



GROUPE : DEVOWFS 204

RAPPORT DE STAGE

Développement d'un Système de Gestion des Visiteurs

Présenté par : BELKHAIR Mohamed

Encadré par : Mr. GHENNOU Mohamed et Mr. ESSAMRI Amine

Période du stage : Du 23 Février au 21 Mars

Année universitaire : 2024 - 2025

Remerciements :

Ce travail marque l'aboutissement d'un parcours de formation enrichi par une immersion professionnelle au sein d'un environnement aussi stimulant qu'exigeant. Sa réalisation n'aurait été possible sans l'appui, les orientations méthodologiques et l'implication de plusieurs personnes à qui je tiens à adresser mes plus sincères remerciements.

Je souhaite avant tout exprimer ma profonde gratitude à **M. ESSAMRI Amine**, mon encadrant de stage au sein du **direction de la modernisation et des systèmes d'information**, pour la confiance qu'il m'a accordée, la qualité de son encadrement, ainsi que la pertinence de ses remarques et recommandations, toujours formulées avec rigueur et bienveillance. Sa vision technique et sa maîtrise du domaine ont constitué un cadre d'apprentissage particulièrement formateur.

Mes remerciements vont également à l'**ensemble de l'équipe technique** du Ministère, dont la disponibilité, l'esprit collaboratif et la richesse des échanges ont favorisé une intégration harmonieuse et un travail efficace, tant sur le plan technique qu'humain.

Je tiens également à saluer et remercier chaleureusement **M. GHENNOU Mohamed**, mon encadrant pédagogique au sein de l'**ISTA NTIC**, pour le suivi rigoureux de ce projet, la clarté de ses orientations, et la richesse de ses retours critiques. Son accompagnement constant a fortement contribué à donner à ce rapport une structure cohérente et un niveau d'exigence académique à la hauteur des attentes.

Enfin, j'exprime ma reconnaissance à l'**ensemble du corps professoral** de l'**ISTA NTIC**, pour la transmission des savoirs, mais aussi pour leur implication à former des profils compétents, autonomes et responsables, prêts à relever les défis du monde professionnel.

Dédicace :

À travers ces lignes, je souhaite adresser une pensée sincère à ceux qui ont accompagné chaque étape de ce chemin avec constance, bienveillance et lumière. Ce travail, fruit d'efforts soutenus et de convictions silencieuses, leur est humblement dédié.

À Dieu Tout-Puissant,
source infinie de sagesse, de force et de patience. Que chaque réussite soit un rappel de Sa miséricorde, et chaque difficulté, un pas vers une foi plus ancrée.

À mes parents,
pour leur amour inconditionnel, leur soutien indéfectible, et les valeurs qu'ils m'ont transmises avec exigence et tendresse. Leur présence silencieuse et leur foi en moi ont été le socle de mon engagement.

À mes amis les plus sincères, pour leur présence apaisante, leur écoute bienveillante et leur confiance indéfectible dans les instants de doute comme dans les éclats de réussite.

À toutes celles et ceux qui ont semé, parfois sans le savoir, les graines de ce parcours : je vous adresse ces pages avec respect, affection et gratitude.

Mohamed

Table des matières:

Introduction générale	7
Chapitre I : Présentation de l'Organisation :	8
I.1 Présentation de l'Organisation :	8
I.2 Présentation du Ministère :	9
I.3 DMSI :	9
I.4 Contexte et encadrement du stage :	10
Chapitre II : Étude de l'Existant et Analyse des Besoins	11
II.1 Introduction :	11
II.2 Étude de l'Existant :	11
II.3 Limite de l'Existant :	12
II.4 Objectifs du nouveau système :	12
II.5 Structure du projet :	13
II.5.1 Besoins Fonctionnels :	13
II.5.2 Besoins Non Fonctionnels :	14
Chapitre III : Conception du Système	15
III.1 Introduction :	15
III.2 Architecture Générale du Système :	15
III.2.1 Couche présentation (Front-End) :	15
III.2.2 Couche logique métier (Back-End) :	16
III.2.3 Couche logique métier (Back-End) :	17
III.3 Diagrammes Uml :	17
III.3.1 Diagramme de Cas d'Utilisation :	17
III.3.2 Diagramme de Classes :	19
III.3.3 Diagramme de Séquence :	20

III.4 Modélisation de la base de données :	21
III.4.1 Choix du Système de Gestion de Base de Données :	21
III.4.2 Schéma conceptuel (MCD) :	21
III.4.3 Modèle Logique de Données (MLD) :	23
III.5 Architecture des dossiers :	25
III.5.1 Structure du projet Laravel (Back-end):	25
III.5.2 Structure du projet Frontend :	26
III.6 Sécurité et gestion des accès :	27
III.7 Tests préparatoires et simulations:	28
 Chapitre IV : Développement et Implémentation du Système	 31
IV.1 Introduction :	31
IV.2 Mise en place de l'environnement de développement :	31
IV.3 Développement du Backend (Laravel) :	33
IV.4 Développement du Frontend :	34
IV.5 Connexion à la base de données :	37
IV.6 Gestion des fichiers PDF et CSV :	39
IV.7 Fonctionnalités implémentées par module :	42
IV.8 Captures d'écran et exemples d'interface :	45
 Chapitre IV : Conclusion et Perspectives :	 50
Conclusion :	51
Perspectives :	51
Mot de la fin :	51

Liste des tableaux :

Tableau 1 : Table utilisateurs	24
Tableau 2 : Table visiteurs	24
Tableau 3 : Table rendezvous	25
Tableau 4 : Table entrees_sorties	25
Tableau 5 : Technologies utilisées	32
Tableau 6 : Pages principales développées	37
Tableau 7 : Exportation CSV PDF.....	43
Tableau 9 : Module Authentification & Sécurité	43
Tableau 10 : Module Gestion des Visiteurs	44
Tableau 11 : Module Rendez-vous	44
Tableau 12 : Module Suivi Entrée / Sortie	44
Tableau 13 : Module Tableau de Bord & Statistiques	45
Tableau 14 : Module Exportation	45

Introduction :

À l'heure où la transformation digitale occupe une place centrale dans les stratégies de modernisation des institutions publiques, la mise en place de solutions numériques efficaces devient un enjeu incontournable. Le **Ministère de la Justice**, à travers **la Direction de la Modernisation et des Systèmes d'Information**, s'inscrit dans cette dynamique en déployant des systèmes informatisés visant à optimiser la gestion interne, améliorer la sécurité et faciliter le suivi des processus administratifs.

C'est dans ce contexte que s'inscrit le projet qui m'a été confié dans le cadre de mon stage : le développement d'un **Système de Gestion des Visiteurs**, destiné à moderniser et sécuriser l'enregistrement, le suivi et l'analyse des visites dans les locaux du ministère.

Ce système vise à :

- Numériser l'**enregistrement** des visiteurs et des rendez-vous,
- Suivre les **entrées** et **sorties** en temps réel,
- Générer des **rapports statistiques** fiables,
- Renforcer la **traçabilité** et la **sécurité** des accès.

Le projet repose sur un socle technologique moderne, combinant **React (avec Vite)** et **Tailwind CSS** pour la partie front-end, **Laravel** pour la logique back-end, et **MySQL** pour la gestion de base de données.

Ce rapport a pour objectif de retracer l'ensemble des étapes de ce projet, depuis l'analyse des besoins jusqu'à sa réalisation technique. Il se structure autour de plusieurs chapitres, abordant tour à tour :

- La présentation du cadre institutionnel du stage,
- L'étude de l'existant et la définition des besoins fonctionnels,
- La conception technique et les choix technologiques retenus,
- La phase de développement, de test et de déploiement,
- Les difficultés rencontrées et les perspectives d'évolution du système.

À travers cette expérience, j'ai pu consolider mes compétences en développement web full-stack, tout en contribuant à un projet concret au sein d'une structure publique exigeante et engagée dans la voie du numérique.

Chapitre I :

I.1. Présentation de l'Organisation :

Le stage que j'ai eu l'opportunité de réaliser s'est déroulé au sein de la **Direction de la Modernisation et des Systèmes d'Information (DMSI)**, structure rattachée au **Ministère de la Justice**. Cette direction joue un rôle central dans l'évolution numérique de l'administration judiciaire, en assurant la mise en œuvre de solutions technologiques adaptées aux besoins institutionnels.



Ce chapitre a pour objectif de présenter brièvement le **Ministère de la Justice**, ainsi que le cadre organisationnel spécifique de la **DMSI**, qui m'a accueilli durant cette période de formation professionnelle.

I.2. Présentation du Ministère :



Le **Ministère de la Justice** constitue l'un des piliers fondamentaux de l'État. Il a pour mission principale de garantir le bon fonctionnement de l'appareil judiciaire, le respect des droits fondamentaux et l'application des lois. En tant qu'acteur central de l'autorité judiciaire, il veille à la régulation et à la transparence du système juridique à l'échelle nationale.



Parmi ses missions clés, on retrouve :

- L'organisation et la supervision des juridictions nationales,
- L'élaboration et la réforme des textes de loi,
- La gestion de l'administration judiciaire et pénitentiaire,
- Le renforcement de l'accès à la justice pour tous les citoyens,
- L'accompagnement de la transformation numérique de l'institution judiciaire.

I.3. Dmsi :

La Direction de la Modernisation et des Systèmes d'Information

constitue l'un des moteurs de l'innovation technologique au sein du ministère. Elle est en charge de la conception, du développement, de la maintenance et de la sécurisation des systèmes d'information utilisés par l'ensemble des services du ministère.



Ses principales missions sont :

- Définir et piloter la stratégie numérique du ministère,
- Concevoir des solutions informatiques répondant aux besoins fonctionnels des services internes,
- Assurer la gestion et la sécurité des systèmes d'information,
- Accompagner les utilisateurs dans la transition vers les outils numériques,
- Promouvoir l'interopérabilité et l'efficacité des outils technologiques.

C'est dans cette structure que s'est déroulé mon stage, qui s'inscrivait dans une logique de modernisation des processus liés à la gestion des visiteurs.

I.4. Contexte et encadrement du stage :

Intégré à une équipe technique multidisciplinaire, j'ai participé activement à la mise en place d'une solution numérique permettant de gérer l'enregistrement, le suivi et l'analyse des visiteurs accédant aux locaux du ministère. Ce projet répondait à une problématique concrète exprimée par le département, en lien direct avec ses priorités en matière de transformation digitale et de sécurisation des accès.

Mon travail a été encadré par **M. ESSAMRI Amine**, ingénieur en développement au sein de la DMSI, dont l'accompagnement a été décisif dans la bonne conduite du projet. Le stage s'est déroulé dans un environnement professionnel rigoureux, propice à l'apprentissage, avec un accès structuré aux outils de développement, des échanges réguliers avec l'équipe, et une méthodologie de travail agile.

Chapitre II :

II.1. Introduction :

Avant toute phase de conception ou de développement, il est essentiel de comprendre l'environnement dans lequel s'insérera la solution, d'identifier les limites des outils existants et de définir les attentes des utilisateurs. Cette étape permet d'assurer la cohérence entre les besoins réels du terrain et les choix techniques envisagés.

II.2. Étude de l'Existant :

Avant la mise en œuvre du système proposé, la gestion des visiteurs au sein du ministère se faisait selon des méthodes principalement manuelles ou semi-numériques (tableurs, registres physiques, saisie ponctuelle).

Parmi les pratiques observées :

- Enregistrement manuel des visiteurs sur papier ou Excel,
- Absence de centralisation des données de fréquentation,
- Suivi approximatif des rendez-vous et des horaires d'accès,
- Manque de visibilité sur les visiteurs en cours ou passés.



II.3. Limite de l'Existant :

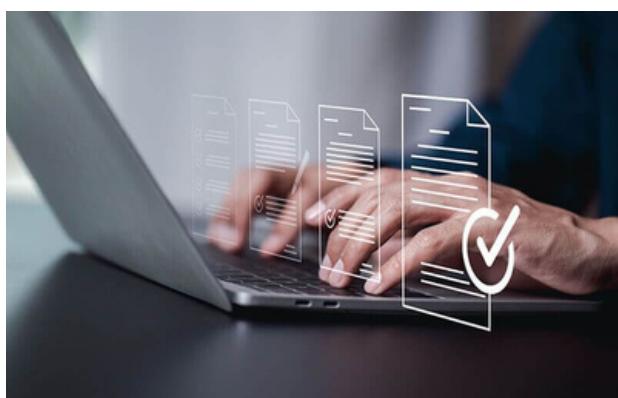
Cette organisation, bien qu'utilisée depuis plusieurs années, présentait plusieurs limitations importantes :

- ✗ Temps de traitement élevé pour enregistrer chaque visiteur ;
- ✗ Manque de traçabilité des données (aucun historique consolidé) ;
- ✗ Failles de sécurité possibles (absence de vérification croisée ou de validation de l'identité) ;
- ✗ Aucune capacité d'analyse (pas de statistiques, ni d'indicateurs de fréquentation) ;
- ✗ Risque de perte de données, notamment pour les documents papier non archivés numériquement.

II.4. Objectifs du nouveau système :

Face à ces constats, la DMSI a exprimé le besoin de mettre en place une solution numérique moderne permettant :

- ✓ Un enregistrement structuré, rapide et sécurisé des visiteurs ;
- ✓ Une gestion fluide des rendez-vous avec affectation par agent et statut (confirmé, annulé, en attente) ;
- ✓ Un suivi en temps réel des entrées et sorties dans les locaux ;
- Une consultation de l'historique des visites par période ou par personne ;
- ✓ La génération automatique de rapports statistiques ;
- ✓ Une interface utilisateur claire et intuitive.



II .5. Spécification des Besoins :

L'analyse fonctionnelle permet de traduire les attentes exprimées par les utilisateurs en fonctionnalités concrètes. Elle constitue une base solide pour la conception technique du système. On distingue ici les besoins fonctionnels (ce que le système doit faire) des besoins non fonctionnels (comment le système doit se comporter).

II.5.1. Besoins Fonctionnels :

Le système doit permettre aux utilisateurs d'exécuter un ensemble d'actions précises, organisées selon leur rôle (administrateur ou agent). Voici les principales fonctionnalités attendues :

- ◆ Authentification et gestion des accès
 - Connexion sécurisée via identifiants ;
 - Gestion des rôles : administrateur, agent, utilisateur ;
 - Redirection automatique vers l'espace correspondant au rôle.
- ◆ Gestion des visiteurs
 - Création, modification et suppression de fiches visiteurs ;
 - Saisie des données personnelles (nom, prénom, CIN, téléphone, etc.) ;
 - Possibilité de rechercher et filtrer des visiteurs par date, nom ou statut.
- ◆ Gestion des rendez-vous
 - Planification d'un rendez-vous avec affectation à un agent ;
 - Visualisation du planning et des disponibilités ;
 - Modification, annulation ou confirmation des rendez-vous ;
 - Statut du rendez-vous : en attente, confirmé, annulé.
- ◆ Suivi des entrées et sorties
 - Enregistrement de l'heure d'entrée/sortie d'un visiteur ;
 - Validation automatique selon rendez-vous existant ;
 - Identification par saisie manuelle ou QR code.
- ◆ Statistiques et rapports
 - Génération de tableaux de bord : nombre de visites par jour/semaine/mois, répartition par service ou motif ;
 - Export des données en format PDF ou CSV ;
 - Visualisation graphique (courbes, histogrammes).

II .5.2. Besoins Non Fonctionnels :

Outre les fonctionnalités, le système doit répondre à un ensemble d'exigences techniques et qualitatives :

◆ Sécurité

- Protection des accès via JWT token et middleware ;
- Séparation des priviléges selon les rôles ;
- Stockage sécurisé des données (hashing des mots de passe, validation côté serveur).



◆ Ergonomie et accessibilité

- Interface intuitive avec React + Tailwind CSS ;
- Adaptation aux différents formats d'écran (responsive design) ;
- Utilisation d'icônes, d'indicateurs visuels et d'animations légères.



◆ Performance

- Temps de chargement optimisé grâce à Vite ;
- Requêtes asynchrones (Axios, Fetch) pour une meilleure fluidité ;
- Gestion efficace des volumes de données avec pagination et filtres.



◆ Fiabilité et maintenabilité

- Architecture modulaire et évolutive ;
- Code structuré avec Laravel (MVC) et composants réutilisables côté React ;
- Tests unitaires (PHPUnit, Jest) pour assurer la stabilité.



Chapitre III:

III.1. Introduction :

Une fois les besoins identifiés et analysés, il est essentiel de passer à une phase de **modélisation** permettant de visualiser la structure et le fonctionnement du futur système. Cette phase repose principalement sur l'utilisation de diagrammes UML et la définition d'une architecture technique claire. Elle constitue la passerelle entre la réflexion fonctionnelle et l'implémentation concrète.

III.2. Architecture Générale du Système :

Le système repose sur une **architecture à trois couches (ou architecture 3-tiers)**, permettant une séparation logique entre l'interface utilisateur, la logique métier et la gestion des données. Ce modèle favorise la clarté, la maintenabilité, la réutilisabilité du code et la sécurité.

III.2.1. Couche présentation (Front-End) :

→ **Technologie utilisée** : React.js avec Vite + Tailwind CSS



React.js est une bibliothèque JavaScript développée par Facebook, utilisée pour construire des interfaces utilisateur dynamiques, performantes et modulaires. Son modèle basé sur **les composants réutilisables** et son système de **rendu virtuel (Virtual DOM)** permettent une excellente réactivité et une manipulation fluide de l'interface.



Vite est un outil de build moderne ultra-rapide, parfaitement adapté à React. Il offre un démarrage quasi-instantané du projet et un **rechargement à chaud (Hot Module Replacement)** performant. Cela accélère considérablement le développement en local.



Tailwind CSS est un framework CSS utilitaire qui permet de **styliser les interfaces rapidement et efficacement** via des classes prédéfinies. Il facilite la création de designs épurés, modernes, **100% responsives**, tout en maintenant une bonne lisibilité du code.

III.2.2. Couche logique métier (Back-End) :



Laravel est un framework PHP populaire, reconnu pour sa structure élégante, sa **syntaxe claire** et son **écosystème riche**. Il repose sur le modèle **MVC (Modèle-Vue-Contrôleur)**, ce qui permet de structurer proprement l'application, en séparant la logique métier de la présentation.

Pourquoi ce choix ?

→ **Laravel** permet un développement rapide, sécurisé et structuré d'APIs robustes. Il est parfaitement adapté à un environnement professionnel et assure une bonne scalabilité.

III.2.3. Couche données (Base de données) :



MySQL est un système de gestion de base de données relationnelle open-source largement utilisé dans les projets web.

Pourquoi ce choix ?

→ **MySQL** est fiable, rapide et bien documenté, avec une courbe d'apprentissage maîtrisée. Il est particulièrement adapté aux applications nécessitant des relations entre plusieurs entités, comme ici : visiteurs, rendez-vous, agents, etc.

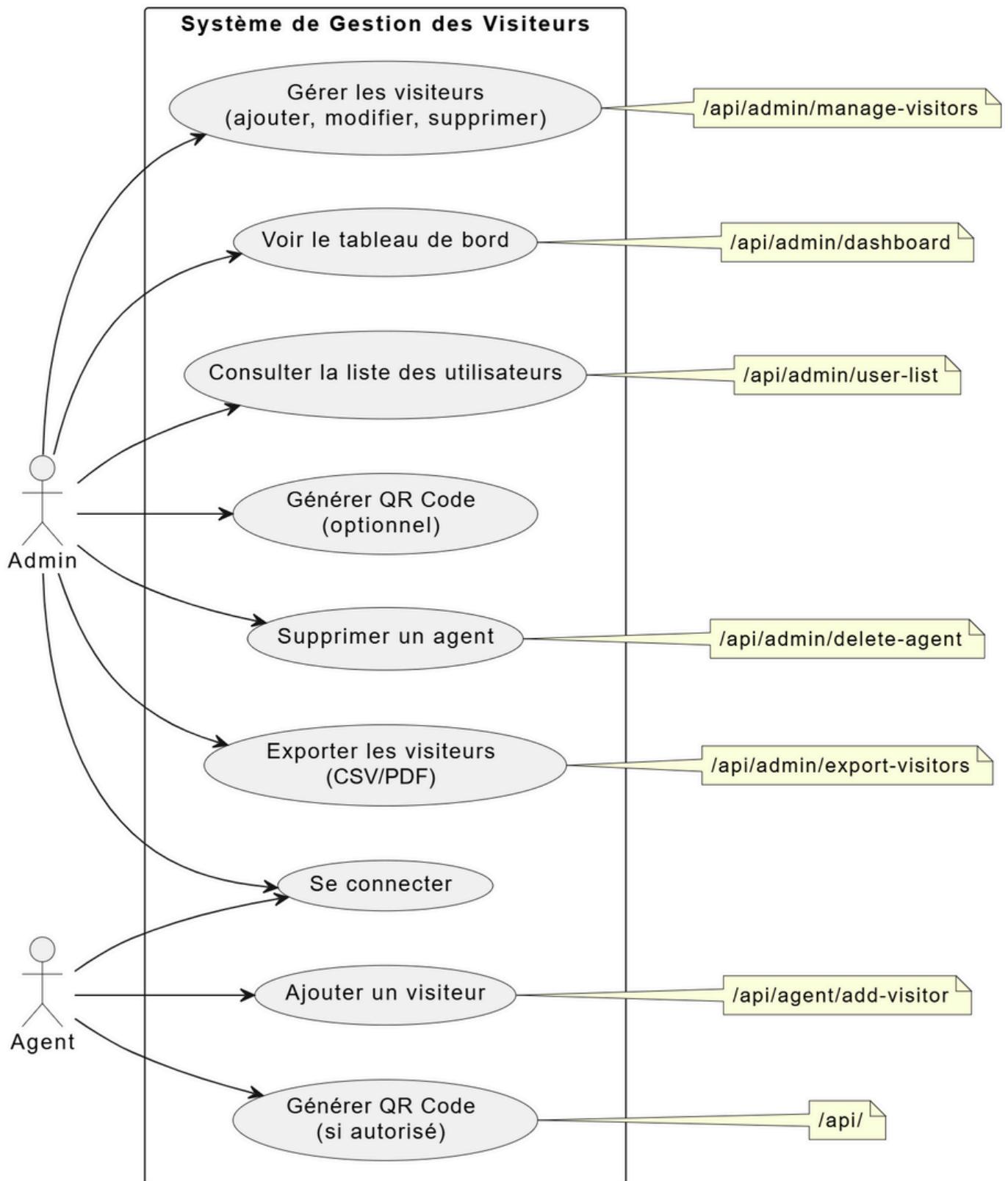
III.3. Diagrammes Uml :



La modélisation **UML** (Unified Modeling Language) permet de représenter graphiquement la structure et le comportement du système avant sa mise en œuvre. Elle facilite la compréhension globale du projet et sert de référence tout au long du développement.

III.3.1. Diagramme de Cas d'Utilisation :

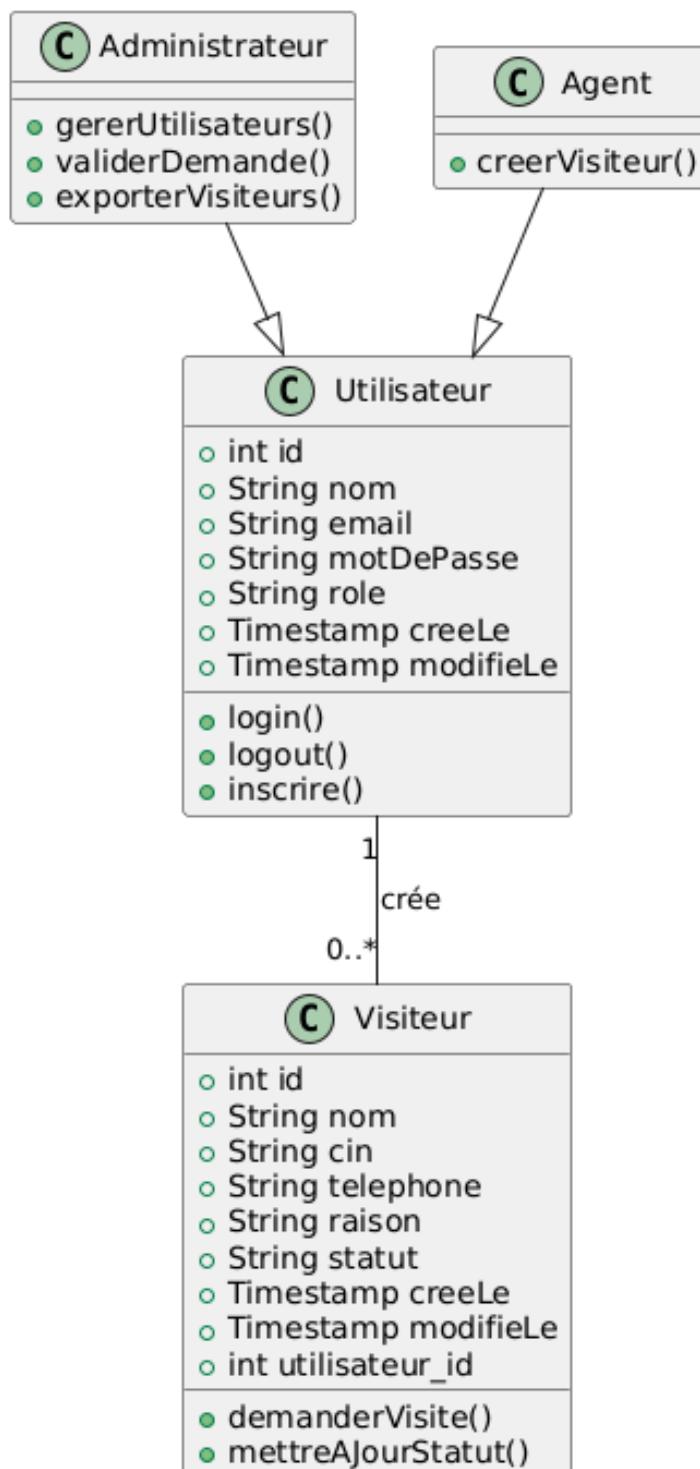
Ce diagramme illustre les fonctionnalités principales du système ainsi que les différents acteurs (visiteur, agent administratif, administrateur). Il permet de visualiser les interactions entre les utilisateurs et les cas d'usage du système, tels que : l'enregistrement d'une demande, la validation par un agent, ou encore la gestion des utilisateurs par l'administrateur.



III.3.2. Diagramme de Classes :

Ce diagramme illustre les fonctionnalités principales du système ainsi que les différents acteurs (visiteur, agent administratif, administrateur). Il permet de visualiser les interactions entre les utilisateurs et les cas d'usage du système, tels que : l'enregistrement d'une demande, la validation par un agent, ou encore la gestion des utilisateurs par l'administrateur.

Diagramme de Classes - Système de Gestion des Visiteurs

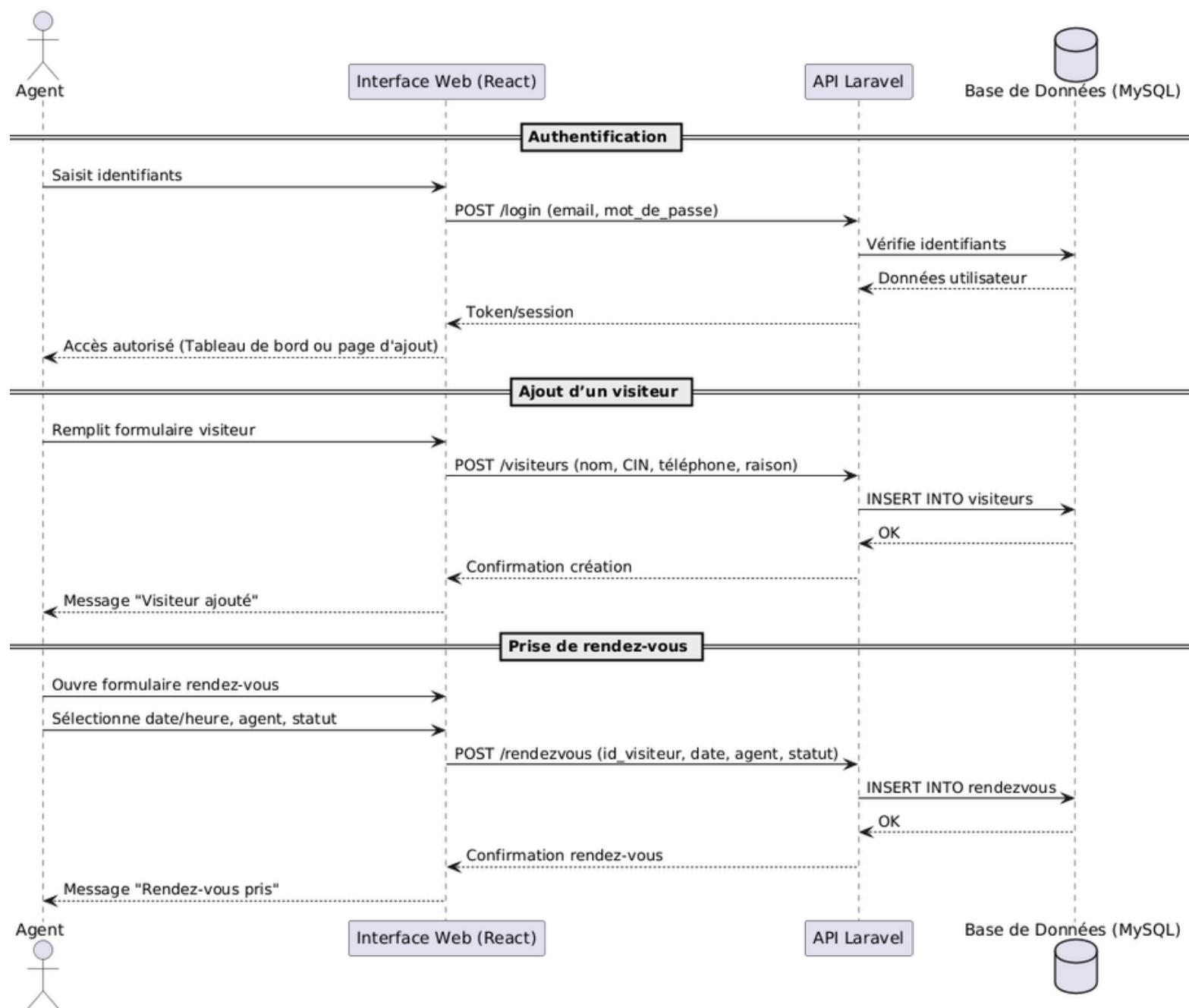


III.3.3. Diagramme de Séquence :

Ce diagramme illustre la chronologie des échanges entre les composants du système lors d'un scénario spécifique.

Exemple : enregistrement d'un visiteur

1. L'agent initie la demande d'ajout ;
2. Le système affiche un formulaire ;
3. L'agent soumet les données ;
4. Le système valide les informations ;
5. Les données sont enregistrées dans la base ;
6. Un message de confirmation est retourné.



III.4. Modélisation de la base de données :

La base de données constitue **le cœur du système**, permettant de stocker, organiser et interroger efficacement toutes les informations nécessaires à la gestion des visiteurs. La conception de cette base suit un processus rigoureux de modélisation pour assurer **la cohérence, la sécurité et la performance** du système.

III.4.1. Choix du Système de Gestion de Base de Données :

Le système utilise **MySQL**, un SGBD relationnel open-source largement adopté pour sa robustesse, sa compatibilité avec plusieurs langages, et sa facilité de gestion. Il permet :

- Une gestion structurée des données via des relations entre tables ;
- Un support performant pour les requêtes SQL complexes ;
- Une intégration fluide avec **Express.js**, le backend choisi.

III.4.2. Schéma conceptuel (MCD) :

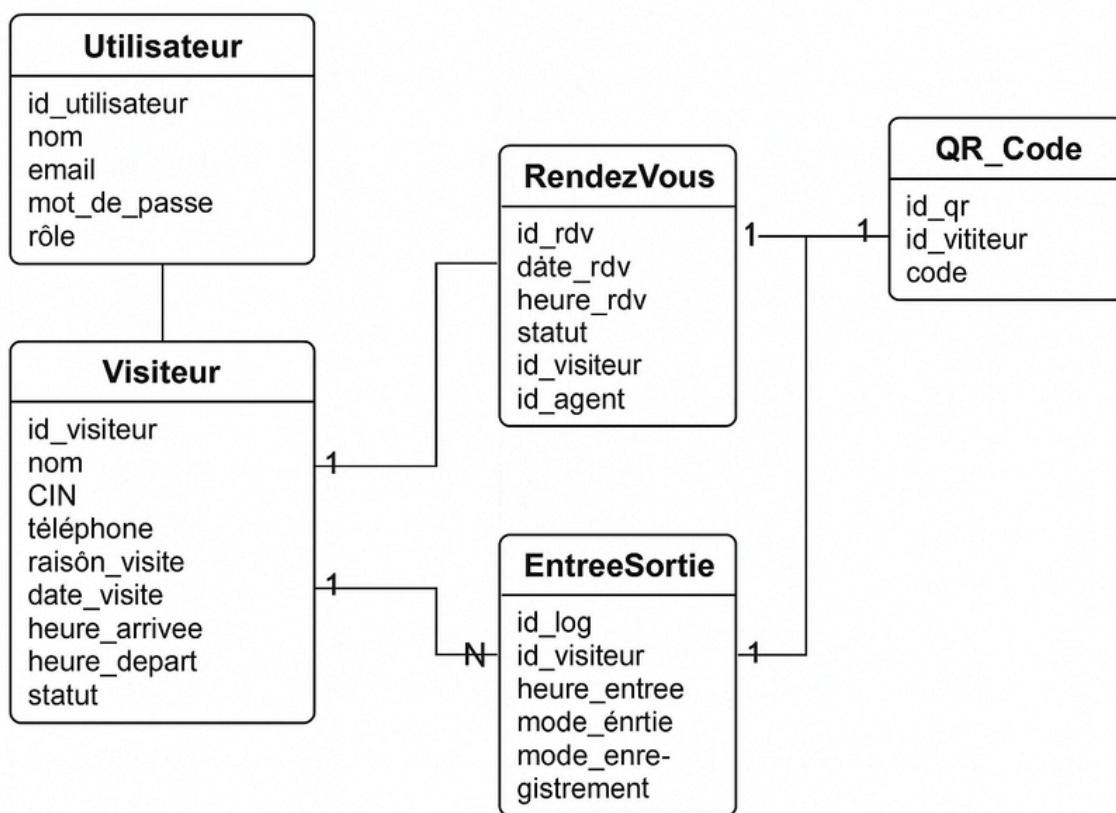
Le **Modèle Conceptuel de Données** représente les entités principales du système, leurs attributs, et les relations logiques entre elles. Voici un aperçu des entités identifiées :

- Utilisateur
 - id_utilisateur (Identifiant unique)
 - nom
 - email
 - mot_de_passe
 - rôle (admin ou agent)
- Visiteur
 - id_visiteur (Identifiant unique)
 - nom
 - CIN
 - téléphone
 - raison_visite
 - date_visite
 - heure_arrivee
 - heure_depart
 - statut (en_attente, en_cours, terminée)

- RendezVous
 - id_rdv (Identifiant unique)
 - date_rdv
 - heure_rdv
 - statut (en_attente, confirmé, annulé)
 - id_visiteur (FK vers Visiteur)
 - id_agent (FK vers Utilisateur)
- EntreeSortie
 - id_log (Identifiant unique)
 - id_visiteur (FK)
 - heure_entree
 - heure_sortie
 - mode_enregistrement (manuel ou QR code)
- QR_Code (facultatif)
 - id_qr
 - id_visiteur (FK)
 - code

Relations :

- Un Utilisateur peut gérer plusieurs Visiteurs.
- Un Visiteur peut avoir plusieurs RendezVous.
- Un RendezVous est associé à un Visiteur et à un Agent.
- Un Visiteur peut avoir plusieurs entrées/sorties enregistrées.
- Un Visiteur peut posséder un QR Code unique (relation 1-1).



III.4.3. Modèle Logique de Données (MLD) :

Le MLD détaille les tables avec leurs attributs, types de données et relations (clés primaires et étrangères) :

Table : utilisateurs

Champ	Type	Description
id	INT (PK)	Clé primaire
name	VARCHAR	Nom complet de l'utilisateur
email	VARCHAR	Identifiant de connexion
password	VARCHAR	Mot de passe chiffré
role	ENUM	admin ou agent
created_at	TIMESTAMP	Date de création

Table : visiteurs

Champ	Type	Description
id	INT (PK)	Clé primaire
name	VARCHAR	Nom complet
cin	VARCHAR	Numéro de carte d'identité
phone	VARCHAR	Numéro de téléphone
reason	TEXT	Motif de la visite
created_at	TIMESTAMP	Enregistrement

Table : rendezvous

Champ	Type	Description
id	INT (PK)	Clé primaire
visitor_id	INT (FK)	Référence au visiteur
agent_id	INT (FK)	Référence à l'agent concerné
date	DATE	Date prévue
time	TIME	Heure prévue
status	ENUM	en_attente , confirmé , annulé

Table : entrees_sorties

Champ	Type	Description
id	INT (PK)	Clé primaire
visitor_id	INT (FK)	Visiteur concerné
checkin_time	DATETIME	Heure d'entrée
checkout_time	DATETIME	Heure de sortie

Contraintes et sécurité

- Clés étrangères (`visitor_id`, `agent_id`) assurent l'intégrité référentielle.
- Contraintes d'unicité sur les champs sensibles (CIN, email).
- Middleware Laravel pour **restreindre l'accès aux routes** en fonction du rôle.

III.5. Architecture des dossiers :

L'application développée repose sur une architecture **full-stack** avec deux projets distincts : le **frontend** développé avec **React**, et le **backend** développé avec **Laravel**. Cette séparation favorise la modularité, la maintenabilité et la scalabilité du système.

III.5.1. Structure du projet Laravel (Back-end) :

Le **backend**, basé sur le framework **Laravel**, est nommé gestion-visiteurs et organisé de manière modulaire en suivant les bonnes pratiques de l'architecture **MVC** (Modèle-Vue-Contrôleur). Voici un aperçu de la structure principale :

bash

```
gestion-visiteurs/
|
|   app/                                # Logique métier : modèles, contrôleurs, middlewares
|   |   Http/
|   |   |   Controllers/                 # Contrôleurs API comme AuthController, VisitorController, etc.
|   |   |   Middleware/                # Middlewares de gestion des rôles (Admin, Agent)
|   |   |   Models/                   # Modèles comme User, Visitor, Appointment
|
|   routes/
|   |   api.php                         # Toutes les routes d'API REST (auth, visiteurs, dashboard...)
|
|   database/
|   |   migrations/                  # Création et mise à jour des tables
|   |   seeders/                      # Données de test ou initiales
|
|   config/                            # Configuration de Laravel (auth, database...)
|   public/                            # Point d'entrée HTTP (index.php)
|   resources/                         # Vues, composants Blade (non utilisés ici car API)
|   tests/                             # Tests unitaires et fonctionnels (PHPUnit)
|   .env                               # Variables d'environnement (DB, clé API...)
|   composer.json                     # Dépendances PHP (Laravel, Sanctum...)
```

- **Authentification** : via **Laravel Sanctum**
- **Sécurité** : Middleware selon le rôle utilisateur (admin, agent)
- **API REST** : exposée via routes/api.php
- **Fonctionnalités** : CRUD visiteurs, utilisateurs, statistiques, export CSV

III.5.2. Structure du projet Frontend :

Le **frontend** est développé avec **React** et utilise **Vite** comme environnement de bundling pour une compilation rapide, ainsi que **Tailwind CSS** pour le design de l'interface utilisateur. Cette combinaison permet une interface moderne, responsive et facile à maintenir. L'organisation des fichiers respecte une logique modulaire et claire :

```

graphql

frontend/
|
|   public/           # Fichiers statiques (index.html, favicon)
|
|   src/
|       components/  # Composants réutilisables (Navbar, Sidebar, Cards, Formulaire...)
|       pages/        # Pages principales (Login, Dashboard, AddVisitor, Stats...)
|       services/     # Fichiers pour les appels API via Axios (authService.js, visitorService.js...)
|       context/      # Contexte React pour gérer l'état global (AuthContext, RoleContext...)
|       utils/        # Fonctions utilitaires (format de date, gestion des rôles...)
|       App.jsx       # Composant racine de l'application, inclut le Router
|       main.jsx      # Point d'entrée, rend l'application dans le DOM
|
|   tailwind.config.js # Configuration personnalisée de Tailwind
|   postcss.config.js # Configuration de PostCSS (nécessaire pour Tailwind)
|   vite.config.js    # Configuration Vite (dev server, alias, plugins...)
|   package.json      # Dépendances JavaScript du projet
|   .gitignore / README.md # Fichiers de configuration et documentation

```

Outils et technologies utilisés :

- **React Router Dom** : pour la gestion des routes et des accès protégés.
- **Tailwind CSS** : pour un design fluide, responsive, basé sur des classes utilitaires.
- **Axios** : pour l'envoi des requêtes API au backend Laravel.
- **Contexte React** : pour centraliser la gestion de l'état utilisateur (connexion, rôle).
- **Protected Routes** : sécurisent l'accès selon le rôle (Admin vs Agent).

III.6. Sécurité et gestion des accès :

La **sécurité** est un aspect fondamental du système de gestion des visiteurs, étant donné la sensibilité des données traitées (informations personnelles, documents administratifs, etc.). Plusieurs mesures ont été mises en œuvre afin de garantir la protection du système et des utilisateurs.

1. Authentification et Autorisation

- **JWT** (JSON Web Token) est utilisé pour gérer l'authentification. Lorsqu'un utilisateur (admin ou agent) se connecte, un jeton est généré et envoyé au frontend pour être stocké de manière sécurisée (souvent dans localStorage ou sessionStorage).
- Le backend vérifie ce jeton pour chaque requête protégée afin de s'assurer que l'utilisateur est bien authentifié.



2. Gestion des rôles

Le système distingue deux types d'utilisateurs :

- **Administrateur principal** : a tous les droits (gestion des comptes agents, traitement des demandes, accès aux rapports).
- **Agents administratifs** : peuvent consulter et traiter les demandes de visiteurs.

Des middlewares de vérification de rôle sont mis en place pour restreindre l'accès aux routes selon les priviléges de l'utilisateur.



AGENT



ADMIN

3. Validation des données

Toutes les entrées utilisateurs (formulaires, API) sont validées côté client et côté serveur afin d'éviter les injections SQL, les XSS (cross-site scripting), et autres attaques.

4. Protection des fichiers

Les fichiers PDF téléchargés sont stockés dans un répertoire sécurisé, non directement accessible depuis l'extérieur. L'accès aux fichiers passe obligatoirement par une route authentifiée.

5. Hashing des mots de passe

Les mots de passe sont **hachés** à l'aide de l'algorithme **bcrypt** avant d'être stockés dans la base de données, garantissant ainsi qu'ils ne sont jamais stockés en clair.

Password → bcrypt → #23?k\$4&jJnc9

6. Politique de sessions

Une expiration automatique des sessions est définie pour renforcer la sécurité en cas d'inactivité prolongée.

III.7. Tests préparatoires et simulations:

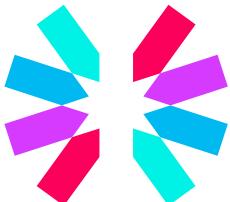
Avant de procéder au développement complet du système de gestion des visiteurs, une série de tests préparatoires et de simulations a été menée. L'objectif principal était de **vérifier la faisabilité technique** du projet, **d'évaluer les outils choisis** et de s'assurer de leur **compatibilité avec les besoins fonctionnels** identifiés.

1. Tests de faisabilité des technologies retenues

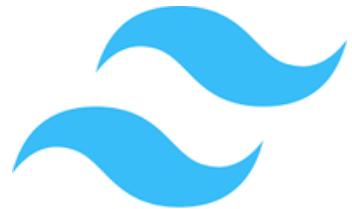
Des modules isolés ont été créés pour tester les principaux composants techniques du système :

- **Connexion entre React.js et Express.js** : un mini projet a été monté pour tester le dialogue entre frontend et backend via des requêtes HTTP (GET/POST).

- **Connexion Express → MySQL** : test de lecture/écriture dans une base de données MySQL, pour s'assurer de la fluidité et de la fiabilité des échanges.
- **JWT Authentication** : une simulation d'authentification sécurisée par jetons a été réalisée pour valider l'utilisation de jsonwebtoken côté serveur et la gestion du token côté client.
- **Téléversement de fichiers** : test d'un petit module Express permettant l'envoi et la sauvegarde de fichiers (PDF) dans un répertoire local avec multer.
- **Responsive UI avec Tailwind CSS** : création de maquettes testées sur divers formats d'écran pour valider l'ergonomie et l'adaptabilité du design.



Express **JS**



2. Simulations fonctionnelles

En parallèle, des **scénarios fonctionnels simplifiés** ont été simulés pour visualiser le comportement attendu du système :

- **Création d'un visiteur** via un formulaire React lié à une API Express simulée.
- **Affichage d'une liste dynamique** (tableau de visiteurs) avec données mockées.
- **Système de rôles simulé** (admin/agent) avec redirection conditionnelle dans React.

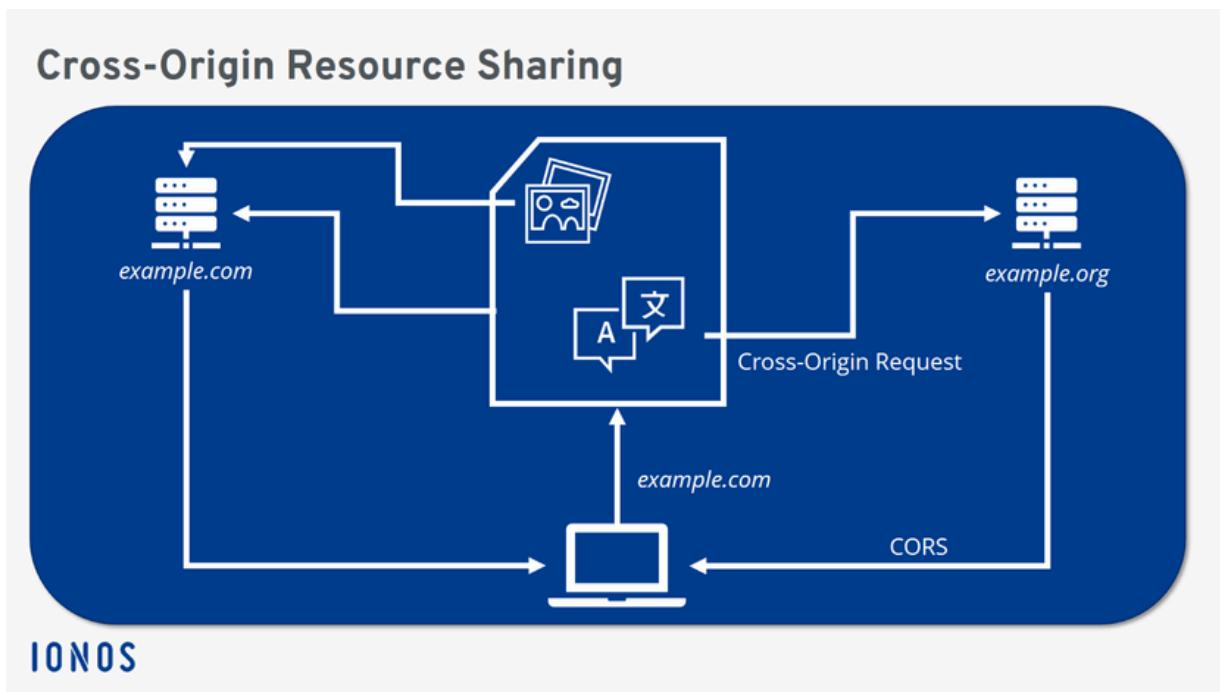
Ces simulations ont permis de valider :

- Le découpage des composants React,
- Le fonctionnement des routes Express,
- Le besoin réel en structure de base de données.

3. Résultats et ajustements

Les tests préparatoires ont permis :

- De confirmer la **pertinence des choix technologiques** ;
- D'identifier certains besoins techniques spécifiques (gestion du CORS, middleware de rôle, architecture des dossiers) ;
- De poser les **fondations du projet final**, avec plus de clarté et d'assurance technique.



Chapitre IV:

IV.1. Introduction :

Cette partie du rapport présente le déroulement technique de la conception du système. Le développement s'est appuyé sur une architecture **full-stack**, avec un **backend Laravel** pour la gestion des données et de la sécurité, et un **frontend React** avec **Tailwind CSS** pour une interface utilisateur moderne et responsive.

Le système a été développé en local dans un environnement de test, en respectant une méthodologie progressive : mise en place de l'environnement, création de la base de données, développement des API, mise en œuvre des composants front-end, puis intégration et tests.

IV.2. Mise en place de l'environnement de développement :

Catégorie	Outil / Techno	Rôle
Front-end	React.js	Interface utilisateur
	Tailwind CSS	Stylisation rapide et responsive
	Vite	Bundling léger et rapide
Back-end	Laravel	Framework PHP pour l'API
	Sanctum	Authentification avec tokens
	MySQL	Base de données relationnelle
Outils	Postman	Tests API
	Git / GitHub	Suivi de version
	VS Code	Environnement de développement



Étapes d'installation :

- **Backend :**

```
bash

composer create-project laravel/laravel gestion-visiteurs
cp .env.example .env
php artisan key:generate
php artisan migrate
```

- **Frontend :**

```
bash

npm create vite@latest frontend --template react
cd frontend
npm install
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Les deux projets ont été lancés sur des serveurs distincts :

- Laravel API sur <http://localhost:8000>
- React sur <http://localhost:5173>

IV.3. Développement du Backend (Laravel) :

Le backend repose sur une architecture RESTful. Chaque fonctionnalité est encapsulée dans des **routes API**, **contrôleurs** et **middlewares**.

a) Routes API (`routes/api.php`)

Les routes principales ont été définies pour gérer :

- L'authentification (`/login`, `/logout`, `/user`)
- Les visiteurs (`/visitors`)
- Les rendez-vous (`/appointments`)
- Les statistiques (`/admin /dashboard`, `/admin /visitor-stats`)
- L'export CSV (`/admin /export-visitors`)

b) Contrôleurs principaux :

- **AuthController** : gère la connexion, la déconnexion et le token.
- **VisitorController** : CRUD des visiteurs avec validation.
- **AppointmentController** : gestion des rendez-vous.
- **StatsController** : agrégation des données statistiques.

Exemple d'un extrait :

```
php

public function login(Request $request)
{
    $credentials = $request->only('email', 'password');
    if (!Auth::attempt($credentials)) {
        return response()->json(['message' => 'Invalid credentials'], 401);
    }
    return $request->user()->createToken('api-token')->plainTextToken;
}
```

c) Middlewares

Deux middlewares principaux sont utilisés :

- **auth:sanctum** : pour s'assurer que l'utilisateur est connecté
- **CheckAdmin / CheckAgent** : pour restreindre les routes selon le rôle

Extrait middleware :

```
php

if ($request->user()->role !== 'admin') {
    return response()->json(['error' => 'Unauthorized'], 403);
}
```

IV.4. Développement du Frontend :

Le développement du front-end a été réalisé avec **React.js** pour la logique d'interface et **Tailwind CSS** pour le style. Le projet est structuré de manière modulaire avec des composants réutilisables, une gestion claire des routes, et une séparation des responsabilités.

A. Initialisation et configuration

Le projet a été initialisé avec **Vite**, un bundler rapide et léger adapté à React. Tailwind CSS a été installé avec PostCSS pour une stylisation utilitaire et responsive.

Dans le fichier `tailwind.config.js`, les chemins des fichiers ont été configurés pour inclure tous les composants React :

```
js
content: ["./index.html", "./src/**/*.{js,ts,jsx,tsx}"]
```

B. Authentification et gestion des rôles

Un système d'authentification a été mis en place avec :

- Saisie des identifiants dans un formulaire (`Login.jsx`)
- Envoi à l'API Laravel pour vérification
- Stockage du **token Sanctum** et du rôle dans le contexte React
- Redirection automatique selon le rôle

Extrait de logique :

```
jsx
useEffect(() => {
  if (user?.role === 'admin') {
    navigate("/dashboard");
  } else {
    navigate("/add-visitor");
  }
}, [user]);
```

Les routes sont protégées par un composant **ProtectedRoute** qui vérifie la connexion et le rôle :

```
jsx
<Route
  path="/dashboard"
  element={user?.role === "admin" ? <Dashboard /> : <Navigate to="/add-visitor" />}
/>
```

C. Pages principales développées

Page	Description
/login	Formulaire de connexion sécurisé
/dashboard	Tableau de bord avec statistiques (admin uniquement)
/add-visitor	Formulaire pour ajouter un visiteur (admin et agent)
/visitors	Liste complète des visiteurs avec filtres et suppression (admin)
/appointments	Gestion des rendez-vous associés à chaque visiteur (admin)

D. Utilisation de Tailwind CSS

Toutes les interfaces sont stylisées avec **Tailwind CSS**, permettant :

- Une grande **réactivité** sur tous types d'écrans
- Un design **propre et cohérent**
- Une rapidité de développement

Exemple d'un bouton stylisé :

```
jsx

<button className="bg-blue-600 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">
  Ajouter
</button>
```

E. Appels API (Axios)

Les appels API vers Laravel sont centralisés dans des services (authService.js, visitorService.js...) et utilisent Axios avec gestion des tokens.

Exemple :

```
js

axios.post("/api/login", { email, password })
  .then((res) => {
    setToken(res.data.token);
  });
}
```

IV.5. Connexion à la base de données :

L'application repose sur une **base de données MySQL**, utilisée pour stocker toutes les informations relatives aux visiteurs, utilisateurs, rendez-vous, et enregistrements d'entrées/sorties. Cette base est gérée côté backend avec **Laravel**, qui offre une intégration fluide via **Eloquent ORM**.

A. Configuration de la connexion (Laravel)

La configuration s'effectue dans le fichier `.env` situé à la racine du projet Laravel. Les paramètres principaux définissent l'hôte, le nom de la base de données, l'utilisateur et le mot de passe.

```
env

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=visitors_db
DB_USERNAME=root
DB_PASSWORD=
```

Une fois la configuration effectuée, la connexion est vérifiée via la commande artisan :

```
bash

php artisan migrate
```

Cette commande exécute les fichiers de migration situés dans le dossier **database/migrations**, permettant de créer automatiquement les tables dans la base de données.

B. Structure des tables principales

Voici un aperçu des principales tables créées à l'aide de migrations Laravel :

1. users

Contient les informations des agents et administrateurs :

```
php

Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->string('password');
    $table->enum('role', ['admin', 'agent']);
    $table->timestamps();
});

});
```

2. visitors

Enregistre les visiteurs avec leurs informations personnelles :

```
php
Copy Edit

Schema::create('visitors', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('cin')->unique();
    $table->string('phone');
    $table->text('reason');
    $table->timestamps();
});
```

3. appointments

Relie un visiteur à un rendez-vous, éventuellement avec un agent :

```
php
```

```
Schema::create('appointments', function (Blueprint $table) {
    $table->id();
    $table->foreignId('visitor_id')->constrained();
    $table->foreignId('agent_id')->nullable()->constrained('users');
    $table->date('date');
    $table->time('time');
    $table->enum('status', ['en_attente', 'confirmé', 'annulé']);
    $table->timestamps();
});
```

4. checkins

Relie un visiteur à un rendez-vous, éventuellement avec un agent :

```
php
```

```
Schema::create('checkins', function (Blueprint $table) {
    $table->id();
    $table->foreignId('visitor_id')->constrained();
    $table->timestamp('checkin_time');
    $table->timestamp('checkout_time')->nullable();
    $table->timestamps();
});
```

C. Sécurité et intégrité des données

- Utilisation de **clés étrangères** pour lier les tables entre elles.
- Validation des champs côté backend (via les **Request** personnalisées).
- Unicité imposée sur des champs sensibles comme **CIN** ou **email**.
- Requêtes sécurisées via **Eloquent ORM** pour éviter les injections SQL.

IV.6. Gestion des fichiers PDF et CSV :

L'application offre aux administrateurs la possibilité d'**exporter les données des visiteurs** sous deux formats : **PDF** pour une version imprimable et officielle, et **CSV** pour l'analyse et le traitement dans des logiciels comme Excel. Ces exports facilitent la **gestion documentaire** et améliorent la **tracabilité** des visites.

A. Objectifs des exportations

- Fournir un **rappor t clair et lisible** (PDF) pour l'impression ou l'archivage.
- Permettre **l'analyse des données** (CSV) dans des outils de type Excel/Google Sheets.
- Offrir à l'admin une **interface simple et rapide** pour générer ces fichiers.

B. Exportation PDF (Laravel + DomPDF)

a) Outil utilisé :

```
bash
composer require barryvdh/laravel-dompdf
```

b) Fonctionnalité :

Une route API `/api/admin/export-visitors-pdf` permet de générer un fichier PDF contenant les informations de tous les visiteurs.

Exemple de code :

```
php
public function exportPDF()
{
    $visitors = Visitor::all();
    $pdf = PDF::loadView('exports.visitors', compact('visitors'));
    return $pdf->download('visiteurs.pdf');
}
```

c) Résultat :

- Le PDF est **automatiquement téléchargé** dans le navigateur.
- Il contient un **tableau propre et bien formaté**, avec les colonnes : nom, CIN, téléphone, motif, date.

C. Exportation CSV (Laravel + réponse brute)

a) Route API :

/api/admin/export-visitors-csv

b) Exemple de méthode :

```
php

public function exportCSV()
{
    $visitors = Visitor::all();
    $csv = Writer::createFromString('');
    $csv->insertOne(['Nom', 'CIN', 'Téléphone', 'Motif']);

    foreach ($visitors as $v) {
        $csv->insertOne([$v->name, $v->cin, $v->phone, $v->reason]);
    }

    $response = response($csv->toString(), 200);
    $response->header('Content-Type', 'text/csv');
    $response->header('Content-Disposition', 'attachment; filename="visiteurs.csv"');

    return $response;
}
```

D. Intégration dans l'interface utilisateur (React)

Un **double bouton** est placé dans le tableau de bord Admin :

```
jsx

<button onClick={handlePDFExport} className="bg-red-600 text-white px-4 py-2 mr-2 rounded">
    Exporter en PDF
</button>

<button onClick={handleCSVExport} className="bg-green-600 text-white px-4 py-2 rounded">
    Exporter en CSV
</button>
```

Les deux boutons appellent des fonctions Axios qui déclenchent le téléchargement automatique des fichiers dans le navigateur.

E. Bilan

Format	Avantages
PDF	Lecture humaine, impression facile, officiel
CSV	Manipulation avec Excel, analyse statistique, tri

Grâce à ces deux options, le système s'adapte à **plusieurs usages administratifs**, avec une **interface intuitive et pratique**.

IV.7. Fonctionnalités implémentées par module :

Le système de gestion des visiteurs a été divisé en plusieurs modules fonctionnels. Chaque module couvre un ensemble de fonctionnalités spécifiques, répondant aux besoins du Ministère de la Justice : enregistrement, contrôle, consultation, statistiques et exportation.

◆ 1. Module Authentification & Sécurité

Fonctionnalité	Détails
Connexion sécurisée	Vérification des identifiants via Laravel Sanctum
Attribution des rôles	<code>admin</code> ou <code>agent</code> déterminé à la connexion
Redirection automatique	En fonction du rôle (<code>/dashboard</code> ou <code>/add-visitor</code>)
Routes protégées (React)	Blocage des pages sensibles pour les utilisateurs non connectés
Middleware Laravel	Filtrage des routes par rôle (<code>admin</code> , <code>agent</code>)

◆ 2. Module Gestion des Visiteurs

Fonctionnalité	Détails
Ajout d'un visiteur	Accessible à tous les rôles
Modification / Suppression	Réservé aux administrateurs
Consultation des visiteurs	Liste avec recherche et filtres (admin uniquement)
Validation des données	Contrôle des champs obligatoires (nom, CIN, téléphone...)
Historique enregistré	Date de création, mises à jour automatiques

◆ 3. Module Rendez-vous

Fonctionnalité	Détails
Création d'un rendez-vous	Lien entre visiteur et agent
Status gérés	en_attente , confirmé , annulé
Modification / Annulation	Par l'administrateur
Sélection par calendrier	Date et heure précises, avec validation

◆ 4. Module Suivi Entrée / Sortie

Fonctionnalité	Détails
Enregistrement d'entrée	Heure d'entrée horodatée automatiquement
Enregistrement de sortie	Autorisé uniquement après entrée
Statut de visite	En cours / terminée, visible dans l'interface
Gestion QR Code	Non implémenté, mais architecture prête

◆ 5. Module Tableau de Bord & Statistiques

Fonctionnalité	Détails
Accès réservé	Admin uniquement
Indicateurs clés	Nombre de visiteurs, visites du jour/mois
Graphiques dynamiques	Présentation visuelle avec Chart.js ou Recharts
Statistiques par période	Données agrégées par jour, semaine, agent, etc.

◆ 6. Module Exportation

Fonctionnalité	Détails
Export PDF	Téléchargement d'un fichier PDF propre et formaté
Export CSV	Export brut des données pour analyse Excel
Intégration React	Deux boutons distincts sur le dashboard admin
API Laravel dédiées	<code>/export-visitors-pdf</code> , <code>/export-visitors-csv</code>

Bilan

Toutes les fonctionnalités attendues ont été implémentées selon le cahier des charges. L'interface est fluide, intuitive, et chaque module est protégé, testé et fonctionnel, aussi bien côté back-end que front-end.

IV.8. Captures d'écran et exemples d'interface :

Cette section présente les principales interfaces de l'application web développée. Chaque capture d'écran est accompagnée d'une brève description pour contextualiser son rôle dans le système.

1. Page d'accueil

Description :

Page initiale visible avant la connexion. Elle présente brièvement le système avec un bouton "Se connecter" qui redirige vers la page de login.

Elle peut contenir le logo du ministère, un titre, une courte description, et un style épuré via Tailwind CSS.



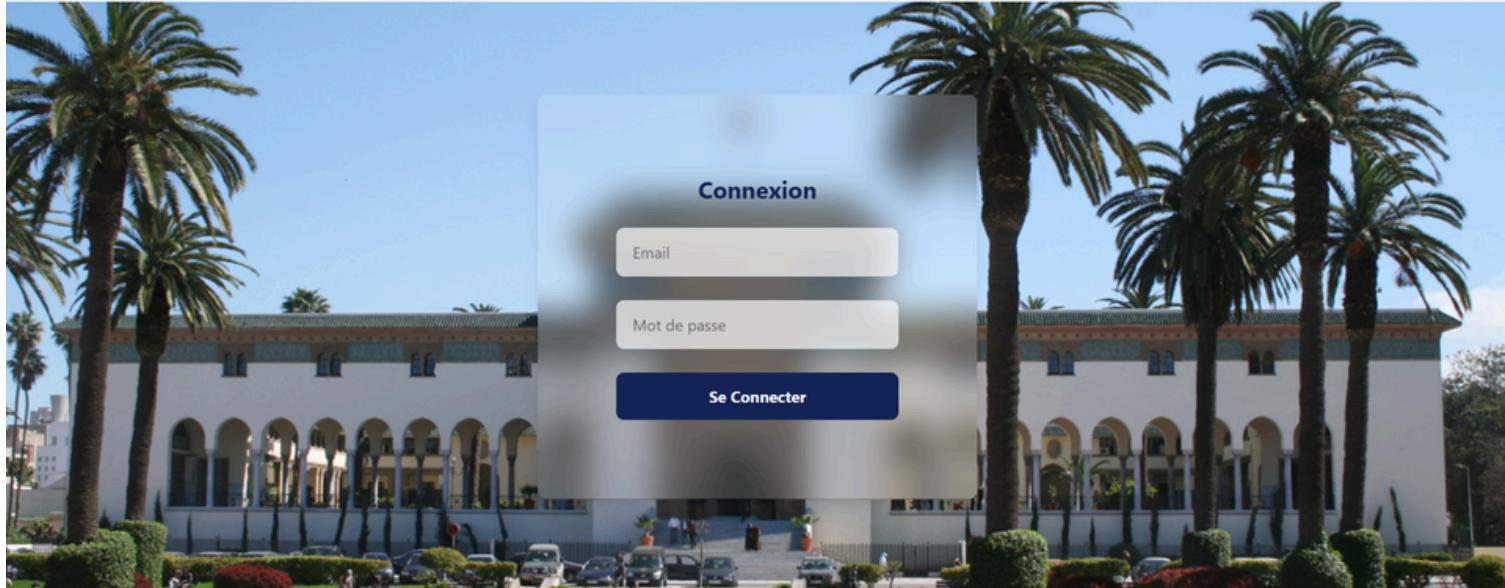
2. Page de Connexion

Description :

Permet à l'utilisateur de se connecter à l'aide de son email et mot de passe. Redirige automatiquement en fonction du rôle.



Ministère de la Justice



3. Tableau de Bord (Admin uniquement)

Description :

Permet à l'utilisateur de se connecter à l'aide de son email et mot de passe. Redirige automatiquement en fonction du rôle.

Ministère de la Justice Déconnexion

Ajouter un Visiteur Statistiques Exporter ▾ Historique

Tableau de Bord - Ministère de la Justice

[Ajouter un Visiteur](#)

Rechercher un visiteur...

Total Visiteurs 14	Visites Aujourd'hui 2	Visites ce Mois 2
-----------------------	--------------------------	----------------------

Mois	Visites
March	12
April	2

4. Formulaire d'ajout de visiteur (Window Modal)

Description :

Interface pour ajouter un nouveau visiteur avec saisie de nom, CIN, téléphone et motif. Accessible aux admins et agents.

The screenshot shows the 'Ministère de la Justice' dashboard. At the top, there is a logo and navigation links: 'Ajouter un Visiteur', 'Statistiques', 'Exporter ▾', and 'Historique'. On the right, there is a 'Déconnexion' button. The main area is titled 'Tableau de Bord - Ministère de la Justice'. It displays a chart showing a downward trend from March to April. Below the chart, there are two boxes: 'Total Visiteurs' (14) and 'Visites ce Mois' (2). A central modal window is open, titled 'Ajouter un Visiteur', containing input fields for 'Nom complet', 'CIN', 'Téléphone', and 'Raison de la visite', along with a blue 'Ajouter' (Add) button. A green success message 'Visiteur ajouté avec succès !' is shown in the top right corner of the modal.

5. Liste des visiteurs

Description :

Tableau affichant les visiteurs enregistrés avec options de modification, suppression, recherche et tri. Accessible uniquement aux administrateurs.

Liste des Visiteurs

✓ Statut mis à jour : Entré

Nom	CIN	Téléphone	Raison	status
Med Blk	Az123456	0601234569	demande destage	Entré
Salah Ben	AB123457	0701234577	demande de stage	Entré
Taha	BB012365	0145236987	stage	Sorti
John Doe	AC123456	0612345678	Réunion	Sorti
youssef	TA1547896	021456987	stage	Sorti
test	test	test	test	Sorti
test1	test1	test1	test1	Sorti
test4	test4	test4	test4	Sorti
test5	test5	test5	test5	Sorti
test4	test9	test4	test4	Entré
test9	test10	test9	test9	Entré
test9	test11	test9	test9	Sorti

6. Page historique des visiteurs

Description :

Cette page permet de **consulter l'historique des passages** de chaque visiteur, avec les heures d'entrée et de sortie.

Elle offre également la possibilité de filtrer les visites par date, par statut ou par agent.

Ministère de la Justice

Déconnexion



Tableau de Bord Ajouter un Visiteur Historique des Visiteurs

Historique des Visiteurs

Tous les statuts

mm/dd/yyyy

mm/dd/yyyy

Rechercher par CIN/Nom

Rechercher

Résultats

Nom	CIN	Statut	Date
test	test	Sorti	4/15/2025, 2:13:14 PM
test	test69	En attente	4/15/2025, 2:10:51 PM
test	test7	En attente	4/15/2025, 2:08:01 PM
Marwane	AA1234455	En attente	4/15/2025, 2:07:27 PM
test9	test11	Entré	4/15/2025, 2:03:22 PM
Med Blk	Az123456	Sorti	4/15/2025, 1:59:59 PM
test9	test10	Sorti	4/15/2025, 1:58:51 PM
test4	test9	Sorti	4/15/2025, 1:58:49 PM

7. Exportation des données (PDF / CSV)

Description :

Boutons permettant aux administrateurs de télécharger la liste des visiteurs en format PDF bien présenté ou CSV brut.

The screenshot shows a success message 'Exportation PDF réussie!' (PDF export successful) with a green checkmark icon. Below it is a navigation bar for the 'Ministère de la Justice' website. The 'Exporter' button is highlighted with a red arrow pointing to it. A dropdown menu from this button shows two options: 'PDF' and 'CSV'. Another red arrow points to the 'CSV' option. The main content area displays a table titled 'Tableau de Bord - Min' with columns for 'Nom', 'CIN', 'Téléphone', 'Raison', and 'Statut'. At the bottom left, there's a green button for 'Ajouter un Visiteur' and a search bar.

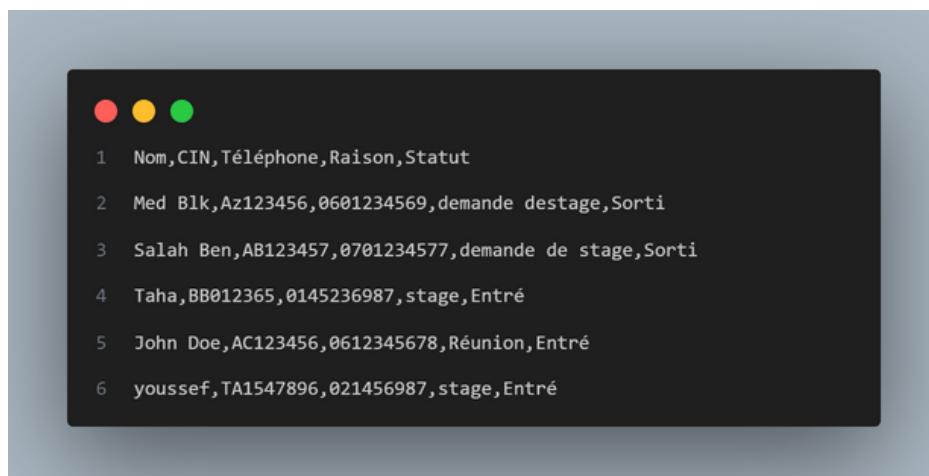
8. Exemple de PDF / CSV généré

Description :

Aperçu du fichier PDF / CSV généré à partir de la base de données. Format prêt à l'impression, bien structuré.

Liste des Visiteurs

Nom	CIN	Téléphone	Raison	Statut
Med Blk	Az123456	0601234569	demande destage	Sorti
Salah Ben	AB123457	0701234577	demande de stage	Sorti
Taha	BB012365	0145236987	stage	Entré
John Doe	AC123456	0612345678	Réunion	Entré
youssef	TA1547896	021456987	stage	Entré



Chapitre V :

Conclusion et Perspectives :

Ce projet a représenté bien plus qu'un simple exercice de fin de formation. Il a été une véritable immersion dans un environnement professionnel, avec ses exigences, ses imprévus, ses défis... et ses satisfactions.

L'un des principaux défis rencontrés dès le départ fut l'utilisation de **Laravel**, un framework que nous n'avions pas encore étudié officiellement durant notre formation. Il a donc fallu faire preuve de **curiosité, d'autonomie et de persévérance** pour comprendre ses mécanismes, maîtriser Eloquent ORM, créer des API sécurisées, et gérer les middlewares.

De même, d'autres technologies comme **React** avec **Vite, Tailwind CSS, Axios, Sanctum** ou encore la **génération de PDF/CSV** étaient nouvelles pour moi. Cela m'a poussé à sortir de ma zone de confort, à chercher, expérimenter, apprendre et recommencer.

Au-delà des aspects purement techniques, ce projet m'a permis de **me confronter pour la première fois aux exigences du marché du travail** :

- organiser mon temps,
- respecter des délais,
- livrer un projet structuré, documenté et fonctionnel,
- mais aussi penser comme un développeur face à un **client réel**, avec des besoins concrets à satisfaire.

J'ai également développé de nombreuses **compétences comportementales (soft skills)**, souvent négligées mais tout aussi essentielles :

- autonomie dans l'apprentissage,
- résilience face aux bugs persistants,
- rigueur dans le code et la documentation,
- sens de l'organisation,
- et surtout, une meilleure gestion du stress et de mes priorités.

Ce projet a renforcé ma confiance en mes capacités à résoudre des problèmes, à construire une application de A à Z, et à évoluer dans un environnement technique moderne.

Perspectives :

Même si la version actuelle du système est pleinement fonctionnelle, plusieurs pistes d'amélioration et d'extension peuvent être envisagées pour l'avenir :

- **Ajout d'un système de QR Code** pour faciliter l'enregistrement des visiteurs à l'entrée.
 - **Développement d'une version mobile** (responsive améliorée ou application mobile dédiée).
 - **Ajout de notifications** pour les agents ou les visiteurs (confirmation de rendez-vous, alertes).
 - **Ajout de filtres avancés et statistiques détaillées** par motif, par service, par période.
 - **Archivage automatique** des anciennes visites avec possibilité de les consulter par période.
 - **Multilingue** : ajout d'un support pour plusieurs langues (français, arabe, anglais...).
-

Mot de la fin :

Ce projet est une étape charnière dans mon parcours. Il m'a permis de passer de la théorie à la pratique, de coder non plus par devoir, mais avec une réelle finalité.

Il m'a prouvé que je suis capable de m'adapter, de progresser seul, de livrer un projet professionnel, même dans un environnement que je ne maîtrisais pas au départ.

Je ressors de cette expérience non seulement avec plus de savoir-faire, mais aussi avec plus de confiance, de méthode, et d'envie d'évoluer encore.

C'est avec cette énergie que je me prépare à affronter les défis du monde professionnel, mieux outillé, plus motivé, et surtout prêt à apprendre encore plus.