Programmation Web & Serveur

QuoiCouB'Achat

Site de e-commerce conçu par :

- Simoes Ferreira Thomas
- Cheramy Benoît
- Galli Gabriel
- Rathaux Antonin

Sommaire

1.1 / Contexte du Projet

En vue du projet de développement web & serveur, notre groupe a choisi de créer une plateforme de e-commerce en ligne.

Le thème choisi est celui de la culture populaire japonaise, nous pensons qu'avec l'ascension de celle-ci en Europe ces 10 dernières années, notamment grâce à la croissance des plateformes de streaming, ce genre de site de vente en ligne pourrait-être très demandé et ce pendant encore de longues années.

Dans ce contexte, notre récent site QuoiCouB'Achat propose à l'achat des articles liés à cette culture tel que la mode « Kawaii », les mangas / animés, le cosplay et autres...

Notre site Web est donc graphiquement en liaison avec la mode « Kawaii », le nom du site y est aussi étroitement lié.

Figure 1 : Illustration d'accueil

1.2 / Cibles du Projet

Le public cible de notre site e-commerce est diversifié, mais il partage un intérêt commun pour la culture pop japonaise, y compris les éléments kawaii, les mangas et les animés. Nous nous adressons à plusieurs sections de consommateurs passionnés par ces domaines spécifiques :

- Les adolescents et jeunes adultes passionnées :

  Principalement des adolescents et des jeunes adultes, âgés de 15 à 30 ans, qui sont passionnés par la culture pop japonaise. Leurs intérêts seront principalement nos produits dérivés de mangas / animés populaires.

- Collectionneurs et Fans Dévoués :

  Des collectionneurs avides et des fans dévoués de certaines franchises ou séries animées. Intérêt porté sur des articles de qualité tel que des figurines de valeur ou encore des vêtements de collection.

- Amateurs de Cosplay :

  Les personnes qui s'adonnent au cosplay et recherchent des costumes, accessoires et produits liés à leurs personnages préférés. Leurs recherches seront donc portées sur les costumes, accessoires ou vêtements.

- Cadeaux pour les Enfants et les Adolescents :

  Parents, amis et proches cherchant des cadeaux pour des enfants et des adolescents qui apprécient la culture pop japonaise. Notre site propose donc des jeux de société ou encore des figurines de toutes gammes.

Notre stratégie est donc de viser un public large en proposant des produits hauts de gamme comme de qualité plus abordable pour répondre à tous les budgets, tout en offrant une variété articles pour couvrir un maximum de demandes.

1.3 / Fonctionnalités du Projet

Notre site propose les fonctionnalités élémentaires d'un site de e-commerce robuste, pratique et conviviale en fournissant une plateforme qui répond aux besoins spécifiques de notre publique :

- Un catalogue de produit

  Le site propose un affichage détaillé des produits disponibles contenant toutes les informations nécessaires.

- Un filtrage des articles

  Possibilité d'utiliser des filtres, des tris ou encore une recherche écrite rapide.

- Gestion de compte utilisateur

  Les utilisateurs pourront créer et se connecter à un compte personnel sécurisé pour passer des commandes par exemple.

- Un panier d'achat

  Un système de panier simple pour ajouter des articles sans les perdre et procéder à l'achat ou les supprimer plus tard. Possibilité de modifier les quantités, calculer le prix total des articles...

- Contact pour des réclamations

  Nous avons créé une messagerie pour que les utilisateurs puissent nous contacter en cas de réclamations sur une commande.

- Interface intuitive

  Pour assurer un bonne expérience utilisateur, la navigation est simple et intuitive tout en respectant les différentes lois d'ergonomies.

- Interface Admin

  Pour l'administration, les utilisateurs possédant le rôle auront accès à une interface leur permettant d'ajouter, modifier et supprimer certaines données. Elle permet aussi de répondre aux différents messages envoyés par les clients.

2.1 / Framework pour le développement

Nous avons choisi le Framework Symfony comme base pour le développement de notre site, c'est un Framework PHP robuste et modulaire.

Symfony offre une architecture modulaire qui facilite le développement et la maintenance du code. Les composants Symfony peuvent être réutilisés de manière efficace, améliorant ainsi l'évolutivité du projet.

Ce Framework propose des fonctionnalités de sécurité avancées ainsi qu'une gestion des erreurs précises et efficaces.

La grande communauté Symfony offre un support continu, ce qui facilite la résolution des problèmes et l'adoption de meilleures pratiques.

Symfony est aussi connu pour ses bonnes performances, ce qui garantit aux utilisateurs une expérience fluide.

Figure 2 : Logo Symfony

Nous avons aussi utilisé le Framework « Tailwind CSS » pour nous permettre de créer une belle interface utilisateur de manière simple et efficace, ce qui nous a permis de réduire le temps de développement pour l'interface et nous concentrer sur le backend.

2.2 / Technologies utilisées

Les langages de programmation utilisés pour concevoir notre site sont :

- PHP, utilisé côté serveur pour la génération de contenu dynamique et autres...
- HTML, utilisé pour la structure de base des pages web.
- Twig : Moteur de template utilisé avec Symfony pour la gestion des vues...
- SQL, Langage de requête utilisé pour interagir avec la base de données.

La base de données est en MySQL qui est un système de gestion de BDD relationnelle et qui permet de stocker les différentes informations du site.

L'ORM utilisé est Doctrine qui, avec Symfony, facilite la manipulation de la BDD en utilisant des objets PHP plutôt que des requêtes SQL directes comme les fichiers de migrations.

Pour la collaboration de notre groupe, nous avons utilisé GitHub qui est une plateforme de gestion de code source utilisée pour le suivi des modifications ou encore pour la gestion des branches de code de chacun.

Notre serveur web est Apache qui est utilisé pour déployer Symfony et lancé depuis XAMPP.

Nous avons aussi utilisé des extensions de VScode comme « GitHub Pull Request » qui permet de synchroniser notre code avec la branche Master sur GitHub en fonction des pull requests des différents membres ou encore un bundle « EasyAdmin » pour nous aider à créer un backend admin efficace, pratique et agréable.

2.3 / Procédures d'installations

Voir le read.me sur le dépôt GitHub.

Voici le lien vers le GitHub https://github.com/beubeu28/quoicoubachat

Ci-dessous, le lien d'invitation (vous avez déjà été ajouté au projet mais au cas où)

« https://github.com/beubeu28/quoicoubachat/invitations »

3.1 / Modèle de données

La base de données est en MySQL comme dit précédemment et administrer sur PHPmyAdmin. Le fichier de remplissage de la BDD s'appelle data .sql et est constitué d'une simple commande « insert into article ».

Entité Article :

Attributs : id, nom, prix, description, stock, fichier, image_name, date.

Entité Commande :

Attributs : id, utilisateurid_id (clé étrangère faisant référence à l'entité User), date, montant_total, statut.

Relation : Une commande est liée à un utilisateur (relation ManyToOne).

Entité DetailCommande :

Attributs : id, commandeid_id (clé étrangère faisant référence à l'entité Commande), articleid_id (clé étrangère faisant référence à l'entité Article), quantite, prix_unitaire, prix_total.

Relations :

Une ligne de détail de commande est liée à une commande (relation ManyToOne).

Une ligne de détail de commande est liée à un article (relation ManyToOne).

Entité Messagerie :

Attributs : id, utilisateur_id (clé étrangère faisant référence à l'entité User), user_mail, motif, description, date, fichier, image_name.

Relation : Un message de messagerie est lié à un utilisateur (relation ManyToOne).

Entité User :

Attributs : id, email, roles, nom, prenom, password, telephone.

Aucune relation explicite n'est définie dans le schéma de migration, mais il semble que l'entité "User" soit indépendante.

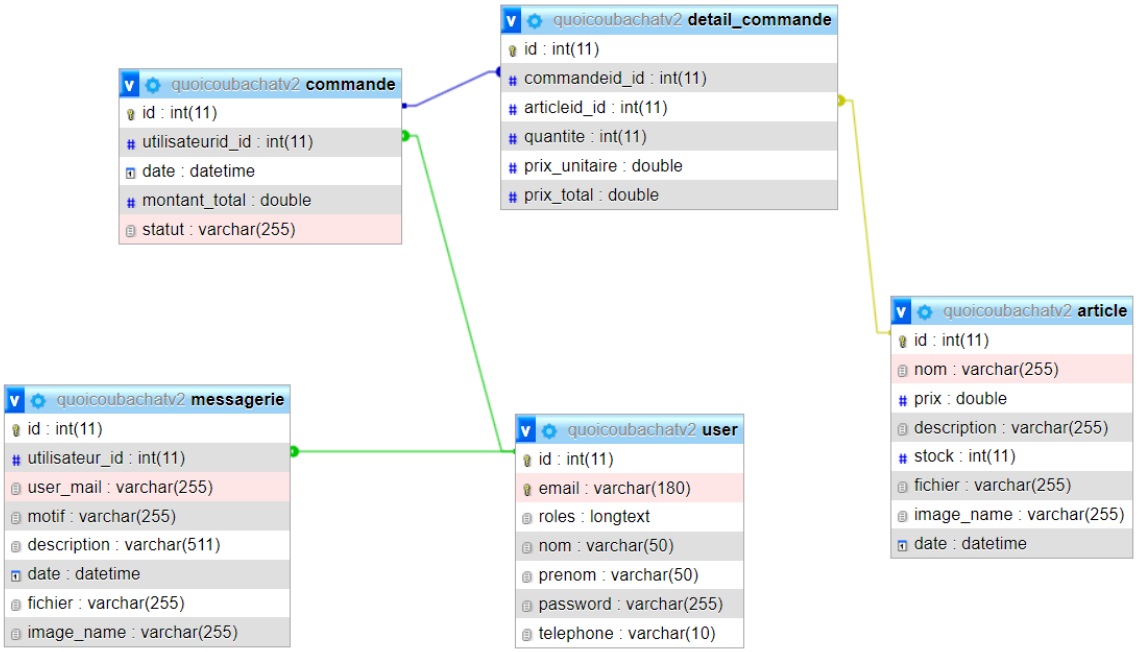


Figure 3 : MLD de la BDD

3.2 / Découpage des controllers

Les controllers contiennent des classes PHP qui définissent des méthodes (actions) pour traiter les requêtes HTTP, interagir avec le modèle, et renvoyer des réponses aux clients. Les actions sont associées à des routes, le contrôleur va utiliser les fonctions des repository pour ensuite afficher le résultat avec les fichiers .twig.

Ils « contrôlent » les fonctionnalités du site.

Dans l'architecture du projet, nous avons 2 groupes de contrôleurs différents, un pour la partie admin et l'autre pour les contrôleurs de base.

Nous avons 7 fichiers de contrôleurs de base pour le site :

- AccueilController, permet d'afficher des articles et utiliser une barre de recherche.

- ArticleController, permet d'afficher des articles dans la collection, d'utiliser les tris, de créer un article (avec le rôle manager), d'ajouter au panier, d'afficher les informations d'un article, de modifier un article (manager) et d'afficher le panier.

- CommandeController, permet d'afficher toutes les commandes (manager) ainsi que leurs détails, créer une nouvelle commande, modifier une commande (manager), valider le panier, afficher ses commandes, supprimer une commande (manager).

- DetailCommandeControlleur, permet d'augmenter/diminuer le nombre d'un article dans le panier, lister toutes les commandes, créer un nouveau détail de commande (quand on valide un panier), afficher les détails plus précis d'une certaine commande.

- MessagerieController, d'afficher les messages envoyés, créer un nouveau message, supprimer un message.

- SecurityController, permet de se login/logout/register.

- UserController, permet d'afficher le profil.

Nous avons aussi 6 fichiers de contrôleurs utilisés en mode administrateur pour le site qui permettent de lire, créer, modifier, supprimer :

- ArticleCrudController, permet d'afficher les articles, désactive les actions « ajouter » et « modifier » sur les articles.

- CommandeCrudController, affiche les commandes, enlève l'action de créer une commande.

- DashboardController, affiche un tableau de bord conçu pour l'administration afin de naviguer, configure les différents éléments de ce menu.

- DetailCommandeCrudController, affiche les commandes et leur détail, enlève les actions d'ajout, modification et suppression.

- MessagerieCrudController, affiche tous les messages, enlève la possibilité d'en envoyer.

- UserCrudController, affiche les utilisateurs.

3.3 / Présentation de templating

Le moteur de template que nous avons utilisé pour le site est Twig (moteur par défaut avec Symfony).

Le fichier de structure de base du site est « base.hmtl.twig », il contient le header, qui est la barre de navigation avec le logo, le panier, le profil, les boutons de connexion/déconnexion ainsi que la barre de recherche, certains éléments cités sont inaccessibles depuis certaines page, exemple on ne peut pas envoyer un message si nous ne sommes pas connectés. Ce fichier contient aussi le footer qui affiche nos noms ainsi que des liens vers différents réseaux comme notre GitHub, il s'y trouve aussi une mention copyright pour la « PNHM Company ».

Des fichiers Twig sont utilisés pour chaque affichage sur le site et sont triés en fonction de sur quoi ils pratiquent leur action.

Par exemple le fichier /templates/accueil/index.html.twig comporte les balises nécessaires pour créer l'accueil du site, tel que la grande image d'accueil ainsi que son texte, l'affichage des 16 articles en dessous ect..

Pour citer d'autres exemples moins précis d'affichage avec les fichiers nous avons :

Toute la collection, le dashboard (admin), des pop-ups de confirmation d'actions (comme pour supprimer un message), les boutons, les tris/filtres ...

3.4 / Gestion des rôles et de la sécurité

L'objectif principal des mesures de sécurité prises sont de garantir la confidentialité, l'intégrité et la disponibilité des données. Les mesures prises sont les suivantes :

- Sécurisation des mots de passe, lorsque l'on s'inscrit sur le site, le mot de passe enregistré est scripté par un algorithme.

- Autorisations, nous utilisons des rôles pour définir les droits d'accès de chaqu'un des utilisateurs, il y a le rôle 'USER', 'ADMIN' et 'MANAGER'. Il est impossible de créer un compte admin ou manager d'une quelconque manière sur le site car ils doivent être créé depuis Symfony.

- Les administrateurs n'ont pas tous les droits, ils ne peuvent pas aller au-delà de leurs responsabilités.

- Utilisation de EasyAdmin pour avoir une interface administrateur solide et efficace.

Etant donné que le site ne sera pas mis en ligne, nous n'avons pas réalisé d'autres manœuvre de sécurité.

3.5 / Upload de fichiers

Le site propose un upload de fichier, il s'utilise soit en mode « manager » pour ajouter un article à la base de données, il permet de déposer une image d'illustration de l'article qui sera envoyé dans le dossier « public/images/ » qui est le dossier où sont rangés toutes les images des articles de base du site. Cet upload est utilisé une seconde fois dans les contacts laissant la possibilité aux utilisateurs de déposer une photo montrant précisément la raison de leur plainte et ainsi améliorer la compréhension du problème et sa résolution.

Pour réaliser cela, nous avons utilisé le « VichUploaderBundle », celui si consiste en 3 étapes à :

- configurer un mappage de téléchargement, ce qui signifie configurer l'endroit où le fichier doit être stocké, puis donner le chemin web vers ce répertoire et enfin donner un nom à ce mappage pour le réutiliser dans les étapes suivantes.

- créer un lien entre l'entité et le fichier que l'on veut upload, il faut donc indiquer au bundle quelle entité doit utiliser quel mappage. Ensuite créer un champ pour stocker le nom puis un autre pour stocker l'objet « UploadedFile »

- configurer les évènements du cycle de vie du fichier, comme est ce que le fichier doit-il être supprimé lorsque l'entité est supprimée ou encore est ce que le fichier doit être supprimé lorsqu'un nouveau fichier est téléchargé à la place, comme pour l'exemple des contacts.

3.6 / Usage des librairies communautaires

Nous avons déjà parlé des librairies communautaires dans les autres sections du compte rendu, mais pour les résumés :

EasyAdmin :

Description : EasyAdmin est utilisé comme interface d'administration générée automatiquement pour simplifier la gestion des entités Symfony.

Configuration : Personnalisation de l'apparence de l'interface d'administration en utilisant les fonctionnalités de configuration de EasyAdmin.

VichUploaderBundle :

Description : VichUploaderBundle est utilisé pour gérer le téléchargement des fichiers, comme les images de produits.

Configuration : Paramétrage des configurations VichUploaderBundle pour définir les emplacements de stockage, les noms de fichiers, etc.

Intégration : Intégration avec les entités Symfony pour gérer les téléchargements de fichiers de manière efficace.

Tailwind CSS :

Description : Tailwind CSS est utilisé comme Framework de style utility-first pour simplifier le développement front-end.

Configuration : Personnalisation de la configuration Tailwind pour définir des couleurs, des polices, des tailles, etc., conformément à l'esthétique de la culture pop japonaise.

Intégration : Application des classes Tailwind directement dans le HTML.

3.7 / Structure des fichiers de style

L'objectif principal de notre structure des fichiers de style est de tirer parti de la simplicité et de la flexibilité de Tailwind CSS. Nous l'avons utilisé comme Framework de style utility-first pour simplifier le développement front-end.

Etant donné que la partie style n'était pas importante au projet, nous n'avons pas créé de dossier pour le CSS. Le code CSS est directement mis dans les fichiers qu'ils modifient.

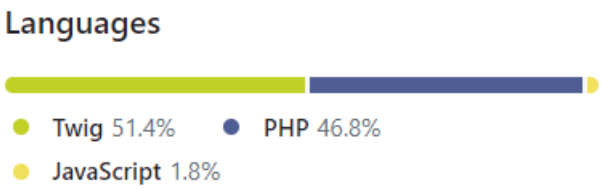
A l'aide de Tailwind, nous avons utilisé des composants préfabriqués de Tailwind CSS en ligne pour structurer le site.

Nous avons utilisé des transitions pour donner un peu de mouvements au site.

Avec les derniers projet Web que nous avons réalisés, nous avons conscience que la création de dossier pour le CSS est importante pour faciliter la maintenance et la cohérence visuelle.

3.8 / Structure des fichiers JS

Nous avons très peu utilisé JavaScript dans notre site :



En effet nos seules utilisations sont dans des fichiers de configurations, notamment pour Tailwind CSS ainsi que le bundler utilisé avec : « web pack » et enfin pour Bootstrap que nous n'utilisons pas.

Nous n'utilisons pas non plus de JS dans nos fichiers Twig.

4 / Conclusion et point à améliorer

En conclusion, la conception et le développement de notre site e-commerce dédié à la culture pop japonaise ont été guidés par une vision claire. À travers ce projet, nous avons cherché à créer une plateforme immersive qui non seulement répond aux besoins spécifiques de notre public cible, mais qui célèbre également la richesse et la diversité de la culture japonaise.

Points à améliorer :

- Mot de passe : pouvoir mettre des caractères spéciaux , pouvoir modifier son mot de passe

- téléphone : Pouvoir uniquement mettre des numéros valides.

- CSS : Mettre tout le code CSS dans des fichiers séparés et les organiser dans un dossier.

- Manager : Résoudre quelques petits bugs du manager comme le formulaire de modification d'article qui dépasse sur le footer...

- Message de validation pour le changement de statut