DOCUMENTATION TECHNIQUE

Vite & Gourmand

1. Réflexions initiales technologiques :

Dans le cadre du projet *Vite & Gourmand*, une application web de gestion de commande de menus traiteur, plusieurs choix technologiques ont été étudiés avant le démarrage du développement :

Analyse des besoins :

L'application doit permettre :

- La visualisation des prestations proposées
- La création de compte utilisateurs ainsi que la connexion et déconnexion
- La gestion des utilisateurs (clients, employés, administrateurs)
- La gestion des menus et plats
- La gestion des commandes avec suivi des états
- La gestion des avis clients
- L'affichage de statistiques

Le projet nécessite donc :

- Un environnement web adapté et responsive
- Une architecture structurée
- Une base de données relationnelle
- Une base de données non relationnelle pour les statistiques
- Une séparation claire entre logique métier et affichage

Choix des technologies :

Front-end : **HTML**, **CSS** (Bootstrap) et **JavaScript**

- Affichage à partir de navigateur web (ordinateur et mobile)
- Mise en forme des pages facilitée et moderne
- Affichage dynamique de contenus

Back-end : **PHP** avec utilisation de PDO et Framework **Symfony**

- Application du modèle MVC (Modèle Vue Contrôleur)
- Adapté aux projets web professionnels
- Sécurise l'authentification et les formulaires
- Assure maintenabilité et évolutivité

Bases de données : **MariaDB** (MySQL) et **MongoDB** (NoSQL)

- *MariaDB* pour son côté relationnel qui gère les relations prédéfinies entre les données, organisées en tables, colonnes et lignes.

- *MongoDB* pour sa nature non relationnelle orienté documents qui stocke les données (paires de champs et valeurs) dans des documents de manière comparable aux objets JSON (JavaScript Object Notation).

Déploiement : **AlwaysData**

- Permet un hébergement simple, fiable et compatible avec le stack PHP/Symfony/MariaDB.
- Possibilité d'installer des extensions non disponible par défaut comme « mongodb ».

2. Configuration de l'environnement de travail :

Poste de travail :

- Ordinateur sous système d'exploitation **Windows**
- Navigateur web principal : **Google Chrome**
- Suite logiciel **XAMPP** (Apache, MySQL, FileZilla)
- Environnement de développement : **PHPStorm 2024.3**
- Gestion de versions : **Git** (Github)

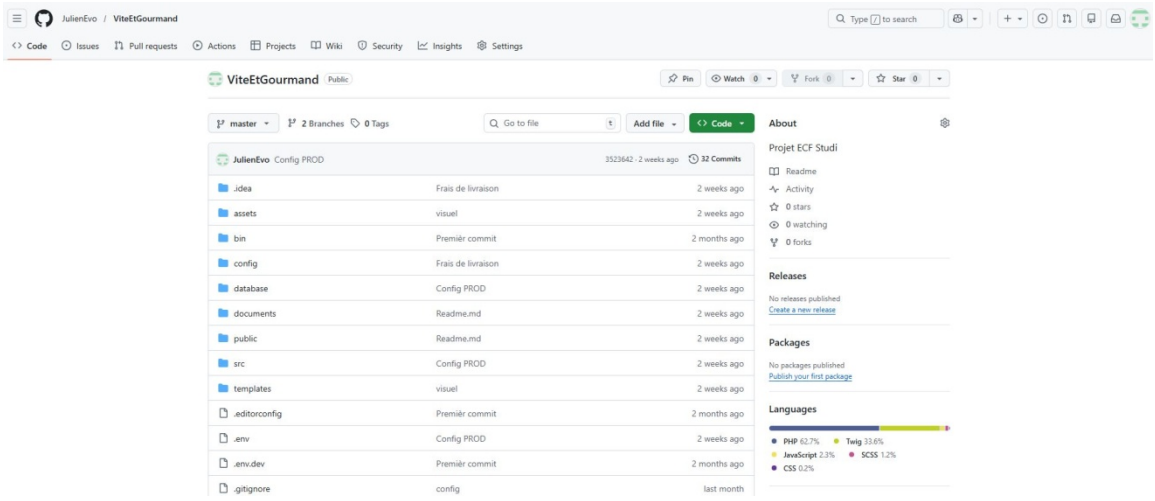

Figure 1 - Github (branche master)

Environnement local :

- **PHP 8.2**
- **Symfony 7.4**

- **Composer** pour la gestion des dépendances
- **Apache 2.4** (avec VirtualHost)
- Paramétrage des variables d'environnent dans le fichier *.env*

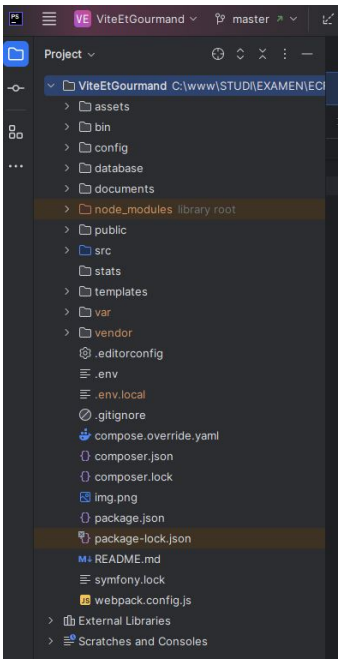

Figure 2 - Structure Symfony (PhpStorm)

Bases de données :

- **MariaDB 10.4** pour la base de données relationnelle
- **PhpMyAdmin** comme outil d'administration de MariaDB
- Création de la base de données via import d'un script SQL

- **MongoDB 2.1** pour la base de données non relationnelle
- **MongoDB Compass** comme outil d'administration de MoongoDB

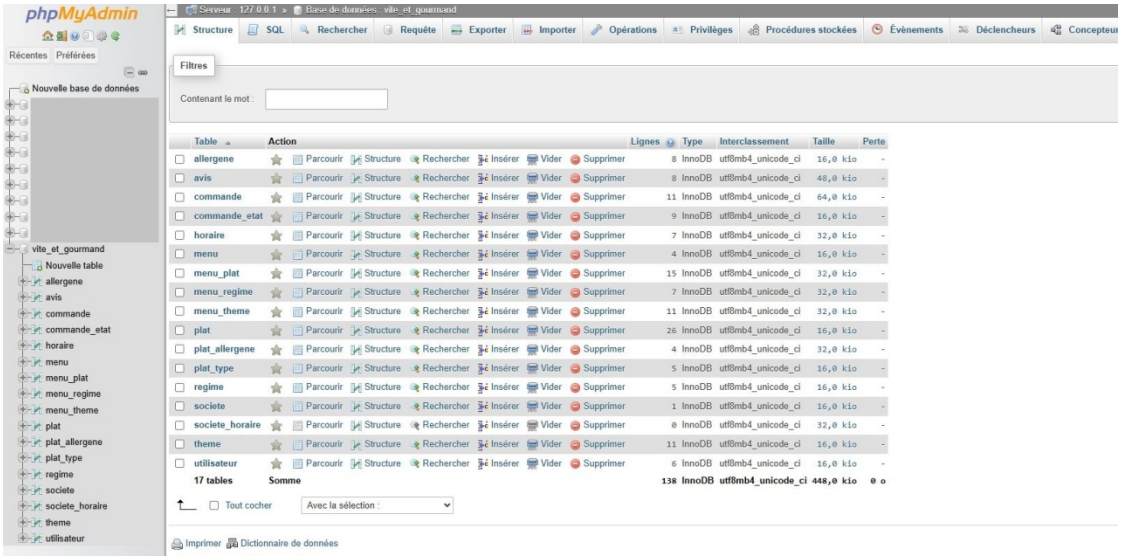

Figure 3 – PhpMyAdmin

3. Modèle conceptuel de données (MCD) :

Le MCD (Modèle Conceptuel de Données) représente l'ensemble des entités ainsi que leurs relations et permet de visualiser clairement :

- Les entités principales (ex : utilisateur, commande, menu, plat), représentées par des rectangles avec leurs attributs
- Les associations entre elles (ex : un utilisateur peut passer plusieurs commandes), identifiées par des ovales
- Les cardinalités pour chaque relation (1:1, 0:N, 1:N)

Cette représentation a été réalisée avec l'application **Looping**, en respectant la méthode *Merise.*

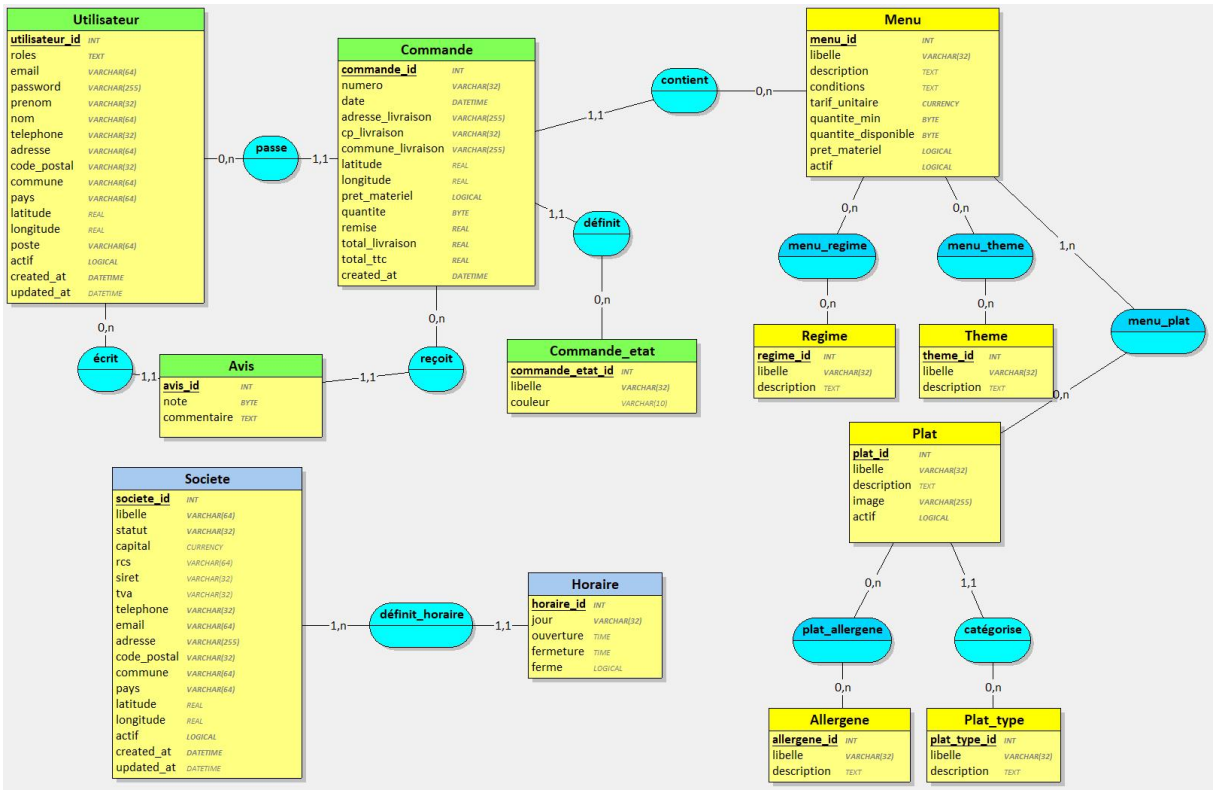

Figure 4 - MCD (Export Looping)

4. Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation permet de représenter les interactions entre les différents acteurs (visiteurs, utilisateurs, employé et administrateurs) et les fonctionnalités proposées par l'application qui donne une vision globale sur les droits de chaque acteur.

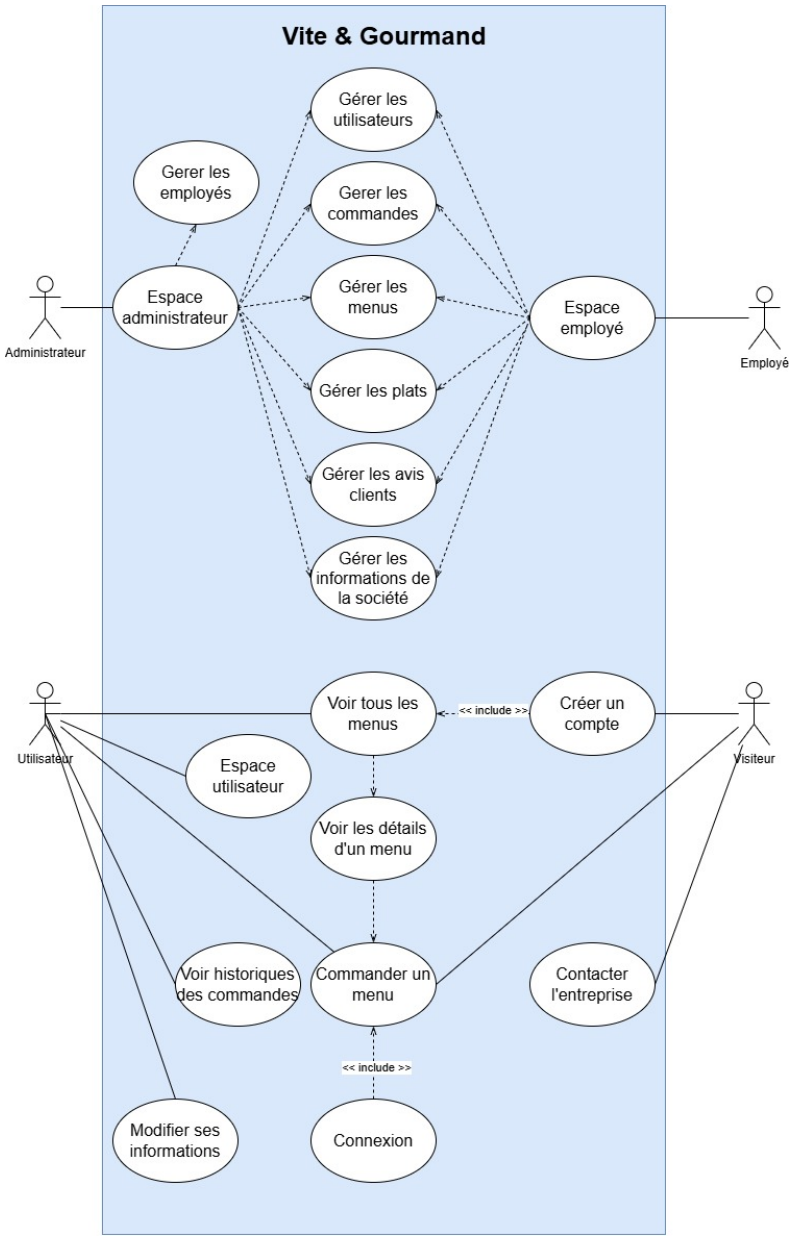

Figure 5 - Diagramme de cas d'utilisation

5. Diagramme de séquence – Validation d'une commande :

Le diagramme de séquence ci-dessous illustre le déroulement technique lorsqu'un client valide une commande dans l'application.

Il met en évidence la chronologie des interactions entre les différents composants du système, et garantit une compréhension précise du fonctionnement interne.

Le scénario fait intervenir :

- **Le client** : Utilisateur connecté avec ses identifiants
- **Le navigateur** : Interface entre l'utilisateur et l'application

- **CommandeController** : Contrôleur du modèle MVC, qui gère la logique métier
- **CommandeRepository** : Permet les interactions avec la base de données relationnelle
- **Bases de données** : Enregistrement persistant des données

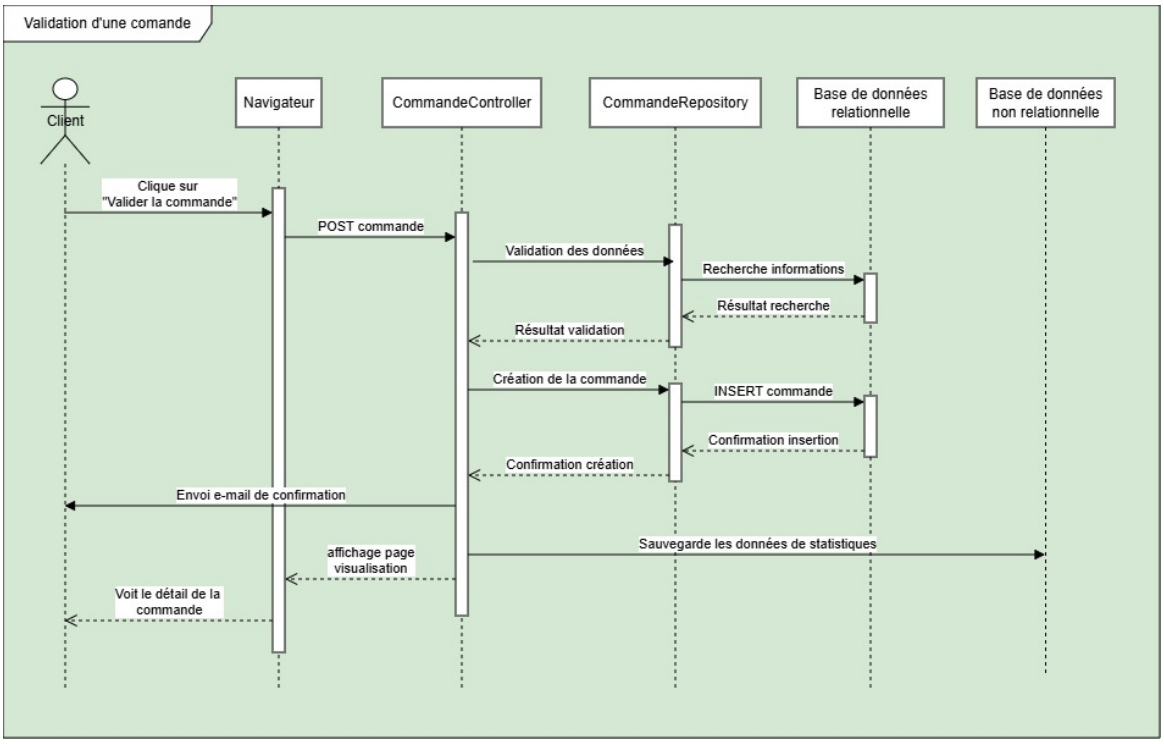

Figure 6 - Diagramme de séquence (Validation d'une commande)

Déroulement du processus :

1. Le client clique sur le bouton « valider la commande »
2. Le navigateur envoie une requête HTTP de type **POST** vers la route */commande_validation*
3. Le *CommandeController* réceptionne la requête et effectue la validation des données en appelant le *CommandeRepository* afin d'interroger la base de données
4. Le repository renvoi les résultats des requêtes au contrôleur pour validation
5. *CommandeController* crée la commande et la transmet au repository pour l'enregistrement
6. Le service exécute une requête d'insertion (INSERT) en base de données
7. La base de données confirme l'enregistrement
8. Le contrôleur envoi un e-mail de confirmation au client
9. Le contrôleur enregistre des informations de statistiques en base de données NoSQL
10. Le contrôleur génère une réponse HTTP 200 avec la vue de visualisation de la commande
11. Le navigateur affiche la page de visualisation de la commande au client

6. Diagramme de séquence – Validation d'une commande :

Démarche générale :

Le projet a été développé en environnement local avant d'être déployé en production. Cette démarche permet :

- De développer et tester sans impacter le site en production
- De corriger les erreurs avant mise en production
- De garantir la stabilité de l'application

Le déploiement a été réalisé sur un hébergement mutualisé AlwaysData (www.alwaysdata.com/fr)

Environnement de développement local :

Le développement a été effectué avec :

- PHP 8.2
- Symfony 7.4
- Composer pour la gestion des dépendances
- Apache (VirtualHost configuré vers le dossier */public*)
- MariaDB
- MongoDB
- Git pour la gestion de versions

Configuration de la base de données :

La base de données (bdd) MariaDB est configurée via des variables d'environnement dans deux fichiers :

- *.env* : configuration par défaut
- *.env.local* : configuration contenant les informations de connexion (non versionné)

La bdd peut être initialisée en exécutant un script SQL contenant :

- La création de tables
- L'insertion de données initiales permettant le bon fonctionnement de l'application

Cela permet une réinitialisation rapide de l'environnement.

Sécurisation de l'environnement :

Plusieurs bonnes pratiques ont été appliquées :

- Les secrets ne sont pas versionnés
- Utilisation du système de sécurité Symfony (authentification et rôles)
- Protection des routes selon les rôles attribués
- Utilisation de PDO dans les requêtes SQL pour éviter les injections

Déploiement en production :

L'hébergeur a été réalisé sur **AlwaysData** pour :

- Sa compatibilité avec PHP 8 et MariaDB
- La gestion simple des variables d'environnement
- La configuration facile des différents services (base de données, accès distant, installation d'extensions)

Étapes du déploiement :

- Création du compte *vite-et-gourmand* et ajout d'un nouveau site
- Configuration de l'adresse : *vite-et-gourmand.alwaysdata.net*
- Configuration du répertoire racine et de la version de PHP
- Ajout des variables d'environnement
- Transfère des fichiers via FTP (FileZilla)
- Test fonctionnels post-déploiement

Tests après mise en production :

Après le déploiement, plusieurs test ont été effectués :

- Création de compte utilisateur
- Connexion / Déconnexion
- Visualisation des menus
- Création de commande
- Accès aux espaces utilisateur et administrateur
- Vérification de l'enregistrement en base de données

Ces vérifications permettent de garantir la conformité entre l'environnement local et production.

Les instructions détaillées d'installation sont disponible dans le fichier README.md du projet. La présente section synthétise la démarche technique adoptée.