

OUATTARA BOUBACAR

**DOSSIER DE SOUTENANCE POUR
LE TITRE CONCEPTEUR
DEVELOPPEUR INFORMATIQUE**

CONTENU

INTRODUCTION AU PROJET.....	P
REMERCIEMENTS	P

PARTIE 1 – DESCRIPTION GÉNÉRALE DU PROJET

1.1 Résumé du projet	P
1.2 Analyse de l'existant	P
1.2.1 Contexte métier	
1.2.2 Limites des solutions actuelles	
1.3 Présentation des utilisateurs	P
1.3.1 Utilisateurs finaux	
1.3.2 Équipe projet (développeurs, product owner...)	
1.4 Objectifs du projet	P
1.4.1 Objectifs fonctionnels	
1.4.2 Objectifs techniques	
1.5 Contraintes et périmètre	P
1.6 Contexte technique.....	P
1.7 Définition des entités.....	P

PARTIE 2 – GESTION DE PROJET

2.1 Méthodologie utilisée (Agile / Scrum)	P
2.2 Planification et suivi	P
2.3 Outils de gestion et communication	P

PARTIE 3 – ANALYSE FONCTIONNELLE

3.1 Étude des besoinsP

3.2 Acteurs et cas d'utilisationP

3.2.1 Cas d'utilisation : authentification

3.2.2 Cas d'utilisation : création d'interventions

3.2.3 Cas d'utilisation : validation technique, rendez-vous, etc.

3.3 Maquettes de l'application mobileP

3.4 Diagramme de navigationP

PARTIE 4 – ANALYSE TECHNIQUE

4.1 Architecture logicielle globale (Mobile + Backend)P

4.2 Choix techniques et langagesP

4.3 Structure du projet mobile (React Native)P

4.4 Structure du projet backend (Symfony)P

4.5 Gestion des états, navigation, sécuritéP

4.6 Base de données (PostgreSQL)P

4.6.1 Modèle relationnel

4.6.2 Dictionnaire des données

PARTIE 5 – RÉALISATION DES BLOCS DE COMPÉTENCES

5.1 Bloc 1 – Développement Front-End Mobile Multiplateforme

5.1.1 Implémentation de l'interface

5.1.2 Composants, navigation et formulaires

5.1.3 Connexion à l'API REST sécurisée

5.1.4 Valorisation des compétences RNCP

5.2 Bloc 2 – Conception et Exploitation de la Base de Données

5.2.1 Conception du schéma

5.2.2 Requêtes, indexation, cohérence

5.2.3 Export / Import / Sauvegardes

5.3 Bloc 3 – Développement Back-End d'une API

5.3.1 API REST avec Symfony + API Platform

5.3.2 Sécurité JWT, rôles, validation

5.3.3 Exemples de routes métiers

5.3.4 Documentation (OpenAPI / Swagger)

5.4 Bloc 4 – Déploiement et Mise en Production

5.4.1 Dockerisation (Docker Compose, images)

5.4.2 CI/CD avec GitHub Actions

5.4.3 Variables d'environnement et .env

5.4.4 Déploiement local & distant

CONCLUSION GÉNÉRALEP

Synthèse des apprentissages et compétences développées

Perspectives d'évolution du projet

ANNEXESP

Maquettes Figma

Extraits de code clés

Captures d'écran des tests

Documentation API

Introduction au projet

La gestion efficace des interventions techniques représente aujourd'hui un défi majeur pour de nombreuses entreprises œuvrant dans les domaines de la maintenance industrielle, des services après-vente, ou encore des prestations techniques sur site. L'organisation des plannings, la coordination entre les techniciens et les clients, ainsi que le suivi en temps réel des opérations sont autant d'éléments clés pour garantir une prestation de qualité et maintenir un haut niveau de satisfaction client.

Dans ce contexte, le recours à des outils numériques devient indispensable. C'est dans cette optique que nous avons repris et adapté une application mobile multiplateforme existante, utilisée par une entreprise locale, afin d'en analyser la conception et d'en valoriser les aspects techniques. Cette application propose un ensemble de fonctionnalités centrées sur l'utilisateur : création de demandes d'intervention, planification de rendez-vous, gestion des statuts des interventions, notifications, messagerie, accès par rôles (technicien, administrateur, client), et bien d'autres.

Le projet s'inscrit dans le cadre de la validation du Titre Professionnel de Concepteur Développeur d'Applications (RNCP 24449). Il mobilise les compétences des quatre blocs du référentiel, couvrant à la fois le développement frontend et backend, la gestion des bases de données relationnelles, ainsi que le déploiement dans des environnements conteneurisés avec Docker.

L'architecture repose sur des technologies modernes :

- Backend en Symfony (PHP), avec API REST sécurisée via JWT.
- Base de données PostgreSQL, avec modélisation en MCD/MLD.
- Frontend mobile développé avec React Native (Expo).
- Infrastructure conteneurisée avec Docker et orchestration via Docker Compose.

Ce projet a été conçu selon une approche agile, avec des itérations de développement courtes, des tests fonctionnels continus, une documentation technique rigoureuse, et une attention particulière portée à l'expérience utilisateur.

L'objectif final est de livrer une solution robuste, ergonomique et facilement maintenable, capable de s'intégrer dans un environnement professionnel réel et d'évoluer selon les besoins de l'entreprise.

Remerciements

Je souhaite exprimer ma profonde reconnaissance à toutes les personnes qui ont contribué, de près ou de loin, à la réussite de ce projet.

Tout d'abord, mes remerciements vont à mon tuteur pédagogique, pour son accompagnement tout au long du projet. Ses retours constructifs, son expertise technique et sa disponibilité ont joué un rôle déterminant dans l'aboutissement de ce travail.

Je remercie également l'ensemble de l'équipe pédagogique, pour la qualité de l'enseignement dispensé tout au long de la formation. Grâce à eux, j'ai pu acquérir des compétences solides, tant sur le plan technique que méthodologique, et développer une véritable posture professionnelle.

Je tiens aussi à remercier mes camarades de promotion, avec qui j'ai partagé cette aventure humaine et professionnelle. Les échanges, l'entraide et l'esprit de groupe ont grandement enrichi mon parcours.

Enfin, je remercie l'entreprise d'accueil pour la confiance accordée, et pour m'avoir permis de travailler sur un cas concret, enrichissant tant sur le plan technique que sur le plan humain.

PARTIE 1 – DESCRIPTION GÉNÉRALE DU PROJET

1.1 Résumé du projet






Le projet « Gestion d'Interventions Techniques » est une application complète, conçue dans le cadre de la validation du titre professionnel Concepteur Développeur d'Applications (RNCP 24449). Il s'agit d'un système d'information moderne et modulaire, permettant de piloter tout le cycle de vie d'une intervention technique, depuis la demande initiale émise par un client jusqu'à sa réalisation sur le terrain par un technicien.

Le projet s'inspire d'une solution existante utilisée par une entreprise locale. Nous avons choisi de la reprendre, de l'adapter et d'en étudier tous les aspects techniques pour en faire un cas concret d'application des compétences du référentiel CDA.



Problématique métier et enjeux

Dans de nombreux secteurs — maintenance, service après-vente, gestion de parc matériel — la planification des interventions techniques reste un défi. Les entreprises doivent pouvoir :

-  Digitaliser la gestion des demandes d'intervention client
-  Optimiser l'affectation des techniciens, selon leurs disponibilités et compétences
-  Suivre en temps réel l'état des interventions
-  Gérer de façon fluide les rendez-vous entre clients et techniciens
-  Centraliser la gestion des équipements installés chez les clients

Ce projet répond à ces besoins avec une solution mobile moderne, connectée à une API sécurisée.



Architecture technique

L'architecture de la solution repose sur deux blocs fondamentaux : un backend sécurisé et une application mobile intuitive.

Backend — API REST Symfony

- Développé en Symfony (PHP 8) avec API Platform
- Authentification sécurisée via JWT (LexikJWTAuthenticationBundle)
- Gestion des rôles (client, technicien, admin, super admin)
- Base de données PostgreSQL relationnelle
- Système de planification de créneaux de rendez-vous et d'interventions
- Documentation automatisée des endpoints
- Tests fonctionnels avec PHPUnit
- Conteneurisé via Docker Compose





Frontend — Application Mobile React Native

- Développée avec React Native + Expo en TypeScript
- Interfaces dynamiques et responsives

- Gestion des formulaires avec React Hook Form
 - Authentification JWT via l'API
 - Stockage local sécurisé avec AsyncStorage
 - Navigation conditionnelle selon le profil utilisateur
 - Intégration continue via GitHub Actions
-

Profils utilisateurs et rôles

L'application gère plusieurs niveaux d'accès avec des droits distincts :

-  Client : soumet des demandes d'intervention, suit ses équipements, gère ses rendez-vous
-  Technicien : consulte ses interventions, met à jour leur statut (acceptée, en cours, terminée)
-  Administrateur d'entreprise : supervise les techniciens et assigne les interventions
-  Super administrateur : gère l'ensemble du système, utilisateurs et entreprises

Objectifs pédagogiques

Ce projet m'a permis de mobiliser un large éventail de compétences couvrant les quatre blocs du référentiel CDA :

Bloc	Compétence mobilisée
Bloc1	Développement mobile avec React Native (multi-plateforme)
Bloc2	Conception et manipulation de bases relationnelles PostgreSQL
Bloc3	Création d'API REST sécurisées avec Symfony
Bloc4	Conteneurisation avec Docker, CI/CD via GitHub Actions

En complément, j'ai également pratiqué :

- La gestion agile du projet (kanban, suivi de tâches, découpage itératif)
- La rédaction de documentation technique et fonctionnelle
- L'usage d'outils professionnels : Postman, DBeaver, Figma, GitHub Projects



1.2 Analyse de l'existant

1.2.1 Contexte métier



Dans un monde professionnel où la réactivité et la satisfaction client sont devenues des leviers de compétitivité majeurs, la gestion efficace des interventions techniques est un enjeu stratégique pour les entreprises spécialisées dans les services : maintenance industrielle, assistance informatique, réparation d'équipements électroniques ou électroménagers, gestion de parcs matériels, etc.

Ces entreprises doivent traiter quotidiennement un volume important de demandes d'intervention, émanant de clients particuliers ou professionnels. Chaque demande doit être réceptionnée, planifiée, affectée à un technicien compétent, suivie en temps réel, et faire l'objet d'un compte rendu — souvent accompagné d'une mise à jour de la fiche équipement concerné.

Or, dans de nombreuses structures, ce processus reste largement manuel ou fragmenté :

-  Les demandes sont collectées par téléphone, par email ou via des formulaires papier.
-  Les rendez-vous sont planifiés à la main, parfois sans tenir compte de la charge de travail réelle des

techniciens.

-  Le suivi des interventions se fait via des tableurs Excel, des notes manuelles ou des fichiers stockés localement, souvent non synchronisés entre les membres de l'équipe.
-  La communication entre les différents acteurs (client, technicien, gestionnaire) repose sur des échanges asynchrones, difficilement traçables.

Cette organisation entraîne des risques importants : retards, doublons, oublis, mauvaise affectation des techniciens, perte d'historique, absence de reporting fiable. Le manque d'outillage adapté nuit à l'efficacité opérationnelle, à la satisfaction des clients et à la rentabilité des missions.

C'est dans ce contexte métier exigeant, confronté à une transition numérique encore inachevée, qu'est né le projet « Gestion d'Interventions Techniques » : une solution numérique sur mesure, conçue pour digitaliser, centraliser et automatiser tout le processus d'intervention, du dépôt de la demande jusqu'au compte rendu final.

1.2.2 Limites des solutions actuelles

Après une phase de veille technologique et d'analyse de terrain, plusieurs limites récurrentes ont été identifiées dans

les solutions actuellement utilisées par les entreprises ciblées (notamment les PME) :

Manque de centralisation

Les données sont souvent réparties entre plusieurs outils (boîte email, cahier de planification, Google Sheets, SMS, etc.), ce qui nuit à la cohérence du processus. Il est difficile d'avoir une vue d'ensemble sur l'ensemble des interventions en cours ou passées.

Absence de gestion des rôles et permissions

Beaucoup d'outils internes ne prévoient aucune différenciation des accès : un client peut voir les interventions des autres, un technicien peut modifier les informations administratives, etc. Cette faille de sécurité peut entraîner des erreurs ou des violations de confidentialité.

Faible traçabilité

Les interventions ne sont pas historisées de manière fiable. Il est difficile de retracer qui a fait quoi, quand, sur quel équipement, et de conserver un journal d'activité clair.

Planification manuelle et non optimisée

La planification repose souvent sur une affectation manuelle, sans lien avec les plannings réels, les spécialisations techniques ou la zone géographique des techniciens. Cela entraîne une mauvaise répartition de la charge et des temps d'intervention non optimisés.

Solutions existantes inadaptées

Certains outils du marché (ERP spécialisés, logiciels SaaS) sont :

- Trop lourds ou complexes pour des structures de petite taille
- Peu personnalisables (pas d'adaptation aux process internes)
- Coûteux, avec un retour sur investissement difficile à mesurer
- Ou encore non mobiles, alors que les techniciens sont toujours en déplacement

1.3 Présentation des utilisateurs

La réussite d'une application de gestion repose avant tout sur une compréhension fine des utilisateurs finaux et des acteurs du projet. L'architecture fonctionnelle et l'ergonomie de l'application ont été pensées autour de ces profils. Cette section détaille les deux grandes catégories d'utilisateurs : les utilisateurs finaux et les membres de l'équipe projet.

1.3.1 Utilisateurs finaux


L'application Gestion d'Interventions Techniques est destinée à être utilisée par quatre types d'utilisateurs, chacun ayant des besoins et des droits spécifiques. L'objectif est de fluidifier la communication entre les clients, les techniciens et les administrateurs, tout en assurant un suivi efficace et sécurisé des interventions.

Les clients (ROLE_CUSTOMER)

Les clients peuvent être des entreprises ou des particuliers ayant besoin d'une assistance technique pour leurs équipements. L'application leur offre une interface simple, fluide et intuitive, avec les fonctionnalités suivantes :

- Création de demandes d'intervention : En quelques clics, un client peut soumettre une demande en précisant le type de problème rencontré, le matériel concerné, et éventuellement une description ou des fichiers joints.
- Planification de rendez-vous : Le client peut choisir un créneau selon les disponibilités des techniciens ou faire une demande spécifique.
- Suivi en temps réel : L'état de chaque intervention (en attente, assignée, en cours, terminée) est visible à tout moment.

- Historique : Les clients peuvent consulter l'historique des interventions passées, incluant les commentaires du technicien.
- Gestion des équipements : Chaque client peut enregistrer, modifier ou supprimer les équipements liés à son compte, facilitant la récurrence des interventions.

 Objectif principal : offrir une expérience utilisateur fluide, réactive et rassurante pour les clients, tout en les rendant autonomes dans la gestion de leurs besoins.




Les techniciens (ROLE_TECHNICIAN)

Les techniciens sont les acteurs opérationnels du terrain. Leur interface a été pensée pour les aider à gérer efficacement leurs interventions, même dans des conditions de mobilité.

- **Consultation des interventions assignées** : Chaque technicien voit les tâches qui lui sont attribuées avec les détails essentiels (lieu, client, équipement, description...).
- **Mise à jour du statut d'une intervention** : Ils peuvent faire évoluer une intervention dans le cycle pending → assigned → in_progress → completed.

- **Commentaires techniques** : À la fin de l'intervention, un rapport peut être saisi et joint à la fiche de l'intervention.
- **Gestion des tâches** : Certaines interventions peuvent comporter plusieurs étapes ou tâches. Le technicien peut les cocher au fur et à mesure de leur réalisation.
- **Organisation personnelle** : Grâce à leur espace personnel, les techniciens peuvent visualiser leur planning et s'organiser en conséquence.

 **Objectif principal** : accompagner les techniciens dans leur quotidien avec une interface mobile simple, rapide et fonctionnelle, même hors ligne si besoin.




Les administrateurs d'entreprise (ROLE_ADMIN)

Chaque entreprise cliente dispose d'au moins un administrateur, responsable de la gestion globale de son compte et de son équipe.

- **Gestion des utilisateurs** : Création, modification ou suppression des techniciens ou clients rattachés à l'entreprise.
- **Validation et assignation des interventions** :
Lorsqu'un client soumet une demande, l'administrateur peut l'approuver, la rejeter, ou l'assigner manuellement

à un technicien.

- **Suivi d'activité** : Des tableaux de bord permettent de suivre l'ensemble des interventions de l'entreprise, par technicien ou par équipement.
- **Gestion des rendez-vous** : Les demandes de rendez-vous peuvent être acceptées ou déplacées en fonction des contraintes internes.
- **Gestion des équipements** : L'administrateur peut gérer tout le parc d'équipements pour l'ensemble de ses clients internes.


 **Objectif principal** : donner aux administrateurs les outils nécessaires pour superviser efficacement leur équipe et maintenir une qualité de service optimale.

Le super administrateur (ROLE_SUPER_ADMIN)

Ce rôle est réservé à l'opérateur principal de la plateforme (souvent l'organisation qui développe et héberge la solution). Il a des droits étendus sur toute l'architecture du système.

- **Gestion globale des entreprises** : Création, suspension ou suppression des entreprises inscrites sur la plateforme.

- **Contrôle des utilisateurs** : Visualisation et modification des comptes utilisateurs de tous les rôles.
- **Audit et sécurité** : Possibilité de consulter l'activité des utilisateurs, de vérifier les logs et de prendre des décisions correctives.
- **Modération** : En cas d'abus, de comportement inapproprié ou de bugs, le super admin peut intervenir immédiatement.

 **Objectif principal** : assurer la cohérence, la sécurité et la stabilité de l'ensemble du système.

1.3.2 Équipe projet (développeurs, product owner...)

Le développement de cette application a été mené dans un cadre académique rigoureux, en simulant les rôles typiquement rencontrés dans un projet Agile de type **Scrum**.



Développeur Full Stack – Jules-Jean-Louis et Boubacar ouattara)

En tant que **concepteur et développeur principal**, j'ai pris en charge l'intégralité des phases du cycle de vie de l'application :

- **Recueil des besoins** et analyse métier,

- **Modélisation de la base de données** (PostgreSQL & MySQL),
- **Conception du backend Symfony** avec API REST sécurisée par JWT,
- **Développement du front mobile avec React Native** (Expo + TypeScript),
- **Mise en place de la logique métier** (cycle de vie des interventions, assignation, planification...),
- **Rédaction de la documentation technique**, du README et des guides d'utilisation,
- **Réalisation des tests manuels et partiels automatisés.**



Product Owner (rôle assumé)

Dans ce projet, j'ai également endossé le rôle de **Product Owner** afin de :

- Définir les **priorités fonctionnelles**,
- Maintenir un **backlog réaliste**,
- Simuler des **user stories** adaptées à différents profils utilisateurs,

- **Valider les livrables** selon un cahier des charges défini.



Utilisateurs testeurs (profils fictifs)

Des jeux de données réalistes ont été conçus afin de simuler :

- Des **clients d'entreprises fictives**,
- Des **techniciens** avec différents niveaux de charge de travail,
- Des **administrateurs avec plusieurs entreprises sous leur responsabilité**.

Ces profils ont été utilisés pour tester l'ensemble des fonctionnalités de l'application en conditions quasi-réelles.

1.4 Objectifs du projet

Le projet Gestion d'Interventions Techniques vise à développer une application mobile multiplateforme (Android/iOS) accompagnée d'une API backend sécurisée, dédiée à la gestion des interventions terrain pour une entreprise de services techniques. Ce projet répond à un besoin métier réel, identifié auprès d'une entreprise

existante, confrontée à des problématiques de coordination entre clients, techniciens, administrateurs et équipements.

Dans un contexte où les entreprises doivent gagner en efficacité, en traçabilité et en réactivité, la digitalisation des interventions est devenue un enjeu stratégique. L'objectif est donc de proposer une solution numérique complète, modulaire, sécurisée et évolutive, permettant de suivre chaque étape d'une intervention de la demande initiale à la clôture technique.

Ce projet s'inscrit dans une démarche agile, avec des livrables fonctionnels progressifs, une architecture découplée et une approche DevOps pour le déploiement. Il constitue une preuve de compétences en conception, développement, sécurisation, test, documentation et conteneurisation, en lien direct avec les blocs de compétences du RNCP24449.

1.4.1 Objectifs fonctionnels

Les objectifs fonctionnels expriment les besoins utilisateurs métiers que l'application doit satisfaire, avec des fonctionnalités intuitives et structurées autour de profils distincts (client, technicien, administrateur, super administrateur).

Objectifs fonctionnels principaux

- **Soumission simplifiée des demandes** : les clients peuvent initier une demande d'intervention via l'application mobile, en sélectionnant le type d'intervention,

l'équipement concerné, et en joignant des précisions.

- Planification fluide des rendez-vous : l'administrateur valide les demandes et propose des créneaux selon les disponibilités des techniciens.
- Affectation intelligente des interventions : en tenant compte des spécialisations, zones géographiques, et disponibilités des techniciens.
- Suivi dynamique du statut d'une intervention : du dépôt à la réalisation finale, chaque étape est horodatée et associée à des acteurs identifiables.
- Espace technicien dédié : les techniciens disposent d'un tableau de bord clair, avec les interventions à venir, en cours ou réalisées.
- Portail d'administration : les administrateurs d'entreprise ont une vision globale des interventions, des techniciens et des équipements.
- Gestion hiérarchisée des rôles : avec une différenciation claire des accès et actions entre les profils utilisateur (client, technicien, admin, super admin).



Objectifs fonctionnels secondaires

- **Inventaire d'équipements techniques** : gestion d'un parc client (modèle, marque, statut, date d'installation, interventions associées).
- **Consultation de l'historique** : filtrage par date, technicien, entreprise, statut, etc., avec possibilité de génération de rapports PDF.
- **Gestion avancée des droits d'accès** : via des rôles stockés dans la base et interprétés côté API et front-end (middleware de protection des routes).
- **Confidentialité renforcée** : chaque utilisateur ne voit que les données associées à son rôle et à son entreprise.
- **Confort mobile natif** : l'interface mobile est conçue pour les utilisateurs terrain, avec simplicité, rapidité, et une expérience utilisateur optimisée.

1.4.2 Objectifs techniques

Les objectifs techniques concernent la qualité logicielle, la robustesse de l'architecture, la sécurité des flux, et la maintenabilité du code, dans une logique de production logicielle professionnelle.

Objectifs techniques principaux

- **Développement d'une API RESTful robuste:** avec Symfony 6, JWT, Doctrine ORM, validation des entrées, sérialisation des entités, pagination, filtrage.
- **Sécurisation complète :** authentification avec JWT, rafraîchissement de token, vérification des rôles sur chaque route, stockage chiffré des mots de passe (bcrypt).
- **Base de données relationnelle cohérente :** MCD/MLD validés, PostgreSQL comme moteur principal, relations bien définies (OneToMany,ManyToOne, etc.), intégrité garantie.
- **Application mobile professionnelle :** architecture modulaire React Native avec Expo, Typage strict (TypeScript), navigation conditionnelle, état global via Context API.
- **Validation dynamique des formulaires :** React Hook Form avec contrôle des erreurs, feedback utilisateur, formatage des champs (dates, emails, statuts).
- **Communication front-back fluide :** gestion centralisée d'Axios, hooks personnalisés, loading states, gestion d'erreurs backend.
- **Architecture découplée et scalable :** séparation stricte des responsabilités entre front-end mobile, back-end API, et services d'infrastructure.

Objectifs techniques secondaires

- **Conteneurisation complète avec Docker** : backend, frontend, bases de données PostgreSQL et MySQL orchestrés avec `docker-compose`, modes dev et prod.
- **Automatisation DevOps** : CI avec GitHub Actions, publication d'images Docker vers GitHub Container Registry, scripts d'init et de migration de BDD.
- **Documentation et maintenabilité** : endpoints documentés avec Swagger, README clairs, arborescences propres, commentaires explicites, bonnes pratiques Git.
- **Tests API et intégration** : jeux de tests avec Postman (collections), tests fonctionnels avec PHPUnit côté backend, tests Go dans le projet SafeBase (injection de base, suppression, vérification d'état).
- **Préparation à la publication mobile** : configuration complète des icônes, splash screens, permissions (caméra, stockage), gestion multilingue (à terme).

1.5 Environnement technique

Le projet « **Gestion d'Interventions Techniques** » a été conçu dans un contexte pédagogique simulant les conditions réelles d'un développement professionnel, en

tenant compte des **ressources disponibles**, des **contraintes imposées** et des **exigences du référentiel CDA (RNCP24449)**. Il s'est déroulé dans un **environnement technique complet**, allant de l'analyse des besoins jusqu'à la mise en production conteneurisée, avec une approche DevOps et agile.

1.5.1 Contraintes du projet

Le projet a été encadré par plusieurs types de contraintes qui ont influencé les choix technologiques, l'organisation du travail, la priorisation des fonctionnalités et la structuration du code.



Contraintes temporelles

La première contrainte majeure fut le temps imparti. Le développement du projet s'est déroulé sur une période de quelques mois, avec des jalons intermédiaires (maquettes, version alpha, tests, documentation), et une date butoir imposée par l'échéance du titre professionnel.

- Une planification rigoureuse a donc été mise en place, à l'aide d'un outil de gestion de tâches (Trello) et de sprints hebdomadaires simulés.
- Certaines fonctionnalités secondaires ont été dépriorisées ou reportées à une version post-livraison, pour garantir la livraison d'un MVP (Minimum Viable Product) fonctionnel.

- Les phases de tests utilisateurs réels ont été remplacées par des tests simulés ou automatisés, faute d'un client réel.

Contraintes humaines

Le projet a été réalisé en autonomie complète, sans équipe de développement, sans product owner réel, et sans testeurs externes.

- Toutes les étapes (conception, développement, test, documentation, CI/CD) ont été gérées seul, nécessitant une organisation personnelle rigoureuse et une polyvalence technique.
- Le rôle de Product Owner a été assumé par moi-même, en imaginant des scénarios utilisateurs réalistes et en priorisant les fonctionnalités à développer.
- L'absence d'équipe m'a également conduit à automatiser certaines tâches (test, build, déploiement) via GitHub Actions pour limiter les interventions manuelles.

Contraintes techniques

Les choix technologiques ont été faits en accord avec les attendus du référentiel CDA et les bonnes pratiques du développement web et mobile moderne.

- Backend : Symfony 6, LexikJWTAuthenticationBundle pour la sécurité, API Platform pour la documentation, PostgreSQL en base principale.
- Mobile : React Native avec Expo (TypeScript), React Navigation, Axios, React Hook Form, AsyncStorage pour la gestion locale des sessions.
- Conteneurisation : Docker et Docker Compose, environnement multi-conteneurs (frontend, backend, base de données), volume persistant pour les données.

Autres contraintes techniques :

- L'application devait fonctionner à la fois sur Android, iOS et en mode web (via Expo Go).
- Le backend devait permettre l'isolation des rôles et des données, avec une protection fine des endpoints selon les autorisations JWT.
- Le code devait être versionné, documenté et modulaire, avec possibilité d'ajouter des modules ultérieurement (ex. : gestion des pièces détachées).



Contraintes de sécurité

La sécurité des données utilisateurs était une exigence fondamentale dès la phase de conception.

- L'authentification a été implémentée via JWT (token d'accès + token de rafraîchissement), avec stockage local dans l'application mobile.
- Chaque route backend est protégée par un middleware de rôle, garantissant l'accès uniquement aux utilisateurs habilités (ex. : seul un admin peut valider un rendez-vous).
- Les mots de passe sont hachés via bcrypt côté API.
- Les données sensibles (logs, tokens, mots de passe) sont exclues des réponses JSON et stockées dans des variables d'environnement `.env` (non versionnées).

1.5.2 Périmètre fonctionnel

Pour garantir une cohérence globale et livrer un produit fonctionnel dans les délais, un périmètre fonctionnel réaliste et ciblé a été défini en amont. Celui-ci couvre l'essentiel des cas d'usage métier, sans tomber dans l'excès de complexité.

✓ Fonctionnalités incluses dans le périmètre

Catégorie	Fonctionnalité
Authentification	Inscription, connexion, tokens JWT, refresh, rôles
Gestion des utilisateurs	Création/modification/suppression (super admin, admin, technicien)
Gestion des interventions	Création de demandes, affectation, suivi des statuts
Workflow interventions	Pending → Approved → Assigned → In progress → Completed/Canceled
Rendez-vous	Planification, validation, refus, calendrier
Équipements clients	Création, liaison avec client/intervention, consultation
Planning technicien	Vue calendrier / liste des interventions affectées
Tableau de bord admin entreprise	Vue globale des demandes, techniciens, clients
Historique	Consultation filtrée par date, utilisateur, statut, entreprise
Rôles et permissions	Séparation claire des accès et actions par rôle JWT

✗ Fonctionnalités hors périmètre (versions futures)

Certaines fonctionnalités à forte valeur ajoutée ont été mises de côté volontairement pour des raisons de temps ou de complexité :

- Notifications push en temps réel (via Firebase ou OneSignal).
- Facturation automatique des interventions (génération de devis/factures).

- Statistiques métiers (taux de satisfaction, durée moyenne d'intervention, etc.).
- Application web responsive dédiée à l'administration (version bureau).
- Mode hors-ligne mobile avec synchronisation des données.
- Gestion des pièces détachées ou stocks associés aux interventions.
- Multi-langue pour déploiement international.

Ces limitations n'enlèvent rien à la qualité du socle technique, conçu dès le départ pour accueillir ces extensions à l'avenir, grâce à une architecture modulaire, découpée, et documentée.

1.6 Contexte technique

Le projet *Gestion d'Interventions Techniques* a été conçu en mobilisant un ensemble d'outils, de langages et d'environnements techniques modernes, respectant les standards professionnels en matière de développement fullstack web et mobile. Ce contexte technique couvre l'architecture logicielle, les technologies choisies, ainsi que les outils de développement, de versioning, de conteneurisation et de documentation.

1.6.1 Architecture générale

Le projet est basé sur une architecture en couches découplées et modulaire, distinguant clairement les responsabilités :

- Backend REST API (Symfony + API Platform) : Gère la logique métier, les utilisateurs, les rôles, les interventions, les équipements, les rendez-vous, etc.
- Frontend Mobile (React Native + Expo) : Fournit l'interface utilisateur pour les clients et techniciens.
- Base de données relationnelle (PostgreSQL) : Stocke de manière persistante les données métiers.
- Communication via HTTP + JWT : L'échange entre le front et le back est sécurisé par des jetons JWT et des rôles utilisateurs.

1.6.2 Technologies principales

Backend

- **Langage** : PHP 8.1
- **Framework** : Symfony 6
- **Outils** :

- API Platform (exposition automatique des entités en REST)
- Doctrine ORM (mapping objet-relationnel)
- LexikJWTAuthenticationBundle (authentification JWT)
- PHPUnit (tests unitaires et fonctionnels)



Frontend Mobile

- **Langage** : TypeScript
- **Framework** : React Native avec **Expo**
- **Librairies utilisées** :
 - `axios` pour les appels API sécurisés
 - `jwt-decode` pour décoder les tokens côté mobile
 - `expo-router` pour la navigation dynamique
 - `react-hook-form` pour la gestion des formulaires

- @expo/vector-icons et react-native-paper pour l'UI



Base de données

- **SGBD utilisé** : PostgreSQL
- **Modèle relationnel** conçu avec des entités métiers comme :
 - User, Company, Intervention, AppointmentRequest, Equipment, Task, TypeIntervention, etc.
- Contraintes d'intégrité (relations, clés étrangères, validations)

1.6.3 Environnement de développement

- **IDE** : Visual Studio Code
- **Gestionnaire de versions** : Git + GitHub
- **Conteneurisation** : Docker + Docker Compose
 - Backend, base de données, nginx
- **Scripts de build mobile** : Expo CLI

- **Stockage sécurisé mobile** : Expo SecureStore (pour les tokens)
- **Environnement de test** : Postman, PHPUnit

1.6.4 Hébergement et déploiement

- Environnement de développement local via **Docker** Compose.
- API et base PostgreSQL conteneurisées pour portabilité.
- Possibilité de déploiement cloud (GitHub Container Registry, Railway ou Render).
- Documentation API accessible via `/api/docs` générée automatiquement par **API Platform**.

1.7 Définition des entités

Dans le cadre de l'application *Gestion d'Interventions Techniques*, plusieurs entités métier ont été modélisées pour représenter les acteurs, les objets manipulés ainsi que les processus du système. Chaque entité a été pensée pour répondre à un besoin métier spécifique et est interconnectée selon des relations logiques (OneToMany,ManyToOne, etc.). Ces entités sont implémentées au

niveau du backend Symfony à l'aide du composant Doctrine ORM.

1.7.1 Entité User

L'entité centrale représentant les différents profils utilisateurs de l'application :

- **Champs principaux :**
 - `id`, `email`, `password`, `roles`, `first_name`, `last_name`
- **Rôles possibles :** `ROLE_CUSTOMER`, `ROLE_TECHNICIAN`, `ROLE_ADMIN`, `ROLE_SUPER_ADMIN`
- **Relations :**
 - Liée à une `Company` (optionnelle)
 - Liée à plusieurs `Intervention`, `AppointmentRequest`, `Equipment` selon le rôle

1.7.2 Entité Company

Représente une entreprise cliente pouvant regrouper plusieurs utilisateurs et équipements :

- **Champs principaux** : `id`, `name`, `address`, `status`, `approved_by`, etc.
 - **Relations** :
 - Plusieurs User rattachés
 - Plusieurs Equipment et Intervention
-

1.7.3 Entité Intervention

Modélise une intervention technique dans le système :

- **Champs principaux** : `title`, `description`, `status`, `created_at`, `updated_at`
- **Statuts** :
 - `pending`, `assigned`, `in_progress`, `completed`, `canceled`
- **Relations** :
 - Assignée à un **User** technicien

- Associée à un `TypeIntervention`, `Equipment`, et une `Company`
 - Contient des `Task`
-

1.7.4 Entité `AppointmentRequest`

Représente une demande de rendez-vous :

- **Champs principaux** : `date`, `status`, `created_at`, `approved_by`
 - **Statuts** :
 - `pending`, `accepted`, `rejected`
 - **Relations** :
 - Créée par un `User`
 - Associée à une `Company` et à un `Equipment`
-

1.7.5 Entité `Equipment`

Matériel ou objet à réparer/intervenir :

- **Champs** : serial_number, name, description, status
 - **Relations** :
 - Associé à un User, une Company, un Brand, un OperatingSystem
-

1.7.6 Autres entités

- **Task** : sous-éléments liés à une Intervention, avec un titre, un statut, un utilisateur responsable
- **TypeIntervention** : typologie de l'intervention (ex. : maintenance, diagnostic, remplacement)
- **Brand et OperatingSystem** : tables de référence pour caractériser les équipements

1.8 Fonctionnalités attendues

L'application *Gestion d'Interventions Techniques* a été conçue pour offrir un ensemble de fonctionnalités répondant aux besoins spécifiques des utilisateurs métiers identifiés précédemment. Ces fonctionnalités sont réparties

entre fonctionnalités primordiales (indispensables au bon fonctionnement de l'application) et fonctionnalités secondaires (complémentaires ou à valeur ajoutée).

1.8.1 Fonctionnalités primordiales

Fonctionnalité	Description
Authentification sécurisée (JWT)	Connexion des utilisateurs via une API sécurisée par token, avec rafraîchissement automatique du token.
Gestion multi-rôles	Attribution dynamique des rôles CUSTOMER, TECHNICIAN, ADMIN, SUPER_ADMIN avec des droits d'accès distincts.
Création d'interventions	Les clients peuvent soumettre une demande d'intervention à partir d'un formulaire mobile.
Assignation des interventions	Les admins peuvent assigner une intervention à un technicien spécifique.
Planification de rendez-vous	Les clients peuvent faire une demande de rendez-vous, qui doit être validée par un administrateur.
Gestion des équipements	Enregistrement des équipements, systèmes d'exploitation, marques et modèles.

Gestion des tâches techniques	Chaque intervention peut être décomposée en tâches à exécuter par le technicien.
Espace personnel	Chaque utilisateur a accès à son espace personnel pour consulter ses interventions ou actions à mener.
Suivi du statut d'une intervention	Les statuts <code>pending</code> , <code>assigned</code> , <code>in_progress</code> , <code>completed</code> , <code>canceled</code> permettent de suivre l'avancement.

1.8.2 Fonctionnalités secondaires

Fonctionnalité	Description
Recherche et filtrage	Tri des interventions selon le statut, la date, le technicien, ou l'équipement concerné.
Visualisation historique	Historique des interventions réalisées pour chaque client.
Notifications techniques	Affichage de statuts colorés, indicateurs d'état, messages de confirmation d'action.
Comptes de démonstration	Mise en place de comptes de test pour

	simuler l'ensemble des rôles utilisateurs.
Maquettes mobiles ergonomiques	Interface utilisateur adaptée aux mobiles grâce à React Native et Expo.
Documentation API	Documentation Swagger accessible à l'URL <code>/api/docs</code> pour faciliter les tests et l'intégration.
Tests unitaires backend	Vérification automatique de la logique métier avec PHPUnit.
CI/CD simplifié (Docker)	Lancement rapide en local via Docker pour un environnement homogène et reproductible.

PARTIE 2 – GESTION DE PROJET

2.1 Méthodologie utilisée (Agile / Scrum)

Pour le développement de l'application *Gestion d'Interventions Techniques*, j'ai adopté une approche Agile, plus précisément inspirée du framework Scrum, adaptée à un projet mené individuellement dans le cadre du titre professionnel de Concepteur Développeur d'Applications.

Cette méthode m'a permis de structurer mon travail, de prioriser les tâches et d'assurer un avancement régulier avec des livrables clairs à chaque étape. Voici les grands principes Scrum appliqués au projet :

2.1.1 Découpage en Sprints

Le projet a été découpé en sprints hebdomadaires, chacun correspondant à une itération de développement complète (analyse, développement, tests et validation). Chaque sprint visait à livrer une ou plusieurs fonctionnalités opérationnelles.

Exemples de découpage :

- Sprint 1 : Mise en place de l'environnement de développement et du backend Symfony.
- Sprint 2 : Authentification JWT + Gestion des rôles.
- Sprint 3 : Création des interventions + gestion des statuts.
- Sprint 4 : Intégration du frontend mobile avec Expo.
- Sprint 5 : Planification des rendez-vous.
- Sprint 6 : Finalisation, tests, corrections et documentation.

2.1.2 Backlog produit

Un backlog a été constitué dès le départ, listant l'ensemble des fonctionnalités à implémenter. Chaque élément était priorisé selon sa valeur métier et sa complexité technique.

Le backlog a été mis à jour à la fin de chaque sprint pour :

- Réajuster les priorités,
- Réévaluer les tâches restantes,
- Ajouter de nouveaux besoins identifiés en cours de développement.

2.1.3 Tableau de suivi Kanban

J'ai utilisé **Trello** comme outil de gestion visuelle du projet. Le tableau Kanban comportait trois colonnes principales :

- **À faire** (To Do)
- **En cours** (In Progress)
- **Terminé** (Done)

Chaque tâche était associée à un sprint, une estimation de temps et une fonctionnalité cible. Cela m'a permis de suivre l'avancement et de rester concentré sur les priorités du moment.

2.1.4 Revue de sprint et ajustements

À la fin de chaque sprint, une **revue de sprint** était réalisée :

- Vérification des fonctionnalités livrées,

- Tests unitaires et fonctionnels manuels,
- Réflexion sur les améliorations possibles.

Ces mini bilans m'ont permis d'ajuster la stratégie de développement (réduction de dette technique, meilleure organisation du code, gestion des bugs).

2.1.5 Avantages de l'approche Agile (Scrum)

L'utilisation d'une approche Scrum, bien que adaptée à un contexte individuel, a permis de :

- Structurer efficacement mon temps et mon avancement,
- Livrer un produit fonctionnel étape par étape,
- Réagir rapidement en cas d'erreur ou d'imprévu,
- Garder une documentation à jour tout au long du développement.

