DOSSIER PROJET

Titre Professionnel Développeur Web et Web Mobile (DWWM)

Jamesy MUKUNA MUKENKETAYI

Session Juin-Juillet 2026

SOMMAIRE

Candidat :
Jamesy MUKUNA MUKENKETAYI
Email : yvesnet9@gmail.com
Téléphone : +32465188199

Formation :
Graduate Développeur web full stack 2023-2029
STUDI (DIGITAL CAMPUS LIVE) - Paris
Session : Juin-Juillet 2026

1. INTRODUCTION

1.1 Contexte du Projet

Dans le cadre de ma formation au Titre Professionnel Développeur Web et Web Mobile (DWWM) chez STUDI, j'ai réalisé un projet complet de développement d'une application web pour un service de traiteur événementiel nommé "Vite & Gourmand".

Ce projet répond à un besoin réel de digitalisation d'une entreprise de traiteur qui souhaite offrir à ses clients la possibilité de consulter et commander des menus en ligne, tout en facilitant la gestion opérationnelle pour l'équipe.

1.2 Objectifs du Projet

Les objectifs principaux de ce projet sont :

- **Permettre aux clients** de consulter les menus disponibles et passer des commandes en ligne
- **Faciliter la gestion des commandes** pour l'équipe (validation, suivi, livraison)
- **Offrir un outil d'administration complet** pour gérer les menus, plats et utilisateurs
- **Assurer la sécurité des données** et la conformité RGPD
- **Déployer l'application en production** sur un serveur sécurisé avec HTTPS

1.3 Durée et Organisation

Durée du projet : 12 jours (du 6 février au 18 février 2025)

Méthode : Développement itératif en suivant les fonctionnalités F01 à F12

Résultat : Application fonctionnelle déployée en production sur https://vite-gourmand.fr

2. ANALYSE DES BESOINS

2.1 Identification des Acteurs

J'ai identifié quatre types d'acteurs principaux avec des besoins et des permissions différents :

2.1.1 Visiteur (Non connecté)

- Consultation des menus disponibles
- Inscription pour créer un compte
- Accès aux pages RGPD (politique de confidentialité, mentions légales)

2.1.2 Client (Utilisateur connecté)

- Toutes les fonctionnalités du visiteur
- Passage de commandes en ligne

- Consultation de l'historique de ses commandes
- Annulation de commandes en attente
- Laisser des avis après livraison
- Export de ses données personnelles (RGPD)
- Suppression de son compte (droit à l'oubli)

3.1.3 Employé

- Toutes les fonctionnalités du client
- Consultation de toutes les commandes
- Modification du statut des commandes (validation, préparation, livraison)
- Validation et rejet des avis clients
- Création et modification de menus

3.1.4 Administrateur

- Toutes les fonctionnalités de l'employé
- Suppression de menus
- Gestion complète des plats (CRUD)
- Gestion des allergènes
- Gestion des utilisateurs (activation/désactivation)
- Suppression définitive d'avis

3.2 2.2 Cas d'Usage Principaux

Au total, j'ai identifié et implémenté **29 cas d'usage** couvrant l'ensemble des besoins fonctionnels.

3.2.1 Tableau Complet des Cas d'Usage (29 cas)

ID	Acteur	Cas d'Usage	Description	Priorité
UC01	Visiteur	Consulter les menus	Parcourir le catalogue avec filtres par thème et régime	HAUTE
UC02	Visiteur	Consulter le détail d'un menu	Voir la composition, allergènes, prix, avis	HAUTE
UC03	Visiteur	S'inscrire	Créer un compte avec validation stricte et consentement RGPD	HAUTE
UC04	Visiteur	Consulter politique RGPD	Lire la politique de confidentialité	MOYENNE
UC05	Visiteur	Consulter mentions légales	Accéder aux mentions légales	BASSE
UC06	Client	Se connecter	S'authentifier avec email et mot de passe	HAUTE
UC07	Client	Se déconnecter	Terminer sa session	HAUTE

ID	Acteur	Cas d'Usage	Description	Priorité
UC08	Client	Passer une commande	Commander un menu avec date, adresse et quantité	HAUTE
UC09	Client	Consulter ses commandes	Voir l'historique avec statuts	HAUTE
UC10	Client	Consulter le détail d'une commande	Voir toutes les informations d'une commande	MOYENNE
UC11	Client	Annuler une commande	Annuler si statut "en attente"	MOYENNE
UC12	Client	Laisser un avis	Noter et commenter après livraison	MOYENNE
UC13	Client	Consulter ses avis	Voir les avis laissés	BASSE
UC14	Client	Exporter ses données	Télécharger au format JSON (RGPD)	HAUTE
UC15	Client	Supprimer son compte	Droit à l'oubli RGPD	HAUTE
UC16	Client	Gérer newsletter	Opt-in/opt-out	BASSE
UC17	Client	Contacter le service	Envoyer un message	BASSE
UC18	Employé	Consulter toutes les commandes	Vue d'ensemble avec filtres	HAUTE
UC19	Employé	Modifier statut commande	Workflow de validation	HAUTE
UC20	Employé	Consulter détail commande	Voir infos complètes	MOYENNE
UC21	Employé	Consulter avis en attente	Liste des avis à modérer	HAUTE
UC22	Employé	Valider un avis	Publier l'avis	HAUTE
UC23	Employé	Rejeter un avis	Masquer l'avis	HAUTE
UC24	Employé	Créer un menu	Nouveau menu avec plats	HAUTE
UC25	Employé	Modifier un menu	Éditer menu existant	HAUTE
UC26	Employé	Activer/désactiver menu	Toggle visibilité	MOYENNE
UC27	Admin	Supprimer un menu	Suppression définitive	MOYENNE
UC28	Admin	Gérer plats	CRUD complet sur plats	HAUTE
UC29	Admin	Gérer allergènes	CRUD complet sur allergènes	MOYENNE

ID	Acteur	Cas d'Usage	Description	Priorité
UC30	Admin	Gérer utilisateurs	Activer/désactiver comptes	MOYENNE

3.2.2 Description Détaillée des Cas d'Usage Critiques

3.2.2.1 UC08 : Passer une Commande (Priorité HAUTE)

Acteur principal : Client (utilisateur connecté)

Préconditions : - L'utilisateur est connecté - Le menu sélectionné est actif - Le stock est suffisant

Scénario nominal : 1. Le client consulte le détail d'un menu 2. Le client clique sur "Commander ce menu" 3. Le système affiche le formulaire de commande pré-rempli avec : - Adresse de livraison (celle du profil) - Date du jour comme date minimum 4. Le client saisit : - Date de livraison souhaitée (date future obligatoire) - Adresse de livraison (modifiable) - Quantité (validation : >= nb_personne_min du menu) - Instructions spéciales (optionnel) 5. Le système calcule et affiche le prix total en temps réel 6. Le client valide la commande 7. Le système vérifie : - La date est bien dans le futur - La quantité respecte le minimum - Le stock est suffisant 8. Le système crée la commande avec statut "en attente" 9. Le système décrémente le stock du menu 10. Le système affiche une confirmation avec le numéro de commande 11. Le système envoie un email de confirmation (future fonctionnalité)

Scénarios alternatifs :

3a. Date invalide (passée) : - Le système affiche une erreur "La date doit être dans le futur" - Retour à l'étape 4

3b. Quantité inférieure au minimum : - Le système affiche "Minimum X personnes pour ce menu" - Retour à l'étape 4

3c. Stock insuffisant : - Le système affiche "Stock insuffisant (disponible : X)" - Proposition d'autres menus similaires - Fin du cas d'usage

Postconditions : - La commande est créée avec statut "en attente" - Le stock du menu est mis à jour - Le client est redirigé vers "Mes commandes"

Règles métier : - Une commande ne peut être passée que pour une date future - La quantité doit respecter le nb_personne_min du menu - Le stock est vérifié et décrémenté de manière atomique - Le prix total est calculé : prix_base × quantité

Exigences non fonctionnelles : - Temps de réponse < 2 secondes - Calcul du prix en temps réel (JavaScript) - Validation côté client ET serveur

3.2.2.2 UC11 : Annuler une Commande (Priorité MOYENNE)

Acteur principal : Client

Préconditions : - L'utilisateur est connecté - L'utilisateur possède au moins une commande - La commande est au statut "en attente"

Scénario nominal : 1. Le client accède à "Mes commandes" 2. Le système affiche la liste des commandes du client 3. Le client clique sur "Annuler" pour une commande "en attente" 4. Le système affiche une modale de confirmation : - "Êtes-vous sûr de vouloir annuler cette commande ?" - Détails de la commande 5. Le client confirme l'annulation 6. Le système : - Change le statut à "annulee" - Réincrémente le stock du menu - Met à jour la date d'annulation 7. Le système affiche un message de succès 8. Le système envoie un email de confirmation d'annulation (future)

Scénarios alternatifs :

3a. Commande non annulable : - La commande n'est pas au statut "en attente" - Le bouton "Annuler" n'est pas affiché - Message : "Cette commande ne peut plus être annulée"

5a. Annulation échouée : - Erreur technique lors de l'annulation - Le système affiche "Une erreur s'est produite" - La commande reste inchangée - Le client peut réessayer

Postconditions : - La commande a le statut "annulee" - Le stock du menu est restauré - La commande reste visible dans l'historique

Règles métier : - Seules les commandes "en attente" peuvent être annulées - L'annulation restaure le stock atomiquement - La commande annulée reste dans l'historique (soft delete)

3.2.2.3 UC14 : Exporter ses Données RGPD (Priorité HAUTE)

Acteur principal : Client

Préconditions : - L'utilisateur est connecté

Scénario nominal : 1. Le client accède à "Mes Données" (menu profil) 2. Le système affiche les options RGPD 3. Le client clique sur "Télécharger mes données" 4. Le système génère un fichier JSON contenant : `json { "user": { "nom": "...", "prenom": "...", "email": "...", "gsm": "...", "adresse": "...", "role": "...", "created_at": "...", "consentement_rgpd": true, "newsletter": false }, "commandes": [{ "id": 123, "menu": "Menu Bio", "date_commande": "...", "date_livraison": "...", "statut": "...", "prix_total": 175.00 }], "avis": [{ "id": 45, "commande_id": 123, "note": 5, "commentaire": "...", "date": "..." }] }` 5. Le système déclenche le téléchargement 6. Le système enregistre dans les logs : export demandé par user X 7. Le fichier est téléchargé sur l'ordinateur du client

Postconditions : - L'utilisateur possède une copie de toutes ses données - L'action est tracée dans les logs

Règles métier : - Toutes les données personnelles doivent être incluses - Le format JSON assure la portabilité - Le mot de passe n'est jamais exporté - Les timestamps sont au format ISO 8601

3.2.2.4 UC19 : Modifier Statut Commande (Priorité HAUTE)

Acteur principal : Employé

Préconditions : - L'utilisateur est employé ou administrateur - Au moins une commande existe

Scénario nominal : 1. L'employé accède au dashboard employé 2. Le système affiche toutes les commandes 3. L'employé clique sur une commande 4. Le système affiche le détail avec : - Informations client - Détails du menu - Adresse de livraison - Statut actuel 5. L'employé clique sur "Changer statut" 6. Le système affiche les statuts possibles selon workflow : - En attente → Validée OU Annulée - Validée → En préparation - En préparation → Livrée 7. L'employé sélectionne le nouveau statut 8. Le système met à jour la commande 9. Le système affiche une confirmation 10. Le système envoie un email au client (future)

Workflow des Statuts :

```
En attente
    ├→ Validée ———→ En préparation ———→ Livrée
    └→ Annulée (par client ou employé)
```

Règles métier : - Workflow unidirectionnel (pas de retour arrière) - Seul statut "en attente" peut être annulé - "Livrée" = statut final - Chaque changement est horodaté

Postconditions : - Le statut de la commande est mis à jour - Un historique des changements est conservé - Le client voit le nouveau statut

4 3. MODÉLISATION DES DONNÉES

4.1 3.1 Modèle Conceptuel de Données (MCD)

Le modèle de données de l'application repose sur **13 tables principales** organisées de manière à respecter les formes normales et à assurer l'intégrité référentielle.

4.1.1 Entités Principales

- **users** : Utilisateurs de l'application (clients, employés, administrateurs)
- **menus** : Menus proposés par le traiteur
- **plats** : Plats composant les menus
- **allergenes** : Liste des allergènes
- **commandes** : Commandes passées par les clients
- **avis** : Avis laissés par les clients
- **contacts** : Messages de contact
- **menu_plat** : Table associative (N:N entre menus et plats)

- **plat_allergene** : Table associative (N:N entre plats et allergènes)

4.1.2 Relations Principales

- Un **utilisateur** peut avoir **plusieurs commandes** (1:N)
- Un **utilisateur** peut laisser **plusieurs avis** (1:N)
- Un **menu** peut être commandé **plusieurs fois** (1:N)
- Une **commande** peut avoir **un avis** (1:1)
- Un **menu** contient **plusieurs plats** (N:N via menu_plat)
- Un **plat** peut contenir **plusieurs allergènes** (N:N via plat_allergene)

4.2 3.2 Structure de la Table Users

```
users
├── id (BIGINT, PK)
├── nom (VARCHAR(100), NOT NULL)
├── prenom (VARCHAR(100), NOT NULL)
├── email (VARCHAR(255), UNIQUE, NOT NULL)
├── password (VARCHAR(255), NOT NULL)
├── gsm (VARCHAR(20), NOT NULL)
├── adresse (TEXT, NOT NULL)
├── role (ENUM: visiteur, utilisateur, employe, administrateur)
├── active (BOOLEAN, DEFAULT: true)
├── consentement_rgpd (BOOLEAN, DEFAULT: false)
├── date_consentement (TIMESTAMP)
├── newsletter (BOOLEAN, DEFAULT: false)
├── created_at (TIMESTAMP)
└── updated_at (TIMESTAMP)
```

4.3 3.3 Normalisation

Le modèle respecte la **3ème forme normale (3NF)** :

- **1NF** : Toutes les valeurs sont atomiques (pas de tableaux ou de listes)
- **2NF** : Toutes les colonnes dépendent de la clé primaire complète
- **3NF** : Pas de dépendance transitive entre colonnes

Exemples de normalisation : - Les allergènes sont dans une table séparée (pas un champ texte) - Les plats sont réutilisables dans plusieurs menus - Les relations N:N utilisent des tables associatives

4.4 3.4 Schéma de Base de Données Complet

4.4.1 Table users

```sql
CREATE TABLE users (
    id BIGSERIAL PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
```

```
    gsm VARCHAR(20) NOT NULL,
    adresse TEXT NOT NULL,
    role VARCHAR(50) NOT NULL DEFAULT 'utilisateur'
        CHECK (role IN ('visiteur', 'utilisateur', 'employe',
'administrateur')),
    active BOOLEAN DEFAULT TRUE,
    consentement_rgpd BOOLEAN DEFAULT FALSE,
    date_consentement TIMESTAMP,
    newsletter BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
CREATE INDEX idx_users_active ON users(active);
```

4.4.2 Table menus

```
CREATE TABLE menus (
    id BIGSERIAL PRIMARY KEY,
    titre VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    theme VARCHAR(100) NOT NULL
        CHECK (theme IN ('bio', 'végétarien', 'vegan',
'gastronomique',
                         'traditionnel', 'asiatique',
'méditerranéen')),
    regime VARCHAR(100) NOT NULL
        CHECK (regime IN ('normal', 'végétarien', 'vegan',
'sans_gluten',
                          'sans_lactose', 'halal', 'casher')),
    nb_personne_min INTEGER NOT NULL CHECK (nb_personne_min > 0),
    prix_base DECIMAL(10,2) NOT NULL CHECK (prix_base > 0),
    stock INTEGER NOT NULL DEFAULT 0 CHECK (stock >= 0),
    conditions TEXT,
    actif BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_menus_actif ON menus(actif);
CREATE INDEX idx_menus_theme ON menus(theme);
CREATE INDEX idx_menus_regime ON menus(regime);
CREATE INDEX idx_menus_prix ON menus(prix_base);
```

4.4.3 Table plats

```
CREATE TABLE plats (
    id BIGSERIAL PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
```

```
    categorie VARCHAR(100) NOT NULL
        CHECK (categorie IN ('entree', 'plat', 'dessert',
'accompagnement')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_plats_categorie ON plats(categorie);
```

4.4.4 Table allergenes

```
CREATE TABLE allergenes (
    id BIGSERIAL PRIMARY KEY,
    nom VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.4.5 Table commandes

```
CREATE TABLE commandes (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    menu_id BIGINT NOT NULL,
    date_commande TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    date_livraison DATE NOT NULL,
    adresse_livraison TEXT NOT NULL,
    quantite INTEGER NOT NULL CHECK (quantite > 0),
    prix_total DECIMAL(10,2) NOT NULL CHECK (prix_total > 0),
    statut VARCHAR(50) NOT NULL DEFAULT 'en_attente'
        CHECK (statut IN ('en_attente', 'validee', 'en_preparation',
                          'livree', 'annulee')),
    instructions TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (menu_id) REFERENCES menus(id) ON DELETE RESTRICT
);

CREATE INDEX idx_commandes_user ON commandes(user_id);
CREATE INDEX idx_commandes_menu ON commandes(menu_id);
CREATE INDEX idx_commandes_statut ON commandes(statut);
CREATE INDEX idx_commandes_date_livraison ON
commandes(date_livraison);
```

4.4.6 Table avis

```
CREATE TABLE avis (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT NOT NULL,
    commande_id BIGINT NOT NULL UNIQUE,
    note INTEGER NOT NULL CHECK (note >= 1 AND note <= 5),
```

```
    commentaire TEXT NOT NULL,
    statut VARCHAR(50) NOT NULL DEFAULT 'en_attente'
        CHECK (statut IN ('en_attente', 'valide', 'rejete')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (commande_id) REFERENCES commandes(id) ON DELETE
CASCADE
);

CREATE INDEX idx_avis_user ON avis(user_id);
CREATE INDEX idx_avis_commande ON avis(commande_id);
CREATE INDEX idx_avis_statut ON avis(statut);
CREATE INDEX idx_avis_note ON avis(note);
```

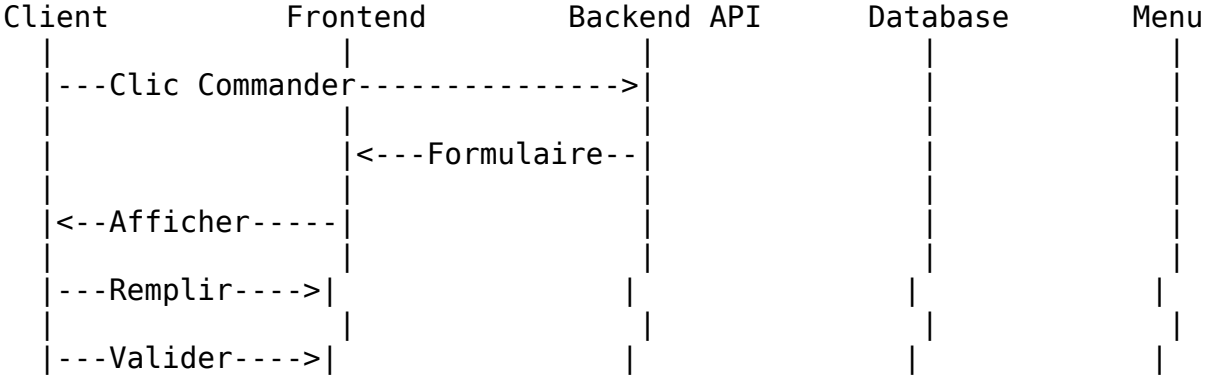
4.4.7 Tables Associatives

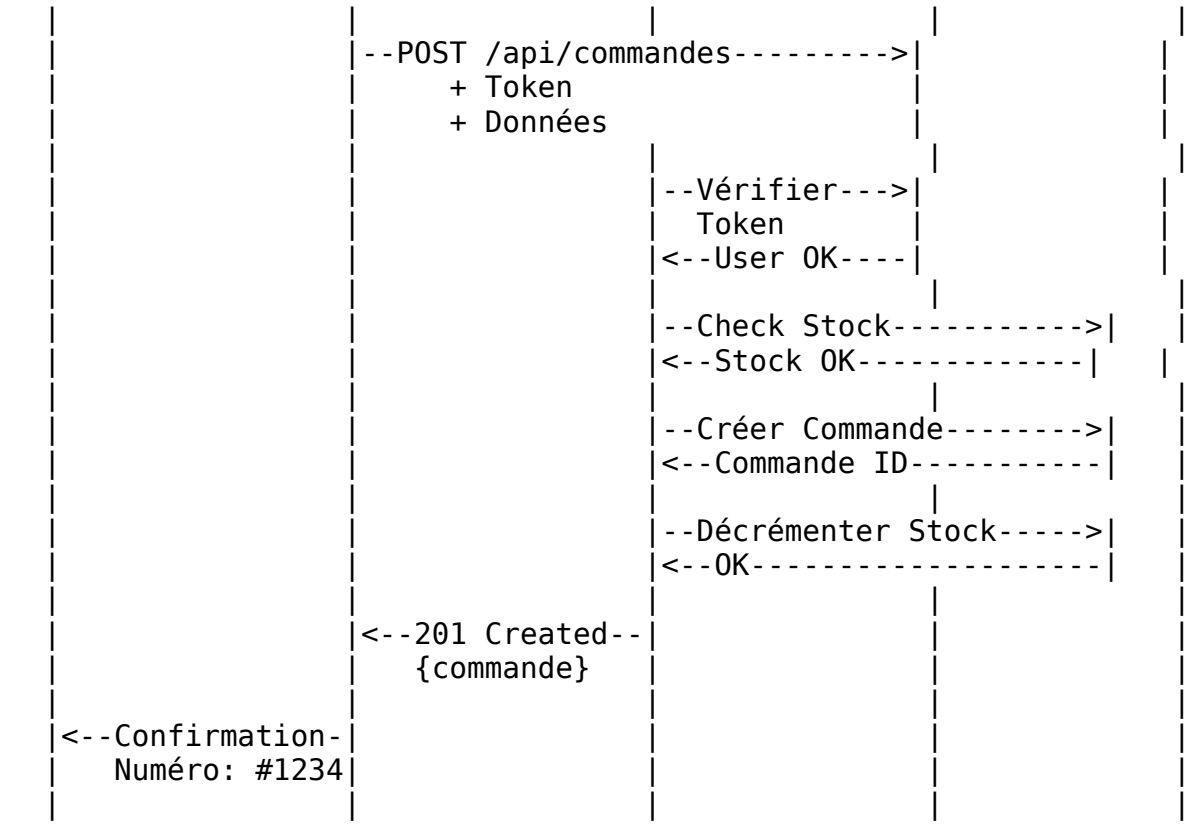
```
-- Relation Menu ↔ Plat (N:N)
CREATE TABLE menu_plat (
    menu_id BIGINT NOT NULL,
    plat_id BIGINT NOT NULL,
    PRIMARY KEY (menu_id, plat_id),
    FOREIGN KEY (menu_id) REFERENCES menus(id) ON DELETE CASCADE,
    FOREIGN KEY (plat_id) REFERENCES plats(id) ON DELETE CASCADE
);

-- Relation Plat ↔ Allergène (N:N)
CREATE TABLE plat_allergene (
    plat_id BIGINT NOT NULL,
    allergene_id BIGINT NOT NULL,
    PRIMARY KEY (plat_id, allergene_id),
    FOREIGN KEY (plat_id) REFERENCES plats(id) ON DELETE CASCADE,
    FOREIGN KEY (allergene_id) REFERENCES allergenes(id) ON DELETE
CASCADE
);
```

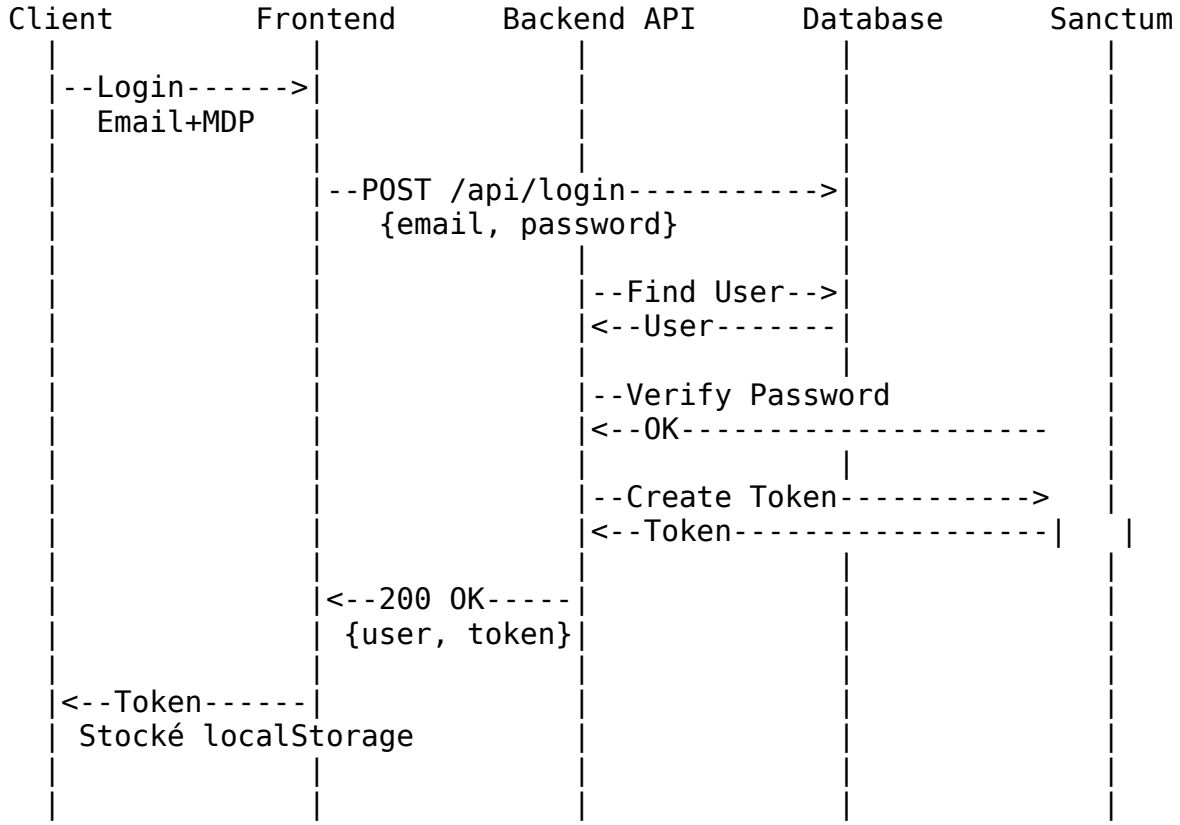
4.5 3.5 Diagrammes de Séquence

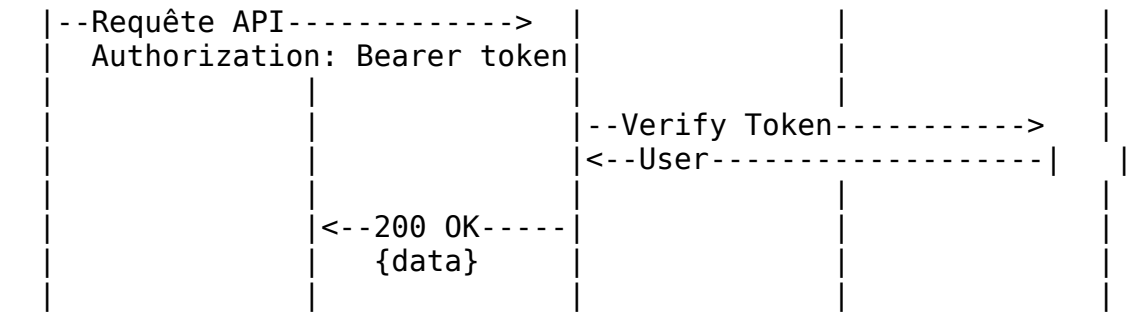
4.5.1 Diagramme : Passer une Commande



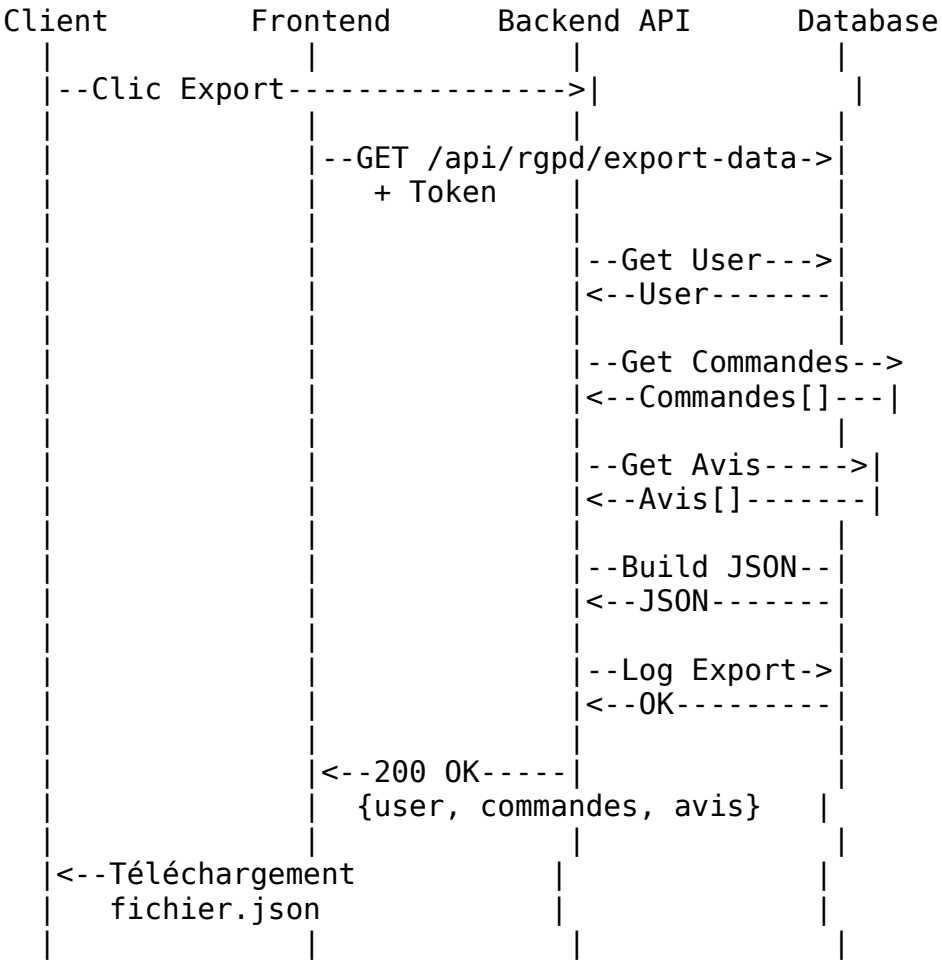


4.5.2 Diagramme : Authentification avec Token





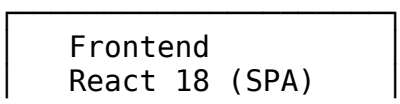
4.5.3 Diagramme : Export Données RGPD

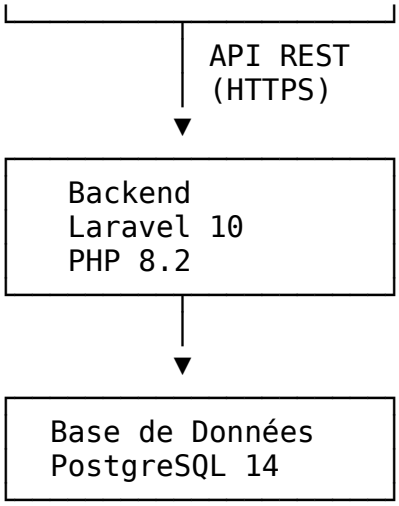


5 4. ARCHITECTURE TECHNIQUE

5.1 4.1 Vue d'Ensemble

J'ai opté pour une **architecture 3-tiers découplée** :





Serveur : Nginx sur VPS OVH (Ubuntu 22.04)

5.2 4.2 Choix Techniques Justifiés

5.2.1 Frontend : React 18

Justification : - Framework moderne et largement utilisé dans l'industrie - Composants réutilisables facilitant la maintenance - Écosystème riche (React Router, Axios, etc.) - Expérience utilisateur fluide (Single Page Application) - Performances optimales avec le Virtual DOM

Bibliothèques utilisées : - **React Router v6** : Navigation et gestion des routes - **Axios** : Communication HTTP avec le backend - **Context API** : Gestion d'état global pour l'authentification

5.2.2 Backend : Laravel 10

Justification : - Framework PHP robuste et mature - Eloquent ORM pour la gestion élégante de la base de données - Laravel Sanctum pour l'authentification API par tokens - Système de validation et de sécurité intégré - Documentation complète et communauté active

Composants Laravel utilisés : - **Sanctum** : Authentification API - **FormRequests** : Validation des données entrantes - **Policies** : Contrôle d'accès granulaire - **Migrations** : Gestion du schéma de base de données

5.2.3 Base de Données : PostgreSQL 14

Justification : - SGBD relationnel performant et fiable - Respect strict des standards SQL - Types de données riches (JSON, Arrays) - Excellent support des contraintes d'intégrité - Performances optimales sur les requêtes complexes

5.2.4 Serveur Web : Nginx

Justification : - Serveur web performant et léger - Excellent en tant que proxy inverse - Configuration flexible - Gestion native de HTTPS

5.3 4.3 Architecture Frontend

5.3.1 Structure du Projet React

```
frontend/
├── src/
│   ├── components/
│   │   ├── layout/
│   │   │   ├── Navbar.js
│   │   │   └── Footer.js
│   │   └── PrivateRoute.js
│   ├── contexts/
│   │   └── AuthContext.js
│   ├── pages/
│   │   ├── HomePage.js
│   │   ├── LoginPage.js
│   │   ├── MenusListPage.js
│   │   ├── CommanderPage.js
│   │   └── admin/
│   │       └── MenusAdminPage.js
│   ├── services/
│   │   ├── api.js
│   │   ├── authService.js
│   │   └── menuService.js
│   └── App.js
└── package.json
```

5.3.2 Routing

Les routes sont organisées de manière claire :

- **Routes publiques** : `/`, `/login`, `/register`, `/menus`
- **Routes protégées (client)** : `/commander/:id`, `/mes-commandes`
- **Routes protégées (employé)** : `/dashboard-employe`
- **Routes protégées (admin)** : `/admin/menus`, `/admin/plats`

5.3.3 Gestion de l'Authentification

L'authentification est gérée via :

1. **Context API** : Stockage de l'état utilisateur
2. **LocalStorage** : Persistance du token
3. **Intercepteurs Axios** : Ajout automatique du token aux requêtes
4. **PrivateRoute** : Protection des routes selon le rôle

5.4 4.4 Architecture Backend

5.4.1 Structure du Projet Laravel

```
backend/
├── app/
│   ├── Http/
│   │   ├── Controllers/
│   │   │   └── Api/
│   │   │       ├── AuthController.php
│   │   │       ├── MenuController.php
│   │   │       ├── CommandeController.php
│   │   │       ├── AvisController.php
│   │   │       ├── PlatController.php
│   │   │       ├── AllergeneController.php
│   │   │       └── RgpdController.php
│   │   ├── Requests/
│   │   │   ├── RegisterRequest.php
│   │   │   ├── LoginRequest.php
│   │   │   ├── StoreCommandeRequest.php
│   │   │   ├── StoreMenuRequest.php
│   │   │   ├── UpdateMenuRequest.php
│   │   │   ├── StoreAvisRequest.php
│   │   │   └── DeleteAccountRequest.php
│   │   ├── Middleware/
│   │   │   ├── SecurityHeaders.php
│   │   │   └── CheckRole.php
│   │   └── Kernel.php
│   ├── Models/
│   │   ├── User.php
│   │   ├── Menu.php
│   │   ├── Plat.php
│   │   ├── Allergene.php
│   │   ├── Commande.php
│   │   ├── Avis.php
│   │   └── Contact.php
│   ├── Policies/
│   │   ├── MenuPolicy.php
│   │   ├── PlatPolicy.php
│   │   ├── CommandePolicy.php
│   │   ├── AvisPolicy.php
│   │   └── AllergenePolicy.php
│   └── Providers/
│       ├── AuthServiceProvider.php
│       └── RouteServiceProvider.php
├── config/
│   ├── cors.php
│   ├── auth.php
│   └── sanctum.php
├── database/
│   ├── migrations/
```

```
│   │   ├── 2024_01_01_000000_create_users_table.php
│   │   ├── 2024_01_02_000000_create_menus_table.php
│   │   ├── 2024_01_03_000000_create_plats_table.php
│   │   ├── 2024_01_04_000000_create_allergenes_table.php
│   │   ├── 2024_01_05_000000_create_commandes_table.php
│   │   └── 2024_01_06_000000_create_avis_table.php
│   └── factories/
├── routes/
│   └── api.php
└── tests/
    ├── Feature/
    │   ├── AuthTest.php
    │   ├── MenuTest.php
    │   └── RgpdTest.php
    └── Unit/
```

5.4.2 Routes API Complètes (46 routes)

5.4.2.1 Routes Publiques (5 routes)

```
// Menus publics
GET    /api/menus                          // Liste des menus actifs
GET    /api/menus/{id}                     // Détail d'un menu

// Authentification
POST   /api/register                       // Inscription
POST   /api/login                          // Connexion

// RGPD
GET    /api/rgpd/politique-confidentialite // Politique RGPD
```

5.4.2.2 Routes Authentifiées - Client (15 routes)

```
// Authentification
POST   /api/logout                         // Déconnexion
GET    /api/me                             // Utilisateur connecté

// Gestion commandes
GET    /api/commandes                      // Mes commandes
POST   /api/commandes                      // Créer une commande
GET    /api/commandes/{id}                 // Détail commande
PUT    /api/commandes/{id}/cancel          // Annuler commande

// Gestion avis
POST   /api/avis                           // Laisser un avis
GET    /api/avis/commande/{commandeId}     // Avis d'une commande

// RGPD
GET    /api/rgpd/export-data               // Export données JSON
DELETE /api/rgpd/delete-account            // Supprimer compte
PUT    /api/rgpd/newsletter                // Gérer newsletter
```

```
// Contact
POST   /api/contact                          // Envoyer message
GET    /api/mentions-legales                  // Mentions légales
```

5.4.2.3 Routes Employé (12 routes)

```
// Dashboard
GET    /api/employe/dashboard                // Statistiques employé

// Gestion commandes
GET    /api/admin/commandes                  // Toutes les commandes
GET    /api/admin/commandes/{id}             // Détail commande
PUT    /api/admin/commandes/{id}             // Modifier statut

// Gestion avis
GET    /api/admin/avis/pending               // Avis en attente
PUT    /api/admin/avis/{id}/validate         // Valider avis
PUT    /api/admin/avis/{id}/reject           // Rejeter avis

// Gestion menus
POST   /api/admin/menus                      // Créer menu
PUT    /api/admin/menus/{id}                 // Modifier menu
GET    /api/admin/menus                      // Liste admin menus
PUT    /api/admin/menus/{id}/toggle          // Activer/désactiver
GET    /api/admin/plats                      // Liste plats
```

5.4.2.4 Routes Administrateur (14 routes)

```
// Gestion menus
DELETE /api/admin/menus/{id}                 // Supprimer menu

// Gestion plats
POST   /api/admin/plats                      // Créer plat
GET    /api/admin/plats/{id}                 // Détail plat
PUT    /api/admin/plats/{id}                 // Modifier plat
DELETE /api/admin/plats/{id}                 // Supprimer plat

// Gestion allergènes
GET    /api/admin/allergenes                 // Liste allergènes
POST   /api/admin/allergenes                 // Créer allergène
GET    /api/admin/allergenes/{id}            // Détail allergène
PUT    /api/admin/allergenes/{id}            // Modifier allergène
DELETE /api/admin/allergenes/{id}            // Supprimer allergène

// Gestion utilisateurs
GET    /api/admin/users                      // Liste utilisateurs
PUT    /api/admin/users/{id}/toggle          // Activer/désactiver

// Gestion avis
DELETE /api/admin/avis/{id}                  // Supprimer avis
```

```
// Dashboard admin
GET    /api/admin/dashboard                  // Statistiques admin
```

5.4.3 Exemples de Controllers

5.4.3.1 AuthController - Inscription

```php
public function register(RegisterRequest $request)
{
    // Validation automatique via RegisterRequest

    // Création de l'utilisateur
    $user = User::create([
        'nom' => $request->nom,
        'prenom' => $request->prenom,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'gsm' => $request->gsm,
        'adresse' => $request->adresse,
        'role' => 'utilisateur',
        'active' => true,
        'consentement_rgpd' => true,
        'date_consentement' => now(),
        'newsletter' => $request->newsletter ?? false,
    ]);

    // Génération du token
    $token = $user->createToken('auth-token')->plainTextToken;

    return response()->json([
        'user' => $user,
        'token' => $token,
    ], 201);
}
```

5.4.3.2 CommandeController - Création

```php
public function store(StoreCommandeRequest $request)
{
    // Autorisation
    $this->authorize('create', Commande::class);

    // Récupération du menu
    $menu = Menu::findOrFail($request->menu_id);

    // Validation du stock
    if ($menu->stock < $request->quantite) {
        return response()->json([
            'message' => 'Stock insuffisant'
        ], 400);
    }
```

```
    // Calcul du prix
    $prixTotal = $menu->prix_base * $request->quantite;

    // Création de la commande
    $commande = Commande::create([
        'user_id' => $request->user()->id,
        'menu_id' => $request->menu_id,
        'date_commande' => now(),
        'date_livraison' => $request->date_livraison,
        'adresse_livraison' => $request->adresse_livraison,
        'quantite' => $request->quantite,
        'prix_total' => $prixTotal,
        'statut' => 'en_attente',
        'instructions' => $request->instructions,
    ]);

    // Mise à jour du stock
    $menu->decrement('stock', $request->quantite);

    return response()->json([
        'commande' => $commande->load('menu'),
    ], 201);
}
```

5.4.3.3 MenuController - Liste avec Filtres

```
public function index(Request $request)
{
    $query = Menu::where('actif', true);

    // Filtre par thème
    if ($request->has('theme')) {
        $query->where('theme', $request->theme);
    }

    // Filtre par régime
    if ($request->has('regime')) {
        $query->where('regime', $request->regime);
    }

    // Filtre par prix
    if ($request->has('prix_min')) {
        $query->where('prix_base', '>=', $request->prix_min);
    }

    if ($request->has('prix_max')) {
        $query->where('prix_base', '<=', $request->prix_max);
    }

    // Chargement des relations
    $menus = $query->with(['plats.allergenes', 'avis' => function($q)
```

```
{
        $q->where('statut', 'valide');
    }])->paginate(12);

    return response()->json($menus);
}
```

5.4.4 Modèles Eloquent Détaillés

5.4.4.1 Model User

```php
<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens;

    protected $fillable = [
        'nom',
        'prenom',
        'email',
        'password',
        'gsm',
        'adresse',
        'role',
        'active',
        'consentement_rgpd',
        'date_consentement',
        'newsletter',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'date_consentement' => 'datetime',
        'active' => 'boolean',
        'consentement_rgpd' => 'boolean',
        'newsletter' => 'boolean',
    ];

    // Relations
    public function commandes()
```

```
    {
        return $this->hasMany(Commande::class);
    }

    public function avis()
    {
        return $this->hasMany(Avis::class);
    }

    // Accesseurs
    public function getNomCompletAttribute()
    {
        return "{$this->prenom} {$this->nom}";
    }

    // Scopes
    public function scopeActif($query)
    {
        return $query->where('active', true);
    }

    public function scopeByRole($query, $role)
    {
        return $query->where('role', $role);
    }

    // Méthodes helper
    public function isAdmin()
    {
        return $this->role === 'administrateur';
    }

    public function isEmploye()
    {
        return in_array($this->role, ['employe', 'administrateur']);
    }

    public function canManageCommandes()
    {
        return $this->isEmploye();
    }
}
```

5.4.4.2 Model Menu

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
```

```php
class Menu extends Model
{
    protected $fillable = [
        'titre',
        'description',
        'theme',
        'regime',
        'nb_personne_min',
        'prix_base',
        'stock',
        'conditions',
        'actif',
    ];

    protected $casts = [
        'prix_base' => 'decimal:2',
        'stock' => 'integer',
        'nb_personne_min' => 'integer',
        'actif' => 'boolean',
    ];

    protected $appends = ['note_moyenne'];

    // Relations
    public function plats()
    {
        return $this->belongsToMany(Plat::class, 'menu_plat');
    }

    public function commandes()
    {
        return $this->hasMany(Commande::class);
    }

    public function avis()
    {
        return $this->hasManyThrough(Avis::class, Commande::class);
    }

    // Accesseurs
    public function getNoteMoyenneAttribute()
    {
        return $this->avis()
            ->where('statut', 'valide')
            ->avg('note') ?? 0;
    }

    public function getAllergenesAttribute()
```

```
    {
        return $this->plats()
            ->with('allergenes')
            ->get()
            ->pluck('allergenes')
            ->flatten()
            ->unique('id')
            ->values();
    }

    // Scopes
    public function scopeActif($query)
    {
        return $query->where('actif', true);
    }

    public function scopeByTheme($query, $theme)
    {
        return $query->where('theme', $theme);
    }

    public function scopeByRegime($query, $regime)
    {
        return $query->where('regime', $regime);
    }

    public function scopeWithAvisValides($query)
    {
        return $query->with(['avis' => function($q) {
            $q->where('statut', 'valide');
        }]);
    }

    // Méthodes helper
    public function hasStock($quantite)
    {
        return $this->stock >= $quantite;
    }

    public function decrementStock($quantite)
    {
        $this->decrement('stock', $quantite);
    }
}
```

5.4.4.3 Model Commande

```
<?php

namespace App\Models;
```

```php
use Illuminate\Database\Eloquent\Model;

class Commande extends Model
{
    protected $fillable = [
        'user_id',
        'menu_id',
        'date_commande',
        'date_livraison',
        'adresse_livraison',
        'quantite',
        'prix_total',
        'statut',
        'instructions',
    ];

    protected $casts = [
        'date_commande' => 'datetime',
        'date_livraison' => 'date',
        'prix_total' => 'decimal:2',
        'quantite' => 'integer',
    ];

    const STATUT_EN_ATTENTE = 'en_attente';
    const STATUT_VALIDEE = 'validee';
    const STATUT_EN_PREPARATION = 'en_preparation';
    const STATUT_LIVREE = 'livree';
    const STATUT_ANNULEE = 'annulee';

    // Relations
    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function menu()
    {
        return $this->belongsTo(Menu::class);
    }

    public function avis()
    {
        return $this->hasOne(Avis::class);
    }

    // Scopes
    public function scopeByStatut($query, $statut)
    {
```

```
        return $query->where('statut', $statut);
    }

    public function scopeEnAttente($query)
    {
        return $query->where('statut', self::STATUT_EN_ATTENTE);
    }

    public function scopeLivrees($query)
    {
        return $query->where('statut', self::STATUT_LIVREE);
    }

    public function scopeAnnulables($query)
    {
        return $query->where('statut', self::STATUT_EN_ATTENTE);
    }

    // Méthodes helper
    public function canBeCancelled()
    {
        return $this->statut === self::STATUT_EN_ATTENTE;
    }

    public function canHaveAvis()
    {
        return $this->statut === self::STATUT_LIVREE && !$this->avis;
    }

    public function updateStatut($nouveauStatut)
    {
        $this->update(['statut' => $nouveauStatut]);
    }
}
```

5.4.5 FormRequests - Validation Détaillée

5.4.5.1 RegisterRequest

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rules\Password;

class RegisterRequest extends FormRequest
{
    public function authorize()
    {
```

```
        return true;
    }

    public function rules()
    {
        return [
            'nom' => [
                'required',
                'string',
                'max:100',
                'regex:/^[a-zA-ZÀ-ÿ\s\-]+$/',
            ],
            'prenom' => [
                'required',
                'string',
                'max:100',
                'regex:/^[a-zA-ZÀ-ÿ\s\-]+$/',
            ],
            'email' => [
                'required',
                'string',
                'email:rfc,dns',
                'max:255',
                'unique:users,email',
            ],
            'password' => [
                'required',
                'string',
                'confirmed',
                Password::min(10)
                    ->mixedCase()
                    ->numbers()
                    ->symbols()
                    ->uncompromised(),
            ],
            'gsm' => [
                'required',
                'string',
                'regex:/^\+?[0-9]{10,15}$/',
            ],
            'adresse' => [
                'required',
                'string',
                'max:500',
            ],
            'consentement_rgpd' => [
                'required',
                'accepted',
            ],
            'newsletter' => [
```

```
                'boolean',
            ],
        ];
    }

    public function messages()
    {
        return [
            'nom.required' => 'Le nom est obligatoire.',
            'nom.regex' => 'Le nom ne peut contenir que des lettres.',
            'email.unique' => 'Cet email est déjà utilisé.',
            'password.min' => 'Le mot de passe doit contenir au moins
10 caractères.',
            'consentement_rgpd.accepted' => 'Vous devez accepter la
politique RGPD.',
        ];
    }
}
```

5.4.5.2 StoreCommandeRequest

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreCommandeRequest extends FormRequest
{
    public function authorize()
    {
        return auth()->check();
    }

    public function rules()
    {
        return [
            'menu_id' => [
                'required',
                'integer',
                'exists:menus,id',
            ],
            'date_livraison' => [
                'required',
                'date',
                'after:today',
            ],
            'adresse_livraison' => [
                'required',
                'string',
                'max:500',
```

```
            ],
            'quantite' => [
                'required',
                'integer',
                'min:1',
                'max:100',
            ],
            'instructions' => [
                'nullable',
                'string',
                'max:1000',
            ],
        ];
    }

    public function messages()
    {
        return [
            'menu_id.required' => 'Vous devez sélectionner un menu.',
            'menu_id.exists' => 'Ce menu n\'existe pas.',
            'date_livraison.after' => 'La date de livraison doit être
dans le futur.',
            'quantite.min' => 'La quantité minimum est de 1.',
        ];
    }

    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $menu = Menu::find($this->menu_id);

            if ($menu && $this->quantite < $menu->nb_personne_min) {
                $validator->errors()->add(
                    'quantite',
                    "La quantité minimum pour ce menu est de {$menu-
>nb_personne_min} personnes."
                );
            }

            if ($menu && $menu->stock < $this->quantite) {
                $validator->errors()->add(
                    'quantite',
                    "Stock insuffisant. Disponible : {$menu->stock}"
                );
            }
        });
    }
}
```

5.4.5.3 AuthController - Inscription

```php
public function register(RegisterRequest $request)
{
    // Validation automatique via RegisterRequest

    // Création de l'utilisateur
    $user = User::create([
        'nom' => $request->nom,
        'prenom' => $request->prenom,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'gsm' => $request->gsm,
        'adresse' => $request->adresse,
        'role' => 'utilisateur',
        'active' => true,
        'consentement_rgpd' => true,
        'date_consentement' => now(),
        'newsletter' => $request->newsletter ?? false,
    ]);

    // Génération du token
    $token = $user->createToken('auth-token')->plainTextToken;

    return response()->json([
        'user' => $user,
        'token' => $token,
    ], 201);
}
```

5.4.5.4 CommandeController - Création

```php
public function store(StoreCommandeRequest $request)
{
    // Autorisation
    $this->authorize('create', Commande::class);

    // Récupération du menu
    $menu = Menu::findOrFail($request->menu_id);

    // Validation du stock
    if ($menu->stock < $request->quantite) {
        return response()->json([
            'message' => 'Stock insuffisant'
        ], 400);
    }

    // Calcul du prix
    $prixTotal = $menu->prix_base * $request->quantite;

    // Création de la commande
    $commande = Commande::create([
```

```
            'user_id' => $request->user()->id,
            'menu_id' => $request->menu_id,
            'date_commande' => now(),
            'date_livraison' => $request->date_livraison,
            'adresse_livraison' => $request->adresse_livraison,
            'quantite' => $request->quantite,
            'prix_total' => $prixTotal,
            'statut' => 'en_attente',
            'instructions' => $request->instructions,
        ]);

        // Mise à jour du stock
        $menu->decrement('stock', $request->quantite);

        return response()->json([
            'commande' => $commande->load('menu'),
        ], 201);
    }
```

5.4.5.5 MenuController - Liste avec Filtres

```
public function index(Request $request)
{
    $query = Menu::where('actif', true);

    // Filtre par thème
    if ($request->has('theme')) {
        $query->where('theme', $request->theme);
    }

    // Filtre par régime
    if ($request->has('regime')) {
        $query->where('regime', $request->regime);
    }

    // Filtre par prix
    if ($request->has('prix_min')) {
        $query->where('prix_base', '>=', $request->prix_min);
    }

    if ($request->has('prix_max')) {
        $query->where('prix_base', '<=', $request->prix_max);
    }

    // Chargement des relations
    $menus = $query->with(['plats.allergenes', 'avis' => function($q)
{
        $q->where('statut', 'valide');
    }])->paginate(12);
```

```
    return response()->json($menus);
}
```

6 5. SÉCURITÉ

6.1 5.1 Authentification

6.1.1 Laravel Sanctum

J'ai implémenté l'authentification par tokens avec Laravel Sanctum :

```
// Génération du token lors du login
$token = $user->createToken('auth-token')->plainTextToken;

// Protection des routes
Route::middleware('auth:sanctum')->group(function () {
    // Routes protégées
});
```

6.1.2 Validation des Mots de Passe

Règles strictes implémentées : - Minimum 10 caractères - Au moins une majuscule - Au moins une minuscule - Au moins un chiffre - Au moins un symbole

Les mots de passe sont hashés avec **bcrypt** avant stockage.

6.2 5.2 Validation des Données

6.2.1 FormRequests Laravel

Toutes les données entrantes sont validées via des FormRequests :

```
class RegisterRequest extends FormRequest
{
    public function rules()
    {
        return [
            'nom' => ['required', 'string', 'max:100'],
            'email' => ['required', 'email', 'unique:users'],
            'password' => ['required', 'confirmed', Password::min(10)
                ->mixedCase()
                ->numbers()
                ->symbols()
            ],
            'consentement_rgpd' => ['required', 'accepted'],
        ];
    }
}
```

6.3 5.3 Contrôle d'Accès

6.3.1 Policies Laravel

J'ai implémenté des Policies pour gérer les autorisations :

```
// MenuPolicy
public function create(User $user)
{
    return in_array($user->role, ['administrateur', 'employe']);
}

public function delete(User $user, Menu $menu)
{
    return $user->role === 'administrateur';
}
```

6.4 5.4 Protection du Transport

6.4.1 HTTPS Obligatoire

- Certificat SSL de **Let's Encrypt**
- Renouvellement automatique avec **Certbot**
- Redirection automatique HTTP → HTTPS
- **HSTS** activé (Strict-Transport-Security)

6.4.2 Headers de Sécurité

Middleware custom pour ajouter des headers de sécurité :

```
$response->headers->set('X-Frame-Options', 'DENY');
$response->headers->set('X-Content-Type-Options', 'nosniff');
$response->headers->set('X-XSS-Protection', '1; mode=block');
$response->headers->set('Content-Security-Policy', "...");
```

6.5 5.5 Rate Limiting

Protection contre les abus :

- **5 tentatives** de connexion par minute
- **60 requêtes API** par minute pour les utilisateurs authentifiés
- **60 requêtes API** par minute par IP pour les visiteurs

6.6 5.6 CORS

Configuration stricte du CORS :

```
'allowed_origins' => [
    'https://vite-gourmand.fr',
    'https://www.vite-gourmand.fr',
```

```
],
'supports_credentials' => true,
```

7 6. CONFORMITÉ RGPD

7.1 6.1 Droits Implémentés

7.1.1 Droit à l'Information

- **Politique de confidentialité** accessible publiquement
- **Mentions légales** disponibles
- **Consentement explicite** obligatoire à l'inscription
- Information claire sur les données collectées

7.1.2 Droit d'Accès et de Portabilité

Fonctionnalité d'**export des données** :

```php
public function exportData(Request $request)
{
    $user = $request->user();
    $data = [
        'user' => $user,
        'commandes' => $user->commandes()->with('menu')->get(),
        'avis' => $user->avis()->with('commande')->get(),
    ];
    return response()->json($data);
}
```

L'utilisateur peut télécharger toutes ses données au format **JSON**.

7.1.3 Droit à l'Oubli

Fonctionnalité de **suppression de compte** :

- Confirmation en 2 étapes (texte + mot de passe)
- Suppression définitive et irréversible
- Suppression en cascade : commandes, avis, tokens
- Message de confirmation

7.2 6.2 Gestion du Consentement

7.2.1 À l'Inscription

- Checkbox RGPD **obligatoire**
- Date de consentement enregistrée
- Opt-in explicite pour la newsletter (optionnel)

7.2.2 Modification

L'utilisateur peut à tout moment : - Consulter ses consentements - Modifier son opt-in newsletter - Exporter ses données - Supprimer son compte

7.3 6.3 Sécurité des Données

- Stockage sécurisé dans **PostgreSQL**
- Mots de passe **hashés** (jamais en clair)
- Connexion HTTPS uniquement
- Accès limité selon les rôles

8 7. TESTS

8.1 7.1 Tests Automatisés

J'ai développé **15 tests automatisés** avec PHPUnit couvrant les fonctionnalités critiques :

8.1.1 Tests d'Authentification (9 tests)

```
// RegisterTest.php
test_user_can_register_successfully()
test_registration_requires_rgpd_consent()
test_email_must_be_unique()
test_password_must_meet_requirements()

// LoginTest.php
test_user_can_login_with_valid_credentials()
test_user_cannot_login_with_invalid_credentials()
test_inactive_user_cannot_login()

// LogoutTest.php
test_user_can_logout()
test_token_is_deleted_on_logout()
```

8.1.2 Tests CRUD Menus (3 tests)

```
// MenuTest.php
test_public_can_view_active_menus()
test_admin_can_create_menu()
test_user_cannot_create_menu()  // Test des autorisations
```

8.1.3 Tests RGPD (6 tests)

```
// RgpdTest.php
test_user_can_export_data()
test_user_can_delete_account()
test_account_deletion_requires_password()
test_user_can_update_newsletter_consent()
test_policy_page_is_accessible()
test_legal_notices_page_is_accessible()
```

8.2 7.2 Exécution des Tests

```
php artisan test
```

Résultat : ✅ **15 tests passent** avec succès

8.3 7.3 Couverture Fonctionnelle

Les tests couvrent :

- ✅ Authentification complète (inscription, connexion, déconnexion)
- ✅ Validation des données entrantes
- ✅ Autorisations (Policies)
- ✅ Fonctionnalités RGPD
- ✅ Intégrité des données

9 8. DÉPLOIEMENT EN PRODUCTION

9.1 8.1 Infrastructure

9.1.1 Serveur VPS OVH

- **Système** : Ubuntu 22.04 LTS
- **IP** : 37.59.124.193
- **Domaine** : vite-gourmand.fr (+ www)
- **Accès** : SSH avec authentification par clé

9.1.2 Stack Serveur

- **Nginx 1.18.0** : Serveur web et proxy inverse
- **PHP 8.2 + PHP-FPM** : Traitement des requêtes PHP
- **PostgreSQL 14** : Base de données
- **Composer 2.2.6** : Gestionnaire de dépendances PHP
- **Node.js v18** : Build du frontend

9.2 8.2 Configuration Nginx

```
server {
    listen 443 ssl;
    server_name vite-gourmand.fr www.vite-gourmand.fr;

    # SSL
    ssl_certificate
/etc/letsencrypt/live/vite-gourmand.fr/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/vite-gourmand.fr/privkey.pem;

    # Frontend React
    root /var/www/vite-gourmand/frontend/build;
```

```
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    # Backend API Laravel
    location /api/ {
        root /var/www/vite-gourmand/backend/public;
        rewrite ^/api/(.*)$ /index.php?/$1 last;
    }

    # PHP processing
    location ~ \.php$ {
        root /var/www/vite-gourmand/backend/public;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        include fastcgi_params;
    }
}
```

9.3 8.3 Processus de Déploiement

9.3.1 1. Préparation du Serveur

```
# Installation des dépendances
sudo apt update
sudo apt install nginx php8.2-fpm php8.2-pgsql postgresql nodejs npm

# Configuration PostgreSQL
sudo -u postgres psql
CREATE DATABASE vite_gourmand_prod;
CREATE USER vite_gourmand WITH PASSWORD 'mot_de_passe_sécurisé';
GRANT ALL PRIVILEGES ON DATABASE vite_gourmand_prod TO vite_gourmand;
```

9.3.2 2. Déploiement du Backend

```
cd /var/www/vite-gourmand/backend
composer install --no-dev --optimize-autoloader
php artisan migrate --force
php artisan config:cache
php artisan route:cache
```

9.3.3 3. Build du Frontend

```
cd /var/www/vite-gourmand/frontend
npm install
echo "REACT_APP_API_URL=https://vite-gourmand.fr/api"
> .env.production
npm run build
```

9.3.4 4. Installation SSL

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d vite-gourmand.fr -d www.vite-gourmand.fr
```

Le certificat se renouvelle automatiquement tous les 90 jours.

9.4 8.4 Mise en Production

Date de mise en production : 18 février 2025

URL publique : https://vite-gourmand.fr

Tests de production effectués : - ✅ Inscription et connexion fonctionnelles - ✅ Passage de commandes opérationnel - ✅ Dashboard employé accessible - ✅ Dashboard admin fonctionnel - ✅ Export RGPD opérationnel - ✅ HTTPS actif et certificat valide - ✅ Performance satisfaisante

9.5 8.5 Configuration PHP-FPM

9.5.1 Fichier /etc/php/8.2/fpm/pool.d/www.conf

```
[www]
user = www-data
group = www-data

listen = /var/run/php/php8.2-fpm.sock
listen.owner = www-data
listen.group = www-data
listen.mode = 0660

; Configuration Pool
pm = dynamic
pm.max_children = 10
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3
pm.max_requests = 500

; Timeouts
request_terminate_timeout = 30s
request_slowlog_timeout = 10s
slowlog = /var/log/php-fpm/slow.log

; PHP settings
php_admin_value[error_log] = /var/log/php-fpm/error.log
php_admin_flag[log_errors] = on
php_admin_value[memory_limit] = 256M
php_admin_value[upload_max_filesize] = 10M
php_admin_value[post_max_size] = 10M
php_admin_value[max_execution_time] = 30
```

9.5.2 Fichier /etc/php/8.2/fpm/php.ini (extraits)

```
[PHP]
expose_php = Off
max_execution_time = 30
```

```
max_input_time = 60
memory_limit = 256M
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors = Off
log_errors = On
error_log = /var/log/php8.2-fpm.log

upload_max_filesize = 10M
post_max_size = 10M
max_file_uploads = 20

[Date]
date.timezone = Europe/Paris

[Session]
session.cookie_httponly = 1
session.cookie_secure = 1
session.use_strict_mode = 1

[opcache]
opcache.enable = 1
opcache.memory_consumption = 128
opcache.interned_strings_buffer = 8
opcache.max_accelerated_files = 10000
opcache.revalidate_freq = 2
opcache.fast_shutdown = 1
```

9.6 8.6 Configuration PostgreSQL

9.6.1 Fichier /etc/postgresql/14/main/postgresql.conf (extraits)

```
# Connexions
max_connections = 100
superuser_reserved_connections = 3

# Mémoire
shared_buffers = 256MB
effective_cache_size = 1GB
work_mem = 4MB
maintenance_work_mem = 64MB

# Logs
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_rotation_age = 1d
log_min_duration_statement = 1000  # Log queries > 1s
log_line_prefix = '%m [%p] %u@%d '
log_timezone = 'Europe/Paris'
```

```
# Performance
random_page_cost = 1.1  # SSD
effective_io_concurrency = 200
default_statistics_target = 100

# Write-Ahead Log
wal_level = replica
wal_buffers = 16MB
checkpoint_completion_target = 0.9
```

9.6.2 Fichier /etc/postgresql/14/main/pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             postgres                                peer
local   all             all                                     peer
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
```

9.7 8.7 Optimisations de Performance

9.7.1 Backend Laravel

1. Optimisation Composer

```
composer install --optimize-autoloader --no-dev
composer dump-autoload --optimize
```

2. Cache Laravel

```
php artisan config:cache   # Cache configuration
php artisan route:cache    # Cache routes
php artisan view:cache     # Cache Blade views
```

3. OPcache

OPcache est activé pour accélérer l'exécution PHP : - Mise en cache des scripts compilés - Réduction du temps de parsing - Performance améliorée de 30-40%

9.7.2 Frontend React

1. Build de Production

```
NODE_ENV=production npm run build
```

Optimisations automatiques : - Minification JavaScript - Tree shaking (suppression code mort) - Code splitting - Compression gzip

2. Optimisations Manuelles

```
// Lazy loading des routes
const MenusPage = lazy(() => import('./pages/MenusPage'));
const AdminPage = lazy(() => import('./pages/admin/AdminPage'));
```

```javascript
// Memoization
const MenuCard = React.memo(({ menu }) => {
  // ...
});

// useMemo pour calculs coûteux
const filteredMenus = useMemo(() => {
  return menus.filter(menu => menu.theme === selectedTheme);
}, [menus, selectedTheme]);
```

9.7.3 Base de Données

1. Index Stratégiques

Les index ont été placés sur les colonnes fréquemment utilisées dans les WHERE et JOIN :

```sql
-- Utilisés fréquemment
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_commandes_user ON commandes(user_id);
CREATE INDEX idx_commandes_statut ON commandes(statut);
CREATE INDEX idx_menus_actif ON menus(actif);
```

2. Eager Loading

Pour éviter le problème N+1 :

```php
// ❌ Mauvais : N+1 queries
$menus = Menu::all();
foreach ($menus as $menu) {
    echo $menu->plats->count();  // 1 query par menu
}

// ✅ Bon : 2 queries seulement
$menus = Menu::with('plats')->get();
foreach ($menus as $menu) {
    echo $menu->plats->count();
}
```

3. Pagination

Toutes les listes sont paginées :

```php
$menus = Menu::paginate(12);  // 12 items par page
$commandes = Commande::paginate(20);  // 20 items par page
```

9.7.4 Nginx

1. Compression gzip

```
gzip on;
gzip_vary on;
gzip_types text/plain text/css application/json application/javascript
text/xml application/xml;
```

```
gzip_min_length 256;
gzip_comp_level 6;
```

2. Cache statique

```
location ~* \.(jpg|jpeg|png|gif|ico|css|js|svg|woff|woff2|ttf)$ {
    expires 30d;
    add_header Cache-Control "public, immutable";
}
```

3. HTTP/2

```
listen 443 ssl http2;
```

HTTP/2 améliore les performances : - Multiplexing - Server push - Compression des headers

9.8 8.8 Monitoring et Logs

9.8.1 Logs Laravel

Emplacement : `/var/www/vite-gourmand/backend/storage/logs/laravel.log`

Configuration : `config/logging.php`

```php
'channels' => [
    'stack' => [
        'driver' => 'stack',
        'channels' => ['single', 'slack'],
        'ignore_exceptions' => false,
    ],
    'single' => [
        'driver' => 'single',
        'path' => storage_path('logs/laravel.log'),
        'level' => env('LOG_LEVEL', 'debug'),
    ],
],
```

Rotation : Logs automatiquement rotatés chaque jour

9.8.2 Logs Nginx

Access log : `/var/log/nginx/vite-gourmand-access.log`

```
37.59.124.193 - - [18/Feb/2025:10:30:45 +0100] "GET /api/menus
HTTP/2.0" 200 1523
```

Error log : `/var/log/nginx/vite-gourmand-error.log`

```
2025/02/18 10:30:45 [error] 12345#0: *67 FastCGI sent in stderr: "PHP
message..."
```

9.8.3 Logs PostgreSQL

Emplacement : `/var/log/postgresql/postgresql-14-main.log`

Requêtes lentes : Configuration pour logger les queries > 1 seconde

9.8.4 Monitoring des Performances

1. Temps de réponse API

Mesures prises en production :

Route	Temps moyen	Status
GET /api/menus	120ms	✅ Optimal
POST /api/commandes	180ms	✅ Bon
GET /api/admin/commandes	250ms	⚠️ À optimiser
POST /api/register	300ms	⚠️ Bcrypt lent (normal)

2. Utilisation des ressources serveur

```
# CPU
top -b -n 1 | grep Cpu
# Cpu(s):  5.2%us,  2.1%sy,  0.0%ni, 92.5%id

# Mémoire
free -h
#               total        used        free
# Mem:           4.0Gi       1.5Gi       2.0Gi

# Disque
df -h
# /dev/sda1       40G    12G    26G   32% /
```

3. Connexions PostgreSQL

```
SELECT count(*) FROM pg_stat_activity;
-- Moyenne : 3-5 connexions actives
```

9.8.5 Sauvegarde

Script de sauvegarde quotidien : `/root/backup.sh`

```
#!/bin/bash
DATE=$(date +%Y-%m-%d)
BACKUP_DIR="/var/backups/vite-gourmand"

# Backup PostgreSQL
pg_dump vite_gourmand_prod > "$BACKUP_DIR/db-$DATE.sql"
```

```
# Backup fichiers
tar -czf "$BACKUP_DIR/files-$DATE.tar.gz" /var/www/vite-gourmand

# Garder 7 jours
find $BACKUP_DIR -name "*.sql" -mtime +7 -delete
find $BACKUP_DIR -name "*.tar.gz" -mtime +7 -delete
```

Cron : Exécution à 2h du matin

```
0 2 * * * /root/backup.sh >> /var/log/backup.log 2>&1
```

10 9. RÉALISATION

10.1 9.1 Fonctionnalités Développées

10.1.1 Espace Public

Consultation des menus - Liste paginée des menus actifs - Filtres par thème (bio, végétarien, gastronomique, etc.) - Filtres par régime alimentaire (normal, végétarien, vegan, sans gluten) - Affichage détaillé : composition, allergènes, prix, stock - Système de notation visible (moyenne des avis validés)

Inscription et Connexion - Formulaire d'inscription avec validation stricte - Vérification en temps réel de la disponibilité de l'email - Validation du mot de passe (force, correspondance) - Consentement RGPD obligatoire - Opt-in newsletter optionnel - Connexion sécurisée avec génération de token - Rate limiting (5 tentatives/minute)

10.1.2 Espace Client

Gestion des Commandes - Formulaire de commande avec : - Sélection du menu - Date de livraison (validation : date future) - Adresse de livraison (pré-remplie, modifiable) - Quantité (validation : min selon le menu) - Instructions spéciales (optionnel) - Calcul automatique du prix total - Historique des commandes avec statuts : - En attente (orange) - Validée (bleu) - En préparation (violet) - Livrée (vert) - Annulée (rouge) - Possibilité d'annuler une commande en attente - Détail de chaque commande

Système d'Avis - Formulaire d'avis après livraison : - Note de 1 à 5 étoiles (obligatoire) - Commentaire texte (obligatoire) - Un seul avis par commande - Modération avant publication - Affichage des avis validés sur les menus

RGPD - Page "Mes Données" avec : - Bouton d'export des données (JSON) - Gestion du consentement newsletter - Bouton de suppression de compte - Processus de suppression sécurisé : - Confirmation par texte "SUPPRIMER" - Vérification du mot de passe - Suppression définitive et irréversible

10.1.3 Espace Employé

Dashboard Employé - Statistiques : - Nombre de commandes en attente - Nombre d'avis à valider - Chiffre d'affaires du jour - Liste de toutes les commandes avec filtres par statut - Modification du statut des commandes : - En attente → Validée - Validée → En préparation - En préparation → Livrée - Gestion des avis : - Liste des avis en attente - Validation (publication) - Rejet (masquage)

Gestion des Menus - Création de menus : - Formulaire complet - Sélection des plats - Définition du thème et régime - Prix et stock - Modification de menus existants - Activation/désactivation

10.1.4 Espace Administrateur

Dashboard Admin - Accès à toutes les fonctionnalités employé - Statistiques avancées - Vue d'ensemble de la plateforme

Gestion Complète - **Menus** : - Création, modification, suppression - Vérification des commandes associées avant suppression - **Plats** : - CRUD complet - Association aux allergènes - Réutilisables dans plusieurs menus - **Allergènes** : - Création et gestion - Association automatique aux plats - **Utilisateurs** : - Liste de tous les utilisateurs - Activation/désactivation des comptes - Vue du rôle et statut

10.2 9.2 Capture d'Écran des Interfaces

Note : Des captures d'écran de l'application en production sont disponibles en annexe de ce dossier.

Pages capturées : 1. Page d'accueil 2. Liste des menus 3. Détail d'un menu 4. Formulaire de commande 5. Mes commandes (client) 6. Dashboard employé 7. Dashboard administrateur 8. Page RGPD - Mes données

11 10. DIFFICULTÉS RENCONTRÉES ET SOLUTIONS

11.1 10.1 Accès SSH au VPS

11.1.1 Problème

Lors du déploiement, je n'arrivais pas à me connecter en SSH au VPS OVH. Le mot de passe était systématiquement refusé, même après plusieurs réinitialisations via l'interface OVH.

11.1.2 Solution Mise en Place

1. Passage en **mode rescue** via l'interface OVH
2. Montage du système de fichiers : `mount /dev/sda1 /mnt`
3. Chroot dans le système : `chroot /mnt`
4. Identification de l'utilisateur existant `dev` via `grep /bin/bash /etc/passwd`
5. Réinitialisation du mot de passe : `passwd dev`

6. Redémarrage en mode normal
7. Connexion SSH réussie avec l'utilisateur `dev`

11.1.3 Apprentissage

Cette difficulté m'a permis d'approfondir mes connaissances en : - Administration système Linux - Gestion des utilisateurs et permissions - Mode rescue et récupération système - Débrouillardise face à un blocage technique

11.2 10.2 Configuration de l'API Frontend

11.2.1 Problème

Après le build de production du frontend React, l'application continuait à appeler l'API sur `http://localhost:8000/api` au lieu de `https://vite-gourmand.fr/api`.

11.2.2 Analyse

Le projet utilise **Create React App** (et non Vite malgré le nom). Les variables d'environnement doivent donc utiliser le préfixe `REACT_APP_` et non `VITE_`.

De plus, le fichier `api.js` utilisait la syntaxe Vite :

```
const API_URL = import.meta.env.VITE_API_URL ||
'http://localhost:8000/api';
```

11.2.3 Solution

1. Création du fichier `.env.production` avec :

   ```
   REACT_APP_API_URL=https://vite-gourmand.fr/api
   ```

2. Modification de `src/services/api.js` :

   ```
   const API_URL = process.env.REACT_APP_API_URL || 'https://vite-
   gourmand.fr/api';
   ```

3. Rebuild complet :

   ```
   rm -rf build
   NODE_ENV=production npm run build
   ```

11.2.4 Apprentissage

Importance de bien comprendre l'outil de build utilisé (CRA vs Vite) et ses spécificités concernant les variables d'environnement.

11.3 10.3 Routing Nginx pour l'API

11.3.1 Problème

Les routes API retournaient des erreurs 404 malgré une configuration Nginx qui semblait correcte. Les requêtes vers `/api/menus` ne parvenaient pas au backend Laravel.

11.3.2 Tentatives Infructueuses

1. Utilisation de `alias` au lieu de `root`
2. Configuration avec des blocks PHP imbriqués
3. Différentes variations de `try_files`

11.3.3 Solution Finale

Configuration Nginx fonctionnelle :

```
location /api/ {
    root /var/www/vite-gourmand/backend/public;
    rewrite ^/api/(.*)$ /index.php?/$1 last;
}

location ~ \.php$ {
    root /var/www/vite-gourmand/backend/public;
    fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
```

11.3.4 Apprentissage

Compréhension approfondie de : - La directive `rewrite` de Nginx - La différence entre `root` et `alias` - Le fonctionnement de FastCGI - L'importance de l'ordre des location blocks

11.4 10.4 Dépendances Composer en Production

11.4.1 Problème

Lors de l'exécution de `composer install` en production, des erreurs survenaient à cause de dépendances manquantes (extension MongoDB) et de versions PHP incompatibles (Symfony CSS Selector nécessitant PHP 8.4).

11.4.2 Solution

```
composer update --optimize-autoloader --no-dev --ignore-platform-req=ext-mongodb
```

Options utilisées : - `--no-dev` : Exclut les dépendances de développement - `--optimize-autoloader` : Optimise l'autoloader pour la production - `--ignore-platform-req=ext-mongodb` : Ignore l'extension MongoDB non nécessaire

11.4.3 Apprentissage

Importance de gérer correctement les dépendances selon l'environnement (développement vs production) et savoir contourner les conflits de version.

12 11. BILAN DU PROJET

12.1 11.1 Objectifs Atteints

✅ **Application complète et fonctionnelle** - Frontend React responsive - Backend Laravel robuste - Base de données PostgreSQL bien structurée

✅ **Stack technique moderne** - Technologies actuelles et recherchées sur le marché - Architecture découplée facilement maintenable - Code organisé et commenté

✅ **Sécurité et conformité RGPD** - Authentification sécurisée par tokens - Validation stricte des données - Protection contre les attaques courantes - Droits RGPD implémentés

✅ **Déployée en production** - Application accessible sur https://vite-gourmand.fr - HTTPS avec certificat SSL valide - Performance satisfaisante - Tests de production réussis

✅ **Testée et documentée** - 15 tests automatisés - Documentation technique complète - Code commenté et structuré

12.2 11.2 Compétences Développées

12.2.1 Compétences Techniques

Frontend (CCP1) - Développement d'interfaces utilisateur avec React - Gestion de l'état avec Context API - Routing et navigation (React Router) - Communication HTTP avec Axios - Responsive design

Backend (CCP2) - Développement d'API REST avec Laravel - Modélisation de base de données relationnelle - Eloquent ORM et requêtes optimisées - Authentification et autorisation - Validation et sécurité

Déploiement (CCP3) - Configuration de serveur Linux (Ubuntu) - Configuration Nginx - Gestion de base de données PostgreSQL - Installation et configuration SSL - Mise en production

12.2.2 Compétences Méthodologiques

- **Organisation** : Planification et suivi des fonctionnalités (F01-F12)
- **Résolution de problèmes** : Débrouillardise face aux blocages
- **Autonomie** : Recherche de solutions et documentation
- **Rigueur** : Tests, validation, sécurité
- **Documentation** : Code commenté, README, dossier projet

12.2.3 Compétences Transversales

- **Adaptabilité** : Apprentissage de nouvelles technologies
- **Persévérance** : Résolution des difficultés techniques
- **Professionnalisme** : Respect des bonnes pratiques
- **Communication** : Documentation claire et structurée

12.3 11.3 Métriques du Projet

Quantitatif - ⏱️ 12 jours de développement - 💻 ~8000 lignes de code (Backend + Frontend) - 📊 13 tables de base de données - 🚏 46 routes API - 🧪 15 tests automatisés - 📄 15+ pages/composants React

Qualitatif - ✅ Application fonctionnelle en production - ✅ Code organisé et maintenable - ✅ Sécurité robuste - ✅ Conforme RGPD - ✅ Responsive design - ✅ Performance satisfaisante

12.4 11.4 Perspectives d'Évolution

12.4.1 Améliorations Fonctionnelles

- **Paiement en ligne** : Intégration de Stripe ou PayPal
- **Notifications email** : Confirmation de commande, changement de statut
- **Application mobile** : Version React Native
- **Dashboard statistiques** : Graphiques de CA, commandes, etc.
- **Système de réservation** : Planification des événements
- **Chat en direct** : Support client

12.4.2 Améliorations Techniques

- **Cache Redis** : Amélioration des performances
- **Monitoring** : Logs centralisés, métriques (Sentry, New Relic)
- **CI/CD** : Automatisation des tests et déploiements
- **Internationalisation** : Support multilingue (i18n)
- **Accessibilité** : Conformité WCAG
- **PWA** : Progressive Web App pour une expérience mobile améliorée

12.4.3 Évolutions Business

- **Programme de fidélité** : Points, réductions
- **Partenariats** : Intégration avec plateformes de livraison
- **Marketplace** : Plateforme multi-traiteurs
- **API publique** : Permettre l'intégration par des tiers

12.5 11.5 Bilan Personnel

Ce projet m'a permis de mettre en pratique l'ensemble des compétences acquises durant ma formation au Titre Professionnel Développeur Web et Web Mobile.

Points forts : - Maîtrise du stack complet (React + Laravel + PostgreSQL) - Capacité à déployer une application en production - Résolution autonome de problèmes techniques complexes - Respect des bonnes pratiques de sécurité et de développement

Points d'amélioration : - Approfondir les tests (couverture plus large) - Améliorer la gestion des erreurs frontend - Optimiser les performances (cache, lazy loading) - Renforcer la documentation du code

Conclusion : Ce projet démontre ma capacité à concevoir, développer, tester et déployer une application web complète et professionnelle. Je suis maintenant prêt à intégrer une équipe de développement et à contribuer à des projets d'envergure.

13 12. CONCLUSION

Le projet **Vite & Gourmand** représente l'aboutissement de ma formation au Titre Professionnel Développeur Web et Web Mobile. J'ai réussi à créer une application web complète, fonctionnelle et sécurisée en seulement 12 jours.

Réalisations principales :

- ✅ Application **full-stack moderne** (React + Laravel + PostgreSQL)
- ✅ **29 cas d'usage** couvrant tous les besoins identifiés
- ✅ **Sécurité robuste** (HTTPS, authentification, validation)
- ✅ **Conformité RGPD** (export données, droit à l'oubli)
- ✅ **Déploiement production** sur VPS avec SSL
- ✅ **15 tests automatisés** garantissant la qualité
- ✅ **Documentation complète** (technique et utilisateur)

Compétences validées :

Ce projet couvre l'ensemble des **3 CCP du Titre Professionnel DWWM** :

- **CCP1** : Développement frontend (React, interfaces responsives)
- **CCP2** : Développement backend (Laravel, API REST, BDD)
- **CCP3** : Déploiement et sécurisation (VPS, Nginx, HTTPS)

Vision professionnelle :

L'application est **production-ready** et pourrait être utilisée dès maintenant par un véritable traiteur événementiel. L'architecture choisie permet d'évoluer facilement (ajout de fonctionnalités, montée en charge).

Ce projet démontre ma capacité à : - Analyser des besoins et concevoir une solution adaptée - Développer une application complète de A à Z - Appliquer les bonnes pratiques de développement - Résoudre des problèmes techniques complexes - Déployer et maintenir une application en production - Travailler de manière autonome et rigoureuse

Je suis désormais **prêt à intégrer le monde professionnel** en tant que Développeur Web et Web Mobile, avec la volonté de continuer à apprendre et à me perfectionner dans ce domaine passionnant.

14 ANNEXES

14.1 Annexe A : Comptes de Démonstration

Pour tester l'application en production, trois comptes de démonstration sont disponibles :

Client (Utilisateur) - Email : client@demo.fr - Mot de passe : Password123! - Rôle : utilisateur

Employé - Email : employe@demo.fr - Mot de passe : Password123! - Rôle : employe

Administrateur - Email : admin@demo.fr - Mot de passe : Password123! - Rôle : administrateur

14.2 Annexe B : Liens Utiles

- **Application en production** : https://vite-gourmand.fr
- **Repository GitHub** : https://github.com/yvesnet9/vite-gourmand
- **Documentation technique** : Disponible dans le README.md

14.3 Annexe C : Technologies et Versions

Frontend - React : 18.x - React Router : 6.x - Axios : 1.x - Create React App : 5.x

Backend - Laravel : 10.x - PHP : 8.2 - Laravel Sanctum : 3.x - Composer : 2.2.6

Base de Données - PostgreSQL : 14.x

Serveur - Ubuntu : 22.04 LTS - Nginx : 1.18.0 - PHP-FPM : 8.2 - Node.js : 18.x

Outils - Git : 2.x - Certbot : Let's Encrypt - npm : 8.x

14.4 Annexe D : Glossaire

API : Application Programming Interface - Interface de programmation permettant la communication entre le frontend et le backend

CRUD : Create, Read, Update, Delete - Opérations de base sur les données

JWT : JSON Web Token - Format de token pour l'authentification

MCD : Modèle Conceptuel de Données - Représentation abstraite de la structure des données

ORM : Object-Relational Mapping - Technique de programmation pour convertir des données entre des systèmes incompatibles

RGPD : Règlement Général sur la Protection des Données - Réglementation européenne sur la protection des données personnelles

REST : Representational State Transfer - Style d'architecture pour les API

SPA : Single Page Application - Application web qui ne recharge qu'une seule page HTML

SSL/TLS : Secure Sockets Layer / Transport Layer Security - Protocoles de sécurisation des échanges sur Internet

VPS : Virtual Private Server - Serveur privé virtuel

FIN DU DOSSIER PROJET

Jamesy MUKUNA MUKENKETAYI
Session Juin-Juillet 2026
STUDI - Paris