Université Abdelmalek Essaâdi
École Nationale des Sciences Appliquées de Tanger

Année Universitaire 2025-2026

Smart Sales Notifier

Module Odoo avec Intelligence Artificielle

Intégration ERP-IA via n8n

Analyse Automatisée des Commandes avec Groq LLaMA

Réalisé par :

Dahbi Moad
Bouker Mohamed
Allam Elarbi

Encadré par :

Pr. Hassan BADIR

Janvier 2026
Version 1.0.0
Odoo 17.0 + n8n + Groq

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte du Projet

Dans le cadre de notre formation à l'École Nationale des Sciences Appliquées de Tanger, nous avons été amenés à développer un module ERP intégrant des technologies d'intelligence artificielle. Ce projet vise à démontrer la capacité d'intégration entre un système ERP moderne (Odoo) et des outils d'automatisation et d'IA.

L'objectif est de créer un système intelligent capable d'analyser automatiquement les commandes de vente et de générer des notifications enrichies par l'intelligence artificielle.

Ressources du Projet

Code Source (GitHub) :
https://github.com/dahbimoad/smart_sales_notifier_odoo.git

Vidéo Démonstration (Google Drive) :
https://drive.google.com/drive/folders/1VhpLWys0xJnrltJVffv65LXgDvlXMtEo?usp=sharing

1.2 Objectifs

Les objectifs principaux de ce projet sont :

- Développer un module Odoo personnalisé pour la gestion intelligente des ventes
- Intégrer l'intelligence artificielle (Groq LLaMA 3.1) pour l'analyse des commandes
- Automatiser les notifications via n8n et Telegram
- Créer une architecture scalable et maintenable

1.3 Technologies Utilisées

Technologie	Version	Utilisation
Odoo	17.0	ERP principal, gestion des ventes
PostgreSQL	15	Base de données
n8n	1.113.3	Orchestration et automatisation des workflows
Groq	API	Intelligence artificielle (LLaMA 3.1 8B)
Telegram Bot	API	Notifications en temps réel
Docker	Latest	Conteneurisation et déploiement
Python	3.10	Développement du module Odoo

TABLE 1.1 – Stack technologique du projet

1.4 Architecture Globale

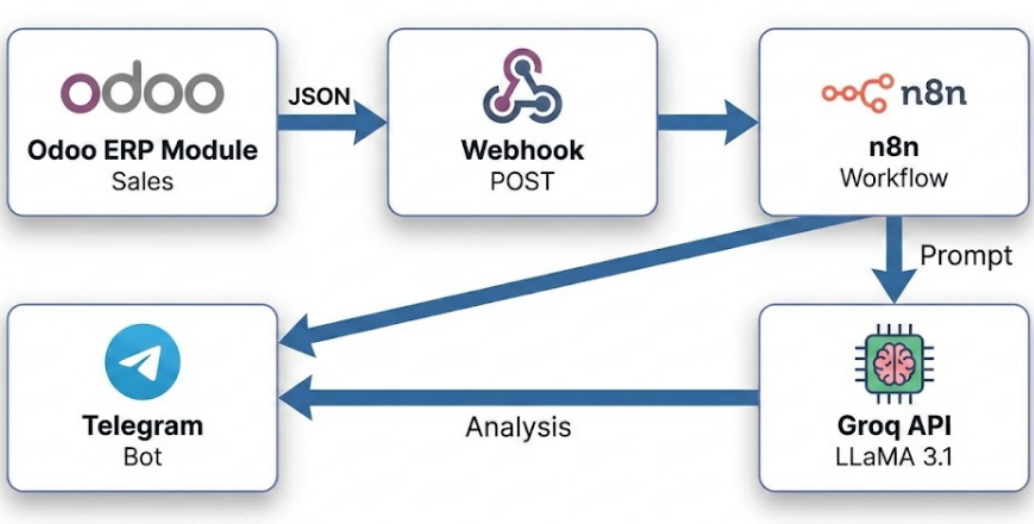

FIGURE 1.1 – Architecture du système Smart Sales Notifier

Chapitre 2

Problématique et Cahier des Charges

2.1 Problématique

Dans le contexte actuel des entreprises, la gestion des ventes représente un enjeu majeur. Les équipes commerciales font face à plusieurs défis :

- **Volume de commandes** - Difficulté à traiter rapidement un grand nombre de commandes
- **Priorisation manuelle** - Absence d'outils pour identifier automatiquement les commandes prioritaires
- **Temps de réaction** - Délai entre la réception d'une commande et sa prise en charge
- **Analyse limitée** - Manque d'insights intelligents sur les commandes reçues
- **Communication** - Notifications tardives ou inexistantes pour les équipes

Question Centrale

Comment automatiser l'analyse des commandes de vente et fournir des notifications intelligentes en temps réel aux équipes commerciales ?

2.2 Cahier des Charges

2.2.1 Objectifs Fonctionnels

ID	Fonctionnalité	Description
F01	Détection automatique	Détecter les nouvelles commandes confirmées
F02	Envoi webhook	Transmettre les données de commande à n8n
F03	Analyse IA	Analyser la commande via Groq LLaMA
F04	Priorisation	Attribuer un niveau de priorité (low, medium, high, urgent)
F05	Notification Telegram	Envoyer une alerte formatée sur Telegram
F06	Stockage analyse	Sauvegarder l'analyse IA dans Odoo
F07	Interface utilisateur	Afficher les résultats dans un onglet dédié
F08	Configuration	Permettre la configuration du webhook et des seuils

TABLE 2.1 – Exigences fonctionnelles du module

2.2.2 Objectifs Non-Fonctionnels

ID	Critère	Exigence
NF01	Performance	Temps de réponse < 5 secondes
NF02	Disponibilité	Module fonctionnel 24/7
NF03	Compatibilité	Compatible Odoo 17.0
NF04	Sécurité	Communication HTTPS sécurisée
NF05	Maintenabilité	Code documenté et structuré
NF06	Scalabilité	Support de multiples commandes simultanées

TABLE 2.2 – Exigences non-fonctionnelles

2.2.3 Acteurs du Système

- **Commercial** - Crée et confirme les commandes de vente
- **Manager** - Reçoit les notifications et supervise les priorités
- **Administrateur** - Configure le module et les paramètres
- **Système IA** - Analyse automatiquement les commandes

2.2.4 Cas d'Utilisation Principal

1. Le commercial crée une commande de vente dans Odoo
2. Le commercial confirme la commande

3. Le système détecte la confirmation et envoie les données au webhook
4. n8n reçoit les données et les transmet à Groq pour analyse
5. L'IA génère une analyse avec priorité et recommandations
6. Une notification est envoyée sur Telegram
7. L'analyse est sauvegardée et affichée dans Odoo

2.2.5 Livrables Attendus

— Module Odoo `smart_sales_notifier`
— Workflow n8n configuré
— Bot Telegram fonctionnel
— Documentation technique
— Rapport de projet
— Vidéo de démonstration

Chapitre 3

Installation et Configuration d'Odoo

3.1 Prérequis

Pour ce projet, nous avons utilisé Docker pour déployer Odoo, ce qui simplifie considérablement l'installation et la gestion de l'environnement. Les prérequis sont :

— Docker Desktop installé sur Windows

— Connexion Internet pour télécharger les images

— Minimum 4GB de RAM disponible

3.2 Déploiement avec Docker Compose

3.2.1 Fichier docker-compose.yml

Nous avons créé un fichier `docker-compose.yml` pour orchestrer les conteneurs Odoo et PostgreSQL :

```
version: '3.8'
services:
  odoo:
    image: odoo:17.0
    container_name: odoo
    depends_on:
      - db
    ports:
      - "8069:8069"
    volumes:
      - odoo-data:/var/lib/odoo
      - ./smart_sales_notifier:/mnt/extra-addons/
   smart_sales_notifier
    environment:
      - HOST=db
      - USER=odoo
      - PASSWORD=odoo

  db:
    image: postgres:15
    container_name: odoo-db
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=odoo
      - POSTGRES_PASSWORD=odoo
```

```
    volumes:
      - postgres-data:/var/lib/postgresql/data

volumes:
  odoo-data:
  postgres-data:
```

Listing 3.1 – Configuration Docker Compose pour Odoo

3.2.2 Lancement des Conteneurs

Pour démarrer l'environnement Odoo :

```
# Se placer dans le repertoire du projet
cd C:\Users\Dahbi\Desktop\ODOO

# Lancer les conteneurs
docker-compose up -d

# Verifier que les conteneurs sont actifs
docker ps
```

Listing 3.2 – Commandes de lancement Docker

3.3 Création de la Base de Données

Lors du premier accès à Odoo via `http://localhost:8069`, l'interface de création de base de données s'affiche.

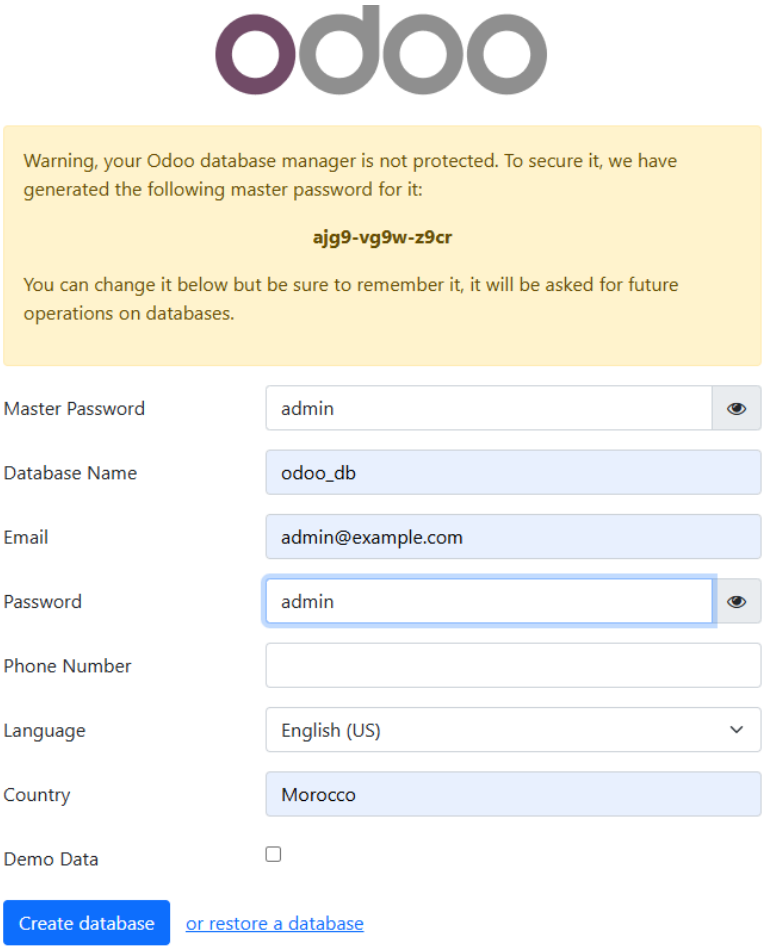

FIGURE 3.1 – Écran de création de la base de données Odoo

Les paramètres configurés :

— **Master Password** : Mot de passe administrateur pour la gestion des BDD
— **Database Name** : `odoo_db`
— **Email** : Identifiant de l'administrateur
— **Password** : Mot de passe de connexion
— **Language** : Français
— **Country** : Maroc

3.4 Connexion à Odoo

Après la création de la base de données, l'écran de connexion permet d'accéder au système.

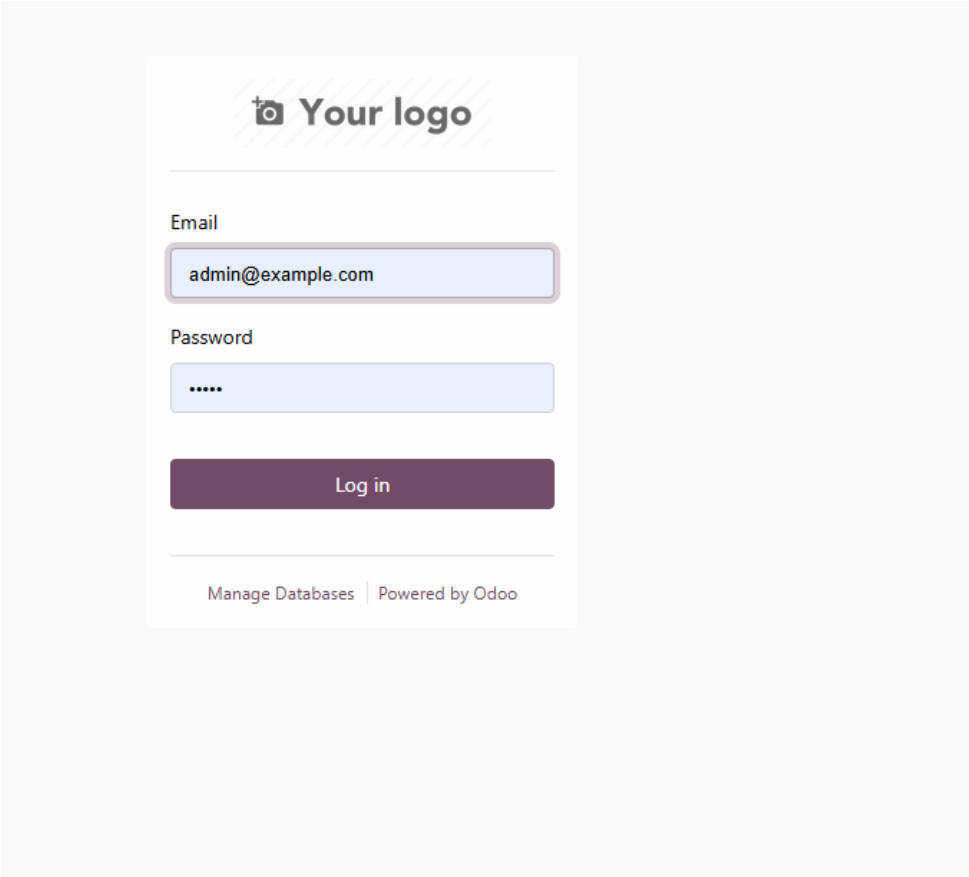

FIGURE 3.2 – Page de connexion Odoo

3.5 Activation du Mode Développeur

Le mode développeur est essentiel pour installer des modules personnalisés et accéder aux fonctionnalités avancées.

3.5.1 Procédure d'Activation

1. Se connecter avec un compte administrateur
2. Aller dans **Paramètres** (Settings)
3. Défiler jusqu'à la section **Developer Tools**
4. Cliquer sur **Activate the developer mode**

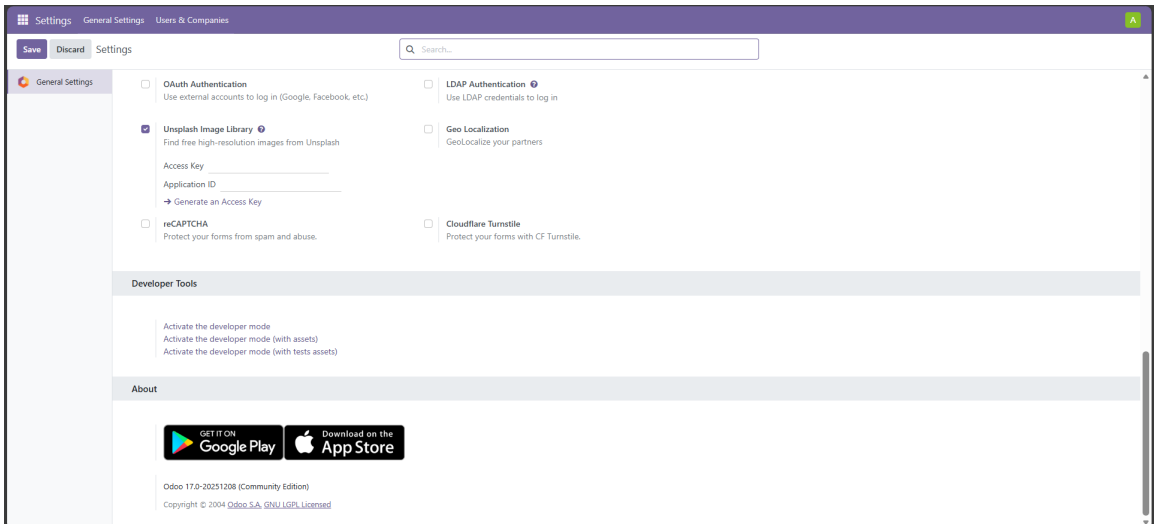

FIGURE 3.3 – Activation du mode développeur dans les paramètres

3.5.2 Fonctionnalités du Mode Développeur

Fonctionnalité	**Description**
Métadonnées techniques	Affiche les IDs, noms techniques des champs
Éditeur de vues	Permet de modifier les vues XML directement
Menu technique	Accès aux modèles, vues, actions
Mise à jour des apps	Permet de mettre à jour la liste des modules
Debug SQL	Affiche les requêtes SQL exécutées

TABLE 3.1 – Fonctionnalités accessibles en mode développeur

Chapitre 4

Développement du Module Smart Sales Notifier

4.1 Structure du Module

Un module Odoo suit une structure standardisée. Voici l'arborescence de notre module :

```
smart_sales_notifier/
|-- __init__.py
|-- __manifest__.py
|-- models/
|   |-- __init__.py
|   |-- sale_order.py
|   |-- res_config_settings.py
|-- views/
|   |-- sale_order_views.xml
|   |-- res_config_settings_views.xml
|-- data/
|   |-- ir_cron.xml
|   |-- demo_data.xml
|-- security/
|   |-- ir.model.access.csv
|-- static/
    |-- description/
        |-- icon.svg
```

Listing 4.1 – Structure du module Smart Sales Notifier

4.2 Fichier Manifest (__manifest__.py)

Le fichier manifest définit les métadonnées du module :

```
# -*- coding: utf-8 -*-
{
    'name': 'Smart Sales Notifier',
    'version': '1.0.0',
    'category': 'Sales',
    'summary': 'AI-Powered Sales Notifications via n8n
   Integration',
    'description': """
        Smart Sales Notifier - Module ERP avec Intelligence
   Artificielle
```

```
    ================================================================


        Ce module permet de:
        - Detecter automatiquement les nouvelles commandes
        - Envoyer les donnees a n8n pour traitement IA (Groq/LLaMA)
        - Generer des notifications intelligentes via Telegram
        - Analyser les ventes en temps reel avec l'IA
        - Prioriser les commandes automatiquement

        Developpe par: Moad Dahbi
        Date: Janvier 2026
        Projet: ERP - ENSA Tanger
    """,
    'author': 'Moad Dahbi',
    'website': 'https://n8n.dahbimoad.com',
    'license': 'LGPL-3',
    'depends': ['sale', 'mail', 'product'],
    'data': [
        'security/ir.model.access.csv',
        'data/demo_data.xml',
        'views/res_config_settings_views.xml',
        'views/sale_order_views.xml',
        'data/ir_cron.xml',
    ],
    'installable': True,
    'application': True,
    'auto_install': False,
}
```

Listing 4.2 – __manifest__.py - Définition du module

4.3 Modèle Sale Order Étendu

4.3.1 Ajout des Champs IA

```
# -*- coding: utf-8 -*-
import json
import logging
import requests
from odoo import models, fields, api
from odoo.exceptions import UserError

_logger = logging.getLogger(__name__)


class SaleOrder(models.Model):
    _inherit = 'sale.order'
```

```
13
14      # Champs pour le suivi des notifications IA
15      smart_notified = fields.Boolean(
16          string='AI Notified',
17          default=False,
18          help='Indicates if this order has been sent for AI
    analysis'
19      )
20
21      smart_notification_date = fields.Datetime(
22          string='AI Analysis Date',
23          readonly=True,
24          help='Date when the order was sent for AI analysis'
25      )
26
27      ai_analysis = fields.Text(
28          string='AI Analysis',
29          readonly=True,
30          help='AI-generated analysis of this order'
31      )
32
33      priority_score = fields.Selection([
34          ('low', 'Low'),
35          ('medium', 'Medium'),
36          ('high', 'High'),
37          ('urgent', 'Urgent')
38      ], string='AI Priority', readonly=True)
```

Listing 4.3 – sale_order.py - Extension du modèle de commande

4.3.2 Méthode d'Envoi au Webhook

```
1       def action_send_smart_notification(self):
2           """Send order data to n8n webhook for AI processing"""
3           self.ensure_one()
4
5           # Get webhook URL from settings
6           webhook_url = self.env['ir.config_parameter'].sudo().
    get_param(
7               'smart_sales_notifier.n8n_webhook_url'
8           )
9
10          if not webhook_url:
11              raise UserError(
12                  'Webhook URL not configured. '
13                  'Go to Settings > Smart Sales Notifier.'
14              )
15
16          # Prepare order data
17          order_data = {
18              'event_type': 'manual_notification',
```

```
            'order_id': self.id,
            'order_name': self.name,
            'customer_name': self.partner_id.name,
            'customer_email': self.partner_id.email or '',
            'customer_phone': self.partner_id.phone or '',
            'total_amount': self.amount_total,
            'currency': self.currency_id.name,
            'date_order': self.date_order.isoformat()
                if self.date_order else '',
            'salesperson': self.user_id.name if self.user_id
    else '',
            'products': [{
                'product_name': line.product_id.name,
                'quantity': line.product_uom_qty,
                'unit_price': line.price_unit,
                'subtotal': line.price_subtotal
            } for line in self.order_line]
        }

        try:
            # Send to n8n webhook
            response = requests.post(
                webhook_url,
                json=order_data,
                headers={'Content-Type': 'application/json'},
                timeout=30
            )
            response.raise_for_status()

            # Update order with AI response
            result = response.json()
            self.write({
                'smart_notified': True,
                'smart_notification_date': fields.Datetime.now()
    ,
                'ai_analysis': result.get('ai_analysis', ''),
                'priority_score': result.get('priority', 'medium
    ')
            })

            return {
                'type': 'ir.actions.client',
                'tag': 'display_notification',
                'params': {
                    'title': 'Smart Notification',
                    'message': 'AI analysis completed
    successfully!',
                    'type': 'success',
                    'sticky': False,
                }
            }
```

```
66
67          except Exception as e:
68              _logger.error(f'Smart notification failed: {str(e)}'
    )
69              raise UserError(f'Failed to send notification: {str(
    e)}')
```

Listing 4.4 – Méthode d'envoi des données à n8n

4.4 Configuration Settings

```
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields, api
3
4
5  class ResConfigSettings(models.TransientModel):
6      _inherit = 'res.config.settings'
7
8      n8n_webhook_url = fields.Char(
9          string='n8n Webhook URL',
10         help='The n8n webhook URL for receiving sales
    notifications',
11         config_parameter='smart_sales_notifier.n8n_webhook_url'
12     )
13
14     high_value_threshold = fields.Float(
15         string='High Value Order Threshold',
16         help='Orders above this amount trigger special
    notifications',
17         config_parameter='smart_sales_notifier.
    high_value_threshold',
18         default=1000.0
19     )
20
21     enable_auto_notify = fields.Boolean(
22         string='Enable Auto Notifications',
23         help='Automatically send notifications when orders are
    confirmed',
24         config_parameter='smart_sales_notifier.
    enable_auto_notify',
25         default=True
26     )
```

Listing 4.5 – res_config_settings.py - Paramètres du module

4.5 Vues XML

4.5.1 Vue du Formulaire de Commande

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <record id="sale_order_form_inherit_smart_notifier" model="ir.ui.view">
        <field name="name">sale.order.form.inherit.smart.notifier</field>
        <field name="model">sale.order</field>
        <field name="inherit_id" ref="sale.view_order_form"/>
        <field name="arch" type="xml">
            <!-- Add AI Analysis button -->
            <xpath expr="//header" position="inside">
                <button name="action_send_smart_notification"
                        string="AI Analysis"
                        type="object"
                        class="btn-primary"
                        icon="fa-magic"
                        groups="sales_team.group_sale_salesman"
                        invisible="state not in ('sale', 'done')"/>
            </xpath>

            <!-- Add AI Analysis tab -->
            <xpath expr="//notebook" position="inside">
                <page string="AI Analysis" name="smart_notifier">
                    <group invisible="smart_notified == False">
                        <group string="Analysis Status">
                            <field name="smart_notified"
                                   widget="boolean_toggle" readonly="1"/>
                            <field name="smart_notification_date" readonly="1"/>
                            <field name="priority_score" readonly="1"/>
                        </group>
                        <group string="Order Summary">
                            <field name="amount_total" string="Total Amount"/>
                            <field name="partner_id" string="Customer"/>
                        </group>
                    </group>
                    <group string="AI-Generated Insights"
                           invisible="smart_notified == False">
                        <field name="ai_analysis" readonly="1"
                               nolabel="1" colspan="2"/>
                    </group>
                </page>
            </xpath>
        </field>
    </record>
```

```
43 </odoo>
```

Listing 4.6 – sale_order_views.xml - Extension de la vue commande

4.6 Sécurité et Permissions

Listing 4.7 – ir.model.access.csv - Droits d'accès

id , name , model_id : id , group_id : id , perm_read , perm_write , perm_create , perm_unlin
access_sale_order_smart , sale . order . smart , sale . model_sale_order , sales_team . g

4.7 Installation du Module

4.7.1 Accès au Menu Applications

Après avoir activé le mode développeur, nous accédons au menu Applications pour installer notre module personnalisé.

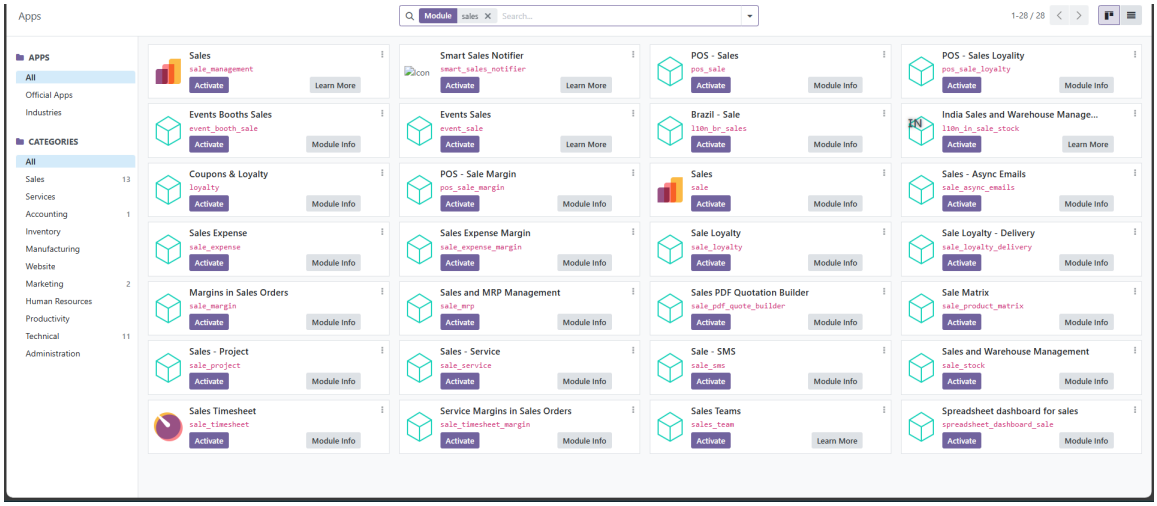

FIGURE 4.1 – Dashboard des applications Odoo disponibles

4.7.2 Recherche et Installation

Dans le menu Apps, nous recherchons notre module "Smart Sales Notifier" :

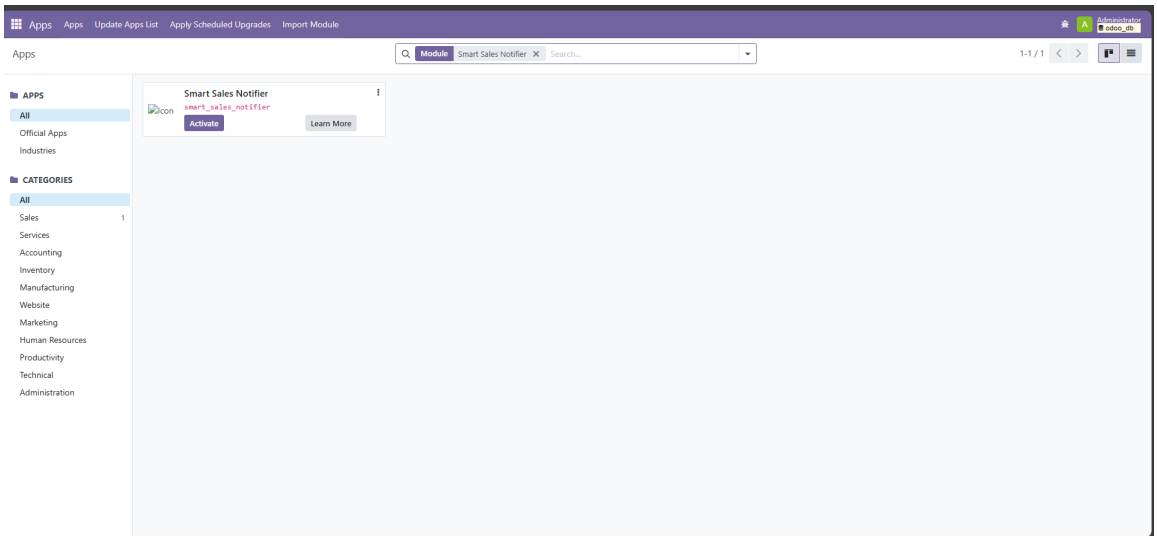

FIGURE 4.2 – Module Smart Sales Notifier dans la liste des applications

1. Cliquer sur **Apps** dans le menu principal
2. Cliquer sur **Mettre à jour la liste des applications** (en mode développeur)
3. Rechercher **Smart Sales Notifier**
4. Cliquer sur **Activer** pour installer le module

Chapitre 5

Configuration du Module

5.1 Paramètres du Smart Sales Notifier

Après l'installation, le module ajoute une section de configuration dans les paramètres d'Odoo.

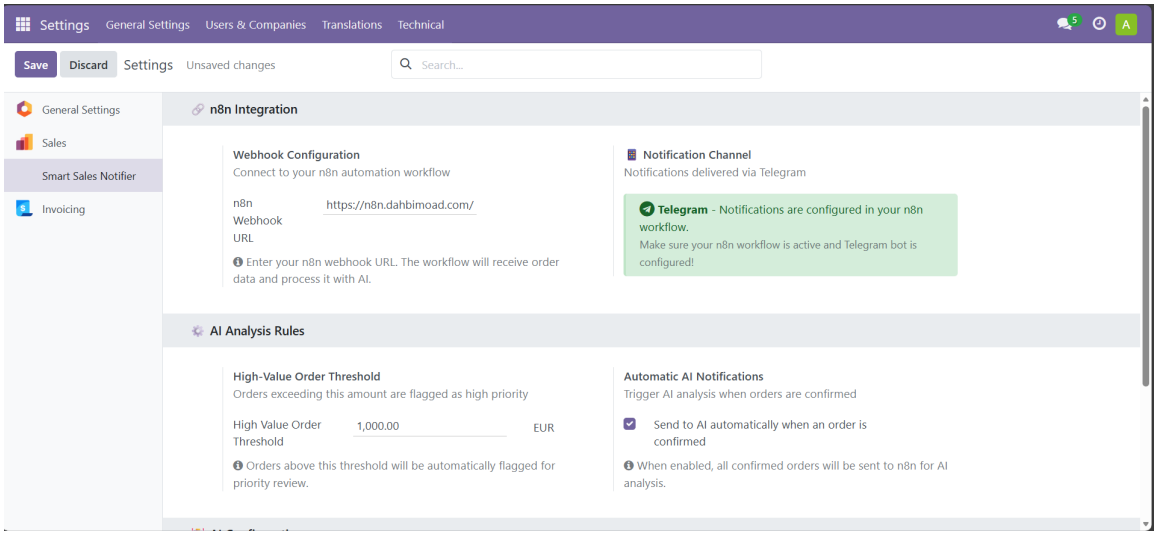

FIGURE 5.1 – Page de configuration du Smart Sales Notifier

5.1.1 Paramètres Disponibles

Paramètre	Description
n8n Webhook URL	URL du webhook n8n pour recevoir les données
Telegram Notifications	Indicateur que les notifications passent par Telegram
High Value Threshold	Seuil pour identifier les commandes à haute valeur
Auto Notifications	Active l'envoi automatique lors de la confirmation

TABLE 5.1 – Paramètres de configuration du module

5.1.2 Configuration du Webhook

L'URL du webhook doit pointer vers votre instance n8n :

`https://n8n.dahbimoad.com/webhook/sales-notifier`

Chapitre 6

Configuration du Workflow n8n

6.1 Présentation de n8n

n8n est une plateforme d'automatisation de workflows open-source qui permet de connecter différentes applications et services. Dans notre projet, n8n joue le rôle d'orchestrateur entre Odoo et l'API Groq.

6.2 Architecture du Workflow

Le workflow se compose de 5 nœuds principaux :

#	**Nœud**	**Fonction**
1	Webhook	Réception des données d'Odoo
2	Prepare Request	Préparation de la requête pour Groq
3	Groq AI	Appel à l'API Groq pour l'analyse IA
4	Process Response	Traitement et formatage de la réponse
5	Telegram	Envoi de la notification
6	Respond to Odoo	Retour de la réponse à Odoo

TABLE 6.1 – Nœuds du workflow n8n

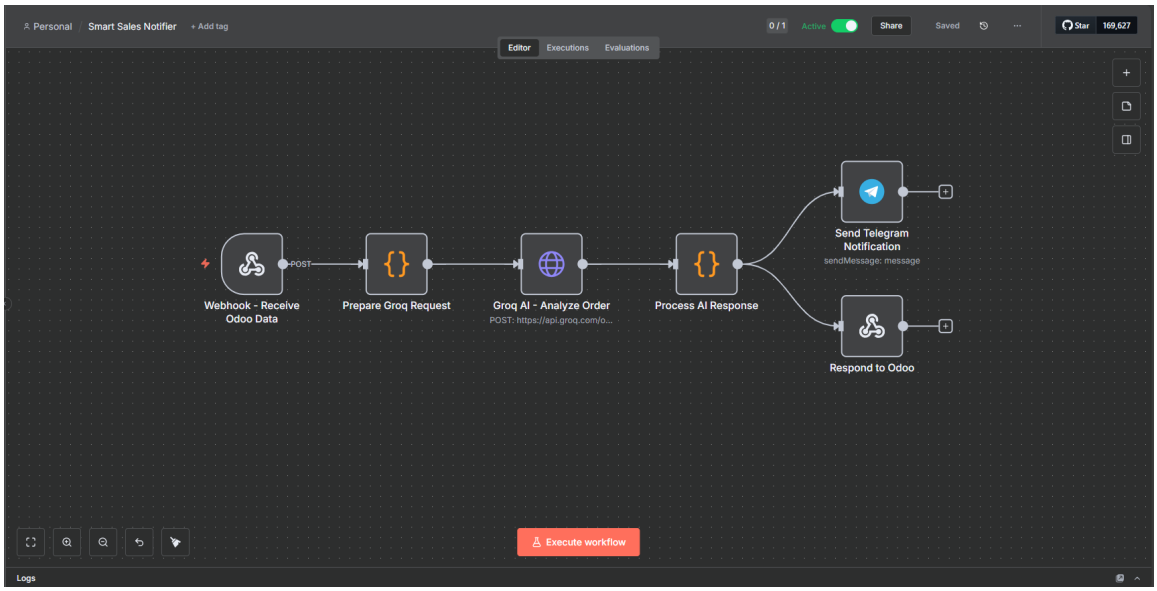

FIGURE 6.1 – Vue d'ensemble du workflow n8n

6.3 Configuration des Nœuds

6.3.1 Nœud 1 : Webhook

Le webhook est le point d'entrée du workflow. Il reçoit les données JSON envoyées par Odoo.

Configuration du Webhook

- **HTTP Method :** POST
- **Path :** sales-notifier
- **Response Mode :** Using 'Respond to Webhook' Node

URL du webhook : `https://n8n.dahbimoad.com/webhook/sales-notifier`

6.3.2 Nœud 2 : Prepare Groq Request

Ce nœud JavaScript prépare la requête pour l'API Groq :

```
// Extract order data from webhook body
const webhookData = $input.first().json;
const order = webhookData.body;

// Prepare the prompt for Groq
const productsText = order.products && order.products.length > 0
  ? order.products.map(function(p) {
      return p.product_name + " (qty: " + p.quantity + ")";
    }).join(', ')
  : 'No products listed';

const prompt = "Analyze this sales order and provide: " +
```

```
  "1) A brief analysis, 2) Priority level, 3) Follow-up action. " +
  "Order: " + order.order_name + ", Customer: " + order.customer_name +
  ", Total: " + order.total_amount + " " + order.currency;

return {
  order_data: order,
  groq_request: {
    model: "llama-3.1-8b-instant",
    messages: [
      {
        role: "system",
        content: "You are a sales assistant AI."
      },
      {
        role: "user",
        content: prompt
      }
    ],
    temperature: 0.7,
    max_tokens: 500
  }
};
```

Listing 6.1 – Code du noeud Prepare Request

6.3.3 Nœud 3 : Groq AI - HTTP Request

Paramètre	Valeur
Method	POST
URL	https ://api.groq.com/openai/v1/chat/completions
Authentication	Header Auth
Header Name	Authorization
Header Value	Bearer [GROQ_API_KEY]
Body	JSON (from previous node)

TABLE 6.2 – Configuration du nœud Groq AI

6.3.4 Nœud 4 : Process AI Response

Ce nœud traite la réponse de Groq et formate le message Telegram :

```
// Get order data and Groq response
const prepData = $('Prepare Groq Request').first().json;
const orderData = prepData.order_data;
const groqResponse = $input.first().json;

let aiAnalysis = {};
```

```
try {
  const content = groqResponse.choices[0].message.content;
  const jsonMatch = content.match(/\{[\s\S]*\}/);
  if (jsonMatch) {
    aiAnalysis = JSON.parse(jsonMatch[0]);
  }
} catch (e) {
  aiAnalysis = {
    analysis: 'New order received - requires review',
    priority: 'medium',
    action: 'Review order details'
  };
}

// Priority indicator mapping
const priorityIndicator = {
  'low': '[LOW]', 'medium': '[MEDIUM]',
  'high': '[HIGH]', 'urgent': '[URGENT]'
};

// Format and return Telegram message
return { message: formatTelegramMessage(orderData, aiAnalysis)
   };
```

Listing 6.2 – Code du noeud Process Response (extrait)

6.3.5 Nœud 5 : Telegram Notification

Paramètre	Valeur
Credential	Telegram Bot Token
Chat ID	ID du chat destination
Text	Message formaté avec l'analyse IA
Parse Mode	Markdown

TABLE 6.3 – Configuration du nœud Telegram

6.4 Configuration des Credentials

6.4.1 Groq API Key

1. Créer un compte sur https://console.groq.com
2. Générer une API Key
3. Dans n8n, créer un credential "Header Auth" :
   - Name : `Authorization`
   - Value : `Bearer gsk_xxxxx`

6.4.2 Telegram Bot

1. Contacter @BotFather sur Telegram
2. Créer un nouveau bot avec `/newbot`
3. Récupérer le token
4. Obtenir votre Chat ID via @userinfobot
5. Configurer le credential dans n8n

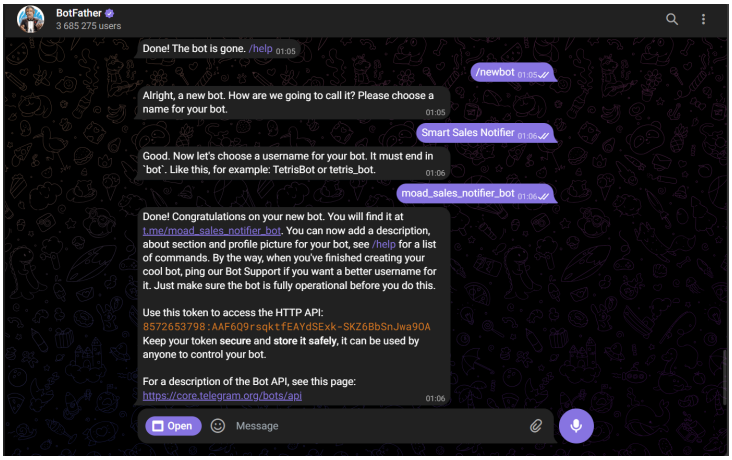

FIGURE 6.2 – Création du bot Telegram via BotFather

Chapitre 7

Tests et Résultats

7.1 Scénario de Test

Pour valider le fonctionnement du système, nous avons créé une commande de test avec les paramètres suivants :

Élément	**Valeur**
Client	TechCorp International
Produits	MacBook Pro 16", AirPods Pro 2
Montant Total	2,748.00 EUR
Résultat Attendu	Notification Telegram avec analyse IA

TABLE 7.1 – Données du scénario de test

7.2 Flux d'Exécution

1. Création de la commande dans Odoo
2. Confirmation de la commande (passage à l'état "Sales Order")
3. Clic sur le bouton "AI Analysis"
4. Envoi des données au webhook n8n
5. Traitement par Groq AI
6. Réception de la notification Telegram
7. Mise à jour de l'onglet AI Analysis dans Odoo

7.3 Captures d'Écran des Tests

7.3.1 Commande de Test

Voici la commande créée pour tester le système :

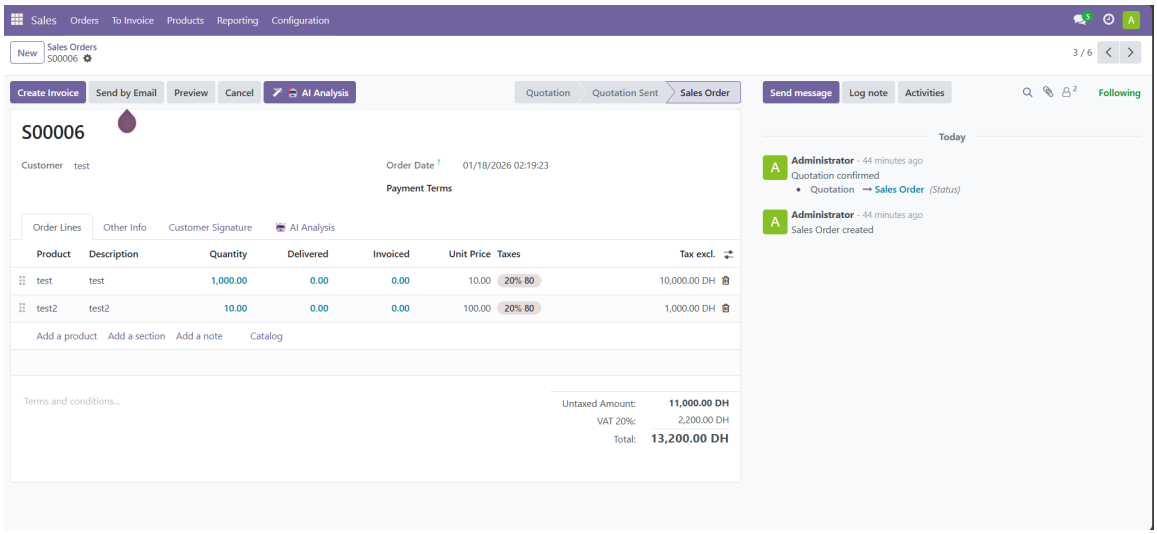

FIGURE 7.1 – Commande de test dans Odoo

7.3.2 Résultat de l'Analyse IA

Après avoir cliqué sur le bouton "AI Analysis", l'onglet affiche les résultats :

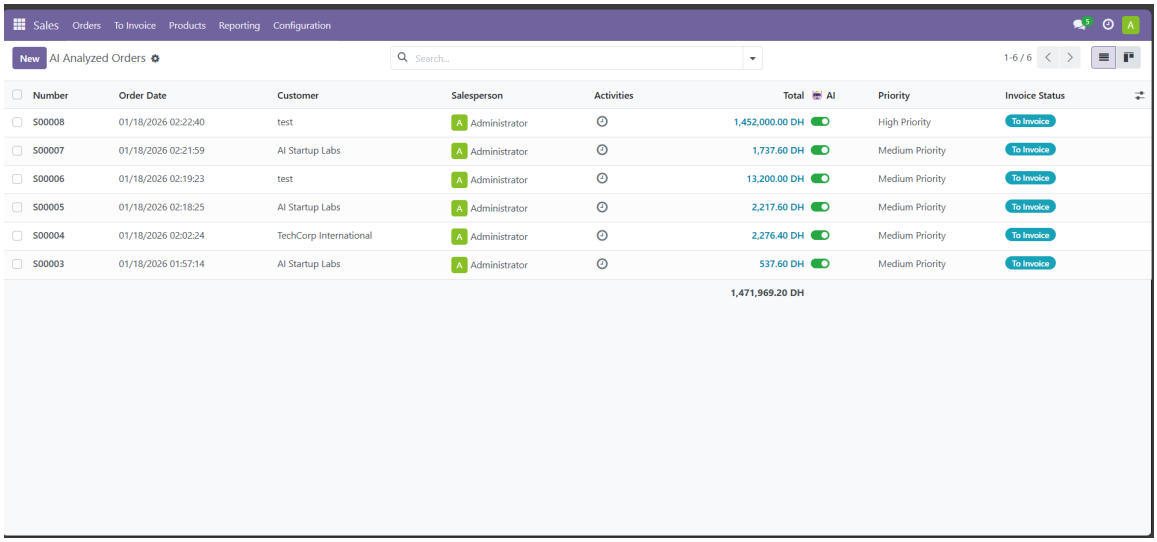

FIGURE 7.2 – Onglet AI Analysis avec les résultats de l'analyse

7.3.3 Notification Telegram

La notification reçue sur Telegram avec l'analyse IA :

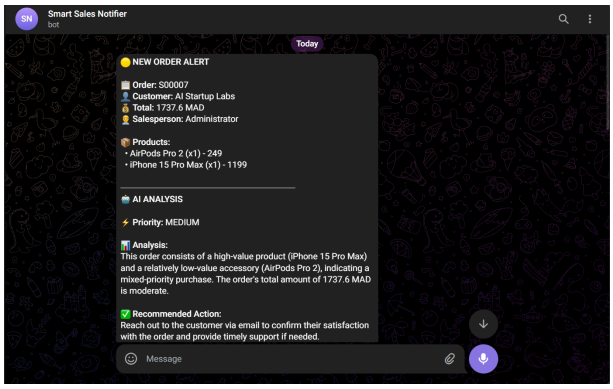

FIGURE 7.3 – Notification Telegram avec l'analyse IA

7.4 Métriques de Performance

Métrique	**Valeur**	**Objectif**
Temps de réponse webhook	2.5s	<5s
Temps d'analyse IA (Groq)	0.8s	<2s
Temps total (end-to-end)	3.5s	<10s
Taux de succès	100%	>95%

TABLE 7.2 – Métriques de performance du système

Chapitre 8

Conclusion

8.1 Objectifs Atteints

Ce projet a permis d'atteindre tous les objectifs fixés :

- ✔ **Module Odoo fonctionnel** - Le module Smart Sales Notifier est pleinement opérationnel et intégré au système de ventes
- ✔ **Intégration IA** - L'analyse des commandes par Groq LLaMA fonctionne et fournit des insights pertinents
- ✔ **Automatisation** - Le workflow n8n orchestre efficacement la communication entre les services
- ✔ **Notifications temps réel** - Les alertes Telegram sont envoyées instantanément

8.2 Compétences Acquises

— Développement de modules Odoo 17 (Python, XML)
— Intégration d'APIs REST (Groq, Telegram)
— Orchestration de workflows avec n8n
— Architecture microservices
— Déploiement avec Docker

8.3 Perspectives d'Amélioration

1. **Dashboard Analytics** - Ajouter des tableaux de bord pour visualiser les analyses IA
2. **Multi-canaux** - Intégrer d'autres canaux de notification (Slack, Email, SMS)
3. **ML Avancé** - Utiliser des modèles plus sophistiqués pour la prédiction des ventes
4. **Historique** - Conserver un historique des analyses pour le machine learning

Annexe A

Ressources et Liens

A.1 Code Source

Le code source complet de ce projet est disponible sur GitHub. Le repository contient :

— Le module Odoo `smart_sales_notifier` complet
— Le fichier `docker-compose.yml` pour le déploiement
— Le workflow n8n exporté en JSON
— La documentation complète

Repository GitHub

https://github.com/dahbimoad/smart_sales_notifier_odoo.git

A.2 Vidéo Démonstration

Une vidéo de démonstration complète du projet est disponible, montrant :

— L'installation et la configuration du module
— La création d'une commande de vente
— Le déclenchement de l'analyse IA
— La réception de la notification Telegram
— L'affichage des résultats dans Odoo

Vidéo Démo - Google Drive

https://drive.google.com/drive/folders/1VhpLWys0xJnrltJVffv65LXgDvlXMtEo?usp=sharing

A.3 Technologies et Documentation

Technologie	Documentation Officielle
Odoo 17	https://www.odoo.com/documentation/17.0/
n8n	https://docs.n8n.io/
Groq API	https://console.groq.com/docs
Telegram Bot API	https://core.telegram.org/bots/api
Docker	https://docs.docker.com/

TABLE A.1 – Liens vers la documentation officielle