Cahier des Charges

Application "Eat Fast"

Une solution de livraison de repas 100% camerounaise

Rédigé par : Équipe de développement Eat Fast
Date : 31 mai 2025
Version : 3.0
Statut : Document de référence

Table des matières

---

1. Introduction

1.1 Contexte

L'application **"Eat Fast"** est une solution de livraison de repas **100% camerounaise** qui vise à révolutionner le secteur de la restauration et de la livraison au Cameroun. Dans un contexte où la digitalisation des services prend de l'ampleur et où les habitudes de consommation évoluent, notre application se positionne comme la référence locale pour connecter les consommateurs aux restaurants camerounais.

Le marché camerounais présente des spécificités uniques :

- **Population urbaine croissante** : Plus de 58% de la population réside en zone urbaine
- **Pénétration mobile** : Taux de 47% en 2025 avec une croissance de 8% par an
- **Économie numérique émergente** : Forte adoption des services de mobile money
- **Richesse culinaire** : Diversité exceptionnelle des plats traditionnels camerounais
- **Jeunesse entrepreneuriale** : 65% de la population a moins de 25 ans

1.2 Objectifs du projet

Eat Fast poursuit des objectifs ambitieux et mesurables :

Objectifs techniques :

1. Développer une application web responsive avec React.js + Vite
2. Construire une API robuste avec Django REST Framework
3. Assurer une livraison garantie en **30 minutes maximum**
4. Maintenir un service disponible **20h/24**

Objectifs business :

1. Atteindre **200+ restaurants partenaires** dans les 8 premiers mois
2. Fidéliser **15,000+ utilisateurs actifs** la première année
3. Couvrir **5 grandes villes camerounaises** en 18 mois
4. Atteindre la rentabilité opérationnelle en **24 mois**

Objectifs sociaux :

1. Créer **500+ emplois directs** pour les livreurs
2. Valoriser la gastronomie camerounaise traditionnelle
3. Contribuer à la digitalisation des PME locales

1.3 Types d'utilisateurs

L'application **"Eat Fast"** s'articule autour de **cinq (5) types d'utilisateurs** :

Acteur	Description
Client	Utilisateur final de l'application, il passe des commandes, consulte les menus et suit ses livraisons.
Livreur	Responsable de la livraison des commandes depuis les restaurants jusqu'aux clients.
Gestionnaire de restaurant	Administre le menu, les prix, la disponibilité des plats, et gère les commandes entrantes.
Agent de Support (Service Assistance)	Nouveau nom pour « Call Services ». Cet acteur gère toutes les préoccupations : appels, retards, erreurs de livraison, litiges clients, etc.
Administrateur	Supervise la plateforme entière, valide les inscriptions, contrôle les données, statistiques, et modère les utilisateurs.

1.4 Proposition de valeur unique

"Eat Fast - Saveur locale, service express"

- **Pour les clients** : Accès privilégié à l'authenticité culinaire camerounaise avec une livraison ultra-rapide
- **Pour les restaurants** : Visibilité maximale et croissance du chiffre d'affaires sans investissement initial
- **Pour les livreurs** : Emploi décent avec rémunération équitable et formation gratuite
- **Pour l'économie** : Boost de l'entrepreneuriat local et valorisation du patrimoine culinaire

---

2. Analyse de l'existant

2.1 Situation actuelle au Cameroun

Le paysage de la livraison de repas au Cameroun est caractérisé par :

Méthodes traditionnelles dominantes :

- Commandes téléphoniques directes aux restaurants
- Services de livraison artisanaux et non optimisés
- Paiement exclusivement en espèces
- Délais imprévisibles (1h à 2h30)
- Couverture géographique limitée

Défis identifiés :

- Absence de centralisation des offres
- Manque de traçabilité des commandes
- Inexistence de programmes de fidélisation
- Faible visibilité pour les petits restaurants
- Difficulté d'expansion pour les restaurateurs

2.2 Analyse concurrentielle

Concurrent	Forces	Faiblesses	Part de marché
Services individuels	Relation directe, connaissance client	Offre limitée, pas de tech	75%
JumiaFood	Notoriété Jumia	Service discontinu, cher	12%
Glovo	Présence internationale	Peu adapté au marché local	8%
Initiatives locales	Connaissance du terrain	Couverture limitée, tech basique	5%

2.3 Opportunités de marché

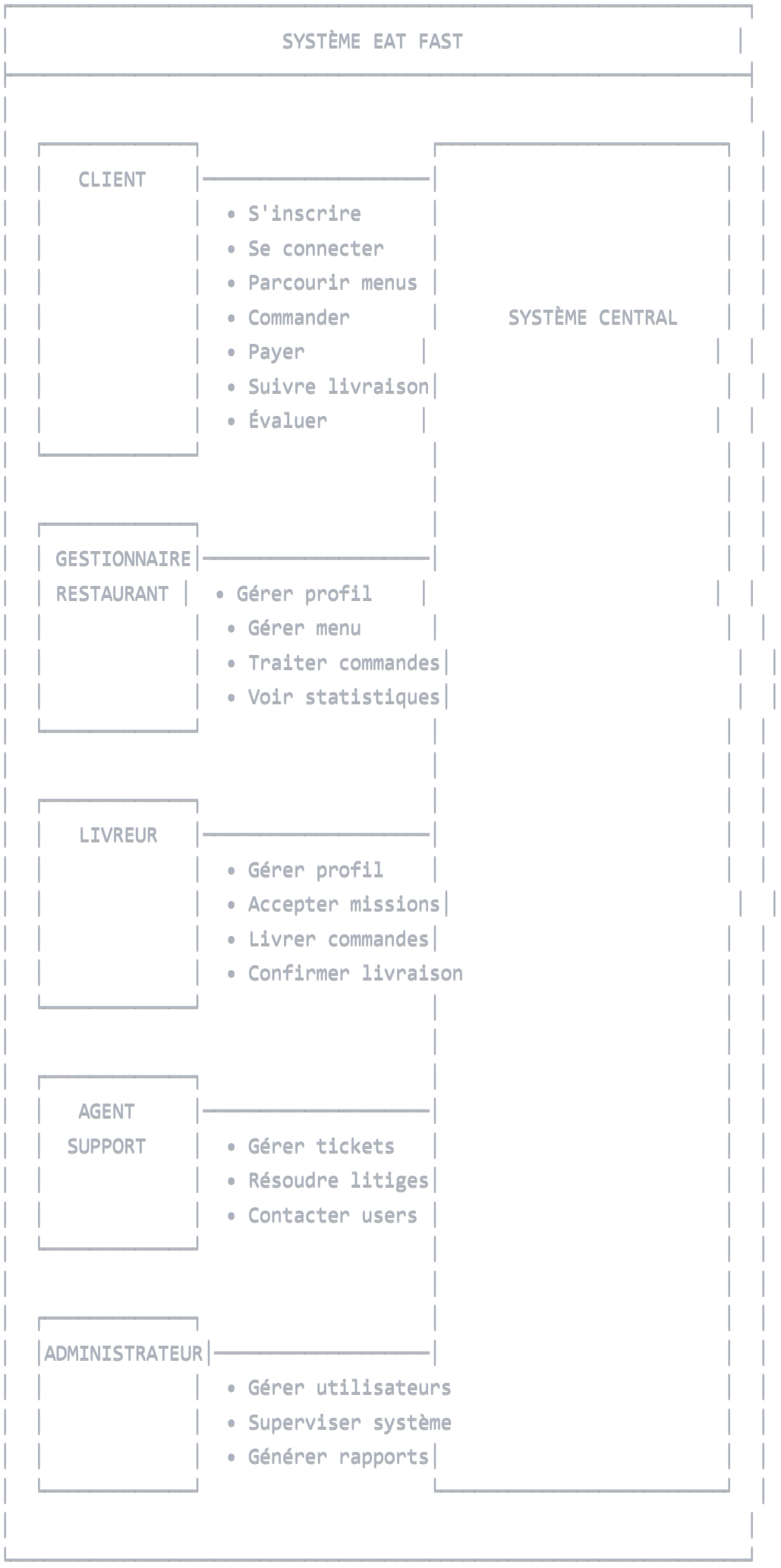
Niches inexploitées :

- Valorisation des plats traditionnels camerounais
- Service adapté aux zones périurbaines
- Intégration optimale du mobile money
- Support multilingue (français, anglais, langues locales)
- Partenariats avec l'économie informelle

---

3. Analyse des besoins

3.1 Diagramme de cas d'utilisation


SYSTÈME EAT FAST
CLIENT
• S'inscrire
• Se connecter
• Parcourir menus
• Commander
• Payer
• Suivre livraison
• Évaluer
GESTIONNAIRE RESTAURANT
• Gérer profil
• Gérer menu
• Traiter commandes
• Voir statistiques
LIVREUR
• Gérer profil
• Accepter missions
• Livrer commandes
• Confirmer livraison
AGENT SUPPORT
• Gérer tickets
• Résoudre litiges
• Contacter users
ADMINISTRATEUR
• Gérer utilisateurs
• Superviser système
• Générer rapports
SYSTÈME CENTRAL

3.2 Exigences fonctionnelles détaillées

Pour les Clients :

- **Gestion de compte** : Inscription/connexion sécurisée avec vérification SMS
- **Recherche avancée** : Filtrage par cuisine, prix, délai, note, proximité
- **Commande personnalisée** : Options de customisation des plats traditionnels
- **Paiement multiple** : Mobile Money (MTN, Orange), cartes, espèces
- **Suivi temps réel** : Géolocalisation du livreur avec ETA précis
- **Programme fidélité** : Points, niveaux, récompenses exclusives
- **Partage social** : Commandes groupées, partage sur réseaux sociaux

Pour les Gestionnaires de Restaurant :

- **Profil riche** : Photos, histoire du restaurant, spécialités
- **Gestion menu dynamique** : Disponibilité temps réel, prix variables
- **Analytics avancées** : Tendances, prévisions, optimisation des ventes
- **Communication directe** : Chat avec clients et livreurs
- **Promotions ciblées** : Offres personnalisées selon les données clients

Pour les Livreurs :

- **Système d'attribution intelligent** : Algorithme d'optimisation des trajets
- **Formation intégrée** : Modules sur l'hygiène alimentaire, service client
- **Rémunération transparente** : Calcul détaillé des gains, bonus performance
- **Équipement fourni** : Sacs isothermes, équipements de sécurité
- **Assurance couverture** : Protection en cas d'accident de travail

Pour les Agents de Support :

- **Dashboard unifié** : Vue 360° des incidents en cours
- **Outils de communication** : Appels, SMS, chat, email
- **Base de connaissances** : FAQ dynamique, procédures standard
- **Escalade automatique** : Routage des problèmes complexes
- **Reporting détaillé** : Métriques de satisfaction, temps de résolution

Pour les Administrateurs :

- **Supervision temps réel** : Monitoring de tous les indicateurs clés
- **Gestion des utilisateurs** : Validation, suspension, statistiques
- **Analytics business** : Revenus, commissions, prévisions
- **Contrôle qualité** : Modération des avis, vérification des restaurants
- **Configuration système** : Paramètres globaux, tarifications

3.3 Exigences non fonctionnelles

Performance :

- **Temps de réponse** : < 2 secondes pour 95% des requêtes
- **Capacité** : Support de 50,000 utilisateurs simultanés
- **Disponibilité** : 99.9% de uptime avec monitoring 24/7
- **Optimisation mobile** : Fonctionnement optimal sur 2G/3G

Sécurité :

- **Chiffrement** : HTTPS/TLS 1.3 pour toutes les communications
- **Authentification** : OAuth 2.0 + JWT avec refresh tokens
- **Paiements** : Conformité PCI DSS niveau 1
- **Données personnelles** : Conformité RGPD et lois locales
- **Audit** : Logs complets et traçabilité des actions

Évolutivité :

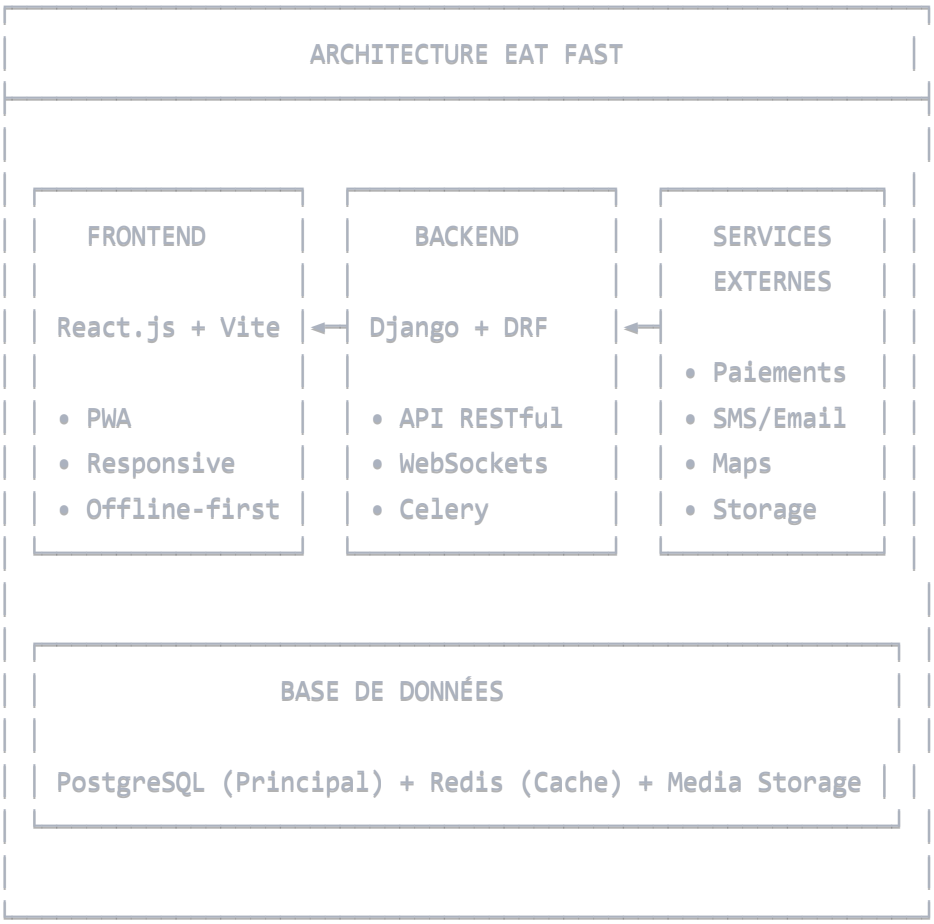
- **Architecture modulaire** : Microservices pour composants critiques
- **API versioning** : Compatibilité ascendante garantie
- **Auto-scaling** : Adaptation automatique à la charge
- **Multi-tenant** : Support de plusieurs villes/pays

---

4. Solution proposée

4.1 Vision architecturale

"Eat Fast" adopte une architecture moderne et évolutive :



4.2 Stack technologique

Frontend (React.js + Vite)

```javascript
// Configuration principale
{
  "framework": "React 18.2+",
  "bundler": "Vite 4.0+",
  "styling": "Tailwind CSS 3.0+",
  "state": "Redux Toolkit + RTK Query",
  "routing": "React Router v6",
  "forms": "React Hook Form + Zod",
  "maps": "Leaflet + OpenStreetMap",
  "charts": "Chart.js",
  "notifications": "React Toastify",
  "PWA": "Vite PWA Plugin"
}
```

Backend (Django)

python

```python
# Requirements principales
DJANGO_VERSION = "4.2 LTS"
DJANGO_REST_FRAMEWORK = "3.14+"
CELERY = "5.3+"  # Tâches asynchrones
CHANNELS = "4.0+"  # WebSockets
PILLOW = "10.0+"  # Traitement d'images
GUNICORN = "21.0+"  # Serveur WSGI
REDIS = "5.0+"  # Cache et broker
POSTGRESQL = "15+"  # Base de données
```

4.3 Innovations spécifiques au Cameroun

1. **Système de paiement hybride**

- Intégration native MTN Mobile Money et Orange Money
- Support des micropaiements (à partir de 100 FCFA)
- Paiement fractionné pour commandes groupées
- Portefeuille virtuel avec recharge automatique

2. **Géolocalisation adaptée**

- Fonctionnement avec adresses informelles
- Points de repère visuels (monuments, marchés)
- Navigation offline avec cartes préchargées
- Instructions vocales en langues locales

3. **Mode économie de données**

- Compression intelligente des images
- Mise en cache agressive
- Mode hors ligne pour consultation des menus
- Synchronisation différée des commandes

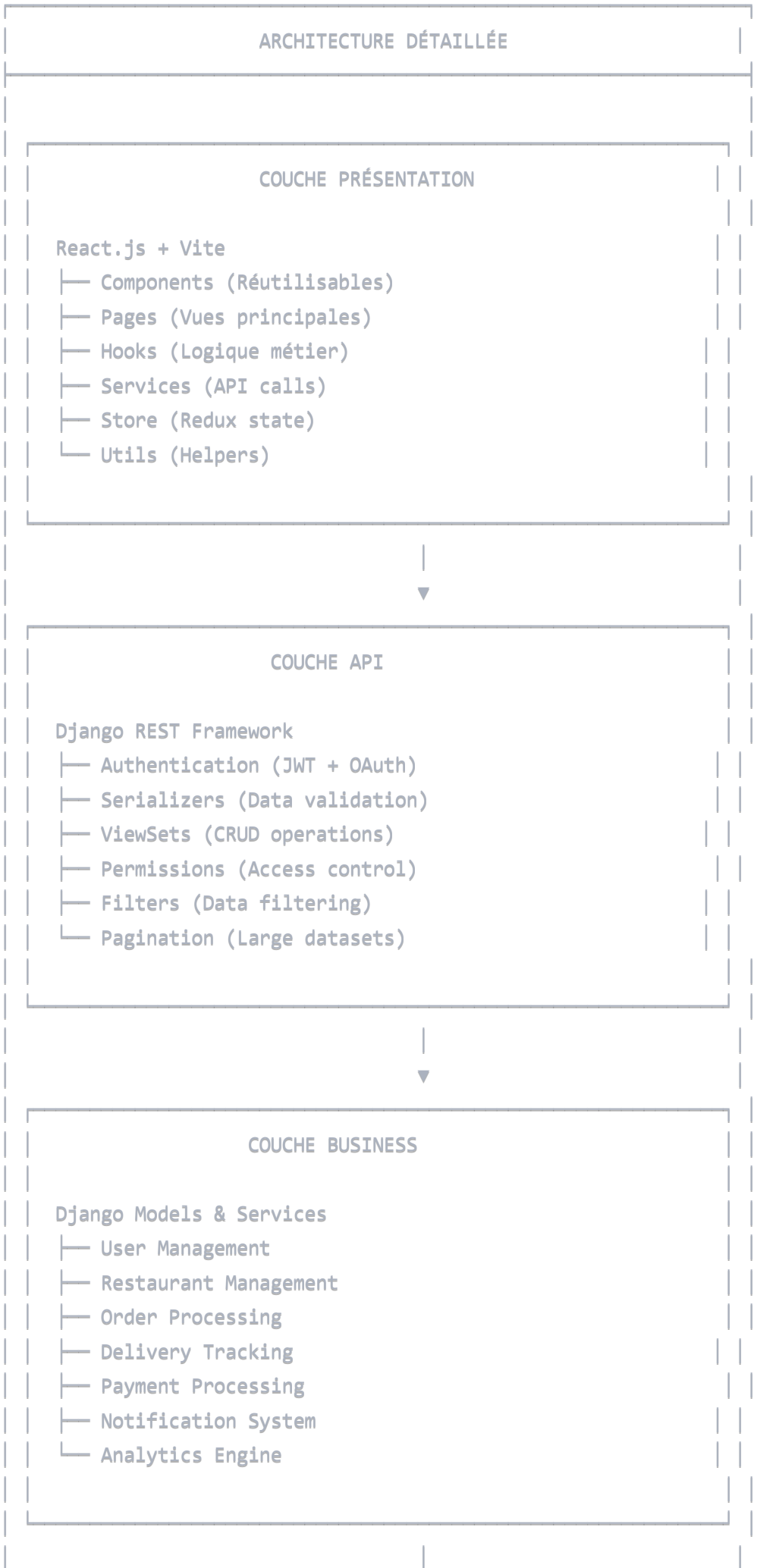
4. **Valorisation culturelle**

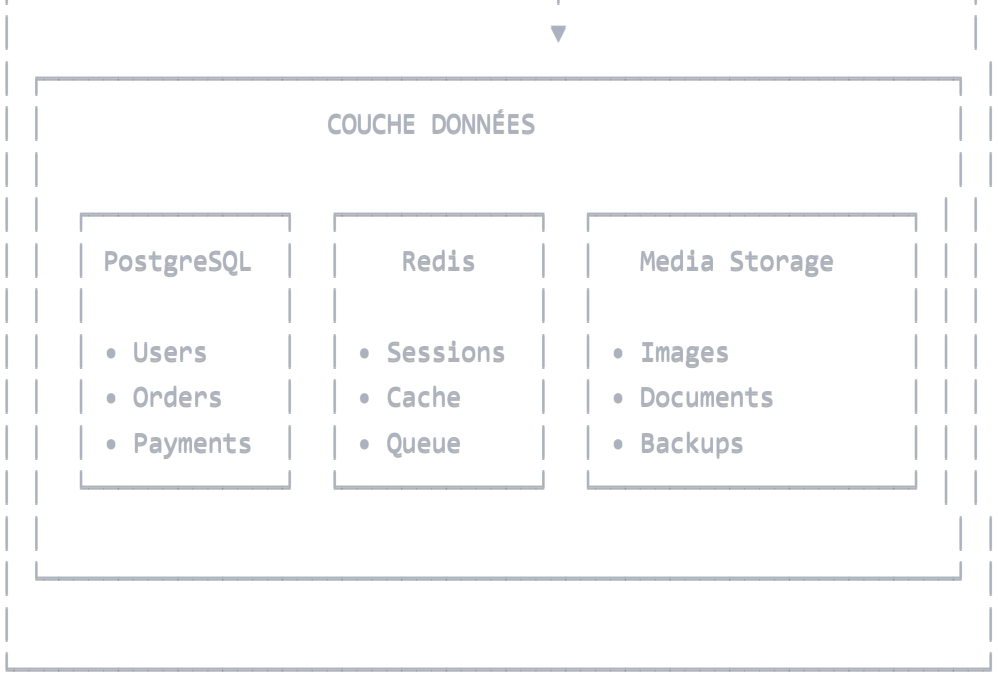
- Catégorisation par régions culinaires
- Histoire et origine des plats traditionnels
- Calendrier des festivités avec menus spéciaux
- Promotion des producteurs locaux

---

5. Architecture technique

5.1 Architecture logicielle détaillée


ARCHITECTURE DÉTAILLÉE
COUCHE PRÉSENTATION
React.js + Vite
├── Components (Réutilisables)
├── Pages (Vues principales)
├── Hooks (Logique métier)
├── Services (API calls)
├── Store (Redux state)
└── Utils (Helpers)
COUCHE API
Django REST Framework
├── Authentication (JWT + OAuth)
├── Serializers (Data validation)
├── ViewSets (CRUD operations)
├── Permissions (Access control)
├── Filters (Data filtering)
└── Pagination (Large datasets)
COUCHE BUSINESS
Django Models & Services
├── User Management
├── Restaurant Management
├── Order Processing
├── Delivery Tracking
├── Payment Processing
├── Notification System
└── Analytics Engine



5.2 Modèle de données

python

```python
# Modèles Django principaux

class User(AbstractUser):
    phone_number = models.CharField(max_length=15, unique=True)
    user_type = models.CharField(max_length=20, choices=USER_TYPES)
    is_verified = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

class Restaurant(models.Model):
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField()
    cuisine_type = models.CharField(max_length=50)
    address = models.TextField()
    phone = models.CharField(max_length=15)
    is_active = models.BooleanField(default=True)
    rating = models.DecimalField(max_digits=3, decimal_places=2)

class MenuItem(models.Model):
    restaurant = models.ForeignKey(Restaurant, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    category = models.CharField(max_length=50)
    is_available = models.BooleanField(default=True)
    image = models.ImageField(upload_to='menu_items/')

class Order(models.Model):
    customer = models.ForeignKey(User, on_delete=models.CASCADE)
    restaurant = models.ForeignKey(Restaurant, on_delete=models.CASCADE)
    items = models.ManyToManyField(MenuItem, through='OrderItem')
    status = models.CharField(max_length=20, choices=ORDER_STATUS)
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
    delivery_address = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    delivered_at = models.DateTimeField(null=True, blank=True)

class Delivery(models.Model):
    order = models.OneToOneField(Order, on_delete=models.CASCADE)
    driver = models.ForeignKey(User, on_delete=models.CASCADE)
    pickup_time = models.DateTimeField(null=True, blank=True)
    delivery_time = models.DateTimeField(null=True, blank=True)
    status = models.CharField(max_length=20, choices=DELIVERY_STATUS)
    tracking_url = models.URLField(blank=True)
```

5.3 API Design

Endpoints principaux

python

```
# API REST structure
/api/v1/
├── auth/
│   ├── register/          # POST - Inscription
│   ├── login/             # POST - Connexion
│   ├── logout/            # POST - Déconnexion
│   ├── refresh/           # POST - Refresh token
│   └── verify-phone/      # POST - Vérification téléphone
├── users/
│   ├── profile/           # GET, PUT - Profil utilisateur
│   ├── addresses/         # GET, POST - Adresses
│   └── preferences/       # GET, PUT - Préférences
├── restaurants/
│   ├── /                  # GET, POST - Liste/Création
│   ├── {id}/              # GET, PUT, DELETE - Détail
│   ├── {id}/menu/         # GET - Menu du restaurant
│   ├── search/            # GET - Recherche avancée
│   └── nearby/            # GET - Restaurants à proximité
├── orders/
│   ├── /                  # GET, POST - Commandes
│   ├── {id}/              # GET, PUT - Détail commande
│   ├── {id}/track/        # GET - Suivi temps réel
│   └── history/           # GET - Historique
├── payments/
│   ├── methods/           # GET, POST - Méthodes de paiement
│   ├── process/           # POST - Traitement paiement
│   └── webhooks/          # POST - Callbacks paiement
└── delivery/
    ├── available/         # GET - Missions disponibles
    ├── accept/            # POST - Accepter mission
    ├── update-location/   # POST - Mise à jour position
    └── complete/          # POST - Confirmer livraison
```

---

6. Description détaillée des fonctionnalités

6.1 Interface Client

Authentification et profil

```javascript
// Fonctionnalités d'authentification
const authFeatures = {
  registration: {
    methods: ['email', 'phone', 'social'],
    verification: 'SMS OTP',
    validation: 'real-time'
  },
  login: {
    methods: ['email/phone + password', 'biometric', 'social'],
    security: '2FA optional',
    session: 'persistent with refresh'
  },
  profile: {
    management: 'complete profile editing',
    addresses: 'multiple delivery addresses',
    preferences: 'cuisine, dietary, budget'
  }
}
```

Recherche et découverte

- **Recherche intelligente** : Autocomplete avec suggestions
- **Filtres avancés** : Prix, temps de livraison, note, type de cuisine
- **Géolocalisation** : Restaurants dans un rayon personnalisable
- **Recommandations** : IA basée sur l'historique et les préférences
- **Mode découverte** : Suggestions de nouveaux plats/restaurants

Processus de commande

1. **Sélection restaurant** : Visualisation complète (photos, avis, temps)
2. **Composition panier** : Personnalisation des plats, options
3. **Validation** : Récapitulatif détaillé, codes promo
4. **Paiement** : Multiple choix, sauvegarde sécurisée
5. **Confirmation** : Estimation précise du temps de livraison

Suivi temps réel

- **Tracking visuel** : Carte interactive avec position du livreur
- **Notifications push** : Chaque étape du processus
- **Communication** : Chat direct avec restaurant/livreur
- **ETA dynamique** : Mise à jour en temps réel selon trafic

6.2 Interface Gestionnaire de Restaurant

Dashboard principal

```javascript
// Vue d'ensemble temps réel
const restaurantDashboard = {
  metrics: {
    todaysOrders: 'Nombre et CA du jour',
    avgPreparationTime: 'Temps moyen de préparation',
    customerSatisfaction: 'Note moyenne et commentaires',
    popularItems: 'Top 5 des plats commandés'
  },
  alerts: {
    newOrders: 'Notifications sonores',
    lowStock: 'Articles bientôt épuisés',
    delays: 'Retards de préparation',
    reviews: 'Nouveaux avis clients'
  }
}
```

Gestion du menu

- **Interface drag & drop** : Organisation facile des catégories
- **Gestion des stocks** : Disponibilité temps réel des plats
- **Prix dynamiques** : Modulation selon demande/heure/jour
- **Photos professionnelles** : Upload optimisé, compression automatique
- **Descriptions riches** : Ingrédients, allergènes, temps de préparation

Gestion des commandes

- **File d'attente intelligente** : Priorisation automatique
- **Temps de préparation** : Estimation et tracking
- **Communication** : Chat avec clients et livreurs
- **Historique détaillé** : Analyse des performances

6.3 Interface Livreur

Système de missions

```python
# Algorithme d'attribution des livraisons
class DeliveryMatcher:
    def assign_delivery(self, order, available_drivers):
        factors = {
            'distance': 0.4,      # Distance au restaurant
            'rating': 0.2,        # Note du livreur
            'completion_rate': 0.2, # Taux de réussite
            'current_load': 0.2    # Charge actuelle
        }
        return optimize_assignment(order, drivers, factors)
```

Navigation optimisée

- **Itinéraires intelligents** : Évitement du trafic en temps réel
- **Multi-livraisons** : Optimisation pour plusieurs commandes
- **Points de repère** : Navigation avec landmarks locaux
- **Mode offline** : Cartes préchargées pour zones sans connexion

Gestion des revenus

- **Transparence totale** : Détail de chaque gain
- **Paiement flexible** : Journalier, hebdomadaire ou mensuel
- **Bonus performance** : Primes pour excellents services
- **Objectifs gamifiés** : Défis avec récompenses

6.4 Interface Agent de Support