

Vite & Gourmand

Documentation Technique

Architecture, technologies et déploiement
Bordeaux - 2026

Version 1.0 - Février 2026
Projet Web Developer - RNCP 37674
Pierre - Développeur Web

Table des matières

1. Vue d'ensemble du projet
2. Architecture technique
3. Stack technologique
4. Structure des fichiers
5. Base de données MySQL
6. Base de données MongoDB
7. API REST - Routes backend
8. Authentification et sécurité
9. Frontend
10. Déploiement
11. Variables d'environnement
12. Tests et comptes de démonstration

1. Vue d'ensemble du projet

Vite & Gourmand est une application web full-stack pour un service de traiteur gastronomique. Le projet couvre l'ensemble du cycle de développement : conception (maquettes Figma), développement frontend/backend, base de données relationnelle et NoSQL, et déploiement en production.

1.1 Objectifs techniques

Développer une application web responsive avec gestion d'authentification, CRUD complet sur les ressources (menus, commandes, avis, utilisateurs), système de rôles (client, employé, admin), et déploiement cloud.

1.2 Charte graphique

Élément	Valeur	Usage
Vert foncé	#2E7D32	Couleur principale, headers, boutons
Or	#D4AF37	Accents, éléments premium
Bordeaux	#722F37	Couleur secondaire, alertes
Playfair Display	Serif	Titres et headings
Open Sans	Sans-serif	Corps de texte

2. Architecture technique

L'application suit une architecture client-serveur classique avec séparation frontend/backend.

2.1 Schéma d'architecture

Couche	Technologie	Rôle
Frontend	HTML5, CSS3, JavaScript	Interface utilisateur responsive
Backend	Node.js, Express.js	API REST, logique métier
BDD relationnelle	MySQL	Données transactionnelles
BDD NoSQL	MongoDB	Statistiques et analytics
Hébergement	Railway	Déploiement cloud (PaaS)
Versioning	Git / GitHub	Gestion du code source

2.2 Flux de données

Le navigateur envoie des requêtes HTTP (fetch API) au serveur Express. Le serveur traite la requête, interroge MySQL ou MongoDB selon le besoin, et renvoie une réponse JSON. Le frontend met à jour l'interface dynamiquement sans rechargement de page.

3. Stack technologique

3.1 Backend

Package	Version	Usage
express	^4.x	Framework web HTTP
mysql2	^3.x	Driver MySQL avec support Promises
mongoose	^7.x	ODM MongoDB
bcrypt	^5.x	Hashage des mots de passe
express-session	^1.x	Gestion des sessions utilisateur
nodemailer	^6.x	Envoi d'emails (SMTP)
dotenv	^16.x	Variables d'environnement
cors	^2.x	Cross-Origin Resource Sharing

3.2 Frontend

Technologie	Usage
HTML5	Structure sémantique des pages
CSS3	Mise en page responsive (Flexbox, Grid, Media Queries)
JavaScript (ES5+)	Interactions dynamiques, appels API fetch
Chart.js	Graphiques dans le dashboard admin

4. Structure des fichiers

```
Vite-Gourmand/
├── backend/
│   ├── server.js           # Serveur Express principal
│   ├── package.json        # Dépendances Node.js
│   ├── .env                # Variables d'environnement
│   └── database.sql         # Script création BDD
├── frontend/
│   ├── html/
│   │   ├── index.html      # Page d'accueil
│   │   ├── menus.html      # Page des menus
│   │   ├── contact.html     # Page contact
│   │   ├── connexion.html   # Page connexion
│   │   ├── inscription.html # Page inscription
│   │   ├── espace-utilisateur.html
│   │   ├── espace-employe.html
│   │   └── espace-admin.html
│   ├── Css/
│   │   ├── style.css       # Feuille de style principale
│   │   └── js/              # Scripts JavaScript
│   │       ├── main.js
│   │       ├── auth.js
│   │       ├── contact-form.js
│   │       ├── filters.js
│   │       ├── espace-utilisateur.js
│   │       ├── espace-employe.js
│   │       └── espace-admin.js
│   └── images/              # Images des menus
└── package.json             # Root package (Railway)
```

5. Base de données MySQL

MySQL est utilisé pour toutes les données transactionnelles. La base contient 7 tables avec des relations par clés étrangères.

5.1 Schéma des tables

Table	Description	Clés étrangères
utilisateurs	Comptes (client, employé, admin)	-
menus	Menus avec prix, thème, régime	-
plats	Plats composant chaque menu	menu_id → menus
commandes	Commandes clients avec prix	utilisateur_id, menu_id
avis	Notes et commentaires	utilisateur_id, commande_id
messages_contact	Messages du formulaire contact	-
tokens_reinitialisation	Tokens reset mot de passe	utilisateur_id

5.2 Table utilisateurs (détail)

Colonne	Type	Contraintes
id	INT AUTO_INCREMENT	PRIMARY KEY
nom	VARCHAR(50)	NOT NULL
prenom	VARCHAR(50)	NOT NULL
email	VARCHAR(254)	NOT NULL UNIQUE
telephone	VARCHAR(20)	DEFAULT NULL
mot_de_passe	VARCHAR(255)	NOT NULL (bcrypt hash)
role	ENUM('client','employe','admin')	DEFAULT 'client'
date_creation	DATETIME	DEFAULT CURRENT_TIMESTAMP

5.3 Table commandes (détail)

Colonne	Type	Description
id	INT AUTO_INCREMENT	Identifiant unique
utilisateur_id	INT	FK → utilisateurs(id)
menu_id	INT	FK → menus(id)

nombre_personnes	INT	Minimum selon le menu
prix_unitaire	DECIMAL(6,2)	Prix par personne
reduction	DECIMAL(4,2)	10% si > 5 personnes
prix_total	DECIMAL(8,2)	Total après réduction
statut	ENUM(...)	en_attente/confirmee/en_cours/livree/annulee

6. Base de données MongoDB

MongoDB est utilisé en complément de MySQL pour stocker les statistiques d'utilisation (visites, commandes, inscriptions). Cela permet des analyses sans impacter les performances de la base transactionnelle.

6.1 Collection : stats

Champ	Type	Description
type	String	visite / commande / inscription
page	String	Page visitée
date	Date	Horodatage automatique
details	Object	Données complémentaires (email, prix...)

7. API REST - Routes backend

7.1 Authentification

Méthode	Route	Description	Auth
POST	/api/inscription	Créer un compte	Non
POST	/api/connexion	Se connecter	Non
GET	/api/deconnexion	Se déconnecter	Oui
GET	/api/profil	Voir son profil	Oui
POST	/api/mot-de-passe-oublie	Reset password	Non

7.2 Menus et commandes

Méthode	Route	Description	Auth
GET	/api/menus	Liste des menus actifs	Non
GET	/api/menus/:id	Détail menu + plats	Non
POST	/api/commandes	Passer commande	Client
GET	/api/commandes	Mes commandes	Client
POST	/api/contact	Envoyer message	Non
POST	/api/avis	Laisser un avis	Client
GET	/api/avis	Liste des avis	Non

7.3 Espace employé et admin

Méthode	Route	Description	Auth
GET	/api/employe/commandes	Toutes les commandes	Employé
PUT	/api/employe/commandes/:id	Modifier statut	Employé
POST	/api/employe/menus	Ajouter un menu	Employé
GET	/api/admin/stats	Dashboard statistiques	Admin
GET	/api/admin/commandes	Toutes les commandes	Admin
GET	/api/admin/employes	Liste employés	Admin
POST	/api/admin/employes	Créer employé	Admin

8. Authentification et sécurité

8.1 Hashage des mots de passe

Les mots de passe sont hashés avec **bcrypt** (10 rounds de salting) avant stockage en base de données. Le mot de passe en clair n'est jamais stocké ni loggé.

8.2 Sessions

L'authentification utilise **express-session** avec un cookie HTTP d'une durée de 24 heures. La session stocke l'ID, le nom, l'email et le rôle de l'utilisateur.

8.3 Contrôle d'accès (RBAC)

Trois niveaux de middleware vérifient les droits d'accès :

authRequired : Vérifie que l'utilisateur est connecté

employeRequired : Vérifie le rôle employé ou admin

adminRequired : Vérifie le rôle admin uniquement

8.4 Protection des formulaires

Validation côté client : Pattern HTML5 + validation JavaScript (regex email, longueur mot de passe, correspondance des champs)

Validation côté serveur : Vérification de tous les champs obligatoires, format email, longueur minimum

Requêtes préparées : Toutes les requêtes MySQL utilisent des paramètres préparés (?) pour empêcher les injections SQL

Honeypot anti-spam : Champ invisible détectant les bots sur les formulaires

Content-Security-Policy : En-tête CSP sur les pages sensibles

8.5 Politique de mot de passe

Minimum 8 caractères, au moins une majuscule, une minuscule, un chiffre et un caractère spécial. Un indicateur visuel de force guide l'utilisateur.

9. Frontend

9.1 Responsive Design

Le site utilise une approche mobile-first avec des media queries CSS pour s'adapter aux différentes tailles d'écran. Trois breakpoints principaux sont définis :

Breakpoint	Largeur	Adaptations
Mobile	< 768px	Menu hamburger, colonnes empilées
Tablette	768px - 1199px	Grille 2 colonnes, navigation adaptée
Desktop	>= 1200px	Grille complète, navigation horizontale

9.2 JavaScript côté client

Fichier	Rôle
main.js	Menu hamburger, toggle mot de passe, utilitaires globaux
auth.js	Validation et soumission des formulaires connexion/inscription
contact-form.js	Validation et envoi du formulaire de contact
filters.js	Filtres dynamiques sur la page des menus
espace-utilisateur.js	Logique de l'espace client (commandes, avis)
espace-employe.js	Gestion des commandes et menus (employé)
espace-admin.js	Dashboard admin avec Chart.js

9.3 Appels API

Tous les appels API utilisent **fetch()** avec des chemins relatifs (/api/...) pour assurer la compatibilité entre l'environnement local et la production. Les réponses JSON sont traitées de manière asynchrone.

10. Déploiement

10.1 Plateforme

L'application est déployée sur **Railway** (PaaS), qui fournit l'hébergement Node.js et une instance MySQL. Le déploiement est automatique à chaque push sur la branche main du dépôt GitHub.

10.2 Architecture de déploiement

Service	Type	Détails
Vite-Gourmand	Web Service	Node.js, déploiement auto depuis GitHub
MySQL	Database	Instance MySQL managée par Railway

10.3 Processus de déploiement

1. Push du code sur GitHub (branche main)
2. Railway détecte le changement et lance un build
3. Railpack installe les dépendances (npm install)
4. Le serveur démarre (node backend/server.js)
5. L'application est accessible sur le domaine Railway

10.4 URL de production

URL : <https://vite-gourmand-production.up.railway.app>

11. Variables d'environnement

Les variables sensibles sont stockées dans un fichier `.env` (local) ou dans les variables Railway (production).

Variable	Description	Exemple
PORT	Port du serveur	3000
DB_HOST	Hôte MySQL	mysql.railway.internal
DB_USER	Utilisateur MySQL	root
DB_PASSWORD	Mot de passe MySQL	(généré par Railway)
DB_NAME	Nom de la base	railway
SESSION_SECRET	Clé secrète des sessions	(chaîne aléatoire)
MONGO_URI	URI de connexion MongoDB	mongodb+srv://...
EMAIL_HOST	Serveur SMTP	smtp.gmail.com
EMAIL_PORT	Port SMTP	587
EMAIL_USER	Email expéditeur	contact@vite-gourmand.fr
EMAIL_PASSWORD	Mot de passe email	(app password)

Sécurité : Le fichier `.env` ne doit jamais être commité dans le dépôt Git. Il est listé dans `.gitignore`.

12. Tests et comptes de démonstration

12.1 Comptes de test

Rôle	Email	Mot de passe
Admin	admin@vite-gourmand.fr	Admin123!
Employé	julie@vite-gourmand.fr	Employe123!
Client	pierre@email.com	Client123!

12.2 Scénarios de test

Fonctionnalité	Scénario	Résultat attendu
Inscription	Créer un compte avec email valide	Compte créé, redirection connexion
Inscription	Email déjà existant	Message d'erreur "email déjà utilisé"
Connexion	Email + mot de passe corrects	Redirection vers espace selon rôle
Connexion	Mot de passe incorrect	Message d'erreur
Commande	Commander un menu (6+ pers.)	Commande enregistrée avec récapitulatif
Commande	Réduction > 5 personnes	Prix total avec -10%
Contact	Envoyer formulaire complet	Message de succès
Contact	Champ obligatoire manquant	Message d'erreur sur le champ
Admin	Accéder au dashboard	Stats affichées (users, CA, commandes)
Employé	Changer statut commande	Statut mis à jour