🚚 Smart Delivery

Documentation Technique Complète

التوثيق التقني الكامل

Version 18.0.1.4.0 | Odoo 18

Module de Gestion de Livraison Intelligent

وحدة إدارة التوصيل الذكي

🇫🇷🇬🇧 Documentation en Français

📑 Table des Matières

1. Introduction et Vue d'Ensemble

1.1 Description du Projet

Smart Delivery est un module Odoo 18 complet pour la gestion intelligente des livraisons. Il offre un système de dispatching automatique des livreurs, une validation multi-niveaux des livraisons (OTP, signature, photo, biométrie), et une API REST complète avec authentification JWT pour les applications mobiles.

> **Caractéristiques Principales:**
>
> - Dispatching automatique basé sur la distance, notation et disponibilité
> - Validation OTP, signature électronique, photo et biométrique
> - API REST avec authentification JWT
> - Suivi GPS en temps réel
> - Gestion multi-entreprises
> - Facturation automatique intégrée avec Odoo Accounting

1.2 Acteurs du Système

Acteur	Description	Accès
Administrateur	Gestion complète du système, configuration des règles, approbation des inscriptions	Backend Odoo + API
Entreprise	Client qui envoie des colis. Peut créer et suivre ses commandes	Backend Odoo (limité) + API
Livreur	Effectue les livraisons, met à jour sa position, valide les conditions	API uniquement (App Mobile)
Destinataire	Reçoit le colis, fournit OTP/signature si requis	Interaction via livreur

2. Architecture Technique

2.1 Structure du Module

```
smart_delivery/
├── __manifest__.py         # Configuration du module
├── __init__.py             # Imports Python
├── hooks.py                # Hooks d'installation/migration
├── controllers/
│   └── api.py              # Contrôleur API REST (4000+ lignes)
├── models/
│   ├── delivery_order.py   # Commandes de livraison
│   ├── livreur.py          # Gestion des livreurs
│   ├── enterprise.py       # Gestion des entreprises
│   ├── sector_rule.py      # Règles de secteur et tarification
│   ├── delivery_condition.py # Conditions de validation
│   ├── billing.py          # Facturation
│   ├── delivery_route.py   # Itinéraires GPS
│   ├── api_log.py          # Journal des appels API
│   ├── res_users.py        # Extension utilisateurs
│   ├── res_partner.py      # Extension partenaires
│   └── account_move.py     # Extension factures
├── security/
│   ├── security_groups.xml # Groupes de sécurité
│   ├── ir.model.access.csv # Droits d'accès
│   └── security_rules.xml  # Règles d'enregistrement
├── views/                  # Vues XML
├── data/                   # Données initiales
├── utils/
│   └── jwt_auth.py         # Authentification JWT
└── docs/                   # Documentation
```

2.2 Dépendances

Type	Dépendance	Utilisation
Module Odoo	`base`	Fonctionnalités de base, utilisateurs, partenaires
Module Odoo	`web`	Interface web, contrôleurs HTTP
Module Odoo	`mail`	Chatter, tracking des modifications, activités
Module Odoo	`contacts`	Gestion des contacts (expéditeurs, destinataires)
Module Odoo	`account`	Comptabilité, génération de factures
Module Odoo	`sale`	Produits de service pour la facturation
Python	`PyJWT`	Génération et validation des tokens JWT
Python	`cryptography`	Cryptographie pour JWT

3. Modèles de Données

3.1 delivery.order - Commande de Livraison

Modèle principal représentant une commande de livraison.

Champ	Type	Description
name	Char	Référence auto-générée (DEL00001)
reference	Char	Référence externe optionnelle
sector_type	Selection	standard, premium, express, fragile, medical
sender_id	Many2one	Entreprise expéditrice (res.partner)
receiver_name	Char	Nom du destinataire
receiver_phone	Char	Téléphone du destinataire
pickup_lat/long	Float	Coordonnées GPS point de collecte
drop_lat/long	Float	Coordonnées GPS point de livraison
assigned_livreur_id	Many2one	Livreur assigné
status	Selection	draft, assigned, on_way, delivered, failed
otp_required	Boolean	OTP requis pour validation
signature_required	Boolean	Signature requise
photo_required	Boolean	Photo requise
biometric_required	Boolean	Biométrie requise
distance_km	Float	Distance calculée (Haversine)

3.2 delivery.livreur - Livreur

Gestion des livreurs avec documents d'identification obligatoires.

Champ	Type	Description
name	Char	Nom complet
phone	Char	Numéro de téléphone

email	Char	Email (login)
nni	Char	Numéro National d'Identification
nni_photo	Binary	Photo du document NNI
livreur_photo	Binary	Photo d'identité
carte_grise_photo	Binary	Photo carte grise véhicule
assurance_photo	Binary	Photo assurance
vehicle_type	Selection	motorcycle, car, bicycle, truck
sector_ids	Many2many	Types de secteurs autorisés
availability	Boolean	Disponibilité actuelle
rating	Float	Note moyenne (0-5)
current_lat/long	Float	Position GPS actuelle
verified	Boolean	Compte vérifié
registration_status	Selection	pending, approved, rejected
user_id	Many2one	Utilisateur Odoo lié (auto-créé)

3.3 delivery.enterprise - Entreprise Cliente

Entreprises qui utilisent le service de livraison.

Champ	Type	Description
name	Char	Nom de l'entreprise
email	Char	Email (login)
phone	Char	Téléphone
logo	Binary	Logo de l'entreprise
address, city	Char/Text	Adresse
registration_status	Selection	pending, approved, rejected
user_id	Many2one	Utilisateur Odoo lié

partner_id	Many2one	Contact partenaire lié

3.4 sector.rule - Règles de Secteur

Configuration des types de livraison et leur tarification.

Type de Secteur	Prix Base (MRU)	Frais/km	Exigences
Standard	50	10	Aucune
Premium	100	10	OTP + Signature
Express	150	15	OTP + Photo
Fragile	120	12	OTP + Signature + Photo
Médical	200	20	OTP + Signature + Photo + Biométrie

> **Note:** Les 5 premiers kilomètres sont inclus dans le prix de base. Les frais kilométriques s'appliquent au-delà.

3.5 delivery.condition - Conditions de Validation

Champ	Type	Description
order_id	Many2one	Commande associée
otp_value	Char	Code OTP généré (6 chiffres)
otp_verified	Boolean	OTP vérifié
signature_file	Binary	Fichier signature
photo	Binary	Photo de livraison
biometric_score	Float	Score biométrique (min 0.7)
validated	Boolean	Toutes conditions validées

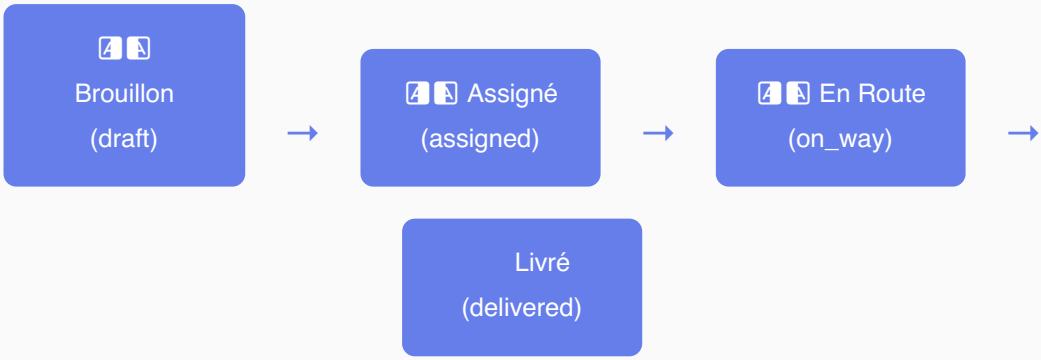
3.6 delivery.billing - Facturation

Champ	Type	Description
order_id	Many2one	Commande associée

distance_km	Float	Distance de livraison
base_tariff	Float	Tarif de base
extra_fee	Float	Frais de distance supplémentaires
total_amount	Float	Montant total
state	Selection	draft, confirmed, paid, cancelled
invoice_id	Many2one	Facture Odoo générée

4. Flux de Travail

4.1 Cycle de Vie d'une Commande



4.2 Processus Détaillé

1. **Création de Commande**
   - L'entreprise crée une commande via le backend Odoo ou l'API
   - Les coordonnées GPS de collecte et livraison sont définies
   - Le type de secteur détermine les exigences de validation
   - Un OTP est généré si requis
2. **Dispatching Automatique**
   - Le système calcule un score pour chaque livreur disponible
   - Score = (Distance × 50%) + (Note × 20%) + (Repos × 10%) + (Vitesse × 20%)
   - Le livreur avec le meilleur score est automatiquement assigné
3. **Exécution de la Livraison**
   - Le livreur accepte la commande via l'app mobile

- Il collecte le colis et passe en statut "En Route"
- Sa position GPS est mise à jour en temps réel

4. **Validation et Livraison**
   - À l'arrivée, le livreur soumet les preuves requises
   - Le destinataire fournit l'OTP reçu par SMS
   - Signature capturée si requise
   - Photo de preuve prise si requise
   - Vérification biométrique si requise (médical)
5. **Facturation**
   - La facturation est générée automatiquement à la livraison
   - Une facture Odoo est créée avec les lignes détaillées
   - Paiement en espèces à la livraison (COD)

4.3 Inscription des Livreurs

1. Le livreur s'inscrit via l'API mobile avec ses documents
2. Statut initial: **En attente** (pending)
3. L'admin vérifie les documents soumis
4. Approbation → Compte activé, peut recevoir des commandes
5. Rejet → Notification avec motif, peut soumettre à nouveau

5. Modules Odoo Utilisés

5.1 Module Base (base)

- **res.users**: Étendu pour ajouter le type d'utilisateur delivery (admin, enterprise, livreur)
- **res.partner**: Étendu pour marquer les entreprises de livraison
- **ir.sequence**: Séquence DEL00001 pour les commandes
- **ir.model.access**: Droits d'accès aux modèles
- **ir.rule**: Règles de sécurité par enregistrement

5.2 Module Mail (mail)

- **mail.thread**: Hérité par delivery.order, delivery.livreur, delivery.enterprise
- Permet le chatter (historique des messages)
- Tracking automatique des changements de champs
- **mail.activity.mixin**: Gestion des activités planifiées

5.3 Module Account (account)

- **account.move**: Création de factures client
- **account.move.line**: Lignes de facture (service + frais distance)
- **account.payment**: Enregistrement des paiements
- **account.journal**: Journaux de vente et caisse
- **account.account**: Comptes comptables (revenus)

5.4 Module Sale (sale)

- **product.product**: Produits de service pour chaque type de livraison
- **product.category**: Catégorie "Services de Livraison"
- Produits créés: DEL-STD, DEL-PRM, DEL-EXP, DEL-FRG, DEL-MED, DEL-DIST

5.5 Module Web (web)

- **http.Controller**: Base des contrôleurs API REST
- Gestion des requêtes HTTP, JSON, authentification
- CSRF désactivé pour les endpoints API

6. Sécurité et Permissions

6.1 Groupes de Sécurité

Groupe	Hérite de	Permissions
Livreur (group_livreur)	-	API uniquement, pas d'accès backend Odoo
Entreprise (group_enterprise)	base.group_user	Accès Smart Delivery, voir/créer ses commandes
Admin (group_admin)	group_enterprise	Accès total, configuration, suppression

6.2 Règles d'Accès (ir.model.access)

Modèle	Admin	Entreprise	Livreur
delivery.order	CRUD	CRU (pas delete)	-
delivery.livreur	CRUD	R (lecture seule)	-
delivery.enterprise	CRUD	-	-

sector.rule	CRUD	R	-
delivery.billing	CRUD	R	-
api.log	CRUD	-	-

> **Note:** Les livreurs n'ont pas d'accès au backend Odoo. Toutes leurs opérations passent par l'API REST qui utilise `sudo()` pour les opérations autorisées.

7. API REST

7.1 Authentification JWT

Toutes les requêtes API (sauf login et inscription) nécessitent un token JWT dans le header:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

7.2 Endpoints Principaux

🔐 Authentification

Méthode	Endpoint	Description
POST	/smart_delivery/api/auth/login	Connexion, retourne token JWT
POST	/smart_delivery/api/auth/logout	Déconnexion

🔐 Inscription (Public)

Méthode	Endpoint	Description
GET	/smart_delivery/api/sectors	Liste des secteurs disponibles
POST	/smart_delivery/api/livreur/register	Inscription livreur
POST	/smart_delivery/api/enterprise/register	Inscription entreprise

🔐 Commandes (Entreprise)

Méthode	Endpoint	Description
GET	/smart_delivery/api/enterprise/orders	Liste des commandes de l'entreprise

POST	/smart_delivery/api/enterprise/orders/create	Créer une commande
GET	/smart_delivery/api/enterprise/orders/{id}	Détails d'une commande

🚴 Livreur

Méthode	Endpoint	Description
GET	/smart_delivery/api/livreur/profile	Profil du livreur
PUT	/smart_delivery/api/livreur/profile	Mettre à jour le profil
POST	/smart_delivery/api/livreur/location	Mettre à jour la position GPS
POST	/smart_delivery/api/livreur/availability	Changer la disponibilité
GET	/smart_delivery/api/livreur/orders	Commandes assignées
POST	/smart_delivery/api/livreur/orders/{id}/accept	Accepter une commande
POST	/smart_delivery/api/livreur/orders/{id}/start	Démarrer la livraison
POST	/smart_delivery/api/livreur/orders/{id}/complete	Terminer avec validation

Validation

Méthode	Endpoint	Description
POST	/smart_delivery/api/orders/{id}/verify-otp	Vérifier le code OTP
POST	/smart_delivery/api/orders/{id}/signature	Soumettre la signature
POST	/smart_delivery/api/orders/{id}/photo	Soumettre la photo
POST	/smart_delivery/api/orders/{id}/biometric	Soumettre le score biométrique

7.3 Exemple de Requête Login

```
POST /smart_delivery/api/auth/login
Content-Type: application/json

{
    "login": "livreur@example.com",
    "password": "password123"
}

// Réponse:
{
    "success": true,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
        "id": 5,
        "name": "Ahmed Livreur",
        "login": "livreur@example.com",
        "type": "livreur"
    },
    "livreur": {
        "id": 3,
        "name": "Ahmed Livreur",
        "phone": "+22212345678",
        "vehicle_type": "motorcycle",
        "availability": true,
        "registration_status": "approved"
    },
    "expires_in": 86400
}
```

8. Facturation

8.1 Calcul du Tarif

```
Montant Total = Prix de Base + Frais de Distance

Frais de Distance = max(0, distance_km - distance_gratuite) × frais_par_km

Exemple pour livraison Express de 12 km:
- Prix de Base: 150 MRU
- Distance gratuite: 5 km
- Frais/km: 15 MRU
- Frais de distance: (12 - 5) × 15 = 105 MRU
- Total: 150 + 105 = 255 MRU
```

8.2 Génération de Facture

La facture Odoo est créée automatiquement avec:

- **Ligne 1:** Service de Livraison [Type] - Commande DEL00001
- **Ligne 2:** Frais de distance (X km au-delà de 5km)

8.3 Paiement COD

Mode de paiement unique: **Espèces à la livraison** (Cash On Delivery)

- Le livreur collecte le paiement du destinataire
- Le bouton "Marquer comme payé" dans le backend crée le règlement
- Réconciliation automatique avec la facture

9. Installation

9.1 Prérequis

```
# Installer les dépendances Python
pip install PyJWT cryptography
```

9.2 Installation du Module

1. Copier le dossier `smart_delivery` dans le répertoire addons
2. Mettre à jour la liste des applications dans Odoo
3. Rechercher "Smart Delivery" et cliquer sur Installer

9.3 Configuration Post-Installation

1. Configurer la comptabilité (Plan comptable)
2. Vérifier les produits de service créés
3. Configurer les règles de secteur si nécessaire
4. Créer le premier utilisateur administrateur

> **Important:** En production, changez la clé secrète JWT dans `utils/jwt_auth.py`

التوثيق باللغة العربية

جدول المحتويات

1. مقدمة ونظرة عامة

1.1 وصف المشروع

Smart Delivery هو وحدة Odoo 18 متكاملة لإدارة التوصيل الذكي. يوفر نظام توزيع تلقائي للسائقين، والتحقق متعدد المستويات من التسليم (OTP، التوقيع، الصورة، البيومترية)، وواجهة برمجة تطبيقات REST كاملة مع مصادقة JWT لتطبيقات الهاتف المحمول.

الميزات الرئيسية:

- التوزيع التلقائي بناءً على المسافة والتقييم والتوفر
- التحقق عبر OTP، التوقيع الإلكتروني، الصورة والبيومترية
- واجهة برمجة تطبيقات REST مع مصادقة JWT
- تتبع GPS في الوقت الفعلي
- إدارة متعددة الشركات
- الفوترة التلقائية المتكاملة مع محاسبة Odoo

1.2 الجهات الفاعلة في النظام

الجهة	الوصف	الوصول
المسؤول	إدارة النظام الكاملة، تكوين القواعد، الموافقة على التسجيلات	واجهة Odoo + API
الشركة	العميل الذي يرسل الطرود. يمكنه إنشاء وتتبع طلباته	واجهة Odoo (محدود) + API
السائق	ينفذ التسليمات، يحدّث موقعه، يتحقق من الشروط	API فقط (تطبيق الهاتف)
المستلم	يستلم الطرد، يقدم OTP/التوقيع إذا لزم الأمر	التفاعل عبر السائق

2. الهندسة التقنية

2.1 هيكل الوحدة

```
/smart_delivery
├── __manifest__.py      # تكوين الوحدة
├── __init__.py          # استيراد Python
├── hooks.py             # خطافات التثبيت/الترحيل
├── /controllers
│   └── api.py           # وحدة تحكم API REST
├── /models
│   ├── delivery_order.py    # طلبات التوصيل
│   ├── livreur.py           # إدارة السائقين
│   ├── enterprise.py        # إدارة الشركات
│   ├── sector_rule.py       # قواعد القطاع والتسعير
│   ├── delivery_condition.py # شروط التحقق
│   ├── billing.py           # الفوترة
│   ├── delivery_route.py    # مسارات GPS
│   └── api_log.py           # سجل مكالمات API
├── /security            # الأمان
├── /views               # العروض XML
├── /data                # البيانات الأولية
├── /utils
│   └── jwt_auth.py      # مصادقة JWT
```

2.2 الإعتماديات

النوع	الاعتمادية	الاستخدام
وحدة Odoo	`base`	الوظائف الأساسية، المستخدمون، الشركاء
وحدة Odoo	`mail`	المحادثات، تتبع التغييرات، الأنشطة
وحدة Odoo	`account`	المحاسبة، إنشاء الفواتير
وحدة Odoo	`sale`	منتجات الخدمة للفوترة
Python	`PyJWT`	إنشاء والتحقق من رموز JWT

3. نماذج البيانات

3.1 delivery.order - طلب التوصيل

النموذج الرئيسي الذي يمثل طلب التوصيل.

الحقل	النوع	الوصف
name	Char	المرجع المُنشأ تلقائياً (DEL00001)
sector_type	Selection	عادي، مميز، سريع، هش، طبي
sender_id	Many2one	الشركة المرسلة
receiver_phone	Char	هاتف المستلم
pickup_lat/long	Float	إحداثيات GPS نقطة الاستلام
drop_lat/long	Float	إحداثيات GPS نقطة التسليم
assigned_livreur_id	Many2one	السائق المُعيّن
status	Selection	مسودة، مُعيّن، في الطريق، تم التسليم، فشل
otp_required	Boolean	OTP مطلوب للتحقق
signature_required	Boolean	التوقيع مطلوب
photo_required	Boolean	الصورة مطلوبة
distance_km	Float	المسافة المحسوبة (Haversine)

3.2 delivery.livreur - السائق

إدارة السائقين مع وثائق التعريف الإلزامية.

الحقل	النوع	الوصف
name	Char	الاسم الكامل
phone	Char	رقم الهاتف
email	Char	البريد الإلكتروني (تسجيل الدخول)
nni	Char	رقم التعريف الوطني
nni_photo	Binary	صورة وثيقة NNI
livreur_photo	Binary	صورة الهوية
carte_grise_photo	Binary	صورة البطاقة الرمادية للمركبة
assurance_photo	Binary	صورة التأمين

vehicle_type	Selection	دراجة نارية، سيارة، دراجة، شاحنة
availability	Boolean	التوفر الحالي
rating	Float	التقييم المتوسط (0-5)
current_lat/long	Float	موقع GPS الحالي
registration_status	Selection	قيد الانتظار، موافق عليه، مرفوض

3.3 sector.rule - قواعد القطاع

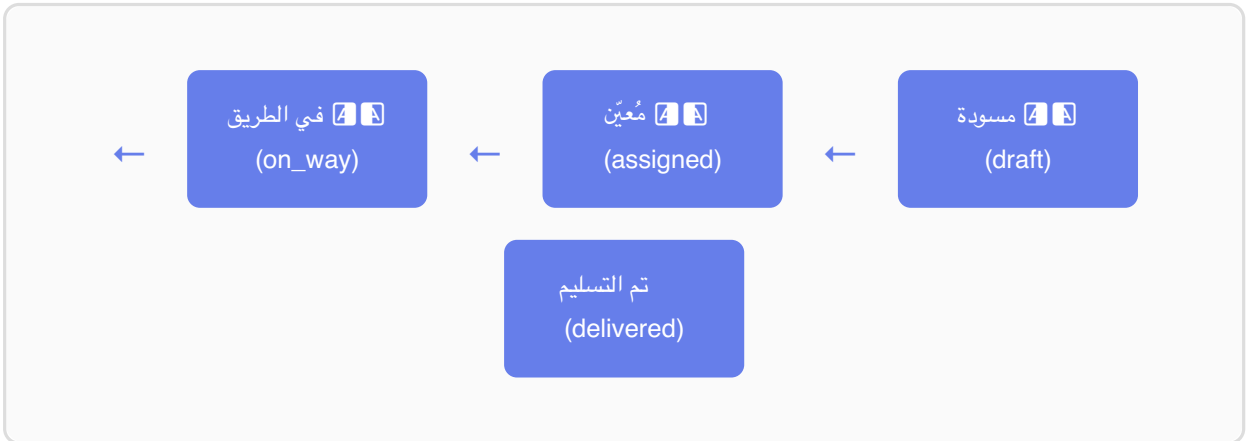
تكوين أنواع التوصيل وتسعيرها.

نوع القطاع	السعر الأساسي (MRU)	الرسوم/كم	المتطلبات
عادي	50	10	لا شيء
مميز	100	10	OTP + التوقيع
سريع	150	15	OTP + الصورة
هش	120	12	OTP + التوقيع + الصورة
طبي	200	20	OTP + التوقيع + الصورة + البيومترية

> **ملاحظة:** أول 5 كيلومترات مشمولة في السعر الأساسي. تُطبق رسوم الكيلومترات بعد ذلك.

4. سير العمل

4.1 دورة حياة الطلب



4.2 العملية المفصلة

1. **إنشاء الطلب**
   - تنشئ الشركة طلباً عبر واجهة Odoo أو API
   - تُحدد إحداثيات GPS للاستلام والتسليم
   - يحدد نوع القطاع متطلبات التحقق
   - يُنشأ OTP إذا كان مطلوباً
2. **التوزيع التلقائي**
   - يحسب النظام درجة لكل سائق متاح
   - الدرجة = (المسافة × 50%) + (التقييم × 20%) + (الراحة × 10%) + (السرعة × 20%)
   - يُعيّنالسائق ذو أفضل درجة تلقائياً
3. **تنفيذ التوصيل**
   - يقبل السائق الطلب عبر تطبيق الهاتف
   - يستلم الطرد ويتحول إلى حالة "في الطريق"
   - يُحدّث موقع GPS في الوقت الفعلي
4. **التحقق والتسليم**
   - عند الوصول، يقدم السائق الأدلة المطلوبة
   - يقدم المستلم OTP المستلم عبر SMS
   - التقاط التوقيع إذا كان مطلوباً
   - التقاط صورة الإثبات إذا كانت مطلوبة
5. **الفوترة**
   - تُنشأ الفوترة تلقائياً عند التسليم
   - تُنشأ فاتورة Odoo مع البنود المفصلة
   - الدفع نقداً عند التسليم (COD)

4.3 تسجيل السائقين

1. يسجل السائق عبر API الهاتف مع وثائقه
2. الحالة الأولية: **قيد الانتظار** (pending)
3. يتحقق المسؤول من الوثائق المقدمة
4. الموافقة ← تفعيل الحساب، يمكنه استلام الطلبات
5. الرفض ← إشعار مع السبب، يمكنه إعادة التقديم

5. وحدات Odoo المستخدمة

5.1 وحدة Base

- **res.users** موسّع لإضافة نوع مستخدم التوصيل

- **res.partner** موسّع لتحديد شركات التوصيل
- **ir.sequence**: تسلسل DEL00001 للطلبات

5.2 وحدة Mail

- **mail.thread** موروث من طلبات التوصيل والسائقين
- يتيح المحادثات (سجل الرسائل)
- تتبع تلقائي لتغييرات الحقول

5.3 وحدة Account

- **account.move**: إنشاء فواتير العملاء
- **account.payment**: تسجيل المدفوعات
- **account.journal**: دفاتر المبيعات والصندوق

5.4 وحدة Sale

- **product.product** منتجات الخدمة لكل نوع توصيل
- المنتجات المُنشأة: DEL-STD, DEL-PRM, DEL-EXP, DEL-FRG, DEL-MED

6. الأمان والصلاحيات

6.1 مجموعات الأمان

المجموعة	ترث من	الصلاحيات
السائق	-	API فقط، بدون وصول لواجهة Odoo
الشركة	base.group_user	وصول Smart Delivery، عرض/إنشاء طلباته
المسؤول	group_enterprise	وصول كامل، التكوين، الحذف

6.2 قواعد الوصول

النموذج	المسؤول	الشركة	السائق
delivery.order	CRUD	CRU	-
delivery.livreur	CRUD	R	-
sector.rule	CRUD	R	-
delivery.billing	CRUD	R	-

7. واجهة برمجة التطبيقات REST

7.1 مصادقة JWT

جميع طلبات API (باستثناء تسجيل الدخول والتسجيل) تتطلب رمز JWT في الرأس:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

7.2 النقاط النهائية الرئيسية

المصادقة

الطريقة	النقطة النهائية	الوصف
POST	smart_delivery/api/auth/login/	تسجيل الدخول، إرجاع رمز JWT
POST	smart_delivery/api/auth/logout/	تسجيل الخروج

التسجيل (عام)

الطريقة	النقطة النهائية	الوصف
GET	smart_delivery/api/sectors/	قائمة القطاعات المتاحة
POST	smart_delivery/api/livreur/register/	تسجيل السائق
POST	smart_delivery/api/enterprise/register/	تسجيل الشركة

الطلبات (الشركة)

الطريقة	النقطة النهائية	الوصف
GET	smart_delivery/api/enterprise/orders/	قائمة طلبات الشركة
POST	smart_delivery/api/enterprise/orders/create/	إنشاء طلب

السائق

الطريقة	النقطة النهائية	الوصف
GET	smart_delivery/api/livreur/profile/	ملف السائق الشخصي
POST	smart_delivery/api/livreur/location/	تحديث موقع GPS

GET	smart_delivery/api/livreur/orders/	الطلبات المُعيّنة
POST	smart_delivery/api/livreur/orders/{id}/complete/	إنهاء مع التحقق

التحقق

الطريقة	النقطة النهائية	الوصف
POST	smart_delivery/api/orders/{id}/verify-otp/	التحقق من رمز OTP
POST	smart_delivery/api/orders/{id}/signature/	تقديم التوقيع
POST	smart_delivery/api/orders/{id}/photo/	تقديم الصورة

8. الفوترة

8.1 حساب التعرفة

```
المبلغ الإجمالي = السعر الأساسي + رسوم المسافة

رسوم المسافة = max(0, المسافة_كم - المسافة_المجانية) × الرسوم_لكل_كم

مثال لتوصيل سريع لمسافة 12 كم:
- السعر الأساسي: 150 MRU
- المسافة المجانية: 5 كم
- الرسوم/كم: 15 MRU
- رسوم المسافة: (12 - 5) × 15 = 105 MRU
- الإجمالي: 150 + 105 = 255 MRU
```

8.2 الدفع عند التسليم

طريقة الدفع الوحيدة: **نقداً عند التسليم** (COD)

- يجمع السائق الدفع من المستلم
- زر "وضع علامة مدفوع" في الواجهة ينشئ التسوية
- مطابقة تلقائية مع الفاتورة

9. التثبيت

9.1 المتطلبات الأولية

```
# تثبيت اعتماديات Python
pip install PyJWT cryptography
```

9.2 تثبيت الوحدة

1. نسخ مجلد `smart_delivery` إلى دليل addons
2. تحديث قائمة التطبيقات في Odoo
3. البحث عن "Smart Delivery" والنقر على تثبيت

9.3 التكوين بعد التثبيت

1. تكوين المحاسبة (جدول الحسابات)
2. التحقق من منتجات الخدمة المُنشأة
3. تكوين قواعد القطاع إذا لزم الأمر
4. إنشاء أول مستخدم مسؤول

> **مهم:** في الإنتاج، قم بتغيير المفتاح السري JWT في `utils/jwt_auth.py`

Pour générer un PDF: Ouvrez ce fichier dans un navigateur et utilisez Imprimer → Enregistrer en PDF

لإنشاء PDF: افتح هذا الملف في متصفح واستخدم طباعة ← حفظ كـ PDF