ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

RAPPORT DE PROJET DE FIN D'ANNÉE

Projet ERPConnect

Module Intelligent d'Aide à la Décision Intégré à l'ERP

Filière : Ingénierie Informatique et Réseaux

Réalisé par :
Ismail Moustatraf & Ibrahim Hanna

Encadrant Académique :
Abderrahim Larhlimi

Année Académique : 2025/2026

Remerciements

Nous tenons à exprimer notre profonde gratitude à notre encadrant académique, **Abderrahim Larhlimi**, pour ses précieux conseils, son soutien et ses retours constructifs tout au long de la réalisation de ce projet. Son expertise et ses encouragements ont été déterminants dans notre développement professionnel et technique.

Nous remercions également le personnel de l'**EMSI** pour nous avoir fourni l'environnement académique et les outils nécessaires au développement de nos compétences.

Enfin, nous sommes reconnaissants envers nos familles et collègues pour leur soutien constant et leur compréhension durant les phases rigoureuses de ce Projet de Fin d'Année.

Résumé

Le projet **ERPConnect** vise à transformer les données opérationnelles des systèmes ERP (Enterprise Resource Planning) en actifs stratégiques grâce à l'intelligence artificielle. Ce rapport documente le développement d'un module d'aide à la décision conçu pour s'intégrer aux ERP open source tels qu'Odoo. La solution s'appuie sur une architecture de micro-services basée sur **FastAPI** pour fournir des fonctionnalités avancées d'analytique prédictive.

Les principales contributions incluent un système de prévision de la demande basé sur l'apprentissage automatique (Polynomial Regression), un mécanisme de détection d'anomalies par score-Z, et un moteur de recommandation de réapprovisionnement optimisé par des algorithmes de gestion de stock (ROP). Une interface utilisateur en **React** permet aux décideurs de visualiser plus de 30 indicateurs de performance (KPI) et de personnaliser leurs tableaux de bord. Les résultats démontrent l'efficacité de l'approche pour réduire les ruptures de stock et valoriser le patrimoine informationnel de l'entreprise.

Mots-clés : ERP, Intelligence Artificielle, Prévision de la demande, Apprentissage automatique, FastAPI, Aide à la décision.

Abstract

The **ERPConnect** project aims to transform operational data from ERP (Enterprise Resource Planning) systems into strategic assets through Artificial Intelligence. This report documents the development of a decision support module designed to integrate with open-source ERPs such as Odoo. The solution relies on a **FastAPI**-based microservices architecture to provide advanced predictive analytics features.

Key contributions include a machine-learning-based demand forecasting system (Polynomial Regression), a Z-score anomaly detection mechanism, and a replenishment recommendation engine optimized by inventory management algorithms (ROP). A **React** user interface allows decision-makers to visualize over 30 Key Performance Indicators (KPIs) and customize their dashboards. The results demonstrate the effectiveness of the approach in reducing stockouts and maximizing the value of the company's informational assets.

Keywords : ERP, Artificial Intelligence, Demand Forecasting, Machine Learning, FastAPI, Decision Support.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Contexte

À l'ère de la transformation numérique, les entreprises accumulent quotidiennement des volumes considérables de données opérationnelles à travers leurs systèmes d'information. Les systèmes ERP (Enterprise Resource Planning) constituent le cœur de cette infrastructure informationnelle, centralisant les processus de gestion des ventes, des achats, des stocks, de la comptabilité et des ressources humaines. Cependant, malgré cette richesse de données, de nombreuses organisations peinent à en extraire une valeur stratégique réelle.

Les ERP traditionnels excellent dans la gestion transactionnelle et le reporting historique, mais demeurent limités en matière d'analyse prédictive et d'aide à la décision proactive. Les gestionnaires doivent souvent s'appuyer sur leur intuition ou sur des analyses manuelles chronophages dans Excel pour anticiper la demande, optimiser les stocks ou détecter les tendances émergentes. Cette situation engendre des coûts cachés importants : ruptures de stock, surstockage, opportunités commerciales manquées et réactivité insuffisante face aux changements du marché.

Parallèlement, les avancées récentes en intelligence artificielle et en apprentissage automatique offrent des opportunités sans précédent pour transformer ces données brutes en insights actionnables. Les algorithmes de prévision, de classification et de détection d'anomalies peuvent désormais être déployés à grande échelle, même dans des environnements de PME, grâce à des frameworks open source performants et accessibles.

Problématique

Face à ce constat, notre projet s'articule autour de la question centrale suivante :

Comment enrichir les systèmes ERP existants avec des capacités d'intelligence artificielle pour transformer les données opérationnelles en leviers de décision stratégique, sans nécessiter de modifications structurelles lourdes ?

Cette problématique se décline en plusieurs sous-questions techniques et fonctionnelles :

— Comment intégrer de manière non-intrusive un module d'IA avec un ERP open source comme Odoo ?

— Quels algorithmes d'apprentissage automatique sont les plus adaptés pour la prévision de la demande dans un contexte de PME avec des historiques de données limités ?

— Comment calculer automatiquement des points de commande optimaux (ROP) en tenant compte de la variabilité de la demande ?
— Comment segmenter intelligemment un catalogue produit pour prioriser les efforts de gestion ?
— Comment détecter proactivement les anomalies de ventes pour permettre une réaction rapide ?
— Comment présenter ces insights de manière claire et actionnable à travers une interface utilisateur moderne ?

Objectifs du Projet

Le projet **ERPConnect** vise à développer un module intelligent d'aide à la décision qui s'intègre aux systèmes ERP existants pour apporter des capacités analytiques et prédictives avancées. Les objectifs spécifiques sont les suivants :

Objectifs Fonctionnels

1. **Prévision de la demande** : Développer un système de prévision à 30 jours basé sur l'apprentissage automatique (régression polynomiale) avec calcul d'intervalles de confiance.
2. **Optimisation des stocks** : Implémenter un moteur de calcul automatique du point de commande (ROP) et générer des recommandations de réapprovisionnement avec niveaux d'urgence.
3. **Segmentation ABC/XYZ** : Classifier automatiquement les produits selon leur contribution au chiffre d'affaires (ABC) et leur variabilité de demande (XYZ) pour adapter les stratégies de gestion.
4. **Détection d'anomalies** : Identifier les pics et chutes anormales de ventes en temps réel via un algorithme de score-Z.
5. **Tableau de bord analytique** : Concevoir une interface utilisateur moderne affichant plus de 30 KPI calculés automatiquement avec visualisations interactives.

Objectifs Techniques

1. Développer une architecture microservices basée sur **FastAPI** pour le backend
2. Intégrer l'ERP Odoo via le protocole **XML-RPC** sans modification du système source
3. Utiliser **scikit-learn** pour l'implémentation des algorithmes d'apprentissage automatique
4. Concevoir une interface utilisateur responsive en **React**
5. Assurer la persistance des modèles ML via **joblib**

6. Implémenter un système d'authentification sécurisé avec **JWT**
7. Optimiser les performances via un système de cache intelligent
8. Documenter l'API avec **OpenAPI/Swagger**

Méthodologie

Pour atteindre ces objectifs, nous avons adopté une approche méthodologique structurée en plusieurs phases :

1. **Analyse et spécification** : Étude des besoins, analyse de l'existant, définition des exigences fonctionnelles et non-fonctionnelles.
2. **Conception** : Modélisation UML (cas d'utilisation, diagrammes de séquence, diagrammes de classes), architecture logicielle, modèle de données.
3. **Développement itératif** : Implémentation en sprints Agile de 2 semaines avec livraisons incrémentales de fonctionnalités.
4. **Tests et validation** : Tests unitaires, tests d'intégration, validation de la qualité des prévisions (métriques MAE, sMAPE).
5. **Déploiement et documentation** : Mise en production, rédaction de la documentation technique et du rapport final.

Chapitre 1

Présentation du Cadre de Projet

Objectifs du Chapitre

Ce chapitre présente le contexte général du projet ERPConnect. Nous commençons par une analyse critique des solutions ERP existantes, identifions leurs limitations en matière d'aide à la décision, puis proposons notre solution basée sur l'intelligence artificielle. Nous justifions ensuite le choix de notre modèle de développement et présentons le planning prévisionnel du projet.

1.1 Introduction

Dans un environnement économique de plus en plus compétitif, les entreprises doivent exploiter efficacement leurs données opérationnelles pour prendre des décisions stratégiques éclairées. Les systèmes ERP (Enterprise Resource Planning) comme Odoo centralisent une quantité considérable d'informations sur les ventes, les stocks, les achats et les clients. Cependant, ces données restent souvent sous-exploitées faute d'outils d'analyse prédictive intégrés.

Le projet **ERPConnect** répond à cette problématique en développant un module intelligent d'aide à la décision qui s'intègre aux ERP open source. Notre objectif est de transformer les données brutes en insights actionnables grâce à des algorithmes d'apprentissage automatique et des techniques d'analyse avancées.

Ce projet s'inscrit dans le cadre de notre Projet de Fin d'Année (PFA) pour la filière Ingénierie Informatique et Réseaux à l'EMSI, année académique 2025/2026.

1.2 Étude de l'Existant

1.2.1 Description de l'Existant

Les systèmes ERP modernes, notamment Odoo, offrent des fonctionnalités de reporting et de suivi opérationnel. Les capacités actuelles incluent :

— **Rapports de ventes** : Agrégation des données par période, produit, client ou région

— **Gestion des stocks** : Visualisation des niveaux de stock avec seuils d'alerte configurables manuellement

— **Tableaux de bord** : Graphiques de performance commerciale (chiffre d'affaires, marge, rotation)
— **Exports de données** : Extraction en CSV/Excel pour analyse externe dans des outils tiers (Excel, Power BI)
— **KPI prédéfinis** : Indicateurs standards (taux de marge, délai de livraison, taux de service)

Ces fonctionnalités permettent un suivi rétrospectif des opérations mais présentent des limitations importantes pour la prise de décision proactive.

1.2.2 Critique de l'Existant

Malgré leurs atouts, les solutions ERP actuelles souffrent de plusieurs lacunes majeures :

Absence de Prévisions Automatisées

— Aucun mécanisme de prévision de la demande basé sur l'historique
— Les gestionnaires doivent estimer manuellement les besoins futurs
— Risque élevé de ruptures de stock ou de surstockage

Analyse Rétrospective Uniquement

— Les rapports montrent ce qui s'est passé, pas ce qui va se passer
— Détection tardive des tendances et anomalies
— Réactivité insuffisante face aux changements du marché

Seuils d'Alerte Statiques

— Configuration manuelle des points de commande (ROP)
— Pas d'adaptation automatique à la saisonnalité ou aux tendances
— Gestion sous-optimale des stocks

Manque d'Intelligence Décisionnelle

— Absence de segmentation automatique des produits (ABC/XYZ)
— Pas de détection proactive des anomalies de ventes
— Aucune recommandation d'action basée sur les données

Dépendance aux Outils Externes

— Nécessité d'exporter les données vers Excel ou Power BI pour analyses avancées
— Processus manuel, chronophage et sujet aux erreurs
— Rupture de la continuité opérationnelle

1.2.3 Solution Proposée

Pour pallier ces limitations, nous proposons **ERPConnect**, un module intelligent qui enrichit les capacités décisionnelles des ERP existants. Notre solution apporte les innovations suivantes :

Prévision de la Demande par Machine Learning

- Modèles de régression polynomiale entraînés automatiquement
- Prévisions à 30 jours avec intervalles de confiance
- Adaptation continue aux tendances et à la saisonnalité

Optimisation Automatique des Stocks

- Calcul dynamique du point de commande (ROP) basé sur la demande réelle
- Recommandations de réapprovisionnement avec niveaux d'urgence
- Réduction des ruptures de stock et optimisation du fonds de roulement

Segmentation Intelligente ABC/XYZ

- Classification automatique des produits par valeur (ABC) et variabilité (XYZ)
- Stratégies de gestion différenciées par segment
- Priorisation des efforts sur les produits à fort impact

Détection Proactive d'Anomalies

- Algorithme de score-Z pour identifier les pics et chutes anormales
- Alertes en temps réel avec niveaux de gravité
- Investigation rapide des causes racines

Tableau de Bord Analytique Avancé

- Plus de 30 KPI calculés automatiquement
- Interface React moderne et responsive
- Visualisations interactives (graphiques, matrices, tendances)
- Personnalisation par rôle utilisateur

Architecture Modulaire et Extensible

- API REST basée sur FastAPI pour intégration facile
- Compatibilité avec Odoo 14+ et autres ERP via XML-RPC
- Déploiement indépendant sans modification de l'ERP existant
- Code open source et documenté

1.3 Choix du Modèle de Développement

Pour la réalisation d'ERPConnect, nous avons adopté une approche **Agile itérative** inspirée de la méthodologie Scrum, adaptée au contexte académique d'un PFA.

1.3.1 Justification du Choix

Complexité et Incertitude Technique

Le projet combine plusieurs domaines techniques (développement web, apprentissage automatique, intégration ERP) avec des défis d'implémentation initialement incertains. L'approche Agile permet d'expérimenter, d'apprendre et d'ajuster rapidement.

Livraisons Incrémentales

Plutôt que d'attendre la fin du projet pour avoir un système complet, nous livrons des fonctionnalités utilisables à chaque itération :

1. **Sprint 1** : Connexion Odoo et extraction de données
2. **Sprint 2** : Prévision de demande (algorithme simple)
3. **Sprint 3** : Modèles ML et entraînement automatique
4. **Sprint 4** : Segmentation ABC/XYZ et détection d'anomalies
5. **Sprint 5** : Tableau de bord et interface utilisateur
6. **Sprint 6** : Tests, optimisation et documentation

Feedback Continu

Chaque sprint se termine par une démonstration à l'encadrant académique, permettant de valider les choix techniques et d'ajuster les priorités.

Gestion des Risques

L'approche itérative permet d'identifier et de résoudre rapidement les problèmes techniques (performance, qualité des prévisions, intégration).

1.3.2 Organisation des Sprints

Chaque sprint dure 2 semaines et suit le cycle suivant :

— **Planification** (1h) : Sélection des user stories et définition des objectifs
— **Développement** (10 jours) : Implémentation, tests unitaires, documentation
— **Revue** (1h) : Démonstration à l'encadrant et validation
— **Rétrospective** (30min) : Identification des améliorations pour le sprint suivant

1.3.3 Outils de Gestion de Projet

— **Git/GitHub** : Versionnement du code et collaboration
— **Documentation** : README.md, docstrings Python, rapport LaTeX

1.4 Planning Prévisionnel

Le projet s'étale sur 16 semaines (4 mois) de février à mai 2026, organisées en 6 sprints de 2 semaines et 1 mois de finalisation.

1.4.1 Livrables

1. **Code source** : Backend FastAPI + Frontend React (GitHub)
2. **Documentation technique** : README, API docs (FastAPI /docs)
3. **Rapport PFA** : Document LaTeX complet (ce document)
4. **Présentation** : Support PowerPoint pour la soutenance
5. **Démonstration** : Application déployée et fonctionnelle

1.5 Conclusion

Ce chapitre a posé les fondations du projet ERPConnect en analysant les limitations des solutions ERP actuelles et en proposant une approche innovante basée sur l'intelligence artificielle. Notre solution apporte une réelle valeur ajoutée en transformant les données opérationnelles en insights stratégiques actionnables.

Le choix d'une méthodologie Agile itérative nous permet de gérer efficacement la complexité technique tout en livrant des fonctionnalités utilisables à chaque sprint. Le planning prévisionnel sur 16 semaines assure une progression structurée vers nos objectifs.

Chapitre 2

Spécification des Exigences

Objectifs du Chapitre

Ce chapitre établit une spécification complète des exigences d'ERPConnect basée sur le système réellement implémenté. Nous identifions tous les acteurs du système, détaillons les exigences fonctionnelles et non fonctionnelles telles qu'elles sont réalisées dans le code, et présentons la documentation des cas d'utilisation reflétant les véritables endpoints et flux de travail de l'API. Cette spécification sert de documentation précise sur ce que fait le système et comment il fonctionne.

2.1 Introduction

La spécification des exigences transforme les besoins métier en spécifications techniques précises. Pour ERPConnect, cette phase documente les capacités réellement implémentées dans notre module intelligent d'aide à la décision intégré aux systèmes ERP Odoo.

Nos exigences sont dérivées de l'architecture système mise en œuvre, comprenant un backend FastAPI communiquant avec Odoo via XML-RPC, des fonctionnalités de prévision avec machine learning utilisant scikit-learn, des mécanismes d'alerte intelligents, et un tableau de bord KPI complet. Nous distinguons les exigences fonctionnelles (ce que fait le système) et les exigences non fonctionnelles (comment il fonctionne).

Ce chapitre documente les acteurs réels interagissant avec ERPConnect, les cas d'utilisation réellement implémentés sous forme d'API REST, et les attributs de qualité obtenus grâce à nos choix techniques.

2.2 Spécification des Exigences Fonctionnelles

Les exigences fonctionnelles définissent les capacités principales fournies par ERPConnect. Elles sont organisées hiérarchiquement selon les modules de service implémentés.

2.2.1 Intégration et Synchronisation des Données ERP

Établir la Connexion à Odoo via XML-RPC

Le système doit établir et maintenir les connexions aux systèmes ERP Odoo via le protocole XML-RPC. Les paramètres de connexion incluent :

— URL du serveur Odoo
— Nom de la base de données
— Identifiants utilisateur et mot de passe
— Authentification via l'endpoint `/xmlrpc/2/common`

Implémentation : classe `OdooConnector` dans `app/services/odoo_connector.py`

Extraire les Données Produit

Le système doit récupérer les informations produit depuis Odoo, incluant :

— ID produit, nom et code de référence
— Quantité en stock actuelle (`qty_available`)
— Quantité prévue (`virtual_available`)
— Prix unitaire (`list_price`)

Implémentation : `search_read` sur le modèle `product.product`

Extraire les Données de Ventes

Le système doit récupérer les lignes de commandes clients avec filtrage par période, incluant :

— Référence produit
— Quantité commandée
— Prix unitaire
— Date de création

Implémentation : `search_read` sur le modèle `sale.order.line`

Extraire les Mouvements de Stock

Le système doit accéder à l'historique des mouvements de stock incluant réceptions et livraisons depuis le modèle `stock.move`.

2.2.2 Prévision de la Demande

Calculer la Prévision de Demande sur 30 Jours

Le système doit prédire la demande sur 30 jours pour chaque produit selon la méthodologie suivante :

1. Récupérer les lignes de commandes clients pour une période configurable (par défaut : 60 jours)
2. Agréger les quantités vendues par jour
3. Appliquer un lissage par moyenne mobile sur 7 jours pour réduire le bruit
4. Multiplier la demande quotidienne actuelle par 30 pour obtenir la prévision

Algorithme : Lissage par moyenne mobile
Implémentation : `predict_demand()` dans `app/services/ai/demand.py`
Endpoint : `GET /ai/demand?lookback_days=60&limit=200`

Détecter les Tendances de Demande

Le système doit classifier les tendances de demande comme HAUSSE, BAISSE ou STABLE en calculant la pente entre les fenêtres temporelles lissées initiales et finales :

— Tendances = HAUSSE si pente >0.2
— Tendances = BAISSE si pente <-0.2
— Tendances = STABLE sinon

Calculer les Scores de Confiance

Le système doit calculer des scores de confiance (0.0 à 1.0) basés sur :
— Couverture des données (nombre de jours de données historiques disponibles)
— Variabilité de la demande (coefficient de variation)
— Nombre d'échantillons (transactions de vente)
Formule : $confiance = 0.4 \times couverture + 0.3 \times (1 - variabilit) + 0.3 \times boost\_chantillon$

2.2.3 Prévision avec Machine Learning

Entraîner des Modèles de Régression Linéaire

Le système doit entraîner des modèles de prévision spécifiques aux produits en utilisant :
— **Algorithme :** scikit-learn LinearRegression
— **Ingénierie des fonctionnalités :** PolynomialFeatures avec degré = 2
— **Données d'entraînement :** ventes quotidiennes lissées (moyenne mobile 7 jours)
— **Période d'historique :** configurable (par défaut : 180 jours, minimum : 14 jours)
Implémentation : `train_product_model()` dans `app/services/ai/ml_forecast.py`
Persistance du modèle : sérialisation joblib vers `models/forecasts/product_{id}.joblib`

Générer des Prévisions Multi-Horizons

Le système doit produire des prévisions pour des horizons temporels configurables (par défaut : 30 jours) avec :

— Tableau quotidien de prévision
— Totaux agrégés : 7 jours, 30 jours, 90 jours
— Intervalles de confiance (niveau 95%) basés sur l'écart-type des résidus : $IC = \hat{y} \pm 1.96\sigma$

Endpoint : `GET /ai/forecast/{product_id}?horizon_days=30`

Implémentation : `app/tasks/scheduler.py`

2.2.4 Recommandations de Réapprovisionnement

Calculer le Point de Commande (ROP)

Le système doit calculer le ROP selon la formule :

$$ROP = ventes\_quotidiennes\_moyennes \times delai\_approvisionnement\_jours + ventes\_quotidiennes\_moy$$

Paramètres :

— `default_lead_time_days` : configurable (par défaut : 7)
— `safety_stock_days` : configurable (par défaut : 3)

Générer des Suggestions de Réapprovisionnement

Le système doit recommander des quantités à commander :

$$quantit\_suggre = \max(0, ROP - stock\_actuel)$$

Catégories d'Action :

— **COMMANDER MAINTENANT** : stock actuel inférieur au ROP et quantité suggérée >0
— **SURVEILLER** : couverture de stock entre 7-14 jours
— **OK** : stock suffisant

Endpoint : `GET /ai/replenishment/with_rop`

Implémentation : `app/routers/ai.py::replenishment_with_rop()`

2.2.5 Segmentation des Produits (Analyse ABC/XYZ)

Classification ABC par Chiffre d'Affaires

Le système doit segmenter les produits en trois catégories selon la contribution au chiffre d'affaires :

— **Classe A** : Top 20% du chiffre d'affaires cumulé
— **Classe B** : 30% suivants (percentile 20-50)
— **Classe C** : 50% restants

Classification XYZ par Variabilité

Le système doit classifier les produits par variabilité de demande en utilisant le coefficient de variation ($CV = \sigma/\mu$) :

— **Classe X** : faible variabilité (CV <0.3)
— **Classe Y** : variabilité moyenne (0.3 $\leq$ CV <0.7)
— **Classe Z** : forte variabilité (CV $\geq$ 0.7)

Générer la Matrice de Segmentation

Le système doit produire une matrice 3×3 affichant le nombre de produits par segment ABC-XYZ avec recommandations stratégiques (ex. "Se concentrer sur les articles AX pour un contrôle strict des stocks").

Endpoint : `GET /ai/segmentation?days=60`
Implémentation : `segment_abc_xyz()` dans `app/services/ai/segmentation.py`

2.2.6 Détection d'Anomalies de Ventes

Détecter les Anomalies Statistiques

Le système doit identifier les pics et chutes de ventes via la méthode du z-score :

$$z = \frac{x - \mu}{\sigma}$$

— Signaler une anomalie si $|z| \geq seuil$ (par défaut : 3.0)
— Classer comme PIC si $z > 0$, CHUTE si $z < 0$

Catégoriser la Sévérité des Anomalies

— **Élevée** : $|z| \geq 4.0$
— **Moyenne** : $3.0 \leq |z| < 4.0$
— **Faible** : seuils inférieurs configurables

Endpoint : `GET /ai/anomalies?days=30&z=3.0`
Implémentation : `detect_sales_anomalies()` dans `app/services/ai/anomalies.py`

2.2.7 Système de KPI (Indicateurs Clés de Performance)

Calculer les Indicateurs Métiers

Le système doit calculer plus de 30 KPI répartis en six catégories :

Chiffre d'affaires & Ventes (5 métriques) :

— Chiffre d'affaires total, croissance CA %, marge brute %
— Valeur moyenne des commandes, CA par produit

Commandes & Clients (4 métriques) :

— Nombre total de commandes, croissance des commandes %

— Taux de fulfillment %, commandes en retard %

Inventaire & Stock (6 métriques) :

— Valeur du stock, taux de rotation, couverture de stock en jours

— Nombre de stocks faibles, nombre de ruptures, produits sous ROP

Produits (4 métriques) :

— Produits actifs, nombre total de produits, nouveaux produits (30j), prix moyen

IA & Prévisions (4 métriques) :

— Précision des prévisions (sMAPE), nombre d'anomalies (7j/30j)

— Produits avec modèles ML, dernière date d'entraînement

Efficacité (3 métriques) :

— Délai moyen de traitement, cycle de commande, livraison à temps %

Fournir les Endpoints KPI

— `GET /kpi/metrics?days=30` — Tous les KPI avec valeurs actuelles

— `GET /kpi/metric/{name}?days=90` — KPI spécifique avec historique

— `GET /kpi/comparison?product_ids=101,102` — Comparaison multi-produits

— `GET /kpi/catalog` — Catalogue complet pour le constructeur de tableau de bord

Implémentation : `app/routers/kpi.py`

2.2.8 Visualisation du Tableau de Bord

Fournir l'Endpoint de Vue d'ensemble

`GET /dashboard/overview` doit retourner :

— Nombre total de produits, valeur du stock, nombre de ventes récentes

— Chiffre d'affaires et statistiques de commandes pour la période actuelle

Générer les Tendances de Ventes

`GET /dashboard/sales_trends?period=daily&days=30` doit fournir des séries temporelles agrégées par :

— Jour, semaine ou mois

— Quantité vendue et chiffre d'affaires

Identifier les Produits Phare

`GET /dashboard/top_products?metric=quantity&days=30&limit=10` doit classer les produits par :

— Quantité vendue

— Chiffre d'affaires généré

Résumer l'État du Stock

`GET /dashboard/stock_status` doit catégoriser l'inventaire :
— Rupture de stock (qty = 0)
— Stock faible (qty <seuil)
— Stock adéquat
— Sur-stock

2.2.9 Authentification et Autorisation

Authentification JWT

Le système doit implémenter le flux OAuth2 par mot de passe avec tokens JWT :
— `POST /auth/login` — Échange des identifiants contre un token d'accès
— Expiration du token : 480 minutes (8 heures)
— Algorithme : HS256
Implémentation : bibliothèque python-jose, `app/auth.py`

Contrôle d'Accès par Rôle

Trois rôles prédéfinis avec permissions différentes :
— **admin** : accès complet, gestion des utilisateurs, configuration système
— **manager** : visualisation du tableau de bord, accès aux prévisions, génération de rapports
— **viewer** : accès lecture seule aux dashboards et KPI
Utilisateurs de démo :
— `admin@erp.com` / `admin123`
— `manager@erp.com` / `manager123`
— `viewer@erp.com` / `viewer123`
Remarque : Les mots de passe en clair sont utilisés à des fins de démonstration ; pour un déploiement en production, il est nécessaire d'utiliser un hachage bcrypt.

Protéger les Endpoints

Tous les endpoints API (sauf `/auth/login`) doivent nécessiter un token JWT valide via l'en-tête `Authorization: Bearer token`.

2.2.10 Surveillance de l'État du Système

Vérification de Santé de l'Application

`GET /health/app` doit retourner :

— État du scheduler (en cours / arrêté)
— Horodatage du dernier job d'entraînement

Vérification de la Connexion Odoo

`GET /health/odoo` doit vérifier :
— Connectivité au serveur Odoo
— Statut d'authentification
— Accessibilité de la base de données

2.3 Spécification des Exigences Non Fonctionnelles

Les exigences non fonctionnelles définissent les attributs de qualité assurant qu'ERPConnect fournit des performances professionnelles et fiables.

2.3.1 Exigences de Performance

Temps de Réponse

— Endpoints en cache : <10ms
— Endpoints dashboard non mis en cache : <500ms
— Génération de prévisions ML : <5 minutes pour 1000 produits

Stratégie de Cache

— Cache en mémoire basé sur TTL implémenté
— Durée du cache : 90-120 secondes selon l'endpoint
— Endpoints mis en cache : `/dashboard/`, `/ai/`, `/kpi/*`
Implémentation : `app/services/cache.py`

Optimisation de la Récupération de Données

— Requêtes XML-RPC groupées pour réduire les allers-retours
— Appel unique $\texttt{search}_r eadpourtouslesproduits(limite : 50000 enregistrements)$

```
-- Agrégation effectuée en mémoire pour éviter les appels API
   répétés
```

2.3.2 Exigences de Scalabilité

Capacité de Volume de Données

— Support de catalogues jusqu'à 10 000 produits

— Gestion de 100 000 transactions de vente dans la fenêtre historique
— Stockage de modèles ML pour jusqu'à 500 produits simultanément

Utilisateurs Simultanés

— Support de 50 à 100 requêtes API simultanées
— CORS activé pour plusieurs clients frontend

2.3.3 Exigences de Sécurité

Sécurité de l'Authentification

— Signature JWT avec clé secrète (configurable via l'environnement)
— Expiration des tokens appliquée (8 heures)
— Schéma de sécurité `OAuth2PasswordBearer`

Sécurité de l'API

— Middleware CORS avec origines configurables
— Validation des entrées via modèles Pydantic
— Gestion des exceptions pour éviter la fuite d'informations

Gestion des Identifiants

— Identifiants Odoo stockés dans les variables d'environnement (`.env`)
— Clé secrète JWT externalisée
— Pas de données sensibles codées en dur dans le code source

2.3.4 Fiabilité et Disponibilité

Gestion des Erreurs

— Dégradation progressive : prévision de base si ML non disponible
— Gestion des exceptions pour les problèmes de connectivité Odoo
— Jeux de résultats vides gérés sans crash

Persistance des Modèles

— Modèles ML persistés sur le système de fichiers via joblib
— Échecs de chargement de modèle gérés avec mécanismes de secours
— Résultats d'entraînement enregistrés pour le débogage

2.3.5 Exigences de Maintenabilité

Organisation du Code

- Architecture modulaire : séparation des routers et services
- Algorithmes IA isolés dans le répertoire `app/services/ai/`
- Séparation claire : connecteurs, logique métier, couche API

Documentation

- Documentation automatique FastAPI OpenAPI à `/docs`
- Docstrings inline pour les algorithmes complexes
- README avec démonstrations rapides et points de discussion

Standards de Développement

- Annotations de type Python pour les signatures de fonction
- Modèles Pydantic pour validation des requêtes/réponses
- Configuration basée sur l'environnement (pas de valeurs codées en dur)

2.3.6 Compatibilité et Intégration

Compatibilité ERP

- Versions Odoo : 14.0+ (testé)
- Support Dolibarr : théorique (protocole XML-RPC compatible)
- Versionnage API : endpoints `/xmlrpc/2/`

Environnement de Déploiement

- Python 3.8+ requis
- Multi-plateforme : Windows, Linux, macOS
- Dépendances légères (GPU non requis)
- Prêt pour containerisation (pas de dépendances OS spécifiques)

Intégration Frontend

- API RESTful avec réponses JSON
- CORS activé pour clients web
- Endpoints conçus pour consommation par React/Vue.js

2.4 Conclusion

Ce chapitre a fourni une spécification complète des exigences d'ERPConnect basée sur le système réellement implémenté. Nous avons détaillé les exigences fonctionnelles organisées par module de service (prévision de la demande, prévision ML, réapprovisionnement, segmentation, anomalies, KPI), en spécifiant les algorithmes exacts, formules et endpoints API.

Les exigences non fonctionnelles abordent la performance (cache, requêtes groupées), la scalabilité (limites produits/transactions), la sécurité (authentification JWT, CORS), la fiabilité (gestion des erreurs, persistance des modèles), la maintenabilité (code modulaire, documentation) et la compatibilité (Odoo 14+, Python 3.8+).

L'identification de quatre types d'acteurs (Administrateur Système, Manager, Analyste de Données, ERP Odoo) clarifie les niveaux d'accès alignés avec le système de rôles JWT. La spécification détaillée des cas d'utilisation sous forme de tableaux documente les flux API réels, fournissant des critères d'acceptation directement traçables au code source.

Avec des exigences documentées de manière précise et reflétant l'implémentation réelle, nous passons au Chapitre 3 qui modélisera la conception du système à travers des diagrammes UML et des vues architecturales.

Chapitre 3

Conception du système

3.1 Introduction

Ce chapitre présente la conception du système ERPConnect. Il décrit les choix architecturaux ainsi que les modèles UML utilisés afin de représenter le comportement dynamique et la structure statique de l'application.

La conception repose sur des diagrammes UML permettant de garantir la cohérence entre les besoins fonctionnels et l'implémentation réelle du système.

3.2 Modélisation dynamique

La modélisation dynamique permet de représenter les interactions entre les acteurs et les composants du système au cours du temps.

3.2.1 Diagrammes de séquences

Diagramme de séquence : Authentification utilisateur (JWT)

Ce diagramme illustre le processus d'authentification d'un utilisateur. Après la soumission des identifiants, le service d'authentification valide les données et génère un jeton JWT permettant l'accès sécurisé aux fonctionnalités du système.

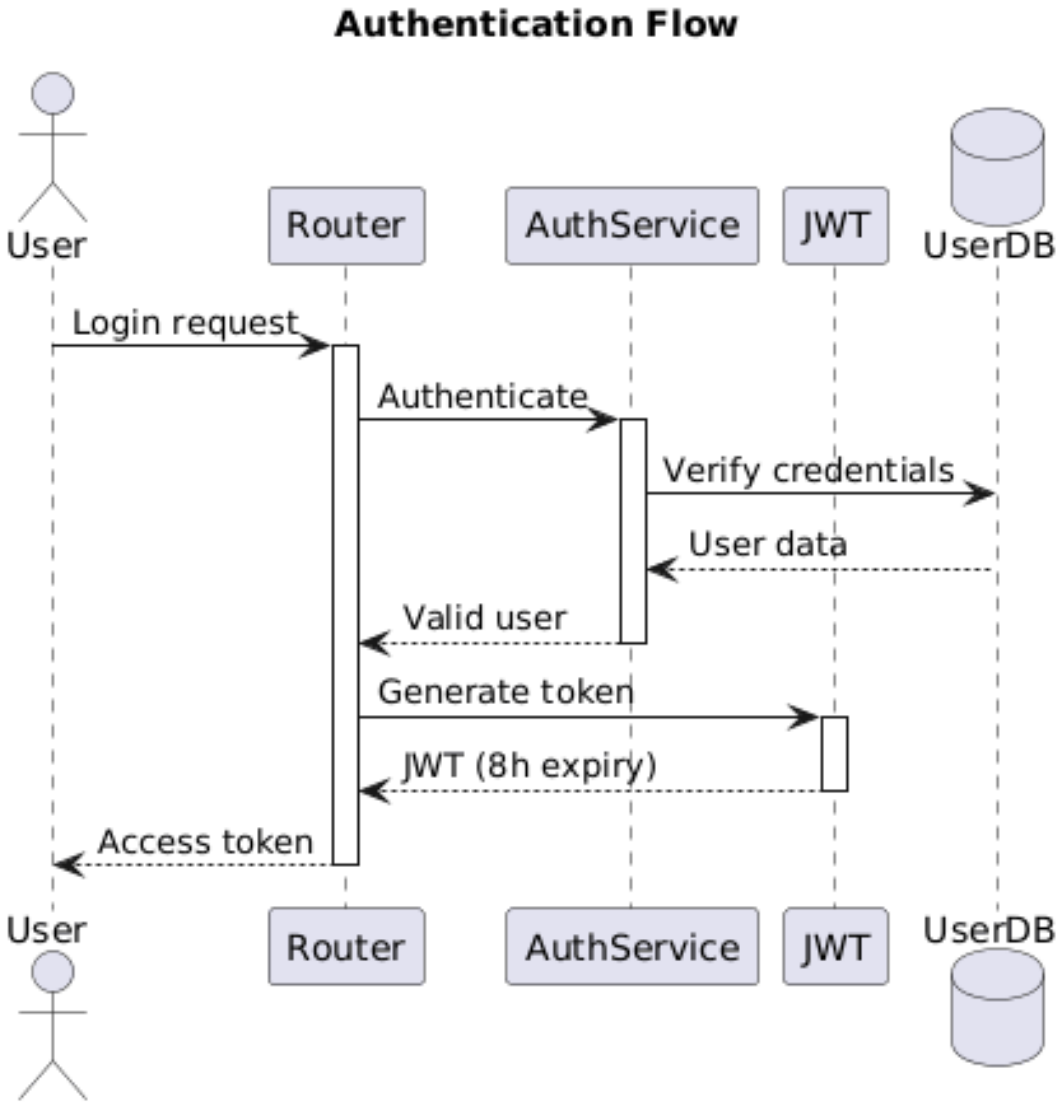


FIGURE 3.1 – Diagramme de séquence – Authentification JWT

Diagramme de séquence : Génération des prévisions

Ce diagramme décrit le flux de génération des prévisions de ventes à partir des données récupérées depuis l'ERP Odoo et traitées par les modèles de machine learning.

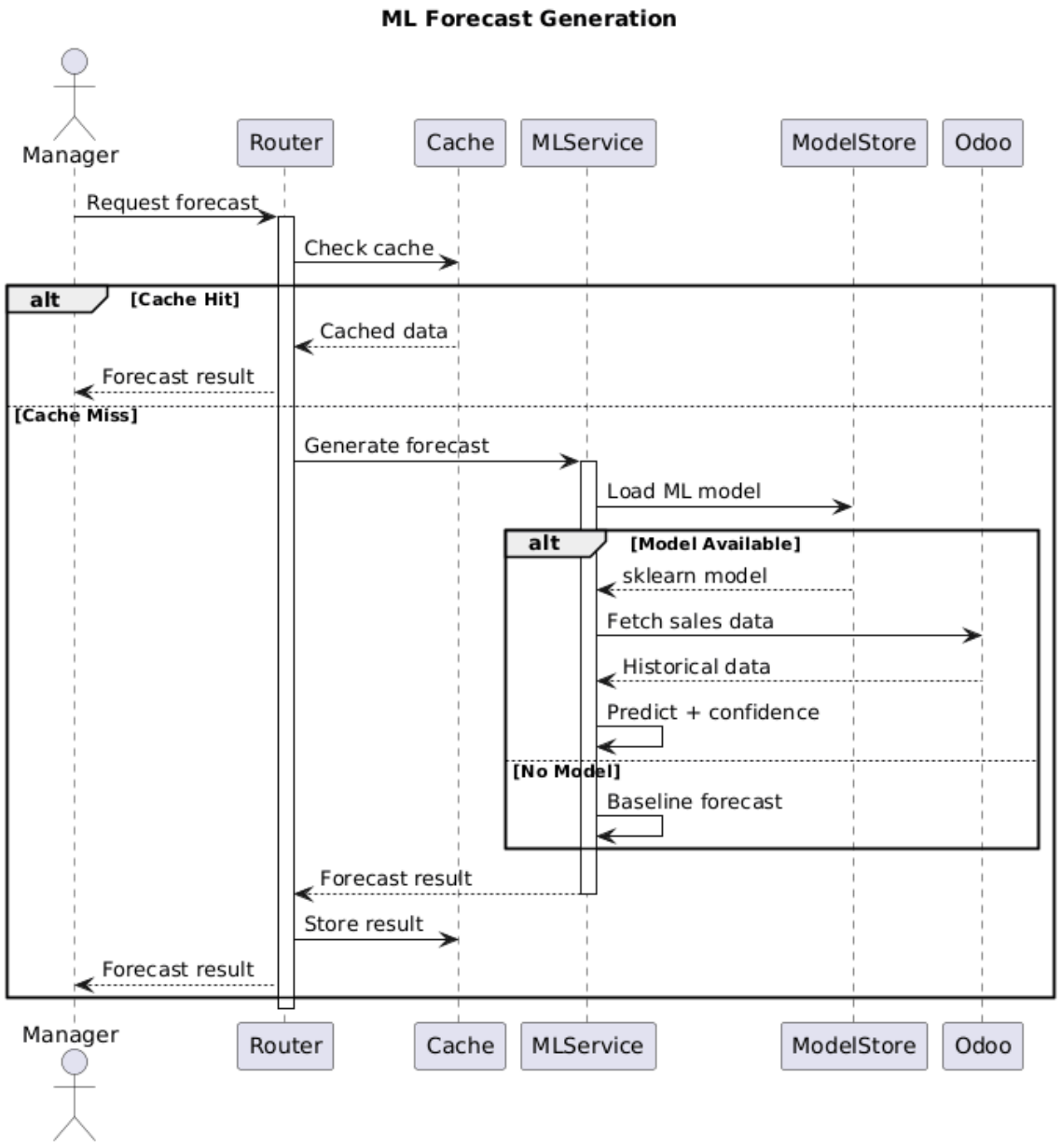


FIGURE 3.2 – Diagramme de séquence – Génération des prévisions

3.2.2 Diagrammes d'états

Diagramme d'états : Gestion d'une Session Utilisateur

Ce diagramme illustre les états d'une session utilisateur dans ERPConnect. Il montre la transition entre les états : connexion, authentification réussie, activité, inactivité et déconnexion.

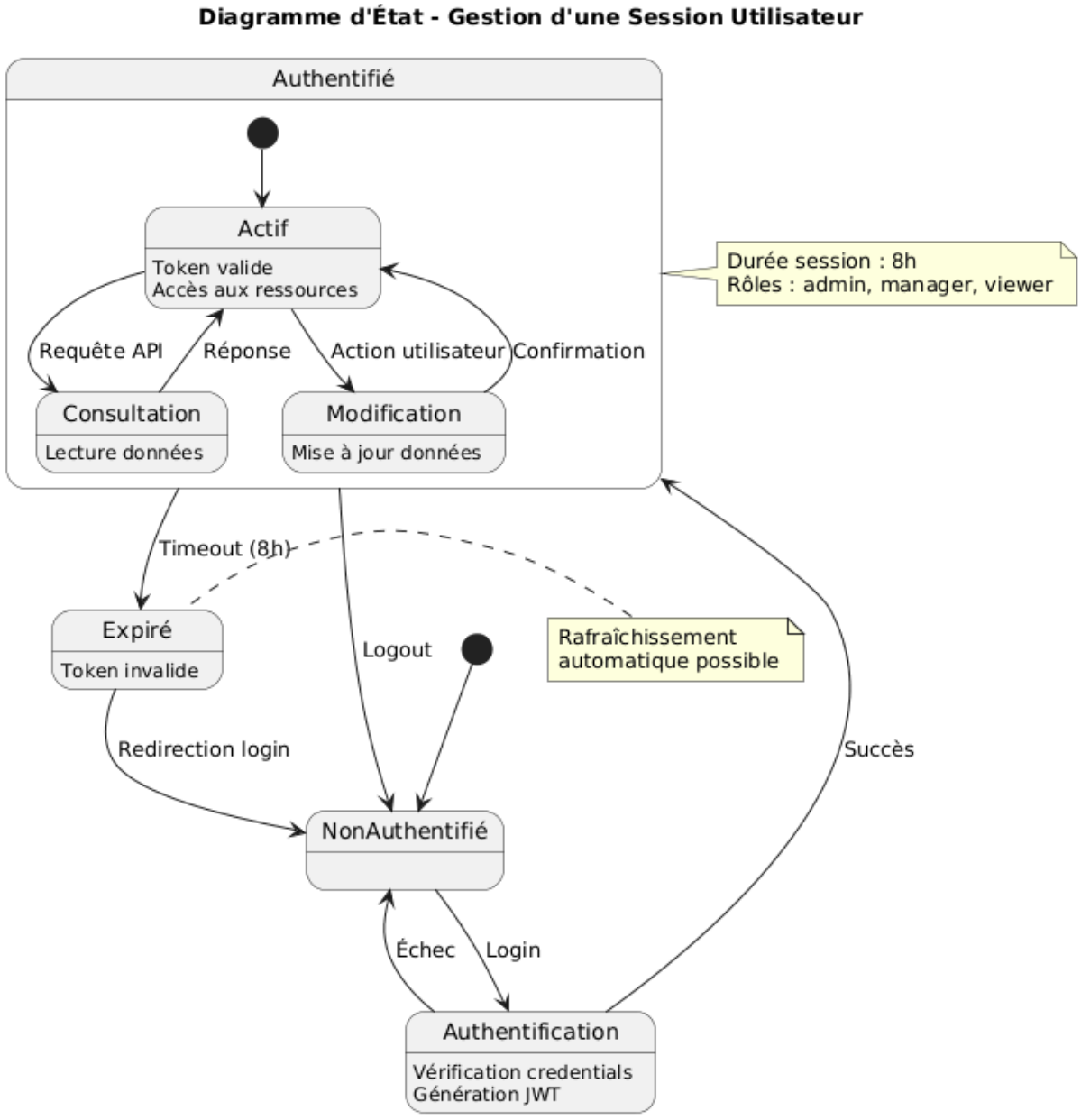


FIGURE 3.3 – Diagramme d'états – Gestion d'une Session Utilisateur

Diagramme d'activité : Consulter Dashboard

Ce diagramme illustre le processus de consultation du dashboard ERPConnect. Il montre les étapes depuis la connexion de l'utilisateur, la récupération des KPIs et graphiques, jusqu'à l'affichage interactif des données.

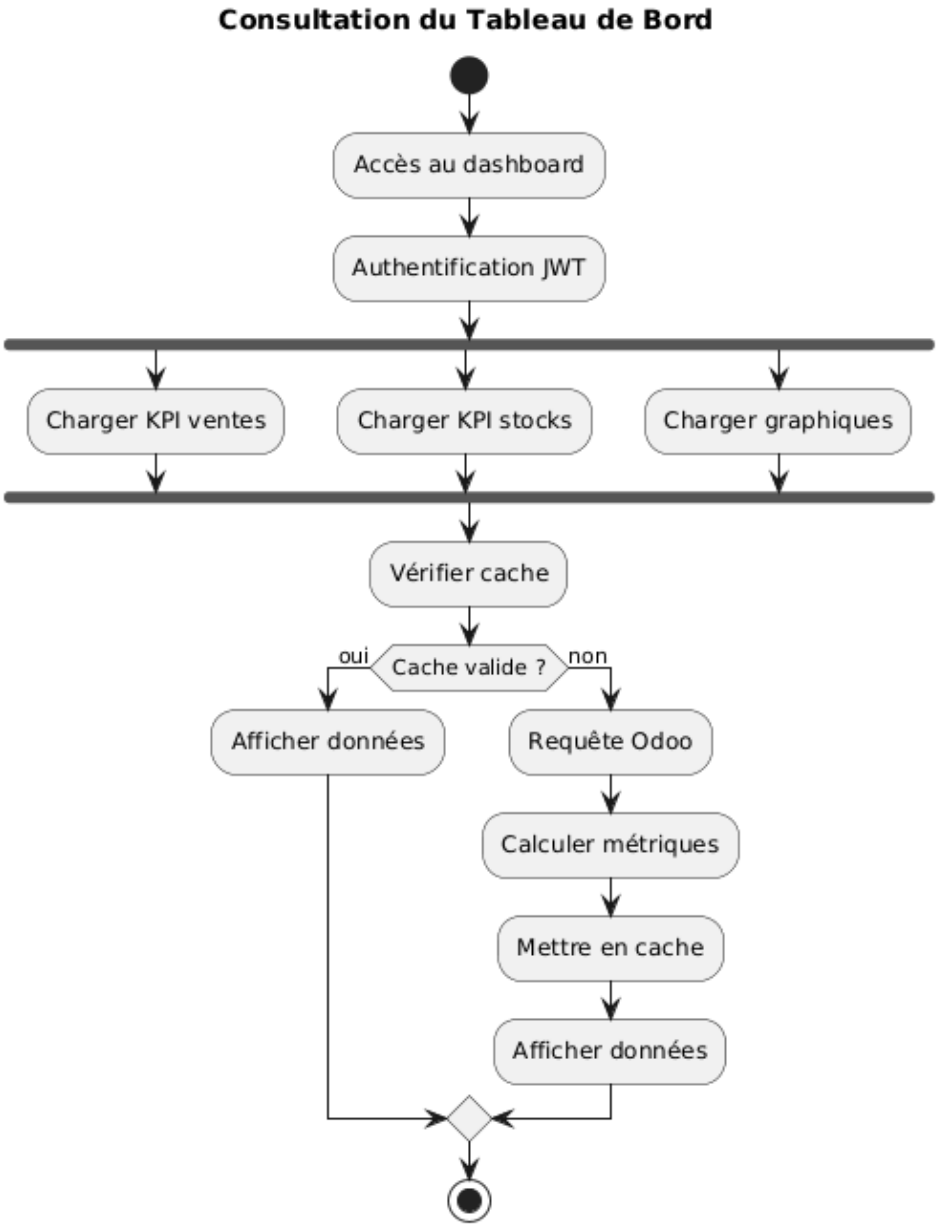


FIGURE 3.4 – Diagramme d'activité – View Dashboard

3.2.3 Diagrammes d'activité

Diagramme d'activité : Export CSV

Ce diagramme illustre le processus d'exportation des données depuis ERPConnect. Il décrit les étapes depuis la sélection des données, la génération du fichier CSV, jusqu'au téléchargement par l'utilisateur.

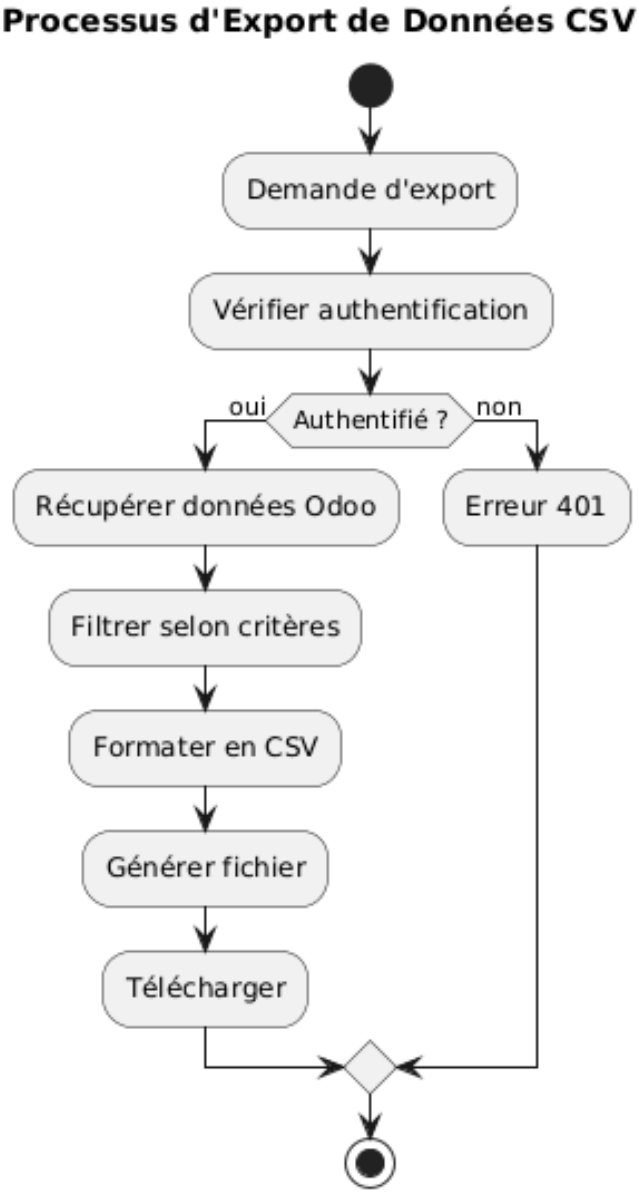


FIGURE 3.5 – Diagramme d'activité – Export CSV

3.3 Modélisation statique

La modélisation statique permet de représenter la structure interne du système ERPConnect ainsi que les relations entre ses différents composants.

3.3.1 Diagramme de classes

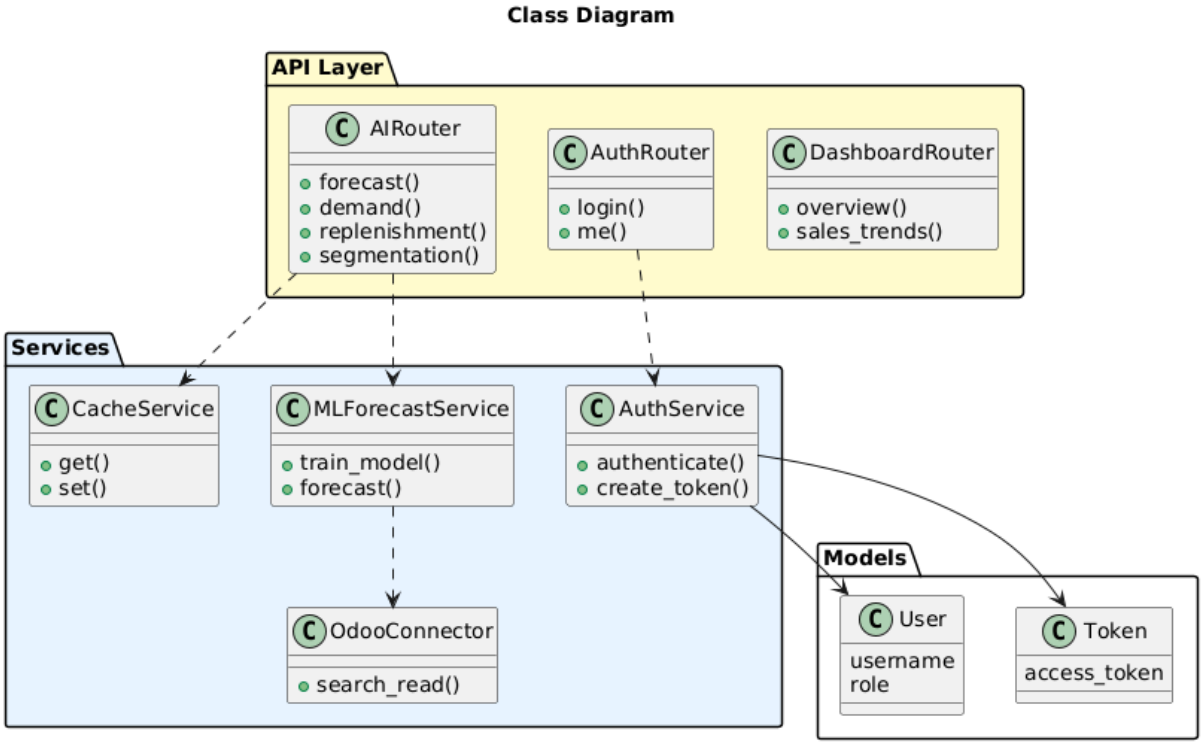


FIGURE 3.6 – Diagramme de classes du système ERPConnect

3.3.2 Modèle relationnel

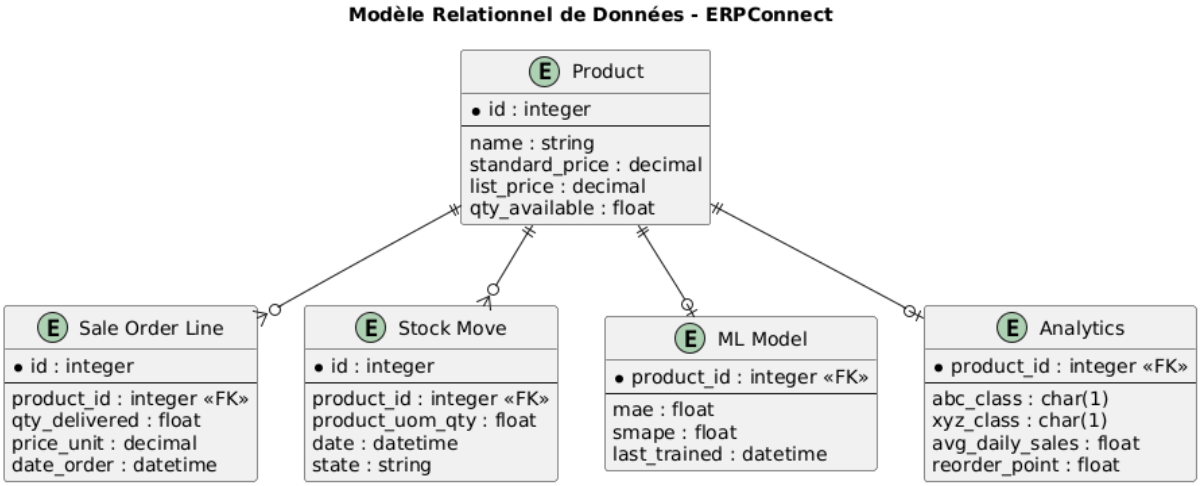


FIGURE 3.7 – Modèle relationnel des données

3.3.3 Dictionnaire de données

TABLE 3.1 – Dictionnaire de Données - Entités Principales

Champ	Type	Source	Description
Entité Product			
`id`	Integer	Odoo	Identifiant unique du produit
`name`	String	Odoo	Nom commercial du produit
`standard_price`	Decimal	Odoo	Prix de revient unitaire
`list_price`	Decimal	Odoo	Prix de vente public
`qty_available`	Float	Odoo	Stock physique disponible
Entité Sale Order Line			
`product_id`	Integer	Odoo	Référence au produit vendu
`qty_delivered`	Float	Odoo	Quantité livrée réelle
`price_unit`	Decimal	Odoo	Prix unitaire de vente
`date_order`	Datetime	Odoo	Date de la commande
Entité Stock Move			
`product_uom_qty`	Float	Odoo	Quantité du mouvement
`date`	Datetime	Odoo	Date du mouvement
`state`	String	Odoo	État (draft, done, cancel)

3.3.4 Architecture de l'application

Architecture logicielle

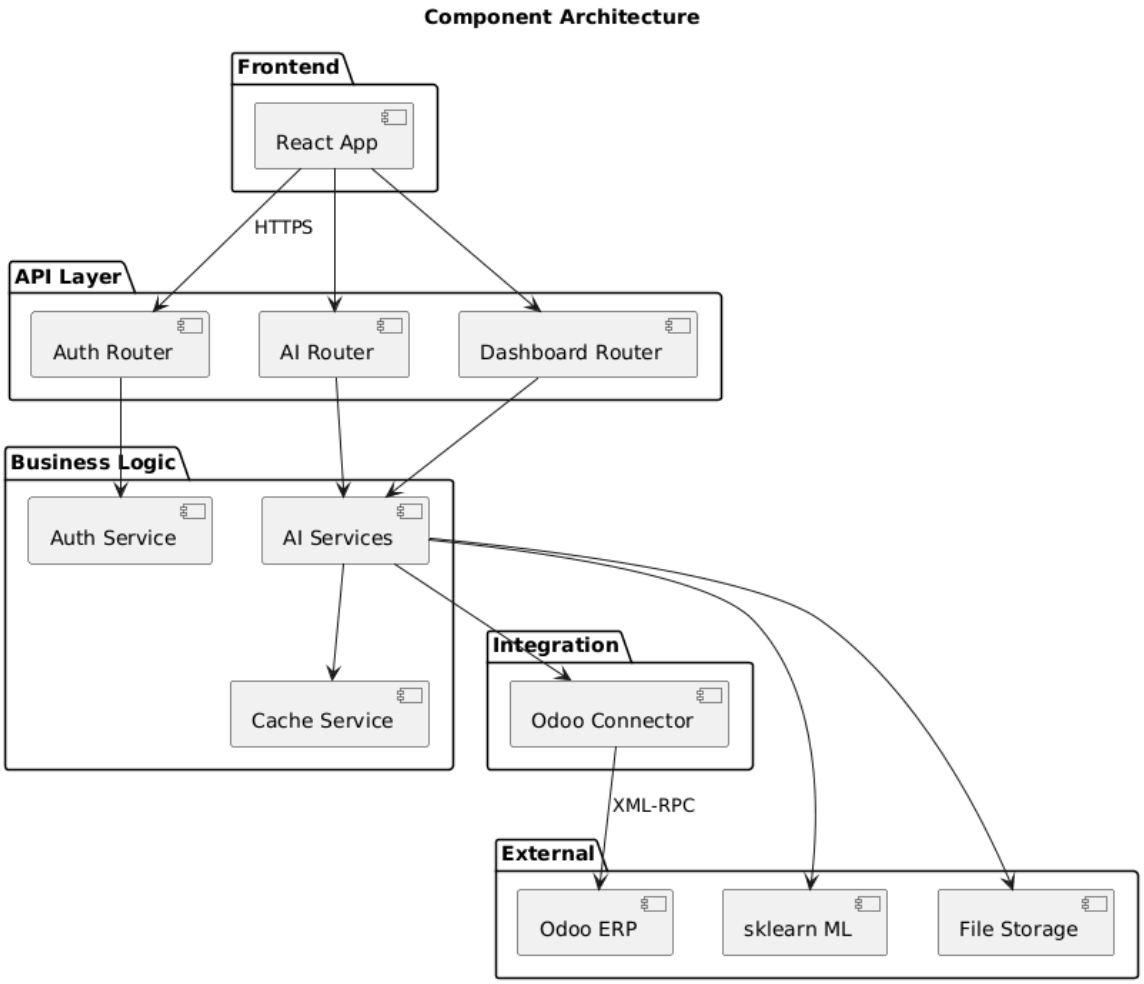


FIGURE 3.8 – Architecture logicielle de ERPConnect

Architecture matérielle

L'architecture matérielle repose sur un serveur applicatif hébergeant l'API FastAPI, connecté à une base de données et à l'ERP Odoo via une communication sécurisée.

3.4 Conclusion

Ce chapitre a présenté la conception complète du système ERPConnect à travers des modèles dynamiques et statiques. Ces modèles assurent une parfaite cohérence entre la conception théorique et l'implémentation réelle du système.

Le chapitre suivant sera consacré à la réalisation et à l'implémentation technique de la solution.

Chapitre 4

Réalisation

4.1 Introduction

Ce dernier chapitre détaille la phase d'implémentation d'ERPConnect. Nous présentons l'environnement de développement, les choix techniques concernant la pile logicielle, et une démonstration des principales interfaces utilisateurs qui concrétisent les exigences fonctionnelles et les capacités analytiques du système.

4.2 Environnement de Développement

Le choix de l'environnement de développement a été guidé par le besoin de haute performance, de modularité et de prototypage rapide des modèles de machine learning.

4.2.1 Environnement Matériel

Le développement et les tests ont été réalisés sur une station de travail avec les spécifications techniques suivantes :

— **Processeur :** Intel Core i7 (11ème génération) avec 8 cœurs ;
— **RAM :** 16 Go DDR4 ;
— **Stockage :** SSD NVMe 512 Go ;

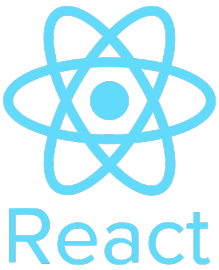
4.2.2 Environnement Logiciel

Une pile logicielle moderne et robuste a été sélectionnée pour gérer le traitement de données en temps réel et les visualisations interactives :

— **Système d'exploitation :** Windows 11 avec WSL2 (Windows Subsystem for Linux)

— **Framework Backend : FastAPI** pour sa haute performance et sa documentation automatique

— **Bibliothèque Frontend : React** pour construire une interface utilisateur réactive et modulaire



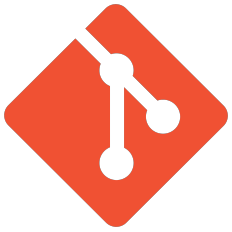
— **Stack IA : scikit-learn** pour la modélisation prédictive et **pandas** pour la manipulation de données

— **Intégration de Données : Odoo ERP API** via la bibliothèque XML-RPC pour une communication fluide

— **Persistance : joblib** pour stocker les modèles ML entraînés sur le disque

— **Outils Avancés :**
  — **Visual Studio Code :** Éditeur de code principal

— **Git :** Système de contrôle de version



— **Postman :** Outil de test et documentation d'API

4.3 Principales Interfaces Graphiques

Cette section présente les interfaces utilisateur clés d'ERPConnect, conçues pour offrir une expérience moderne et intuitive aux décideurs. Toutes les interfaces suivent une charte graphique cohérente en mode sombre pour réduire la fatigue visuelle lors de sessions prolongées.

4.3.1 Interface de Connexion

L'interface de connexion constitue le point d'entrée sécurisé de l'application. Elle implémente le protocole OAuth2 avec authentification JWT, permettant aux utilisateurs de s'identifier selon leur rôle (administrateur, manager ou visualiseur). Le design minimaliste met l'accent sur la sécurité tout en offrant une expérience utilisateur fluide avec validation en temps réel des champs de saisie.

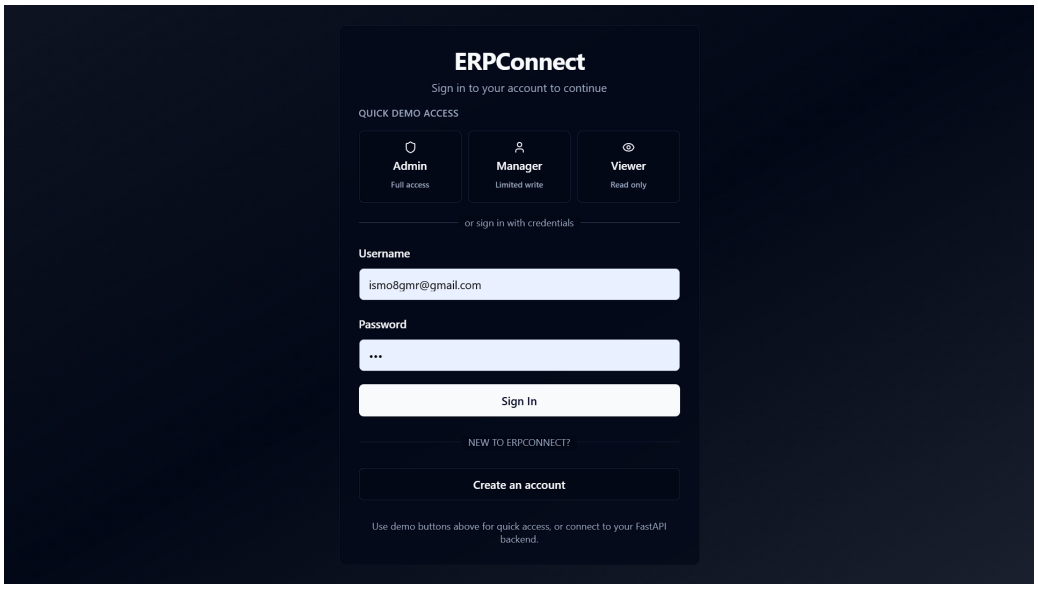


FIGURE 4.1 – Interface de Connexion - Authentification JWT

4.3.2 Tableau de Bord Principal

Le tableau de bord principal représente le cœur de l'application, offrant une vue d'ensemble complète des performances de l'entreprise. Il affiche plus de 30 indicateurs clés de performance (KPI) organisés en cartes modulaires : revenus, croissance, marges, état des stocks, et métriques d'efficacité. Les graphiques interactifs permettent de visualiser les tendances sur différentes périodes (7, 30, 90 jours). Un système de rafraîchissement automatique garantit que les données affichées sont toujours à jour.

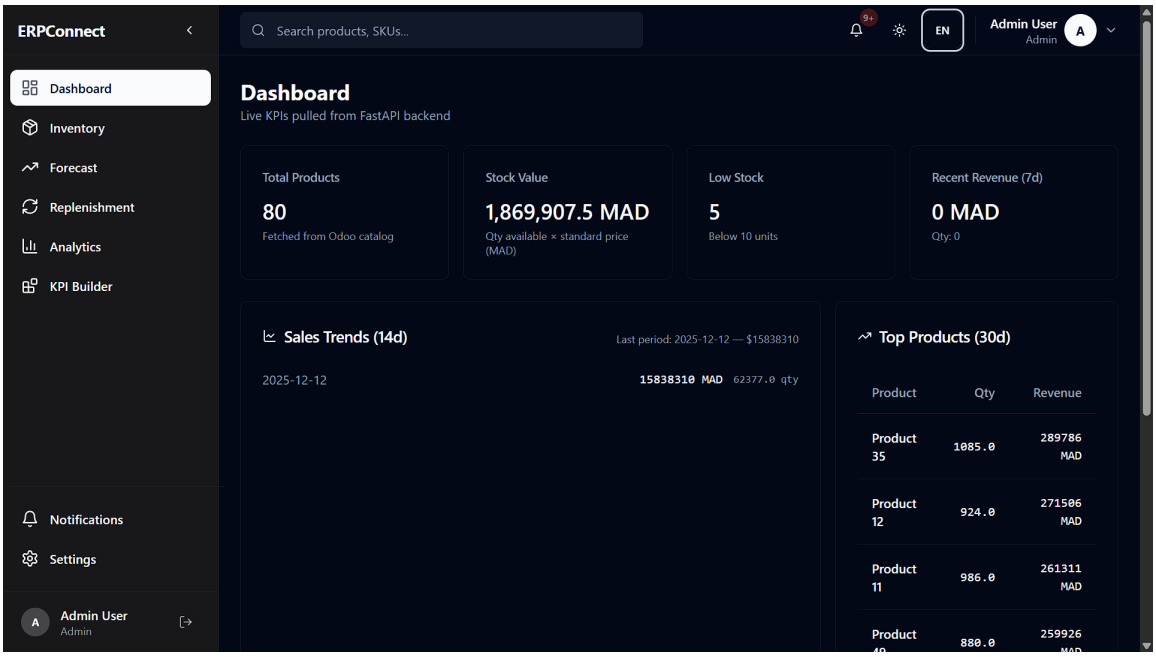


FIGURE 4.2 – Tableau de Bord Principal - Vue d'Ensemble des KPI

4.3.3 Module de Prévision de la Demande et Réapprovisionnement Intelligent

Ces interfaces permettent d'analyser les prévisions de vente et de générer des recommandations de réapprovisionnement. Les prévisions ML affichent l'historique des ventes (en bleu) et la prévision à 30 jours (en orange) avec intervalles de confiance à 95%. Le réapprovisionnement intelligent indique le stock actuel, le point de commande (ROP), la quantité suggérée et le niveau d'urgence codé par couleur.

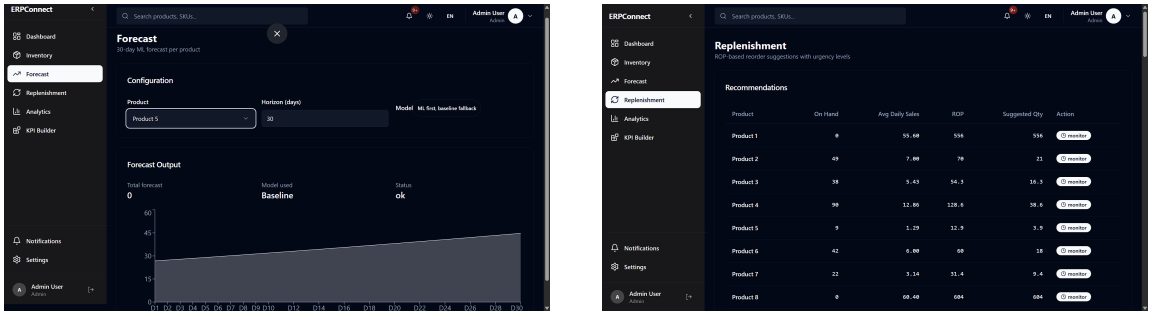


FIGURE 4.3 – Prévision de la Demande (ML) et Réapprovisionnement Intelligent (ROP)

4.3.4 Matrice de Segmentation ABC/XYZ

Cette interface visualise la segmentation automatique du catalogue produit selon deux dimensions : la contribution au chiffre d'affaires (classification ABC) et la variabilité de la demande (classification XYZ). La matrice 3×3 affiche le nombre de produits dans chaque segment avec des recommandations stratégiques associées.

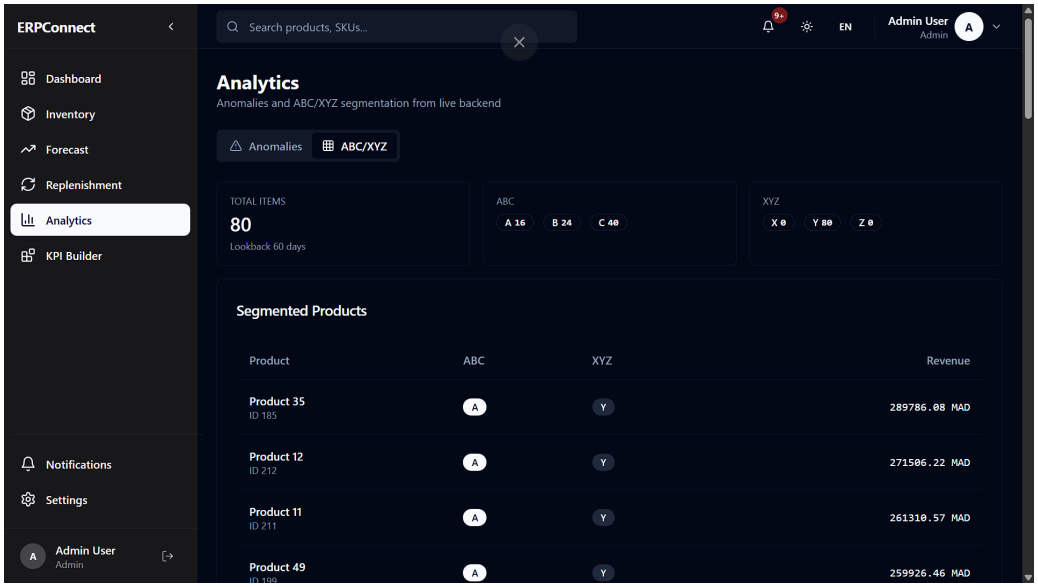


FIGURE 4.4 – Matrice de Segmentation ABC/XYZ - Classification Intelligente du Catalogue

Note : Les autres interfaces secondaires (gestion des utilisateurs, configuration des paramètres, exports de données, logs système) sont présentées en annexe pour ne pas surcharger ce chapitre. acilitent la priorisation des actions par les gestionnaires de stocks.

En conclusion, ERPConnect démontre qu'il est possible d'intégrer des capacités d'intelligence artificielle avancées dans des systèmes ERP existants de manière pragmatique et accessible, même pour des organisations de taille moyenne. Le projet ouvre des perspectives intéressantes pour l'évolution future, notamment l'intégration d'algorithmes de Deep Learning pour la gestion de saisonnalités complexes, l'extension à d'autres ERP (SAP, Microsoft Dynamics), et l'ajout de fonctionnalités de planification collaborative avec les fournisseurs.

Conclusion Générale

Le développement d'**ERPConnect** répond à un besoin critique dans la gestion moderne des entreprises : la capacité à exploiter les données ERP existantes pour une planification stratégique proactive. Tout au long de ce projet, nous avons intégré avec succès des techniques avancées d'intelligence artificielle dans une architecture modulaire capable de fonctionner avec des systèmes professionnels comme Odoo.

Nous avons atteint plusieurs objectifs clés :

— Création d'un backend microservices performant avec FastAPI ;

— Mise en place d'un pipeline ML fiable pour la prévision de la demande ;

— Conception d'une interface modulaire pour le suivi business via les KPI ;

— Développement d'un moteur automatisé d'optimisation des stocks.

Ce projet a démontré que des modules intelligents peuvent améliorer significativement les systèmes ERP traditionnels sans nécessiter de modifications structurelles lourdes. Les perspectives futures pour ERPConnect incluent l'intégration d'algorithmes plus complexes comme le Deep Learning (LSTM) pour les motifs saisonniers très long terme et l'extension du système de connecteurs pour prendre en charge d'autres ERP comme SAP ou Microsoft Dynamics.

Bibliographie et Webographie

Livres et Articles Académiques

— **Montgomery, D. C. (2015) :** *Introduction à la Contrôle Statistique de la Qualité*, Wiley. (Utilisé pour la logique du Z-score et détection d'anomalies).

— **Silver, E. A. (2016) :** *Gestion des Stocks et Planification de Production*, Wiley. (Utilisé pour les formules ROP et stock de sécurité).

Documentation Technique

— **Documentation FastAPI :** https://fastapi.tiangolo.com/

— **Documentation React :** https://reactjs.org/

— **API Web Odoo :** https://www.odoo.com/documentation/14.0/developer/api/external_api.html

— **Guide Utilisateur scikit-learn :** https://scikit-learn.org/stable/user_guide.html