Documentation technique – Vite & Gourmand

1. Contexte et objectifs

Application web de traiteur événementiel. Les besoins principaux sont :
afficher les menus au public, permettre la commande, et offrir un back-office pour employés et administrateurs.

2. Choix techniques (et justification)

- **Front** : HTML/CSS/JS statique + Bootstrap.
  Choix simple, rapide à déployer, facile à évaluer pour un correcteur.
- **Back** : Symfony (PHP 8.2+).
  Framework robuste, structure claire, sécurité et routing intégrés.
- **Base relationnelle** : PostgreSQL.
  Compatible avec les contraintes ECF et modèle relationnel adapté.
- **NoSQL** : MongoDB (optionnel).
  Utilisé pour statistiques admin, avec fallback Postgres si Mongo absent.
- **Déploiement** : Heroku (API + Postgres) + Netlify (front).
  Déploiement fiable, accessible, et facile à présenter.

3. Architecture

L'application est découpée en deux blocs :

- **Front statique** : pages HTML, appels API via `fetch`.
- **API Symfony** : endpoints REST pour menus, commandes, avis, auth, admin.

Flux simplifié :
Client → API Symfony → PostgreSQL
Client → API Symfony → MongoDB (stats, si disponible)

4. Configuration de l’environnement

Prérequis principaux :

- PHP 8.2+ et Composer
- PostgreSQL
- (Optionnel) MongoDB + extension `mongodb`

Variables d’environnement :

- `DATABASE_URL` : connexion Postgres
- `JWT_SECRET` : clé JWT
- `MONGO_URL`, `MONGO_DB` : stats Mongo (optionnel)

Les étapes d’installation et de lancement sont détaillées dans :
`/Users/imaschool/Desktop/vite-gourmand-ecf/README.md`
et `/Users/imaschool/Desktop/vite-gourmand-ecf/docs/DEPLOYMENT.md`.

5. Modèle conceptuel de données (résumé)

Entités principales :

- `utilisateur` et `roles`
- `menu`, `menu_image`
- `plat`, `allergene`, `menu_plat`, `plat_allergene`
- `commande`, `commande_statut_historique`, `commande_annulation`
- `avis`, `contact_message`, `password_reset`, `email_log`

Relations clés :

- Un **menu** contient plusieurs **plats** et plusieurs **images**.
- Un **plat** peut avoir plusieurs **allergènes**.
- Une **commande** est liée à un **utilisateur** et un **menu**.
- Les **statuts de commande** sont historisés.

6. Diagrammes

Les diagrammes (use case + séquence) sont fournis ici :
`/Users/imaschool/Desktop/vite-gourmand-ecf/docs/diagramme.pdf`.

7. Règles de gestion principales

- Commande possible uniquement si le menu est actif.
- Annulation possible tant que la commande n'est pas "acceptée".
- Avis possible uniquement quand la commande est "terminée".
- Workflow de statut :
  `en_attente → accepte → en_preparation → en_livraison → livre → terminee`
  Variante avec matériel prêté : `livre → attente_retour_materiel → terminee` .

8. Sécurité (résumé)

- Mots de passe hashés (bcrypt).
- Authentification JWT + rôles.
- Validation des champs côté serveur.
- Email reset protégé par token.

Détails : `/Users/imaschool/Desktop/vite-gourmand-ecf/docs/SECURITE.md` .

9. Déploiement (résumé)

- **API** : Heroku (PHP) avec `backend_symfony/` .
- **Base** : Heroku Postgres.
- **Front** : Netlify (statique).

Détails : `/Users/imaschool/Desktop/vite-gourmand-ecf/docs/DEPLOYMENT.md` et `/Users/imaschool/Desktop/vite-gourmand-ecf/docs/DEPLOYMENT.pdf` .

10. Limites et pistes d'amélioration

- Upload d'images réel (actuellement URLs/galerie statique).
- Emails réels via un service SMTP.
- Tests automatisés plus complets (API et UI).