

Tea'Coffee



Tea'Coffee
Élixir de feuille
et grain

Découvrez l'excellence chez Tea'Coffee :

Une sélection exquise de café et de thé pour
des moments de dégustation inoubliables.



Boutique en ligne de vente de thé et café

Projet réalisé dans le cadre du passage du Titre Professionnel de

Développeur Web et Web Mobile

Présenté par Alexandre Detroy

LaPlateforme_ Promo Toulon (2024)

SOMMAIRE

REMERCIEMENTS

..... 4

INTRODUCTION

..... 5

COMPÉTENCES DU RÉFÉRENTIEL COUVERTES PAR LE PROJET

..... 6

ACTIVITÉ TYPE N° 1 : Développer la partie front-end d'une application

web ou web mobile sécurisée 6

- Installer et configurer son environnement de travail 6

- Maquetter des interfaces utilisateur web ou web mobile 6

- Réaliser des interfaces utilisateur statiques web ou web mobile 6

- Développer la partie dynamique des interfaces utilisateur web ou web mobile 6

ACTIVITÉ TYPE N° 2 : Développer la partie back-end d'une application

web ou web mobile sécurisée 7

- Créer une base de données 7

- Développer les composants d'accès aux données 7

- Développer la partie back-end d'une application web ou web mobile

..... 7

RÉSUMÉ DU PROJET 8

CAHIER DES CHARGES 8

Objectifs 8

Conceptualisation de l'application 8

Liste des fonctionnalités 8

Mise en place de l'application 9

Arborescence du site 9

Maquette 10

Charte graphique et logo / Accessibilité 11

Utilisateurs et User Stories 11

Public visé 11

Rôles 11

User Stories	11
Évolutions futures	12
SPÉCIFICATIONS TECHNIQUES	12
Versioning	12
Choix technologiques	14
Front-End	14
Back-end	14
Accessibilité	15
Navigateurs compatibles	15
Types d'appareils	15
Architecture du projet	15
La Base de Données	16
MCD	16
MLD	16
Dictionnaire de données	17
MPD	18
Routes	19
Déploiement	21
Sécurité de l'application	21
Formulaires	22
GESTION DE PROJET	23
Présentation de l'équipe et organisation de travail	23
Programme de l'organisation	24
RÉALISATIONS PERSONNELLES	26
Rôle de la classe Render, rendu du contenu à l'utilisateur	26
VEILLE TECHNOLOGIQUE	29
CONCLUSION	30
ANNEXES	31

REMERCIEMENTS

Je tiens à remercier l'ensemble de mes formateurs lors des sessions ALC et DWWM.

Merci à Allan Busi pour son investissement constant dans ma formation, pour toutes les connaissances qu'il m'a apportées, et pour son soutien durant la formation, lors de ma demande de passage à la session DWWM, pendant la recherche de stage, et lors de la préparation des dossiers. Merci d'avoir été à l'écoute et de m'avoir fait découvrir et continuer d'apprendre le vaste milieu du développement web.

Merci à Nicolas DeGabriel pour son investissement tout au long de mes formations, pour sa bienveillance et son implication dans mon passage en DWWM, pour toutes les informations relatives au diplôme, pour être resté positif quelles que soient les conditions, et pour avoir appuyé ma demande de passage en session DWWM.

Merci à Fouzi Benzidane pour ses conseils et son implication dans ma première formation, où il nous a fait découvrir la programmation logicielle à travers des travaux pratiques comme la réalisation d'un jeu Puissance 4 en Python avec développement d'une IA, ou le développement d'un agenda en C++.

Merci à Massinissa Chaouchi pour son investissement et son implication durant les dernières semaines de formation, pour toutes les connaissances partagées et son écoute, pour son aide dans l'organisation et la conception du projet Tea'Coffee, pour la découverte du Framework Angular, et pour sa bonne humeur constante.

Merci à Steve Duran et Nicolas Mallet qui, après le décès de Steve Duran, se sont investis dans cette première partie de ma formation ALC et ont beaucoup contribué à nous faire découvrir le langage SQL et JavaScript.

Enfin, je remercie mon entourage et ma famille qui m'ont soutenu durant ces 9 mois, malgré des moments difficiles.

INTRODUCTION

J'ai passé toute mon enfance et mon début de vie adulte à Montferrat, un petit village du Haut Var. Durant cette période, lorsque l'on me demandait ce que je voulais faire plus tard et que je répondais "devenir développeur de jeux vidéo", cela n'était pas bien compris par mon entourage et mes professeurs au collège. À la fin des années 90 et au début des années 2000, il y avait un manque d'informations sur ce sujet, les cursus correspondants étaient inexistant dans ma région, et mes mauvaises notes scolaires, dues à mon désintérêt pour l'école, ont finalement conduit à une orientation professionnelle par défaut.

Après le collège, j'ai suivi un Bac Professionnel en Mécanique automobile, ce qui m'a permis de faire mes premiers pas dans le monde du travail, et j'ai beaucoup apprécié cette période. Désireux de poursuivre des études supérieures en mécanique, j'ai été admis en BTS Conception et Maintenance des Machines Industrielles, pensant faire principalement de la conception et de la réparation mécanique. Cependant, j'ai été mal informé sur le contenu du cursus et je me suis retrouvé plongé dans la conception électrique et électrotechnique, ce qui a été difficile à rattraper.

Après cela, j'ai intégré une équipe de maintenance industrielle chez SOMECA pendant un an, puis j'ai exploré plusieurs métiers manuels (élagueur, menuisier, mécanicien naval, éboueur, etc.) car je n'étais pas satisfait dans la maintenance industrielle.

Par la suite, j'ai travaillé dans divers postes en mécanique automobile, notamment chez Peugeot Manosque pendant neuf mois, où j'ai beaucoup appris et acquis de l'expérience grâce à mon investissement. Malheureusement, des conflits internes m'ont poussé à quitter Peugeot pour rejoindre l'agence Volkswagen LB Automobile à Hyères.

J'ai passé cinq ans de ma vie professionnelle dans cette agence, et j'ai adoré travailler dans cette petite équipe où la collaboration et les relations entre le personnel étaient primordiales. J'en garde de très bons souvenirs et je suis toujours en contact avec cette entreprise.

Après sept ans dans la mécanique automobile, j'ai ressenti le besoin de changer de voie, n'ayant plus la passion ni la motivation pour continuer dans un domaine que je n'avais pas choisi. Après une réflexion personnelle et la redécouverte de la programmation à travers des projets Arduino et des vidéos partagées par FreeCodeCamp, j'ai décidé de quitter la mécanique pour me consacrer pleinement à ce qui me passionne réellement : le développement.

En septembre, j'ai quitté mon poste chez LB Automobile et j'ai immédiatement recherché une formation dans le développement web. J'ai commencé par la formation de découverte Code A La Carte proposée par la région et La Plateforme_, puis j'ai intégré en cours de session la formation DWWM (Développeur Web et Web Mobile) de La Plateforme_. J'ai travaillé dur pour rattraper mon retard et aujourd'hui, j'ai hâte de passer mon diplôme et de poursuivre mon parcours de développeur en Licence de Conception et Développement d'Applications en alternance.

COMPÉTENCES DU RÉFÉRENTIEL COUVERTES PAR LE PROJET

Tea'Coffee est un projet développé en Php et Javascript avec une architecture MVC (Modèle Vue Controller) et MariaDB comme SGBD (Système de Gestion de Base de Donnée).

Un conteneur Docker a été développé pour les services Php, MariaDB, PhpMyAdmin, Nginx et Nodejs.

ACTIVITÉ TYPE N° 1 : Développer la partie front-end d'une application web ou web mobile sécurisée

- Installer et configurer son environnement de travail

L'ensemble de l'équipe travaillant déjà avec l'environnement de développement intégré (IDE) Visual Studio Code, nous avons d'abord créé un repertoire GitHub puis vérifié nos version de PHP sur nos machine (8.2) afin de pouvoir installer et configurer Composer comme outil de gestion de dépendance en PHP. Une fois cela fait nous avons pu intégrer AltoRouter pour la gestion des routes et créer la base de l'architecture MVC de l'application.

- Maquetter des interfaces utilisateur web ou web mobile

Nous avons tout d'abord rédigé le cahier des charges et conçu tous les documents de conception de Tea'Coffee. Nous avons rédigé les user stories qui nous ont permis de créer les Maquettes du projet. Ceux-ci ont été créés grâce au l'espace de travail collaboratif Figma. Enfin, nous avons intégré tous ces documents dans le cahier des charges.

- Réaliser des interfaces utilisateur statiques web ou web mobile

Nous avons ensuite réalisé une intégration statique et responsive du site (celle-ci n'utilisait que HTML et CSS). Nous voulions avoir une base de travail avant d'intégrer les modèles de données afin de la dynamiser.

- Développer la partie dynamique des interfaces utilisateur web ou web mobile

Nous avons utilisé JavaScript pour afficher les produits favoris de l'utilisateur. Nous avons également utilisé un module appelé teaCoffee.module.js développé par l'un des membres de l'équipe afin de standardiser les requetes fetch et post, ou l'utilisation de fonction par définition d'attribut data (ex:<button data-js='handleScrollX,click' >).

ACTIVITÉ TYPE N° 2 : Développer la partie back-end d'une application web ou web mobile sécurisée

- Mettre en place une base de données relationnelle

En utilisant les user stories nous avons ensuite réalisé le MCD (Modèle Conceptuel des Données) par Workbench qui a automatiquement généré le MLD (Modèle Logique des Données). Nous avons utilisé la méthode MERISE pour les représentations séparées des données et des traitements.

Après avoir écrit le dictionnaire de données, nous avons créé notre base de données. Le MPD (Modèle Physique des Données) a été généré par PhpMyAdmin, l'interface de gestion de base de données.

- Développer les composants d'accès aux données SQL et NoSQL

La connexion à la base de données est faite par l'accès aux informations de connexion stockées dans le dossier appelé secret situé dans docker-data, et récupéré par une classe PHP appelé DockerSecrets. Le dossier secret est enregistré dans le fichier .gitignore.

Les données sont sous la forme d'objets et de collections d'objets, ce qui fait que nous sommes dans la programmation orientée objet (POO).

La manipulation et la consultation des données se font grâce aux classes Models (entités) et aux classes Controller (Contrôller du flux de données) de l'architecture MVC développé.

- Développer des composants métier coté serveur

L'application Tea'Coffee est composée de nombreuses fonctionnalités. Celles-ci ne sont pas toutes accessibles à un visiteur. Des rôles ont été créés afin de séparer les fonctionnalités pour un utilisateur non-connecté, un utilisateur connecté et un administrateur.

De fait un composant métier comme le RegisterController utilisé uniquement pour la gestion de l'inscription ou de la connexion d'un utilisateur a été développé en tenant compte des pratiques de sécurité comme le hachage de mots de passe.

Ainsi, un utilisateur après s'être enregistré ou connecté peut avoir accès à son profil et voir sa liste de favoris, l'historique des commandes ou modifier ses informations.

Un panneau administrateur a été aussi développé afin de gérer le back-office du site.

- Documenter le déploiement d'une application

Nous avons aussi commencé à détailler une documentation technique directement accessible sur le dépôt GitHub du projet dans la partie Wiki, permettant de créer une documentation par classe ou point important. L'application n'a pas encore été déployée sur un site d'hébergement pour le moment.

RÉSUMÉ DU PROJET

Le café et le thé sont bien plus que de simples boissons; ils sont ancrés dans les cultures et les rituels quotidiens à travers le monde. Depuis des siècles, ces deux breuvages ont été au centre de rencontres sociales, de cérémonies et même de débats philosophiques. Le café, souvent associé à l'énergie et à la productivité, a ses origines en Éthiopie et s'est répandu dans le monde entier, devenant un symbole de convivialité et de dynamisme. Le thé, avec ses racines profondément ancrées en Chine, incarne la sérénité et la réflexion, se déclinant en une multitude de variétés et de traditions allant des cérémonies du thé japonaises aux simples tasses de thé noir en Occident. Que l'on préfère l'amertume riche d'un espresso ou la douceur apaisante d'un thé vert, ces boissons continuent de jouer un rôle central dans nos vies, offrant à la fois réconfort et inspiration. TeaCoffee ambitionne de célébrer cette richesse culturelle et de la rendre accessible à tous, en un seul clic.

CAHIER DES CHARGES

Objectifs

Le projet Tea'Coffee vise à créer une boutique en ligne dédiée à l'univers des boissons chaudes, offrant une expérience immersive aux amateurs de thé et de café. Conçu avec une approche centrée sur l'utilisateur, le site Tea'Coffee sera une destination incontournable pour découvrir, partager et apprendre sur une variété de thés et cafés du monde entier.

Conceptualisation de l'application

Liste des fonctionnalités

Le site doit être responsive. La majorité des utilisateurs se sert de son smartphone, il faut donc que l'application puisse être consultée depuis un mobile.

Le logo de l'application est un lien de redirection vers la page d'accueil, et ce sur tout le site.

Le menu (menu burger au format mobile) est accessible sur tout le site et contient les liens vers les différentes pages du site, hors footer.

Le footer (pied de page) est à chaque fin de page et contient les liens vers les conditions de livraisons et de paiement, les conditions d'utilisation et de vente, et la page contact.

L'*utilisateur non-connecté* peut accéder au site. Une fois entré, il peut se rendre sur différentes pages (page Principale, page Produits, page Détails d'un produit, les pages de CGV et CGU, la page Contact, les pages de condition de Livraison et de Paiement) et accéder aux formulaires de connexion et d'inscription.

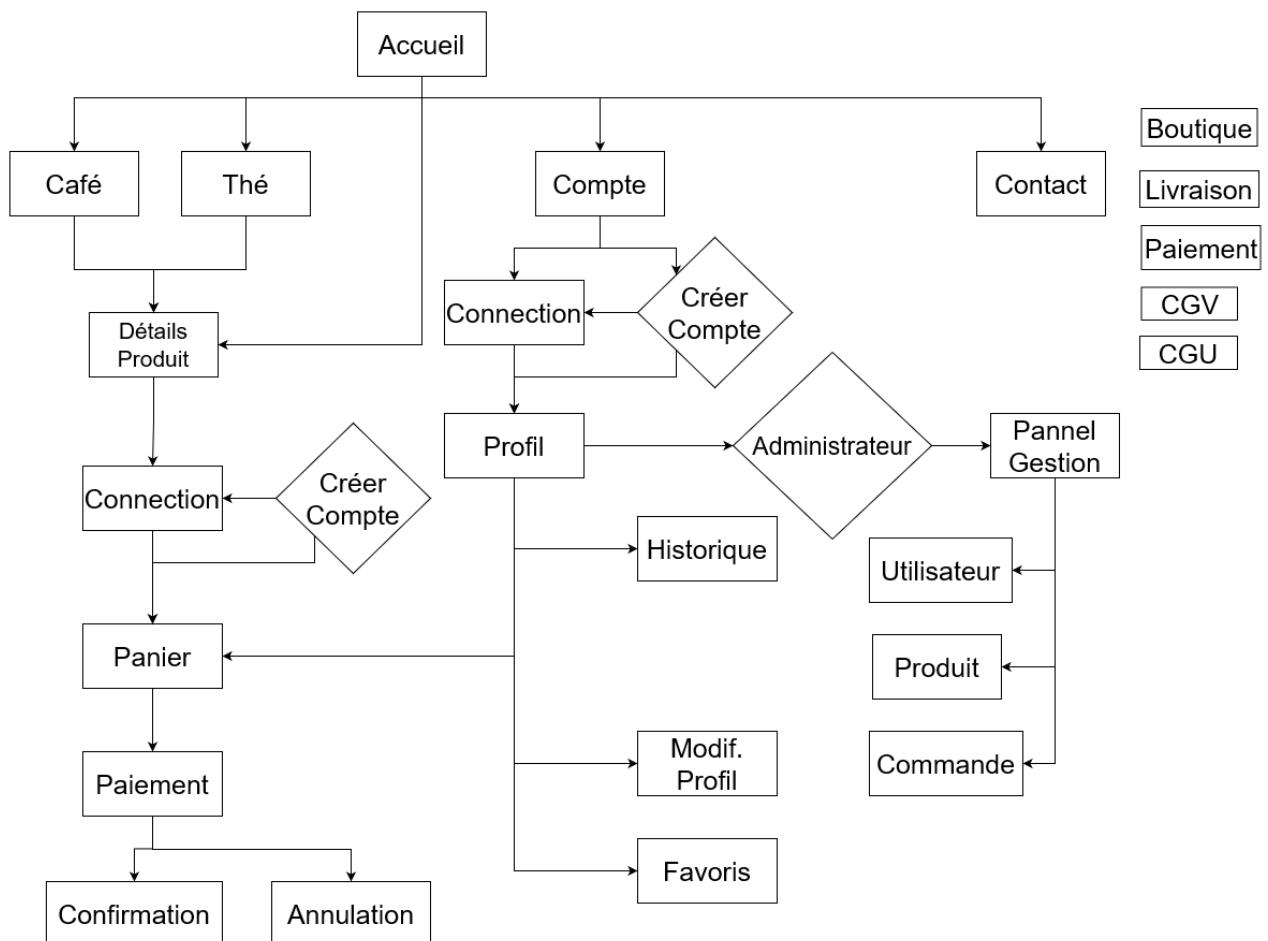
L'utilisateur connecté peut, en plus des droits de l'utilisateur non-connecté, modifier ses informations, mettre des produits dans sa liste de favoris ou dans son panier et y accéder à partir de la page Profil. Il peut executer un paiement par Stripe après validation de son panier.

L'administrateur a les mêmes fonctionnalités que l'utilisateur connecté. Il a en plus l'accès au panneau administrateur où il peut consulter la liste complète des utilisateurs et des produits et voir, ajouter, modifier et supprimer un produit et/ou un utilisateur.

Mise en place de l'application

Arborescence du site

A partir de la page d'accueil nous avons deux points d'entrée des pages Café et Thé pour visualiser l'ensemble des produits. La page de détails d'un produit peut-être accéder par l'accueil où les produits phares sont présenté ou directement en cliquant sur un produit dans les pages Café et Thé. L'accès au point d'entrée du compte utilisateur ce fait soit par l'accès direct dans la barre de navigation, soit l'utilisateur est redirigé automatiquement lors de la volonté de valider une commande. Une fois connecté l'accès au profil est à ses différentes pages est accordé. Les pages Contact ainsi que les différentes conditions sont accessible directement dans la barre de navigation ou le footer.



Maquette

Le nombre de page étant élevé, l'intégralité de la maquette est disponible à cette adresse :

<https://www.figma.com/design/3y6h3wzFPdaCcVcbmR1udM/Boutique-JS?node-id=526-324&t=MbTadbG63LxEH6va-0>

Comme évoqué dans les fonctionnalités, nous avons voulu un site responsive. Pour ce faire, nous avons abordé la conception de la maquette au format Desktop puis j'ai effectuer une maquette de la page d'accueil au format mobile. Le menu est sous la forme d'un menu burger au format mobile et l'affichage, contenu dans l'écran du mobile, nécessite uniquement de scroller vers le bas pour afficher le reste des pages de même que pour déplacer les produits proposés.



Figure 2: Maquette Mobile

Figure 1: Maquette Desktop

Charte graphique et logo / Accessibilité

Dans le but d'avoir la meilleure accessibilité possible, nous avons choisi d'utiliser un mode sombre et éclairé pour les personnes ayant une sensibilité au lumière blanche.

Concernant le choix des couleurs, nous voulions un site moderne unicouleur afin de faire ressortir les composants intégré dans la page.

Nous avons décidé de créer un logo inspiré de différents modèles disponibles sur internet. Après une discussion avec l'équipe, nous avons adopté celui proposé par un membre de l'équipe.



Figure 3: Logo Tea'Coffee

Utilisateurs et User Stories

Public visé

Tous types de personnes amateurs de thé ou de café recherchant des saveurs inédite ou des informations sur les produits consommés.

Rôles

Il n'y a que deux rôles pour l'application. Le rôle user (utilisateur) et le rôle admin (administrateur). Un utilisateur a automatiquement le rôle user lorsqu'il crée son compte. Seul un administrateur peut attribuer le rôle d'admin à un utilisateur.

User Stories

En tant que	Je souhaite	Afin de
Visiteur	Accéder à la page d'accueil	Accéder au site et voir les produits mis en avant
Visiteur	Accéder à la page Café	Voir la liste des Cafés
Visiteur	Accéder à la page Thé	Voir la liste des Thés
Visiteur	Accéder au détail d'un produit	Voir la description, les images, la notation et les commentaires d'un produits

Visiteur	Accéder aux pages de conditions	Voir les conditions d'utilisation, de vente, de livraison, de paiement.
Visiteur	Ajouter des produits au panier	Préparer une commande
Visiteur	Rechercher un produit	Accéder au détail du produit
Visiteur	Accéder aux pages de connection et inscription	M'enregistrer et/ou me connecter
Utilisateur	Accéder au panier	Valider une commande
Utilisateur	Accéder à la page profil	Modifier mes informations, voir l'historique des commandes, voir le panier, voir la liste des produits favoris
Administrateur	Accéder à la page administration	Visualiser les statistiques du site, accéder à la liste des commandes, accéder à la liste des produits, accéder à la liste des utilisateurs

Évolutions futures

La durée de développement du site et les difficultés rencontrées ne nous ont pas permis de mettre en place l'ajout de commentaire et de notation d'un produit, bien que la table soit créée dans la base de données. C'est ce que nous aimerais mettre en place lors de la V2. Entre autre, nous voulons également mettre en place l'utilisation d'un Framework afin de faciliter le développement et la gestion du site. Il faudrait également finir la partie administrateur et ajouter une demande de confirmation lors d'intéraction de mise à jour ou de suppressions des données. Enfin, à titre personnel, j'aimerais revoir la partie Front-End de l'accueil et des pages produits afin de les rendrent plus agréables visuellement.

SPÉCIFICATIONS TECHNIQUES

Versioning

La gestion de version permet de conserver l'historique du code source et de travailler plus facilement en équipe. Il était très important pour nous de garder les versions précédentes du code, notamment en cas de gros problème avec une fonctionnalité suite à des modifications dans le code. Pour cette gestion, nous avons décidé d'utiliser le logiciel de version décentralisé Git et le service web d'hébergement GitHub.

Afin d'optimiser notre manière de travailler et d'avoir la meilleure condition de travail possible, le repository de notre application comporte de multiples branches.

Nous avons notre branche de production main (principale en français). C'est la branche qui est visible par les utilisateurs lorsque le site est mis en production (mis en ligne).

À celle-ci s'ajoute la branche de pré-production dev. Elle recueille les fonctionnalités développées et nous permet de vérifier que celles-ci ne bloquent pas le site lorsque l'on ajoutait des nouvelles fonctionnalités.

Enfin, nous avons plusieurs branches. Chacune d'elles est dédiée à une fonctionnalité ou un problème rencontré. Une fois le travail terminé, le membre de l'équipe crée une demande de requête (Pull Request) et le code était revu par d'autre membre de l'équipe si cela s'avérait nécessaire. Si la demande était validé alors la branche était fusionnée (merge) avec la branch dev.

Ensuite un test était effectué sur les fonctionnalité importé dans la branch dev. Si après avoir validé les tests de la branch dev, une Pull Request de dev à main était proposé, la fusion entre les deux se faisait sans aucun problème.

Au début du projet, nous avions des difficultés avec Git et GitHub. En effet, nous n'avions pas encore mis en place le modèle de fonctionnement de création d'une branch de développement ainsi que de la création d'issue pour chaque fonctionnalité ou problème rencontré et cela nous a coûté du temps de développement.

L'utilisation de Github Desktop par certains membres de l'équipe et/ou le travail continue sur une même branche quelque soit la fonctionnalité par peur de causer des problèmes lors d'un merge a engendré des problèmes plus tard de génération de fichier d'ancienne version dans la branche principal. Il nous a fallu remettre tout à plat et reprendre notre manière d'appréhender le versioning. J'ai donc proposé une réunion afin, d'une manière bienveillante, d'expliquer et de mieux détailler les procédures de versionning pour chacun des membres de l'équipe.

Après un exercice effectué lors de la formation en dehors du cadre de ce projet, l'un des membres de l'équipe à pris la décision de créer la branch dev. J'ai soutenu cette décision qui me semble toujours logique. Concernant les commits, une partie de l'équipe préférait les écrire en français dû à leurs niveau d'anglais, nous avons donc décidé d'utiliser le français dans l'ensemble du projet pour faciliter la cohésion de l'équipe.

Extraits de commits que j'ai effectué :

```
#100 Modification des requêtes du panier suite erreur remove produit thé
Alexandre2383 committed 2 months ago
3801f29 ⌂ <>

#87 test stripe
Alexandre2383 committed 2 months ago
62dded ⌂ <>

Commits on May 23, 2024

#87 Ajout d'une vérification pour accéder à stripe/cancel
Alexandre2383 committed 2 months ago
f57d6dc ⌂ <>

#87 Ajout d'une vérification pour accéder à stripe/success
Alexandre2383 committed 2 months ago
bb6613d ⌂ <>

#100 Problème de suppression des produits thé du panier
Alexandre2383 committed 2 months ago
2c49173 ⌂ <>

Commits on May 21, 2024

Ajout d'un cookie permettant d'insérer les données du panier utilisateur non connecté lors de sa connection pour valider son panier
Alexandre2383 committed 2 months ago
10f7c9c ⌂ <>

#100 Modification de la méthode js post de teaCoffee.modules.js et implementation de l'ajout des produits directement dans slider via la requête effectué sur la route /addtobasket
Alexandre2383 committed 2 months ago
751932e ⌂ <>
```

Choix technologiques

Les choix techniques ont été faits lors de la première réunion du développement de l'application. L'exercice ayant imposé l'utilisation de PHP et JavaScript, et en programmation orienté object

Front-End

Le choix technologique concernant le Front-End (visible par l'utilisateur, côté client) a été fait d'un commun accord.

Nous avons utilisé le langage de structuration de page HTML 5 (HyperText Markup Language) et les Feuilles de Style en Cascade CSS 3 (Cascading Style Sheets).

Nous avons opté, pour la dynamisation de l'application, d'utiliser le langage de programmation de scripts JavaScript.

Puis, nous avons installé et configuré la librairie CSS Tailwind pour nous aider à mettre le site plus rapidement en forme.

Nous avons ainsi pu mettre en place les fonctionnalités suivantes :

- la page d'accueil avec le contrôle du Slide en JavaScript
- la page contact
- rendre le site responsive

Back-end

Le PHP étant obligatoire pour ce projet, nous avons opté pour une architecture MVC afin de mieux définir les responsabilités de chaque partie et de permettre une meilleure maintenabilité.

Étant donné que toute l'équipe avait travaillé depuis le début de la formation avec le langage de programmation PHP, le développement et l'intervention en cas de bug était facilité.

Nous avons installé et configuré Composer afin d'utiliser AltoRouter pour la gestion des Routes de l'application. Concernant notre base de données, nous avons d'abord porté notre choix sur le système de gestion de bases de données relationnelles MySQL, en utilisant l'outil graphique de gestion de bases de données PhpMyAdmin. La base de données et ses tables ont toutefois été créées grâce à Workbench.

Dans un second temps nous avons développé un conteneur Docker (annexes) composé des services MariaDB, PHP, PhpMyAdmin et Nginx, ainsi qu'un service Nodejs pensé pour la création d'une API que nous n'avons pas pu finir.

Nous avons ainsi pu mettre en place les fonctionnalités suivantes:

- la création d'un compte utilisateur
- la connexion d'un utilisateur
- l'accès à différentes parties du site en fonction du rôle attribué
- l'ajout d'un produit et d'un utilisateur
- la modification d'un produit et d'un utilisateur
- la suppression d'un produit et d'un utilisateur

- l'ajout et la suppression d'un produit favori
- l'affichage de la liste des produits en fonction d'une catégorie donnée

Accessibilité

Navigateurs compatibles

Lors du développement de l'application j'ai veillais à ce que la compatibilité avec de nombreux navigateurs soit assurée. Ainsi l'application est compatible avec les navigateur FireFox, Chrome, Edge, DuckDuckGo, Opera et Brave. Ayant eu accès à une tablette Apple j'ai pu m'assurer de la compatibilité avec le navigateur Safari.

En revanche, nous n'avons aucunement prévu de rendre le site compatible pour Internet Explorer, étant donné qu'il a été arrêté par Microsoft et est remplacé depuis par Microsoft Edge.

Types d'appareils

Les types d'appareils compatibles avec notre application sont les smartphones, les tablettes et les ordinateurs. En effet, nous avons développé une application responsive et nous nous sommes assurés que l'affichage soit adapté en fonction de certains types de résolution. J'ai également effectué des tests utilisateurs en créant un serveur temporaire Ngrok afin de me permettre de présenter l'application à plusieurs personnes de mon entourage afin d'avoir des opinions sur la maniabilité et le visuel en format Mobile, celle-ci m'ayant permis de mieux développé la partie Front-End Mobile de l'application.

Architecture du projet

Nous avons utilisé l'architecture MVC (Modèle-Vue-Contrôleur). Comme son nom l'indique, ce type d'architecture comporte trois types de modules :

- le modèle :

Il contient les données ainsi que la logique du flux se rapportant à elles.

- la vue :

C'est la partie visible de l'interface graphique. Elle gère la disposition et l'affichage des pages grâce aux balises HTML qu'elle contient.

- le contrôleur :

Il contient la logique des actions effectuées par l'utilisateur. Il achemine les commandes des parties modèle et vue.

Lorsque l'utilisateur va vouloir afficher une donnée, il va faire une requête HTTP qui va demander une route. Le point d'entrée unique index.php, qui contient toutes les routes du

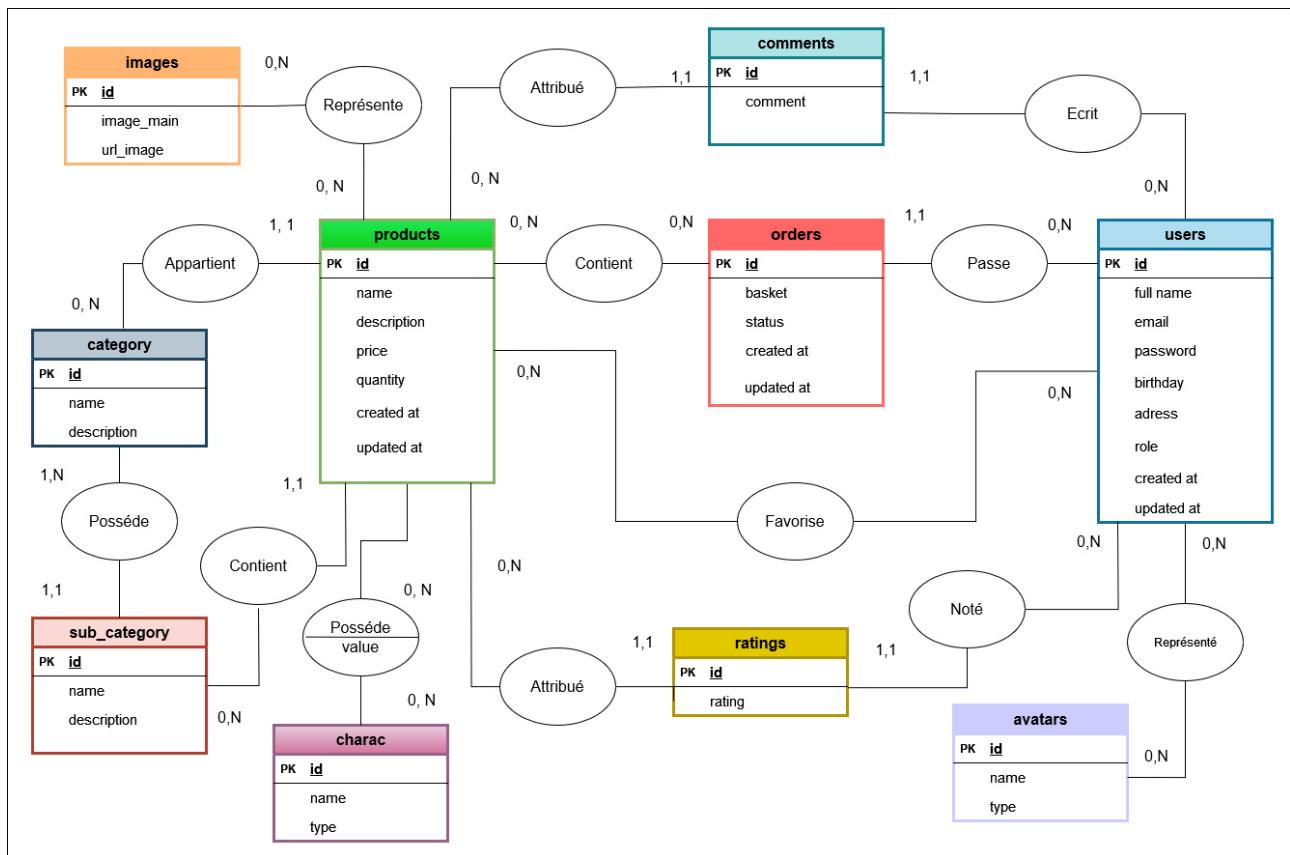
site, va appeler le bon contrôleur et la bonne méthode. Le contrôleur va à son tour appeler une méthode d'un modèle, afin de récupérer dans la base de données, les données que l'utilisateur veut afficher. Le modèle retourne au contrôleur ses résultats sous la forme d'un objet PHP. Ce dernier passe les données à la vue qui affiche la page structurée incluant les données demandées au client.

Nous avons modifié la logique de rendu du Vue par la création d'une classe Render gérant le renvoi d'une template demandé en automatisant l'import du header, de la barre de recherche, du footer ainsi que du SEO.

La Base de Données

Par manque de temps, nous n'avons pas encore utilisé la table des bières, bien qu'elle existe déjà dans la base de données.

MCD



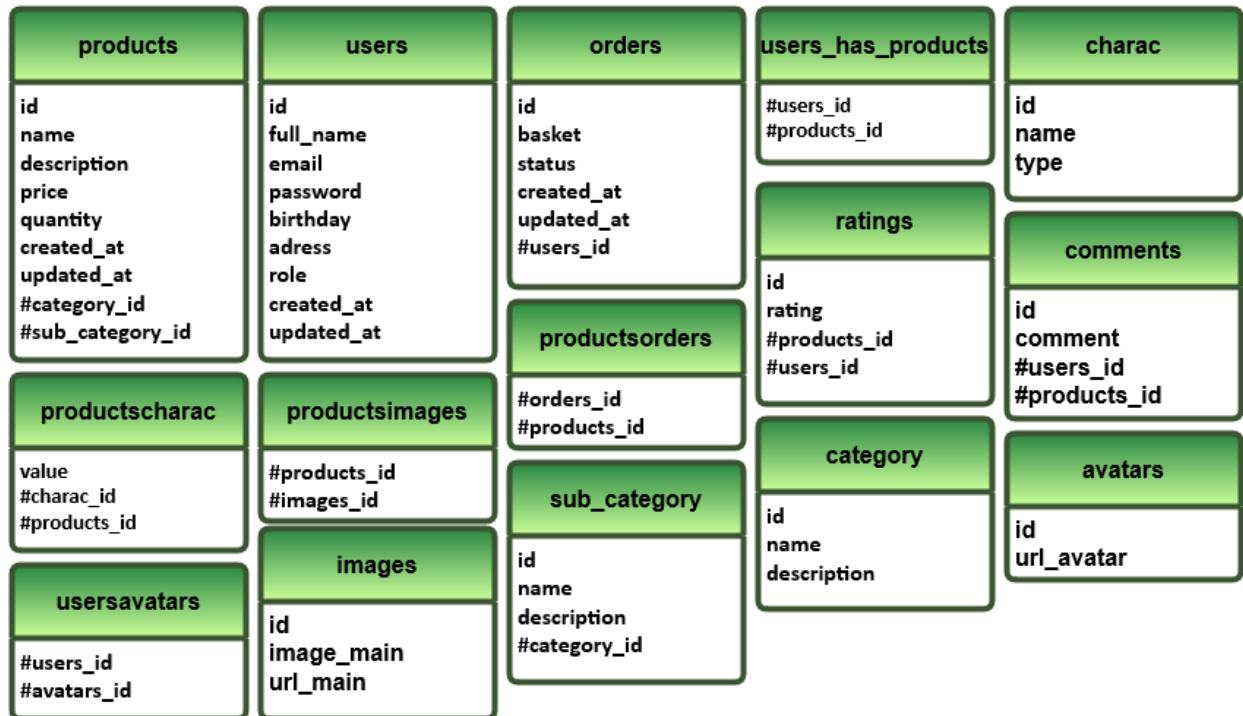
Le Modèle Conceptuel de Données a été pensé et créé lors de la première réunion concernant la base de données. Nous avons réfléchi ensemble au nombre d'entités qu'il faudrait avoir pour que notre site soit fonctionnel.

Le précédent MCD ayant étant généré automatique par Workbench, je l'ai entièrement refait en utilisant la méthode Merise et le site draw.io pour parfaire mes connaissances.

MLD

Le Modèle Logique de Données a automatiquement été généré par Workbench. Le MLD indique les clés étrangères détenues dans les différentes tables et les tables de jointures nécessaire.

Par exemple, les tables Products et Orders ayant une cardinalité de 0,N dans leurs relations (Many-to-Many) une table de jointures PRODUCTSORDERS sera créée contenant l'identifiant du produit en relation avec l'identifiant de la commande. L'attribut Favorise deviendra une table de relation appelé USERS_HAS_PRODUCTS permettant d'ajouté ou enlevé les produits favoris sélectionner par l'utilisateur.



Dictionnaire de données

Pour aider à la création de la base de données, nous avons rédigé ensemble le dictionnaire de données. Celui-ci est une collection de métadonnées ou de données de référence nécessaire à la conception d'une base de données relationnelle. Nous y avons indiqué le type de chaque champ d'une table de la base de données.

orders

Colonne	Type	Null	Valeur par défaut	Commentaires
id (Primaire)	int(11)	Non		
basket	tinyint(1)	Oui	NULL	
status	enum('en attente', 'expédier', 'livrer', 'échec')	Oui	NULL	
created_at	datetime	Oui	current_timestamp()	
updated_at	datetime	Oui	current_timestamp()	
users_id	int(11)	Non		

Index

Nom de l'index	Type	Unique	Comprimé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	id	42	A	Non	
fk_orders_users1_idx	BTREE	Non	Non	users_id	42	A	Non	

products

Colonne	Type	Null	Valeur par défaut	Commentaires
id (Primaire)	int(10)	Non		
name	varchar(100)	Oui	NULL	
description	text	Oui	NULL	
price	float	Oui	NULL	
quantity	int(11)	Oui	NULL	
created_at	datetime	Oui	current_timestamp()	
updated_at	datetime	Oui	current_timestamp()	
category_id	int(11)	Non		
sub_category_id	int(11)	Non		

Index

Nom de l'index	Type	Unique	Comprimé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	id	249	A	Non	
id_product_UNIQUE	BTREE	Oui	Non	id	249	A	Non	
name_UNIQUE	BTREE	Oui	Non	name	249	A	Oui	
fk_products_category1_idx	BTREE	Non	Non	category_id	4	A	Non	
fk_products_sub_category1_idx	BTREE	Non	Non	sub_category_id	12	A	Non	

productsimages

Colonne	Type	Null	Valeur par défaut	Commentaires
products_id	int(10)	Non		
images_id	int(11)	Non		

Index

Nom de l'index	Type	Unique	Comprimé	Colonne	Cardinalité	Interclassement	Null	Commentaire
fk_ProductsImages_images1_idx	BTREE	Non	Non	images_id	83	A	Non	
fk_ProductsImages_products1_idx	BTREE	Non	Non	products_id	83	A	Non	

productscharac

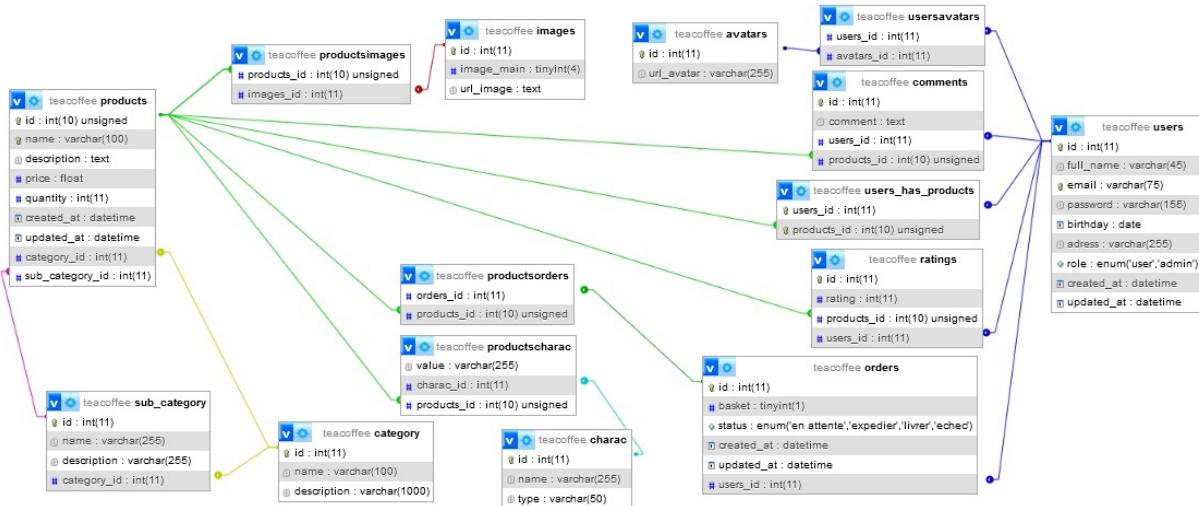
Colonne	Type	Null	Valeur par défaut	Commentaires
value	tinyint(1)	Oui	NULL	

Index

Nom de l'index	Type	Unique	Comprimé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	value	1	A	Non	

MPD

Une fois la base de données créée avec workbench importé dans notre SGBD MariaDB, nous avons pu visualisé les relations en utilisant l'outil conception de PhpMyAdmin.



Pour insérer les données dans la base, nous avons premièrement importé le fichier de création de la base de donnée dans PhpMyAdmin. De mon point de vue personnel et afin de bien comprendre les interactions, j'ai réécrit à la main la création des tables et de leurs contraires.

```
-- Création des tables de données

CREATE TABLE `avatars` (
  `id` int NOT NULL,
  `url_avatar` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `category` (
  `id` int NOT NULL,
  `name` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL,
  `description` varchar(1000) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `charac` (
  `id` int NOT NULL,
  `name` varchar(255) NOT NULL,
  `type` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

//...

-- Ajout des clés primaires et étrangères

ALTER TABLE `avatars`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `category`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `charac`
  ADD PRIMARY KEY (`id`);

//...

-- Ajout des contraintes par table

ALTER TABLE `comments`
  ADD CONSTRAINT `fk_comments_products1` FOREIGN KEY (`products_id`) REFERENCES `products` (`id`),
  ADD CONSTRAINT `fk_comments_users1` FOREIGN KEY (`users_id`) REFERENCES `users` (`id`);

ALTER TABLE `orders`
  ADD CONSTRAINT `fk_orders_users1` FOREIGN KEY (`users_id`) REFERENCES `users` (`id`);

ALTER TABLE `products`
  ADD CONSTRAINT `fk_products_category1` FOREIGN KEY (`category_id`) REFERENCES `category` (`id`),
  ADD CONSTRAINT `fk_products_sub_category1` FOREIGN KEY (`sub_category_id`) REFERENCES `sub_category` (`id`);
```

Enfin, l'un des membres de l'équipe a automatisé un jeu de données permettant d'importer un grand nombre de données dans nos tables.

Extrait de la requête d'ajout d'images :

```
INSERT INTO `images` (`id`, `image_main`, `url_image`) VALUES  
(1, 1, 'café-affogato.png'),  
(2, 1, 'café-américano.png'),  
(3, 1, 'café-bélizien.png'),  
(4, 1, 'café-bio.png'),  
(5, 1, 'café-bolivien.png'),  
(6, 1, 'café-brésilien.png'),  
(7, 1, 'café-cappuccino.png'),  
(8, 1, 'café-colombien.png'),  
(9, 1, 'café-costaricien.png'),  
(10, 1, 'café-cubain.png'),  
(11, 1, 'café-dominicain.png'),  
(12, 1, 'café-Équatorien.png'),  
(13, 1, 'café-Éthiopien.png'),  
(14, 1, 'café-expresso.png'),  
(15, 1, 'café-flat-white.png'),  
(16, 1, 'café-grec.png'),  
(17, 1, 'café-guatémaltèque.png'),  
(18, 1, 'café-guyanien.png'),  
(19, 1, 'café-haïtien.png'),  
(20, 1, 'café-hawaïen.png'),
```

INSERT INTO sert à insérer, comme son nom l'indique, les données dans l'ordre des champs de la table images. VALUES indique les valeurs à ajouter pour chaque champs spécifié. Afin d'établir la connexion à la base de données le dossier appelé secret situé dans docker-data permet l'accès aux informations de connexion stockées. L'utilisation des secrets se fait par une classe PHP appelé DockerSecrets. Pour des raisons de sécurité, le dossier secret est enregistré dans le fichier .gitignore.

Routes

Une fois les fonctionnalités définies et les user stories rédigées, nous avons spécifié les routes pour chacune d'elles.

```
$router = new AltoRouter();  
  
$router->map('GET', '/', 'HomeController#RenderHome', 'accueil');  
$router->map('GET|POST', '/inscription', 'RegisterController#Register', 'inscriptionRegister');  
$router->map('GET|POST', '/connexion', 'RegisterController#ConnectJS', 'connexionForm');  
$router->map('GET', '/deconnexion', 'RegisterController#Disconnect', 'deconnexion');  
$router->map('GET', '/favoris/[i:product]', 'Favoris#VerifyFavorite', 'testIsConnected');  
$router->map('GET', '/addFavoris/[i:product]', 'Favoris#ToggleFavorite', 'addFavorite');  
$router->map('GET', '/contact', 'contact', 'contact');  
$router->map('POST', '/contact', 'RegisterController#ContactMail', 'contactForm');  
$router->map('GET', '/information/livraison', 'information/livraison', 'livraison');  
$router->map('GET', '/information/paiement', 'information/paiement', 'paiement');  
$router->map('GET', '/information/boutique', 'information/boutique', 'boutique');
```

```

$router->map('GET', '/search', 'ApiController#GetProductsAll', 'search');
$router->map('POST', '/addtobasket', 'PanierController#AddToBasket', 'addtobasket');
$router->map('GET', '/produit/addtobasket/[a:product_id]', 'PanierController#AddToBasket',
'addtobasketProduit');
$router->map('POST', '/removefromcart', 'PanierController#RemoveFromCart', 'removefromcart');
$router->map('GET', '/user', 'ProfilController#Profil', 'user-profile');
$router->map('GET', '/modification', 'modification', 'modification');
$router->map('POST', '/modification', 'ModificationController#Modification',
$router->map('GET', '/panier', 'PanierController#Panier', 'panier');
$router->map('GET|POST', '/api-html/form/[a:tableName]', 'HtmlToJsonController#FormAdmin', 'api-
html-tojson-form');
$router->map('GET|POST', '/api-html/template/[a:pageTemplate]/[i:idGet]',
'HtmlToJsonController#Template', 'api-html-tojson-template');
$router->map('GET', '/panel-admin', 'AdminPanel#IndexPanel', 'admin-panel-index');
$router->map('GET|POST', '/panel-admin/[a:tableName]', 'AdminPanel#Index', 'admin-panel-select');
$router->map('GET|POST', '/panel-admin/[a:tableName]/[i:id]', 'AdminPanel#Index', 'admin-panel-
select-page');
}
$router->map('GET', '/js-testAll/[a:idCat]', 'Filters#produitElement', 'queryAll');
$router->map('GET', '/js-testSub/[a:idCat]/[a:idSubCat]', 'Filters#produitElement',
'querySubCat');
$router->map('GET', '/js-testFilter/[a:idCat]/[a:filter]', 'Filters#produitElement',
'queryFilter');
$router->map('GET', '/js-testBoth/[a:idCat]/[a:idSubCat]/[a:filter]', 'Filters#produitElement',
'queryBoth');
$router->map('GET', '/favoris/[i:product]', 'Favoris#VerifyFavorite', 'testIsConnected');
$router->map('GET', '/addFavoris/[i:product]', 'Favoris#ToggleFavorite', 'addFavorite');
$router->map('get', '/wishlist', 'Favoris#ShowFavorites', 'userWishlist');
$router->map('GET', '/orderverif/[i:idProduct]', 'Ratings#ProductOrdered', 'checkIfOrdered');
$router->map('GET', '/productrating/[i:idProduct]', 'Ratings#ProductRating', 'ratingProduct');
$router->map('GET', '/sample-to-favorites', 'Filters#produitElement', 'sample-add-to-favorites');
$router->map('POST', '/sample-connect-js', 'RegisterController#ConnectJS', 'sample-connect-js');
$router->map('GET', '/form-test-inscription', 'FormControllerTest#RegistrationForm', 'form-
registration');
$router->map('POST', '/form-test-inscription', 'FormControllerTest#RegistrationForm', 'form-
registration-validate');
$router->map('GET', '/form-test-connect', 'FormControllerTest#ConnectForm', 'form-connect');
$router->map('POST', '/form-test-connect', 'FormControllerTest#ConnectUser', 'form-connect-
validate');
$router->map('GET', '/condition/cgv', 'condition/cgv', 'cgv');
$router->map('GET', '/condition/cgu', 'condition/cgu', 'cgu');
$router->map('GET', '/panier-modal', 'PanierController#PanierDynamique', 'panier-modal');
$router->map('GET', '/stripe/pay', 'StripeController#Pay', 'pay');
$router->map('GET', '/stripe/cancel', 'StripeController#Cancel', 'cancel');
$router->map('GET', '/stripe/success', 'StripeController#Success', 'success');

```

Nous avons créé les routes de la partie API mais par manque de temps nous n'avons pas pu encore l'utiliser.

```
$router->map('GET', '/api/products_all', 'ApiController#GetProductsAll', 'products-all');
$router->map('GET', '/api/category_all', 'ApiController#GetCategoryAll', 'categorys-all');
$router->map('GET', '/api/orders_all', 'ApiController#GetOrdersAll', 'orders-all');
$router->map('GET', '/api/users_all', 'ApiController#GetUsersAll', 'users-all');
$router->map('GET', '/api/products/[a:category]', 'ApiController#GetProductsByCategory',
'products-category');
$router->map('GET', '/api/users/[i:id]', 'ApiController# GetUserById', 'user');
$router->map('GET', '/api/products/[i:id]', 'ApiController#GetProductById', 'product');
$router->map('GET', '/api/category/[i:id]', 'ApiController#GetCategoryById', 'category');
$router->map('GET', '/api/orders/[i:id]', 'ApiController#GetOrderById', 'order');
$router->map('POST', '/api/Products', 'ApiController#addProducts', 'addProducts');
$router->map('POST', '/api/Category', 'ApiController#addCategory', 'addCategory');
$router->map('POST', '/api/Orders', 'ApiController#addOrders', 'addOrders');
$router->map('POST', '/api/Users', 'ApiController#addUsers', 'addUsers');
$router->map('POST', '/api/Products/[i:id]', 'ApiController#updateProducts', 'updateProducts');

$router->map('POST', '/api/Category/[i:id]', 'ApiController#updateCategory', 'updateCategory');
$router->map('POST', '/api/Orders/[i:id]', 'ApiController#updateOrders', 'updateOrders');
$router->map('POST', '/api/Users/[i:id]', 'ApiController#updateUsers', 'updateUsers');
$router->map('POST', '/api/feedback-validation/[i:id]', 'ApiController#addFeedback',
'addFeedback');
$router->map('DELETE', '/api/Products/[i:id]', 'ApiController#deleteProducts', 'deleteProducts');
$router->map('DELETE', '/api/Category/[i:id]', 'ApiController#deleteCategory', 'deleteCategory');
$router->map('DELETE', '/api/Orders/[i:id]', 'ApiController#deleteOrders', 'deleteOrders');
$router->map('DELETE', '/api/Users/[i:id]', 'ApiController#deleteUsers', 'deleteUsers');
```

Déploiement

Nous voulions déployer la version v.1.0 de l'application sur le serveur Plesk fourni par l'école, mais le manque de temps et la configuration du système de gestion de la base de donnée du serveur étant trop ancienne (MariaDB 5) par rapport à notre version (10.6), au moment où j'écris ces lignes l'application n'a pas encore été déployée car nous n'avons pas pu mettre en place les solutions recherchées.

Sécurité de l'application

La sécurité dans le développement web revêt aujourd'hui une importance cruciale. À une époque où les données personnelles et sensibles circulent en abondance sur internet, chaque faille de sécurité peut offrir une opportunité aux individus mal intentionnés pour accéder à des informations confidentielles des utilisateurs. La Commission Nationale de l'Informatique et des Libertés (CNIL) insiste sur le fait que « tout site web doit garantir son identité et la confidentialité des informations transmises ». Cela signifie que les développeurs ont la responsabilité de mettre en place des mesures robustes pour prévenir tout accès non autorisé aux données.

Dans le cadre de nos propres projets de développement, nous avons mis en œuvre diverses mesures de sécurité pour protéger notre application. Par exemple, nous avons utilisé la bibliothèque Sodium de PHP, reconnue pour ses capacités avancées en cryptographie, ou appliquer des fonctions telle que htmlSpecialChars dans l'application.

Formulaires

Partant d'une simple qui est de ne jamais faire confiance à l'utilisateur et ayant conscience qu'une personne mal intentionné puisse avoir la possibilité de décripter un mot de passe, nous avons fait le choix d'utilisé Sodium inclut dans PHP à partir de la version 8.2 qui est basé sur l'algorithme d'encryptage Argon2, l'un des plus recommandé. A partir de cette décision nous avons développé le composant RegisterController afin d'appliquer le maximum de sécurité dans l'enregistrement et la connection d'un utilisateur.

```
public function Register(...$arguments)
{
    /**
     * condition de validation des données et de hachage du mot de passe
     */
    if (!empty($_POST)) {
        $modelUser->setPassword($arguments['password'] ?? '');
        $errors = ReflectionValidator::validate($modelUser);

        if (!$errors && !isset($arguments['email'])) {

            $crudManager = new CrudManager('users', UsersRegistration::class);

            if ($crudManager->getByEmail($arguments['email']) !== false) {
                throw new ClientExceptions(ClientExceptionEnum::AccountIsRegistered);
            } else {
                $modelUser->setPassword($modelUser->hash($arguments['password']));
                $crudManager->create($modelUser, ['full_name', 'email', 'password', 'role']);
                setcookie('registered_user', $modelUser->getFull_name());
                header('location:/connexion'); // REDIRECT
            }
        }
    }
}
```

Nous avons développé une classe PasswordHashManager spécialisé dans le Hachage du mot de passe et de sa vérification.

```
class PasswordHashManager
{

    public function hash(string $password): string
    {
        if (empty($password)):
            return $password;
        endif;

        return \sodium_crypto_pwhash_str($password, SODIUM_CRYPTO_PWHASH_OPSLIMIT_INTERACTIVE,
SODIUM_CRYPTO_PWHASH_MEMLIMIT_INTERACTIVE);
    }

    public function verify(string $hash, string $password): bool
    {
        if ('' === $password || \strlen($password) < 8):
            return false;
        endif;

        if (\sodium_crypto_pwhash_str_verify($hash, $password)):
            return true;
        endif;

        return false;
    }
}
```

Et pour éviter des doublons d'utilisateur dans base de données, nous avons mis en place l'unicité de l'email avec l'utilisation

```
public function ConnectJS(...$arguments)
{
    /**
     * $verifPassword = new PasswordHashManager();
     * // Vérification du mot de passe.
     * if ($verifPassword->verify($user->getPassword(), $arguments['password']))
     *     // Incorporation des paramètres de l'utilisateur dans la session.
     *     $arguments['render']->addSession([
     *         'email' => $user->getEmail(),
     *         'isConnected' => true,
     *         'full_name' => $user->getFull_name(),
     *         'role' => $user->getRole(),
     *     ]);
     */
    /**
     * $this->responseJson(200, ['isConnected' => true]);
     */
    else {
        $this->responseJson(200, ['errors' => ['email' => "Il n'y a aucune
correspondance entre votre email et votre mot de passe."]]);
    }
}
```

De même pour les données transmises par requête POST on applique une sécurité pour effacer les caractères vides en début et fin de chaîne avec la fonction trim() et on convertis la chaîne de caractères en code html en utilisant htmlspecialchars() afin de s'assurer qu'aucun code malveillant soit transmis par un utilisateur.e la donnée fournie par l'utilisateur soit de type Url.

```
if (isset($_POST)) {
    foreach ($_POST as $key => $value) {
        $match['params'][$key] = htmlspecialchars(trim($value));
    }
}
```

GESTION DE PROJET

Présentation de l'équipe et organisation de travail

Notre équipe était composée à la base de cinq personnes puis nous sommes passé à quatre personnes assez rapidement. Il n'y a pas eu d'attribution de rôle, bien que cela aurait pu nous aider à avancer plus vite, l'organisation de travail c'est faite de manière naturel car chaque membre s'est investi dans le projet et y a contribué. Nous avons commencé par une réunion pour le choix thématique de la boutique et, malgré la condition de télétravail qui a dû être imposé une semaine sur deux, nous avons continué à travailler ensemble de cette façon pour chaque point important comme la maquette, le MCD, ou la création de fonctionnalités importante.

L'organisation se faisant naturellement en utilisant les tableaux de Trello et les Git Issue ce qui a permis à chacun de visualiser où en été le projet.

Nous avons appuyer sur le fait que si un membre de l'équipe n'arrivait pas à comprendre ce qui était demandé ou été bloqué sur une partie du développement alors il en faisait par à l'équipe afin d'être appuyé dans la recherche d'une solution. Cette façon de procédé nous a permis d'avancer tous ensemble sans laisser un membre bloqué dans son coin, ce qui malgré la timidité a renforcé la cohésion de l'équipe. Tous les membres ont travaillés sur la partie Front-End et Back-end.

J'ai notamment mis en place les premières Git Issue après une réunion avec notre formateur afin de faire le point sur le projet et assigné les membres en fonction des issues, ce qui à été apprécié par l'équipe.

<input checked="" type="checkbox"/> Ajout de favoris	enhancement
#23 by Alexandre2383 was closed on May 10	
<input checked="" type="checkbox"/> Notation des produits	enhancement
#22 by Alexandre2383 was closed 4 days ago	
<input checked="" type="checkbox"/> Ajout de commentaire pour les produits	enhancement
#21 by Alexandre2383 was closed 4 days ago ↗ 2	
<input checked="" type="checkbox"/> Page blanche sans controller défini	bug
#20 by Alexandre2383 was closed on May 3	
<input checked="" type="checkbox"/> Rassurer le client	enhancement
#19 by Alexandre2383 was closed on May 14	
<input checked="" type="checkbox"/> Utilisation de JWT API (private routes)	enhancement
#18 by Alexandre2383 was closed on May 17	
<input checked="" type="checkbox"/> Suppression/mise à jour dans la page panier	enhancement
#17 by Alexandre2383 was closed on May 7	
<input checked="" type="checkbox"/> Ajout des produits de la carte produit au panier	enhancement
#16 by Alexandre2383 was closed on May 7	1
<input checked="" type="checkbox"/> Création d'un système de paiement (stripe checkout)	enhancement
#15 by Alexandre2383 was closed on May 8 ↗ 1	

Figure 4: Premières Git Issue

L'organisation ayant dû être adapté pour le télétravail, nous avons créé un espace de travail sur Gmail afin de rester en contact et de continuer à partager notre avancement ou recherches, et chaque semaine nous avons maintenu des réunions par Google Meet.

Programme de l'organisation

Comme dit précédemment, la durée de développement du projet étant très courte, nous avons dû rapidement apprendre à nous organiser. Lors de la premières semaines nous avons mis en place le Trello afin de déployer l'organisation de travail et enregistré toute les idées que nous avions en tête.

Ensuite nous sommes passé à la partie conception de la maquette, création de l'architecture et configuration du projet, et enfin la conception de la base de donnée. Nous avons beaucoup travaillé lors de cette premières semaine car celle qui suivait été en télétravail et certains des membres pensaient qu'ils serait trop difficile de continuer à mettre en place la base du projet.

Cette décision a porté ses fruits car dès la deuxième semaine nous avons pu commencer à développer les fonctionnalités principales de l'application dont la partie d'enregistrement et de connection d'un utilisateur, l'affichage de produits et la logique de rendu du template.

De cette façon nous avons maintenu une organisation de travail et continué à s'entraider sur le développement du projet.

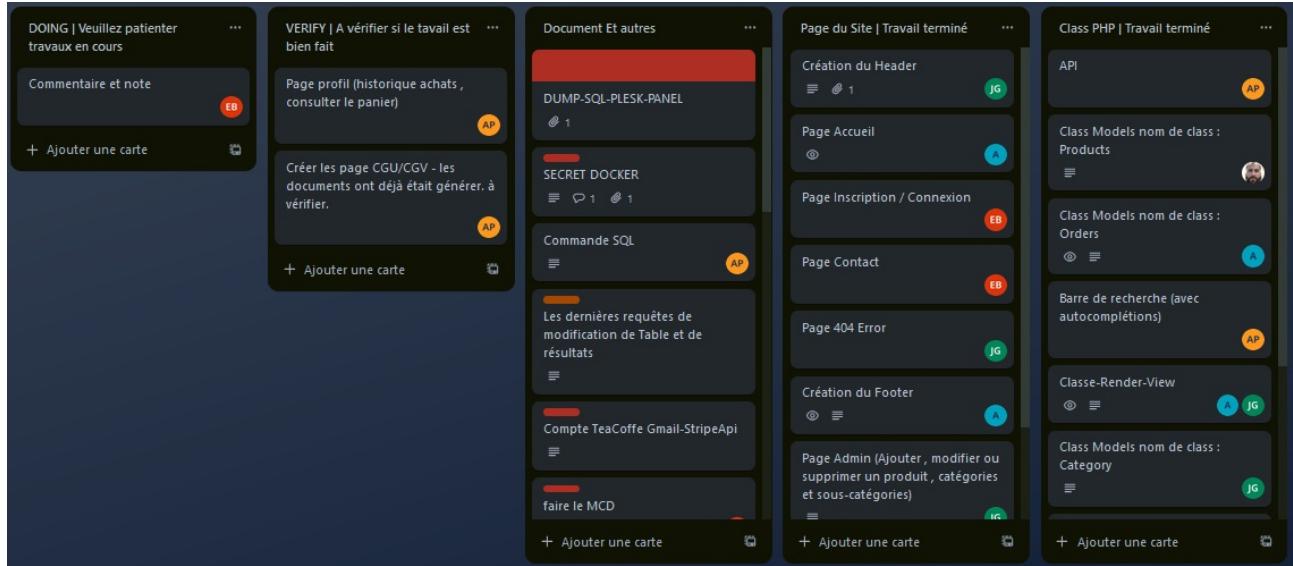


Figure 5: Organisation du projet avec Trello

RÉALISATIONS PERSONNELLES

Pour mes réalisations personnelles, j'ai contribué à la réalisation des documents de conception lors de la première semaine. En outre, j'ai fait les pages Accueil, Thé et Café de la maquette en prenant en compte l'avis des membres de l'équipe et j'ai participé à la création de la base de donnée. J'ai ensuite dans la phase de développement créé la classe Render, une partie de Front-End et l'implémentation d'un système de paiement avec Stripe Checkout (annexes).

Rôle de la classe Render, rendu du contenu à l'utilisateur

La classe Render est responsable de la logique de rendu des templates et de l'affichage final des produits à l'utilisateur. Une fois appelé dans un contrôleur elle utilise les données, les intègre dans les templates php, et génère le contenu final qui est envoyé au navigateur. Cette classe est inspirée de la méthode render du FrameWork Symfony. Les données à rendre renvoyé par le controller (par ex : HomeController) sont contenues dans un tableau de paramètres en utilisant la fonction call_user_function_array() introduite par PHP 8.

```
if (is_callable([$controller, $method])) :  
    // Cela permet de charger dynamique des function ou des méthodes définit dans les class.  
    echo call_user_func_array([$controller, $method], $match['params']);
```

Figure 6: Utilisation de call_user_function_array dans la logique d'affichage du projet

On utilise call_user_func_array pour instancier la classe chargée précédemment dans la variable \$controller et on lui passe en deuxième paramètre un tableau d'argument que nous récupérons dans la méthode que l'on a déclarée dans \$method.

Exemple de la documentation :

```
$func = function($arg1, $arg2) {  
    return $arg1 * $arg2;  
};  
  
var_dump(call_user_func_array($func, array(2, 4)));  
// affiche $arg1 = 2 et $arg2 = 4
```

Figure 7: Exemple de la documentation

Ici la fonction \$func est chargée et un tableau avec deux variables est passé en paramètre. Cela permet de charger dynamique des fonctions ou des méthodes définies dans les classes. De cette manière, nous pouvons appeler de façon récursive des méthodes d'un contrôleur en passant un tableau d'arguments .

Si on prends l'exemple de la méthode RenderHome de la classe HomeController responsable de traiter les données nécessaires pour afficher la page d'accueil de l'application, celle-ci dans un premier temps crée des instances du CrudManager pour les produits et les utilisateurs afin d'utiliser ses méthodes de récupération des données des tables de la base de données. Ensuite elle hydrate les données dans une instance du composant Slider pour afficher les produits sur la page d'accueil.

Afin de renvoyer l'ensemble du traitement à l'utilisateur, nous appelons la méthode render pour utiliser la logique de Rendu en lui passant l'ensemble des données dans le tableau d'arguments 'render', contenant les données à renvoyer par le controller dans le tableau clé/valeur 'accueil' et 'arguments'.

```
// $content contient le passage des données du tableau
// d'arguments 'render' à la méthode render, prenant en paramètre le
// template où les arguments sont passés
$content = $arguments['render']->render('accueil', $arguments);

// rend le template chargé avec les données au navigateur
return $content;
}
```

Figure 8: Utilisation de la méthode render

Une fois la méthode render de la classe Render appelée celle-ci va commencer par démarrer la mise en mémoire tampon avec ob_start(), capturant ainsi tout le contenu généré. Les données de configuration SEO spécifiques au template sont ensuite ajoutées aux paramètres via la méthode addParams. Les arguments passés à la méthode render sont fusionnés avec les paramètres internes et extraits pour être utilisés directement dans le template. Le header, la barre de recherche et le footer sont inclus, suivis du template spécifique (accueil dans notre cas) après vérification de son existence. Le contenu généré est capturé et la mémoire tampon est effacée avec ob_get_clean(). Finalement, le contenu capturé est renvoyé pour être affiché au navigateur.

De cette façon, la variable \$content est renvoyée de façon récursive à la fonction call_user_function_array et affiché avec un fonction echo. La classe Render joue un rôle crucial dans la préparation et l'affichage des données à l'utilisateur. Elle capture les données, les intègre dans les templates PHP et génère le contenu final de la page.

La méthode render assure que les templates sont correctement inclus et que les données sont disponibles dans le contexte des templates, permettant ainsi une présentation dynamique et cohérente des produits et autres informations sur la page. Du point d'entrée de l'application via index.php, qui initialise l'environnement et configure le routage, à la méthode RenderHome de HomeController, le flux de traitement des requêtes et de rendu de contenu est orchestré de manière fluide.

```

/**
 * La fonction render capture la sortie, fusionne les arguments avec les paramètres,
 * inclut les éléments header et footer, et renvoie le contenu final.
 *
 * @param string $template Le nom du template à afficher.
 * @param array ...$arguments Les arguments à fusionner avec les paramètres.
 * @return string|array Le contenu final du template.
 */
public function render(string $template, ...$arguments): string|array
{
    // Démarre la mise en mémoire tampon
    ob_start();

    // Ajoute par la méthode addParams() les données de seoConfig en fonction du
    $template, sinon par les données par défaut
    $this->addParams('seoConfig', $this->seoConfig->{$template} ?? $this->seoConfig-
>Default);

    // Fusionne les arguments avec les paramètres et les extrait dans des variables
    utilisables dans le template
    extract(array_merge($arguments[0] ?? [], $this->params));

    // Inclusion du header en passant les données SEO
    require_once __DIR__ . '/../../element/header.php';

    // Inclusion de la barre de recherche
    require_once __DIR__ . '/../../element/search.php';

    // ENH EXCEPTION MotorMvcException
    if (!file_exists(__DIR__ . DIRECTORY_SEPARATOR . "../../template/{$template}.php"))
    {
        throw new MotorMvcException(ExceptionEnum::TemplateNotFound);
    } else {
        // Inclusion du template
        require_once __DIR__ . "../../template/{$template}.php";
    }

    // Inclusion du footer
    require_once __DIR__ . '/../../element/footer.php';

    // Récupère le contenu mis en mémoire tampon et l'efface
    $content = ob_get_clean();

    // Retourne le contenu
    return $content;
}

```

Figure 9: Méthode Render gérant le rendu des templates avec controller

Cette architecture modulaire et robuste garantit une gestion efficace des requêtes utilisateur, un traitement sécurisé des données, et un affichage final optimisé, répondant aux attentes des utilisateurs.

VEILLE TECHNOLOGIQUE

Pendant le développement de notre projet, j'ai effectué différentes veilles technologiques pour garantir l'efficacité et la sécurité de notre application. Ces recherches m'ont permis de m'appuyer sur des ressources fiables et des méthodologies éprouvées, tout en restant à jour sur les meilleures pratiques du développement web.

Pour la mise en place de notre base de données et des requêtes SQL, j'ai consulté le site SQL.sh, une ressource précieuse pour comprendre et appliquer les concepts avancés de SQL. Cette plateforme m'a aidé à comprendre efficacement la mise en place de notre base de données. En parallèle j'ai fait des recherches sur la méthode Merise, une approche méthodologique pour concevoir et gérer des systèmes d'information de manière rigoureuse et structurée recommandé par l'un de mes anciens formateur. Grâce à cela, j'ai pu comprendre les modèles de données de manière claire et précise.

En parallèle, j'ai étudié la documentation PHP, en particulier pour la fonction call_user_func_array, qui est essentielle pour appeler des fonctions de manière dynamique avec un tableau d'arguments. Cette fonction s'est avérée particulièrement utile pour améliorer la flexibilité et la modularité de notre code, en permettant le développement de la logique de rendu.

De plus, la sécurité de notre application étant une priorité, j'ai également exploré la bibliothèque Sodium de PHP, reconnue pour ses capacités avancées en cryptographie. La documentation détaillée de Sodium m'a permis de comprendre la mise en place des mesures de sécurité pour protéger les données sensibles et prévenir les potentielles vulnérabilités.

Enfin, pour intégrer un système de paiement sécurisé, j'ai consulté la documentation de Stripe Checkout. Stripe est une solution de paiement largement utilisée pour sa simplicité d'intégration et sa conformité aux normes de sécurité les plus strictes. Grâce à la documentation exhaustive de Stripe, j'ai pu implémenter un système de paiement aux utilisateurs.

Ces veilles technologiques ont été cruciales pour le succès de notre projet, en nous permettant de construire une application robuste, sécurisée et conforme aux attentes des utilisateurs. En restant informé des dernières avancées et des meilleures pratiques.

CONCLUSION

Le projet de groupe Tea'Coffee, qui s'est déroulé sur un mois, a été une expérience particulièrement enrichissante. Travailler en équipe sur ce projet m'a permis de développer non seulement mes compétences techniques, mais aussi mes compétences en organisation et en collaboration. J'ai grandement apprécié la dynamique de groupe et la structure organisationnelle que nous avons mise en place, ce qui a permis de mener à bien ce projet dans les délais impartis.

D'un point de vue technique, j'ai énormément appris sur la structuration d'un projet web et sur les langages que nous avons utilisés, en particulier PHP et JavaScript. Ces connaissances m'ont permis de mieux comprendre les aspects fondamentaux du développement web et de renforcer mes compétences en programmation.

Cependant, je ressens une certaine frustration de ne pas pouvoir continuer ce projet avec l'équipe. J'aurais aimé avoir l'opportunité de faire évoluer Tea'Coffee en le passant sous un Framework, ce qui aurait permis de le rendre encore plus robuste et scalable. De plus, j'aurais apprécié de pouvoir consacrer davantage de temps à améliorer le front-end, afin de perfectionner l'interface utilisateur et l'expérience globale.

Cette expérience m'a également renforcé dans mon désir de travailler en entreprise, où je pourrai continuer à évoluer au sein d'équipes collaboratives. Je suis particulièrement enthousiaste à l'idée de commencer une alternance pour l'année à venir, car cela me permettrait de continuer à apprendre et à appliquer mes compétences dans un environnement professionnel tout en poursuivant ma formation académique.

En conclusion, le projet Tea'Coffee a été une étape clé dans mon parcours, m'apportant des compétences précieuses et une meilleure compréhension du travail en équipe dans le développement web. J'ai hâte de poursuivre sur cette lancée et de contribuer à de nouveaux projets en entreprise.

ANNEXES

```
version: "3.8"

services:
  db:
    image: mariadb:10.6
    restart: always
    environment:
      MYSQL_USER_FILE: /run/secrets/user_mysql
      MYSQL_PASSWORD_FILE: /run/secrets/password_mysql
      MYSQL_DATABASE_FILE: /run/secrets/database_mysql
      MYSQL_ROOT_PASSWORD_FILE: /run/secrets/password_root_mysql
      MYSQL_TCP_PORT: 55055
    volumes:
      - db_data:/var/lib/mysql
      - ./docker-data/dump/dump.sql:/docker-entrypoint-initdb.d/dump.sql
    secrets:
      - password_mysql
      - user_mysql
      - database_mysql
      - password_root_mysql

  nginx:
    build: ./docker-data/images/nginx/
    restart: always
    volumes:
      - nginx_conf:/etc/nginx/
      - ./backend-php:/www
      - ./docker-data/nginx_conf:/etc/nginx/conf.d
      - ./docker-data/ssl:/etc/nginx/ssl
      - ./docker-data/php_conf/ini:/usr/local/etc/php/conf.d
    # Port ajouté
    ports:
      # Port 80 mappé backend-php
      - "8880:80"
      # Port 81 mappé node-service
      - "8881:81"
      # Port 82 mappé node-jwt
      - "8882:82"
      # Port 443
      - "443:443"

  php:
    depends_on:
      - db
    hostname: teacoffee.dock
    restart: always
    secrets:
      - api_key_stripe
      - database_mysql
      - user_mysql
      - password_root_mysql
      - password_mysql
      - port_mysql
      - dsn_mysql
      - host_mysql
      - charset_mysql
    build:
      context: ./backend-php
      dockerfile: Dockerfile
    volumes:
      - ./backend-php:/www
    links:
      - db:db
```

Figure 10: Première partie du fichier docker-compose.yaml

```

phpmyadmin:
  depends_on:
    - db
  image: phpmyadmin:latest
  ports:
    - "7779:80"
  restart: always
  environment:
    PMA_HOST: db
    PMA_ROOT_PASSWORD_FILE: /run/secrets/password_root_mysql
    PMA_PORT: 55055
  secrets:
    - password_root_mysql
  links:
    - db:db

node:
  hostname: node.teacoffee.dock
  restart: always
  secrets:
    - api_key_stripe
  build:
    context: ./node-service
    dockerfile: Dockerfile
  volumes:
    - ./node-service:/www

node-jwt:
  hostname: jwt.teacoffee.dock
  restart: always
  secrets:
    - jwt_key
  build:
    context: ./node-jwt
    dockerfile: Dockerfile
  environment:
    - NODE_ENV=development
  volumes:
    - ./node-jwt:/app

secrets:
  jwt_key:
    file: ./docker-data/secrets/jwt_key.txt
  api_key_stripe:
    file: ./docker-data/secrets/api_key_stripe.txt
  database_mysql:
    file: ./docker-data/secrets/database_mysql.txt
  password_mysql:
    file: ./docker-data/secrets/password_mysql.txt
  password_root_mysql:
    file: ./docker-data/secrets/password_root_mysql.txt
  user_mysql:
    file: ./docker-data/secrets/user_mysql.txt
  port_mysql:
    file: ./docker-data/secrets/port_mysql.txt
  dsn_mysql:
    file: ./docker-data/secrets/dsn_mysql.txt
  host_mysql:
    file: ./docker-data/secrets/host_mysql.txt
  charset_mysql:
    file: ./docker-data/secrets/charset_mysql.txt

volumes:
  db_data:
  nginx_conf:

```

Figure 11: Deuxième partie du fichier docker-compose.yaml

The screenshot shows a Stripe Checkout payment page in TEST MODE. At the top left, there are navigation icons and a TEST MODE button. The main area is divided into two sections: a summary table on the left and a payment form on the right.

Payer

44,75 €

Café Cappuccino Corsé	3,00 €
Qté 1	
Café Affogato Corsé	12,50 €
Qté 1	
Café Americano Corsé	9,00 €
Qté 2	
	4,50 € chacun
Thé Blanc Noir	5,25 €
Qté 1	

Afficher les 6 articles ▾

Payer par carte

Informations de livraison

E-mail

Adresse de livraison

Nom complet

France

Adresse

Saisir l'adresse manuellement

Données de paiement

Informations de la carte

1234 1234 1234 1234

VISA MASTERCARD AMERICAN EXPRESS

MM / AA CVC

Les informations de facturation sont identiques aux informations de livraison

Enregistrer mes informations en toute sécurité pour le

Figure 12: Page de paiement Stripe Checkout

```

class StripePayment
{
    public function __construct()
    {
    }

    public function StartPayment($basket)
    {
        $stripeSecretKey = DockerSecrets::getSecrets(SecretsEnum::Api_Key_Stripe);
        try {
            \Stripe\Stripe::setApiKey($stripeSecretKey);

            setcookie('stripe', true, time() + 3600, '/');
        }

        $YOUR_DOMAIN = 'https://teacoffee.dock/';

        // Initialiser un tableau pour suivre le nombre d'occurrences de chaque produit
        $line_items = [];

        foreach ($basket as $product) {
            $found = false;
            // Passage de la variable $item en reference avec l'opérateur &
            foreach ($line_items as &$item) {
                // Si une valeur du tableau $line_items correspond alors on modifie la valeur
                quantity de + 1
                if ($item['price_data']['product_data']['name'] === $product->name) {
                    $item['quantity'] += 1;
                    $found = true;
                    break;
                }
            }
            if (!$found) {
                // Si aucune correspondance alors on ajoute la valeur directement
                $line_items[] = [
                    'quantity' => 1,
                    'price_data' => [
                        'currency' => 'EUR',
                        'product_data' => [
                            'name' => $product->name,
                        ],
                        'unit_amount' => $product->price * 100,
                    ],
                ];
            }
        }
    }

    $session = Session::create([
        'line_items' => $line_items,
        'mode' => 'payment',
        'success_url' => $YOUR_DOMAIN . 'stripe/success',
        'cancel_url' => $YOUR_DOMAIN . 'stripe/cancel',
        'billing_address_collection' => 'required',
        'shipping_address_collection' => [
            'allowed_countries' => ['FR'],
        ],
    ]);
} catch (Exception $e){

    unset($_COOKIE['stripe']);
    setcookie('stripe', '', -1, '/');
    throw new ClientExceptions(ClientExceptionEnum::KeyNotFound);
}

header('HTTP/1.1 303 See Other');
header('Location: ' . $session->url);
}
}

```

Figure 13: Classe StripePaiement et sa gestion des données avant envoie du formulaire