ECF - Développeur Web et Web Mobile

Vite & Gourmand - Application Web de Traiteur Événementiel

Jamesy MUKUNA MUKENKETAYI

Session Juin-Juillet 2026

Table of Contents

Fonctionnalités suivies : - Analyse des besoins & MCD - Backend Laravel API REST - Frontend React SPA - Authentification Sanctum - CRUD Menus, Commandes, Avis - Fonctionnalités RGPD - Tests automatisés - Déploiement production

Lien du Déploiement

Application en production : https://vite-gourmand.fr

Infrastructure : - VPS OVH (Ubuntu 22.04) - Serveur web : Nginx - Base de données : PostgreSQL 14 - SSL/HTTPS : Let's Encrypt - Backend : Laravel 10 + PHP 8.2 - Frontend : React 18 (build statique)

Login et Mot de Passe Administrateur

Email : admin@demo.fr

Mot de passe : Password123!

Autres comptes de test : - **Employé** : employe@demo.fr / Password123! - **Client** : client@demo.fr / Password123!

PARTIE 1 : ANALYSE DES BESOINS

Résumé du Projet

Vite & Gourmand est une application web développée pour un service de traiteur événementiel souhaitant digitaliser son activité. Le projet répond au besoin de créer une plateforme permettant aux clients de consulter et commander des menus en ligne, tout en offrant aux employés et administrateurs des outils de gestion complets.

L'application permet aux visiteurs de parcourir un catalogue de menus classés par thème (bio, gastronomique, végétarien) et régime alimentaire (normal, végétarien, vegan, sans gluten). Les clients authentifiés peuvent passer des commandes en sélectionnant un menu, une date de livraison, une adresse et une quantité. Ils ont accès à un historique de leurs commandes avec suivi en temps réel du statut (en attente, validée, en préparation, livrée) et peuvent laisser des avis après livraison.

L'interface employé centralise la gestion des commandes avec possibilité de modifier les statuts selon un workflow défini, et de valider ou rejeter les avis clients avant publication. L'administrateur dispose d'un accès complet pour gérer les menus, plats, allergènes et utilisateurs.

Une attention particulière a été portée à la conformité RGPD avec consentement obligatoire à l'inscription, export des données personnelles au format JSON, et droit à l'oubli permettant la suppression définitive du compte. La sécurité est renforcée par une authentification par tokens, validation stricte des données, HTTPS obligatoire et rate limiting.

Le projet a été développé en 12 jours, déployé en production sur un VPS sécurisé avec certificat SSL, et comprend 15 tests automatisés garantissant la qualité du code. L'application est opérationnelle et prête pour une utilisation réelle.

Cahier des Charges

Objectif principal : Permettre la commande en ligne de menus pour événements

Fonctionnalités essentielles : - Catalogue de menus avec filtres - Système de commande en ligne - Gestion du workflow de commandes - Système d'avis clients avec modération - Conformité RGPD complète - Interface d'administration

Contraintes techniques : - Application web responsive (mobile et desktop) - Sécurité renforcée (HTTPS, authentification, validation) - Performance optimale (temps de réponse < 2 secondes) - Conformité RGPD obligatoire

Livrables : - Application web fonctionnelle en production - Code source versionné sur GitHub - Documentation technique complète - Tests automatisés

PARTIE 2 : SPÉCIFICATIONS TECHNIQUES

Technologies Utilisées et Justifications

Frontend : React 18

Justification du choix : - Framework JavaScript moderne très demandé sur le marché du travail - Approche par composants réutilisables facilitant la maintenance - Virtual DOM assurant des performances optimales - Écosystème riche (React Router pour la navigation, Axios pour les requêtes HTTP) - Large communauté et documentation abondante - Permet de créer une Single Page Application (SPA) offrant une expérience utilisateur fluide

Bibliothèques complémentaires : - **React Router v6** : Gestion de la navigation et des routes - **Axios** : Communication HTTP avec le backend - **Context API** : Gestion d'état global pour l'authentification

Backend : Laravel 10

Justification du choix : - Framework PHP robuste et mature, standard de l'industrie - Eloquent ORM simplifiant les requêtes de base de données - Système de validation intégré (FormRequests) - Laravel Sanctum pour l'authentification API par tokens - Policies pour la gestion fine des autorisations - Migrations pour le versionnement du schéma de base de données - Artisan CLI facilitant les tâches courantes - Documentation exhaustive en français et anglais

Composants Laravel utilisés : - **Sanctum** : Authentification stateless par tokens pour l'API - **FormRequests** : Validation centralisée et réutilisable des données - **Policies** : Contrôle d'accès granulaire selon les rôles - **Eloquent ORM** : Manipulation élégante de la base de données

Base de Données : PostgreSQL 14

Justification du choix : - SGBD relationnel open-source performant et fiable - Respect strict des standards SQL et de l'intégrité des données - Gestion avancée des types de données (JSON, Arrays) - Excellent support des contraintes et des index - Performance supérieure sur les requêtes complexes - Stabilité en production reconnue

Serveur Web : Nginx

Justification du choix : - Serveur web performant et léger - Excellent en tant que proxy inverse - Configuration flexible et simple - Gestion native du HTTPS - Performances supérieures à Apache pour les fichiers statiques - Faible consommation de ressources

Environnement de Travail

Configuration Locale (Développement)

Système d'exploitation : macOS (développement) / Ubuntu 22.04 (production)

Backend Laravel : - PHP 8.2 avec Composer 2.2.6 - PostgreSQL 14 local - Serveur de développement : `php artisan serve` (port 8000) - Variables d'environnement : `.env` avec configuration locale

Frontend React : - Node.js v18 avec npm - Create React App pour le scaffolding - Serveur de développement : `npm start` (port 3000) - Variables d'environnement : `.env.local` avec `REACT_APP_API_URL=http://localhost:8000/api`

Outils de développement : - **IDE** : Visual Studio Code - **Extensions** : PHP Intelephense, ESLint, Prettier - **Git** : Versionnement du code avec commits réguliers - **Postman** : Tests manuels de l'API - **Browser DevTools** : Débogage frontend

Structure des Projets

Backend :

```
backend/
├── app/
│   ├── Http/Controllers/Api/
│   ├── Models/
│   ├── Policies/
│   └── Http/Requests/
├── database/
│   └── migrations/
├── routes/
│   └── api.php
├── tests/
└── .env
```

Frontend :

```
frontend/
├── src/
│   ├── components/
│   ├── pages/
│   ├── contexts/
│   ├── services/
│   └── App.js
├── public/
└── .env.production
```

Mécanismes de Sécurité

Sécurité Frontend

Validation côté client : - Vérification en temps réel des formulaires (email, mot de passe, dates) - Messages d'erreur explicites avant soumission - Désactivation des boutons pendant les requêtes (prévention double-soumission)

Protection des routes : - Composant `PrivateRoute` vérifiant l'authentification - Redirection automatique vers `/login` si non authentifié - Vérification du rôle utilisateur pour les routes admin/employé

Gestion sécurisée du token : - Stockage dans `localStorage` (alternative : cookies httpOnly) - Token ajouté automatiquement via intercepteur Axios - Suppression du token à la déconnexion

Protection XSS : - React échappe automatiquement les données affichées - Validation des entrées utilisateur avant affichage

Sécurité Backend

Authentification robuste : - Laravel Sanctum avec tokens uniques par session - Mots de passe hashés avec bcrypt (12 rounds) - Validation stricte des mots de passe : - Minimum 10 caractères - Au moins une majuscule et une minuscule - Au moins un chiffre - Au moins un symbole

Validation des données : - FormRequests Laravel pour toutes les entrées - Validation côté serveur obligatoire (ne jamais faire confiance au client) - Types de données vérifiés (email, date, integer, etc.) - Règles métier appliquées (stock suffisant, date future, quantité minimum)

Autorisations : - Policies Laravel vérifiant les droits selon le rôle - Vérification systématique avant chaque action sensible - Middleware `auth:sanctum` sur toutes les routes protégées

Protection contre les attaques : - **CSRF** : Protection Laravel intégrée - **SQL Injection** : Eloquent ORM avec requêtes préparées - **XSS** : Échappement automatique dans Blade (non utilisé ici) et React - **Rate Limiting** : - 5 tentatives de connexion par minute - 60 requêtes API par minute - **Mass Assignment** : `$fillable` défini dans chaque modèle

HTTPS obligatoire : - Certificat SSL Let's Encrypt - Redirection automatique HTTP → HTTPS - Headers de sécurité : - `Strict-Transport-Security` (HSTS) - `X-Frame-Options: DENY` - `X-Content-Type-Options: nosniff` - `X-XSS-Protection: 1; mode=block`

Middleware de sécurité personnalisé :

```
class SecurityHeaders
{
    public function handle($request, Closure $next)
    {
        $response = $next($request);
        $response->headers->set('X-Frame-Options', 'DENY');
        $response->headers->set('X-Content-Type-Options', 'nosniff');
        $response->headers->set('X-XSS-Protection', '1; mode=block');
        return $response;
    }
}
```

Veille Technologique : Vulnérabilités de Sécurité

Sujet : Vulnérabilités d'Injection SQL dans les ORM

Date de veille : Février 2025

Contexte : Bien que Laravel Eloquent protège contre les injections SQL grâce aux requêtes préparées, j'ai effectué une veille sur les cas où des vulnérabilités peuvent subsister.

Découvertes principales :

1. **Utilisation de `whereRaw()` et `DB::raw()`** :
   - Ces méthodes permettent d'injecter du SQL brut
   - Sans échappement approprié, elles sont vulnérables
   - **Solution appliquée** : Éviter `whereRaw()` ou utiliser des bindings
2. **Mass Assignment Vulnerability** :
   - Exploitation de `$fillable` mal configuré
   - Permet de modifier des colonnes non autorisées (ex: `role`, `is_admin`)
   - **Solution appliquée** : Définition stricte de `$fillable` dans chaque modèle
3. **Ordre des colonnes dans `orderBy()`** :
   - Input utilisateur directement dans `orderBy($request->sort)` est dangereux
   - **Solution appliquée** : Whitelist des colonnes autorisées

Application dans le projet : - Utilisation exclusive d'Eloquent sans SQL brut - `$fillable` défini explicitement dans tous les modèles - Validation des inputs avant toute requête de tri/filtrage

Sources : - OWASP Top 10 (https://owasp.org/www-project-top-ten/) - Laravel Security Best Practices - CVE Database pour Laravel

PARTIE 3 : RECHERCHE

Situation de Travail Nécessitant une Recherche

Contexte : Lors du déploiement de l'application sur le VPS, j'ai rencontré un problème critique : l'impossibilité de me connecter en SSH malgré la réinitialisation du mot de passe via l'interface OVH. Le serveur était accessible (ping réussi) mais refusait systématiquement l'authentification.

Problème rencontré :

```
ssh dev@37.59.124.193
Permission denied (publickey,password).
```

J'ai tenté plusieurs solutions : - Réinitialisation du mot de passe root via OVH - Vérification des permissions SSH - Tentative avec différents clients SSH

Recherche effectuée : N'ayant jamais rencontré ce problème auparavant, j'ai effectué une recherche en anglais sur Stack Overflow et la documentation Ubuntu.

Source principale : - **Stack Overflow** : "Ubuntu VPS SSH access denied after password reset" - URL : https://stackoverflow.com/questions/tagged/ssh+ubuntu - **Ubuntu Server Documentation** : "How to recover SSH access using rescue mode" - URL : https://ubuntu.com/server/docs

Solution trouvée : Utiliser le mode rescue (mode secours) du VPS pour accéder au système de fichiers et reconfigurer l'accès SSH.

Extrait Anglophone et Traduction

Extrait Original (Anglais)

Source : Ubuntu Server Documentation - SSH Recovery

> "If you've lost SSH access to your Ubuntu server, you can use rescue mode to regain control. Boot your server into rescue mode from your hosting provider's control panel. Once in rescue mode, mount your system partition and chroot into it. From there, you can reset passwords using the `passwd` command or modify SSH configuration files. After making changes, unmount the partition, exit rescue mode, and reboot normally."
>
> "Common steps in rescue mode: 1. Boot into rescue mode 2. Mount the system partition: `mount /dev/sda1 /mnt` 3. Chroot into the mounted system: `chroot /mnt` 4. Reset the user password: `passwd username` 5. Exit chroot: `exit` 6. Unmount: `umount /mnt` 7. Reboot into normal mode"

Traduction Française

Source : Documentation Serveur Ubuntu - Récupération SSH

> “Si vous avez perdu l’accès SSH à votre serveur Ubuntu, vous pouvez utiliser le mode de secours pour reprendre le contrôle. Démarrez votre serveur en mode secours depuis le panneau de contrôle de votre hébergeur. Une fois en mode secours, montez votre partition système et utilisez chroot pour y accéder. De là, vous pouvez réinitialiser les mots de passe avec la commande `passwd` ou modifier les fichiers de configuration SSH. Après avoir effectué les modifications, démontez la partition, quittez le mode secours et redémarrez normalement.”
>
> “Étapes courantes en mode secours : 1. Démarrer en mode secours 2. Monter la partition système : `mount /dev/sda1 /mnt` 3. Chroot dans le système monté : `chroot /mnt` 4. Réinitialiser le mot de passe utilisateur : `passwd nom_utilisateur` 5. Quitter chroot : `exit` 6. Démonter : `umount /mnt` 7. Redémarrer en mode normal”

Application Concrète

Commandes exécutées :

```
# 1. Passage en mode rescue via interface OVH
# 2. Connexion SSH en mode rescue
ssh root@37.59.124.193

# 3. Montage du système
mount /dev/sda1 /mnt

# 4. Chroot
chroot /mnt

# 5. Identification de l'utilisateur existant
grep /bin/bash /etc/passwd
# Résultat : dev:x:1000:1000::/home/dev:/bin/bash

# 6. Réinitialisation du mot de passe
passwd dev

# 7. Sortie et redémarrage
exit
umount /mnt
reboot

# 8. Connexion réussie
ssh dev@37.59.124.193
```

Résultat : L’accès SSH a été restauré avec succès et j’ai pu poursuivre le déploiement de l’application.

Apprentissage : Cette expérience m’a permis de comprendre le fonctionnement du mode rescue, l’importance de la documentation en anglais (source principale d’information technique), et les mécanismes de récupération système sous Linux.

PARTIE 4 : INFORMATIONS COMPLÉMENTAIRES

Architecture Globale

Type : Application web 3-tiers avec API REST

Schéma simplifié :

```
Client (Navigateur)
    ↓ HTTPS
Frontend React (SPA)
    ↓ API REST (JSON)
Backend Laravel 10
    ↓ Eloquent ORM
PostgreSQL 14
```

Métriques du Projet

Durée de développement : 12 jours (6-18 février 2025)

Lignes de code : ~8000 lignes (Backend + Frontend)

Base de données : - 13 tables - Relations 1:N et N:N - Normalisation 3NF

API REST : - 46 routes - Authentification Sanctum - Rate limiting

Tests : - 15 tests automatisés (PHPUnit) - Couverture : authentification, CRUD, RGPD

Pages/Composants : - 15+ pages React - 20+ composants réutilisables

Points Forts du Projet

Technique : - Stack moderne et demandée sur le marché - Code organisé et maintenable - Tests automatisés - Déploiement production sécurisé

Fonctionnel : - Application complète et utilisable - Expérience utilisateur fluide - Gestion complète du workflow métier

Sécurité : - Authentification robuste - Validation stricte - HTTPS + headers sécurisés - Conformité RGPD

Difficultés Rencontrées et Solutions

1. Accès SSH au VPS - Problème : Mot de passe refusé - Solution : Mode rescue + chroot + passwd

2. Configuration API en Production - Problème : URL localhost hardcodée - Solution : Variables d'environnement `.env.production`

3. Routing Nginx pour l'API - Problème : Routes 404 - Solution : Configuration `rewrite` correcte

4. Dépendances Composer - Problème : Extensions manquantes - Solution : `--ignore-platform-req` en production

Perspectives d'Évolution

Court terme : - Notifications email (confirmation commande, changement statut) - Système de paiement en ligne (Stripe) - Amélioration du dashboard avec graphiques

Moyen terme : - Application mobile React Native - API publique pour partenaires - Module de gestion de stock avancé

Long terme : - Intelligence artificielle pour recommandations - Multi-langue (i18n) - Marketplace multi-traiteurs

---

FIN DU DOCUMENT ECF

Jamesy MUKUNA MUKENKETAYI
Session Juin-Juillet 2026
STUDI - Paris