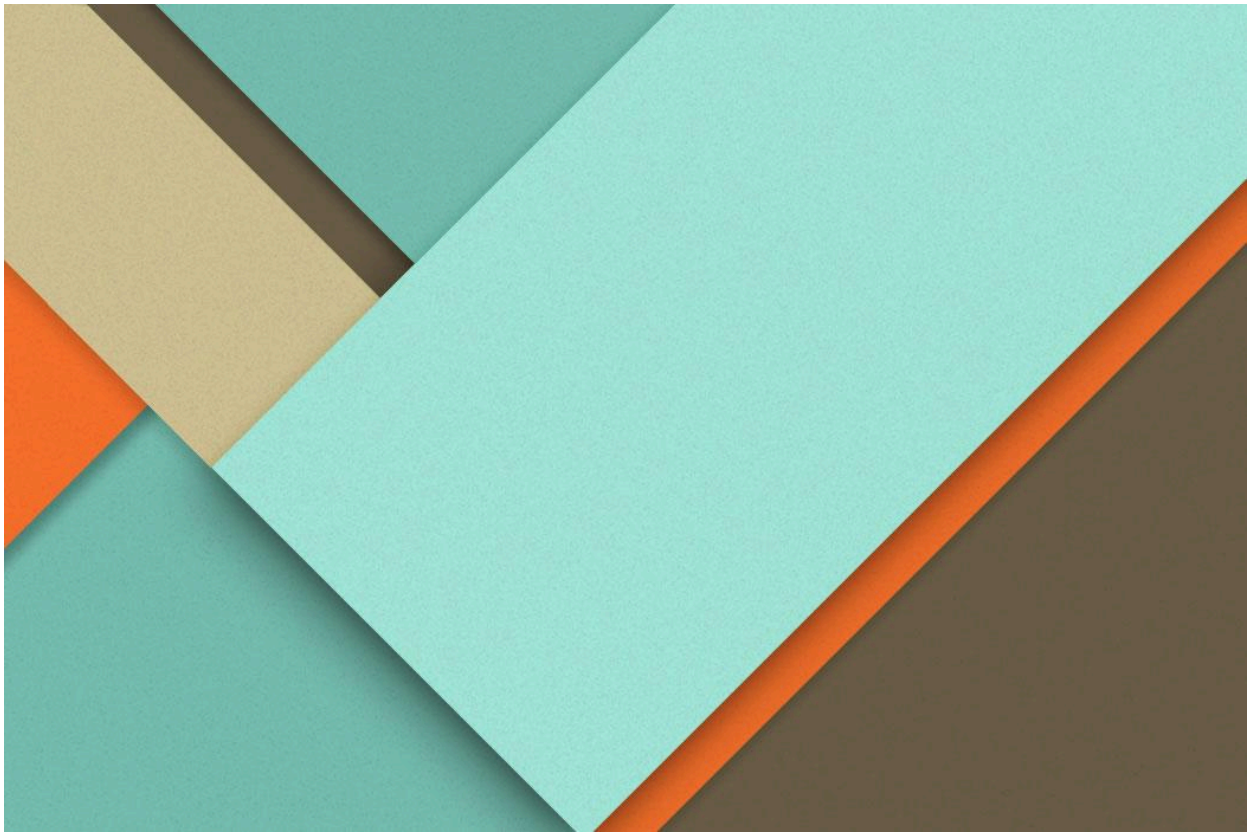


Gestion d'intervention

06/25

GRAS Matteo
Lycée Ozenne
31000 Toulouse

1. Introduction et Contexte du Projet

Le projet a consisté à développer une application web pour informatiser et automatiser la gestion des interventions techniques et de la facturation, en remplacement d'un processus manuel basé sur un fichier Excel.

L'objectif principal était de garantir une gestion plus rapide, plus claire et significativement moins sujette aux erreurs.

1.1 Stack Technique

Le développement a reposé sur l'utilisation des technologies suivantes :

Composant	Technologie	Rôle
Backend	Laravel 12	Framework principal (structure, logique métier, API REST).
Frontend	Blade (Laravel) + Bootstrap	Interface utilisateur simple et réactive.
Base de Données	MySQL	Stockage des données, avec utilisation des migrations.
Tests & Documentation	Postman, Scribe, PHPUnit	Validation de l'API et génération de documentation.

Ce projet a été une immersion dans l'écosystème Laravel, permettant la maîtrise des API RESTful, de la gestion des rôles utilisateurs (RBAC) et des bonnes pratiques de développement.

2. Préparation et Conception

Le succès du développement a été assuré par une phase de préparation menée en collaboration avec le maître de stage.

2.1 Analyse Fonctionnelle et Spécifications

Identification des besoins : Discussion approfondie sur les rôles utilisateurs (Administrateur, Technicien) et les fonctionnalités critiques à automatiser.

Cahier des charges : Rédaction d'un document listant les fonctionnalités essentielles (CRUD sociétés/sites, tickets, interventions, facturation) et les fonctionnalités secondaires.

Modélisation des données : Conception du Modèle Conceptuel de Données (MCD) et du Modèle Logique de Données (MLD) pour définir les tables et leurs relations.

2.2 Montée en Compétence sur Laravel

N'ayant pas d'expérience préalable avec Laravel, un effort de formation a été mené :

Tutoriels pratiques : Suivi de ressources vidéo pour l'apprentissage des concepts fondamentaux (migrations, modèles, relations, routes).

Documentation officielle : Étude approfondie des concepts clés de Laravel, notamment l'ORM Eloquent, les migrations, l'authentification via Sanctum et la sécurisation des API.

2.3 Architecture Technique et Structure

Choix de l'Architecture : Application de type **API RESTful** pour le backend, découplée d'un frontend simple en Blade/Bootstrap.

Initialisation du Projet : Installation de Laravel 12 et configuration de l'environnement (serveur de développement, connexion MySQL).

Définition des Bases : Création des migrations et des modèles pour les entités principales : `utilisateurs`, `sociétés`, `sites`, `tickets`, `interventions`, `matériel`, `factures`. Définition des relations `One-to-Many` et `Many-to-Many` nécessaires.

3. Développement des Fonctionnalités Clés

Le développement s'est concentré sur les modules essentiels pour l'automatisation du processus de gestion.

3.1 Liste des Fonctionnalités Implémentées

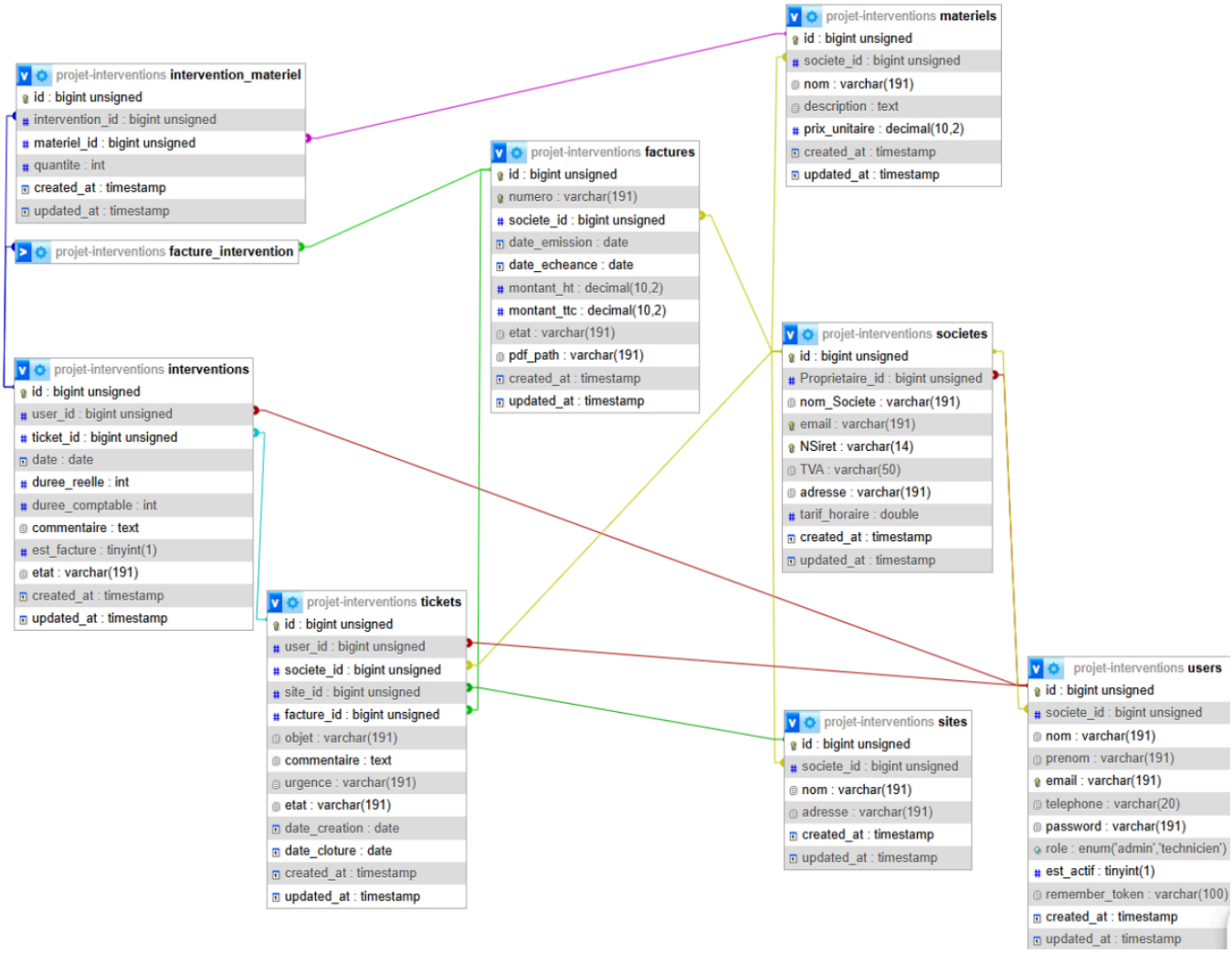
Authentification Sécurisée (Sanctum) : Mise en place des mécanismes de `login` et `logout`, et vérification des `Tokens Bearer` pour sécuriser l'accès à l'API.

Gestion Utilisateurs (CRUD) : Création et administration des comptes pour l'Admin et les Techniciens.

Gestion des Clients (CRUD) : Opérations complètes sur les Sociétés et leurs Sites associés.

Cycle Tickets/Interventions (CRUD) : Création d'un flux de travail pour la gestion des demandes (Tickets) et des interventions associées, incluant l'ajout de matériel utilisé.

3.2 Modèle Conceptuel de la Base de Données



4. Sécurité et Contrôle d'Accès

Un accent particulier a été mis sur la robustesse et la sécurité applicative.

4.1 Mise en Œuvre du Contrôle d'Accès Basé sur les Rôles (RBAC)

Policies Laravel : Utilisation des *Policies* pour définir et restreindre l'accès aux ressources selon le rôle de l'utilisateur (`Admin` ou `Technicien`).

- **Admin** : Permissions totales (CRUD complet) au sein de sa propre société (utilisateurs, clients, factures).
- **Technicien** : Accès limité aux fonctionnalités de terrain (consultation et modification de ses propres interventions, gestion du matériel).

Cybersécurité Applicative : Cette approche constitue la base d'une sécurité applicative solide.

4.2 Validation et Normalisation

Form Requests : Utilisation des `Form Requests` pour valider les données entrantes. Cette méthode permet de sécuriser les inputs contre les injections et assure la cohérence des données avant qu'elles n'atteignent le modèle.

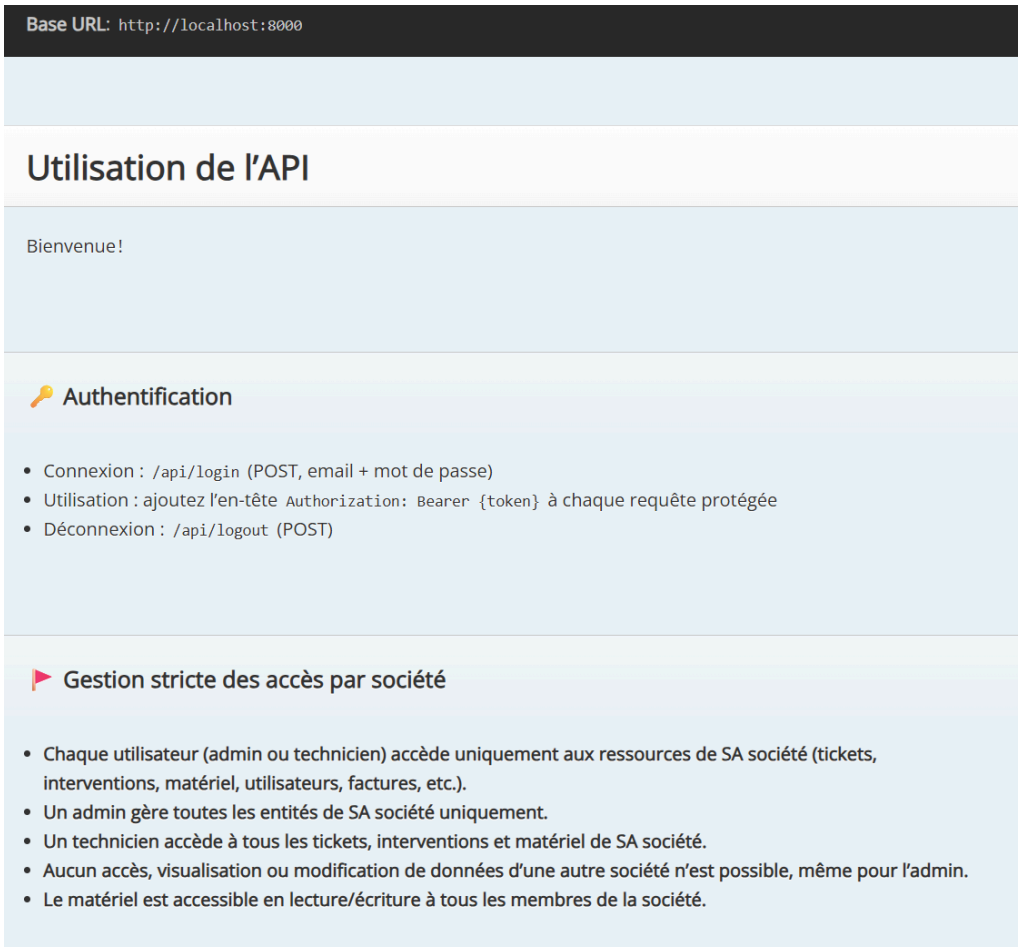
Contrôleurs : Mise en place de contrôles au niveau des contrôleurs pour s'assurer que le format des données correspond aux attentes.

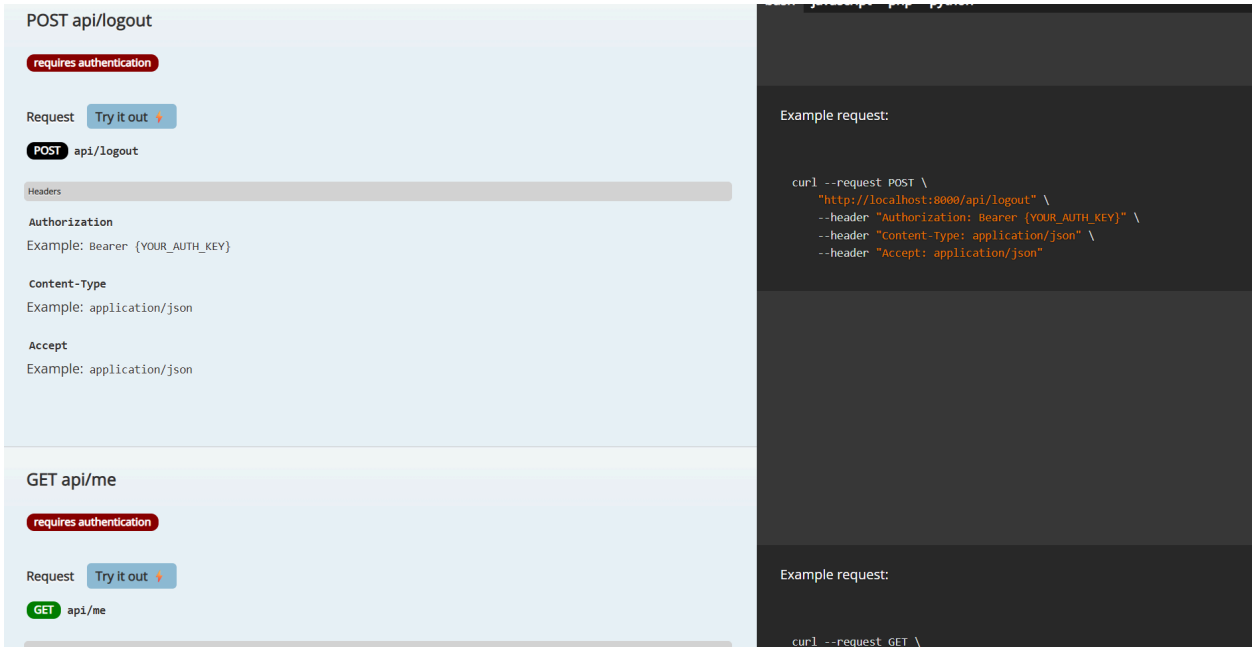
5. Documentation et Assurance Qualité

5.1 Documentation API Professionnelle (Scribe)

Documentation Interactive : Mise en place d'une documentation API auto-générée via l'outil **Scribe**.

Avantage Développeur : Cette documentation est accessible via `/docs` et permet aux développeurs front-end de tester les requêtes en direct (bouton *Try It Out*) sans nécessiter d'outils tiers (comme Postman).





5.2 Tests et Qualité du Code

Tests Automatisés (PHPUnit) : Écriture de tests unitaires et fonctionnels pour vérifier la logique métier critique.

Factories et Seeders : Création de jeux de données cohérents grâce aux *Factories* de Laravel pour faciliter les tests et le développement.

Tests Manuels (Postman) : Utilisation de Postman pour valider manuellement chaque `endpoint` de l'API.

5.3 Architecture RESTful et Standards

Le projet respecte les standards du web moderne :

Authentification : Utilisation de **Laravel Sanctum** et des **Tokens Bearer** pour une sécurité accrue par rapport aux sessions classiques.

Réponses Normalisées : Utilisation des codes de statut HTTP standardisés :

- `200/201` : Succès et Création.
- `403` : Accès Interdit (autorisation échouée).
- `422` : Erreur de Validation du Formulaire.

Optimisation des Performances (Pagination) : Les listes de ressources volumineuses (ex: tickets) sont paginées (15 par page) pour optimiser les temps de chargement et réduire la consommation de ressources serveur.

6. Résultats et Perspectives

6.1 Bilan du Projet

Réalisation : Une API sécurisée et partiellement fonctionnelle, posant la base pour l'application complète.

Valeur Ajoutée : La transformation d'un processus manuel (Excel) en une base de données centralisée et structurée offre un gain de temps immédiat et une réduction drastique des erreurs.

6.2 Perspectives d'Évolution

Le projet est conçu pour être évolutif, avec plusieurs pistes d'amélioration identifiées :

Finalisation de l'API : Poursuite du développement des *endpoints* (CRUD) pour résoudre les problèmes de droits/redirection restants.

Dashboard Complet : Développement d'un tableau de bord avancé pour une vision synthétique de l'activité.

Interface Client : Création d'un portail dédié permettant aux clients de créer et suivre leurs propres tickets.

Module de Géolocalisation : Implémentation d'un outil pour calculer automatiquement les frais de déplacement.

Automatisation de l'Envoi : Mise en place de l'envoi automatique des factures par email.