté dans le premier essai et non l'individu Nicolas de base qui est l'identification espérée. Du côté de la caméra IP, aucune amélioration n'est perçue. Le résultat du 3^{ème} essai est pire que les deux premiers. Après l'essai n° 3, une fusion des individus a été exécutée. On remarque alors que les résultats obtenus sont loin



Figure 52 Nouvelle position de la caméra IP

d'être parfaits mais nettement meilleurs que ceux avant la fusion. La luminosité du corridor était faible à ce moment-là.

5.3 Tests utilisateurs

Les tests utilisateurs sont très importants dans le sens où lorsque l'on développe un produit depuis un certain temps, il est parfois difficile de prendre du recul et garder un regard externe le plus objectif possible. C'est pourquoi des tests utilisateurs ont été menés dans le cadre de ce projet.

5.3.1 Tests externes

Les tests externes concernent, dans le cadre de ce projet, les évaluations sur des maquettes par des personnes externes au domaine de l'informatique. Ils ont été effectués par trois personnes différentes de mon entourage proche. Les remarques et conseils de personnes non-impliquées et qui ne connaissent pas l'informatique nous aident à avoir un regard neutre sur la situation. Chaque avis est à prendre en compte.

5.3.2 Test interne

Selon la demande de l'expert, monsieur Robert Van Kommer, un entretien de dix minutes avec le concierge de la Haute Ecole d'ingénierie et d'architecture de Fribourg, monsieur Olivier Monney, a pu être organisé. Le but de cet entretien était de découvrir son avis sur l'utilité et l'utilisabilité du prototype développé et de son contexte.

Les points principaux suivants en sont ressortis :

- Interface cohérente et visiblement fonctionnelle
- Les contraintes des angles et la luminosité doivent être prises en compte
- Besoin de tester le fonctionnement du système avec plus de 2 ou 3 personnes
- Eventuellement aborder le sujet de l'extension d'une telle application avec une base de données au sein d'une entreprise
- Le sujet de la législation a été abordé

Aucune remarque venant de sa part concernant un manque dans les fonctionnalités implémentées ou le design n'a été faite. Globalement, l'impression de monsieur Olivier Monney a été positive.

5.4 Tests de compatibilité

Le système d'exploitation utilisé pour ce projet est un Windows 10. Les autres systèmes d'exploitations n'ont pas été testés en raison de la contrainte du temps.

Le navigateur utilisé pour la réalisation du projet est google chrome. Les navigateurs firefox et edge ont été testés et les résultats sont les suivants :



Figure 53 Compatibilité avec les navigateurs firefox et edge

5.5 Conditions d'amélioration

Dans le but d'améliorer les résultats obtenus par le système, un problème récurrent important à relever est la différence de taille entre un visage détecté et la taille minimale d'un visage pour demander une reconnaissance faciale. Tel qu'on peut le voir dans la Figure 54 ci-dessous, un visage n'a



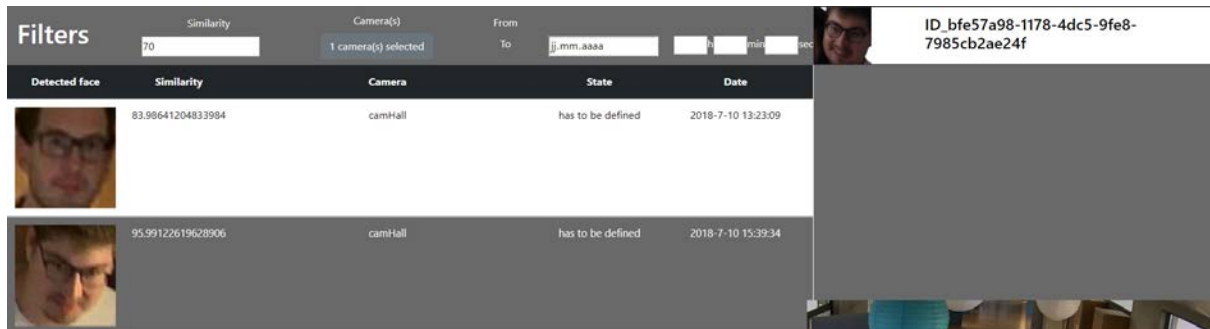
Figure 54 Petite taille de visage requise pour une détection faciale

pas besoin d'être très grand pour être détecté. Dans le prototype actuel, le problème est résolu par un agrandissement de l'image contenant le visage détecté. Cette opération provoque alors une pixellisation du visage, ce qui réduit le taux de succès de reconnaissance faciale sur cette image. Une première solution pourrait être la suivante : si la taille du visage détecté est en-dessous d'un certain seuil comme par exemple les $\frac{3}{4}$ de 80 pixels, le résultat de la détection n'est pas traité par la suite de l'algorithme. Une telle solution signifie une baisse dans le nombre de capture d'individus car comme on n'a pas pu l'observer avec les tests du scénario, il n'est pas rare que la caméra IP soit capable de ne détecter qu'un seul visage sur l'espace des 3 secondes d'intervalle entre deux captures. Si l'intervalle de capture d'images peut être réduit et que les vidéos restent fluides, cette solution pourrait être envisageable.

Une autre amélioration possible, qui était d'ailleurs un des objectifs secondaires, est d'optimiser le nombre d'appels à l'API pour la détection de visage. Pour faire cela, il suffirait de soustraire l'image capturée avec la précédente et de calculer la somme résiduelle des carrés afin d'obtenir une différence quantifiable. Si la SSE (Sum of Squared Errors) ne dépasse pas un certain seuil, on considère qu'aucun changement n'est intervenu dans l'image et donc qu'aucun déplacement n'a été signalé. Il n'est donc pas nécessaire de lancer une détection de visages sur une image dont on sait qu'elle ne contient aucun visage.

Le threshold (seuil) demandé par Amazon pour retourner un résultat de reconnaissance faciale pourrait être augmenté. De la même façon que la première solution, un compromis doit être fait. Si le threshold est réhaussé, les résultats corrects avec une similarité plus basse due à la mauvaise qualité par exemple ne seront pas pris en compte. Le challenge réside dans le tuning de ces paramètres. On

remarque dans la Figure 55 ci-dessous que la similarité du faux-positif est déjà très haute. Il semble donc qu'il est difficile de réhausser encore le seuil de similarité sans perdre de correspondances pourtant correctes.



The screenshot shows a web application interface for facial recognition. At the top, there are filters for 'Similarity' (set to 70), 'Camera(s)' (1 camera selected), and 'From' (jj.mm.aaaa). A user profile is visible on the right with ID 'bfe57a98-1178-4dc5-9fe8-7985cb2ae24f'. Below the filters is a table with the following data:



Detected face	Similarity	Camera	State	Date
	83.98641204833984	camHall	has to be defined	2018-7-10 13:23:09
	95.99122619628906	camHall	has to be defined	2018-7-10 15:39:34

Figure 55 La similarité entre Guillaume et Jérôme est déjà très haute.

5.6 Synthèse

Les tests permettent d'évaluer efficacement la valeur d'un prototype ou d'une application. Ceux-ci doivent être menés tout au long du projet et pas seulement à la fin. Des tests des conditions de simulation, de scénario et des tests utilisateurs ont été utiles dans le cadre de ce projet. Il existe encore une quantité impressionnante d'autres types de tests qui sont tout aussi intéressants.

6 Conclusion

Dans cette partie, les résultats obtenus à la fin de ce projet vont être discutés. Les problèmes rencontrés vont être exposés. Une synthèse de ce travail est également présente puis ce chapitre se termine par une ouverture sur de potentielles améliorations.

6.1 Discussions des résultats

Le prototype développé a rempli les objectifs fixés par le projet. Par contre, il ne fonctionne pas de manière optimale et comprend encore plusieurs bugs et incohérences notamment avec la persistance et la restauration de l'état du programme après son arrêt lors de son redémarrage, ainsi que la concurrence.

6.2 Problèmes rencontrés

La section des problèmes rencontrés s'étend sur des sujets allant du concept abstrait au concret. Elle comprend les problèmes rencontrés ainsi que les difficultés éprouvées.

Affichage des caméras

- Pour les webcams USB, paramétrisation et utilisation d'openCV (le module opencv disponible via npm fait le binding entre opencv installé au préalable et nodejs)

Tests du prototype du projet de semestre

- Opencv capture vidéo webcam, mauvais numéro (1 = faux, 0 = juste) -> changer le numéro
- Dossiers "working" et "temp" non-existants -> création de ces dossiers
- Identifiant pour ajouter un visage sans espace

- Analyse de vidéo enregistrée impossible sur Windows -> utiliser FFmpegFrameGrabber

Concurrence et délai de traitements

- lors de l'ajout d'une webcam, au fur et à mesure que le prototype continue son exécution, les frames capturées par la webcam deviennent de moins en moins fluides.
-> le module threads.js, des workers ainsi que le tuning de l'interval de capture pour obtenir un effet vidéo ont été testés sans pour autant réussir à résoudre le problème.

Format des images

- Le fait de travailler avec plusieurs librairies traitant des images dans différents formats n'a pas été simple.
-> Il a fallu soit trouver des méthodes pour les convertir directement, soit écrire l'image dans un fichier et rouvrir ce même fichier avec l'autre librairie.

6.3 Synthèse

J'ai beaucoup aimé travailler sur ce travail de Bachelor. La durée de ce travail est passée très rapidement. Si une suite était envisagée, je continuerais volontiers de travailler dans ce domaine très actuel.

6.4 Possibilité future

Dans les perspectives proches, il s'agirait dans un premier temps de résoudre les bugs présents dans le prototype actuel. Ensuite, il serait judicieux de transformer les fichiers de persistance en base de données pour une meilleure efficacité. Les objectifs secondaires auraient aussi un intérêt à être atteints. Puis une extension avec un système d'alertes et de notifications aurait du sens.

7 Déclaration d'honneur

Je, soussigné, Nicolas Fuchs, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

8 Licences et versions de logiciels / bibliothèques



Visual Paradigm Version 15.0
Standard Academic License



WebStorm 2017.2.6
Licensed to Nicolas



OpenCV
3-clause BSD License



Node Package Manager v3.10.10
No License



Node v6.11.4
No License



aws-sdk v2.263.1
License Apache-2.0

Interface v1.2.1
License MIT

mjpeg-camera v1.7.2
License MIT

morgan v1.9.0
License MIT

opencv2 v5.0.1
License MIT

serve-favicon
License MIT

socket.io v2.1.1
License MIT

uuid v3.3.2

9 Bibliographie

- [1] «Facebook Research,» Deepface: Closing the Gap To Human-level Performance in Face Verification, [En ligne]. Available: <https://research.fb.com/publications/deepface-closing-the-gap-to-human-level-performance-in-face-verification/>. [Accès le 30 mai 2018].
- [2] R. Brandom, «The Verge,» Facial Recognition Is Coming To Us Airports, Fast-tracked By Trump, 18 avril 2017. [En ligne]. Available: <https://www.theverge.com/2017/4/18/15332742/us-border-biometric-exit-facial-recognition-scanning-homeland-security>. [Accès le 30 mai 2018].
- [3] S. Biget, «Reconnaissance Faciale : 1 Seconde Pour Repérer Un Visage Parmi 36 Millions,» 30 mai 2018. [En ligne]. Available: <https://www.futura-sciences.com/tech/actualites/technologie-reconnaissance-faciale-1-seconde-reperer-visage-parmi-36-millions-38662/>.
- [4] R. Brandom, «The Verge,» Orlando Police Scramble To Defend Amazon Facial Recognition Pilot, 24 mai 2018. [En ligne]. Available: <https://www.theverge.com/2018/5/24/17391632/amazon-facial-recognition-orlando-police-rekognition>. [Accès le 30 mai 2018].
- [5] «Face Recognition,» [En ligne]. Available: https://www.nec.com/en/global/solutions/safety/face_recognition/index.html. [Accès le 04 06 2018].
- [6] «Gemalto,» Video-based Facial Recognition - Gemalto Cogent Live Face Identification System, [En ligne]. Available: <https://www.gemalto.com/govt/biometrics/biometric-software/live-face-identification-system>. [Accès le 6 juin 2018].
- [7] [En ligne]. Available: https://fr.business.panasonic.ch/solutions-de-securite/sites/default/files/security-solutions/specsheet_uploads/WV-ASM200_Fr.pdf. [Accès le 5 juin 2018].
- [8] M. O.-J. Thornton, «The most popular HTML, CSS, and JS library in the world.,» [En ligne]. Available: <https://getbootstrap.com/>. [Accès le 6 juin 2018].
- [9] «Documentation - Materialize,» [En ligne]. Available: <https://materializecss.com/>. [Accès le 6 juin 2018].
- [10] «The most advanced responsive front-end framework in the world. |,» [En ligne]. Available: <https://foundation.zurb.com/>. [Accès le 6 juin 2018].
- [11] «Python.org,» [En ligne]. Available: <https://www.python.org/>. [Accès le 6 mai 2018].
- [12] «Node.js,» [En ligne]. Available: <https://nodejs.org/en/>. [Accès le 6 juin 2018].
- [13] [En ligne]. Available: <http://www.cplusplus.com/>. [Accès le 6 juin 2018].
- [14] «Ecmascript 6: New Features: Overview and Comparison,» [En ligne]. Available: <http://es6-features.org/#ClassDefinition>. [Accès le 2 juillet 2018].

- [15] «GitHub,» 27 décembre 2017. [En ligne]. Available: <https://github.com/drejkim/face-detection-node-opencv>. [Accès le 10 juin 2018].
- [16] «Amazon Rekognition – Questions Fréquentes – Aws,» [En ligne]. Available: <https://aws.amazon.com/fr/rekognition/faqs/>. [Accès le 5 juillet 2018].
- [17] «Html5 Local Storage Vs. Session Storage,» [En ligne]. Available: <https://stackoverflow.com/questions/5523140/html5-local-storage-vs-session-storage>. [Accès le 10 07 2018].

10 Remerciements

J'exprime tous mes remerciements aux personnes qui m'ont soutenu et aidé pendant la réalisation de ce travail de Bachelor, spécialement mes superviseurs, Elena Mugellini, Omar Abou Khaled, Julien Tscherrig, mon expert, Robert Van Kommer, ma famille et mes collègues qui ont de près ou de loin participé à mon projet.

11 Annexes

En annexes, on trouve les liens d'accès aux plannings initial et final ainsi que le lien d'accès au cahier des charges. Le manuel d'utilisation explique comment faire fonctionner le programme. La version 2 des wireframes est accessible, suivie des données de compte Amazon pour une éventuelle reprise de projet.

11.1 Planning

Planning initial : https://gitlab.forge.hefr.ch/nicolas.fuchs/HumanRec/blob/master/Cahier_des_charges/Planning_initial.pdf

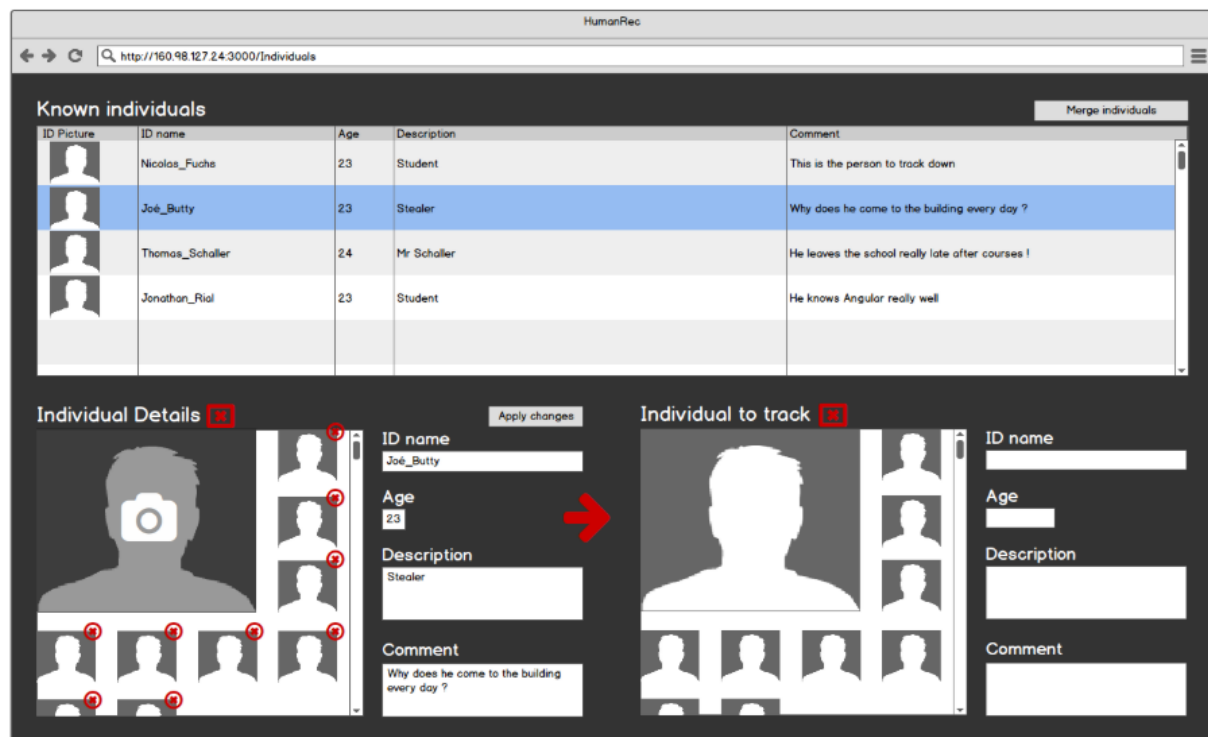
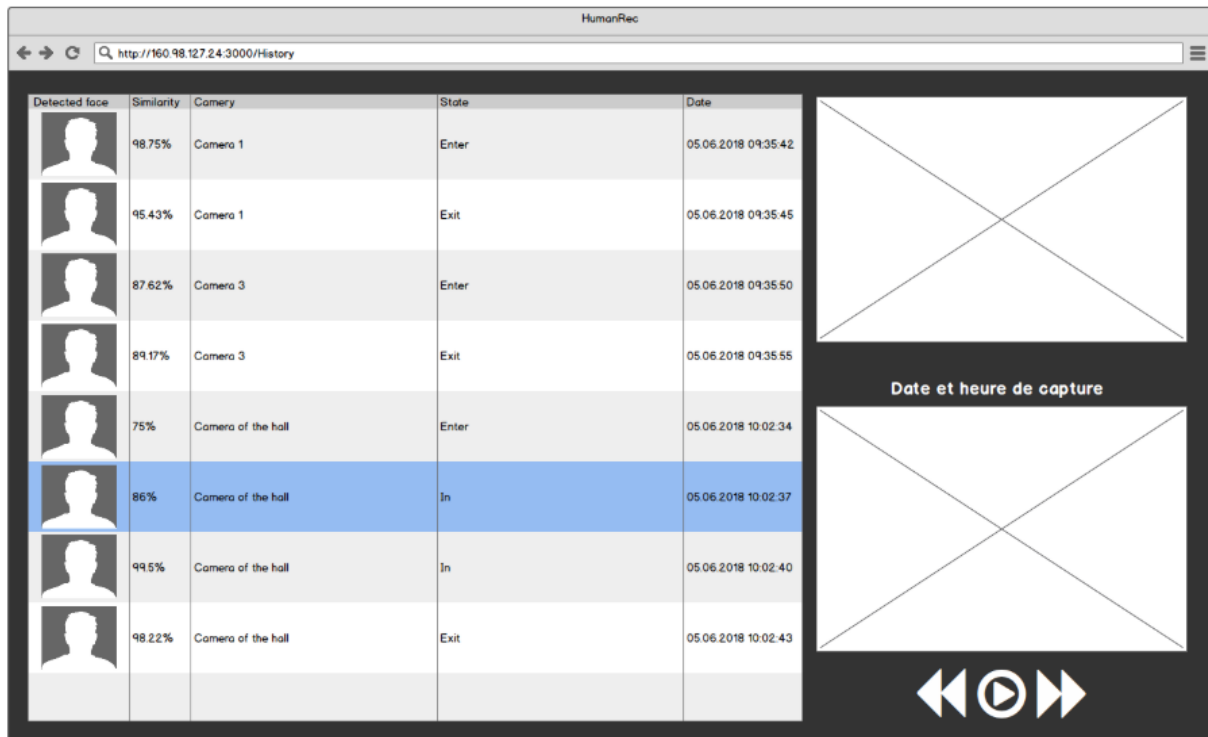
Planning final : https://gitlab.forge.hefr.ch/nicolas.fuchs/HumanRec/blob/master/Cahier_des_charges/Planning_final.pdf

11.2 Cahier des charges

https://gitlab.forge.hefr.ch/nicolas.fuchs/HumanRec/blob/master/Cahier_des_charges/Cahier_des_charges.pdf

11.3 Autres versions des maquettes

Version 2



Les maquettes ci-dessus et sur la page précédente sont celles pour lesquelles il y a une différence avec la dernière version.

11.4 Données de compte Amazon

Compte AWS

adresse e-mail : nicolas.fuchs@edu.hefr.ch
mot de passe : Humanrec2018
nom de compte AWS : NicoHumanRec
type de compte : professionnel
Nom Complet : NicoHumanRec
Nom de l'entreprise : HEIA-FR
Numéro de téléphone : 077 472 11 29
Pays/Région : Suisse
Adresse : Boulevard de Pérolles 80, Bâtiment D
Ville : Fribourg
Etat/Province ou région : Fribourg
Code Postal : 1700
Plan de support : gratuit
Carte de crédit utilisée : *****

Compte IAM

Utilisateur IAM : HumanRecUser
ID de clé d'accès : AKIAJPOUGNYW2ECI3FVA
Clé d'accès secrète : qJhnHJe0SjnMUCvUYKu531/sSOSoGeaTjuEmUF+l

Ses données sont placées dans le fichier Settings.js, là où les paramètres se trouvent.