Projet LogiFête PGI

Solution de Gestion Intégrée pour l'Événementiel

Dossier Complet de Conception et Réalisation

Auteurs	Mohamed Boughmadi Mohamet Thiam
Date	24 novembre 2025
Version	1.0

Année Universitaire 2025-2026

Module : Système d'Information

Table des matières

Chapitre 1

Initialisation et Lancement

Ce premier chapitre pose les bases du projet **LogiFête**. Il définit le périmètre, analyse la faisabilité et planifie la réalisation.

1.1 Cahier des charges fonctionnel

1.1.1 Contexte et Problématique

L'entreprise *LogiFête* loue du matériel audiovisuel et scénique (Son, Lumière, Vidéo, Structure). Actuellement, la gestion est effectuée manuellement ou via des tableurs dispersés, entraînant des erreurs de stock, des pertes de matériel et des oublis de facturation des pénalités de retard.

1.1.2 Objectifs du système

Le projet vise à développer un Progiciel de Gestion Intégré (PGI/ERP) Web permettant de :

— **Centraliser l'information** : Une base de données unique pour les clients, le stock et les commandes.

— **Optimiser la logistique** : Suivi unitaire du matériel par numéro de série et QR Codes.

— **Fiabiliser la facturation** : Calcul automatique des devis, des remises et du ROI (Rentabilité) par produit.

— **Digitaliser les processus** : Signature électronique des bons de livraison sur tablette.

1.1.3 Besoins Fonctionnels (Macro)

Le système devra couvrir les domaines suivants :

— **Gestion Commerciale (CRM)** : Fichiers clients, Devis, Commandes, Factures.

— **Gestion des Stocks** : Entrées/Sorties, Inventaire, Kits, Fournisseurs.

— **Logistique & Technique** : Planification des missions, Signalement de pannes, Maintenance.

— **Administration** : Gestion des utilisateurs, Logs d'audit, Sauvegardes.

1.1.4 Contraintes

— **Techniques** : Solution Full Web (Accessible via navigateur), Stack LAMP (Linux, Apache, MySQL, PHP 8).

— **Ergonomie** : Interface responsive (PC/Tablette) avec charte graphique “Chic & Minimaliste”.

— **Sécurité** : Hachage des mots de passe, protection contre les injections SQL, gestion stricte des rôles (RBAC).

1.2 Étude de faisabilité

1.2.1 Faisabilité Technique

L’analyse du code existant et de l’environnement cible confirme la faisabilité technique :

— **Architecture** : Client/Serveur Web standard, compatible avec l’hébergement actuel (InfinityFree).

— **Compétences** : L’équipe maîtrise PHP natif et SQL.

— **Modules complexes** : Signature électronique (Canvas) et graphiques (Chart.js) déjà fonctionnels.

1.2.2 Faisabilité Économique

— **Coûts de licence** : Nuls, grâce aux technologies Open Source.

— **Infrastructure** : Hébergement mutualisé suffisant.

1.2.3 Faisabilité Organisationnelle

Trois profils utilisateurs existent :

— **Admin**

— **Commercial**

— **Technicien**

1.3 Plan de gestion du projet

1.3.1 Méthodologie

Le projet suit une approche itérative :

1. Spécifications.
2. Conception.
3. Développement.
4. Tests.
5. Déploiement.

1.3.2 Planification (Gantt)

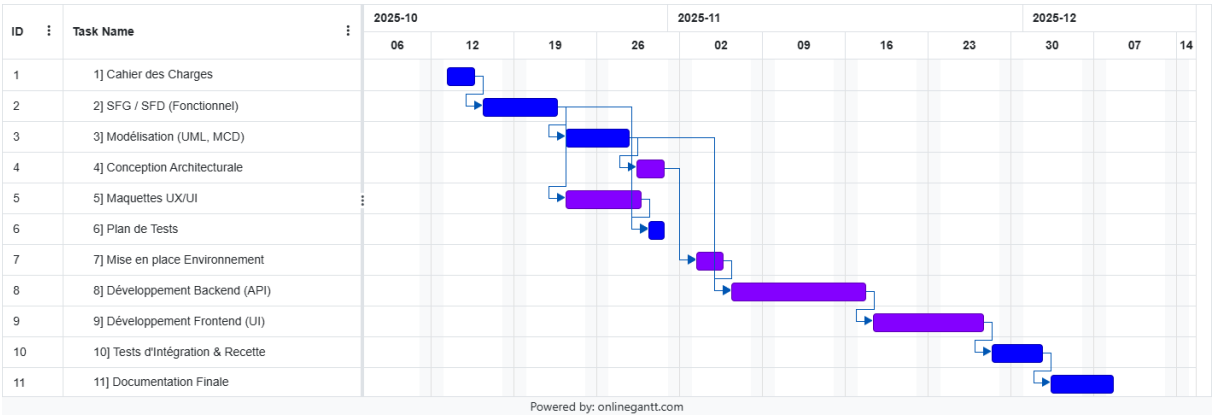

FIGURE 1.1 – Calendrier prévisionnel du projet LogiFête

1.3.3 Gestion des Risques

— **Risque** : Perte de données.

— **Mitigation** : Sauvegardes automatiques et transactions SQL.

Chapitre 2

Analyse et Spécifications

Ce chapitre détaille les fonctionnalités du système, les règles de gestion et les choix techniques.

2.1 Spécifications Fonctionnelles Générales (SFG)

Le système comprend quatre modules principaux.

2.1.1 Module Commercial

— Gestion du fichier clients.
— Édition de devis et commandes.
— Vérification de la disponibilité.
— Génération de PDF.

2.1.2 Module Logistique et Stock

— Inventaire temps réel.
— Gestion des kits.
— Bons de livraison numériques.
— Signature électronique.

2.1.3 Module Technique et Maintenance

— Planification des missions.
— Signalement et suivi des pannes.
— Calcul de rentabilité.

2.1.4 Module Administration

— Gestion des utilisateurs.
— Logs d'audit.
— Sauvegarde de la base.

2.2 Spécifications Fonctionnelles Détaillées (SFD)

2.2.1 Tarification et Pénalités

$$\text{Prix Total} = (\text{Prix Unitaire Journalier} \times \text{Durée}) - \text{Remise}$$

Pénalité de retard :

$$\text{Pénalité} = \text{Prix Journalier} \times \text{Jours de Retard} \times 1.5$$

2.2.2 Cycle de vie du matériel

1. Disponible
2. Loué
3. Panne
4. Maintenance

2.2.3 Signature Électronique

— Signature via Canvas.
— Encodage Base64.
— Stockage en base dans `commande`.

2.2.4 Calcul du ROI

$$\text{ROI (\%)} = \frac{\text{Total Revenus Locatifs} - \text{Coût Réparations}}{\text{Prix Achat Initial}} \times 100$$

2.3 Spécifications Techniques

2.3.1 Architecture Logicielle

— **Client** : Navigateur Web.
— **Serveur** : Apache + PHP.
— **Base** : MariaDB.

2.3.2 Stack Technique

— PHP 8.2
— MariaDB (InnoDB)
— HTML5 / CSS3 / JavaScript ES6
— Chart.js, FullCalendar

2.3.3 Sécurité

- `password_hash()`
- `PDO prepared statements`
- `htmlspecialchars()`
- Vérification de session et rôles RBAC

2.4 Modélisation UML

2.4.1 Cas d'utilisation

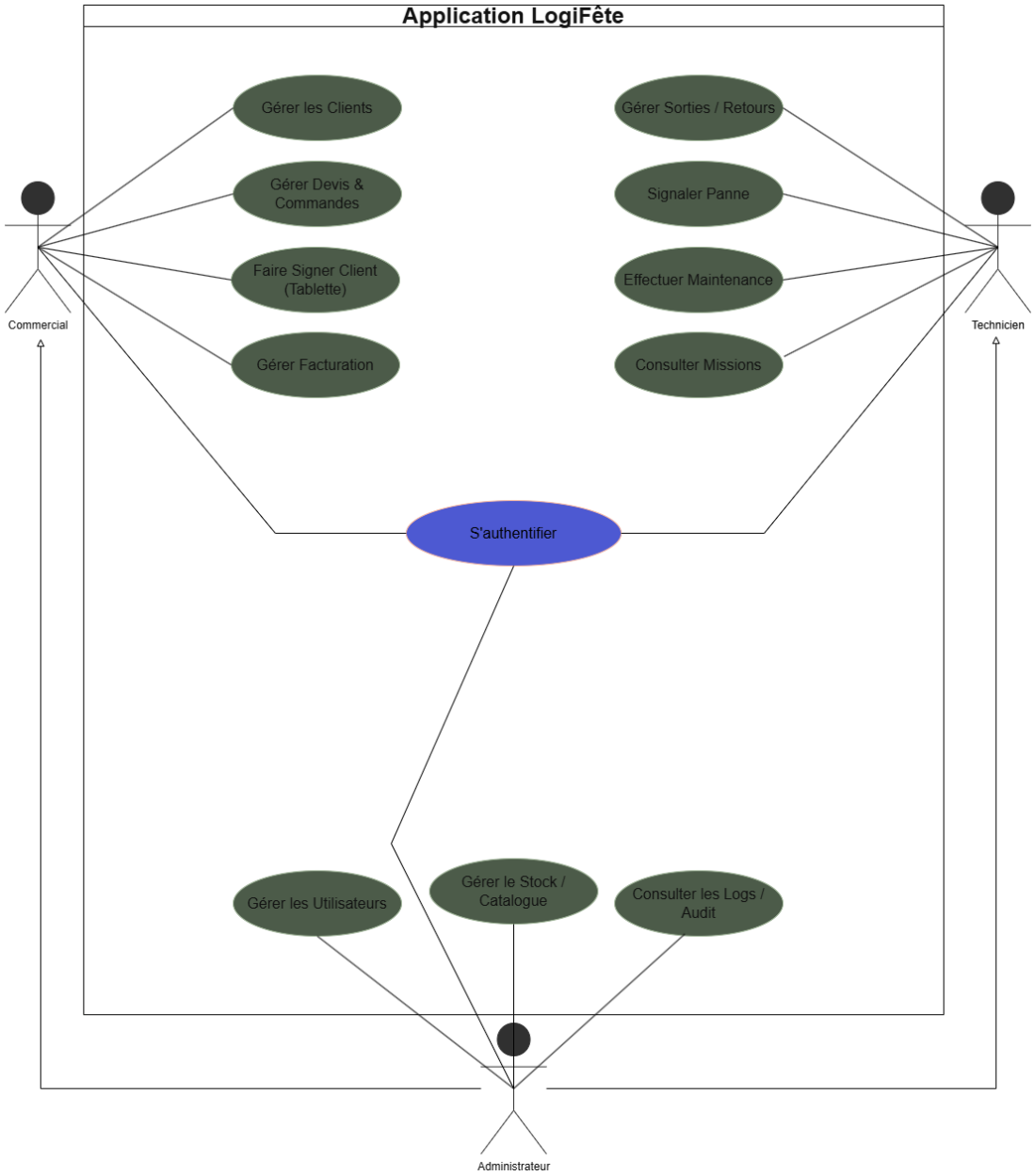

FIGURE 2.1 – Diagramme des Cas d'Utilisation Global

2.4.2 Séquence

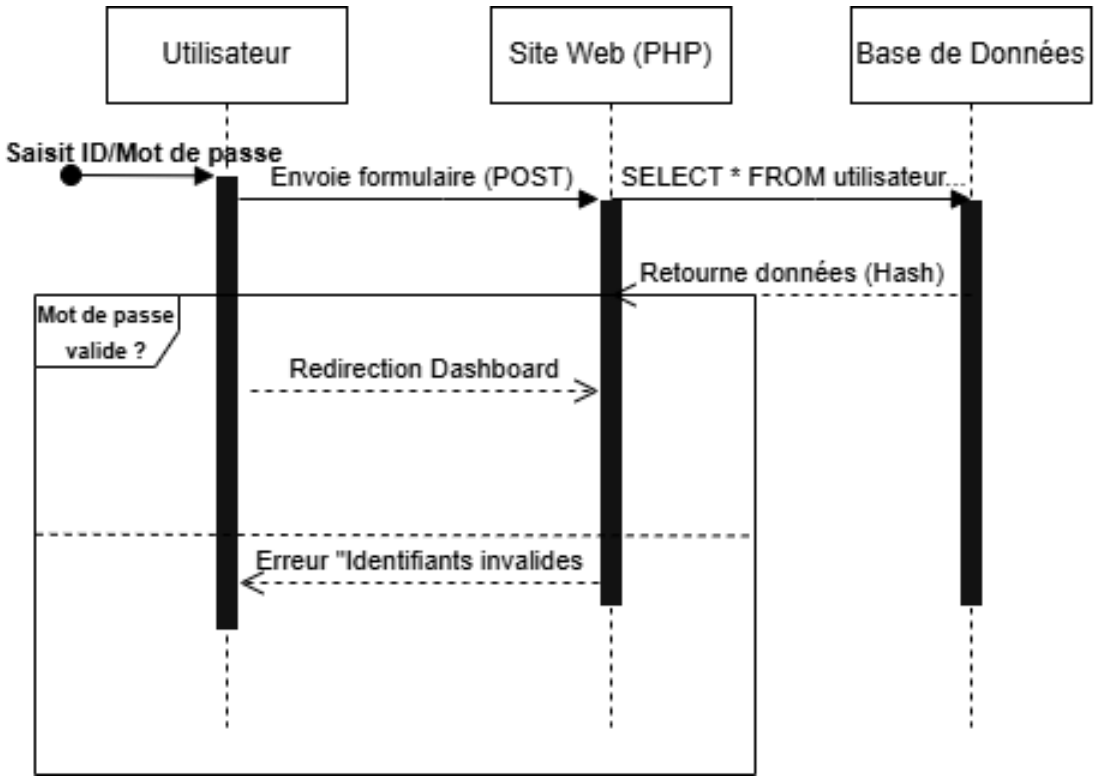

FIGURE 2.2 – Diagramme de Séquence : Authentification

2.4.3 Classes

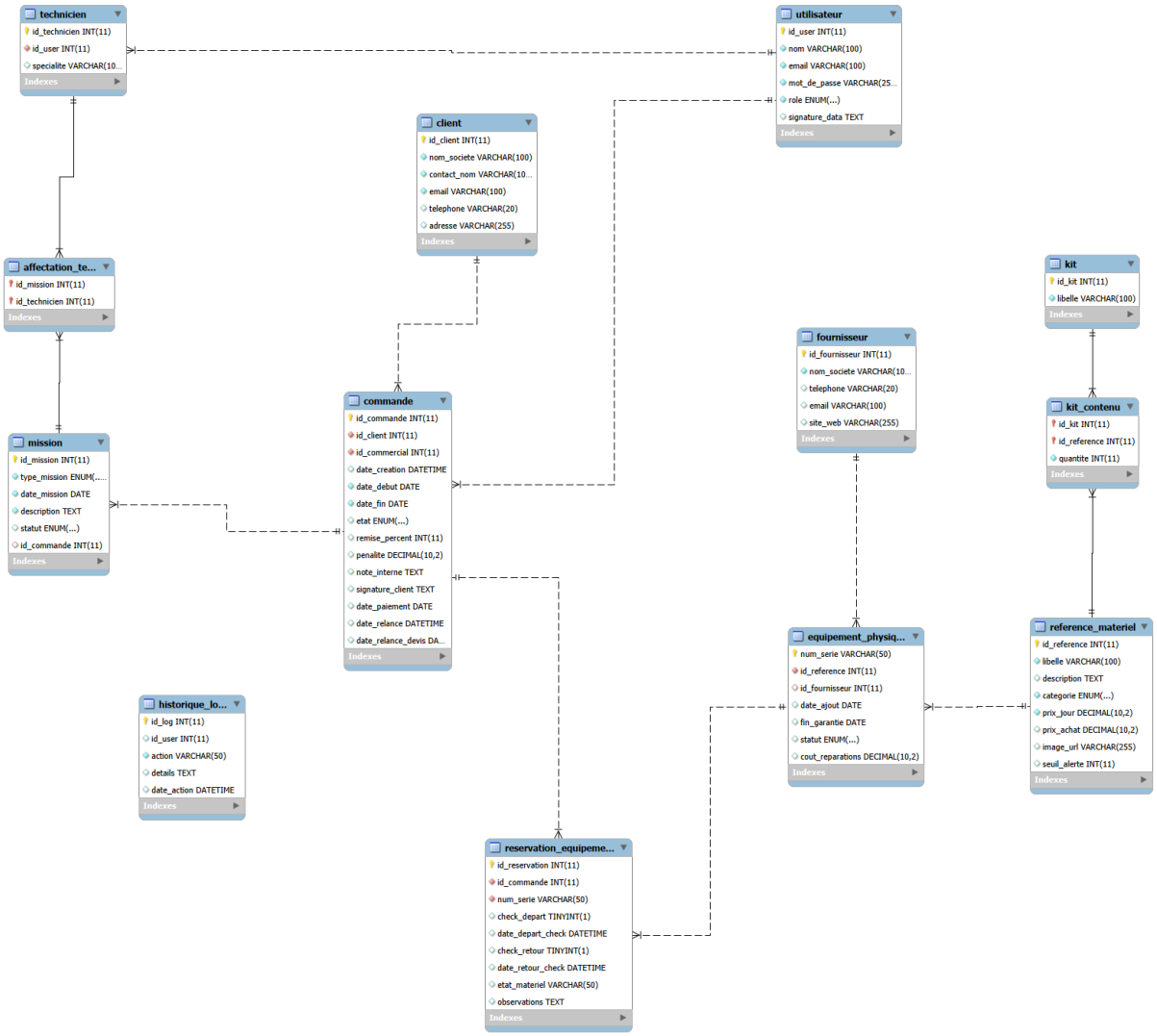

FIGURE 2.3 – Diagramme de Classes d'analyse

Chapitre 3

Conception du Système

Cette phase fait le pont entre l'analyse fonctionnelle (Chapitre 2) et le développement. Elle définit la structure technique de l'application et l'ergonomie des interfaces (livrables 3.1 et 3.2).

3.1 Conception Architecturale

3.1.1 Architecture Physique et Logique

Le système LogiFête repose sur une architecture **3-tiers** Web standard, compatible avec un environnement LAMP.

- **Niveau Client (Front-end)** : Gère l'affichage des pages (HTML/CSS) et l'interactivité (JavaScript). Il est responsable de la capture des interactions utilisateurs, notamment via le **composant Canvas** pour la signature électronique.
- **Niveau Applicatif (Back-end)** : Le serveur Apache exécute les scripts PHP 8.2 (Natif). Il contient la **logique métier** (calculs de ROI, pénalités, validation des accès).
- **Niveau Données (Persistance)** : La base de données MariaDB stocke les informations de manière relationnelle.

3.1.2 Organisation du Code (Dossier Technique)

Bien que le code soit procédural (PHP natif), l'organisation vise une séparation claire des responsabilités, inspirée du modèle MVC :

- **Modèle (Données/Logique)** : Géré par les requêtes PDO dans les scripts et les fonctions globales (`db.php`, `retours.php`).
- **Vue (Présentation)** : Le code HTML et CSS est directement généré par les fichiers PHP, et le design est centralisé dans `style.css`.
- **Contrôleur (Routage)** : Chaque fichier PHP (ex : `clients.php`) sert de contrôleur, gérant la session, traitant le `$_POST` et affichant la vue.

3.2 Conception de la Base de Données

Le Modèle Physique de Données (MPD) est implémenté sous MariaDB avec moteur InnoDB.

3.2.1 Intégrité et Richesse du Modèle

— **Gestion des États** : La table `commande` utilise le type `ENUM` pour garantir des états valides.

— **Contraintes Logiques** : Le champ `date_paiement` est initialisé à `NULL` et mis à jour lors de l'encaissement.

— **Intégrité Physique** : Les clés étrangères assurent la cohérence des données (ex : `equipement_physique` → `reference_materiel`).

3.3 Maquettes et Interface Utilisateur (UI)

L'UI a été conçue pour garantir ergonomie et efficacité.

3.3.1 Charte Graphique et Style

Le thème est "Chic & Minimaliste".

— **Couleurs** : Palette neutre avec accents dorés.

— **Ergonomie** : Badges colorés pour les états des commandes.

— **Navigation** : Barre avec menus déroulants et accès par rôle.

3.3.2 Prototype de Signature

— Grand Canvas HTML5 optimisé pour tablette.

— Conversion Base64 et stockage en base.

3.3.3 Prototypes de Reporting

— Dashboard : Graphiques avec Chart.js.

— Fiche Produit : Indicateur visuel de ROI.

Chapitre 4

Développement et Tests

Cette phase couvre la documentation technique à destination des développeurs et définit la stratégie de validation de la qualité du système, essentielle pour le maintien d'un PGI.

4.1 Documentation Technique (Livrable 4.1)

Le projet étant développé en PHP natif avec un modèle inspiré du MVC, la documentation technique est centrée sur l'organisation des fichiers et l'usage des fonctions critiques.

4.1.1 Organisation des Fichiers et Rôle

L'architecture de l'application est organisée par fonction métier.

- **Cœur / Services :** Le fichier `db.php` centralise la connexion PDO et expose la fonction d'audit `ajouterLog($pdo, $action, $details)`.
- **Navigation / Sécurité :** Le fichier `navbar.php` est une fonction de service critique pour la gestion des droits d'accès et la construction de l'interface utilisateur.
- **Contrôleurs Métier :** Chaque fonctionnalité métier est gérée par un fichier dédié (`clients.php`, `commandes.php`, `missions.php`) qui agit comme un contrôleur, gérant la session, traitant le `$_POST` et affichant la vue.
- **Documents / Impressions :** Les fichiers `facture.php`, `imprimer.php`, `bon_livraison.php` sont des scripts autonomes pour la génération des documents commerciaux et logistiques.

4.1.2 Règles de Codage et Maintenance

- **Audit et Traçabilité :** Tout événement significatif (validation de commande, création d'utilisateur, déclaration de panne) doit appeler la fonction `ajouterLog`. Ceci est central pour le module d'audit `admin_logs.php`.
- **Gestion des Erreurs :** Les blocs `try/catch` (non visibles dans tous les fichiers) doivent être systématiquement utilisés autour des requêtes PDO pour des transactions fiables.
- **Contraintes de Version :** Le système est optimisé pour **PHP 8.2**.

4.2 Plan de Tests (Livrable 4.2)

Le plan de tests vise à garantir la fiabilité des fonctionnalités critiques du PGI (financières, logistiques et sécuritaires).

4.2.1 Stratégie de Test

La stratégie de test adopte une approche de la boîte blanche (vérification interne du code) pour les modules critiques et de la boîte noire (vérification des résultats utilisateur) pour l'interface.

— **Tests Unitaires (Calculs) :** Validation des fonctions de calcul isolées, notamment `calculerMontants` dans `retours.php` pour la logique des pénalités (Jours de Retard × Prix Jour × 1.5).

— **Tests d'Intégration (Flux Métier) :** Validation du cycle complet (Devis → Validation → Mission → Clôture → Facturation).

— **Tests de Performance :** Mesure du temps de chargement des tableaux de bord (KPIs) et de l'inventaire complet (`stock.php`) avec un grand volume de données. L'objectif de temps de réponse ne doit pas excéder 3 secondes.

— **Tests de Sécurité (RBAC et Données) :** Tentatives d'accès aux pages restreintes (`admin_users.php`) avec des rôles non autorisés et vérification de la non-acceptation de chaînes de caractères malveillantes dans les formulaires (`htmlspecialchars()`).

4.2.2 Cas de Tests d'Acceptation (Sélection)

Le succès du projet repose sur la validation des scénarios utilisateurs suivants :

Module	Scénario de Test	Résultat Attendu	Fichiers Impactés
Finance	Clôture d'une commande 2 jours après la `date_fin`.	Application d'une pénalité via `calculerMontants`.	`retours.php`, `commande.penalite`
Logistique	Déclaration d'un équipement `Endommagé`.	Statut = `panne` + création d'une mission réparation.	`bon_livraison.php`, `equipement_physique`
Stock / ROI	Enregistrement d'un coût de 500 €.	Mise à jour de `cout_reparations` + ROI recalculé.	`maintenance.php`, `fiche_produit.php`
Sécurité	Suppression d'un équipement loué.	Blocage via FK + message d'erreur.	`stock.php`, `db.php`

4.3 Rapports de Tests (Livrable 4.3)

4.3.1 Rapport de Validation Fonctionnelle

Ce rapport sera produit suite à l'exécution des tests d'intégration. Il validera que les règles de gestion complexes sont respectées. Une attention particulière sera portée à la

conformité des documents générés (Devis / Facture / Bon).

4.3.2 Rapport de Validation de Données

Le rapport doit s'assurer que :

— L'encaissement (`comptabilite.php`) met correctement à jour le champ `commande.date_paiement`.

— Les exports CSV (`export_finance.php`) sont conformes au format comptable (séparateur ; et BOM UTF-8).

Chapitre 5

Déploiement

Ce chapitre décrit l'ensemble du processus d'installation et de mise en production du Progiciel de Gestion Intégrée **LogiFête**. Il constitue le Livrable 5.1 : *Manuel d'installation.* Les instructions sont adaptées à tout serveur Linux moderne équipé de la stack LAMP.

5.1 Guide d'Installation (Livrable 5.1)

L'application LogiFête repose exclusivement sur des technologies Open Source et ne nécessite aucune dépendance externe complexe. L'installation peut être réalisée en moins d'une heure.

5.1.1 Exigences d'Infrastructure et Prérequis

Le serveur doit disposer des éléments suivants :

- **Système d'exploitation :** Distribution Linux recommandée (Debian, Ubuntu Server, CentOS, AlmaLinux).
- **Serveur Web :** Apache 2.x avec modules activés (`rewrite`, `php8-module`).
- **Base de Données :** MariaDB ou MySQL. Le moteur `InnoDB` est requis pour gérer les clés étrangères et les transactions.
- **Langage :** PHP 8.2 ou supérieur.
- **Extensions PHP obligatoires :**
  - `pdo_mysql`
  - `mbstring`
  - `gd` (pour la signature numérique et les images)
  - `openssl`
- **Configuration PHP :**
  - `session.auto_start = Off`
  - `upload_max_filesize >= 10M` (pour les signatures et images)
  - `post_max_size >= 20M`

Ces prérequis garantissent la pleine compatibilité du PGI avec les modules critiques : authentification, gestion documentaire, signatures numériques, génération de PDF et reporting.

5.1.2 Procédure de Déploiement du Système

Le déploiement de LogiFête suit un processus standard en quatre phases principales.

1. **Création de la Base de Données**
   - Créer une base de données nommée `logifete`.
   - Importer le fichier SQL complet : `logifete_ultimate_demo.sql`. Ce fichier contient :
     - la structure complète (tables, clés étrangères, contraintes),
     - les données minimales,
     - les comptes administrateurs initiaux.

   Note : l'importation peut être réalisée via phpMyAdmin ou via la commande CLI `mysql -u root -p logifete < fichier.sql`.

2. **Configuration de la Connexion à la Base**
   - Ouvrir le fichier `db.php`.
   - Renseigner les paramètres suivants :
     - `$host` = adresse du serveur MariaDB,
     - `$db` = `"logifete"`,
     - `$user` = utilisateur SQL autorisé,
     - `$pass` = mot de passe correspondant.
   - Tester la connexion via `test_db.php` si nécessaire.

3. **Déploiement des Fichiers Applicatifs**
   - Copier tous les fichiers du projet dans le dossier racine du serveur :
     - `/var/www/html/logifete/` (Linux),
     - `htdocs/logifete/` (XAMPP).
   - Vérifier les permissions :
     - `chmod -R 755 .`
     - `chown -R www-data:www-data .` (serveurs Linux).
   - S'assurer que le dossier `/img` est accessible en lecture.

4. **Initialisation et Vérifications**
   - Accéder à : `http://localhost/logifete`.
   - La page `login.php` doit apparaître.
   - Se connecter avec le compte administrateur par défaut : **admin@logifete.com** / **charlie123** (défini dans `reset_live.php`).
   - Vérifier les points suivants :
     - chargement correct de la barre de navigation (`navbar.php`),
     - redirection selon le rôle utilisateur,
     - accès au **Dashboard**,
     - génération PDF des devis/commandes,
     - création d'une commande test.

5.2 Plan de Continuité des Affaires (PCA)

Ce plan garantit la résilience du système face aux incidents techniques, pertes de données ou erreurs humaines.

5.2.1 Sauvegarde et Restauration

— **Sauvegarde :** Le module intégré `admin_backup.php` permet de générer un dump SQL complet de la base. Il est recommandé de programmer une tâche **cron** quotidienne, par exemple :

```
0 3 * * * mysqldump -u root -p logifete > /backups/logifete_$(date +\%F).sql
```

— **Restauration :** En cas d'incident majeur :

  — recréer la base `logifete`,

  — importer le fichier SQL le plus récent.

5.2.2 Gestion des Mises à Jour et Correctifs

— Toute mise à jour (patch, correctif sécurité, ajout de module) doit impérativement être testée en environnement de pré-production (UAT).

— Les fichiers critiques du cœur du PGI (`db.php`, `retours.php`, `commande.php`, `auth.php`) doivent être sauvegardés avant toute modification.

— Le fichier `reset_live.php` peut être utilisé en dernier recours pour réinitialiser un accès administrateur.

Ce processus garantit la continuité de service même en cas d'erreurs de mise à jour ou d'incident technique majeur.

Chapitre 6

Exploitation et Maintenance

Ce chapitre fournit les informations nécessaires à l'utilisation quotidienne, à la maintenance et au support du système **LogiFête**, organisées par profil utilisateur pour le **Manuel d'Utilisation** (Livrable 6.1) et le **Support/FAQ** (Livrable 6.2).

6.1 Manuel Utilisateur (Livrable 6.1)

Le manuel se concentre sur les flux métiers des rôles **Commercial** et **Technicien**, ainsi que sur les opérations d'administration courantes.

6.1.1 Flux Commercial : Devis à Facturation

1. **Création de devis :** Accéder à la page `nouvelle_commande.php`, sélectionner le client, définir les dates, puis ajouter les équipements depuis le stock disponible (`commande_details.php`).
2. **Validation client :** Faire signer le client sur tablette via la page `sign.php` ou valider manuellement le devis. Le statut passe automatiquement de `devis` à `validee`.
3. **Facturation et encaissement :** Une fois la commande validée, générer la facture via `facture.php`. Enregistrer le paiement dans `comptabilite.php`.
4. **Relance de devis :** Utiliser le bouton `→ Relancer Client` dans `commandes.php` pour notifier un client sur un devis ancien.

6.1.2 Flux Technicien : Logistique et Maintenance

1. **Missions quotidiennes :** Consulter le **Planning des Interventions** (`missions.php`) pour visualiser les missions assignées (livraison, récupération, réparation).
2. **État des lieux :** Lors du départ et du retour, utiliser le **Bon de Livraison** (`bon_livraison.php`) pour cocher les états (`check_depart`, `check_retour`) et noter l'état du matériel (`etat_materiel`).
3. **Déclaration de panne :** Pour tout matériel endommagé ou défectueux non lié à une commande, utiliser le formulaire `signaler_panne.php`.
4. **Atelier :** Pour les équipements au statut `panne`, le technicien enregistre le coût de réparation dans `maintenance.php` et remet le matériel à l'état `disponible`.

6.2 Support et FAQ (Livrable 6.2)

6.2.1 FAQ et Procédures de Support Niveau 1

Ce document fournit des réponses rapides aux incidents courants et aux erreurs fréquentes.

— **Q : Le tableau de bord affiche une alerte de stock critique ? R :** Un responsable Stock/Admin doit se rendre sur `admin_stock.php` pour ajouter de nouvelles unités et remonter le stock au-dessus du `seuil_alerte`.

— **Q : Que se passe-t-il si le client rend le matériel en retard ? R :** Le système calcule automatiquement une **pénalité** sur la page `retours.php`, basée sur la durée du retard, qui est ajoutée à la facture finale.

— **Q : Un technicien a oublié son mot de passe ? R :** L'administrateur peut réinitialiser le mot de passe via `admin_users.php`.

— **Q : Comment vérifier la rentabilité d'un équipement ? R :** Scanner le QR Code du matériel ou rechercher son N° de Série dans `stock.php` pour accéder à la `fiche_produit.php` et consulter le **ROI**, intégrant revenus et coûts de maintenance.

6.2.2 Procédures de Maintenance (Admin)

— **Audit des logs :** Consulter régulièrement `admin_logs.php` pour suivre les actions critiques (connexions, suppressions, modifications).

— **Nettoyage des logs :** Le système purge automatiquement les logs de plus de 6 mois pour maintenir la performance de la base.

— **Sauvegarde :** Télécharger un dump SQL depuis `admin_backup.php` au moins une fois par semaine et vérifier l'intégrité des fichiers sauvegardés.

6.2.3 Recommandations Générales d'Exploitation

— Vérifier quotidiennement les alertes sur le tableau de bord (stock, missions, retards).

— Réaliser des tests fonctionnels après toute mise à jour ou correctif.

— Documenter toute anomalie ou incident dans le module audit (`admin_logs.php`).

Conclusion Générale

Le projet **LogiFête PGI** répond pleinement à la problématique initiale de l'entreprise : la nécessité de **centraliser, optimiser et fiabiliser** la gestion de son activité de location de matériel événementiel.

Synthèse des Objectifs Atteints

Ce Dossier de Conception et de Réalisation confirme la faisabilité technique et la robustesse de la solution mise en œuvre sur une **Stack LAMP** (PHP 8.2, MariaDB).

— **Gestion complète du cycle de vie :** La solution couvre l'ensemble des flux métiers, du devis (`nouvelle_commande.php`) à la gestion logistique (`missions.php`), jusqu'à la facturation (`facture.php`) et l'encaissement (`comptabilite.php`).

— **Traçabilité et fiabilité des données :** L'architecture assure l'intégrité de la base de données (clés étrangères) et la traçabilité des actions via la fonction d'audit `ajouterLog`. Les calculs financiers critiques (ROI, pénalités) sont automatisés et fiables.

— **Digitalisation et sécurité :** L'implémentation de la **Signature Électronique** (`sign.php`) et le contrôle d'accès strict par rôles (RBAC) modernisent les processus tout en garantissant un haut niveau de sécurité (`password_hash`, `PDO prepared statements`).

— **Maintenance et exploitation :** Un **Manuel d'Installation** et un **Manuel Utilisateur** détaillé (Chapitres 5 et 6) permettent une prise en main rapide et assurent la pérennité du système pour les équipes IT internes.

Perspectives d'Évolution

Bien que le PGI soit fonctionnel et complet, plusieurs pistes d'évolution sont envisageables :

1. Ajout d'un module de **Planification du Technicien** (Drag-and-Drop) pour optimiser les temps de trajet entre les missions.
2. Intégration d'un module de **Relation Fournisseur** (GRF) pour le suivi automatisé des commandes de matériel et des retours SAV.
3. Développement d'une **API REST** pour permettre l'interconnexion avec d'autres outils (ex : logiciels de comptabilité externes).

Le projet **LogiFête PGI** constitue ainsi un succès, offrant une base solide pour la croissance future et l'évolution des processus de l'entreprise.