

Plateforme de Livraison de Nourriture - Documentation Complète

Table des Matières

1. [Vue d'ensemble du Projet](#)
 2. [Architecture Technique](#)
 3. [Installation et Configuration](#)
 4. [Guide d'Utilisation](#)
 5. [Déploiement](#)
 6. [API Documentation](#)
 7. [Intégration Stripe](#)
 8. [Maintenance et Support](#)
-

Vue d'ensemble du Projet

Description

Cette plateforme de livraison de nourriture est une solution complète comprenant : - **Application Web Client** : Interface pour les utilisateurs finaux - **Application Mobile** **Restaurateur** : Interface de gestion pour les restaurateurs - **API Backend** : Serveur Flask avec base de données PostgreSQL - **Système d'Abonnement** : Intégration Stripe pour les paiements récurrents

Fonctionnalités Principales

Pour les Clients :

- Navigation et recherche de restaurants

- Consultation des menus et plats
- Système d'abonnement avec plans multiples
- Interface responsive (desktop et mobile)

Pour les Restaurateurs :

- Tableau de bord avec statistiques en temps réel
- Gestion des plats (ajout, modification, suppression)
- Suivi des commandes en temps réel
- Gestion des horaires d'ouverture
- Interface optimisée mobile

Système Backend :

- API RESTful complète
 - Authentification sécurisée
 - Intégration Stripe (abonnements, webhooks)
 - Base de données PostgreSQL
 - Support CORS pour les applications frontend
-

Architecture Technique

Stack Technologique

Backend

- **Framework** : Flask (Python 3.11)
- **Base de données** : PostgreSQL
- **ORM** : SQLAlchemy avec Flask-SQLAlchemy
- **Migrations** : Flask-Migrate
- **Paielements** : Stripe API
- **Serveur** : Gunicorn (production)

Frontend Client

- **Framework** : React 19.1.0
- **Build Tool** : Vite
- **Styling** : Tailwind CSS
- **Composants UI** : shadcn/ui
- **Icônes** : Lucide React
- **HTTP Client** : Axios

Frontend Mobile Restaurateur

- **Framework** : React (optimisé mobile)
- **Build Tool** : Vite
- **Styling** : Tailwind CSS
- **Composants UI** : shadcn/ui
- **Interface** : Progressive Web App (PWA)

Modèle de Données

Entités Principales

Users (Utilisateurs)

```
- id (Primary Key)
- username (Unique)
- email (Unique)
- password_hash
- role (client/restaurateur)
- created_at
- subscription_id (Stripe)
- subscription_status
```

Restaurants

- id (**Primary Key**)
- name
- address
- phone
- email
- owner_id (**Foreign Key** -> Users)
- created_at
- is_active

Dishes (Plats)

- id (**Primary Key**)
- name
- description
- price
- restaurant_id (**Foreign Key** -> Restaurants)
- image_url
- is_available
- created_at

Orders (Commandes)

- id (**Primary Key**)
- customer_id (**Foreign Key** -> Users)
- restaurant_id (**Foreign Key** -> Restaurants)
- total_amount
- status (pending/preparing/ready/delivered)
- created_at
- delivery_address

Order_Items (Articles de Commande)

- id (**Primary Key**)
- order_id (**Foreign Key** -> Orders)
- dish_id (**Foreign Key** -> Dishes)
- quantity
- unit_price

Installation et Configuration

Prérequis

- Python 3.11+
- Node.js 20+

- PostgreSQL 13+
- Compte Stripe (pour les paiements)
- Git

Installation Locale

1. Cloner le Projet

```
git clone <repository-url>
cd food-delivery-platform
```

2. Configuration du Backend

```
cd backend

# Créer l'environnement virtuel
python3.11 -m venv venv
source venv/bin/activate # Linux/Mac
# ou
venv\Scripts\activate    # Windows

# Installer les dépendances
pip install -r requirements.txt

# Configuration des variables d'environnement
cp .env.example .env
# Éditer .env avec vos paramètres :
# DATABASE_URL=postgresql://user:password@localhost/food_delivery_db
# STRIPE_SECRET_KEY=sk_test_...
# STRIPE_WEBHOOK_SECRET=whsec_...
```

3. Configuration de la Base de Données

```
# Créer la base de données PostgreSQL
createdb food_delivery_db

# Initialiser les migrations
flask db init
flask db migrate -m "initial migration"
flask db upgrade
```

4. Configuration du Frontend Client

```
cd ../food-delivery-client

# Installer les dépendances
pnpm install

# Démarrer en mode développement
pnpm run dev
```

5. Configuration du Frontend Mobile

```
cd ../restaurant-manager-mobile

# Installer les dépendances
pnpm install

# Démarrer en mode développement
pnpm run dev --port 5174
```

Configuration Stripe

1. Créer un Compte Stripe

1. Aller sur stripe.com
2. Créer un compte développeur
3. Récupérer les clés API (test et production)

2. Configurer les Produits

```
# Dans le tableau de bord Stripe, créer :
# - Produit "Abonnement Basic" (9.99€/mois)
# - Produit "Abonnement Premium" (19.99€/mois)
# - Produit "Abonnement Enterprise" (39.99€/mois)
```

3. Configurer les Webhooks

```
# URL du webhook : https://votre-backend.onrender.com/stripe/webhook
# Événements à écouter :
# - customer.subscription.created
# - customer.subscription.updated
# - customer.subscription.deleted
# - invoice.payment_succeeded
# - invoice.payment_failed
```

Guide d'Utilisation

Application Web Client

Accès

- URL locale : `http://localhost:5173`
- URL production : `https://votre-frontend.onrender.com`

Fonctionnalités

Navigation 1. Page d'accueil avec présentation 2. Section "Restaurants" avec liste et recherche 3. Section "Abonnements" avec plans tarifaires 4. Authentification (connexion/inscription)

Processus d'Abonnement 1. Choisir un plan d'abonnement 2. Cliquer sur "S'abonner" 3. Redirection vers Stripe Checkout 4. Saisir les informations de paiement 5. Confirmation et retour sur la plateforme

Application Mobile Restaurateur

Accès

- URL locale : `http://localhost:5174`
- URL production : `https://votre-mobile.onrender.com`

Connexion

- **Mode démonstration** : `demo / demo`
- **Production** : Identifiants fournis par l'administrateur

Fonctionnalités

Tableau de Bord - Statistiques en temps réel - Commandes totales - Revenus du jour - Plats actifs - Commandes en attente

Gestion des Plats - Ajouter un nouveau plat - Modifier les informations (nom, description, prix) - Supprimer un plat - Gérer la disponibilité

Gestion des Commandes - Visualiser les commandes par statut - Mettre à jour le statut des commandes - Voir les détails client et livraison - Filtrer par statut (en attente, en préparation, prêt, livré)

Gestion des Horaires - Définir les horaires d'ouverture par jour - Activer/désactiver l'ouverture par jour - Actions rapides (horaires standards, étendus, fermeture temporaire)

API Backend

Endpoints Principaux

Authentification

```
POST /register - Inscription utilisateur
POST /login - Connexion utilisateur
POST /logout - Déconnexion
```

Restaurants

```
GET /restaurants - Liste des restaurants
POST /restaurants - Créer un restaurant (restaurateur)
PUT /restaurants/{id} - Modifier un restaurant
DELETE /restaurants/{id} - Supprimer un restaurant
```

Plats

```
GET /restaurants/{id}/dishes - Plats d'un restaurant
POST /dishes - Ajouter un plat
PUT /dishes/{id} - Modifier un plat
DELETE /dishes/{id} - Supprimer un plat
```

Commandes

```
GET /orders - Liste des commandes
POST /orders - Créer une commande
PUT /orders/{id} - Mettre à jour une commande
GET /orders/{id} - Détails d'une commande
```

Stripe


```
POST /create-checkout-session - Créer une session de paiement
POST /stripe/webhook - Webhook Stripe
GET /subscription-status - Statut d'abonnement utilisateur
```

Déploiement

Plateforme Recommandée : Render

Render offre un excellent rapport qualité-prix avec un niveau gratuit généreux.

Coûts Estimés

- **Backend** : Gratuit (750h/mois)
- **Frontend Client** : Gratuit
- **Frontend Mobile** : Gratuit
- **Base de données PostgreSQL** : Gratuit (90 jours), puis 7\$/mois
- **Total** : Gratuit pendant 90 jours, puis 7\$/mois

Étapes de Déploiement

1. Préparation du Code

```
# Assurez-vous que tous les fichiers sont prêts
./deploy.sh # Script de vérification inclus
```

2. Déploiement du Backend

Sur Render.com : 1. Créer un nouveau "Web Service" 2. Connecter votre repository Git
3. Configuration : - **Root Directory** : `backend` - **Build Command** : `pip install -r requirements.txt` - **Start Command** : `gunicorn app:app` - **Environment** : Python 3.11

Variables d'environnement :

```
DATABASE_URL=<fournie automatiquement>
STRIPE_SECRET_KEY=sk_live_...
STRIPE_WEBHOOK_SECRET=whsec_...
```

3. Déploiement de la Base de Données

Sur Render.com : 1. Créer une nouvelle "PostgreSQL Database" 2. Copier l'URL de connexion 3. L'ajouter automatiquement au backend

4. Déploiement du Frontend Client

Sur Render.com : 1. Créer un nouveau "Static Site" 2. Connecter votre repository Git 3. Configuration : - **Root Directory :** `food-delivery-client` - **Build Command :** `pnpm install && pnpm run build` - **Publish Directory :** `dist`

5. Déploiement du Frontend Mobile

Sur Render.com : 1. Créer un nouveau "Static Site" 2. Connecter votre repository Git 3. Configuration : - **Root Directory :** `restaurant-manager-mobile` - **Build Command :** `pnpm install && pnpm run build` - **Publish Directory :** `dist`

6. Configuration Post-Déploiement

Stripe Webhooks : 1. Dans le tableau de bord Stripe 2. Ajouter l'endpoint : `https://votre-backend.onrender.com/stripe/webhook` 3. Sélectionner les événements requis 4. Copier le secret webhook dans les variables d'environnement

Test de Production : 1. Vérifier l'accès aux applications 2. Tester l'inscription/connexion 3. Tester le processus d'abonnement 4. Vérifier les webhooks Stripe

Alternatives de Déploiement

Vercel (Frontend uniquement)

- Excellent pour les applications React
- Déploiement automatique via Git
- CDN global inclus

Heroku

- Plus cher que Render
- Excellente documentation

- Add-ons nombreux

DigitalOcean App Platform

- Prix compétitifs
 - Bonne performance
 - Interface simple
-

API Documentation

Format des Réponses

Succès

```
{
  "success": true,
  "data": { ... },
  "message": "Operation successful"
}
```

Erreur

```
{
  "success": false,
  "error": "Error message",
  "code": "ERROR_CODE"
}
```

Authentication

Toutes les routes protégées nécessitent un token JWT dans l'en-tête :

```
Authorization: Bearer <token>
```

Endpoints Détaillés

POST /register

Description : Inscription d'un nouvel utilisateur

Body :

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "role": "client|restaurateur"
}
```

Réponse :

```
{
  "success": true,
  "data": {
    "user_id": 1,
    "token": "jwt_token_here"
  }
}
```

POST /login

Description : Connexion utilisateur

Body :

```
{
  "username": "string",
  "password": "string"
}
```

Réponse :

```
{
  "success": true,
  "data": {
    "user_id": 1,
    "role": "client",
    "token": "jwt_token_here"
  }
}
```

GET /restaurants

Description : Liste des restaurants actifs

Paramètres de requête : - `search` (optionnel) : Recherche par nom - `page` (optionnel) : Numéro de page (défaut: 1) - `limit` (optionnel) : Éléments par page (défaut: 10)

Réponse :

```
{
  "success": true,
  "data": {
    "restaurants": [
      {
        "id": 1,
        "name": "Restaurant Name",
        "address": "123 Street",
        "phone": "0123456789",
        "email": "contact@restaurant.com"
      }
    ],
    "total": 25,
    "page": 1,
    "pages": 3
  }
}
```

POST /create-checkout-session

Description : Créer une session Stripe Checkout

Body :

```
{
  "price_id": "price_stripe_id",
  "success_url": "https://yoursite.com/success",
  "cancel_url": "https://yoursite.com/cancel"
}
```

Réponse :

```
{
  "success": true,
  "data": {
    "checkout_url": "https://checkout.stripe.com/..."
  }
}
```

Intégration Stripe

Configuration des Produits

Plans d'Abonnement Recommandés

Basic (9.99€/mois) - Accès aux restaurants partenaires - Commandes illimitées - Support par email

Premium (19.99€/mois) - Tout du plan Basic - Livraison gratuite - Support prioritaire - Accès aux nouveautés en avant-première

Enterprise (39.99€/mois) - Tout du plan Premium - Gestionnaire de compte dédié - API d'intégration - Rapports avancés

Gestion des Webhooks

Événements Importants

customer.subscription.created

```
# Activer l'abonnement utilisateur
user.subscription_status = 'active'
user.subscription_id = subscription.id
```

customer.subscription.updated

```
# Mettre à jour le statut
if subscription.status == 'active':
    user.subscription_status = 'active'
elif subscription.status == 'canceled':
    user.subscription_status = 'canceled'
```

invoice.payment_failed

```
# Gérer l'échec de paiement
user.subscription_status = 'past_due'
# Envoyer notification à l'utilisateur
```

Sécurité

Validation des Webhooks

```
import stripe

def verify_webhook(payload, sig_header):
    try:
        event = stripe.Webhook.construct_event(
            payload, sig_header, webhook_secret
        )
        return event
    except ValueError:
        # Payload invalide
        return None
    except stripe.error.SignatureVerificationError:
        # Signature invalide
        return None
```

Gestion des Erreurs

```
try:
    stripe.Subscription.create(...)
except stripe.error.CardError as e:
    # Carte refusée
    pass
except stripe.error.RateLimitError as e:
    # Trop de requêtes
    pass
except stripe.error.InvalidRequestError as e:
    # Paramètres invalides
    pass
```

Maintenance et Support

Monitoring

Métriques Importantes

- Temps de réponse API
- Taux d'erreur
- Utilisation de la base de données
- Succès des paiements Stripe
- Activité utilisateur

Outils Recommandés

- **Render Metrics** : Monitoring intégré
- **Stripe Dashboard** : Suivi des paiements
- **PostgreSQL Logs** : Surveillance base de données

Sauvegarde

Base de Données

```
# Sauvegarde automatique Render (quotidienne)
# Sauvegarde manuelle :
pg_dump $`DATABASE_URL` > backup_`$(date +%Y%m%d)` .sql
```

Code Source

- Repository Git avec branches de production
- Tags pour les versions stables
- Documentation des releases

Mise à Jour

Processus de Déploiement

1. Tests en local
2. Déploiement sur branche de staging
3. Tests d'intégration
4. Déploiement en production
5. Monitoring post-déploiement

Rollback

```
# Render permet le rollback via l'interface web
# ou via l'API Render
```


Support Utilisateur

Canaux de Support

- Email : support@votre-plateforme.com
- Chat intégré (optionnel)
- FAQ et documentation utilisateur

Problèmes Courants

Problème de Connexion 1. Vérifier les identifiants 2. Réinitialiser le mot de passe 3. Vérifier le statut du compte

Problème de Paiement 1. Vérifier le statut Stripe 2. Contrôler les webhooks 3. Vérifier la carte de crédit

Performance Lente 1. Vérifier les métriques Render 2. Optimiser les requêtes base de données 3. Mettre en cache les données fréquentes

Évolutions Futures

Fonctionnalités Prévues

Court Terme (1-3 mois)

- Notifications push
- Système de notation des restaurants
- Programme de fidélité
- Chat client-restaurateur

Moyen Terme (3-6 mois)

- Application mobile native (iOS/Android)
- Géolocalisation et livraison
- Intégration avec des services de livraison

- Tableau de bord analytique avancé

Long Terme (6-12 mois)

- Intelligence artificielle pour recommandations
- Système de franchise
- API publique pour partenaires
- Expansion internationale

Optimisations Techniques

Performance

- Mise en cache Redis
- CDN pour les images
- Optimisation des requêtes SQL
- Compression des assets

Sécurité

- Authentification à deux facteurs
- Chiffrement des données sensibles
- Audit de sécurité régulier
- Conformité RGPD






Scalabilité

- Architecture microservices
 - Load balancing
 - Base de données distribuée
 - Containerisation Docker
-

Conclusion

Cette plateforme de livraison de nourriture offre une base solide pour un service commercial. L'architecture modulaire permet une évolution progressive et l'intégration Stripe assure une monétisation efficace.

Points Forts

-  Architecture moderne et scalable
-  Interface utilisateur intuitive
-  Intégration paiement robuste
-  Coût de déploiement minimal
-  Documentation complète

Prochaines Étapes

1. Déployer en production
2. Tester avec des utilisateurs réels
3. Collecter les retours
4. Itérer sur les fonctionnalités
5. Planifier les évolutions

Contact Technique Pour toute question technique ou support, référez-vous à cette documentation ou contactez l'équipe de développement.

Version de la Documentation : 1.0

Dernière Mise à Jour : 30 juin 2025

Auteur : Équipe de Développement Manus