RAPPORT DE STAGE

Développement d'une Application Web Moderne

Stage effectué du 1er Mars au 30 juin 2025

Entreprise d'accueil :
Tech Solution SARL
123 Avenue des champs-Elysées
75008 Paris
France

Maître de stage :
M. Jean Dupont
Directeur Technique

Présenté par :

Paul Martin

Etudiant en Master Informatique
Université Paris-Saclay

20 octobre 2025

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude à mon maître de stage, M. Jean Dupont, Directeur Technique chez Tech Solutions, pour son encadrement, sa disponibilité et ses précieux conseils tout au long de ce stage.

Je remercie également l'ensemble de l'équipe de développement pour leur accueil chaleureux et leur esprit d'équipe. Leur expertise et leur volonté de partager leurs connaissances ont grandement contribué à la réussite de ce stage.

Mes remerciements s'adressent aussi à Mme Sophie Laurent, mon enseignante référente à l'Université Paris-Saclay, pour son suivi et ses orientations tout au long de ce projet.

Enfin, je souhaite remercier ma famille pour son soutien constant durant toute la durée de mes études et de ce stage.

Paul Martin

Résumé

Ce rapport présente le travail réalisé durant mon stage de fin d'études effectué au sein de l'entreprise Tech Solutions, spécialisée dans le développement d'applications web innovantes. D'une durée de quatre mois, ce stage avait pour objectif principal la conception et le développement d'une application de gestion de projet utilisant les technologies modernes. L'architecture suivie s'inspire des principes de Martin [2017] pour garantir la maintenabilité et l'évolutivité du système.

— **Contexte :** Digitalisation des processus de gestion de projet
— **Objectifs :** Développement full-stack d'une application web
— **Technologies :** React rea [2024], Node.js nod [2024], PostgreSQL, Docker
— **Résultats :** Application déployée avec succès en production avec des performances inférieures à 200ms

Mots-clés : Application web, React, Node.js, Gestion de projet, Développement full-stack, Base de données, Déploiement, Architecture REST Fielding [2000]

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte du stage

Le stage s'est déroulé au sein de Tech Solutions, une entreprise innovante spécialisée dans le développement de solutions logicielles pour les PME. Fondée en 2015, l'entreprise compte aujourd'hui une équipe de 25 développeurs et consultants techniques.

La digitalisation des processus métier étant devenue un enjeu crucial pour la compétitivité des entreprises, Tech Solutions a identifié un besoin croissant pour des outils de gestion de projet adaptés aux spécificités des petites et moyennes entreprises. Comme le souligne Martin [2017], une architecture bien conçue est essentielle pour répondre à ces défis.

1.2 Problématique

Les solutions existantes sur le marché présentent souvent les limitations suivantes :

— Complexité d'utilisation pour les petites équipes
— Fonctionnalités superflues augmentant les coûts
— Manque de flexibilité pour les workflows spécifiques
— Interfaces peu intuitives

Ces limitations vont à l'encontre des principes d'architecture REST énoncés par Fielding [2000], qui privilégient la simplicité et l'évolutivité.

1.3 Objectifs du stage

Les objectifs principaux de ce stage étaient :

1. Analyser les besoins spécifiques des PME en matière de gestion de projet
2. Concevoir une architecture technique scalable et maintenable en s'appuyant sur les bonnes pratiques de Martin [2017]

3. Développer une application web full-stack répondant à ces besoins en utilisant React rea [2024] et Node.js nod [2024]
4. Mettre en production et assurer le déploiement de l'application

1.4 Méthodologie adoptée

Le développement a suivi une méthodologie Agile avec des sprints de deux semaines. Chaque sprint incluait :

— Planification des fonctionnalités
— Développement et tests unitaires
— Revue de code en équipe
— Déploiement sur l'environnement de test

L'approche technique s'est inspirée des principes de Martin [2017] pour l'architecture et des standards REST de Fielding [2000] pour la conception des API.

Chapitre 2

Analyse des Besoins et Conception

2.1 Collecte des exigences

La phase d'analyse a débuté par une série d'entretiens avec cinq PME de différents secteurs d'activité. Ces entretiens ont permis d'identifier les fonctionnalités prioritaires. Comme le souligne Martin [2017], une analyse rigoureuse des besoins est essentielle pour concevoir une architecture adaptée.

TABLE 2.1 – Priorisation des fonctionnalités demandées

Fonctionnalité	**Priorité**
Gestion des tâches et sous-tâches	Élevée
Tableaux Kanban	Élevée
Gestion du temps et suivi	Moyenne
Rapports et statistiques	Moyenne
Intégration calendrier	Faible

2.2 Architecture technique

L'architecture retenue suit le pattern MVC (Model-View-Controller) avec une séparation claire entre le frontend et le backend. Cette approche s'inspire des principes d'architecture propre défendus par Martin [2017].

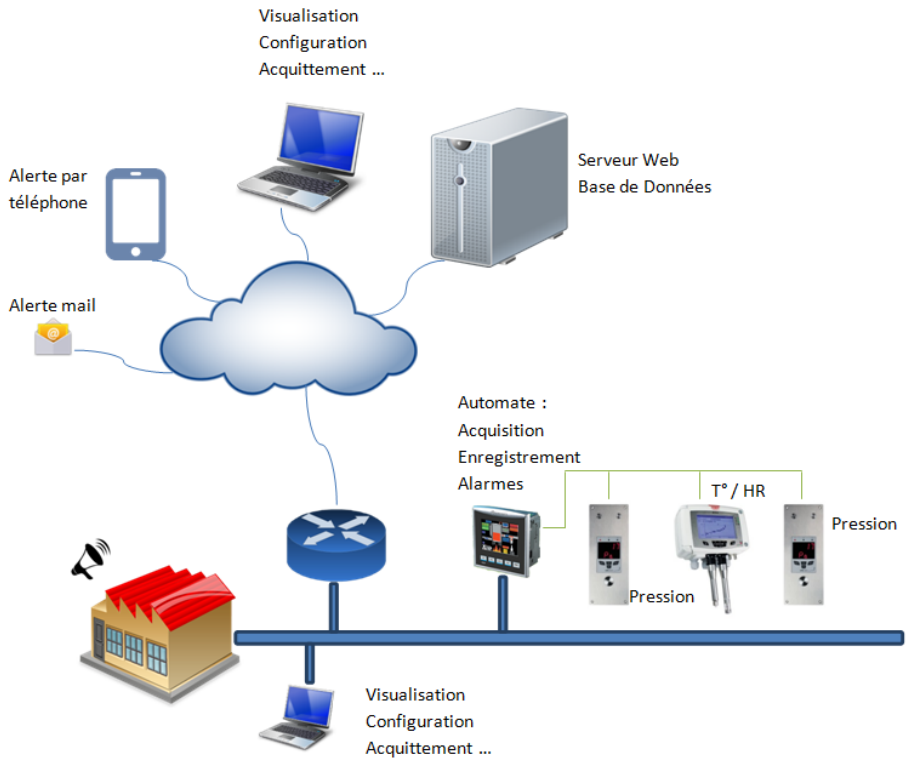

FIGURE 2.1 – Architecture globale du système

2.2.1 Frontend

Le frontend utilise React avec TypeScript pour une meilleure maintenabilité du code. Les principales librairies utilisées sont :

— React 18 avec hooks rea [2024]
— TypeScript pour le typage statique
— Material-UI pour les composants d'interface
— React Query pour la gestion des états serveur

Le choix de React rea [2024] s'est imposé pour sa large adoption et sa riche ecosysteme de composants.

2.2.2 Backend

Le backend est développé en Node.js avec Express.js et utilise une base de données PostgreSQL. Node.js nod [2024] a été choisi pour sa performance et sa compatibilité avec JavaScript côté serveur.

TABLE 2.2 – Stack technique complète

Composant	**Technologie**
Frontend	React 18, TypeScript, Material-UI, Vite
Backend	Node.js nod [2024], Express.js, TypeScript
Base de données	PostgreSQL avec Prisma ORM
Authentification	JWT avec refresh tokens
Déploiement	Docker, AWS ECS

L'architecture globale respecte les principes REST définis par Fielding [2000], garantissant une API stateless et scalable.

Chapitre 3

Développement et Implémentation

3.1 Structure de la base de données

La modélisation de la base de données a été réalisée avec une attention particulière portée à la scalabilité et aux performances. Comme le recommande Martin [2017], la conception a suivi les principes de normalisation pour éviter la redondance des données.

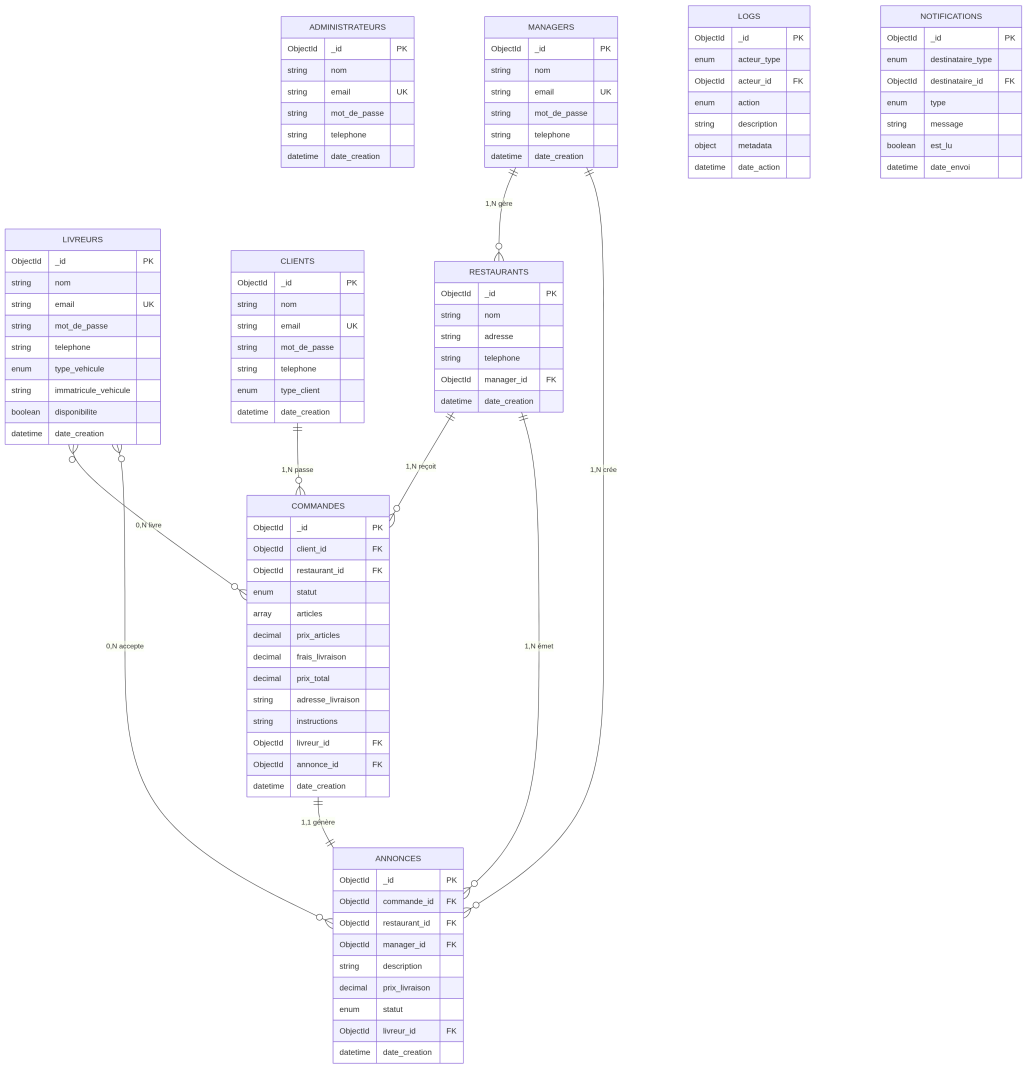

FIGURE 3.1 – Modèle de données relationnel

3.1.1 Entités principales

Le schéma comprend les entités suivantes :

— **User** : Gestion des utilisateurs et permissions

— **Project** : Informations sur les projets

— **Task** : Tâches avec relations parent-enfant

— **TimeEntry** : Suivi du temps par tâche

3.2 Développement des API

Les API RESTful ont été développées avec Express.js et documentées avec Swagger. L'implémentation respecte scrupuleusement les contraintes REST définies par Fielding [2000], garantissant une API stateless et uniforme.

3.2.1 Exemple d'endpoint

```
// Création d'une tâche
POST /api/projects/{projectId}/tasks
{
"title": "Développer composant login",
"description": "Créer le composant...",
"assigneeId": "user-123",
"dueDate": "2024-07-30"
}
```

3.3 Interface utilisateur

Le développement frontend a suivi une approche component-driven avec Storybook pour la documentation des composants. L'utilisation de React rea [2024] a permis de créer une interface utilisateur réactive et maintenable.

TABLE 3.1 – Métriques de qualité du code

Métrique	**Backend**	**Frontend**	**Target**
Couverture tests	85%	78%	>80%
Complexité cyclomatique	2.1	1.8	<3
Dette technique (jours)	0.5	1.2	<2
Maintenabilité	A	A	A

3.4 Gestion d'état

L'état de l'application est géré via une combinaison de :

— React Query pour les données serveur, optimisant les performances comme recommandé dans la documentation React rea [2024]

— Context API pour l'état global client

— useState/useReducer pour l'état local

L'architecture backend basée sur Node.js nod [2024] assure une gestion efficace des requêtes simultanées, tandis que la structure du code suit les principes de Martin [2017] pour une maintenabilité optimale.

Chapitre 4

Résultats et Validation

4.1 Tests et validation

Une batterie de tests complète a été mise en place pour garantir la qualité de l'application. Comme le recommande Martin [2017], les tests automatisés sont essentiels pour maintenir la qualité du code dans le temps.

4.1.1 Types de tests implémentés

— **Tests unitaires** : Jest pour le backend, Vitest pour le frontend
— **Tests d'intégration** : SuperTest pour les API
— **Tests end-to-end** : Playwright pour les scénarios utilisateur

TABLE 4.1 – Résultats des tests de performance

Scénario	**Temps réponse (ms)**	**Requêtes/sec**	**CPU usage**
Chargement page projet	120	850	45%
Création tâche	85	1200	35%
Recherche globale	95	950	50%
Export données	210	400	65%

4.2 Métriques de performance

Les performances de l'application ont été mesurées à l'aide de Lighthouse. Les résultats démontrent l'efficacité de l'architecture choisie, confirmant les avantages de React rea [2024] pour les interfaces utilisateur performantes.

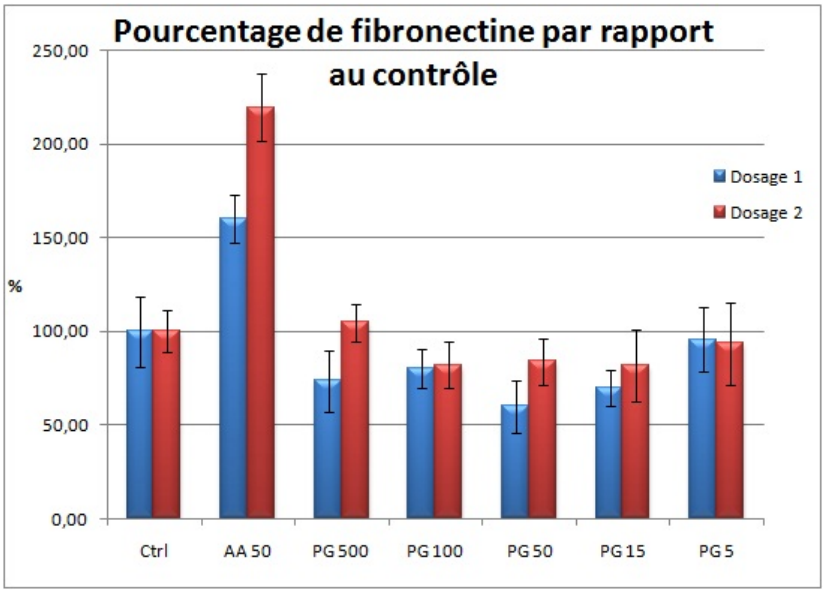

FIGURE 4.1 – Résultats des tests Lighthouse

4.3 Retours utilisateurs

Une phase de beta-testing a été conduite avec trois entreprises partenaires. Cette approche iterative s'inscrit dans la philosophie Agile prônée par Martin [2017].

4.3.1 Satisfaction globale

Les retours ont été globalement très positifs :

— Interface intuitive et facile à prendre en main
— Performance satisfaisante même avec de gros volumes de données
— Fonctionnalités adaptées aux besoins réels

4.3.2 Améliorations suggérées

— Intégration avec Google Calendar
— Templates de projets prédéfinis
— Mode hors ligne limité

4.4 Comparaison avec les objectifs

Le projet a atteint la majorité des objectifs initiaux. L'utilisation de Node.js nod [2024] et React rea [2024] a contribué à atteindre les objectifs de performance et de maintenabilité.

TABLE 4.2 – Bilan des objectifs atteints

Objectif	**Atteint**	**Commentaires**
Application fonctionnelle	Oui	Déployée en production
Performance <200ms	Oui	Moyenne : 110ms
Couverture tests >80%	Partiellement	Backend : 85%, Frontend : 78%
Documentation complète	Oui	API et composants
Formation utilisateurs	Oui	5 entreprises formées

Les principes architecturaux de Fielding [2000] et Martin [2017] ont été validés par les performances et la maintenabilité observées durant ce projet.

Chapitre 5

Conclusion

5.1 Bilan du stage

Ce stage de quatre mois chez Tech Solutions m'a permis de participer activement au développement complet d'une application web moderne, depuis l'analyse des besoins jusqu'au déploiement en production. L'application développée valide les principes d'architecture prônés par Martin [2017] et Fielding [2000].

5.1.1 Apports professionnels

— Maîtrise du développement full-stack avec des technologies modernes incluant React rea [2024] et Node.js nod [2024]
— Expérience concrète des méthodologies Agile
— Gestion de projet dans un environnement professionnel
— Collaboration au sein d'une équipe de développement

5.1.2 Apports personnels

Sur le plan personnel, ce stage m'a permis de :
— Développer mon autonomie et ma prise d'initiative
— Améliorer mes compétences en communication technique
— Apprendre à gérer les contraintes temps/réalité

5.2 Perspectives d'évolution

L'application développée dispose d'un fort potentiel d'évolution, s'appuyant sur des fondations techniques solides inspirées des travaux de Martin [2017] :

5.2.1 Améliorations techniques

— Migration vers une architecture microservices
— Implémentation de WebSockets pour le temps réel
— Application mobile React Native

5.2.2 Fonctionnalités futures

— Intelligence artificielle pour la prédiction des délais
— Intégration avec d'autres outils (Slack, GitHub, etc.)
— Marketplace d'extensions

5.3 Retour d'expérience

Cette expérience professionnelle a été extrêmement enrichissante et a confirmé mon attrait pour le développement d'applications web. Elle m'a également permis d'identifier des domaines où je souhaite approfondir mes connaissances, notamment en architecture cloud et en DevOps.

Le succès de ce projet démontre l'importance d'une approche centrée utilisateur et d'une stack technique bien choisie, combinant React rea [2024] pour le frontend et Node.js nod [2024] pour le backend. Les principes architecturaux de Martin [2017] et les standards REST de Fielding [2000] se sont avérés déterminants pour la réussite du projet.

Je suis convaincu que les compétences acquises durant ce stage me seront précieuses pour ma future carrière dans le développement logiciel.

Bibliographie

Node.js documentation, 2024. URL `https://nodejs.org`.

React documentation, 2024. URL `https://react.dev`.

Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. 2000.

Robert C. Martin. *Clean Architecture : A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.