

MaxAirain – H3HITEMA 2023

**Documentation technique**

**Informations**

|  |  |
| --- | --- |
| **Nom du projet** | MaxAirain |
| **Type de document** | Documentation technique |
| **Date** | 28/06/2023 |
| **Auteurs** | |  | | --- | | GOUJON Cédric | | GROETSCHEL Jonas | | DI PISA Luca | | PIOTROWSKI Thibault | |

**Table des matières**

[**1-** **Résumé du document** 3](#_Toc138941496)

[**2-** **Rappel sur le fonctionnement de l’application** 3](#_Toc138941497)

[2.1- Rappel sur le fonctionnement de l’application 3](#_Toc138941498)

[2.2- Décomposition du projet 3](#_Toc138941499)

[2.3- Architecture globale 4](#_Toc138941500)

[**3-** **L’application MaxAirain** 5](#_Toc138941501)

[3.1- Architecture 5](#_Toc138941502)

[3.2- Technologies utilisées 6](#_Toc138941503)

[3.3- Diagramme des classes 6](#_Toc138941504)

[3.4- Modèle des données 7](#_Toc138941505)

[**4.** **Charte graphique** 8](#_Toc138941506)

[4.1- Couleurs 8](#_Toc138941507)

[**5.** **Serveur** 9](#_Toc138941508)

[5.1- API 9](#_Toc138941509)

[**6.** **Bugs rencontrés** 10](#_Toc138941510)

[**7.** **Améliorations possibles** 10](#_Toc138941511)

[**5.** **Annexes** 11](#_Toc138941512)

[5.1- Table des illustrations 11](#_Toc138941513)

# **Résumé du document**

Ce document est la documentation technique officielle de la suite applicative MaxAirain. Il est divisé en trois parties :

* La documentation technique du client : application fonctionnant sur Web, Android et iOS
* La documentation technique du serveur : APIs
* Les bugs connus au sein de la suite logicielle

# **Rappel sur le fonctionnement de l’application**

## 2.1- Rappel sur le fonctionnement de l’application

L'application MaxAirain développée dans le cadre de ce projet vise à fournir une solution intelligente pour vérifier l'accès à un local sécurisé en utilisant la reconnaissance faciale. Elle permet de prendre une photo de l'utilisateur via une webcam, puis de comparer cette photo avec une base de données d'images d'identité préalablement enregistrées sur un serveur distant.

L'application assure l'acquisition d'image à partir de la webcam de l'utilisateur, la comparaison de cette image avec les images enregistrées sur le serveur distant, ainsi que l'affichage d'une interface utilisateur conviviale. Cette interface permet à l'utilisateur de valider ou de refuser l'accès au local sécurisé en fonction du résultat de la comparaison. En cas d'accès autorisé, un écran supplémentaire s'affiche, permettant au responsable de mission de sélectionner le matériel utilisé et rendu par l'agent de sécurité lors de son intervention. Ces informations sont ensuite enregistrées dans la base de données pour une traçabilité et une gestion efficace.

## 2.2- Décomposition du projet

Notre projet se décompose en différentes parties :

• Une application Web, qui est une application bureau fonctionnant sur Windows, Linux et MacOS, permettant de gérer la distribution et la réception du matériel

• La même application responsive fonctionnant sur Android et iOS

• Une base de donnée contenant toutes les informations nécessaires au fonctionnement du projet

Dans la suite de ce document, chacune de ces différentes parties sera développée.

## 2.3- Architecture globale

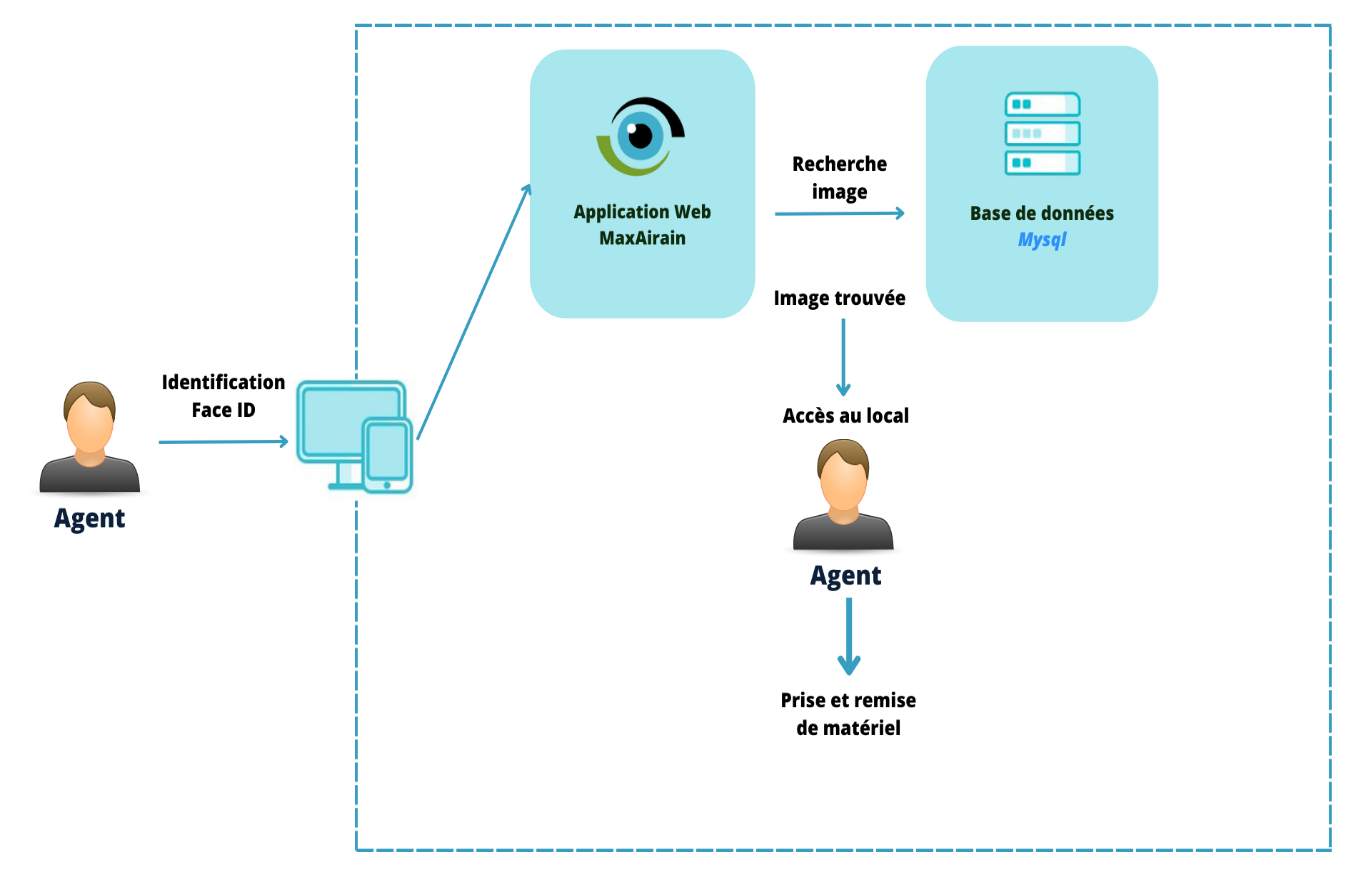
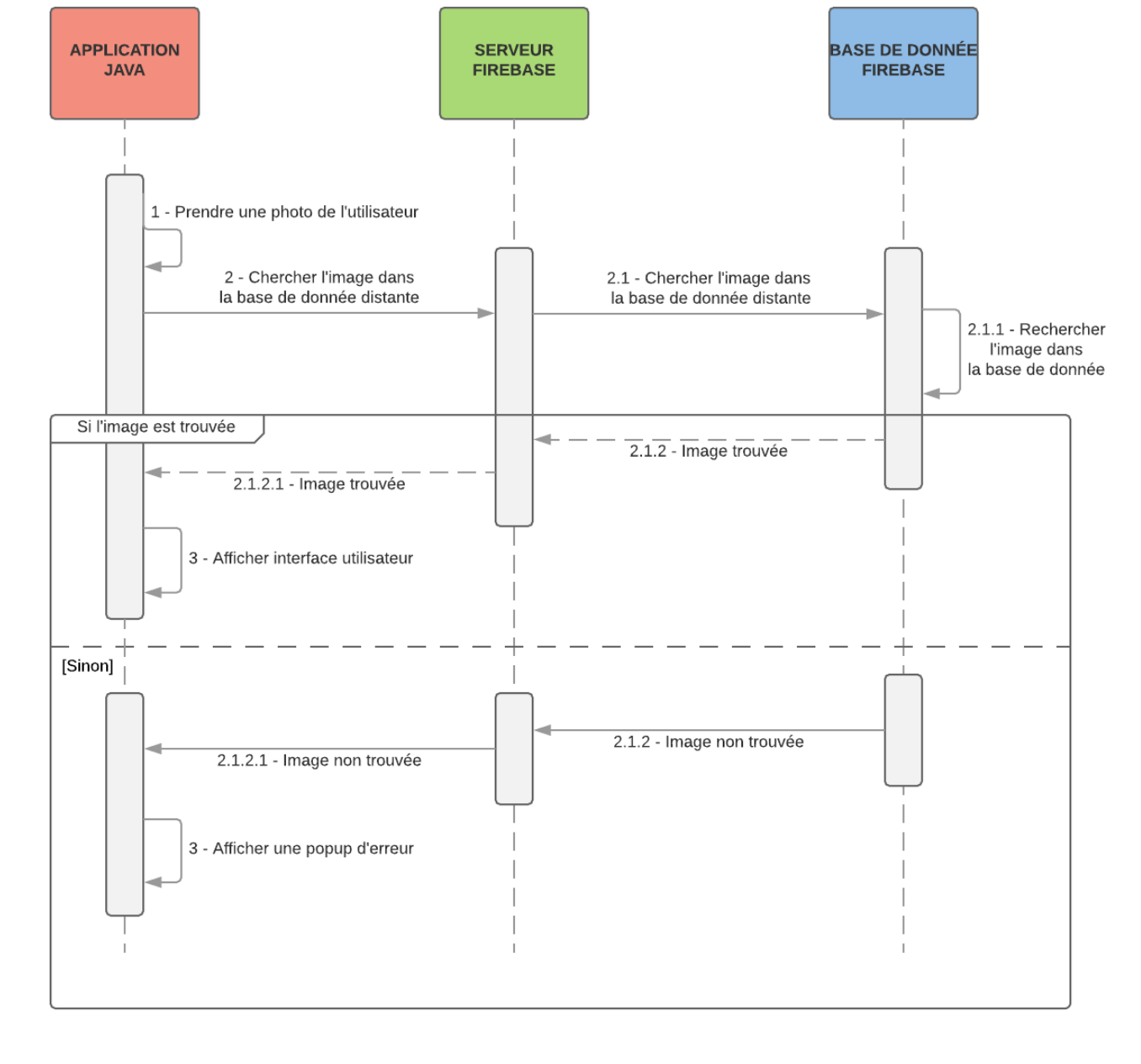


Illustration 1 : Architecture globale

# **L’application MaxAirain**

## 3.1- Architecture



**Base de Données**

**Serveur**

Ce diagramme de séquence représente l’enchainement des actions et des interactions entre l’utilisateur, l’application et le serveur. Il ne représente que la partie identification qui permet ensuite de procéder :

1/ À la prise de matériel disponible, pour la mission

2/ À la remise de matériel, après la mission

**Application**

Illustration 2 : Diagramme de séquences

## 3.2- Technologies utilisées

L’application est codé en Symfony et utilise API Platform pour faciliter l’accès aux données ainsi qu’un algorithme Python et ses bibliothèques pour la reconnaissance faciale.

La version mobile correspond à la même application web responsive, adaptée pour une utilisation sur les appareils mobiles.

**API Platform** : Framework open source qui facilite la création d’API Rest et l’accès aux données. Il fournit une infrastructure complète pour développer, documenter, sécuriser et tester des API. API Platform est basé sur des technologies modernes telles que Symfony, PHP, Doctrine

La technologie utilisés liés à la base de données est : **Mysql**.  
La base de données MySQL est utilisée pour stocker les données de l'application. Elle est utilisée pour stocker les informations relatives aux utilisateurs, aux réservations, aux outils, etc. Les entités de l'application sont mappées sur des tables MySQL à l'aide de **l'ORM** (Object-Relational Mapping) **Doctrine**, qui est intégré à Symfony.

L'application accède à la base de données via API Platform, qui fournit une interface simple.

## 3.3- Diagramme des classes

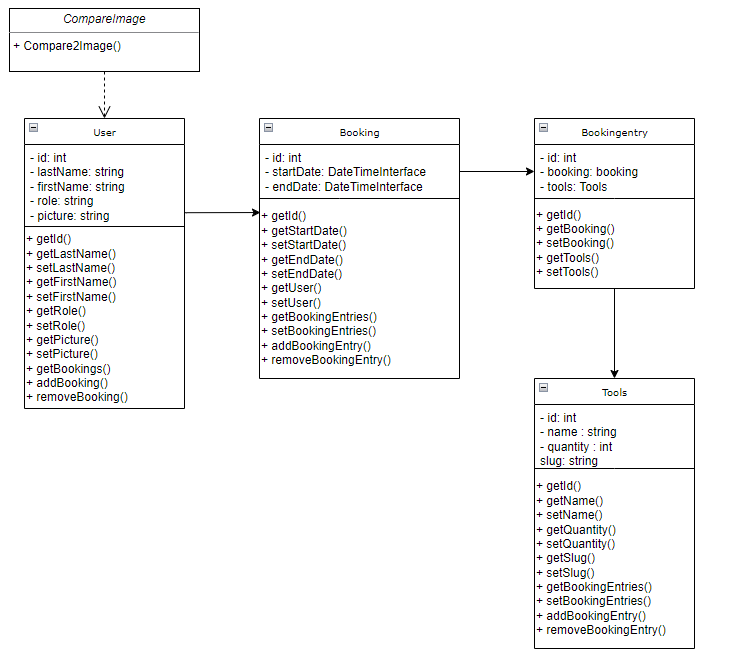


Illustration 3 : Diagramme des classes

« **CompareImage** » est la classe principale qui représente le service de comparaison d’images (authentification par reconnaissance faciale). Elle possède la méthode « **Compare2Images** » qui est responsable de l’exécution du script Python et du retour de résultat de comparaison.

Un utilisateur est représenté par la classe « **User** », celui-ci peut effectuer plusieurs réservations, représentées par la classe « **Booking** ».

Chaque réservation possède une entrée, la classe « **BookingEntry** », qui inclus les outils réservés par l’utilisateur, représentés par la classe « **Tools** »

## 3.4- Modèle des données

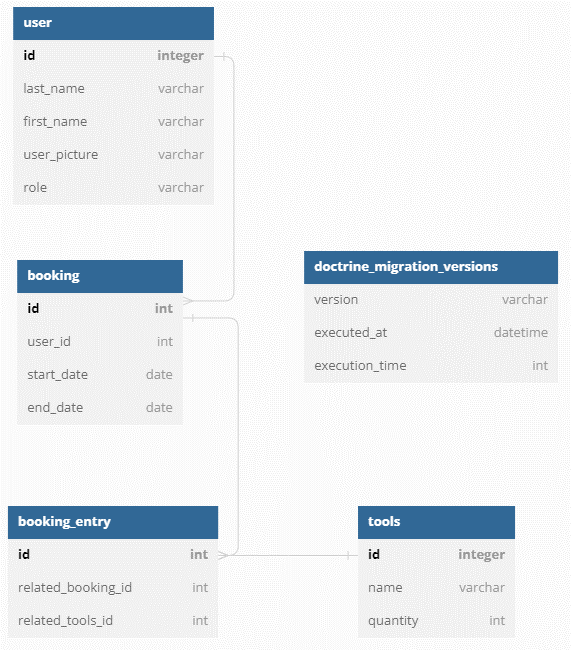
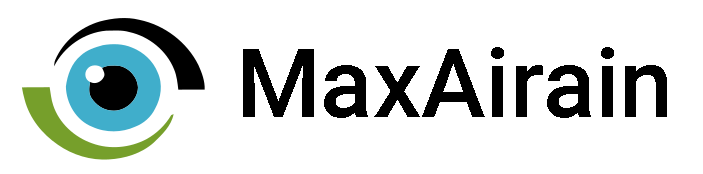


Illustration 4 : Modèle de base de données

La base de données permet de stocker certaines informations concernant l’application, telles que les informations relatives aux utilisateurs, ainsi que la partie de réservation de matériel avec les outils correspondants.

# **Charte graphique**



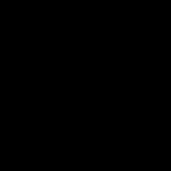
Le logo de la société MaxAirain :



Icône du logo seule :

## 4.1- Couleurs





#659224

VERT

#000000

TEXTES

#379EC1

TITRES

# **Serveur**

## 5.1- API

Notre API, créée grâce à API Platform de Symfony, est le pont de communication essentiel entre le serveur et les clients de notre application. Elle permet aux clients d'accéder de manière sécurisée et structurée aux fonctionnalités clés telles que la gestion des réservations, des utilisateurs, des outils et la comparaison d'images. Avec notre API, les clients peuvent intégrer facilement ces fonctionnalités dans leurs propres systèmes et applications.

|  |  |  |  |
| --- | --- | --- | --- |
| **URL** | **Verbes** | **Format servi** | **Description** |
| /api/users | GET | JSON | Récupère la liste des utilisateurs enregistrés dans le système. |
| /api/users | POST | JSON | Crée un nouvel utilisateur en utilisant les données fournies. |
| /api/users/{id} | GET | JSON | Récupère les détails d'un utilisateur spécifique à partir de son identifiant. |
| /api/users/{id} | PUT | JSON | Remplace les informations d'un utilisateur existant. |
| /api/users/{id} | PATCH | JSON | Met à jour les informations d'un utilisateur existant. |
| /api/users/{id} | DELETE | JSON | Supprime un utilisateur spécifique identifié par son identifiant. |

Dans cet exemple, l'API expose des fonctionnalités liées aux utilisateurs et aux outils. Le verbe GET est utilisé pour récupérer des informations spécifiques, le verbe POST permet d’ajouter de nouvelles données, DELETE pour en supprimer. Enfin, PUT les remplace et PATCH les met à jour.

Pour les autres routes de l’API, le fonctionnement des verbes HTTP est le même et nous avons :

* **/api/bookings** pour les réservations
* **/api/bookings\_entries** pour l’entrée de réservation (association entre une réservation et un/plusieurs outil(s) )
* **/api/tools** pour les outils

Pour récupérer des données spécifiques, il suffit d'ajouter **"/{id}"** à l'URL correspondante. Cela permet d'identifier de manière unique l'élément souhaité et d'obtenir ses informations détaillées.

# **Bugs rencontrés**

1. Problèmes liés au système d’authentification par reconnaissance faciale

# **Améliorations possibles**

1. Implémentation de JSON Web Tokens (JWT) : Utiliser des JWT pour l'authentification et l'autorisation des utilisateurs, offrant ainsi une couche supplémentaire de sécurité et de gestion des sessions.
2. Restriction de l'inscription des utilisateurs : Mettre en place une fonctionnalité permettant à l'administration uniquement d'ajouter de nouveaux agents au système, afin de garantir un contrôle plus strict sur les utilisateurs et d'éviter les inscriptions non autorisées.
3. Interface d'administration : Créer une interface d'administration dédiée accessible uniquement par les administrateurs, offrant des fonctionnalités avancées telles que la gestion des utilisateurs, la modification des autorisations, et la visualisation des statistiques.
4. Amélioration de l’authentification par reconnaissance faciale : Créer notre propre algorithme sans utiliser des bibliothèques Python, ainsi nous avons un contrôle total sur l'algorithme et les techniques utilisées, ce qui peut conduire à des performances améliorées et à une adaptation plus précise aux besoins de l'application, notamment sur la rapidité de comparaison d’images.
5. Aborder la problématique de RGPD

# **Annexes**

## 5.1- Table des illustrations

Illustration 1 : Architecture globale……………………………………………………………………………………………4

Illustration 2 : Diagramme des séquences…………………………………………………………………………….……5

Illustration 3 : Diagramme des classes………………………………………………………………………….……………6

Illustration 4 : Modèle de base de données……………………………………………………………………………….7