Documentation technique

Vite & Gourmand

ECF Studi TP Développer Web et Web Mobile

Sommaire

1-Réflexion initiale Technologique :

Pour la création du site « Vite & Gourmand », j'ai tout d'abord analysé les besoins fonctionnels et contraintes techniques du projet.

Le but étant de développer une plate-forme de vente de menus en ligne avec livraison à domicile. Le site doit pouvoir proposer plusieurs menus avec diverses caractéristiques.

Il y a également plusieurs interfaces avec des fonctionnalités différentes. Les clients doivent pouvoir consulter et filtrer une liste de menus selon différents critères (prix, thème, régime …), passer commande avec une gestion de panier et calcul des frais de livraison. Le client doit également pouvoir visualiser l'évolution de sa commande et y déposer un avis une fois terminée. Les employés doivent gérer les menus (création, modification, suppression), les commandes (modification, actualisation), les avis (validé, rejeté) et les horaires d'ouverture (modification). L'administrateur a les mêmes fonctionnalités qu'un employé mais il a également la création et gestion des comptes employés ainsi que l'accès aux statistiques des menus.

Le site doit être responsive, conforme aux réglementations RGPD, envoyer des notifications par mail et respecter les bonnes pratiques de sécurité (cryptage des mots de passe, protection contre les injections SQL).

Front-end : Bootstrap

Pour la partie front-end, j'ai choisi le framework Bootstratp. Pour sa gratuité et son accès à sa vaste bibliothèque d'éléments réutilisables. Il s'intègre parfaitement avec les langages HTML, CSS et JavaScript. Bootstrap favorise l'adaptabilité pour le responsive design grâce à son système de grille. Il est facile d'utilisation et possède une documentation officielle complète et structurée. Tout ceci m'a permis de gagner du temps en développement, une adaptation rapide aux différents supports et une expérience utilisateur optimisée.

Back-end : Symfony

Pour le back-end j'ai choisi le framework Symfony. Ce choix s'appuie sur plusieurs critères techniques et fonctionnels. Symfony est un framwork PHP open-source sans coût de licence, reconnu pour sa robustesse et sa fiabilité dans le développement d'application professionnelles. Il répond à des exigences de performance et de sécurité nécessaires pour mon site.

Pour mon projet j'ai utilisé : Doctrine ORM, Symfony Sécurity et API Platform.

Base de données relationnelle : MySQL

Pour gérer les données de l'application, j'ai opté pour MySQL, un système de gestion de base de données relationnel (SGBDR) open-source et accessible librement. Pour le site j'ai donc choisi une de base de données relationnelle SQL, celle-ci permet de stocker, récupérer et gérer les données. Elle garantie l'intégrité et la cohérence des données grâce aux relations entre les tables et à l'utilisation de clés primaires et étrangère.

Cette structure me permet de lier facilement les commandes aux clients et aux menus via des relations. De gérer automatique la mise à jour des stocks des plats. Tracer les modifications des commandes effectuées par les employées et l'administrateur.

Pour le développement local, j'utilise XAMPP, qui intègre MySQL, Apache et PHP. Pour l'administration de la base de données, j'ai choisi Adminer plutôt que car il est léger, rapide et offre une interface épurée pour gérer les tables et visualiser les données.

Base de données non relationnelle : Mongodb :

Comme base de données NoSQL, j'ai choisi MongoDBo qui est open-source et gratuit pour la gestion des statistiques. Elle utilise un système de données non relationnel, orienté document et un langage de requête flexible. Cela permet de stocker et d'analyser facilement les données du site telles que : le nombre de commandes par menu ainsi que les montants générés, afin d'effectuer divers calculs statistiques (menus les plus vendus, chiffre d'affaires).

Déploiement :Docker et Railway :

Afin de garantir la réussite du déploiement de mon application, j'ai opté pour la conteneurisation en utilisant Docker Desktop. Cette étape importante m'a permis de tester mon application dans un environnement identique à celui de la production.

Pour la mise en ligne, j'ai choisi Render pour plusieurs critères : sa gratuité, son intégration continue via une synchronisation directe avec mon dépôt GitHub, et sa gestion multi-services. Cette plateforme centralise, au sein d'une même interface, le déploiement du front-end, du back-end.

2-Configuration de l'environnement de travail:

Pour la configuration de mon environnement de travail, j'ai opter pour la démarche ci-dessous :

Figma :

J'ai utilisé Figma pour réaliser le maquettage (wireframes et mockup) de mon application. Cette étape m'a permit de réaliser la charte graphique et travailler sur l'ergonomie avant de commencer le développement, afin de garantir une interface cohérente et logique pour les utilisateurs.

Miro :

Afin de mieux identifier les logiques métier, les interactions entre les différents utilisateurs et les actions de chacun, j'ai utilisé Miro pour réaliser l'ensemble de mes diagrammes UML (Cas d'utilisation, Classe et Séquence). Cette étape de modélisation a été essentielle pour cerner précisément les objectifs, les contraintes techniques et les finalités de chaque acteur au sein de l'application.

Éditeur de code : Visual Studio Code

Mon choix s'est porté sur Visual Studio Code (VS Code) comme éditeur de texte gratuit et léger. Il propose un ensemble de fonctionnalités riches ainsi qu'un grand nombre d'extensions, qui m'ont permis de configurer un IDE adapté à la stack Symfony, et d'augmenter ma productivité.

-Intelephense, PHP et SonarQube : Pour la complétion intelligente, la navigation rapide et l'analyse statique du code afin d'éviter les erreurs de syntaxe.

-Live Sass Compiler : Pour la compilation en temps réel de mes styles CSS personnalisés.

-Extension Git intégrée : Pour la gestion de version simplifiée directement depuis l'éditeur.

L'ensemble de ces extensions m'a permis de réduire mon temps de débogage et d'assurer une meilleure qualité de code.

Git/ Ghithub

Pour suivre l'évolution de mon projet, j'ai choisi Git. Grâce à ce système de contrôle de version, j'ai pu suivre et gérer les modifications apportées au code. Pour

l'hébergement de mon dépôt distant, j'ai utilisé la plateforme GitHub, qui permet la sauvegarde de mes fichiers ainsi qu'une intégration et un déploiement continus (CI/CD). L'objectif à court terme est de parfaire ma maîtrise du système de branches pour isoler plus efficacement le développement de chaque fonctionnalité.

Front-end : Bootstrap

Pour la partie front-end, j'ai choisi le framework Bootstrap. Celui-ci est responsive, ce qui lui permet de s'adapter à tous les types d'écrans (mobiles, tablettes, ordinateurs). J'ai principalement exploité ses composants natifs tels que la barre de navigation (Navbar), les fenêtres modales (Modals) ainsi que le système de Grille pour structurer mes sections. Pour les ajustements visuels spécifiques à la charte graphique du site, j'ai complété Bootstrap avec des styles CSS personnalisés compilés en temps réel grâce à l'extension Live Sass Compiler.

Back-end : Symfony :

J'ai installer le projet via Symfony CLI, ce qui m'a permis d'exécuter un serveur local et de visualiser les résultats en temps réel pendant le développement.

J'ai structuré l'application autour des entités Doctrine, générées et migrées via les commandes : make : entity et doctrine:migration:migrate. La gestion des accès et des rôles (CLIENT, EMPLOYE, ADMIN) a été créé à l'aide du composant Security de Symfony. Enfin, j'ai utilisé API Platform pour exposer certaines routes en tant qu'endpoints REST, testés ensuite via Postman.

MySQL

Pour la gestion de la base de données relationnelle (SGBDR), j'ai choisi MySQL. Ce système est nativement compatible avec Doctrine ORM et le framework Symfony, ce qui assure une communication fluide entre le code et les données.

Cette structure permet de stocker les informations relatives aux utilisateurs, aux menus et aux commandes, tout en garantissant l'intégrité des relations entre ces différentes tables. Pour l'administration et l'exécution locale de la base de données, j'ai utilisé l'environnement XAMPP. Ce dernier fournit le service MySQL indispensable au stockage, ainsi que l'outil Adminer pour visualiser et gérer les schémas de données.

MongoDB NoSQL :

Pour la gestion des données non relationnelles, j'ai choisi MongoDB. Ce système s'intègre parfaitement à mon environnement de développement.

Je l'utilise spécifiquement pour l'agrégation de données afin d'établir des statistiques (chiffre d'affaires, volume de commandes, etc.). Sa structure flexible, basée sur des documents JSON, permet de croiser des informations hétérogènes plus rapidement qu'avec un modèle relationnel classique, facilitant ainsi l'analyse décisionnelle de l'application.

Tests :

Pour tester mon application j'ai utiliser :

-Postman pour tester les endpoints de l'application afin de vérifier la conformité des réponses du serveur.

-Mailtrap m'a permis de tester l'intégralité du système d'envoi d'e-mails (confirmations de commande, création de compte....) en toute sécurité.

-Docker pour garantir que mon application fonctionnera de manière identique sur mon poste de travail et sur le serveur de production, évitant ainsi les problèmes de compatibilité de versions.

-Tests Unitaires (PHPUnit) : Pour valider le comportement isolé de certaines parties du code, notamment en vérifiant la logique des getters et setters de mes entités.

-Tests Fonctionnels : Réalisés pour simuler le parcours utilisateur et vérifier que certaines routes de l'application répondent correctement aux requêtes.

3-Diagramme de classe :

La réalisation du diagramme de classe m'a permit de décrire précisément la structure de mon application en modélisant les classes, les attributs, les opération ainsi que les relations entre les objets.

Voir annexe fichier readme.md

4-Diagramme de cas d'utilisation:

Le diagramme d'utilisation m'a permit de visualiser les actions des utilisateurs ainsi qu'identifier les relations afin de simplifier le développement.

Voir annexe fichier Readme.md

5-Diagramme de séquence :

La réalisation du diagramme de séquence m'a permis d'identifier les interactions des objets dans un système au fil du temps. Il représente les échanges entre les objets et l'ordre dans lequel ces interactions se produisent.

Voir annexe fichier Readme.md

6-Documentation du déploiement :

Pour le déploiement de mon application, j'ai commencé par installer Docker Desktop. J'ai créé un fichier *docker-compose.yml* nommé « vite-gourmand-docker » afin de regrouper mon dossier front-end et back-end dans un même environnement conteneurisé. Cette étape importante m'a permis de valider le fonctionnement de mon site en conditions proches de la production, avant le déploiement réel.

Pour l'hébergement, j'ai choisi Render. Cette plateforme est simple d'utilisation et permet de se connecter à plusieurs environnements. Elle offre également une intégration native avec GitHub, ce qui facilite le déploiement continu.

Configuration des bases de données :

J'ai créé un compte MongoDB Atlas afin d'héberger mes données NoSQL, jusque-là gérées localement via MongoDB Compass. J'ai ensuite renseigné les variables correspondantes dans mon fichier *.env.local :*

MONGODB_URL=""
MONGODB_DB="viteetgourmand"

J'ai également créé un compte sur Aiven.io pour héberger ma base de données MySQL, puis j'ai renseigné l'URL de connexion dans le même fichier :
DATABASE_URL=""

J'ai ensuite effectué une migration afin de créer les tables de ma base de données

php bin/console doctrine:migrations:migrate

Déploiement sur Render :

J'ai créé un compte Render et l'ai connecté à mon compte GitHub afin de faciliter le déclenchement automatique des builds. J'ai sélectionné le dépôt `vite_gourmand_back` et configuré les variables d'environnement suivantes :

DATABASE_URL
MONGODB_URL
MONGODB_DB
APP_ENV : prod
APP_SECRET
MAILER_DSN
CLOUDINARY_URL
J'ai ensuite lancé le déploiement du back-end.

Import des données

Afin de migrer mes données de développement vers la production, j'ai exporté ma base SQL depuis Adminer et l'ai importée dans Aiven.io. J'ai réalisé la même opération pour MongoDB Compass vers Atlas.

Déploiement du front-end

Une fois le déploiement du back-end terminé, je suis retourné(e) sur Render et j'ai créé un nouveau site statique en sélectionnant le dépôt front-end. À l'issue du build, j'ai obtenu l'URL d'accès à mon application : https://vite-gourmand-front.onrender.com/

7-Conclusion:

Le projet « Vite et Gourmand », m'a permis de mettre en pratique l'ensemble des compétences acquises au cours de ma formation BTS.

Du choix des technologies jusqu'au déploiement de l'application, j'ai dû prendre des décisions techniques concrètes et les mettre en œuvre de façon autonome. La combinaison d'un front-end en Bootstrap, d'un back-end Symfony, et de deux bases de données complémentaires MySQL et MongoDB m'ont confrontée à des problématiques réelles de développement full-stack.

Ce projet m'a également permis de découvrir et de maîtriser des outils professionnels tels que Docker et Render, qui sont utilisés dans le monde du travail.

Si je devais prolonger ce projet, j'envisagerais d'améliorer l'expérience utilisateur en ajoutant plus de choix dans mes menus.

Diagramme de classe Vite et Gourmand

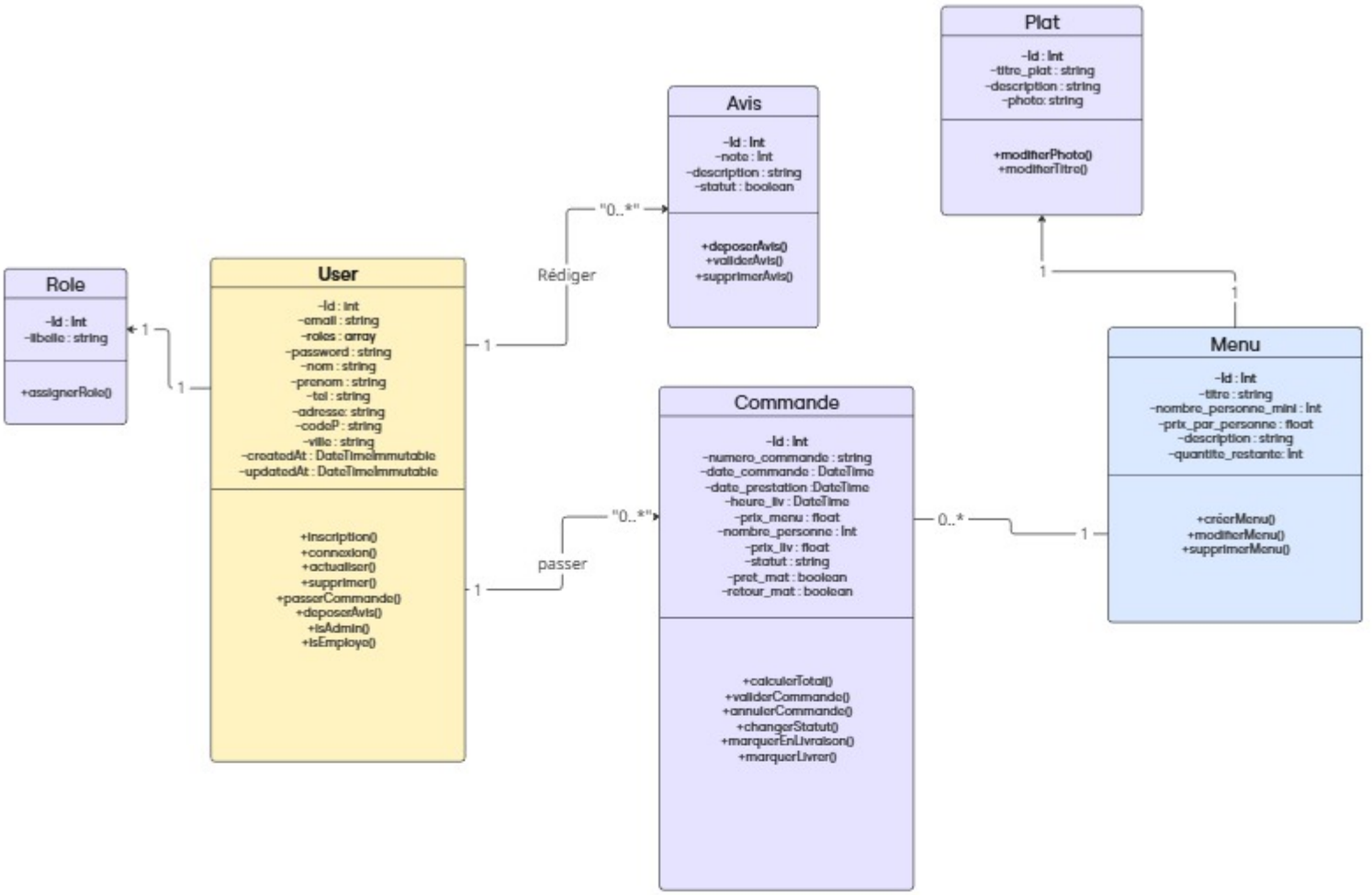

Diagramme de cas d'utilisation Vite et Gourmand

Diagramme de séquence Vite et Gourmand