WESTELYNCK

CAHIER DES CHARGES COMPLET

Literie Processor

Application de Traitement Automatisé des Devis Matelas

Développé par SCINNOVA pour SAS Literie Westelynck

Projet	**Literie Processor (anciennement MatelasApp)**
Client	**SAS Literie Westelynck**
Développeur	**SCINNOVA**
Version	**3.0.1**
Date	**17/07/2025**
Contact	**sebastien.confrere@scinnova.fr**

Table des Matières

PARTIE 1 : RÉSUMÉ EXÉCUTIF

1. Problématique et solution

La SAS Literie Westelynck traite manuellement des centaines de devis PDF par mois, ce qui représente un défi majeur pour la productivité et la qualité :

- Temps perdu : 4-6 heures par jour de traitement manuel
- Erreurs humaines : 15-20% de corrections nécessaires
- Coût : 2000-3000€/mois en temps de traitement
- Délais : Retards dans la production
- Standardisation : Processus difficile à uniformiser

Solution proposée :

- Application automatisée utilisant l'Intelligence Artificielle
- Analyse automatique des devis PDF avec GPT-4
- Génération de fichiers Excel prêts pour l'inscription
- Standardisation complète du processus

2. Fonctionnalités clés

■ Traitement automatique

- Lecture des PDF avec extraction de texte
- Analyse LLM (GPT-4) pour comprendre le contenu
- Extraction structurée des données (client, matelas, dimensions, prix)
- Validation automatique des informations

■ Génération Excel

- Templates spécialisés par type de matelas
- Remplissage automatique des cellules
- Coloration conditionnelle selon les caractéristiques
- Support de 6 types de matelas différents

■ Interface utilisateur

- Interface moderne PyQt6 avec logo Westelynck
- Drag & drop des fichiers PDF
- Suivi en temps réel du traitement
- Gestion des erreurs et corrections

3. Avantages et ROI

■ Gain de productivité

- Temps de traitement : 30 secondes vs 15 minutes manuellement
- Précision : 95%+ vs 80% manuellement
- Capacité : 10 devis simultanés vs 1 manuellement
- Standardisation : Processus uniforme

■ Retour sur investissement

- Économies : 2000-3000€/mois en temps de travail
- ROI : Amortissement en 3-6 mois
- Scalabilité : Pas de coût supplémentaire pour plus de volume
- Qualité : Réduction drastique des erreurs

4. Technologies et architecture

Technologies utilisées :

- Python 3.8+ : Langage principal
- PyQt6 : Interface graphique moderne
- OpenAI GPT-4 : Intelligence artificielle
- PyPDF2 : Lecture des PDF
- openpyxl : Génération Excel
- Cryptographie : Sécurisation des clés API

Architecture :

```
Frontend (PyQt6) ←→ Backend (Python) ←→ LLM APIs (OpenAI/OpenRouter)
```

- Interface utilisateur PyQt6 moderne
- Modules backend spécialisés par type de matelas
- Intégration multi-providers LLM
- Système de sécurité et chiffrement

5. Planning et métriques

Planning :

■ **Phase 1 : Développement initial (TERMINÉE)**

■ **Phase 2 : Optimisation (EN COURS)**

■ **Phase 3 : Évolutions futures**

Métriques de succès :

- Temps de traitement : < 30 secondes par devis
- Précision d'extraction : > 95%
- Taux d'adoption : 100% des utilisateurs formés
- ROI : Amortissement en 6 mois maximum
- Satisfaction utilisateur : > 90%

PARTIE 2 : CAHIER DES CHARGES DÉTAILLÉ

1. Présentation du projet

1.1 Contexte et objectifs

Projet : Literie Processor (anciennement MatelasApp)
Client : SAS Literie Westelynck
Développeur : SCINNOVA
Date de création : 2025
Version actuelle : 3.0.1

1.2 Problématique

La SAS Literie Westelynck reçoit quotidiennement de nombreux devis de matelas au format PDF. Le traitement manuel de ces devis présente plusieurs défis majeurs :

- Chronophage : Plusieurs heures par jour de traitement manuel
- Source d'erreurs humaines : 15-20% de corrections nécessaires
- Difficile à standardiser : Processus non uniforme
- Coûteux en ressources humaines : 2000-3000€/mois
- Délais de production : Retards dus au traitement manuel

2. Spécifications fonctionnelles

2.1 Fonctionnalités principales

2.1.1 Traitement des devis PDF

- Extraction de texte : Lecture et extraction du contenu textuel des fichiers PDF
- Analyse LLM : Utilisation d'intelligence artificielle pour comprendre et structurer les informations
- Validation : Vérification de la cohérence des données extraites
- Gestion d'erreurs : Traitement robuste des cas particuliers et des erreurs

2.1.2 Génération de fichiers Excel

- Templates spécialisés : Modèles Excel adaptés aux différents types de matelas
- Remplissage automatique : Insertion des données dans les cellules appropriées
- Coloration conditionnelle : Mise en forme visuelle selon les caractéristiques
- Validation des données : Vérification de la cohérence avant export

2.2 Types de matelas supportés

- Latex Naturel - Densité et épaisseur variables
- Latex Mixte 7 Zones - Zones de confort différenciées
- Latex Renforcé - Structure renforcée
- Mousse Viscoélastique - Mémoire de forme
- Mousse Rainurée 7 Zones - Structure aérée
- Select 43 - Mousse haute densité

3. Spécifications techniques

3.1 Architecture générale

3.1.1 Frontend (Interface utilisateur)

- Framework : PyQt6
- Design : Interface moderne et intuitive
- Responsive : Adaptation à différentes résolutions
- Thème : Intégration du logo Westelynck

3.1.2 Backend (Traitement)

- Langage : Python 3.8+
- Modules spécialisés : Un module par type de matelas
- API LLM : Intégration multi-providers
- Gestion des données : JSON et Excel

4. Contraintes et exigences

4.1 Contraintes techniques

4.1.1 Compatibilité

- Systèmes d'exploitation : Windows 10+, macOS 10.15+, Linux
- Python : Version 3.8 ou supérieure
- Mémoire : Minimum 4 GB RAM
- Espace disque : 500 MB pour l'installation

4.1.2 Sécurité

- Chiffrement : AES-256 pour les clés API
- Validation : Vérification des entrées utilisateur
- Isolation : Pas d'accès réseau non autorisé

5. Livrables

5.1 Application principale

- Exécutable standalone : Installation simple
- Interface graphique : PyQt6 moderne
- Documentation : Guide utilisateur complet

5.2 Documentation technique

- Cahier des charges : Ce document
- Documentation développeur : Guide technique
- API documentation : Référence des modules

6. Planning et phases

6.1 Phase 1 : Développement initial (Terminée)

- Architecture : Structure modulaire
- Core modules : Modules de base
- Interface : GUI PyQt6
- Tests : Validation fonctionnelle

6.2 Phase 2 : Optimisation (En cours)

- Performance : Optimisation des traitements
- Robustesse : Gestion d'erreurs avancée
- Documentation : Guides complets
- Packaging : Distribution standalone

7. Tests et validation

7.1 Tests unitaires

- Modules backend : Tests de chaque fonction
- Calculs : Validation des formules
- API : Tests des appels externes

7.2 Tests d'intégration

- Workflow complet : PDF → Excel
- Interface : Tests utilisateur
- Performance : Tests de charge

8. Maintenance et évolution

8.1 Maintenance préventive

- Monitoring : Surveillance des performances
- Logs : Analyse des erreurs
- Mises à jour : Corrections et améliorations

8.2 Maintenance corrective

- Bugs : Correction des erreurs
- Compatibilité : Adaptation aux évolutions
- Sécurité : Corrections de vulnérabilités

9. Risques et mitigation

9.1 Risques techniques

- Évolution des APIs → Monitoring et adaptation
- Compatibilité → Tests réguliers
- Performance → Optimisation continue

9.2 Risques fonctionnels

- Précision LLM → Amélioration des prompts
- Évolution des devis → Adaptation des modèles
- Formats → Support de nouveaux formats

10. Conclusion

Le projet Literie Processor répond parfaitement aux besoins de la SAS Literie Westelynck en automatisant le traitement des devis PDF. L'application combine technologies modernes (IA, interface graphique) et robustesse technique pour offrir une solution complète et évolutive.

Points forts :

- Automatisation complète du processus
- Précision élevée grâce à l'IA
- Interface utilisateur intuitive
- Architecture modulaire et évolutive
- Documentation complète

Perspectives :

- Extension à d'autres types de documents
- Interface web pour accès distant
- Intégration avec les systèmes existants
- Développement d'une version SaaS

PARTIE 3 : SPÉCIFICATIONS TECHNIQUES

1. Architecture système

1.1 Vue d'ensemble

L'application Literie Processor suit une architecture modulaire avec les composants suivants :

- Frontend PyQt6 : Interface utilisateur moderne
- Backend Python : Modules de traitement spécialisés
- APIs externes : OpenAI, OpenRouter pour l'IA
- Système de sécurité : Chiffrement des clés API
- Gestion des données : JSON et Excel

1.2 Communication inter-modules

La communication entre les modules s'effectue via :

- backend_interface.py : Interface principale
- JSON : Format de sérialisation des données
- Exceptions structurées : Gestion d'erreurs
- Logs détaillés : Traçabilité complète

2. Modules détaillés

2.1 Module Interface (app_gui.py)

- Classe MatelasApp : Application principale PyQt6
- Widgets : QFileDialog, QProgressBar, QTextEdit, QTableWidget
- Événements : Drag & drop, sélection multiple, raccourcis clavier
- Threading : Traitement asynchrone avec QThread

2.2 Module Backend Interface

- Workflow de traitement : Validation → Extraction → Analyse → Génération
- Gestion d'erreurs : Fichiers corrompus, APIs indisponibles, parsing échoué
- Validation : Vérification des données avant traitement
- Logs : Traçabilité complète des opérations

3. Structures de données

3.1 Format JSON d'entrée

Le format JSON généré par l'IA contient :

- societe : Informations de la société
- client : Données client (nom, adresse, téléphone)
- configurations : Liste des matelas avec caractéristiques
- total_ht, tva, total_ttc : Calculs financiers

3.2 Format de sortie Excel

- Feuille 1 : Données client et société
- Feuille 2 : Configurations matelas
- Feuille 3 : Configurations sommiers
- Feuille 4 : Résumé et totaux

4. Performance et sécurité

4.1 Métriques de performance

- Temps de traitement : < 30 secondes par devis
- Mémoire utilisée : < 500 MB
- CPU : < 50% d'utilisation moyenne
- I/O : Optimisation des lectures/écritures

4.2 Optimisations implémentées

- Threading : Traitement asynchrone
- Caching : Mise en cache des référentiels
- Lazy loading : Chargement à la demande
- Compression : Réduction de la taille des données

4.3 Sécurité

- Chiffrement AES-256 : Pour les clés API
- Validation : Sanitisation des entrées utilisateur
- Isolation : Contrôle des accès réseau
- Logs d'audit : Traçabilité complète

5. Tests et déploiement

5.1 Tests unitaires

- Modules backend : Tests de chaque fonction
- Calculs : Validation des formules
- API : Tests des appels externes
- Interface : Tests des widgets

5.2 Tests d'intégration

- Workflow complet : PDF → JSON → Excel
- APIs externes : Tests des providers LLM
- Interface : Tests utilisateur
- Performance : Tests de charge

5.3 Déploiement

- PyInstaller : Création d'exécutables standalone
- Cross-platform : Windows, macOS, Linux
- Dépendances : Inclusion automatique
- Installation : Scripts d'installation

Cahier des Charges Complet

Literie Processor

Développé par SCINNOVA

Pour SAS Literie Westelynck

Merci de votre confiance

Contact technique	**sebastien.confrere@scinnova.fr**
Téléphone	**06.66.05.72.47**
Éditeur	**SCINNOVA**
Client	**SAS Literie Westelynck**