

# RAPPORT DE PROJET

Omar Hussein

BTS SNIR

2022  
2023



## Sommaire

I.	Introduction.....	3
a.	Présentation du projet.....	3
b.	Objectifs.....	4
c.	Méthodologie utilisée .....	5
d.	Gestion du projet avec une structure de dossiers organisée .....	6
e.	Gestion du projet avec un diagramme de Gantt .....	6
II.	Architecture et conception du système.....	8
f.	Description de la solution proposée.....	8
g.	Diagramme des cas d'utilisation.....	9
h.	Architecture globale du système .....	10
i.	Rôle du transistor IRF520 .....	12
III.	Conception de la base de données (Omar Hussein) .....	13
j.	Le rôle de la base de données synchronisée .....	13
k.	Hébergement de la base de données.....	13
l.	Processus de conception .....	14
m.	Evènements phpmyadmin.....	17
IV.	Conception de l'interface de gestion (IG) (Omar Hussein).....	19
n.	Fonctionnalités de l'interface de gestion .....	19
o.	Demandes d'accès .....	21
p.	Envoi des courriels électroniques.....	24
q.	Génération des QR-Codes .....	26
r.	Visualisation de l'état d'occupation des ressources.....	29
s.	Historique d'accès aux ressources.....	31
t.	Gestion de l'accès par reconnaissance faciale.....	32
u.	Accès aux serrures .....	34
v.	Conception visuelle : L'optimisation de l'interface grâce au CSS .....	35
V.	La serrure QR-code (Adama Sylla).....	39
w.	Présentation de la serrure QR-code .....	39
x.	Étude préliminaire - QR code : .....	41
y.	Le matériel utilisé .....	44
z.	Choix technologiques effectués .....	47
aa.	Installation de ressources .....	51
bb.	Conception et réalisation .....	53

cc.	Explication du code.....	56
dd.	Etablir une connexion avec la base de donne : .....	61
ee.	Configuration de la led : .....	63
ff.	Configuration de la serrure :.....	65
gg.	Code final :.....	67
hh.	Conclusion .....	70
VI.	La serrure biométrique par reconnaissance faciale (Joël Nyobe) .....	71
ii.	Le rôle de la serrure biométrique:.....	71
jj.	Matériels à disposition : .....	71
kk.	Connecter à la Raspberry : .....	71
II.	Synchronisation avec la base de données :.....	73
mm.	Reconnaissance faciale:.....	74
nn.	Accès au ressources et enregistrement dans l'historique : .....	75
oo.	LED:.....	76
pp.	Télécharger OpenCV : .....	76
qq.	Serrure électrique :.....	78
rr.	Schéma électrique : .....	78
ss.	Test : .....	80
tt.	Totalité du code : .....	81
uu.	Notice d'utilisation : .....	83
VII.	Conclusion .....	84
VIII.	Annexe.....	85
vv.	Codes sources.....	85
ww.	Photo du Tribunal d'Instance d'Evry .....	86

## I. Introduction

### a. Présentation du projet

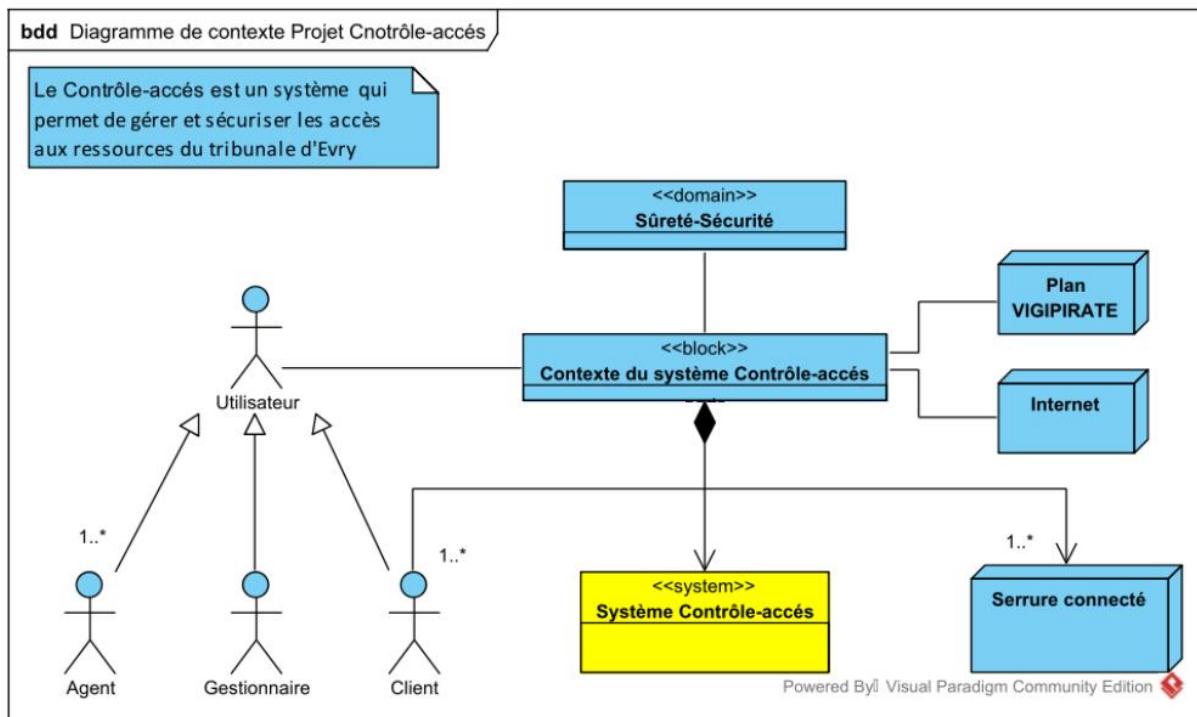
Le contrôle d'accès est un enjeu majeur pour la sécurité des bâtiments, notamment dans les institutions judiciaires. C'est dans ce contexte que s'inscrit le présent projet, qui vise à développer une solution de contrôle d'accès pour le tribunal de grande instance d'Evry.

Actuellement, l'accès est géré par un vigile qui délivre des badges permettant de se déplacer librement dans le bâtiment. Cependant, ce système présente une faille de sécurité majeure, permettant un accès non autorisé à des zones sensibles. Afin de résoudre ce problème, le système Contrôle-Accès a été conçu pour renforcer la sécurité et simplifier la gestion des accès.

L'objectif principal de cette solution est de donner aux responsables du tribunal un contrôle total sur les accès en décidant qui peut accéder où et quand. Grâce à l'utilisation de QR codes d'accès temporaires ou permanents, les utilisateurs peuvent accéder aux ressources du tribunal uniquement pendant les périodes autorisées. De plus, l'introduction d'un contrôle biométrique pour l'accès aux locaux sensibles garantit une sécurité renforcée. Cette mesure de sécurité supplémentaire consiste en une reconnaissance faciale pour les dossiers et les pièces à conviction des affaires criminelles.

En somme, le projet Contrôle-Accès pour le tribunal de grande instance d'Evry constitue une avancée majeure dans la gestion de la sécurité et des accès. En renforçant la sécurité, en simplifiant la gestion des accès et en offrant une visibilité complète sur les flux de personnes, cette solution contribue à garantir l'intégrité des ressources du tribunal et la protection des utilisateurs et des biens.

Le présent document expose les spécifications de la solution de contrôle d'accès pour le tribunal de grande instance d'Evry, y compris les diagrammes SysML et les contraintes de réalisation. Le projet ne prend pas en compte la gestion des dysfonctionnements possibles sur le système embarqué, tels que la panne de la serrure ou de la caméra.



**Figure 1.** Diagramme de contexte

b. Objectifs

Les objectifs du projet sont les suivants :

- Développer une solution de contrôle d'accès pour le tribunal de grande instance d'Evry.
  - Permettre la gestion à distance de l'accès aux ressources du tribunal, notamment les immeubles, les étages, les locaux et les salles d'audience.
  - Simplifier la gestion et le contrôle des accès en permettant de décider qui peut accéder, où et quand.
  - Fournir des QR-codes d'accès temporaires ou permanents aux utilisateurs pour accéder aux ressources uniquement pendant les périodes autorisées.
  - Exiger un contrôle biométrique pour l'accès aux locaux sensibles, tels que ceux réservés pour les dossiers et les pièces à conviction des affaires criminelles.
  - Remplacer les clés et les badges par des QR-codes pour réduire les pertes de supports et limiter les usages abusifs.
  - Limiter les accès temporaires ou dérogatoires aux périodes autorisées.
  - Offrir une vue globale et synthétique sur le niveau d'occupation des ressources, ainsi que l'historique des accès en temps réel.
  - Développer trois sous-systèmes : l'interface de gestion, la serrure QR-Code et la serrure biométrique par reconnaissance faciale.

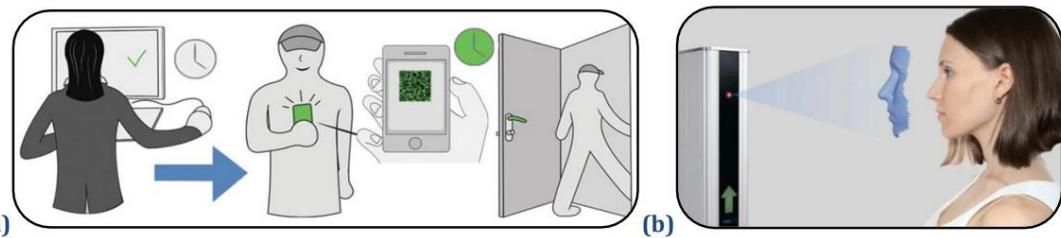


Figure 2. (a) Accès avec QR-code.

(b) Contrôle d'accès par reconnaissance faciale.

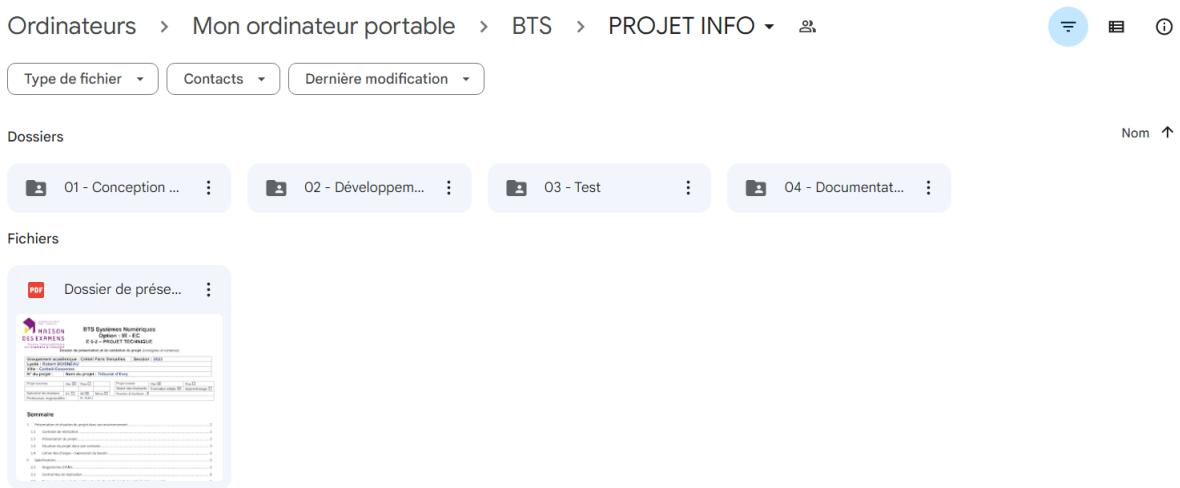
### c. Méthodologie utilisée

Pour répondre aux objectifs du projet, nous avons utilisé une méthodologie en plusieurs étapes :

- **Etape 1. Analyse des besoins** : L'analyse des besoins consiste à identifier les besoins du client et les spécifications fonctionnelles et techniques du système de contrôle d'accès. Cette étape comprend la rédaction du cahier des charges et la réalisation des diagrammes SysML, tels que le diagramme de contexte, le diagramme des cas d'utilisation, le diagramme d'exigence et le diagramme de séquence. Elle permet de comprendre les besoins des utilisateurs et les fonctionnalités du système de contrôle d'accès.
- **Etape 2. Conception** : La conception consiste à concevoir l'architecture globale du système, ainsi que les sous-systèmes de l'interface de gestion, de la serrure QR-Code et de la serrure biométrique par reconnaissance faciale. Cette étape permet de déterminer les différentes fonctionnalités de chaque sous-système et de s'assurer de leur intégration harmonieuse.
- **Etape 3. Réalisation** : La réalisation consiste à mettre en place la plateforme de développement et à programmer les différents sous-systèmes. Cette étape inclut également les tests de validation de chaque composant et du système global. Elle permet de vérifier la conformité du système aux spécifications fonctionnelles et techniques définies lors de l'analyse des besoins.
- **Etape 4. Tests** : Une fois les sous-systèmes développés et intégrés, un test de validation sera effectué pour vérifier le bon fonctionnement de l'ensemble du système. Le test permettra de s'assurer que les serrures s'ouvrent et se ferment correctement, que les QR-codes et la reconnaissance faciale fonctionnent correctement, que les autorisations d'accès sont bien prises en compte et que l'interface de gestion offre une vue synthétique et en temps réel sur l'état du système. Des scénarios de test seront élaborés pour couvrir différents cas d'utilisation possibles.

#### d. Gestion du projet avec une structure de dossiers organisée

Pour gérer le projet, j'ai créé un dossier sur mon PC que j'ai synchronisé avec Google Drive afin que mes camarades puissent également y accéder. Ce dossier est organisé en quatre sous-dossiers principaux : "01 - Conception et planification", "02 - Développement", "03 - Test" et "04 - Documentation finale".



**Figure 3. Organisation des dossiers du projet**

Nous avons organisé le projet en utilisant quatre sous-dossiers principaux. Le premier, "Conception et planification", contenait tous les documents liés à la phase initiale du projet, tels que les spécifications et les diagrammes. Le deuxième, "Développement", regroupait tous les fichiers liés à l'écriture du code et à la création des fichiers HTML, CSS et JavaScript. Le troisième, "Test", était dédié aux tests pour vérifier le bon fonctionnement du site web et corriger les éventuels problèmes. Enfin, le quatrième, "Documentation finale", rassemblait tous les fichiers et documents nécessaires pour présenter le projet de manière complète et professionnelle. Cette organisation nous a permis de travailler de manière structurée et de faciliter la collaboration entre les membres de l'équipe.

#### e. Gestion du projet avec un diagramme de Gantt

Nous avons utilisé un diagramme de Gantt pour planifier et organiser les différentes étapes de notre projet. Le diagramme de Gantt nous a permis d'avoir une vision claire et chronologique des tâches à réaliser. Voici quelques exemples de tâches présentes dans le diagramme :

- "Access Tribunal" : Cette tâche était prévue pour durer 17 semaines et 1 jour et était liée à l'accès aux tribunaux.
- "Recherche et documentation" : Cette tâche avait une durée prévue de 2 semaines et 32 minutes et impliquait la recherche et la collecte de documentation pertinente.
- "Conception et planification" : Cette phase du projet était prévue pour durer 4 semaines et 3 jours et comprenait des tâches telles que la conception de l'architecture et l'élaboration des spécifications détaillées.

- "Développement" : Cette phase était prévue pour durer 6 semaines et 2 jours et incluait des tâches spécifiques telles que le développement de l'interface utilisateur, la serrure QR et la serrure biométrique.
- "Test" : Cette phase était prévue pour durer 1 semaine et 3 jours et comprenait des tests unitaires, des tests d'intégration, des tests de performance et des tests utilisateurs.
- "Documentation finale" : Cette phase était prévue pour durer 1 semaine et 2 jours et impliquait la rédaction du manuel d'utilisation, de la documentation technique et de la documentation de support.

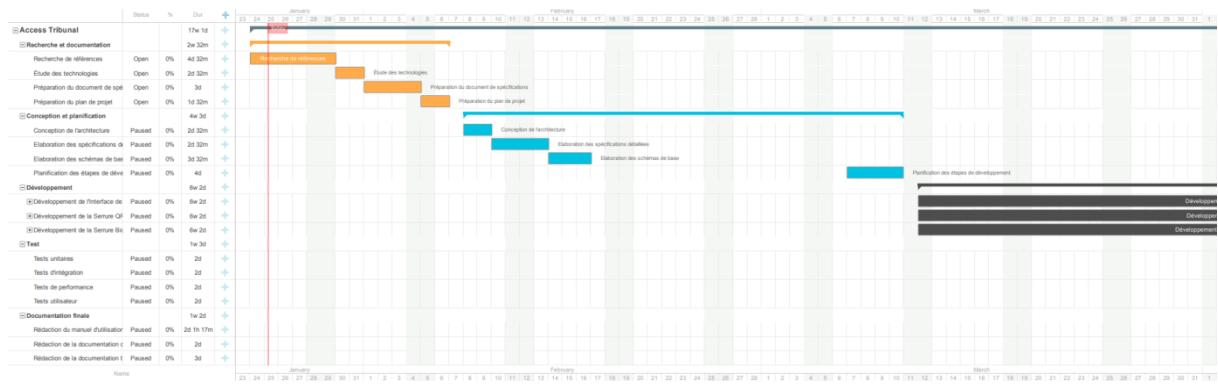
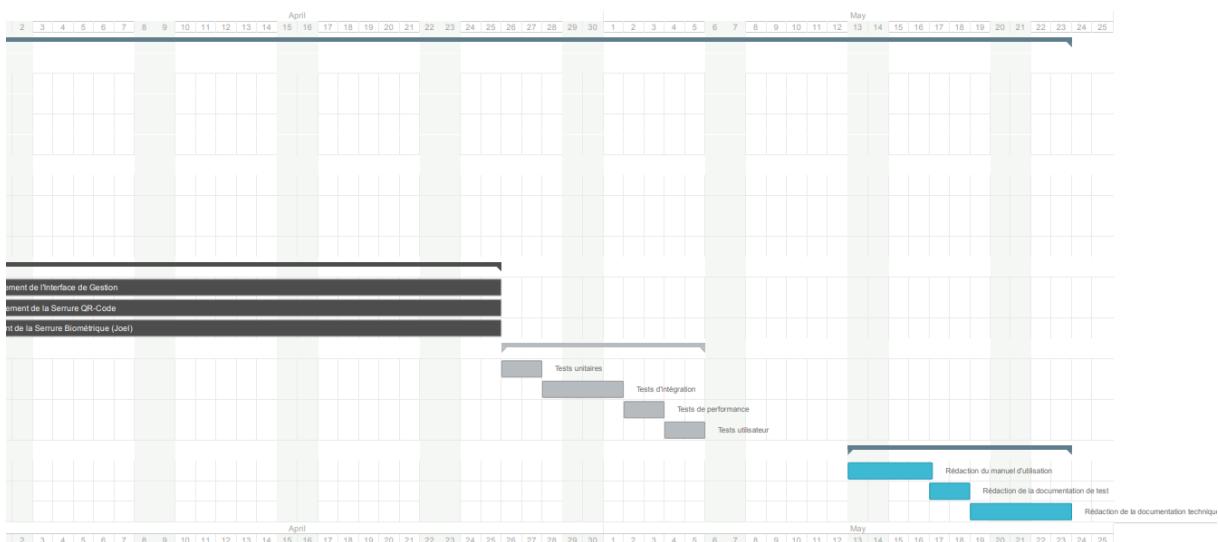


Figure 4. (a) Diagramme de Gantt (1/2)



(b) Diagramme de Gantt (2/2)

Grâce au diagramme de Gantt, nous avons pu visualiser les dépendances entre les différentes tâches, gérer les ressources et respecter les délais fixés pour chaque étape du projet. Cela nous a permis de travailler de manière efficace et de suivre l'avancement du projet de manière claire et organisée.

## II. Architecture et conception du système

### f. Description de la solution proposée

La solution proposée pour le contrôle d'accès au tribunal de grande instance d'Evry comprend trois principaux sous-systèmes interconnectés : l'interface de gestion, la serrure QR-Code et la serrure biométrique par reconnaissance faciale.

- **Etudiant 1 (moi, Omar Hussein), Interface de gestion :** L'interface de gestion est une application web développée en utilisant les langages HTML, CSS, PHP et SQL. Elle permet à l'administrateur de gérer les accès aux différentes ressources du tribunal de manière centralisée. HTML est utilisé pour la structure de la page, CSS pour la mise en forme et le style, PHP pour la logique de gestion des utilisateurs et des accès, et SQL pour interagir avec la base de données.  
L'interface de gestion offre des fonctionnalités telles que la validation des demandes d'accès des utilisateurs, la génération et impression des QR-codes d'accès, la gestion des accès par reconnaissance faciale, la gestion de la messagerie électronique, la visualisation de l'état d'occupation des ressources, la consultation de l'historique des accès et l'accès aux systèmes embarqués des serrures. Elle permet ainsi à l'administrateur de prendre des décisions éclairées sur les autorisations d'accès, de surveiller l'activité en temps réel et de gérer efficacement les utilisateurs et les ressources.
- **Etudiant 2 (Adama Scylla), Serrure QR-Code :** Les serrures QR-Code sont pilotées par des Raspberry Pi à l'aide d'un code Python spécifique. Lorsqu'un utilisateur présente un QR-code valide à la serrure, celle-ci lit et décode le QR-code à l'aide de la caméra intégrée au Raspberry Pi. Si le QR-code est authentifié avec succès, la serrure actionne la gâche de la porte, permettant ainsi l'accès à l'utilisateur. Une LED verte est utilisée pour indiquer l'ouverture de la gâche, tandis qu'une LED rouge indique que la gâche est fermée. Ces indications visuelles permettent de confirmer visuellement si la porte s'est ouverte ou non lors d'une reconnaissance.
- **Etudiant 3 (Joël Nyobe), Serrure biométrique par reconnaissance faciale :** Les serrures biométriques par reconnaissance faciale sont également pilotées par des Raspberry Pi à l'aide d'un code Python spécifique. Lorsqu'un utilisateur se présente devant la serrure, la caméra intégrée au Raspberry Pi capture son visage et utilise un algorithme de reconnaissance faciale pour l'authentifier. Si l'utilisateur est reconnu avec succès, la serrure actionne la gâche de la porte, permettant ainsi l'accès. De même que pour les serrures QR-Code, une LED verte indique l'ouverture de la gâche, tandis qu'une LED rouge indique que la gâche est fermée.

La solution complète de contrôle d'accès offre une gestion centralisée, flexible et sécurisée des accès au tribunal de grande instance d'Evry. Grâce à l'interface de gestion développée en HTML, CSS, PHP et SQL, les administrateurs peuvent prendre des décisions éclairées sur les autorisations d'accès, surveiller l'activité en temps réel et gérer efficacement les utilisateurs et les ressources. Les serrures QR-Code et biométriques, pilotées par des Raspberry Pi à l'aide de code Python, garantissent un contrôle précis des accès et offrent des indications visuelles claires sur l'état de la gâche. Cette solution permet ainsi de renforcer la sécurité du tribunal, de limiter les usages abusifs et de fournir un suivi détaillé de l'historique des accès.

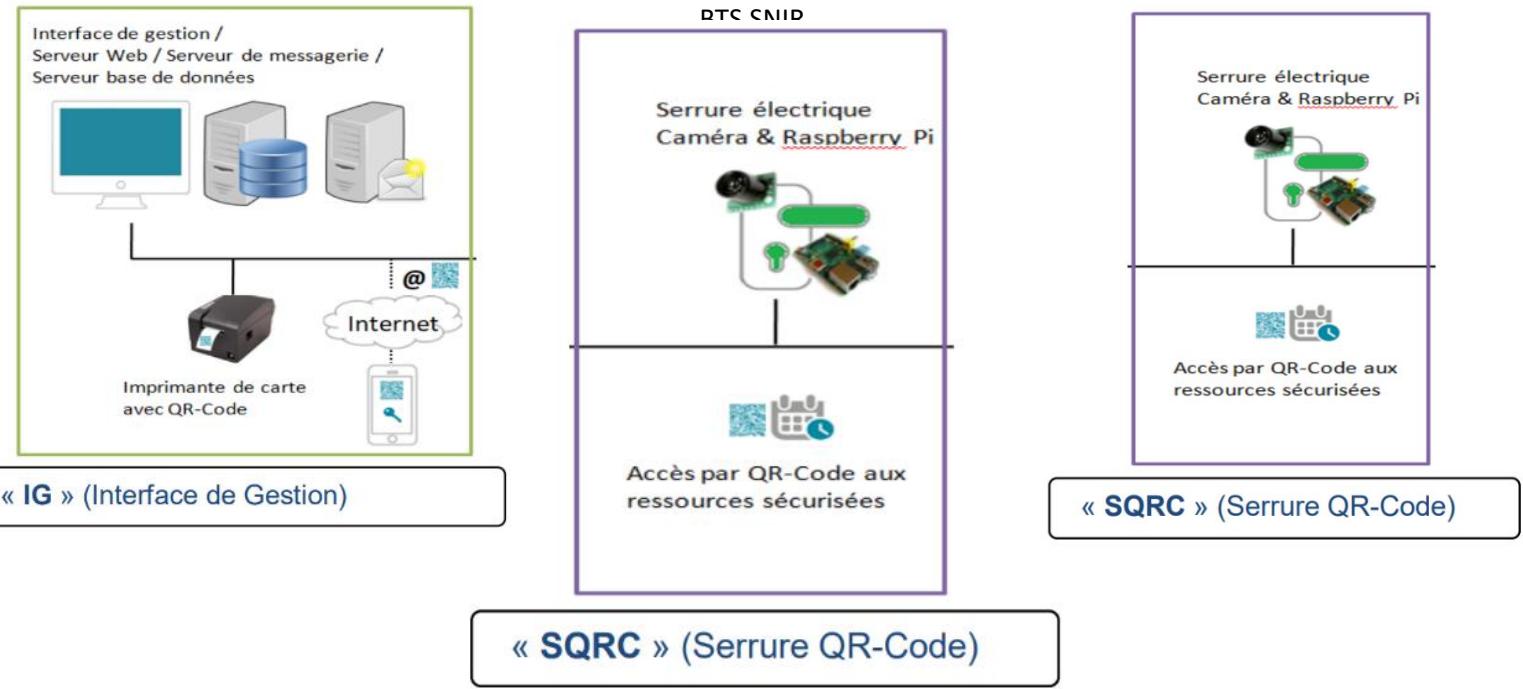


Figure 5. Représentation des trois sous-systèmes.

### g. Diagramme des cas d'utilisation

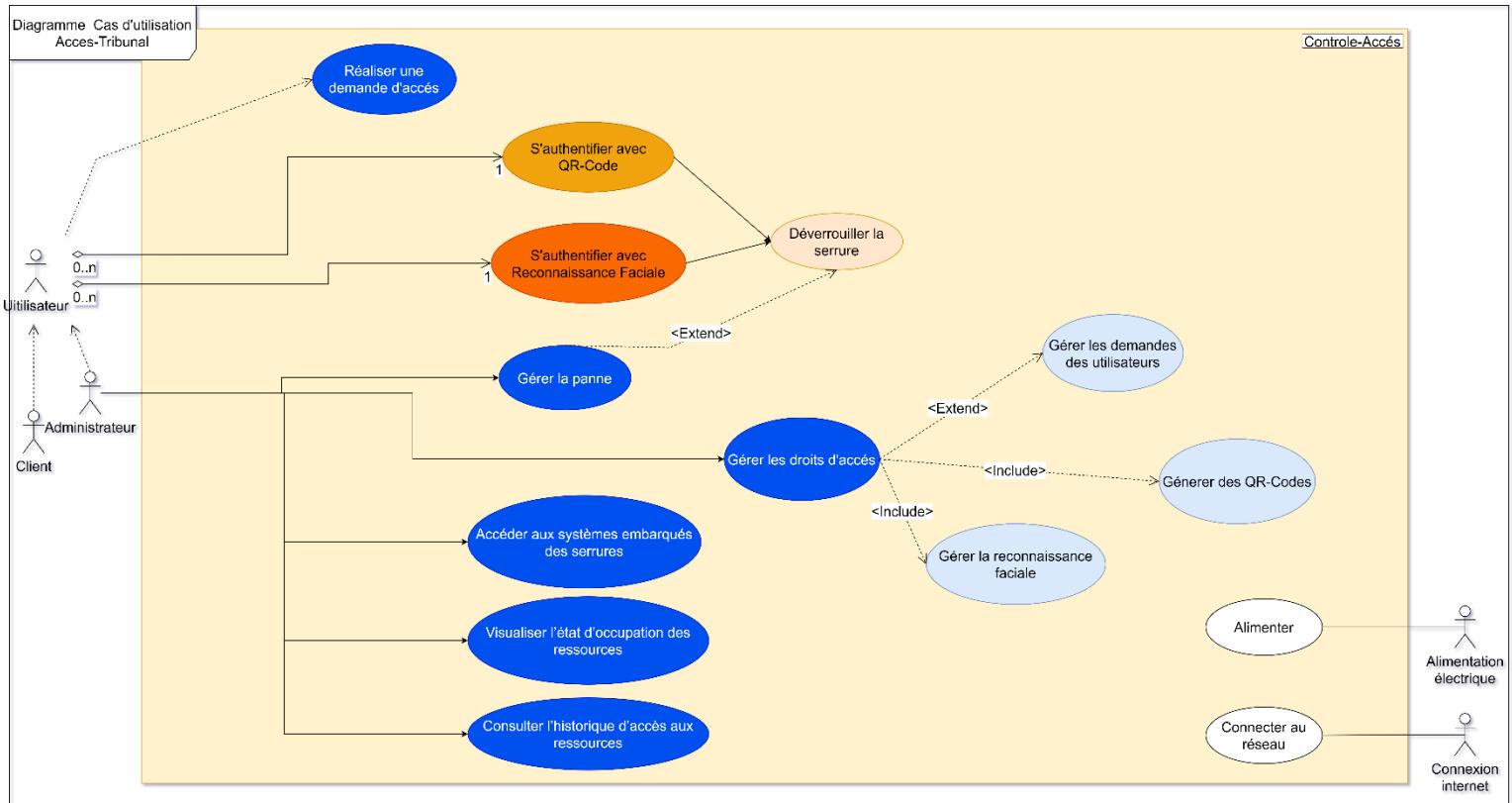


Figure 6. Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation représente de manière visuelle les différentes fonctionnalités et interactions du système de contrôle d'accès. Il met en évidence les acteurs impliqués et les actions qu'ils peuvent effectuer. Le diagramme utilise un code couleur pour distinguer les différentes parties

du système. Les fonctionnalités accessibles depuis l'interface graphique sont en bleu, celles liées aux QR-Codes sont en orange, et celles liées à la reconnaissance faciale sont en rouge.

À droite du diagramme, deux acteurs sont représentés : l'alimentation électrique, qui est connectée à "alimenter", et la connexion internet, qui est connectée à "Connecter au réseau". Ces acteurs sont nécessaires au bon fonctionnement du système. À gauche du diagramme, nous avons trois acteurs principaux : l'utilisateur, l'administrateur et le client. L'administrateur a plusieurs fonctionnalités à sa disposition, notamment l'accès aux systèmes embarqués des serrures, la visualisation de l'état d'occupation des ressources, la consultation de l'historique d'accès aux ressources, la gestion des droits d'accès et la gestion des pannes. La gestion des droits d'accès est étendue par la gestion des demandes des utilisateurs, la génération de QR-Codes et la gestion de la reconnaissance faciale. En cas de panne, l'administrateur peut également déverrouiller la serrure. L'acteur utilisateur est lié à plusieurs actions, dont l'authentification avec un QR-Code, l'authentification avec la reconnaissance faciale et la possibilité de réaliser une demande d'accès. Ces actions sont toutes connectées à l'action de déverrouillage de la serrure, permettant ainsi à l'utilisateur d'obtenir l'accès autorisé.

Le diagramme des cas d'utilisation permet de visualiser clairement les interactions entre les acteurs et les fonctionnalités du système. Il montre comment les utilisateurs peuvent s'authentifier, réaliser des demandes d'accès et bénéficier de la fonctionnalité de déverrouillage de la serrure.

L'administrateur dispose également d'un ensemble d'actions pour gérer les droits d'accès, surveiller l'état d'occupation des ressources et traiter les pannes éventuelles.

En résumé, ce diagramme met en évidence les principales fonctionnalités du système de contrôle d'accès, en utilisant des codes couleur pour distinguer les différentes parties du système. Il permet de comprendre de manière visuelle les interactions entre les acteurs et les actions qu'ils peuvent effectuer, facilitant ainsi la compréhension globale du fonctionnement du système.

#### h. Architecture globale du système

L'architecture globale du système repose sur un PC servant de serveur web, qui héberge l'interface de gestion, la base de données (BDD) et une imprimante. Les Raspberry Pi, connectés aux serrures QR Code et reconnaissance faciale, utilisent des caméras pour lire les QR codes et détecter les visages. Une fois l'accès authentifié, les Raspberry Pi activent les LED correspondantes et contrôlent la gâche électrique pour ouvrir la porte si l'accès est autorisé. La synchronisation des données est assurée grâce à la connexion des Raspberry Pi à la base de données (BDD) hébergée sur le PC.

Juste au-dessus de ce paragraphe, vous trouverez un schéma de montage d'une Raspberry Pi, qui représente le câblage et les connexions nécessaires pour réaliser le système. Ce schéma offre une visualisation claire de la configuration technique, facilitant ainsi la compréhension du fonctionnement global du système de contrôle d'accès.

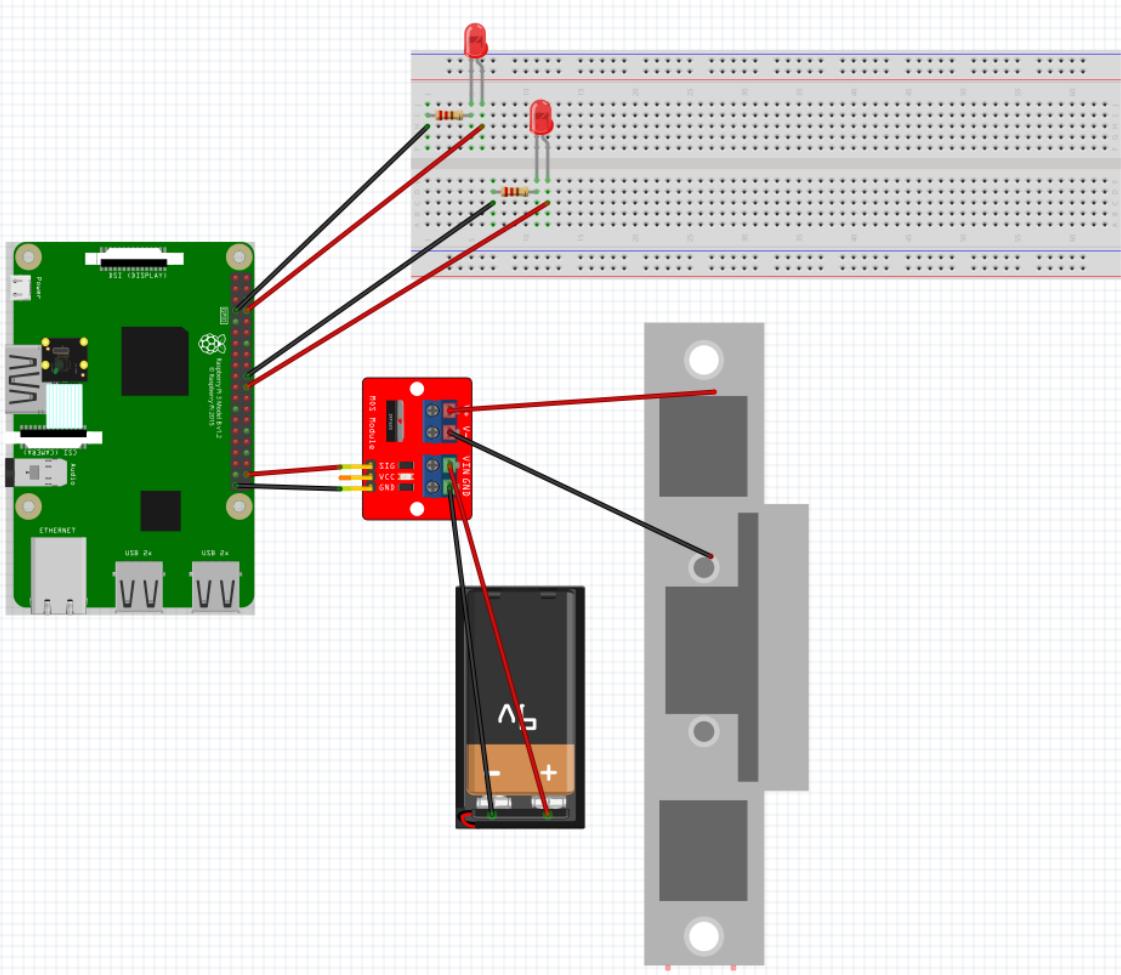


Figure 7. Schéma de montage des Raspberry Pi

Le schéma de montage du système de contrôle d'accès met en évidence les connexions et les composants utilisés pour assurer le bon fonctionnement du système. La Raspberry Pi est le cœur du système et est connectée à deux LED, une rouge et une verte. Chaque LED est alimentée par un câble d'alimentation (câble rouge) et un câble de masse (câble noir).

Ensuite, la Raspberry Pi est reliée à un transistor IRF520, dont les broches SIG et GND sont connectées à la Raspberry Pi. De l'autre côté du transistor, il est connecté à la serrure à l'aide de deux câbles. Le câble V- (câble de masse, noir) et le câble V+ (câble d'alimentation, rouge) permettent de fournir l'alimentation nécessaire à la serrure. Pour alimenter la serrure, le transistor est également connecté à VIN (câble d'alimentation, rouge) et à GND (câble de masse, noir) qui sont reliés à une pile. Cette configuration permet de contrôler l'ouverture et la fermeture de la serrure électriquement.

Grâce à cette configuration électronique, le système est en mesure de contrôler l'activation des LED (rouge et verte) pour indiquer si l'accès est autorisé ou refusé. De plus, il est capable de contrôler la gâche électrique de la serrure pour permettre l'ouverture ou la fermeture de la porte en fonction de l'autorisation d'accès.

### i. Rôle du transistor IRF520

Le transistor IRF520 joue un rôle crucial dans le schéma de montage du système de contrôle d'accès. Sa présence est essentielle pour éviter les risques de court-circuit et protéger la Raspberry Pi.

Le transistor agit comme un interrupteur électronique qui permet de contrôler le courant entre la Raspberry Pi et la serrure électrique. Dans ce cas, le transistor est monté en tant que commutateur de type N, ce qui signifie qu'il conduit le courant lorsque le signal de commande est appliqué à sa broche SIG.

L'un des avantages clés du transistor est sa capacité à isoler électriquement les parties du système. Il divise la Raspberry Pi, qui fonctionne à une tension plus basse, de la serrure électrique, qui peut nécessiter une tension plus élevée pour son fonctionnement. En connectant la broche SIG du transistor à une broche GPIO de la Raspberry Pi, le signal de commande peut être activé ou désactivé. Lorsque le signal de commande est appliqué, le transistor conduit le courant et permet à la Raspberry Pi de contrôler l'alimentation de la serrure électrique. Cela permet à la Raspberry Pi de décider quand activer ou désactiver la serrure.

En évitant un contact direct entre la Raspberry Pi et la serrure, le transistor offre une protection supplémentaire à la Raspberry Pi contre les variations de tension potentielles qui pourraient endommager le circuit ou compromettre son fonctionnement. Il agit comme une barrière électrique en fournissant un chemin de courant contrôlé entre la Raspberry Pi et la serrure. Grâce à la présence du transistor, la Raspberry Pi peut communiquer de manière sûre et fiable avec la serrure électrique, en fournissant les signaux de commande nécessaires pour l'ouverture et la fermeture de la porte. Cette fonctionnalité garantit un fonctionnement stable du système de contrôle d'accès et prolonge la durée de vie de la Raspberry Pi en la protégeant contre d'éventuels dommages.

En résumé, le transistor IRF520 joue un rôle essentiel dans le schéma de montage en fournissant une isolation électrique entre la Raspberry Pi et la serrure électrique. Il agit comme un interrupteur électronique contrôlé par la Raspberry Pi, permettant de gérer l'alimentation de la serrure de manière sécurisée et fiable. Sa présence protège la Raspberry Pi contre les risques de court-circuit et garantit un fonctionnement optimal du système de contrôle d'accès.

### III. Conception de la base de données (Omar Hussein)

#### j. Le rôle de la base de données synchronisée

La base de données synchronisée joue un rôle essentiel dans l'authentification fiable des utilisateurs dans notre système de contrôle d'accès pour le tribunal de grande instance d'Evry.

L'authentification des utilisateurs est une étape critique pour garantir la sécurité du système et des ressources sensibles du tribunal. En centralisant les données d'authentification dans une base de données synchronisée, nous pouvons mettre en place des mécanismes solides pour vérifier l'identité des utilisateurs de manière fiable. La base de données stocke les informations d'identification des utilisateurs autorisés, telles que leurs données personnelles, leurs codes d'accès, et les informations biométriques dans le cas de la reconnaissance faciale. Lorsqu'un utilisateur tente d'accéder aux ressources, l'interface de gestion peut interroger la base de données pour authentifier l'utilisateur en comparant les données fournies avec celles enregistrées.

Grâce à cette authentification fiable basée sur la base de données synchronisée, nous pouvons éviter les accès non autorisés et prévenir les tentatives de contournement du système. Les informations d'authentification sont mises à jour en temps réel dans la base de données, ce qui garantit que seules les personnes autorisées peuvent accéder aux ressources et aux locaux sensibles du tribunal. De plus, la synchronisation de la base de données permet également de maintenir l'historique des accès, ce qui facilite la traçabilité et la vérification ultérieure des activités des utilisateurs. En enregistrant les données d'accès, telles que la date, l'heure et les ressources consultées, dans la base de données synchronisée, nous pouvons créer un journal d'audit complet et fiable pour des raisons de sécurité et de responsabilité.

En résumé, la base de données synchronisée joue un rôle crucial dans l'authentification fiable des utilisateurs dans notre système de contrôle d'accès. Elle stocke les informations d'identification, permettant une vérification précise de l'identité des utilisateurs autorisés, et facilite la traçabilité des activités à des fins d'audit et de sécurité. Grâce à cette synchronisation, nous garantissons un niveau élevé de sécurité et de confiance dans l'accès aux ressources du tribunal de grande instance d'Evry.

#### k. Hébergement de la base de données

Lorsque j'ai conçu la base de données pour gérer les demandes d'accès et les autorisations au tribunal de grande instance d'Evry, il était essentiel de trouver un moyen fiable et accessible d'héberger la base de données. Après mûre réflexion, j'ai opté pour l'utilisation de XAMPP, un environnement de développement web polyvalent, qui m'a permis de créer un serveur local pour héberger la base de données.

XAMPP est une suite logicielle qui combine Apache, MySQL, PHP et phpMyAdmin. J'ai choisi d'installer XAMPP sur mon ordinateur principal, qui agit comme un serveur pour le projet. Cela m'a offert plusieurs avantages. Tout d'abord, en utilisant Apache comme serveur web, j'ai pu accéder à ma base de données et à d'autres fichiers liés au projet, tels que l'interface de gestion, à partir de n'importe quel navigateur web sur mon réseau local. Cela signifie que mes camarades de projet et moi-même pouvions accéder à la base de données via nos Raspberry Pi en utilisant l'adresse IP de mon ordinateur principal.

En ce qui concerne la gestion de la base de données, j'ai utilisé phpMyAdmin, une interface conviviale qui permet d'administrer facilement la base de données MySQL. Avec phpMyAdmin, j'ai pu créer et gérer les tables, exécuter des requêtes SQL et effectuer d'autres tâches d'administration essentielles. J'ai également créé des identifiants d'accès spécifiques pour mes camarades de projet, leur permettant ainsi d'accéder à la base de données avec leurs propres Raspberry Pi.



Figure 8. Logo de XAMPP, phpMyAdmin et Apache

### I. Processus de conception

Lorsque j'ai entrepris de concevoir une base de données pour gérer les demandes d'accès et les autorisations au tribunal de grande instance d'Evry, j'ai fait preuve de créativité et de réflexion approfondie. À l'époque, je cherchais une approche simple et intuitive pour répondre aux besoins de base de ce système.

Pour concevoir la structure de la base de données, j'ai commencé par analyser les différentes entités impliquées. J'ai identifié les ressources, les demandes d'accès et les relations entre elles comme les composants clés du système. Ensuite, j'ai créé la table "Ressources" pour stocker les informations relatives aux ressources, telles que le nom, le type de ressource, la sensibilité, le statut et la capacité. Cependant, avec le recul, j'ai réalisé que l'ancienne version de la base de données présentait quelques lacunes. L'une des erreurs que j'ai commises était de ne pas avoir pris en compte la possibilité d'une capacité variable pour chaque ressource. J'avais utilisé une seule colonne "Capacité" pour stocker la capacité de toutes les ressources, ce qui limitait la flexibilité du système.

De plus, j'ai constaté que la gestion des demandes d'accès était insuffisante dans l'ancienne version. La table "Demandes\_acces" stockait les informations basiques des demandes, comme le nom, le prénom, l'e-mail, les dates et le statut. Cependant, il manquait des fonctionnalités importantes, telles que la génération de QR codes uniques pour chaque demande, ce qui aurait renforcé la sécurité du système.

Pour corriger ces erreurs et améliorer la conception de la base de données, j'ai effectué des recherches approfondies et trouvé de l'inspiration dans d'autres systèmes de contrôle d'accès complexes. J'ai réalisé que j'avais besoin d'une approche plus robuste pour gérer les relations entre les demandes d'accès, les ressources et les visages des utilisateurs.

Dans la nouvelle version de la base de données, j'ai consciemment amélioré la structure en introduisant de nouvelles tables et relations pour permettre une gestion plus précise et flexible des informations. Une des améliorations majeures a été l'ajout de la table "Visages" qui permet de stocker les informations relatives aux visages des utilisateurs, ainsi que leur statut de validation. De plus, j'ai créé les tables intermédiaires "Demandes\_acces\_ressources" et "Visages\_acces\_ressources" pour gérer les relations many-to-many entre les demandes, les

visages et les ressources. Ces tables de liaison permettent de représenter de manière efficace les associations complexes entre ces entités. Elles jouent un rôle essentiel dans la gestion des accès en permettant de déterminer rapidement quelles demandes ont accès à quelles ressources et quels visages sont associés à quelles ressources. La table "Demandes\_acces\_ressources" permet de relier les demandes d'accès aux ressources correspondantes, tandis que la table "Visages\_acces\_ressources" établit les liens entre les visages et les ressources. Grâce à ces relations many-to-many, il devient possible de gérer facilement les scénarios où plusieurs demandes peuvent avoir accès à plusieurs ressources et où plusieurs visages peuvent être associés à différentes ressources.

Ces améliorations ont permis de prendre en compte la capacité variable des ressources en ajoutant les colonnes "Capacite\_totale" et "Capacite\_actuelle" dans la table "Ressources". De plus, j'ai introduit la génération de QR codes uniques pour chaque demande en ajoutant la colonne "Valeur\_QR\_Code" dans la table "Demandes\_acces".

En résumé, l'ancienne version de la base de données, bien que réalisée avec créativité et réflexion, présentait des lacunes en termes de gestion de la capacité des ressources et de fonctionnalités de demande d'accès. Cependant, grâce à des recherches approfondies et à l'inspiration tirée d'autres systèmes de contrôle d'accès, j'ai pu concevoir une nouvelle version de la base de données qui corrigeait ces erreurs et offrait une structure plus solide et flexible pour gérer les demandes d'accès et les autorisations au tribunal de grande instance d'Evry.

Pour mieux illustrer ces changements, voici deux schémas : un représentant l'ancienne version et un représentant la version finale.

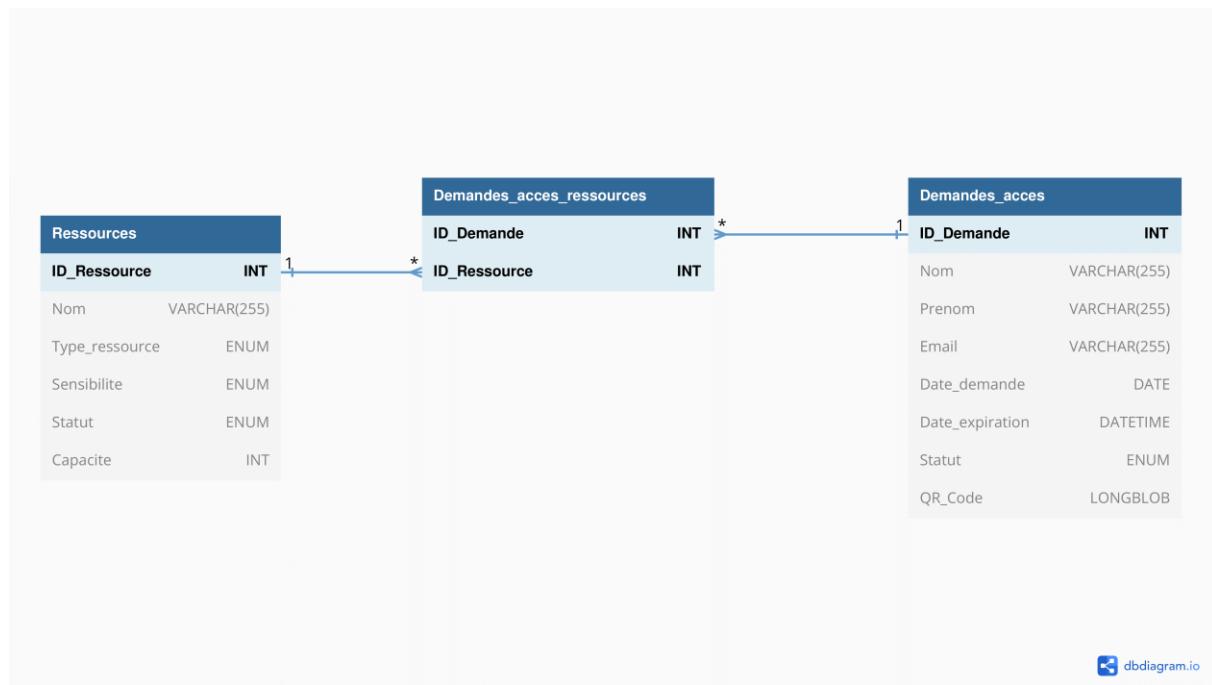
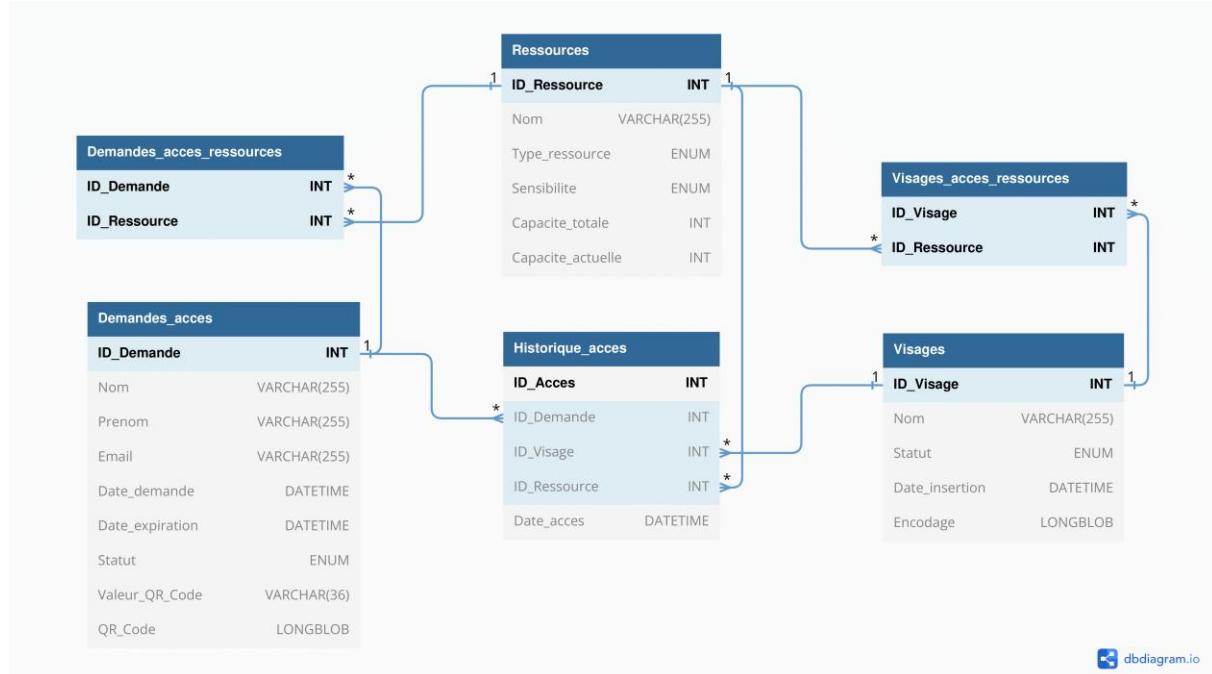


Figure 9. (a) première version de la base de données



(b) version finale de la base de données

#### m. Événements phpmyadmin

Pour assurer une mise à jour constante des données dans la base de données, j'ai intégré des événements dans ma base de données gérée par phpMyAdmin. Les événements sont des fonctionnalités qui permettent d'automatiser des tâches récurrentes et périodiques. Ils jouent un rôle essentiel dans la gestion et la mise à jour automatique des données. Dans le contexte de mon projet, j'ai utilisé les événements pour effectuer des mises à jour régulières et cohérentes, assurant ainsi la fiabilité et l'intégrité de ma base de données.

- **Événement 1 : Mise à jour du statut des demandes expirées toutes les 20 secondes :**  
L'événement "Mise à jour du statut des demandes expirées" a été créé pour garantir que les demandes qui ont atteint leur date d'expiration soient correctement traitées. Toutes les 20 secondes, l'événement se déclenche et exécute une requête de mise à jour. Cette requête vérifie si la date d'expiration d'une demande est antérieure ou égale à l'instant présent (NOW()). Si c'est le cas, le statut de la demande est automatiquement modifié pour passer à "Expiré". Cela permet de maintenir la base de données à jour et de refléter avec précision l'état actuel des demandes.



```
UPDATE Demandes_acces
SET Statut = 'Expiré'
WHERE Date_expiration <= NOW()
```

Figure 10. (a) Événement 1 de la base de données

- **Événement 2 : Mise à jour des capacités des ressources toutes les 30 secondes :** Pour assurer une gestion efficace des capacités des ressources, j'ai mis en place l'événement "Mise à jour des capacités des ressources". Cet événement se déclenche toutes les 30 secondes et exécute une requête de mise à jour. La requête calcule la capacité actuelle des ressources en soustrayant le nombre total de demandes validées associées à chaque ressource de la capacité totale de cette dernière. Cette mise à jour régulière des capacités garantit que les informations affichées dans la base de données sont constamment à jour, offrant ainsi une vision précise des ressources disponibles.

```
...  
  
UPDATE Ressources r  
SET Capacite_actuelle = r.Capacite_totale - (  
    SELECT COUNT(*) FROM Demandes_acces d  
    INNER JOIN Demandes_acces_ressources dr ON d.Id_demande = dr.Id_demande  
    WHERE dr.Id_ressource = r.Id_ressource  
    AND d.Statut = 'Validé')
```

(b) Évènement 2 de la base de données

Ces événements automatisés dans ma base de données contribuent grandement à l'efficacité et à la précision des opérations de gestion de mon projet. Ils permettent de maintenir les données à jour, d'optimiser les processus et de garantir un fonctionnement fluide de ma base de données.

## IV. Conception de l'interface de gestion (IG) (Omar Hussein)

### n. Fonctionnalités de l'interface de gestion

L'interface de gestion que j'ai développée pour le système Contrôle-Accès du tribunal de grande instance d'Evry est un élément clé de ce projet ambitieux. Permettez-moi de vous expliquer le processus de conception que j'ai suivi pour créer cette interface intuitive et fonctionnelle.

L'interface de gestion, comme la base de données, est hébergée sur XAMPP, une plateforme de développement web, dans le répertoire "htdocs". Cela permet d'accéder à l'interface à partir d'un navigateur web sur le même ordinateur où XAMPP est installé. Cette configuration offre une grande flexibilité et facilite la mise en place du système de contrôle d'accès.

Pour commencer, j'ai réalisé un croquis détaillé de l'interface que je souhaitais mettre en place. J'ai envisagé une barre de navigation fixe en haut de l'écran, offrant un accès facile à toutes les pages du système. Au centre de la page, j'ai prévu un encadré où serait affiché le contenu correspondant à la page sélectionnée. Ce choix de conception visait à assurer une navigation fluide et une expérience utilisateur agréable.

En ce qui concerne la réalisation de l'interface, j'ai adopté une approche progressive. J'ai commencé par développer toutes les fonctionnalités principales en utilisant HTML, PHP et SQL, sans me soucier de l'aspect visuel final. Cette approche m'a permis de me concentrer sur la logique et le bon fonctionnement du système. Une fois que toutes les fonctionnalités ont été implémentées et que le système était pleinement opérationnel, j'ai consacré du temps à améliorer l'aspect esthétique de l'interface. J'ai ajouté du CSS (Cascading Style Sheets) pour embellir les éléments visuels, harmoniser les couleurs, les polices et les espaces, et rendre l'interface plus agréable à utiliser.

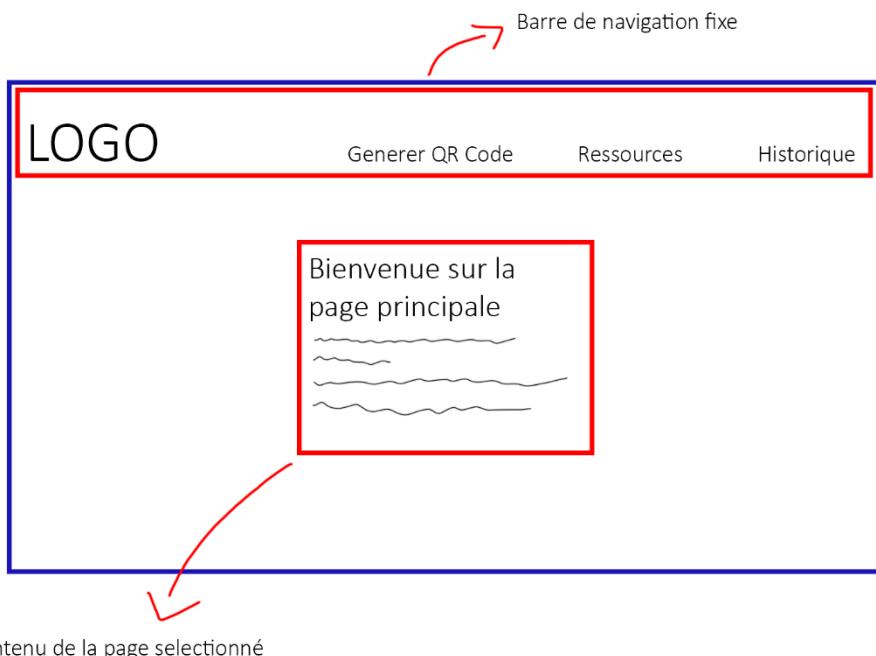


Figure 11. Croquis de l'interface graphique

En conclusion, l'interface de gestion que j'ai développée pour le système Contrôle-Accès du tribunal de grande instance d'Evry offre un large éventail de fonctionnalités visant à simplifier et à optimiser la gestion des accès aux ressources. Voici un récapitulatif des principales fonctionnalités de cette interface graphique :

- **Validation des demandes d'accès des utilisateurs** : L'interface permet de consulter et de valider les demandes d'accès soumises par les utilisateurs, offrant un contrôle précis sur les autorisations accordées.
- **Génération de codes d'accès sous forme de QR codes** : Grâce à cette fonctionnalité, l'interface permet de générer des codes d'accès temporaires ou permanents sous forme de QR codes, facilitant ainsi l'accès aux ressources autorisées.
- **Gestion de l'accès par reconnaissance faciale** : L'interface prend en charge la gestion de l'accès aux locaux sensibles grâce à la reconnaissance faciale, garantissant une sécurité renforcée.
- **Messagerie électronique** : Elle intègre également un système messagerie électronique qui facilite la communication lors de la génération des codes d'accès sous forme de QR codes. Grâce à cette fonctionnalité, les utilisateurs peuvent recevoir des notifications par e-mail contenant les informations relatives à leur demande d'accès.
- **Visualisation de l'état d'occupation des ressources** : L'interface fournit une vue globale et synthétique sur le niveau d'occupation des ressources, offrant ainsi une meilleure gestion et planification des accès.
- **Consultation de l'historique d'accès aux ressources** : Grâce à cette fonctionnalité, les gestionnaires peuvent accéder à l'historique des accès aux ressources en temps réel, permettant une traçabilité complète des activités.
- **Accès aux systèmes embarqués des serrures** : L'interface offre la possibilité d'accéder aux systèmes embarqués des serrures, permettant ainsi la gestion et le contrôle à distance de ces dispositifs.

En combinant ces différentes fonctionnalités, l'interface de gestion offre un environnement convivial et intuitif pour administrer et contrôler efficacement l'accès aux ressources du tribunal de grande instance d'Evry, contribuant ainsi à une gestion simplifiée, une sécurité renforcée et une meilleure utilisation des ressources disponibles.

### **o. Demandes d'accès**

La fonctionnalité de gestion des demandes d'accès permet de gérer efficacement les demandes de codes QR. J'ai identifié deux possibilités pour générer ces codes QR.

- **Génération des codes d'accès par le personnel d'accueil :** Le personnel d'accueil, depuis la page "Génération des codes d'accès", peut générer les codes QR en utilisant le formulaire présent sur la page. Le formulaire, développé en HTML et PHP, permet de collecter les informations nécessaires telles que le nom, le prénom, l'e-mail et la date d'expiration (si applicable) pour le code QR. Une fois que le personnel d'accueil remplit les champs requis et coche les ressources auxquelles donner accès, il clique sur le bouton "Générer QR Code".

Lorsque le bouton est cliqué, les données saisies dans le formulaire sont envoyées au serveur web, où le code PHP correspondant traite les informations. Le code PHP se connecte à la base de données hébergée sur XAMPP, dans le répertoire htdocs, en utilisant les informations d'identification appropriées. Il effectue les opérations nécessaires pour générer le code QR correspondant à la demande d'accès. Les informations de la demande, y compris les données personnelles de l'utilisateur, la date d'expiration et les ressources sélectionnées, sont enregistrées dans la base de données.

Une fois le code QR généré, il est affiché à l'écran dans l'interface de gestion pour que le personnel d'accueil puisse le visualiser. Le code QR est ensuite envoyé à l'utilisateur par e-mail, en utilisant une fonctionnalité que nous détaillerons dans une section ultérieure.

The screenshot shows a web application interface for generating QR codes. At the top, there's a navigation bar with links: 'Tribunal d'Evry', 'Générer un QR Code', 'Gestion des demandes', 'Gestion Visages', 'Historique accès', 'Ressources', and 'Accès aux serrures'. Below the navigation, a modal window titled 'Génération des codes d'accès' is displayed. It contains input fields for 'Nom' (Name), 'Prénom' (First Name), 'Email', and a date picker for 'Date d'expiration' (Expiration Date). There's also a section for 'Ressources' (Resources) with several checkboxes: 'Salle audience 1', 'Salle audience 2', 'Salle de délibération 1', 'Salle des pas perdus', 'Salle de conférence', and 'Salle de réunion 1'. At the bottom of the modal is a large blue button labeled 'GÉNÉRER QR CODE'.

Figure 12. (a) Page acces\_admin.php

- **Envoi de la demande par les utilisateurs :** Les utilisateurs ont également la possibilité de soumettre leur demande en utilisant une page dédiée. Sur cette page, ils saisissent leur nom, prénom, e-mail et éventuellement une date d'expiration pour le code QR. Après avoir rempli le formulaire, les utilisateurs cliquent sur le bouton "Envoyer la demande".

Lorsque le bouton est cliqué, les données du formulaire sont envoyées au serveur web, où le code PHP correspondant traite la demande. Tout comme dans le cas précédent, le code PHP se connecte à la base de données hébergée sur XAMPP et enregistre les informations de la demande. La demande est ensuite visible dans la page "Gestion des demandes", à laquelle le personnel d'accueil a accès. Ils peuvent consulter toutes les informations de la demande.

The screenshot shows a web page titled "Demande d'accès aux ressources" from the "Tribunal d'Evry" website. The page has a light gray background with a white form area. The form fields are as follows:

- Nom :
- Prénom :
- Email :
- Date d'expiration :  (with a calendar icon)
- Ressources :

  - Salle audience 1
  - Salle audience 2
  - Salle de délibération 1
  - Salle des pas perdus
  - Salle de conférence
  - Salle de réunion 1

A large blue button at the bottom right of the form area is labeled "ENVOYER LA DEMANDE".

(b) Page acces\_utilisateur.php

Le personnel d'accueil dispose de fonctionnalités pour gérer ces demandes dans l'interface de gestion :

1. Accepter une demande : Si le personnel d'accueil décide d'accepter une demande, ils peuvent marquer la demande comme "Acceptée" dans l'interface de gestion. Lorsqu'ils cliquent sur le bouton d'acceptation, un e-mail contenant le code QR correspondant est automatiquement envoyé à l'utilisateur à l'adresse e-mail fournie dans la demande. Le statut de la demande est alors mis à jour pour refléter son approbation.
2. Refuser une demande : Si une demande ne peut pas être acceptée, le personnel d'accueil a la possibilité de la refuser. En cliquant sur le bouton de refus, le statut de la demande est mis à jour pour indiquer qu'elle a été "Refusée". Dans ce cas, aucun e-mail contenant un code QR n'est envoyé à l'utilisateur.

**Tribunal d'Evry**

Generer un QR Code    Gestion des demandes    Gestion Visages    Historique accès    Ressources    Accès aux serrures

ID	Nom	Prénom	Email	Date de la demande	Date d'expiration	Statut	Ressources	QR Code	Action
4	Galara	Jaques	venerexbusiness@gmail.com	2023-05-29 17:41:04	2023-05-31 22:41:00	En attente	Salle audience 2 Salle de délibération 1 Salle de réunion 1	-	<a href="#">Accepter</a> <a href="#">Refuser</a>
3	Galara	Jaques	venerexbusiness@gmail.com	2023-05-29 17:40:42	2023-05-30 17:46:00	Validé	Salle audience 1 Salle de délibération 1 Salle des pas perdus Salle de réunion 1	<a href="#">Voir le QR Code</a>	-
2	Omar	Husslein	omarhusslein3366@gmail.com	2023-04-08 13:21:15	2023-04-08 15:19:00	Validé	Salle audience 1 Salle audience 2 Salle de délibération 1 Salle des pas perdus Salle de conférence Salle de réunion 1	<a href="#">Voir le QR Code</a>	-
1	Omar	Hussein	omarhussein3366@gmail.com	2023-04-08 13:19:55	2023-04-08 15:19:00	Validé	Salle audience 1 Salle audience 2 Salle de délibération 1	<a href="#">Voir le QR Code</a>	-

(c) Page gestion\_demandes\_utilisateur.php

Cette approche offre une solution complète pour la gestion des demandes d'accès, en permettant au personnel d'accueil de générer des codes QR pour les utilisateurs en remplissant un formulaire intuitif et en offrant aux utilisateurs la possibilité de soumettre leurs demandes directement. Elle garantit également la sécurité et la confidentialité des informations personnelles tout au long du processus.

p. Envoi des courriels électroniques

L'interface de gestion de l'application utilise le service de messagerie Sendinblue et la bibliothèque PHPMailer pour l'envoi des courriels électroniques.

- Sendinblue est un service de messagerie en ligne qui offre des fonctionnalités d'envoi d'e-mails avancées. Il agit en tant que serveur SMTP (Simple Mail Transfer Protocol) pour notre application, permettant ainsi l'envoi sécurisé et fiable des courriels. Sendinblue offre une infrastructure robuste et une interface conviviale pour la configuration des paramètres d'envoi d'e-mail, tels que les informations d'authentification SMTP, le type de sécurité et le port.
- PHPMailer, quant à lui, est une bibliothèque PHP populaire et puissante qui facilite l'envoi d'e-mails à partir d'applications PHP. Il fournit une interface simple et conviviale pour la création, la personnalisation et l'envoi d'e-mails. PHPMailer offre des fonctionnalités avancées telles que l'envoi d'e-mails en format HTML, la gestion des pièces jointes, la gestion des destinataires multiples et la configuration des paramètres SMTP. Il est largement utilisé dans le développement Web pour faciliter l'envoi d'e-mails personnalisés et automatisés.



**sendinblue**

Figure 13. (a) Logo de Sendinblue



(b) Logo de PHPMailer

Pour utiliser PHPMailer avec XAMPP, il est nécessaire d'activer l'extension OpenSSL dans la configuration de PHP. OpenSSL est une bibliothèque open-source utilisée pour les communications sécurisées via des protocoles tels que HTTPS, SMTPS, etc. L'activation de cette extension est essentielle pour permettre à PHPMailer d'établir une connexion sécurisée avec le serveur SMTP.

Voici les étapes pour activer l'extension OpenSSL sur XAMPP :

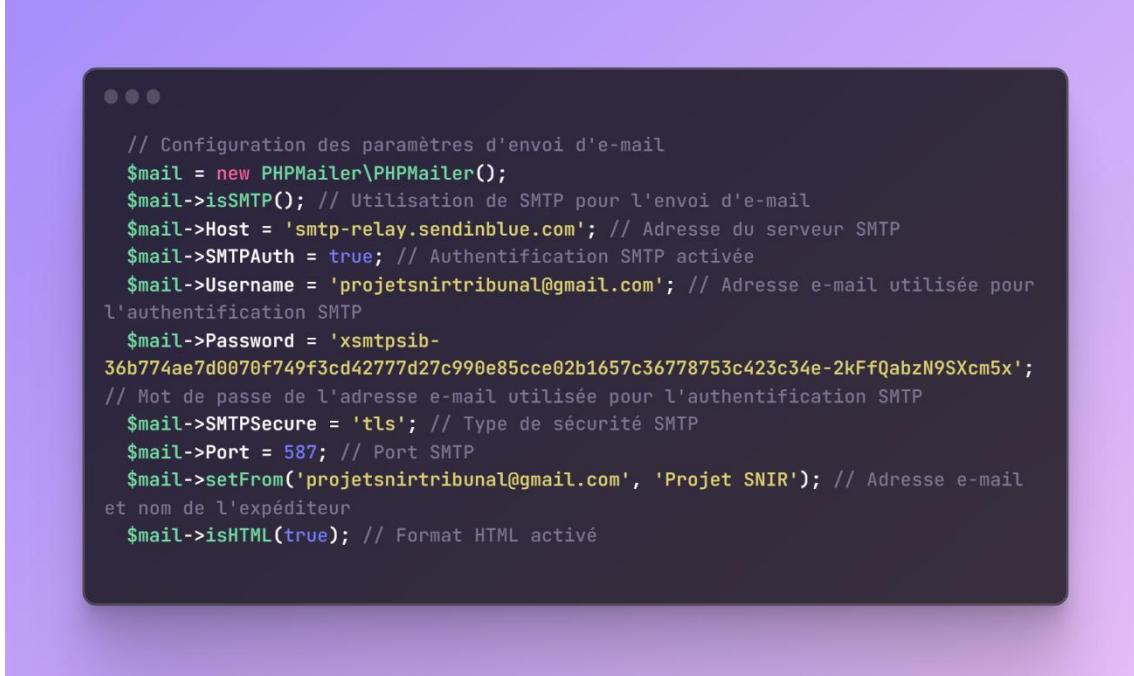
1. Localisez le dossier d'installation de XAMPP sur votre système. Par défaut, il est généralement installé dans le répertoire C:\xampp (sur Windows) ou /Applications/XAMPP (sur macOS)
2. Ouvrez le fichier php.ini avec un éditeur de texte
3. Recherchez la ligne suivante dans le fichier 'php.ini' :



4. Retirez le point-virgule (;) en début de ligne pour décommenter l'extension OpenSSL et enregistrez.

Voici les détails techniques de l'implémentation pour l'envoie pour l'envoi des courriels électroniques avec SendinBlue et PHPMailer :

1. Configuration des paramètres d'envoi d'e-mail : La configuration des paramètres d'envoi d'e-mail se fait dans le code PHP. La bibliothèque PHPMailer est utilisée pour se connecter au serveur SMTP de Sendinblue. Les paramètres d'authentification tels que l'adresse e-mail de l'expéditeur, le nom d'utilisateur, le mot de passe, le type de sécurité et le port SMTP sont spécifiés dans le code.



2. Envoi de l'e-mail de confirmation : Lorsque la demande d'accès est validée, un e-mail de confirmation est envoyé à l'adresse e-mail fournie. Le contenu de l'e-mail est personnalisé et contient un message de confirmation ainsi que le code QR en tant que pièce jointe (Le fonctionnement détaillé de la génération du QR Code sera exploré en détail dans une section ultérieure du rapport).



#### q. Génération des QR-Codes

Dans notre projet, j'ai utilisé la bibliothèque PHPQRCode pour générer les QR codes. Cette bibliothèque m'a permis de simplifier la génération des QR codes uniques et personnalisés associés à chaque demande d'accès.

- **PHP QR Code (phpqrcode)** est une bibliothèque open source largement utilisée pour la génération de codes QR en PHP. Cette bibliothèque offre des fonctionnalités puissantes et faciles à utiliser pour créer des codes QR personnalisés. Elle permet de générer des QR codes à partir de divers types de données, tels que du texte, des URL, des informations de contact, des numéros de téléphone, des adresses email, etc. Grâce à phpqrcode, il est possible de personnaliser l'apparence des QR codes en ajustant leur taille, leur niveau de correction d'erreurs, leurs couleurs et même en ajoutant un logo. Cette bibliothèque est bien documentée, dispose d'une communauté active et offre une solution fiable pour la génération de QR codes dans nos projets PHP

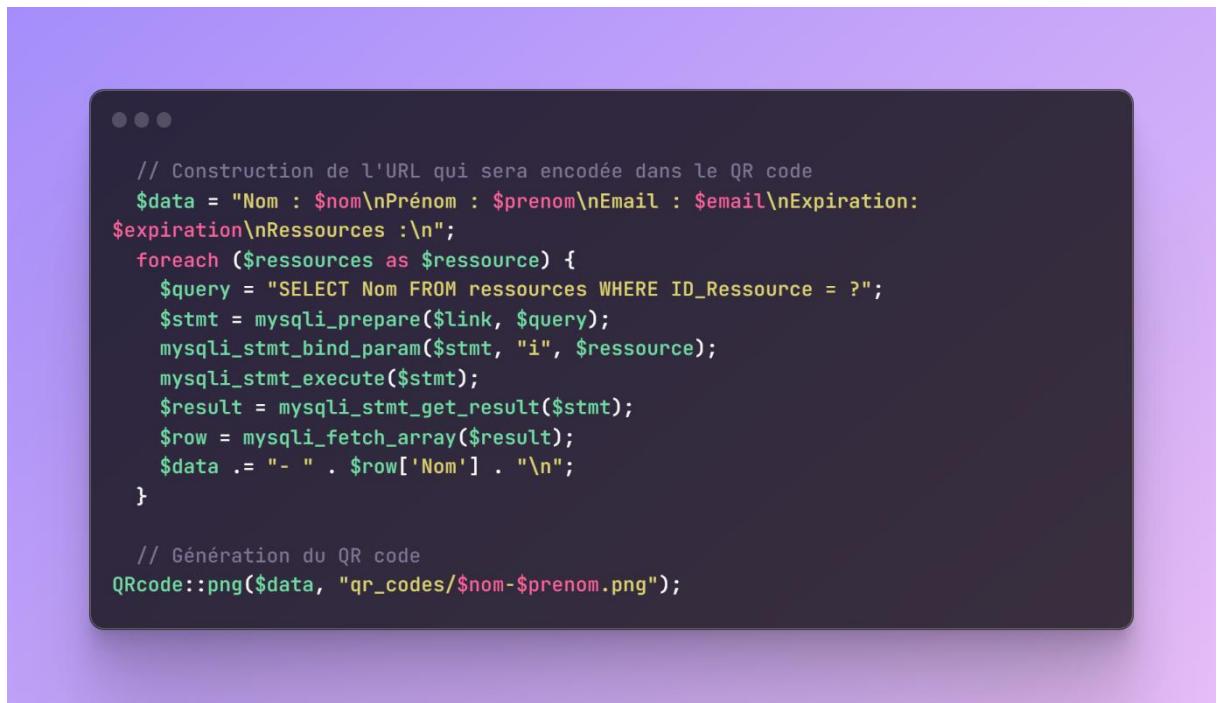
Pour pouvoir utiliser cette bibliothèque, il est nécessaire d'activer l'extension GD (Graphics Draw) dans le serveur web local tel que XAMPP. L'extension GD permet la manipulation d'images, y compris la création de codes QR à l'aide de PHPQRCode :

1. Ouvrez le fichier "php.ini" de votre installation XAMPP
2. Recherchez la ligne qui commence par ";extension=gd" et supprimez le point-virgule (;) au début de la ligne pour activer l'extension GD
3. Enregistrez le fichier "php.ini" et Redémarrez le serveur Apache de XAMPP



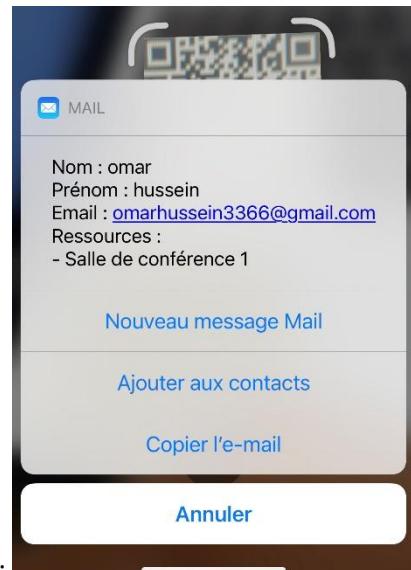
Figure 14. Logo de PHP Qr Code

Dans la version précédente de mon code, je construisais manuellement les données à encoder dans le QR code en utilisant une approche basée sur des chaînes de caractères. Je rassemblais les informations telles que le nom, le prénom, l'email, la date d'expiration et les ressources sélectionnées pour créer une chaîne de caractères représentant le contenu du QR code. Ensuite, j'utilisais la fonction QRcode::png() de la bibliothèque pour générer le QR code à partir de ces



```
// Construction de l'URL qui sera encodée dans le QR code
$data = "Nom : $nom\nPrénom : $prenom\nEmail : $email\nExpiration:
$expiration\nRessources :\n";
foreach ($ressources as $ressource) {
    $query = "SELECT Nom FROM ressources WHERE ID_Ressource = ?";
    $stmt = mysqli_prepare($link, $query);
    mysqli_stmt_bind_param($stmt, "i", $ressource);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    $row = mysqli_fetch_array($result);
    $data .= "- " . $row['Nom'] . "\n";
}
// Génération du QR code
QRcode::png($data, "qr_codes/$nom-$prenom.png");
```

données et je le stockais localement dans le répertoire "qr\_codes" de mon serveur XAMPP



Voici le rendu une fois le QR-Code généré et scanné :

Cependant, j'ai réalisé que cette approche comportait des risques de sécurité et de maintenabilité. Les informations personnelles, comme l'email, étaient directement incluses dans le contenu du QR code, ce qui pouvait compromettre la confidentialité des données sensibles. De plus, la construction manuelle des données à encoder était sujette à des erreurs de format et rendait la gestion des caractères spéciaux plus complexe.

Dans la version améliorée de mon code, j'ai donc décidé d'utiliser la bibliothèque PHP QR Code pour simplifier la génération des QR codes et renforcer la sécurité des données. Maintenant, je génère le contenu du QR code en utilisant une valeur aléatoire unique, générée par la fonction `bin2hex(random_bytes(18))`. Cette valeur est stockée dans la variable `$valeur_qr_code`.

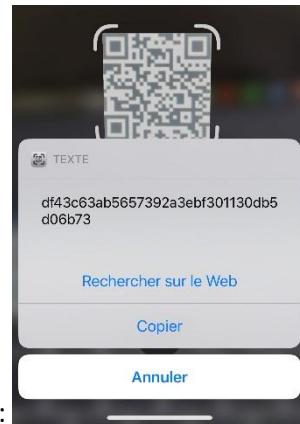
Ensuite, j'utilise la fonction `QRcode::png()` pour générer le QR code à partir de la valeur `$valeur_qr_code`. Le QR code ainsi créé est ensuite enregistré localement dans le répertoire "qr\_codes" de mon serveur XAMPP.

Il est également important de noter que j'enregistre le QR code généré dans la base de données. J'utilise la fonction `file_get_contents()` pour récupérer le contenu du fichier QR code généré, puis j'utilise la fonction `mysqli_real_escape_string()` pour échapper les caractères spéciaux dans le contenu du QR code. Enfin, j'exécute une requête SQL pour mettre à jour la table `Demandes_acces` avec le QR code correspondant à l'ID de la demande d'accès.

```
// Construction de l'URL qui sera encodée dans le QR code
$data = "$valeur_qr_code";

// Génération du QR code
QRcode::png($data, "qr_codes/$nom-$prenom.png");

// Enregistrement du QR code dans la base de données
$qr_code = file_get_contents("qr_codes/$nom-$prenom.png");
$qr_code = mysqli_real_escape_string($link, $qr_code);
$query = "UPDATE Demandes_acces SET QR_Code = '$qr_code' WHERE ID_Demande =
$id_demande";
mysqli_query($link, $query);
```



Voici le rendu une fois le QR-Code généré et scanné :

Cette approche améliorée de génération et de gestion des QR codes offre une meilleure sécurité des données et simplifie la manipulation des informations personnelles. Les QR codes sont stockés à la fois localement sur notre serveur XAMPP et dans la base de données, ce qui permet un accès sécurisé et une traçabilité des demandes d'accès.

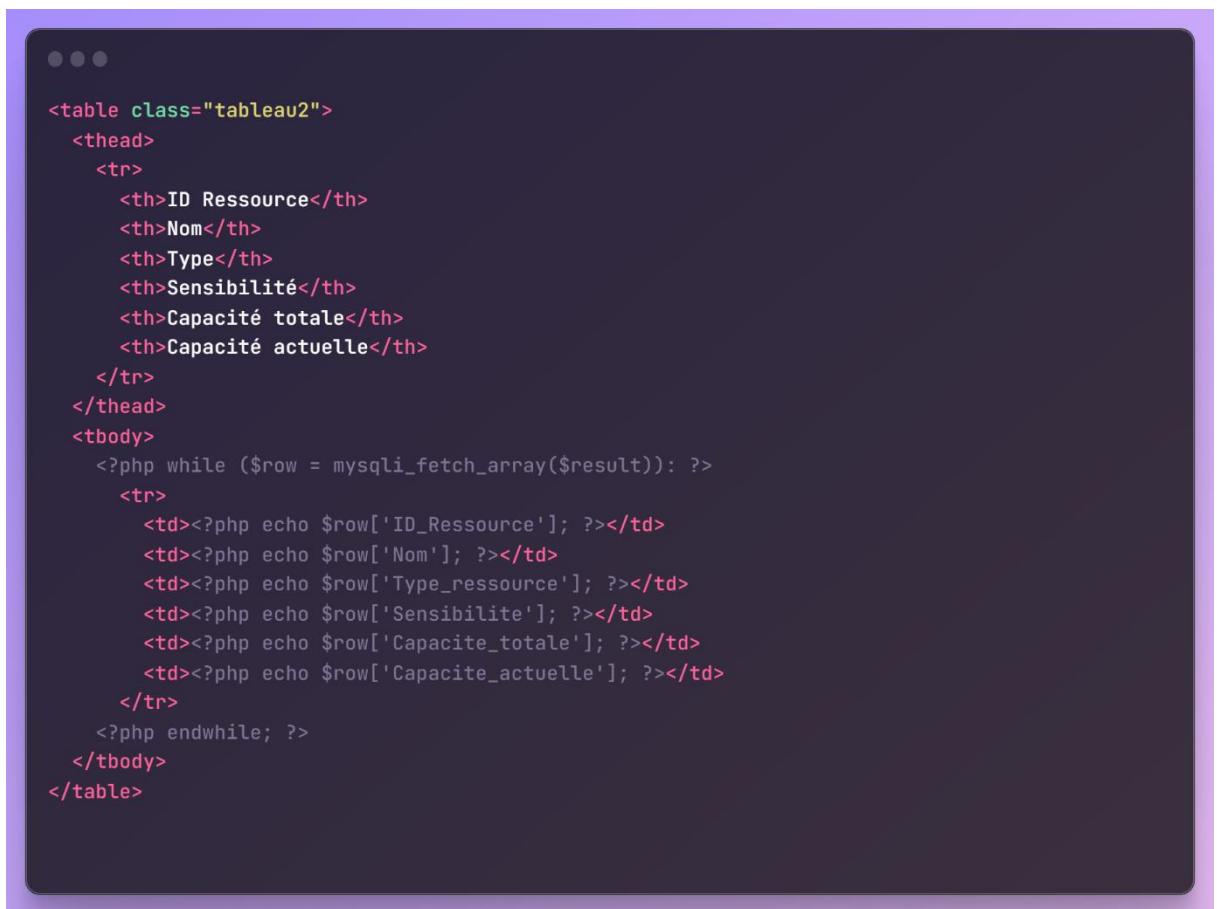
#### r. Visualisation de l'état d'occupation des ressources

Je souhaite maintenant vous présenter la fonctionnalité de visualisation de l'état d'occupation des ressources que j'ai développée dans le cadre de mon projet. J'ai créé une page simple qui affiche les données stockées dans la base de données. Grâce à un événement que j'ai précédemment expliqué dans le rapport, l'attribut "Capacité\_actuelle" des ressources se met automatiquement à jour, ce qui permet d'avoir une vue en temps réel de leur occupation.

ID Ressource	Nom	Type	Sensibilité	Capacité totale	Capacité actuelle
1	Salle audience 1	Salle audience	Sensible	50	50
2	Salle audience 2	Salle audience	Sensible	75	75
3	Salle de délibération 1	Salle de délibération	Sensible	20	20
4	Salle des pas perdus	Salle des pas perdus	Non Sensible	200	200
5	Salle de conférence	Salle de conférence	Sensible	100	100
6	Salle de réunion 1	Salle de réunion	Non Sensible	10	10

Figure 15. Page etat\_occupation\_ressources.php

Lorsque l'utilisateur accède à cette page, une connexion à la base de données est établie. Ensuite, une requête SQL est exécutée pour récupérer les données sur les ressources à partir de la table "Ressources". Une fois les données obtenues, elles sont affichées dans un tableau à l'aide de balises HTML. Chaque ligne du tableau représente une ressource et les colonnes affichent différentes informations telles que l'identifiant de la ressource, son nom, son type, sa sensibilité, sa capacité totale et sa capacité actuelle.



```
<table class="tableau2">
<thead>
<tr>
<th>ID Ressource</th>
<th>Nom</th>
<th>Type</th>
<th>Sensibilité</th>
<th>Capacité totale</th>
<th>Capacité actuelle</th>
</tr>
</thead>
<tbody>
<?php while ($row = mysqli_fetch_array($result)): ?>
<tr>
<td><?php echo $row['ID_Ressource']; ?></td>
<td><?php echo $row['Nom']; ?></td>
<td><?php echo $row['Type_ressource']; ?></td>
<td><?php echo $row['Sensibilite']; ?></td>
<td><?php echo $row['Capacite_totale']; ?></td>
<td><?php echo $row['Capacite_actuelle']; ?></td>
</tr>
<?php endwhile; ?>
</tbody>
</table>
```

### s. Historique d'accès aux ressources

La page "Historique d'accès" fonctionne de manière similaire à celle de l'état des ressources, mais avec une différence clé. Cette page récupère les données de la table "Historique\_acces" de la base de données. L'objectif est de fournir un historique détaillé des accès effectués dans le système.

ID_Acces	Nom Demande	Nom Visage	Ressource	Date accès
1		Omar Hussein	Salle audience 2	2023-04-08 13:14:39
2		Omar Hussein	Salle audience 2	2023-04-08 13:14:40
11	Omar Hussein		Salle des pas perdus	2023-04-08 13:33:47
12	Omar Hussein		Salle des pas perdus	2023-04-08 13:33:47

Figure 16. Page historique.php

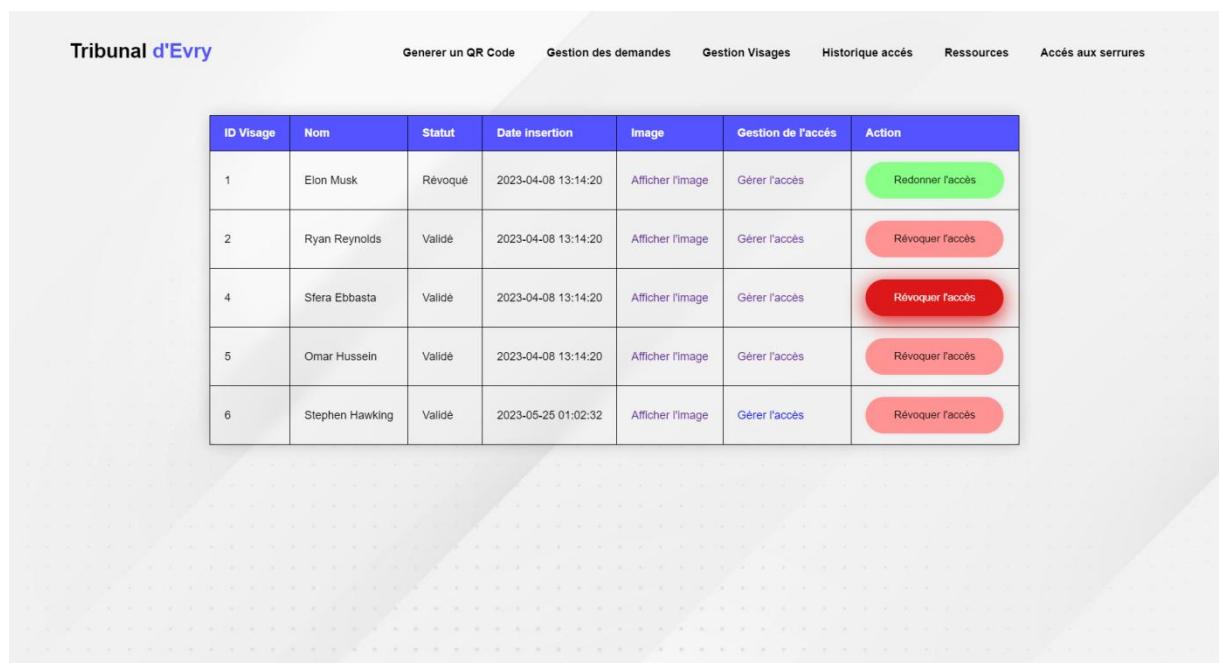
Pour mettre à jour cette page, les Raspberry Pi relié aux serrures sont utilisés pour enregistrer les connexions et les accès. Le fonctionnement de l'enregistrement des accès, sera détaillé ultérieurement dans les parties "Serrure QR-Code" et "Serrure reconnaissance faciale" du rapport.

- Pour la serrure QR-Code, lorsqu'un utilisateur présente son QR-Code généré à partir de la page "Générer un QR-Code", le Raspberry Pi connecté à la serrure scanne le code et envoie une requête à la base de données pour vérifier les informations associées au QR-Code. Si les données correspondent et que l'accès est autorisé, un enregistrement d'accès est créé dans la table "Historique\_acces".
- Quant à la serrure reconnaissance faciale, lorsque l'utilisateur se place devant le système de reconnaissance faciale connecté au Raspberry Pi, celui-ci capture l'image du visage et l'envoie pour un processus de reconnaissance. Si le visage est reconnu avec succès et que l'accès est autorisé, un enregistrement d'accès est également créé dans la table "Historique\_acces".

Dans les parties "Serrure QR-Code" et "Serrure reconnaissance faciale" du rapport, les détails techniques de chaque méthode seront abordés en détail. Cela inclura les algorithmes utilisés, les librairies ou technologies spécifiques, ainsi que les mesures de sécurité mises en place pour garantir l'intégrité du processus d'enregistrement des accès.

#### t. Gestion de l'accès par reconnaissance faciale

Sur la page de gestion de l'accès par reconnaissance faciale, j'ai développé une interface conviviale pour gérer les visages enregistrés dans la table "Visages" de la base de données. Cette page permet aux administrateurs d'afficher les informations associées à chaque visage, y compris l'ID du visage, le nom de la personne, le statut actuel, la date d'insertion et la possibilité de visualiser l'image du visage et d'effectuer une action.

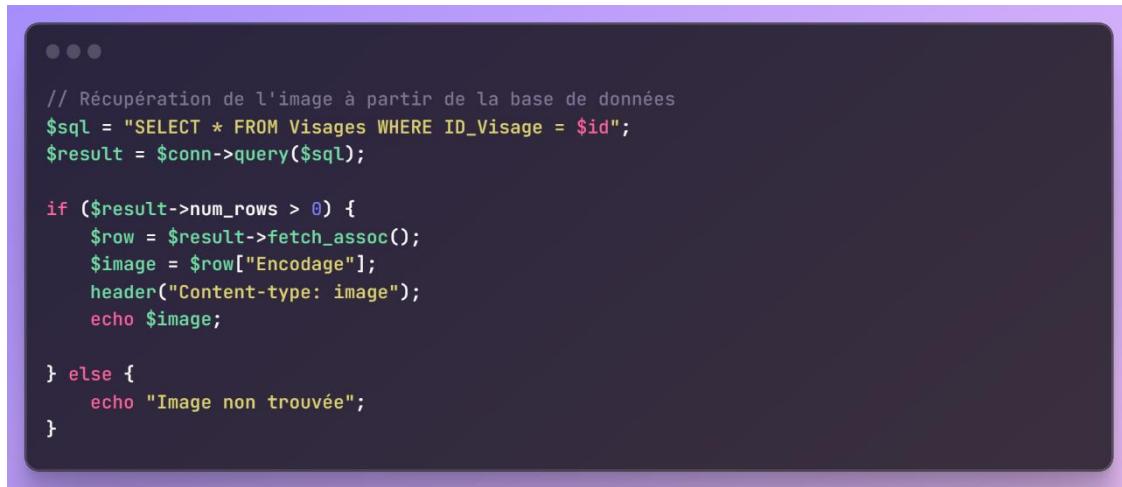


The screenshot shows a web application interface titled "Tribunal d'Evry". At the top, there is a navigation bar with links: "Générer un QR Code", "Gestion des demandes", "Gestion Visages", "Historique accès", "Ressources", and "Accès aux serrures". Below the navigation bar is a table titled "Visages" with the following data:

ID Visage	Nom	Statut	Date insertion	Image	Gestion de l'accès	Action
1	Elon Musk	Révoqué	2023-04-08 13:14:20	Afficher l'image	Gérer l'accès	<button>Redonner l'accès</button>
2	Ryan Reynolds	Validé	2023-04-08 13:14:20	Afficher l'image	Gérer l'accès	<button>Révoquer l'accès</button>
4	Sfera Ebbasta	Validé	2023-04-08 13:14:20	Afficher l'image	Gérer l'accès	<button>Révoquer l'accès</button>
5	Omar Hussein	Validé	2023-04-08 13:14:20	Afficher l'image	Gérer l'accès	<button>Révoquer l'accès</button>
6	Stephen Hawking	Validé	2023-05-25 01:02:32	Afficher l'image	Gérer l'accès	<button>Révoquer l'accès</button>

Figure 17. Page liste\_visages.php

En parcourant les enregistrements de visages, j'ai inclus un bouton permettant de visualiser l'image du visage, ce qui facilite l'identification visuelle de la personne concernée. Cela offre une visualisation effective du visage et aide les administrateurs à prendre des décisions éclairées.



```
// Récupération de l'image à partir de la base de données
$sql = "SELECT * FROM Visages WHERE ID_Visage = $id";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $image = $row["Encodage"];
    header("Content-type: image");
    echo $image;

} else {
    echo "Image non trouvée";
}
```

Ce code récupère une image à partir de la base de données en fonction d'un ID de visage spécifique. Si l'image est trouvée, elle est affichée dans le navigateur ; sinon, un message d'erreur est affiché.

Une autre colonne importante dans cette page est "Gestion de l'accès". Lorsque l'administrateur clique sur ce bouton, il est redirigé vers une autre page qui affiche toutes les ressources disponibles dans le système. Cette page offre la possibilité de sélectionner les ressources auxquelles la personne associée au visage enregistré doit avoir accès. L'administrateur peut simplement cocher les ressources souhaitées pour donner l'autorisation d'accès à ces ressources spécifiques.



The screenshot shows a web application interface for managing access rights. At the top, there is a navigation bar with links: "Tribunal d'Evry", "Générer un QR Code", "Gestion des demandes", "Gestion Visages", "Historique accès", "Ressources", and "Accès aux serrures". Below the navigation bar, there is a modal dialog box containing a list of resources with checkboxes. The resources listed are: Salle audience 1 (checked), Salle audience 2 (unchecked), Salle de délibération 1 (checked), Salle des pas perdus (unchecked), Salle de conférence (unchecked), and Salle de réunion 1 (unchecked). A blue "VALIDER" button is at the bottom of the dialog box.

Figure 18. Page gerer\_acces\_visages.php

Les ressources auxquelles le visage a déjà accès sont préalablement cochées. Lorsque l'utilisateur soumet le formulaire en cliquant sur le bouton "Valider", les ressources sélectionnées sont enregistrées dans la base de données.

Le code de la page commence par établir une connexion à la base de données. Ensuite, il récupère l'ID du visage sélectionné à partir de la page précédente et le nom correspondant à cet ID. Ensuite, il récupère toutes les ressources disponibles et les ressources auxquelles le visage a déjà accès. Le code génère ensuite le formulaire avec des cases à cocher pour chaque ressource, en cochant les cases appropriées en fonction de l'accès actuel du visage. Lorsque le formulaire est soumis, les ressources sélectionnées sont envoyées à la page "enregistrer\_acces\_visages.php" pour être enregistrées dans la base de données.

Enfin, la dernière colonne est intitulée "Action". Cette colonne affiche des boutons d'action qui dépendent du statut actuel du visage enregistré. Si le statut est "Validé", un bouton est affiché pour révoquer l'accès, permettant ainsi à l'administrateur de retirer l'autorisation d'accès pour ce visage spécifique. D'un autre côté, si le statut est "Révoqué", un bouton est affiché pour redonner l'accès, donnant ainsi la possibilité à l'administrateur de rétablir l'autorisation d'accès pour ce visage.

#### u. Accès aux serrures

Sur la page "Accès aux serrures", vous trouverez toutes les informations nécessaires pour vous connecter aux systèmes embarqués des serrures du tribunal. Cette page a été conçue pour faciliter l'accès et la gestion des serrures en fournissant des détails importants et les identifiants de connexion requis.

The screenshot shows a web page titled 'Accès Serrures'. At the top, there is a navigation bar with links: 'Tribunal d'Evry', 'Générer un QR Code', 'Gestion des demandes', 'Gestion Visages', 'Historique accès', 'Ressources', and 'Accès aux serrures'. The main content area has a title 'Accès Serrures' and a paragraph explaining that connection to embedded systems is available via SSH and VNC. It notes that identifiers are the same for both, except that for VNC, one must be on the same network. Below this, there are two sections: 'Identifiants de connexion:' and 'Sécurité'. Under 'Identifiants de connexion:', it lists: IP: 10.3.14.4, Nom utilisateur: snir, Mot de passe: administ. Under 'Sécurité', it lists: IP: 10.194.176.83, Nom utilisateur: pi, Mot de passe: administ.

Figure 19. Page acces\_serrures.html

- La première solution est l'utilisation de VNC (Virtual Network Computing), qui permet d'accéder aux interfaces graphiques des systèmes embarqués des serrures. L'avantage de VNC est qu'il offre une expérience utilisateur conviviale et intuitive. En utilisant VNC, vous pourrez interagir visuellement avec les serrures, ce qui facilite la configuration et la gestion des paramètres.
- La deuxième solution est l'utilisation de SSH (Secure Shell), un protocole sécurisé pour établir une connexion à distance avec les serrures. SSH offre des mesures de sécurité avancées, notamment le cryptage des données et l'authentification à clé publique/privée, ce qui garantit la confidentialité et l'intégrité des communications. L'utilisation de SSH vous permettra de gérer les serrures en ligne de commande, en exécutant des commandes et des scripts pour effectuer diverses opérations.

En choisissant ces deux méthodes de connexion, j'ai cherché à offrir une flexibilité maximale aux utilisateurs. La connexion par VNC permet une interaction graphique directe, tandis que la connexion par SSH offre un accès sécurisé en ligne de commande. Ainsi, l'utilisateur peut choisir la méthode qui convient le mieux à ses besoins et préférences.

#### v. Conception visuelle : L'optimisation de l'interface grâce au CSS

Le CSS joue un rôle crucial dans la conception de l'interface en définissant l'apparence visuelle des éléments. Il permet de contrôler la taille, la couleur, la typographie, les marges, les espacements, les bordures, les arrière-plans, les effets de transition et bien plus encore. En combinant différentes propriétés CSS, il est possible de créer des designs uniques et attrayants, qui reflètent l'identité visuelle d'une entreprise ou d'une marque. Grâce au CSS, il est également possible de structurer les pages web de manière cohérente et organisée.

Ci-dessus, je vous présente l'avant et l'après de l'implémentation du CSS dans la page de génération des codes d'accès, mettant en évidence la différence frappante dans l'apparence et le style des pages web avant et après l'utilisation du CSS.

##### Tribunal d'Evry

- Générer un QR Code
- Gestion des demandes
- Gestion Visages
- Historique accès
- Ressources
- Accès aux services

##### Génération des codes d'accès

Nom :

Prénom :

Email :

Date d'expiration :

Ressources :

- Salle audience 1
- Salle audience 2
- Salle de délibération 1
- Salle des pas perdus
- Salle de conférence
- Salle de réunion 1

Figure 20. (a) Page de génération des codes d'accès avant CSS

(b) Page de génération des codes d'accès après CSS

Avant l'implémentation du CSS, les pages web étaient dépourvues de style et d'esthétique, mais grâce à son utilisation, l'interface graphique a été transformée, offrant une expérience visuelle attrayante et professionnelle.

Dans du développement CSS, j'ai opté pour l'utilisation de l'outil uiverse.io pour la partie CSS. Je souhaite mettre en valeur ce choix, car il s'est avéré être une ressource extrêmement précieuse dans le processus de conception et de stylisation de l'interface graphique.

Uiverse.io m'a permis d'accéder à une vaste collection d'éléments d'interface utilisateur préconstruits, tels que des boutons, des formulaires, des cartes et bien d'autres encore. Ces éléments étaient non seulement visuellement attrayants, mais ils étaient également faciles à intégrer dans mon projet. Le fait qu'ils soient open source et gratuits à utiliser a été un avantage majeur, me permettant d'économiser du temps et des ressources.

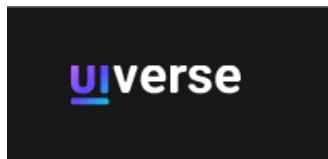


Figure 21. Logo uiverse.io

Ce choix m'a permis d'accélérer considérablement le processus de développement, en me permettant de me concentrer sur d'autres aspects cruciaux de mon projet. L'intégration fluide des éléments de l'interface utilisateur fournis par uiverse.io a contribué à créer une expérience utilisateur attrayante et très professionnelle.

Voici l'explication des parties les plus importantes du code CSS :

- **Sélection globale (\*) :**

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Cette partie du code applique des styles à tous les éléments de la page. Les propriétés margin: 0; et padding: 0; suppriment les marges et les espaces de remplissage par défaut des éléments, assurant une mise en page cohérente. La propriété box-sizing: border-box; garantit que la largeur et la hauteur des éléments incluent également leur bordure et leur rembourrage.

- **Styling des formulaires :**

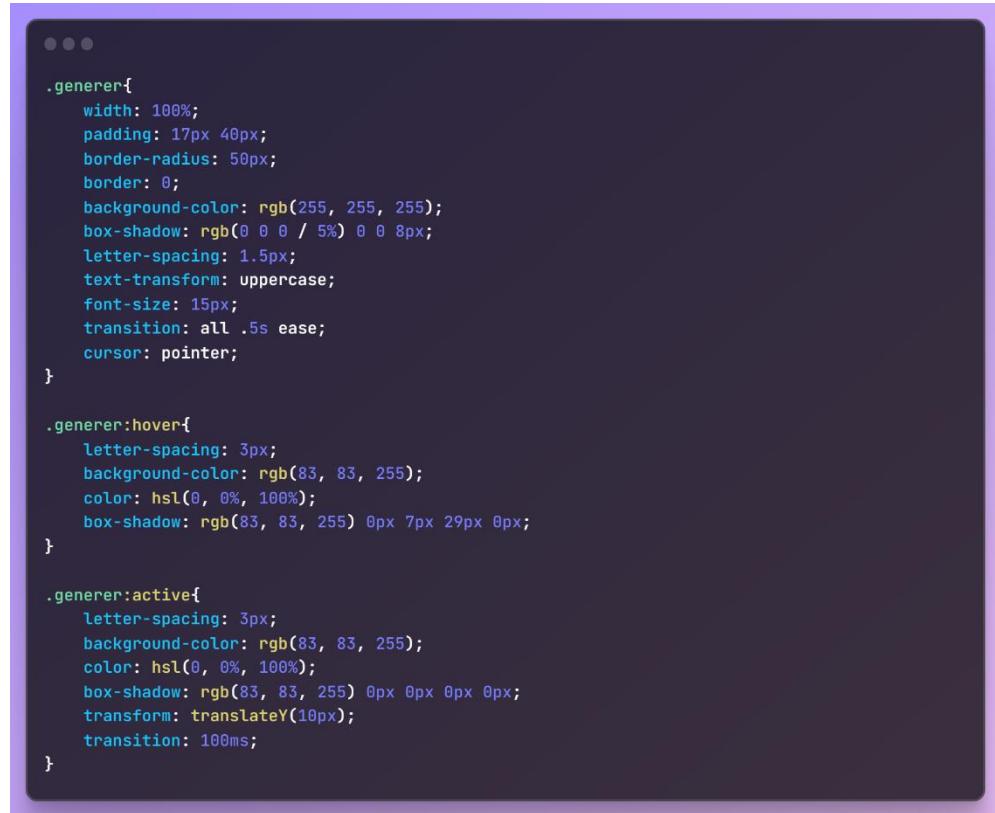
```
.formulaire {
  width: 500px;
  padding: 20px;
  border-radius: 12px;
  backdrop-filter: blur(8px);
  box-shadow: 0 0px 40px #23232333;
  transition: 0.2s ease-out;
  margin: 4% 0% 0% 35%;
}

.formulaire:hover {
  box-shadow: 0 0 20px #23232333;
}

.formulaire h1 {
  font-size: 36px;
  margin-bottom: 25px;
}
```

Cette partie du code stylise un formulaire spécifique avec la classe .formulaire. Il définit la largeur, le rembourrage et le rayon de bordure, créant une boîte bien définie pour le formulaire. La propriété backdrop-filter: blur(8px); ajoute un flou d'arrière-plan au formulaire, lui donnant une apparence plus esthétique. Le box-shadow crée une ombre légère autour du formulaire. Lorsque vous survolez le formulaire, l'ombre est réduite grâce à la pseudo-classe :hover. De plus, la taille de la police et les marges pour le titre <h1> à l'intérieur du formulaire sont spécifiées.

- **Bouton de validation :**



```
.generer{
    width: 100%;
    padding: 17px 40px;
    border-radius: 50px;
    border: 0;
    background-color: #fff;
    box-shadow: #000 / 5% 0 0 8px;
    letter-spacing: 1.5px;
    text-transform: uppercase;
    font-size: 15px;
    transition: all .5s ease;
    cursor: pointer;
}

.generer:hover{
    letter-spacing: 3px;
    background-color: #8383ff;
    color: hsl(0, 0%, 100%);
    box-shadow: #8383ff 0px 7px 29px 0px;
}

.generer:active{
    letter-spacing: 3px;
    background-color: #8383ff;
    color: hsl(0, 0%, 100%);
    box-shadow: #8383ff 0px 0px 0px 0px;
    transform: translateY(10px);
    transition: 100ms;
}
```

Ce code spécifie que le bouton doit avoir une largeur de 100% de son conteneur parent, un padding de 17 pixels en haut et en bas, ainsi que de 40 pixels à gauche et à droite. Les bordures sont supprimées et les coins du bouton sont arrondis avec un rayon de 50 pixels, lui donnant une forme circulaire. La couleur de fond est définie comme blanc, avec une légère ombre pour créer un effet en relief. L'espacement entre les lettres est augmenté de 1,5 pixels pour une meilleure lisibilité. Lorsque le curseur survole le bouton, la couleur de fond change en bleu avec une transition en douceur, la letter-spacing est augmentée à 3 pixels et une nouvelle ombre est ajoutée pour renforcer l'effet de surbrillance. Lorsque le bouton est activé, il est enfoncé de 10 pixels vers le bas avec une transition plus rapide.

## V. La serrure QR-code (Adama Sylla)

### w. Présentation de la serrure QR-code

Dans le cadre du projet de développement d'une solution de contrôle d'accès pour le tribunal de grande instance d'Evry, ma mission était de concevoir et développer la serrure QR code, une composante essentielle du système global. Cette serrure QR code offre une méthode sécurisée et fiable pour gérer les accès aux ressources du tribunal à distance. Dans cette section, je vais vous présenter en détail la conception et le fonctionnement de la serrure QR code, en mettant l'accent sur les mesures de sécurité mises en place pour protéger les utilisateurs et les biens.

La sécurité revêt une importance primordiale dans tout système de contrôle d'accès, et notre solution de serrure QR code est spécifiquement conçue pour répondre à ces exigences. En remplaçant les clés traditionnelles par des QR codes, nous avons pris des mesures significatives pour renforcer la sécurité des accès au tribunal de grande instance d'Evry.

Tout d'abord, le processus d'authentification joue un rôle essentiel dans la sécurisation des accès. La serrure QR code constitue un mécanisme d'authentification avancé qui permet de contrôler les accès de manière précise et sécurisée. Lorsqu'un utilisateur présente un QR code devant la serrure, celle-ci lit et décrypte instantanément les informations contenues dans le code. Une vérification d'authenticité est alors effectuée pour s'assurer que le QR code est valide et correspond à un utilisateur autorisé.

Une fois le QR code authentifié, le système réagit en conséquence. Si le QR code est valide, une LED verte s'allume pour indiquer la réussite de l'authentification, et la serrure s'ouvre automatiquement, permettant à l'utilisateur d'accéder aux ressources souhaitées. Cette réaction rapide et positive crée une expérience fluide pour l'utilisateur légitime.

En revanche, si le QR code est invalide ou non reconnu, le système détecte l'anomalie et réagit en conséquence. Une LED rouge s'allume pour signaler l'échec de l'authentification, et la serrure reste fermée, empêchant l'accès non autorisé. Cette réponse immédiate et claire garantit que seules les personnes autorisées peuvent obtenir l'accès aux ressources protégées.

Cette vérification garantit que seules les personnes autorisées obtiennent l'accès aux ressources sensibles du tribunal. Cette méthode d'authentification robuste constitue une barrière efficace contre les tentatives d'accès non autorisées.

De plus, la gestion centralisée des accès offre une visibilité et un contrôle complets sur les autorisations d'accès. Les QR codes peuvent être générés de manière temporaire ou permanente, permettant ainsi une gestion précise des périodes d'accès autorisées. Cette approche réduit les risques potentiels liés aux accès dérogatoires et offre une gestion granulaire des autorisations.

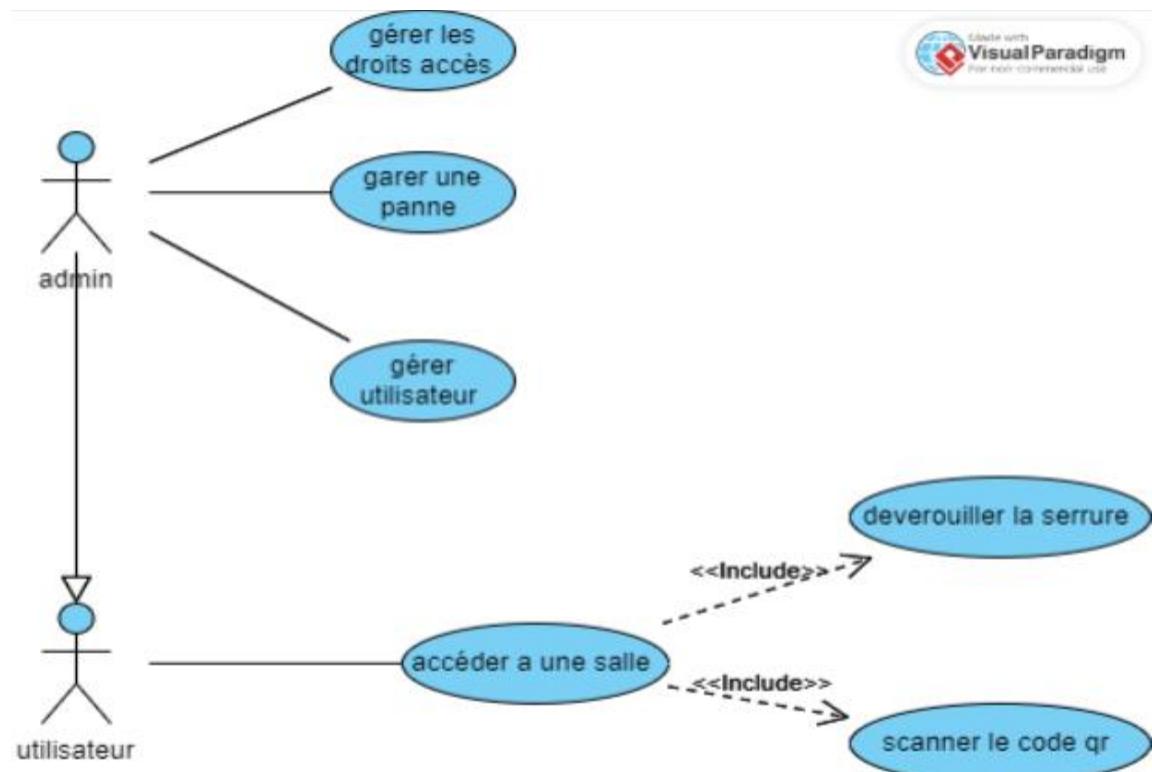
Par ailleurs, notre solution de serrure QR code enregistre toutes les informations relatives aux accès. Chaque fois qu'un QR code est scanné et qu'un utilisateur obtient l'accès, la date et l'heure de l'accès sont enregistrées. Cela permet de maintenir un historique complet des accès en temps réel, facilitant ainsi les audits de sécurité et offrant une traçabilité précise en cas d'incident ou d'enquête.

En intégrant la technologie QR code dans notre solution de contrôle d'accès, nous avons simplifié la gestion des clés et réduit les risques associés aux pertes ou aux utilisations abusives. Les QR codes

sont faciles à générer, à distribuer et à révoquer, offrant ainsi une flexibilité accrue et un contrôle précis sur les accès.

En résumé, la serrure QR code est un élément clé du projet de contrôle d'accès du tribunal de grande instance d'Evry. Son fonctionnement basé sur la lecture et la vérification des QR codes, associé à une réaction instantanée du système à travers l'allumage des LED verte ou rouge, garantit la sécurité et l'efficacité du contrôle d'accès. Au cours de cette présentation, nous explorerons plus en détail les aspects techniques de la serrure QR code, en mettant en évidence ses avantages en termes de sécurité et de gestion des accès.

Voici le diagramme des cas utilisation de la serrure QR-code :



x. Étude préliminaire - QR code :

Dans le cadre de notre projet de contrôle d'accès pour le tribunal de grande instance d'Evry, une partie essentielle était la mise en place d'un système de gestion des accès basé sur des QR codes. Avant de développer cette solution, j'ai réalisé une étude préliminaire approfondie pour évaluer sa faisabilité et son adéquation aux besoins du tribunal. Voici un aperçu des conclusions de cette étude :

Description des besoins du tribunal en matière de contrôle d'accès basé sur des QR codes :

Le tribunal de grande instance d'Evry avait exprimé le besoin d'un système de contrôle d'accès qui soit à la fois sécurisé, efficace et facile à utiliser. Les principales exigences identifiées pour la solution basée sur les QR codes étaient les suivantes :

Gestion des accès à distance : Le tribunal souhaitait pouvoir gérer les accès aux ressources de manière sécurisée à distance, en fournissant des QR codes temporaires ou permanents aux utilisateurs autorisés. Cela permettrait de simplifier la gestion des accès et d'améliorer la flexibilité pour les différentes parties prenantes du tribunal.

Authentification des utilisateurs : Il était primordial de garantir que seules les personnes autorisées puissent accéder aux ressources sensibles du tribunal. Ainsi, le système devait inclure une méthode d'authentification robuste pour vérifier l'identité des utilisateurs avant de leur accorder l'accès.

Traçabilité des accès : Le tribunal avait besoin d'une fonctionnalité de suivi et de traçabilité des accès effectués à l'aide des QR codes. Cela permettrait de disposer d'un historique complet des accès, incluant les dates et les heures, afin de faciliter les audits de sécurité et d'assurer une surveillance appropriée.

Analyse des solutions existantes et justification du choix de développer une solution personnalisée basée sur les QR codes :

Après avoir examiné les solutions de contrôle d'accès existantes basées sur les QR codes, j'ai constaté qu'elles ne répondaient pas pleinement aux besoins spécifiques du tribunal de grande instance d'Evry. Les solutions disponibles étaient soit trop génériques et ne pouvaient pas être adaptées de manière optimale, soit ne proposaient pas les fonctionnalités de sécurité et de traçabilité requises.

C'est pourquoi nous avons pris la décision de développer une solution personnalisée basée sur les QR codes. En développant notre propre système, nous pouvions répondre précisément aux exigences du tribunal et garantir une intégration optimale avec les autres composants du système de contrôle d'accès.

Le développement de la solution de contrôle d'accès basée sur les QR codes impliquait certaines contraintes spécifiques, notamment au niveau matériel, logiciel et des technologies utilisées. Voici un aperçu des principales contraintes identifiées :

Dispositif de lecture des QR codes : Nous devions sélectionner et intégrer un dispositif de lecture des QR codes capable de scanner et de décoder les codes de manière précise et fiable. Des critères tels que la vitesse de lecture, la compatibilité avec notre architecture logicielle et la facilité d'intégration ont été pris en compte lors du choix de ce dispositif.

Sécurité des QR codes : La solution devait inclure des mesures de sécurité robustes pour prévenir la falsification ou la reproduction non autorisée des QR codes. Cela impliquait l'utilisation de techniques de cryptage et d'encodage appropriées, ainsi que des algorithmes de génération des QR codes sécurisés.

Intégration avec le système global de contrôle d'accès : La solution QR code devait être intégrée de manière transparente avec les autres composants du système de contrôle d'accès, tels que la gestion des autorisations, le suivi des accès. Une coordination étroite avec les autres parties du projet était nécessaire pour assurer une compatibilité et une interopérabilité efficaces.

L'une des contraintes majeures identifiées lors de l'étude préliminaire était l'utilisation obligatoire d'une gâche électrique dans la solution de contrôle d'accès basée sur les QR codes. Cette contrainte était stipulée dans le cahier des charges et était motivée par les raisons suivantes :

Niveau de sécurité accru : L'utilisation d'une gâche électrique permet de renforcer le niveau de sécurité du système de contrôle d'accès. Contrairement aux serrures mécaniques traditionnelles, la gâche électrique offre une capacité de verrouillage et de déverrouillage à distance, permettant ainsi un contrôle précis des accès. Elle offre également la possibilité de désactiver rapidement et facilement l'accès en cas de besoin.

Intégration avec le système global : La gâche électrique est compatible avec les autres composants du système de contrôle d'accès, tels que le dispositif de lecture des QR codes et les mécanismes d'authentification des utilisateurs. Elle peut être facilement intégrée dans l'architecture du système pour assurer une gestion cohérente et unifiée des accès.

Conformité aux exigences du tribunal : Le tribunal de grande instance d'Evry avait spécifiquement demandé l'utilisation d'une gâche électrique pour la gestion des accès. Cette demande était motivée par la nécessité d'améliorer la sécurité des ressources sensibles du tribunal et de simplifier la gestion des accès pour le personnel autorisé.

En plus des contraintes techniques et fonctionnelles, une contrainte budgétaire a également été identifiée lors de l'étude préliminaire de la solution de contrôle d'accès basée sur les QR codes. Le budget alloué pour le développement de cette solution était limité à 200 euros. Cette contrainte budgétaire a imposé des défis supplémentaires dans la conception et la mise en œuvre du système. Voici comment nous avons pris en compte cette contrainte :

Sélection des composants : j'ai dû soigneusement sélectionner les composants matériels et logiciels nécessaires pour respecter le budget alloué. Nous avons recherché des options abordables et fiables qui répondaient aux spécifications techniques et fonctionnelles tout en respectant la limite budgétaire.

Optimisation des ressources existantes : j'ai cherché à utiliser au maximum les ressources et les équipements déjà disponibles afin de réduire les coûts. Cela comprenait l'utilisation d'une carte Raspberry Pi existante et d'une caméra Pi plutôt que d'investir dans de nouveaux équipements coûteux.

Développement de solutions économiques : j'ai adopté une approche économique dans le développement de la solution QR code en utilisant des logiciels open-source gratuits et en recherchant des alternatives abordables pour les composants matériels nécessaires. Cela nous a permis de minimiser les coûts de développement tout en maintenant les fonctionnalités et les performances requises.

En conclusion, l'étude préliminaire pour la mise en place d'une solution de contrôle d'accès basée sur les QR codes pour le tribunal de grande instance d'Evry a permis de mettre en évidence plusieurs éléments clés.

Tout d'abord, il a été identifié que le tribunal avait des besoins spécifiques en matière de contrôle d'accès, notamment la gestion sécurisée des accès aux ressources sensibles, la simplification de la gestion des clés, ainsi que l'augmentation globale de la sécurité des utilisateurs et des biens.

En analysant les solutions existantes, il a été décidé de développer une solution personnalisée basée sur les QR codes. Cette solution permettra la gestion des accès à distance, l'authentification des utilisateurs, la sécurisation des QR codes et la traçabilité des accès en temps réel. Le choix d'une solution personnalisée était justifié par sa capacité à répondre spécifiquement aux besoins du tribunal, offrant ainsi une meilleure adéquation fonctionnelle.

Des contraintes de développement ont également été prises en compte, notamment l'utilisation obligatoire d'une gâche électrique pour renforcer la sécurité du système de contrôle d'accès. Cette contrainte a été spécifiée dans le cahier des charges et était essentielle pour assurer un contrôle précis et sécurisé des accès.

De plus, une contrainte budgétaire de 200 euros a été identifiée. Cela a nécessité une sélection minutieuse des composants, une optimisation des ressources existantes et le développement de solutions économiques pour respecter la limite budgétaire tout en maintenant les fonctionnalités requises.

En résumé, l'étude préliminaire a confirmé la faisabilité et la pertinence de la solution de contrôle d'accès basée sur les QR codes pour le tribunal de grande instance d'Evry. Cette solution personnalisée répondra aux besoins spécifiques du tribunal, offrant une gestion sécurisée des accès, une simplification de la gestion des clés et une augmentation de la sécurité. Les contraintes de développement, telles que l'utilisation d'une gâche électrique et le respect du budget, ont été prises en compte pour garantir la réussite de ce projet.

#### y. Le matériel utilisé

Dans le cadre de notre projet de développement d'une solution de contrôle d'accès pour le tribunal de grande instance d'Evry, plusieurs matériaux essentiels ont été utilisés pour la conception de la serrure QR-Code. Ces matériaux ont été soigneusement sélectionnés en fonction des besoins du projet, des contraintes imposées par le cahier des charges et de leur compatibilité avec la carte Raspberry Pi et la caméra Pi. Cette section présente un inventaire détaillé des matériaux utilisés, mettant en évidence leur importance et leur contribution à la réalisation de notre solution de contrôle d'accès sécurisée.

La gâche électrique est un composant essentiel du système de contrôle d'accès. Elle est utilisée pour verrouiller et déverrouiller mécaniquement la porte en fonction des autorisations d'accès. La présence de la gâche électrique était une contrainte imposée par le cahier des charges, afin de garantir un contrôle sécurisé des accès, la gâche électrique se déverrouille, permettant ainsi l'ouverture de la porte. Elle est contrôlée par la carte Raspberry Pi.



Carte Raspberry Pi 3: La carte Raspberry Pi est le cœur du votre système. C'est un ordinateur monocarte compact et puissant qui offre une puissance de calcul suffisante pour exécuter les tâches nécessaires, telles que le décodage des QR codes, l'authentification des utilisateurs et le contrôle de la serrure via le module relais. La carte Raspberry Pi a été choisie pour sa flexibilité et sa compatibilité avec les ressources logicielles utilisées.



Module relais : Le module relais est un dispositif qui permet de contrôler des appareils électriques en fonction de signaux électriques. Dans mon projet, le module relais est utilisé pour contrôler l'ouverture et la fermeture de la gâche électrique. Lorsque la carte Raspberry Pi envoie un signal de commande au module relais, celui-ci réagit en ouvrant ou fermant le circuit électrique. Lorsque le circuit est ouvert, la gâche électrique est désactivée, ce qui signifie que la serrure reste fermée. En revanche, lorsque le circuit est fermé, la gâche électrique est activée, permettant ainsi l'ouverture de la serrure



Générateur : le générateur est utilisé comme une source d'alimentation séparée et indépendante pour la gâche électrique. Il fournit une puissance électrique adéquate pour permettre le fonctionnement correct de la gâche électrique, qui nécessite une alimentation plus élevée que celle fournie par la carte Raspberry Pi.



LED (rouge et verte) : Les LED sont de petites diodes lumineuses qui peuvent émettre de la lumière de différentes couleurs. Dans votre projet, une LED rouge et une LED verte sont utilisées pour signaler visuellement l'état de la serrure. Lorsque la serrure est verrouillée, la LED rouge s'allume, indiquant que l'accès est refusé. Lorsque la serrure est déverrouillée, la LED verte s'allume, indiquant que l'accès est autorisé.

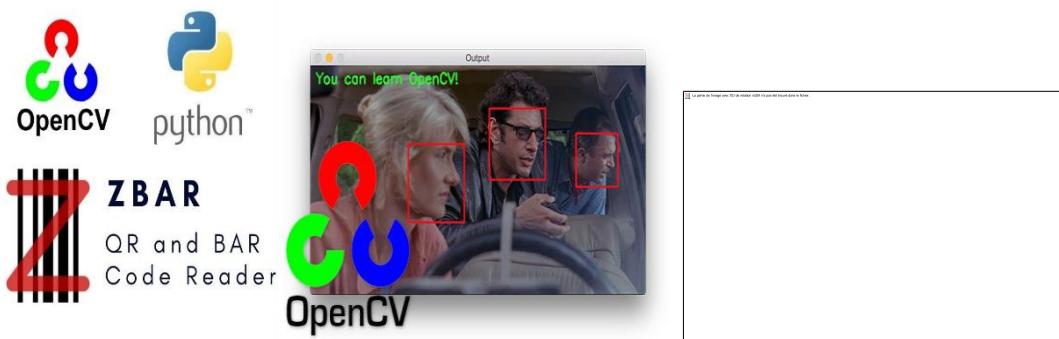


## z. Choix technologiques effectués

Après avoir achevé l'étude préliminaire et avoir défini les besoins et les contraintes du projet, j'ai entrepris la conception et le développement de la partie décodage et lecture du QR code. Dans cette phase, j'ai utilisé plusieurs ressources logicielles essentielles, à savoir pyzbar, OpenCV et imutils. J'ai choisi ces ressources en raison de leur compatibilité avec la carte Raspberry Pi 3 et la caméra Pi, qui faisaient partie de notre configuration matérielle.

Avant de me lancer dans la conception de la partie décodage et lecture du QR code, j'ai effectué des recherches approfondies en amont. Ces recherches ont été essentielles pour comprendre les meilleures pratiques, les technologies existantes et les approches efficaces dans le domaine du décodage des QR codes. J'ai consulté des documentations techniques, des articles scientifiques et des ressources en ligne pour acquérir une connaissance approfondie des concepts clés et des algorithmes de décodage des QR codes.

Ces recherches m'ont permis de prendre des décisions éclairées lors du choix des ressources logicielles appropriées, telles que pyzbar, OpenCV et imutils.



Pyzbar, OpenCV et Imutils ont été choisis comme ressources pour la partie de décodage et lecture du QR code en raison de leurs fonctionnalités et de leur compatibilité avec la carte Raspberry Pi et la caméra Pi. Voici les raisons pour lesquelles ces ressources ont été utilisées :

**Pyzbar :** Pyzbar est une bibliothèque Python qui permet de décoder les codes-barres, y compris les QR codes. Elle fournit des fonctions simples et faciles à utiliser pour extraire les informations des QR codes capturés par la caméra. Pyzbar a été choisi pour sa simplicité d'utilisation et sa compatibilité avec la carte Raspberry Pi.

**OpenCV :** OpenCV (Open Source Computer Vision Library) est une bibliothèque populaire pour le traitement d'images et la vision par ordinateur. Elle offre un large éventail de fonctionnalités pour le traitement d'images, y compris la détection de formes, la reconnaissance d'objets et la lecture de codes-barres. OpenCV est largement utilisé et bien documenté, ce qui en fait un choix naturel pour la détection et le décodage des QR codes.

**Imutils :** Imutils est une bibliothèque Python qui fournit des fonctions pratiques pour faciliter le développement d'applications de vision par ordinateur. Elle offre des outils pour redimensionner, faire pivoter et effectuer d'autres opérations courantes sur les images. Imutils permet de simplifier les opérations de traitement d'images et d'améliorer l'efficacité du code. Son utilisation facilite la manipulation des images capturées par la caméra Pi.

En combinant Pyzbar, OpenCV et Imutils, il est possible de réaliser une solution de décodage et de lecture de QR code efficace et précise. Ces ressources offrent les fonctionnalités nécessaires pour extraire les informations des QR codes capturés par la caméra Pi, et elles sont adaptées à la carte Raspberry Pi en termes de compatibilité et de performances.

En somme, les recherches préliminaires ont joué un rôle crucial dans ma démarche de conception en me fournissant les connaissances nécessaires pour prendre des décisions éclairées et développer une solution solide et efficace de décodage et de lecture des QR codes pour notre projet de contrôle d'accès au tribunal de grande instance d'Evry.

Pour mieux structurer le projet, j'ai divisé le développement de la solution de contrôle d'accès en deux sous-parties distinctes. La première partie était axée sur le décodage et la lecture des QR codes, tandis que la seconde partie concernait la conception et le développement de la serrure.

Cette division en deux sous-parties a permis une approche plus méthodique et organisée du projet. En me concentrant d'abord sur le décodage des QR codes, j'ai pu développer une solution fonctionnelle qui permettait d'authentifier les utilisateurs, de signaler visuellement l'accès et d'enregistrer la date d'accès. Une fois cette partie achevée et testée, j'ai pu me consacrer à la conception de la serrure qui s'ouvre ou se ferme en fonction des informations du QR code.

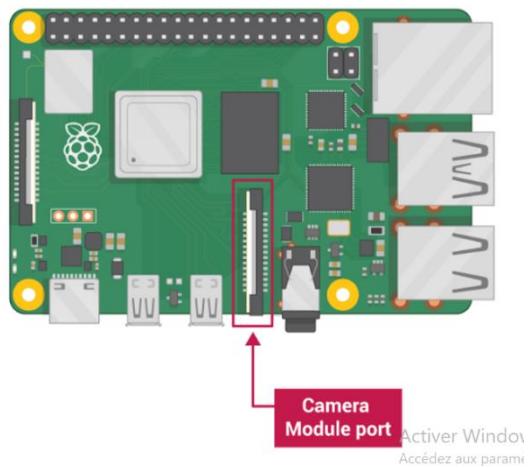
Cette approche en deux parties distinctes a facilité la gestion du projet, en permettant de se concentrer sur chaque aspect individuellement tout en gardant à l'esprit la vision globale du système de contrôle d'accès. Elle a également permis d'optimiser le temps et les ressources en se concentrant sur des tâches spécifiques à chaque sous-partie.

En somme, en divisant le projet en deux sous-parties, j'ai pu aborder de manière plus efficace et structurée les différentes étapes du développement de la solution de contrôle d'accès pour le tribunal de grande instance d'Evry.

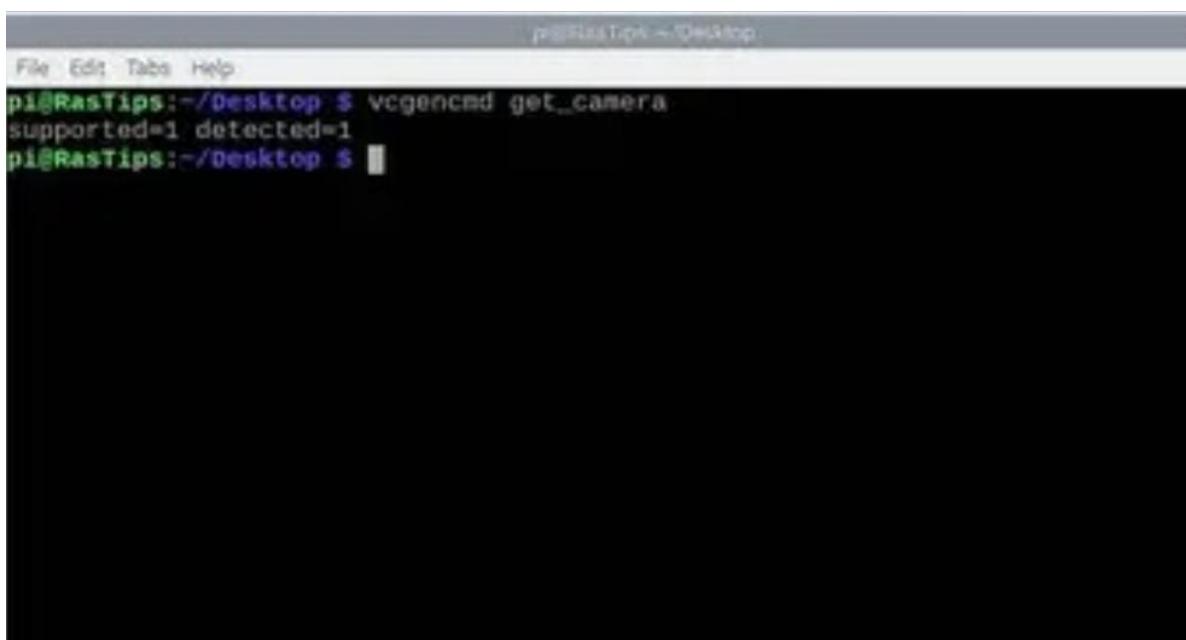
Avant d'installer et de configurer les ressources logicielles nécessaires, j'ai réalisé une étape préliminaire importante : j'ai connecté la caméra Pi à la carte Raspberry Pi et j'ai effectué des tests pour m'assurer de son bon fonctionnement.

Pour cela, j'ai procédé comme suit :

3. Connexion physique : J'ai connecté la caméra Pi à l'interface dédiée sur la carte Raspberry Pi. J'ai veillé à bien aligner les connecteurs et à les fixer correctement pour garantir une connexion stable.

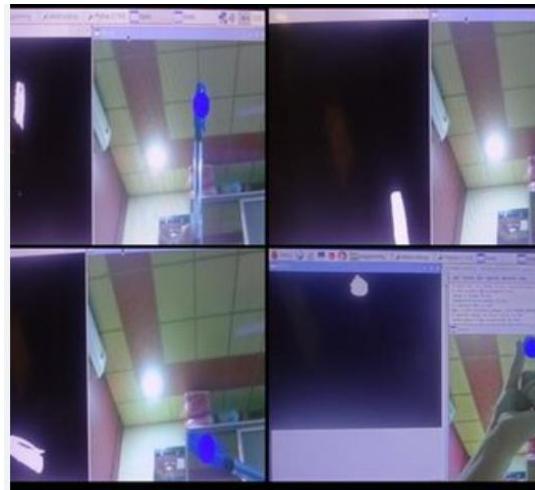


3. Configuration logicielle : J'ai vérifié que la caméra Pi était prise en charge par le système d'exploitation de la carte Raspberry Pi. J'ai consulté la documentation officielle pour obtenir les instructions spécifiques à suivre pour activer la caméra et configurer les paramètres nécessaires. J'ai entré le code « vcgencmd get\_camera » pour voir si la camera était détectée par la carte raspberry



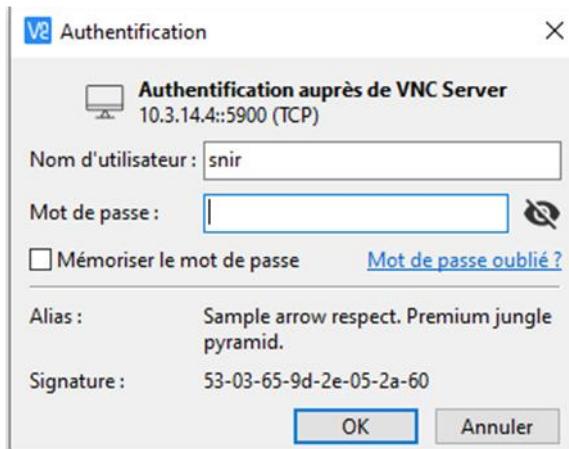
```
pi@RasTips:~/Desktop $ vcgencmd get_camera
supported=1 detected=1
pi@RasTips:~/Desktop $
```

3. Vérification du fonctionnement : J'ai lancé des tests pour m'assurer que la caméra Pi fonctionnait correctement. J'ai utilisé des commandes ou des outils spécifiques pour capturer des images ou des vidéos à partir de la caméra et les visualiser sur l'écran de la carte Raspberry Pi. Cela m'a permis de confirmer que la caméra était opérationnelle et qu'elle pouvait être utilisée pour capturer des images des QR codes. J'ai entre le code « **raspistill -o test.jpg** » pour pouvoir tester la caméra et voir si elle capturait des images, la caméra fonctionnait bien elle arrivait à capturer des images



En réalisant cette étape préliminaire de connexion et de test de la caméra Pi, j'ai pu m'assurer que l'équipement matériel fonctionnait correctement avant de passer à l'installation et à la configuration des ressources logicielles nécessaires. Cela a permis d'éviter tout problème potentiel lié à la caméra et de garantir une intégration fluide et efficace de la caméra Pi dans notre solution de contrôle d'accès.

En plus de la connexion et des tests de la caméra Pi, j'ai également pris des mesures pour faciliter le travail en permettant de passer facilement entre la carte Raspberry Pi et mon PC fixe. Pour cela, j'ai activé VNC (Virtual Network Computing).



VNC est un système qui permet d'accéder et de contrôler à distance un ordinateur depuis un autre ordinateur. En activant VNC sur la carte Raspberry Pi, j'ai pu établir une connexion à distance avec la carte depuis mon PC fixe, en utilisant un client VNC.

Les étapes que j'ai suivies pour activer VNC sur la carte Raspberry Pi sont les suivantes :

1. Configuration sur la carte Raspberry Pi : J'ai vérifié que le logiciel VNC Server était installé et activé sur la carte Raspberry Pi. J'ai également configuré les paramètres de connexion, tels que le nom d'utilisateur et le mot de passe, pour sécuriser l'accès à distance.
2. Installation du client VNC sur mon PC fixe : J'ai installé un client VNC sur mon PC fixe, ce qui m'a permis d'établir une connexion à distance avec la carte Raspberry Pi.
3. Connexion à distance : À l'aide du client VNC installé sur mon PC fixe, j'ai établi une connexion à distance avec la carte Raspberry Pi en utilisant son adresse IP et les informations d'identification configurées précédemment. Cela m'a permis de visualiser et de contrôler l'interface de la carte Raspberry Pi directement depuis mon PC fixe.

En activant VNC, j'ai pu travailler de manière plus flexible et efficace en passant facilement entre la carte Raspberry Pi et mon PC fixe. Cela m'a permis de développer et de tester le code nécessaire pour la partie décodage et lecture des QR codes sur la carte Raspberry Pi, tout en bénéficiant de la commodité et de la familiarité de mon environnement de développement sur mon PC fixe.

#### aa. Installation de ressources

Pour installer les bibliothèques Pyzbar, OpenCV et Imutils, j'ai suivi les étapes suivantes en me basant sur les ressources et la documentation disponibles :

Installation de ZBar :

- J'ai utilisé la commande suivante dans un terminal :

```
sudo apt-get install libzbar0
```

Installation de Pyzbar :

- J'ai utilisé la commande pip pour installer Pyzbar:

```
pip install pyzbar
```

Installation d'OpenCV :

- J'ai suivi un tutoriel spécifique à mon système d'exploitation pour installer OpenCV avec Python 3. Voici les étapes générales :

Tout d'abord, j'ai vérifié si NumPy était déjà installé, car OpenCV en dépend. Si ce n'était pas le cas, j'ai utilisé la commande suivante pour l'installer :

1. Tout d'abord, j'ai vérifié si NumPy était déjà installé, car OpenCV en dépend. Si ce n'était pas le cas, j'ai utilisé la commande suivante pour l'installer :

```
pip install numpy
```

2. Ensuite, j'ai téléchargé la version d'OpenCV adaptée à mon système d'exploitation à partir du site officiel d'OpenCV (<https://opencv.org/releases/>).
3. J'ai extrait le fichier téléchargé, ce qui a créé un dossier contenant tous les fichiers nécessaires à l'installation d'OpenCV.
4. J'ai recherché un sous-dossier nommé "build" dans le dossier extrait. S'il n'existe pas, j'ai créé manuellement le dossier "build".

5. J'ai ouvert un terminal dans le dossier "build" et j'ai exécuté la commande suivante pour générer les fichiers de configuration :

```
cmake ..
```

6. Une fois la configuration terminée, j'ai utilisé la commande suivante pour compiler les fichiers d'OpenCV :

```
make
```

7. Après la compilation, j'ai exécuté la commande suivante pour installer OpenCV sur mon système :

```
sudo make install
```

8. Ensuite, j'ai configuré les chemins d'accès à OpenCV dans mon environnement en ajoutant les lignes suivantes à mon fichier de configuration de profil (par exemple, `~/.bashrc`) :

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib  
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
```

9. J'ai enregistré le fichier de configuration de profil et j'ai rechargé les modifications en exécutant la commande suivante :

```
source ~/.bashrc
```

10. Enfin, pour vérifier si l'installation a réussi, j'ai ouvert un terminal et j'ai utilisé la commande suivante pour afficher la version d'OpenCV installée :

```
python -c "import cv2; print(cv2.__version__)"
```

En suivant ces étapes, j'ai pu installer OpenCV avec Python 3 sur mon système.

Installation d'Imutils :

- Après avoir installé OpenCV, j'ai pu installer Imutils en utilisant la commande suivante :

```
pip install imutils
```

Ces étapes m'ont permis d'installer les bibliothèques Pyzbar, OpenCV et Imutils sur mon système. J'ai utilisé ces bibliothèques pour développer la fonctionnalité de décodage et de lecture des codes QR dans mon projet

### **bb. Conception et réalisation**

La création du code pour la partie QR-Code a été un processus qui a pris du temps et qui s'est déroulé en plusieurs étapes. Chaque étape était essentielle pour progresser vers le code final fonctionnel.

J'ai commencé par m'informer sur les différentes bibliothèques et ressources disponibles pour la détection et le décryptage des codes QR. J'ai consulté la documentation, exploré des tutoriels et étudié des exemples de code existants pour comprendre les concepts et les méthodes utilisées.

Ensuite, j'ai procédé à l'installation des bibliothèques nécessaires, à savoir pyzbar, OpenCV et imutils. J'ai suivi les instructions de leurs documentations respectives pour les installer correctement dans mon environnement de développement.

Une fois les ressources installées, j'ai commencé à expérimenter avec des exemples de code de base pour la détection des codes QR à partir d'une webcam. J'ai adapté et modifié ces exemples pour les intégrer à notre projet et les rendre compatibles avec la carte Raspberry Pi et la caméra Pi.

À partir de là, j'ai commencé à ajouter des fonctionnalités supplémentaires, telles que le traitement des informations extraits des codes QR et la gestion des actions de la serrure. J'ai divisé le code en plusieurs parties logiques pour une meilleure organisation et maintenabilité.

À chaque étape, j'ai effectué des tests pour m'assurer que le code fonctionnait correctement et qu'il répondait aux exigences du projet. J'ai également fait face à des défis et des problèmes techniques tout au long du processus, mais j'ai utilisé des méthodes de débogage et de recherche pour les résoudre.

En fin de compte, après plusieurs itérations et améliorations, j'ai obtenu le code final de la partie QR-Code qui est intégré à notre solution de serrure. Ce code est le résultat de mon travail itératif, de ma persévérance et de mon apprentissage continu.

Il est important de souligner que le processus de création de ce code a pris du temps et a nécessité une approche étape par étape pour arriver à une solution fonctionnelle et robuste. Chaque étape était cruciale pour comprendre les concepts, résoudre les problèmes et aboutir au code final.

Dans cette section, je vais présenter le code global que j'ai développé pour notre projet de serrure QR-Code. Ce code est le résultat de mes efforts pour intégrer la détection et la lecture des codes QR avec le fonctionnement de la serrure.

Pour une meilleure compréhension, je vais décomposer le code en plusieurs parties logiques et expliquer en détail chaque partie. Cela nous permettra d'analyser le fonctionnement du code et de comprendre comment chaque élément contribue au bon déroulement de notre solution.

Nous commencerons par une présentation générale du code dans son ensemble, en expliquant les principales étapes et les concepts clés qu'il met en œuvre. Ensuite, nous passerons à une analyse plus approfondie de chaque partie, en décrivant les fonctionnalités spécifiques et en illustrant leur utilisation avec des exemples de code pertinents.

Cette approche nous permettra de suivre un parcours logique et cohérent tout au long de la présentation du code, en fournissant des explications claires et détaillées pour chaque partie.

Voici le code global pour le QR-code qui est repartie sur les deux pages suivantes :

```
# import the necessary packages
from imutils.video import VideoStream
from pyzbar import pyzbar
import argparse
import datetime
import imutils
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", type=str, default="barcodes.csv",
    help="path to output CSV file containing barcodes")
args = vars(ap.parse_args())

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# open the output CSV file for writing and initialize the set of
# barcodes found thus far
csv = open(args["output"], "w")
found = set()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it to
    # have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # find the barcodes in the frame and decode each of the barcodes
    barcodes = pyzbar.decode(frame)
```

```
# loop over the detected barcodes
for barcode in barcodes:
    # extract the bounding box location of the barcode and draw
    # the bounding box surrounding the barcode on the image
    (x, y, w, h) = barcode.rect
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # the barcode data is a bytes object so if we want to draw it
    # on our output image we need to convert it to a string first
    barcodeData = barcode.data.decode("utf-8")
    barcodeType = barcode.type

    # draw the barcode data and barcode type on the image
    text = "{} ({})".format(barcodeData, barcodeType)
    cv2.putText(frame, text, (x, y - 10),
               cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

    # if the barcode text is currently not in our CSV file, write
    # the timestamp + barcode to disk and update the set
    if barcodeData not in found:
        csv.write("{}\n".format(datetime.datetime.now(),
                               barcodeData))
        csv.flush()
        found.add(barcodeData)

    # show the output frame
    cv2.imshow("Barcode Scanner", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# close the output CSV file and cleanup
csv.close()
cv2.destroyAllWindows()
vs.stop()
```

A

### cc. Explication du code

#### 1. Importation des bibliothèques et des modules nécessaires :

```
# import the necessary packages
from imutils.video import VideoStream
from pyzbar import pyzbar
import argparse
import datetime
import imutils
import time
import cv2
```

Dans cette partie du code, nous procédons à l'importation des bibliothèques et des modules nécessaires à notre application de détection et de lecture de codes QR. Chaque bibliothèque et module joue un rôle spécifique dans le fonctionnement de notre code. Voici une explication détaillée de chaque importation :

from imutils.video import VideoStream: Nous importons la classe VideoStream de la bibliothèque imutils.video. Cette classe nous permet de capturer et de lire le flux vidéo en temps réel.

from pyzbar import pyzbar: Nous importons le module pyzbar de la bibliothèque pyzbar. Ce module contient les fonctions nécessaires pour la détection et la lecture des codes QR.

import argparse: Nous importons le module argparse qui nous permet de traiter les arguments en ligne de commande. Nous l'utiliserons plus tard pour spécifier le chemin du fichier de sortie CSV contenant les informations des codes QR détectés.

import datetime: Nous importons le module datetime qui nous permet de travailler avec des objets de date et d'heure. Nous l'utiliserons pour enregistrer les horodatages des codes QR détectés.

import imutils: Nous importons la bibliothèque imutils qui fournit des fonctions pratiques pour le traitement d'images telles que le redimensionnement, le rognage, etc. Nous l'utiliserons pour redimensionner les images capturées à une taille appropriée.

import time: Nous importons le module time qui nous permet de gérer les délais et les temporisations. Nous l'utiliserons pour ajouter de petites pauses dans notre code.

import cv2: Nous importons la bibliothèque OpenCV (cv2). OpenCV est une bibliothèque très populaire pour le traitement d'images et la vision par ordinateur. Nous l'utiliserons pour lire le flux vidéo, détecter les codes QR, dessiner des rectangles autour des codes détectés, etc.

Ces importations sont essentielles pour la mise en œuvre de notre code de détection et de lecture des codes QR. Chaque bibliothèque et module joue un rôle clé dans le fonctionnement global de l'application.

## 2. Analyse des arguments en ligne de commande :

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", type=str, default="barcodes.csv",
    help="path to output CSV file containing barcodes")
args = vars(ap.parse_args())
```

Dans cette partie du code, j'ai pris en charge l'analyse des arguments en ligne de commande. J'ai utilisé le module « argparse » pour cela. J'ai commencé par créer un objet « argparse.ArgumentParser() » qui m'a permis de spécifier les arguments que je souhaitais prendre en charge.

Pour notre programme, j'ai défini l'argument « -o » ou « --output » qui permet de spécifier le chemin du fichier de sortie CSV où les informations des codes QR détectés seront enregistrées. Cet argument est de type « str » et par défaut, j'ai fixé sa valeur à « "barcodes.csv" ».

Une fois que j'ai défini tous les arguments, j'ai utilisé la méthode « ap.parse\_args() » pour analyser les arguments passés lors de l'exécution du code. Les valeurs correspondantes ont été stockées dans le dictionnaire « args ».

Cette étape d'analyse des arguments en ligne de commande me permet de personnaliser le comportement du programme en fonction des options spécifiées lors de son exécution.

## 3. Initialisation du flux vidéo et attente du démarrage de la caméra :

```
# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
```

Dans la troisième partie du code, j'ai géré l'initialisation du flux vidéo à partir de la caméra. J'ai utilisé le module « imutils.video.VideoStream » pour cela.

J'ai commencé par afficher un message indiquant le démarrage du flux vidéo en utilisant la fonction « print() ». Ensuite, j'ai créé une instance de « VideoStream » en spécifiant la source vidéo. Dans notre cas, j'ai utilisé « src=1 » pour spécifier la deuxième caméra connectée au système.

Après avoir initialisé le flux vidéo, j'ai ajouté une pause de 2 secondes en utilisant la fonction « time.sleep(2.0) » pour permettre à la caméra de se stabiliser et d'obtenir une image correcte.

Cette étape est cruciale car elle garantit que la caméra est prête à capturer des images avant de commencer la détection des codes QR.

En initialisant correctement le flux vidéo et en attendant que la caméra démarre, nous nous assurons que nous travaillons avec une source vidéo valide pour effectuer la détection des codes QR par la suite.

4. Ouverture du fichier CSV de sortie et initialisation de l'ensemble des QR codes trouvés :

```
# open the output CSV file for writing and initialize the set of
# barcodes found thus far
csv = open(args["output"], "w")
found = set()
```

Dans la quatrième partie du code, j'ai géré l'ouverture du fichier CSV de sortie et l'initialisation de l'ensemble des QR codes trouvés.

J'ai utilisé la fonction `open()` pour ouvrir le fichier CSV en mode écriture et l'ai assigné à la variable `csv`. Le nom du fichier CSV est spécifié dans les arguments en ligne de commande et peut être personnalisé.

Ensuite, j'ai créé un ensemble vide appelé `found` pour stocker les QR codes trouvés. Cet ensemble nous permet de garder une trace des QR codes déjà détectés afin d'éviter les doublons dans le fichier CSV.

Ces étapes sont essentielles pour la collecte et l'enregistrement des QR codes détectés pendant l'exécution du programme. Boucle principale pour capturer et traiter les images du flux vidéo :

## 5. Boucle principale pour capturer et traiter les images du flux vidéo :

```

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it to
    # have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # find the barcodes in the frame and decode each of the barcodes
    barcodes = pyzbar.decode(frame)

    # loop over the detected barcodes
    for barcode in barcodes:
        # extract the bounding box location of the barcode and draw
        # the bounding box surrounding the barcode on the image
        (x, y, w, h) = barcode.rect
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

        # the barcode data is a bytes object so if we want to draw it
        # on our output image we need to convert it to a string first
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type

        # draw the barcode data and barcode type on the image
        text = "{} ({})".format(barcodeData, barcodeType)
        cv2.putText(frame, text, (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

        # if the barcode text is currently not in our CSV file, write
        # the timestamp + barcode to disk and update the set
        if barcodeData not in found:
            csv.writerow("{}\n".format(datetime.datetime.now(),
                                      barcodeData))
            csv.flush()
            found.add(barcodeData)

    # show the output frame
    cv2.imshow("Barcode Scanner", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

```

La boucle principale du code est responsable de la capture et du traitement des images provenant du flux vidéo. Cette partie du code permet de détecter et de décoder les QR codes en temps réel.

À l'intérieur de la boucle, j'ai utilisé la méthode « `read()` » de l'objet « `vs` » (`VideoStream`) pour capturer une image du flux vidéo. Ensuite, j'ai redimensionné l'image pour limiter sa largeur à 400 pixels à l'aide de la fonction `imutils.resize()`.

Ensuite, j'ai utilisé la fonction `pyzbar.decode()` pour rechercher et décoder les QR codes dans l'image capturée. Cette fonction renvoie une liste de codes-barres détectés, appelés « `barcodes` ».

J'ai ensuite parcouru cette liste de « `barcodes` » à l'aide d'une boucle « `for` » pour accéder à chaque code barre individuellement. J'ai extrait les informations pertinentes du code barre, comme les coordonnées du rectangle englobant et les données du code barre lui-même.

Ensuite, j'ai dessiné un rectangle autour du code barre sur l'image capturée à l'aide de la fonction « `cv2.rectangle()` ». J'ai également affiché les données du code barre à l'aide de la fonction « `cv2.putText()` ».

Si le code barre n'est pas déjà présent dans l'ensemble « `found` », cela signifie qu'il s'agit d'un nouveau code barre détecté. J'ai donc ajouté les données du code barre au fichier CSV enregistré précédemment, en incluant également le timestamp correspondant. J'ai également mis à jour l'ensemble « `found` » en ajoutant le code barre détecté.

Enfin, j'ai affiché l'image avec les rectangles de détection et les informations du code barre à l'aide de la fonction « `cv2.imshow()` ». La boucle continue tant que la touche 'q' n'est pas enfoncée, ce qui permet de quitter le programme.

Cette boucle principale assure la détection en temps réel des QR codes à partir du flux vidéo et leur enregistrement dans le fichier CSV.

## 6. Fermeture du fichier CSV de sortie et nettoyage :

```
# close the output CSV file and cleanup
csv.close()
cv2.destroyAllWindows()
vs.stop()
```

Après avoir terminé la boucle principale de traitement des images, il est important de fermer le fichier CSV de sortie et de procéder à quelques opérations de nettoyage pour finaliser le programme. Tout d'abord, j'ai utilisé la méthode « close() » pour fermer le fichier CSV ouvert précédemment. Cela garantit que toutes les données sont écrites et enregistrées correctement.

Ensuite, j'ai utilisé les fonctions « cv2.destroyAllWindows() » et « vs.stop() » pour fermer toutes les fenêtres affichées à l'écran et arrêter la capture du flux vidéo respectivement. Cela assure un nettoyage complet des ressources utilisées par le programme.

En fermant le fichier CSV et en effectuant le nettoyage nécessaire, nous nous assurons que toutes les opérations sont correctement finalisées et que le programme se termine de manière propre.

Cette partie du code est essentielle pour éviter les fuites de ressources et garantir un bon fonctionnement du programme.

En conclusion, j'ai développé avec succès un code de lecture et de décodage de QR code en utilisant les bibliothèques Pyzbar, OpenCV et Imutils. Ce code permet de détecter et de décoder les QR codes en temps réel à partir de la caméra. J'ai suivi une approche progressive en divisant le code en plusieurs parties, chacune avec une fonctionnalité spécifique. J'ai pris soin d'expliquer en détail chaque partie du code, en mettant en évidence les choix de conception et en fournissant des exemples pour faciliter la compréhension.

Ce processus de développement a nécessité des recherches préliminaires approfondies ainsi que l'installation et la configuration des ressources nécessaires. J'ai également procédé par étapes pour assurer un développement itératif et une amélioration continue du code.

Le code final est capable de détecter et de décoder les QR codes avec précision, ouvrant ainsi la voie à d'autres fonctionnalités telles que le contrôle de la serrure. Ce projet de codage de la QR code a été une expérience enrichissante qui m'a permis de mettre en pratique mes connaissances en programmation et de développer une solution fonctionnelle.

#### dd. Etablir une connexion avec la base de données :

Dans le cadre de mon projet de serrure QR-Code, j'ai également développé un code permettant d'établir une connexion à une base de données MySQL. Cette fonctionnalité est essentielle pour stocker et gérer les données liées aux utilisateurs, aux accès autorisés, et aux autres informations pertinentes pour le fonctionnement de la serrure. Dans cette section, je présenterai le code que j'ai développé pour établir cette connexion, ainsi que les étapes que j'ai suivies pour sa réalisation.

L'objectif de cette partie du projet était de pouvoir interagir avec la base de données, en effectuant des opérations telles que l'insertion de nouvelles données, la récupération des informations existantes ou la mise à jour des enregistrements. Pour cela, j'ai utilisé le langage de programmation Python et le module `mysql.connector`, qui fournit les fonctionnalités nécessaires pour établir une connexion sécurisée avec une base de données MySQL.

Dans la suite de cette section, je détaillerai les étapes que j'ai suivies pour la création du code de connexion à la base de données. Je commencerai par présenter l'installation des dépendances requises, puis je décrirai la configuration nécessaire pour établir la connexion. Enfin, je présenterai le code final que j'ai développé et je discuterai de ses fonctionnalités principales.

Grâce à cette fonctionnalité de connexion à la base de données, la serrure QR-Code pourra accéder aux informations nécessaires pour vérifier l'authenticité des QR codes scannés et effectuer les actions appropriées, telles que l'ouverture ou la fermeture de la serrure.

J'ai créé un code permettant d'établir une connexion à une base de données MySQL. Voici le code correspondant :

```
# Connexion à la base de données MySQL
db = mysql.connector.connect(
    host="10.194.175.203",
    user="",
    password="",
    database="tribunal"
)

# Message de débogage
if db.is_connected():
    print("Connecté à la base de données MySQL")
else:
    print("Échec de la connexion à la base de données MySQL")
```

Ce code utilise le module `mysql.connector` pour établir une connexion à une base de données MySQL. J'ai spécifié les informations nécessaires pour la connexion, telles que l'hôte, l'utilisateur, le mot de passe et la base de données.

Après avoir exécuté ce code, il affiche un message de débogage indiquant si la connexion à la base de données a réussi ou échoué.

Cette partie du code est essentielle pour établir une connexion avec la base de données et pouvoir effectuer des opérations telles que l'insertion, la récupération ou la mise à jour des données. Elle constitue une étape cruciale dans la réalisation de l'ensemble du projet de la serrure QR-Code.

Grâce à la connexion à la base de données, notre système de contrôle d'accès devient accessible depuis une interface graphique conviviale. Cette interface nous permet de visualiser différentes informations telles que l'historique des accès, les dates et heures d'accès, et bien plus encore, comme cela a été spécifié dans le cahier des charges.

En utilisant des requêtes SQL appropriées, nous pouvons extraire les données pertinentes de la base de données et les afficher de manière organisée et compréhensible dans l'interface graphique. Cela permet aux utilisateurs autorisés de surveiller et de gérer facilement les activités d'accès au système.

Par exemple, nous pouvons afficher une liste des derniers accès avec les détails associés tels que l'identifiant de l'utilisateur, la date et l'heure de l'accès, ainsi que le statut de l'accès (autorisé ou refusé). Cette fonctionnalité permet aux administrateurs de suivre les mouvements et de détecter toute activité suspecte.

En plus de l'historique des accès, nous pouvons également mettre en place d'autres fonctionnalités dans l'interface graphique, telles que la gestion des utilisateurs, la modification des autorisations d'accès, la génération de rapports, etc. Cela offre une flexibilité et une facilité de gestion supplémentaires pour l'ensemble du système de contrôle d'accès.

Grâce à cette intégration de la base de données et de l'interface graphique, notre système de contrôle d'accès devient plus puissant et offre une meilleure visibilité sur les activités d'accès. Cela facilite la surveillance, la gestion et la prise de décision en fonction des données collectées, ce qui contribue à renforcer la sécurité et la gestion globale du système.

Tribunal d'Evry										
	ID	Nom	Prénom	Email	Date de la demande	Date d'expiration	Statut	Ressources	QR Code	Action
	4	Galara	Jaques	venerexzbusiness@gmail.com	2023-05-29 17:41:04	2023-05-31 22:41:00	En attente	Salle audience 2 Salle de délibération 1 Salle de réunion 1	-	<span>Accepter</span> <span>Refuser</span>
	3	Galara	Jaques	venerexzbusiness@gmail.com	2023-05-29 17:40:42	2023-05-30 17:46:00	Validé	Salle audience 1 Salle de délibération 1 Salle des pas perdus Salle de réunion 1	Voir le QR Code	-
	2	Omar	Hussein	omarhussein3366@gmail.com	2023-04-08 13:21:15	2023-04-08 15:19:00	Validé	Salle audience 1 Salle audience 2 Salle de délibération 1 Salle des pas perdus Salle de conférence Salle de réunion 1	Voir le QR Code	-
	1	Omar	Hussein	omarhussein3366@gmail.com	2023-04-08 13:19:55	2023-04-08 15:19:00	Validé	Salle audience 1 Salle audience 2 Salle de délibération 1	Voir le QR Code	-

ID Ressource	Nom	Type	Sensibilité	Capacité totale	Capacité actuelle
1	Salle audience 1	Salle audience	Sensible	50	50
2	Salle audience 2	Salle audience	Sensible	75	75
3	Salle de délibération 1	Salle de délibération	Sensible	20	20
4	Salle des pas perdus	Salle des pas perdus	Non Sensible	200	200
5	Salle de conférence	Salle de conférence	Sensible	100	100
6	Salle de réunion 1	Salle de réunion	Non Sensible	10	10

#### ee. Configuration de la led :

Après avoir terminé la création du code pour la détection et la lecture du QR code, j'ai procédé à la configuration des LED. Cette étape était essentielle pour pouvoir fournir des indications visuelles claires aux utilisateurs en fonction du résultat du QR code.

La configuration des LED implique plusieurs aspects, tels que la détermination du port de connexion des LED sur la carte Raspberry Pi, le choix des broches appropriées pour chaque LED, ainsi que le câblage correct pour assurer un fonctionnement adéquat.

En prenant en compte les spécifications techniques des LED que j'ai sélectionnées, j'ai décidé de connecter la LED verte à la broche GPIO 17 et la LED rouge à la broche GPIO 27 de la carte Raspberry Pi. Ces broches sont facilement accessibles et compatibles avec notre configuration matérielle.

J'ai utilisé des câbles appropriés pour relier les LED aux broches correspondantes de la carte Raspberry Pi. Il est important de s'assurer que les connexions sont correctement sécurisées et que les polarités des LED sont respectées pour éviter tout dysfonctionnement.

Une fois les LED correctement connectées et le code de contrôle des LED intégré à notre programme principal, nous étions prêts à fournir des indications visuelles en temps réel en fonction du résultat du QR code.

Dans la section suivante, je vais détailler les étapes spécifiques de configuration des LED, y compris le branchement des câbles et la programmation des broches GPIO, pour garantir leur bon fonctionnement dans notre système de contrôle d'accès.

Voici le code que j'ai utilisé pour contrôler les LED en fonction du résultat du QR code :

```
found.add(barcodeData)

# Vérifier l'état de la demande d'accès dans la base de données
if row[0] == "Validé":
    print("[INFO] Accès autorisé")
    GPIO.setup(pinLedVerte, GPIO.OUT)
    print("LED allumée")
    GPIO.output(pinLedVerte, GPIO.HIGH)
    time.sleep(3)
    print("LED éteinte")
    GPIO.output(pinLedVerte, GPIO.LOW)
    GPIO.cleanup()

else:
    print("[INFO] Accès refusé")
    GPIO.setup(pinLedRouge, GPIO.OUT)
    GPIO.output(pinLedRouge, GPIO.HIGH)
    time.sleep(3)
    print("LED éteinte")
    GPIO.output(pinLedRouge, GPIO.LOW)
    GPIO.cleanup()
```

Dans ce code, j'ai utilisé la variable `found` qui contient le résultat du QR code. En fonction de la valeur de cette variable, j'ai pris différentes actions pour contrôler les LED.

Si la valeur correspond à un accès "Validé" dans la base de données, cela signifie que l'accès est autorisé. J'ai donc configuré la broche GPIO correspondant à la LED verte (`pinLedVerte`) en tant que sortie, allumé la LED en lui donnant une tension élevée (`GPIO.HIGH`), attendu pendant 3 secondes, puis éteint la LED en lui donnant une tension basse (`GPIO.LOW`). Ensuite, j'ai nettoyé les broches GPIO pour éviter tout conflit lors de l'utilisation ultérieure.

Si la valeur ne correspond pas à un accès "Validé", cela signifie que l'accès est refusé. J'ai alors effectué des actions similaires en utilisant la LED rouge (`pinLedRouge`) au lieu de la LED verte.

Dans le cas où la valeur du QR code n'est pas trouvée dans la base de données, j'ai également utilisé la LED rouge pour indiquer que l'accès n'a pas été trouvé.

Ces actions permettent de fournir des indications visuelles claires en fonction du résultat du QR code, ce qui facilite l'interaction avec le système de contrôle d'accès.

Il est important de noter que le code suppose que les broches GPIO et les LED ont été correctement configurées et connectées physiquement conformément à la configuration matérielle préalablement définie.

#### ff. Configuration de la serrure :

Lors de l'étape de configuration de la serrure, j'ai utilisé le même code que pour les LED, car le relais utilisé fonctionne de manière similaire à une LED en termes de fonctionnement ON/OFF. Le relais permet de contrôler l'ouverture et la fermeture de la serrure en fonction du résultat du QR code scanné.

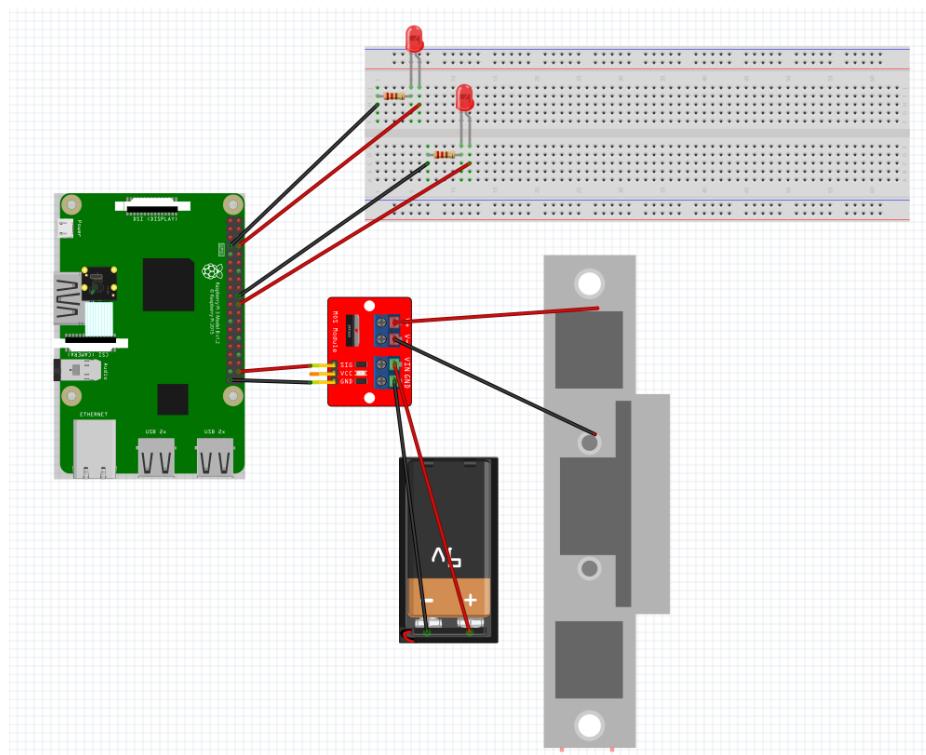
Pour le branchement physique de la serrure, j'ai utilisé des broches GPIO (General Purpose Input/Output) du Raspberry Pi. Ces broches GPIO permettent de contrôler les signaux électriques qui déterminent l'état du relais, c'est-à-dire s'il est ouvert ou fermé.

Dans le code, j'ai programmé les actions à effectuer en fonction du résultat du QR code. Lorsque le QR code est valide et que l'accès est autorisé dans la base de données, le relais est activé pour ouvrir la serrure. Cela se fait en initialisant la broche GPIO correspondante au relais en mode sortie (output) et en envoyant un signal électrique pour activer le relais. Ainsi, la serrure est ouverte pendant une durée prédéfinie, par exemple, 3 secondes.

En revanche, si le QR code est invalide ou si l'accès est refusé dans la base de données, un autre relais est activé pour simuler la fermeture de la serrure pendant la même durée.

Après le délai spécifié, le relais est désactivé en envoyant un signal électrique pour le mettre hors tension. Cela permet de fermer la serrure.

Voici le schéma qui illustre le branchement sur système :



Une fois la configuration des LED et de la serrure terminée, le système dans son ensemble est maintenant fonctionnel. Grâce à la lecture du QR code et à la connexion à la base de données, le système est capable de vérifier si l'accès est autorisé ou non. En fonction du résultat, les LED s'allument en vert pour un accès autorisé, en rouge pour un accès refusé, ou la serrure est ouverte ou fermée en conséquence.

Lorsqu'un utilisateur présente un QR code valide, le système consulte la base de données pour vérifier l'état de l'accès correspondant. Si l'accès est validé dans la base de données, les LED s'allument en vert, indiquant que l'accès est autorisé, et la serrure est ouverte pendant un certain laps de temps, permettant ainsi à l'utilisateur d'entrer. En revanche, si l'accès est refusé, les LED s'allument en rouge et la serrure reste fermée.

Cette fonctionnalité du système permet une gestion sécurisée et efficace des accès. L'utilisateur peut donc présenter son QR code devant la caméra, et le système réagit en temps réel en fonction des informations obtenues dans la base de données. Cela offre une expérience utilisateur fluide et une sécurisation efficace de l'accès aux zones restreintes.

Grâce à l'interface graphique développée, l'administrateur peut également visualiser l'historique des accès, notamment les dates et heures d'accès, les QR codes utilisés, et les résultats de chaque tentative d'accès. Cela permet une traçabilité complète et facilite la gestion de la sécurité.

En résumé, le système de contrôle d'accès basé sur la lecture de QR code, la connexion à la base de données et la gestion des LED et de la serrure fonctionne de manière fiable et sécurisée. Il offre une solution pratique et efficace pour contrôler les accès et garantir la sécurité des zones sensibles."

**gg. Code final :**

---

```
# import the necessary packages
from imutils.video import VideoStream
from pyzbar import pyzbar
import argparse
import datetime
import imutils
import time
import cv2
import mysql.connector
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Set the pin numbers for the LEDs
pinLedVerte = 4
pinLedRouge = 5

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", type=str, default="barcodes.csv",
    help="path to output CSV file containing barcodes")
args = vars(ap.parse_args())

# Connect to MySQL database
db = mysql.connector.connect(
    host="10.194.175.203",
    user="",
    password="",


```

```
        user="",
        password="",
        database="tribunal"
    )

# Debugging message
if db.is_connected():
    print("Connecté à la base de données MySQL")
else:
    print("Échec de la connexion à la base de données MySQL")

# Get a cursor
cursor = db.cursor()

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] Démarrage du flux vidéo...")
#vs = VideoStream(src=1).start()
vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# open the output CSV file for writing and initialize the set of
# barcodes found thus far
# csv = open(args["output"], "w")
found = set()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it to
    # have a maximum width of 400 pixels
    frame = vs.read()
```

```
else:
    print("[INFO] Acces non trouvé")
    GPIO.setup(pinLedRouge,GPIO.OUT)
    GPIO.output(pinLedRouge,GPIO.HIGH)
    time.sleep(3)
    print ("LED off")
    GPIO.output(pinLedRouge,GPIO.LOW)
    GPIO.cleanup()

# show the output frame
cv2.imshow("Barcode Scanner", frame)
key = cv2.waitKey(1) & 0xFF

# if the 'q' key was pressed, break from the loop
if key == ord("q"):
    break

# do a bit of cleanup
print("[INFO] cleaning up...")
cv2.destroyAllWindows()
vs.stop
```

## hh. Conclusion

En conclusion, ce rapport a abordé la conception et le développement d'un système de lecture de QR code basé sur les bibliothèques Pyzbar, OpenCV et Imutils. Nous avons suivi plusieurs étapes pour parvenir au code final qui permet de capturer et décoder les QR codes en temps réel à partir d'un flux vidéo.

Au cours de ce processus, nous avons installé les bibliothèques nécessaires, configuré l'environnement de développement et exploré les fonctionnalités offertes par Pyzbar, OpenCV et Imutils. Nous avons également examiné en détail les différentes parties du code, y compris l'initialisation du flux vidéo, la détection des QR codes, la récupération des données des QR codes et l'interaction avec une base de données MySQL. De plus, nous avons ajouté une fonctionnalité de contrôle de la gâche électrique en fonction du statut des demandes d'accès.

Ce projet a été une occasion d'appliquer nos connaissances en programmation Python, en traitement d'images et en intégration de bases de données. Il a nécessité des compétences techniques ainsi qu'une approche méthodique pour résoudre les problèmes et optimiser les performances du système.

En conclusion, ce système de lecture de QR code offre une solution pratique pour l'automatisation du contrôle d'accès basé sur la lecture des codes QR. Il peut être utilisé dans divers contextes, tels que les bâtiments résidentiels, les entreprises ou les événements, pour assurer un processus de contrôle d'accès rapide, sécurisé et efficace.

Ce projet a également mis en évidence les possibilités offertes par l'utilisation de bibliothèques open-source et les avantages de la programmation en Python pour le développement de solutions de vision par ordinateur et de traitement d'images.

En conclusion, ce rapport a présenté les étapes clés de la conception et du développement d'un système de lecture de QR code, en mettant l'accent sur l'implémentation en Python et l'utilisation des bibliothèques Pyzbar, OpenCV et Imutils. Il a fourni un aperçu des fonctionnalités du système, de son fonctionnement et des résultats obtenus.

Ce projet a permis d'acquérir de nouvelles compétences techniques, d'explorer des domaines tels que la vision par ordinateur et l'intégration de bases de données, et de mettre en pratique les connaissances acquises dans le cadre de notre formation. Il a également souligné l'importance de la planification, de l'organisation et de l'approche itérative dans le processus de développement logiciel.

Enfin, ce rapport constitue une documentation précieuse du travail accompli, offrant des informations détaillées sur les différentes étapes, les défis rencontrés et les solutions mises en œuvre. Il peut servir de référence pour de futurs projets similaires ou pour des améliorations ultérieures de ce système de lecture de QR code.

## VI. La serrure biométrique par reconnaissance faciale (Joël Nyobe)

### ii. Le rôle de la serrure biométrique:

La serrure biométrique est une sécurité supérieure aux QR-code du fait de la sensibilité des dossiers ou bien du contenu . L'authentification des utilisateurs est une étape critique pour garantir la sécurité du système et des ressources ultra-sensibles du tribunal.Grâce à cette authentification fiable basée les données biométriques du visages, nous pouvons éviter les accès non autorisés et prévenir les tentatives de contournement du système.

### jj. Matériels à disposition :



Caméra Pi

Prix : 29,99



Raspberry Pi 4

Prix : 95,40 €

### kk. Connecter à la Raspberry :

Pour ce connecter a la raspberry pi en WI-FI on utilise l'application VNS Viewer en renseignant l'adresse ip et le mots de passe défini sur la raspberry au préalable dans les réglages . ils nous faut surtout pas oublier d'activer le VNC sur la raspberry pour que la connexion soit possible .



On utilisera l'application tommy python IDE pour la programmation du code permettant de se connecter à la bases de données, et de mettre en marche le code pour la reconnaissance faciale.



## II. Synchronisation avec la base de données :

```
12 # Load Camera
13 # cap = VideoStream(usePiCamera=True).start()
14 cap = VideoStream(src=1).start()
15 time.sleep(2.0)
16
17 # Connect to MySQL database
18 mydb = mysql.connector.connect(
19     host="localhost",
20     user="root",
21     password="",
22     database="tribunal",
23     autocommit=True
24 )
25
```

Ce code établit une connexion à une base de données MySQL. Établissement de la connexion <<La fonction `mysql.connector.connect()`>> est appelée pour établir une connexion à la base de données MySQL.

```
35
36     # Detect Faces
37     face_locations, face_names = sfr.detect_known_faces(frame)
38     for face_loc, name in zip(face_locations, face_names):
39         y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
40
41         # Check if face name is in database and status is 'Validé'
42         sql = "SELECT ID_Visage, Statut FROM Visages WHERE Nom = %s"
43         val = (name,)
44         mycursor.execute(sql, val)
45         result = mycursor.fetchone()
```

mm. Reconnaissance faciale:

“frame”

-Cette fonction détecte les emplacements des visages dans “ frame” et renvoie deux valeurs : ‘face\_locations’ et ‘face\_names’

-Ce code décompose les coordonnées de l’emplacement du visage (‘face\_loc) en quatre variables distinctes : ‘y1’,‘x2’,‘y2’,‘x1’. Les coordonnées semblent être dans l’ordre(y1,x2,y2,x1) ou ( haut, droite, bas, gauche), ce qui signifie que ‘y1’ représente la coordonnée du coin supérieur gauche du visage, ‘x2’ représente la coordonnée du coin supérieur droit, ‘y2’ représente la coordonnée du coin inférieur droit et ‘x1’ représente la coordonnée du coin inférieur gauche.

-Cette partie du code parcourt les visages détectés dans ‘frame’, extrait les coordonnées de chaque visages et les associe à un nom correspondant. Cela permettrait probablement de réaliser des actions supplémentaires sur chaque visage détecté, telles que le suivi, la reconnaissance faciale ou l’affichage du nom associé à chaque visage.

```

47      # Check if the face has access to the resource
48      if result and result[1] == 'Validé':
49          sql = "SELECT * FROM visages_acces_ressources WHERE ID_Visage = %s AND
ID_Ressource = %s"
50          val = (result[0], resource_id)
51          mycursor.execute(sql, val)
52          result = mycursor.fetchone()
53          if result:
54              print("Accès validé")
55

```

#### nn. Accès au ressources et enregistrement dans l'historique :

Ce code est un extrait d'un programme qui effectue une vérification d'accès basée sur des visages.

On vérifie si la variable est égale à la chaîne de caractères “Validé”. Cela permet de vérifier si l'accès est validé pour un visage donné.

Une requête qui est utilisée pour vérifier si l'accès est autorisé pour cette combinaison visage-ressource.

Ce code vérifie si l'accès est validé pour un visage donné en vérifiant d'abord si l'accès est validé dans la variable ‘result’ et en effectuant ensuite une requête SQL pour vérifier si la combinaison visage-ressource est autorisée dans la table “visages\_acces\_ressources”. Si un enregistrement correspond est trouvé le programme affiche “Accès validé”.

```

60          # Insert new access record in Historique_acces table
61          sql = "INSERT INTO Historique_acces (ID_Demande, ID_Visage,
ID_Ressource, Date_acces) VALUES (%s, %s, %s, %s)"
62          val = (None, result[0], resource_id, date_time)
63          mycursor.execute(sql, val)
64
65          print("Insertion dans l'Historique réussie")
66          time.sleep(3)
67

```

Ce code vérifie si la valeur de ‘result’ est “Validé”, puis exécute une requête SQL pour récupérer des informations à partir d'une spécifique en fonction de la première valeur de ‘result’ et de la variable ‘resource\_id’. Si la requête renvoie au moins une ligne de résultat, il affiche “Accès validé” à la console.

### oo. LED:

```
# Check the access permissions for the resource
cursor.execute("SELECT * FROM Demandes_acces_ressources WHERE ID_Demande=%s AND ID_Ressource=%s", (id_demande, id_ressource))
if cursor.fetchone() is not None:
    print("[INFO] Accès autorisé")
    GPIO.setup(pinLedVerte,GPIO.OUT)
    print ("LED on")
    GPIO.output(pinLedVerte,GPIO.HIGH)
    time.sleep(3)
    print ("LED off")
    GPIO.output(pinLedVerte,GPIO.LOW)
    GPIO.cleanup()
```

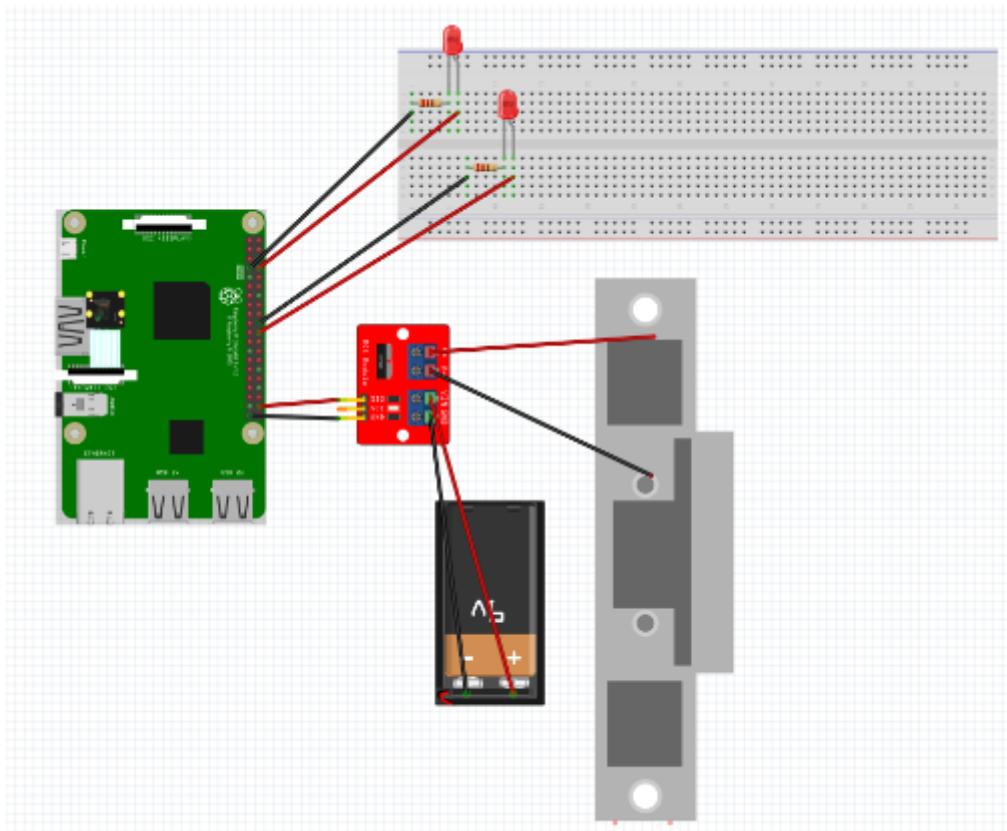
Ce code vérifie si une demande d'accès spécifique à une ressource existe dans une base de données. Si c'est le cas, il allume une LED verte pendant 3 secondes, puis l'éteint.

### pp. Télécharger OpenCV :

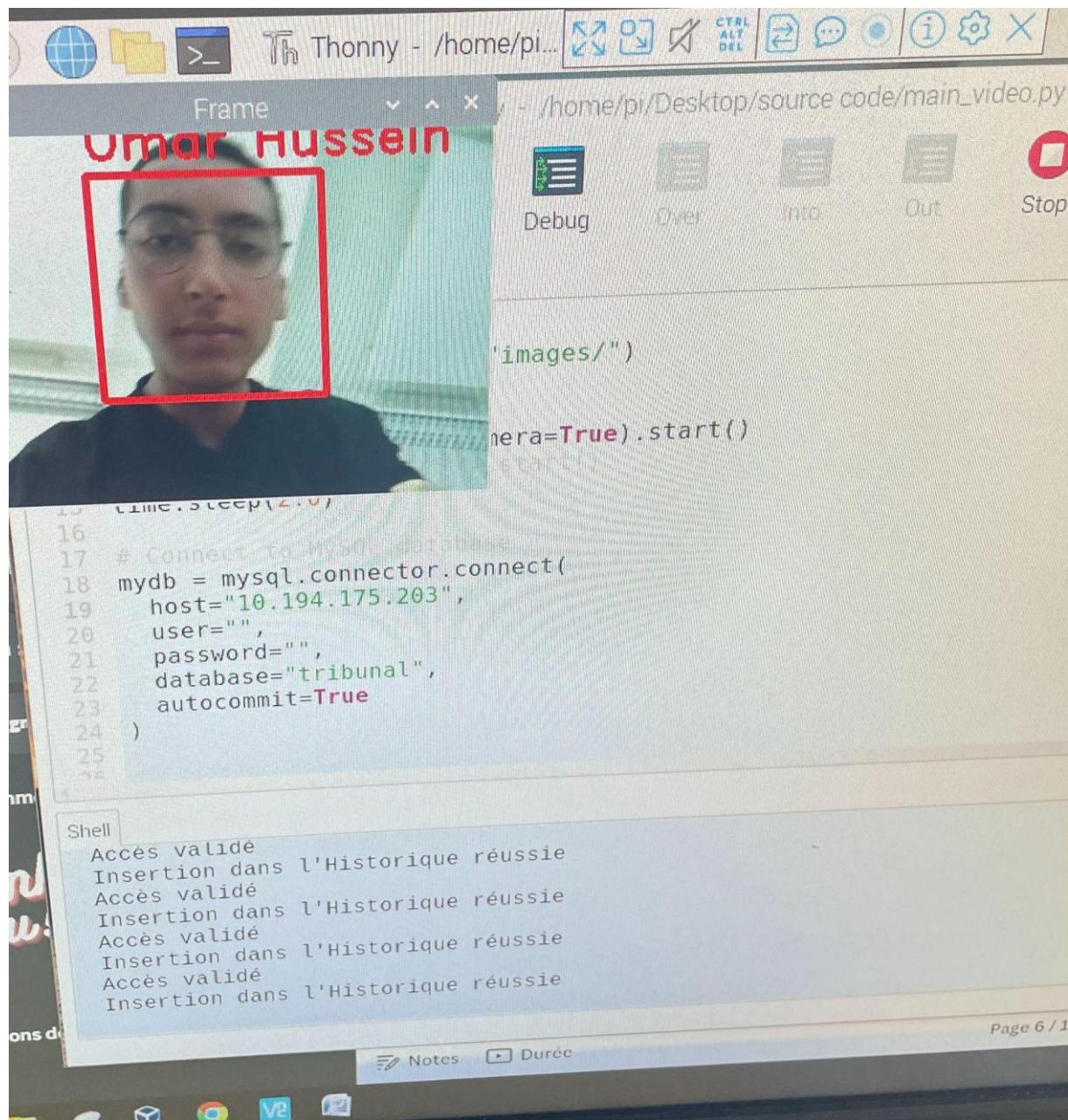
```
# check your memory first
$ free -m
# you need at least a total of 6.5 GB!
# if not, enlarge your swap space as explained earlier
# download the latest version
$ cd ~
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.5.zip
$ wget -O opencv_contrib.zip
https://github.com/opencv/opencv_contrib/archive/4.5.5.zip
# unpack
$ unzip opencv.zip
$ unzip opencv_contrib.zip
# some administration to make live easier later on
$ mv opencv-4.5.5 opencv
$ mv opencv_contrib-4.5.5 opencv_contrib
# clean up the zip files
$ rm opencv.zip
$ rm opencv_contrib.zip
```

qq. Serrure électrique :

rr. Schéma électrique :



Un schéma montrant les branchements électriques de la serrure à la pile en passant par la raspberry .



ss. Test :

On peut voir ici le visage de mon camarades entouré dans un carré rouge avec son nom et prénom au-dessus et en-dessous l'insertion dans l'historique et l'accès a été valider.

tt. Totalité du code :

```
1 import cv2
2 from simple_facerec import SimpleFacerec
3 from imutils.video import VideoStream
4 import time
5 import mysql.connector
6 from datetime import datetime
7
8 # Encode faces from a folder
9 sfr = SimpleFacerec()
10 sfr.load_encoding_images("images/")
11
12 # Load Camera
13 # cap = VideoStream(usePiCamera=True).start()
14 cap = VideoStream(src=1).start()
15 time.sleep(2.0)
16
17 # Connect to MySQL database
18 mydb = mysql.connector.connect(
19     host="localhost",
20     user="root",
21     password="",
22     database="tribunal",
23     autocommit=True
24 )
25
26 # Create cursor
27 mycursor = mydb.cursor(buffered=True)
28
29 # Define resource ID
30 resource_id = 2
31
32 # Loop through video stream
33 while True:
34     frame = cap.read()
35
36     # Detect Faces
37     face_locations, face_names = sfr.detect_known_faces(frame)
38     for face_loc, name in zip(face_locations, face_names):
39         y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
40
```

```

41     # Check if face name is in database and status is 'Validé'
42     sql = "SELECT ID_Visage, Statut FROM Visages WHERE Nom = %s"
43     val = (name,)
44     mycursor.execute(sql, val)
45     result = mycursor.fetchone()
46
47     # Check if the face has access to the resource
48     if result and result[1] == 'Validé':
49         sql = "SELECT * FROM visages_acces_ressources WHERE ID_Visage = %s AND ID_Ressource = %s"
50         val = (result[0], resource_id)
51         mycursor.execute(sql, val)
52         result = mycursor.fetchone()
53         if result:
54             print("Accès validé")
55
56         # Get current date and time
57         now = datetime.now()
58         date_time = now.strftime("%Y-%m-%d %H:%M:%S")
59
60         # Insert new access record in Historique_acces table
61         sql = "INSERT INTO Historique_acces (ID_Demande, ID_Visage, ID_Ressource, Date_acces) VALUES (%s, %s, %s, %s)"
62         val = (None, result[0], resource_id, date_time)
63         mycursor.execute(sql, val)
64
65         print("Insertion dans l'Historique réussie")
66         time.sleep(3)
67
68     else:
69         print("Accès non validé")
70         # Turn on red LED
71         # your code here
72         time.sleep(3)
73
74     else:
75         print("Visage non reconnu")
76         # Turn on red LED
77         # your code here
78         time.sleep(3)
79
80     cv2.putText(frame, name,(x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)

80     cv2.putText(frame, name,(x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
81     cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)
82
83     cv2.imshow("Frame", frame)
84
85     key = cv2.waitKey(1)
86     if key == 27:
87         break
88
89 # Release the camera
90 cap.release()
91 cv2.destroyAllWindows()
92 |

```

uu. Notice d'utilisation :

-Serrure biométrique

Bien se présenter devant la caméra de manière claire et éviter d'avoir une partie du visage cachée.

La porte ne s'ouvre que si vous êtes autorisé à entrer dans la pièce concernée.

## VII. Conclusion

Notre projet de développement d'une solution de contrôle d'accès sécurisé pour le tribunal de grande instance d'Evry a été une expérience extrêmement bénéfique pour notre équipe de trois étudiants. En travaillant ensemble, nous avons pu créer une solution innovante qui simplifie la gestion des accès et améliore la sécurité des ressources du tribunal.

Ce projet nous a permis d'acquérir de précieuses compétences techniques et professionnelles. En tant qu'étudiant en charge de l'Interface de Gestion (IG), j'ai pu développer mes compétences en validation des demandes d'accès, en génération de codes QR, en gestion de l'accès par reconnaissance faciale, en visualisation de l'état d'occupation des ressources et en consultation de l'historique des accès. Mes collègues ont également développé leurs compétences dans des domaines tels que la lecture et le décodage des QR-codes, l'authentification, la commande mécanique des serrures et la gestion des systèmes embarqués.

En plus des compétences techniques, nous avons également renforcé nos compétences en gestion de projet, en communication et en travail d'équipe. Nous avons dû organiser notre travail de manière efficace, respecter des délais et coordonner nos efforts pour assurer le bon déroulement du projet. La collaboration étroite entre nous a favorisé l'échange d'idées, la résolution de problèmes et la prise de décisions conjointes. Ce projet a également eu un impact positif sur notre développement personnel. Nous avons appris à gérer les défis et à faire preuve de résilience face aux obstacles rencontrés. L'aboutissement de ce projet nous a procuré une immense satisfaction et une fierté d'avoir réalisé une solution qui contribuera à améliorer la sécurité et la gestion des accès pour le tribunal d'Evry.

En conclusion, ce projet de contrôle d'accès sécurisé pour le tribunal a été une expérience extrêmement enrichissante pour notre équipe. Nous avons acquis des compétences techniques, renforcé nos compétences en gestion de projet et en communication, et développé notre résilience face aux défis. Nous sommes convaincus que cette expérience positive nous sera bénéfique dans nos futures carrières professionnelles et nous sommes fiers d'avoir contribué à la réalisation d'un projet concret et utile pour le tribunal de grande instance d'Evry.

## VIII. Annexe

### vv. Codes sources

Dans un souci de faciliter l'accès à notre code source, nous avons mis à disposition des liens privés pour chaque partie du projet. Vous trouverez ci-dessous les liens correspondants à chaque composante :

- Interface de gestion : [https://drive.google.com/file/d/13LLgbb8Bkg86f-AE1MQI\\_WQDXRa1sWny/view?usp=sharing](https://drive.google.com/file/d/13LLgbb8Bkg86f-AE1MQI_WQDXRa1sWny/view?usp=sharing)
- Serrure QR Codes : [https://drive.google.com/file/d/1mo89a4GVq5fSB4sCkZGd-9Lun2p8DOqy/view?usp=share\\_link](https://drive.google.com/file/d/1mo89a4GVq5fSB4sCkZGd-9Lun2p8DOqy/view?usp=share_link)
- Serrure Biométrique : [https://drive.google.com/file/d/1Ggl8NI5fpAvZ2DYhqrX8hChrDjvqi1P/view?usp=share\\_link](https://drive.google.com/file/d/1Ggl8NI5fpAvZ2DYhqrX8hChrDjvqi1P/view?usp=share_link)

En cliquant sur les liens respectifs, vous pourrez accéder aux fichiers et au code source associés à chaque composante du projet. Ces liens privés garantissent que seules les personnes autorisées peuvent accéder aux fichiers, préservant ainsi la confidentialité et la sécurité de notre travail.

N'hésitez pas à utiliser ces liens pour explorer notre code source et découvrir en détail l'implémentation de chaque fonctionnalité.

ww.

Photo du Tribunal d'Instance d'Evry

