Faculté des sciences et de génie

UNIVERSITÉ LAVAL

Modèles et langages des bases de données pour l'ingénierie
GLO-2005
Hiver 2024

RAPPORT PROJET DE SESSION

Chedly Mouelhi - 537 071 862

Zeineb Ben Temime - 537 142 922

Naiel Mander - 537 148 699

Ahmed Mannai - 537 117 247

Destinataire
Monsieur Richard Khoury

Date de remise : 15 avril 2025

Table des matières

Table des figures

1 Énonciation du problème et des exigences :

Tout étudiant a un jour dû passer par le moment symbolique de quitter son nid familial et de s'installer ailleurs, en autonomie. C'est précisément à cet instant qu'il commence véritablement à goûter aux vraies difficultés de la vie quotidienne. Parmi ces nombreuses épreuves, l'une des plus marquantes est sans doute celle de meubler un nouvel appartement encore vide. Lorsqu'il entreprend ses recherches sur des sites en ligne ou dans des magasins ayant leurs propres marques, il se rendra rapidement compte que les prix affichés ne sont pas toujours à sa portée. Même en consultant des sites réputés pour proposer des prix abordables, tels que Marketplace par exemple, il n'est jamais garanti de trouver ce que l'on recherche, perdu au milieu d'une multitude d'articles qui ne sont pas nécessairement des meubles.

C'est dans cette optique que notre site, intitulé Style&Home, a vu le jour. Il se présente comme un véritable remède à ce problème fréquent. Style&Home est une plateforme d'achats et de ventes exclusivement dédiée aux meubles et articles de maison. Elle a été conçue dans une logique participative : permettre au peuple de fixer les prix pour le peuple. Elle constitue également un avantage majeur pour les personnes souhaitant vendre leurs meubles sans risquer de voir leur annonce se noyer au milieu d'autres objets divers et variés qui n'ont aucun lien avec l'ameublement.

Pour subvenir correctement aux besoins des utilisateurs de notre plateforme, plusieurs exigences fonctionnelles doivent être respectées. Tout d'abord, chaque utilisateur doit pouvoir consulter l'ensemble des articles affichés de manière claire et intuitive. Il doit également avoir accès à des outils efficaces lui permettant de faciliter sa recherche d'un article spécifique. Ces articles doivent fournir toutes les informations nécessaires pour lui permettre de prendre une décision d'achat réfléchie et éclairée. De plus, l'utilisateur doit pouvoir mettre en vente un article de manière extrêmement accessible et simplifiée. Il doit également pouvoir acheter un article sans être confronté à des procédures trop longues, complexes, ou qui pourraient paraître suspicieuses et décourageantes.

En réponse à ces exigences, notre plateforme doit proposer une page fluide, ergonomique, et équipée d'un système de filtres précis permettant d'affiner les recherches au maximum selon la catégorie de l'article, sa marque et son prix. Évidemment, chaque article est accompagné d'une image représentative, d'un nom, d'une description détaillée, du nom d'utilisateur de son vendeur, de sa catégorie, de sa marque ainsi que de la quantité disponible en stock. En plus de ces informations essentielles, l'article fournit également la note moyenne attribuée par d'anciens acheteurs ainsi que leurs éventuels commentaires. En ce qui concerne la mise en vente d'un article, cette dernière doit être la plus accessible possible. Pour cela, il est indispensable de minimiser les informations demandées à l'utilisateur tout en conservant les données nécessaires : le nom de l'article, une image, le stock disponible, une description, la marque et la catégorie de l'article. Enfin, pour l'achat d'un article, le site ne demande pas plus que les informations nécessaires à la transaction : l'utilisateur devra déposer de l'argent dans un portefeuille virtuel lié à son compte via sa carte bancaire. Une fois le portefeuille crédité, l'achat sera effectué directement à partir de ce dernier. L'utilisateur pourra ensuite, à tout moment, retirer l'argent restant de son portefeuille vers sa carte bancaire.

Sur le plan technique, le client web se doit d'être confortable à l'usage et d'assurer une expérience utilisateur agréable et intuitive. Il est responsable de transmettre toutes les interactions de l'utilisateur au serveur d'application, qui en retour, doit renvoyer les informations nécessaires à afficher sur l'écran de l'utilisateur. Cet affichage doit être clair, lisible et compréhensible pour la quasi-totalité des utilisateurs, sans exception.

Toutes les informations affichées et non affichées à l'utilisateur doivent être stockées de manière sécurisée dans la base de données. Cette dernière est responsable de conserver l'intégralité des données de la plateforme et de répondre aux requêtes du serveur d'application lorsque celui-ci a besoin d'informations particulières.

Enfin, le serveur d'application agit comme un véritable intermédiaire entre la base de données et le client web. Il est responsable de l'intégralité de la logique métier du site : création des profils utilisateurs, gestion des articles, des achats, des avis, des transactions financières, et vérification des données envoyées par les utilisateurs. C'est lui qui veille à la bonne communication entre les différents niveaux du système afin d'assurer un fonctionnement optimal de la plateforme.

2 Modélisation des données :

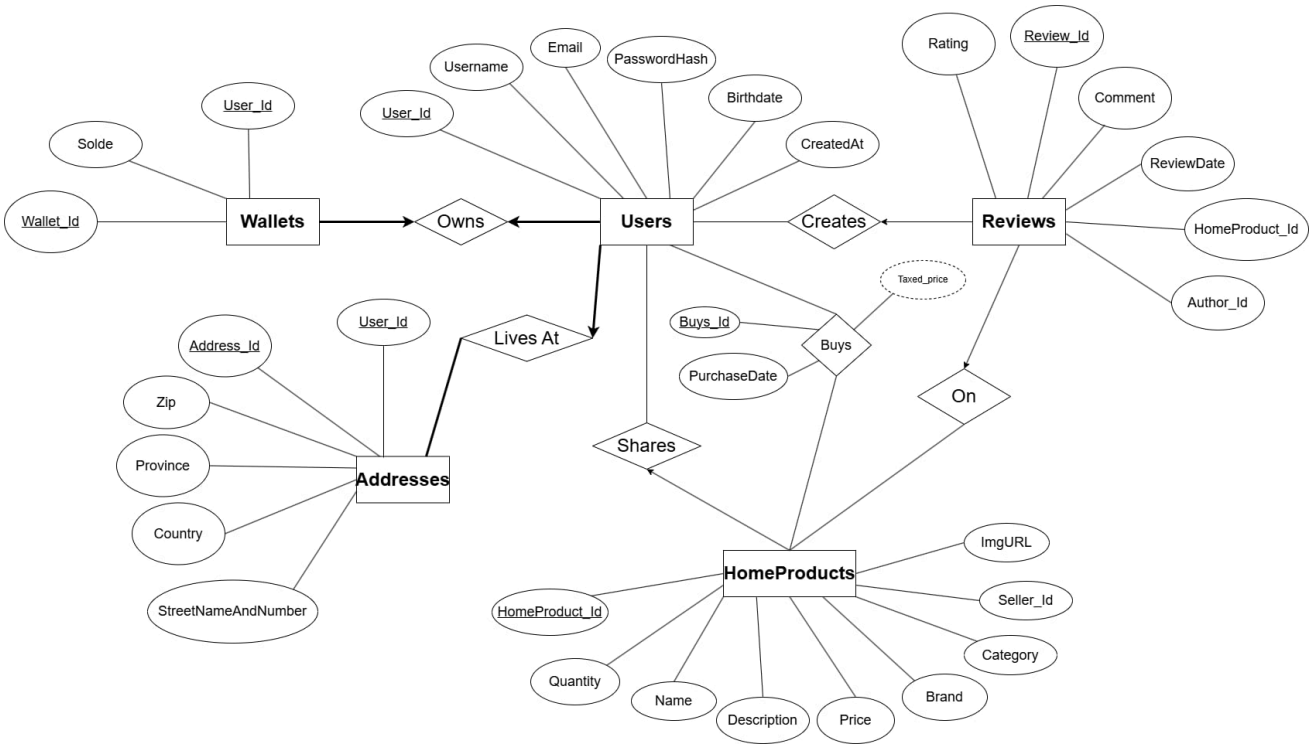

FIGURE 1 - modèle entités-relations

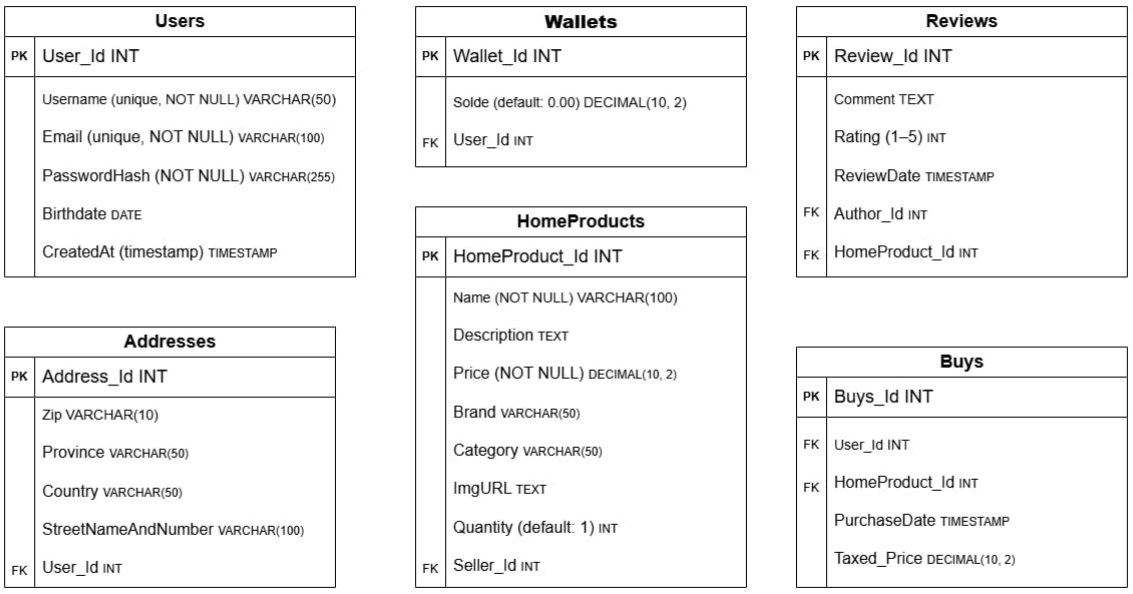

FIGURE 2 - modèle relationnel

3 Création de la base de données :

Notre base de données repose sur un total de 6 relations principales. Parmi ces relations, 3 d'entre elles présentent une cardinalité d'au moins 100, ce qui témoigne de la richesse et de la densité des informations stockées. De plus, 5 relations comportent un minimum de 5 attributs, en incluant les clés primaires et les clés étrangères, ce qui permet de répondre aux besoins fonctionnels complexes de la plateforme.

Users:

La relation Users contient les informations de base des utilisateurs : un identifiant unique (User_Id, clé primaire), un nom d'utilisateur (Username, unique), une adresse email (Email, unique), un mot de passe chiffré (PasswordHash), une date de naissance (Birthdate) et la date de création du profil (CreatedAt). Les champs User_Id, Username et Email sont soumis à une condition NOT NULL, imposant ainsi à l'utilisateur de fournir ces trois données uniques lors de son inscription. Une condition métier exige également que la date de naissance corresponde à un utilisateur âgé d'au moins 18 ans. Le champ CreatedAt est automatiquement rempli et ne peut jamais être nul (par défaut date d'aujourd'hui).

Addresses:

La relation Addresses stocke les informations d'adresse des utilisateurs : un identifiant unique (Address_Id, clé primaire), un code postal (Zip), une province (Province), un pays (Country), le nom et le numéro de rue (StreetNameAndNumber) ainsi que l'identifiant de l'utilisateur

concerné (User_Id, clé étrangère). L'identifiant unique de l'adresse n'est pas une donnée connue par l'utilisateur, mais le reste est fourni par ce dernier et reste sauvegardé dans la base de données. La clé étrangère User_Id fait référence à la relation Users avec une suppression en cascade (ON DELETE CASCADE).

Wallets :

La table Wallets gère les soldes virtuels des utilisateurs. Elle comprend un identifiant unique (Wallet_Id, clé primaire), un champ Solde de type DECIMAL(10,2) initialisé à 0.00 par défaut, et l'identifiant de l'utilisateur (User_Id, clé étrangère). Le champ User_Id fait référence à la relation Users.

HomeProducts :

La relation HomeProducts est au cœur de notre application. Elle contient les attributs suivants : identifiant unique du produit (HomeProduct_Id, clé primaire), nom de l'article (Name, NOT NULL), description (Description), prix (Price, NOT NULL), marque (Brand), catégorie (Category), image (ImgURL), quantité en stock (Quantity, valeur par défaut à 1) et identifiant unique du vendeur (Seller_Id, clé étrangère). La clé étrangère Seller_Id pointe vers la relation Users. Le champ Quantity est automatiquement décrémenté après chaque achat.

Reviews :

La table Reviews permet aux utilisateurs de noter et commenter les articles. Elle contient un identifiant de commentaire (Review_Id, clé primaire), le texte du commentaire (Comment), une note sur 5 (Rating), une date de publication (ReviewDate), l'identifiant unique de l'auteur du commentaire (Author_Id, clé étrangère) et celui de l'article concerné (HomeProduct_Id, clé étrangère). La relation entre HomeProduct_Id et Reviews est définie avec suppression en cascade (ON DELETE CASCADE). Une contrainte supplémentaire est imposée : un utilisateur ne peut laisser qu'un seul avis par produit (UNIQUE(Author_Id, HomeProduct_Id)) à condition qu'il ait réellement acheté l'article.

Buys :

La table Buys conserve l'historique des achats réalisés sur la plateforme. Elle contient un identifiant unique d'achat (Buys_Id, clé primaire), l'identifiant unique de l'acheteur (User_Id, clé étrangère), celui de l'article acheté (HomeProduct_Id, clé étrangère), la date de l'achat (PurchaseDate) ainsi que le prix après taxes (Taxed_Price). Ce dernier est un attribut dérivé calculé automatiquement à partir du prix initial via un trigger SQL, représentant le prix avec une taxe fixe de 15 %. Afin d'assurer l'intégrité des données, les clés étrangères User_Id et HomeProduct_Id relient cette table à d'autres entités de la base : User_Id fait référence à la table Users, identifiant l'acheteur de chaque produit, et HomeProduct_Id fait référence à la table HomeProducts, précisant quel article a été acheté.

Génération des données :

Toutes les tables ont été peuplées avec des données réalistes, cohérentes et représentatives du monde réel. Nous avons généré 100 articles dans la table HomeProducts, inspirés de produits authentiques issus de marques bien connues telles que IKEA, Maison du Monde, Conforama et ZaraHome. Chaque produit contient des informations complètes : nom, description, prix, image, marque et catégorie. La table Users contient 20 utilisateurs générés avec des données valides : noms, prénoms, adresses email, dates de naissance réalistes. Pour chaque utilisateur, un portefeuille (Wallet) est automatiquement créé via un trigger SQL. En revanche, les données d'adresse (Address) sont saisies manuellement par l'utilisateur lors de l'inscription, afin de respecter le processus de collecte volontaire d'informations. Nous avons également généré 200 enregistrements dans la table Buys et 200 dans la table Reviews. Les avis sont uniquement associés à des produits réellement achetés, en conformité avec notre logique métier qui interdit à un utilisateur de commenter un produit qu'il n'a pas acheté. Ces données ont permis de simuler un environnement riche, dynamique et représentatif d'une utilisation réelle de la plateforme. Toutes ces données ont été regroupées dans un fichier nommé « population_data.sql », utilisé pour l'initialisation automatique de la base de données en environnement de test.

4 Création des requêtes et des routines :

Gâchettes : Il y a 9 gâchettes dans notre base de données qui servent à gérer automatiquement quelques règles métier et contraintes d'intégrité non directement gérées par les mots-clés SQL. Ces gâchettes jouent un rôle essentiel dans la logique applicative, en assurant notamment la cohérence des transactions, la gestion des stocks et la sécurité des données.Pour résumer les fonctionnalités des gâchettes principaux:

- ➢ after_user_insert : crée automatiquement un Wallet et une Address vide à chaque nouvel utilisateur enregistré.
- ➢ before_buy_check_stock : bloque toute tentative d'achat si le stock est à zéro pour le produit visé.
- ➢ before_buy_check_balance : empêche l'achat si le solde du Wallet de l'utilisateur est insuffisant, en effectuant également le calcul du prix taxé (avec une taxe de 15%) et en débitant le montant automatiquement.
- ➢ after_buy_credit_seller : crédite le Wallet du vendeur après l'achat.
- ➢ after_buy_decrease_stock : diminue le stock disponible du produit après un achat réussi.
- ➢ after_update_check_stock_zero : supprime automatiquement un produit dont le stock devient nul après une mise à jour.
- ➢ before_user_delete : assure le nettoyage des données (Wallets, Adresses, Produits) liées à un utilisateur supprimé.
- ➢ prevent_duplicate_buys : empêche un utilisateur d'acheter plus d'une fois le même article.

➢ prevent_review_without_purchase : bloque l'ajout d'un commentaire (Review) si l'utilisateur n'a pas acheté l'article concerné.

fonctions : Une fonction utilitaire a été définie pour soutenir nos contrôles dans les triggers :

➢ hasBoughtProduct(userId, productId) : retourne TRUE si l'utilisateur a déjà acheté le produit spécifié, FALSE sinon. Elle est utilisée notamment pour empêcher les évaluations injustifiées (voir trigger prevent_review_without_purchase).

requêtes : Certaines requêtes sont directement encapsulées dans les triggers pour automatiser des processus métiers critiques comme :

- Le calcul du Taxed_Price avec application d'une taxe.
- La gestion automatique du stock après un achat.
- Le suivi du solde des utilisateurs (crédit et débit).

L'ensemble de cette logique a été testé et validé, garantissant que les actions utilisateur respectent les règles métiers tout en allégeant le traitement côté serveur. Ces éléments offrent une base solide pour un système transactionnel sécurisé, robuste et cohérent.

5 Indexation et optimisation du système :

Pour optimiser l'exécution des requêtes dans notre système, nous avons pris en compte les requêtes les plus fréquemment utilisées par les utilisateurs et anticipé la charge de travail attendue. L'optimisation repose sur une indexation automatique des clés primaires et étrangères, générée par MySQL lors de la création des relations. Ces index permettent d'accélérer les recherches, les jointures, les contrôles d'intégrité référentielle et toutes les opérations de filtrage sur les colonnes clés. Ainsi, les recherches par identifiant (dans Users, HomeProducts, ou Buys) et les relations entre tables sont automatiquement optimisées. Les colonnes avec contrainte UNIQUE (comme Email, Username, Author_Id + HomeProduct_Id) bénéficient aussi de cette indexation.

Compte tenu de la structure actuelle de notre base de données et du volume simulé d'utilisateurs et de produits, l'indexation automatique est suffisante pour maintenir des performances élevées. Aucune indexation manuelle supplémentaire n'a été nécessaire, car toutes les requêtes critiques utilisent des clés déjà indexées par défaut.

6 Normalisation des relations :

La base de données Style&Home a été conçue en respectant rigoureusement les principes de normalisation afin de garantir la cohérence, l'intégrité et la clarté des données. Toutes les relations — Users, Wallets, Addresses, HomeProducts, Buys, et Reviews — sont organisées de façon à respecter les formes normales classiques. Chaque table ne contient que des attributs atomiques (1NF), aucune redondance directe ni dépendance partielle (2NF), et aucune dépendance transitive (3NF). Les dépendances fonctionnelles sont bien définies, avec une clé primaire claire et des attributs dépendants uniquement de cette dernière. À l'exception des relations intermédiaires comme Buys ou Reviews qui relient plusieurs entités, toutes les

relations respectent également la forme normale de Boyce-Codd (BCNF). Ainsi, aucune anomalie d'insertion, de mise à jour ou de suppression ne peut survenir, ce qui garantit une intégrité maximale des données tout au long du cycle de vie de l'application.

7 Sécurité de la base de données :

Dans notre projet Style&Home, nous avons intégré plusieurs mesures de sécurité essentielles afin de garantir l'intégrité, la confidentialité et la robustesse de la base de données face aux risques de fuites ou de corruptions.Premièrement, les mots de passes ne sont jamais stockés en clair, puisque ils sont hachés via une cryptographique avant l'insertion dans la base, ce qui rend la récupération directe des mots de passes si il y a une fuite de données.

Pour renforcer la sécurité du processus de connexion sur notre site, nous avons intégré un système d'authentification par e-mail basé sur un code OTP (One-Time Password). Lorsqu'un utilisateur tente de se connecter, ses identifiants (nom d'utilisateur et mot de passe) sont d'abord vérifiés. S'ils sont corrects, un code temporaire à 6 chiffres est généré automatiquement et envoyé par e-mail à l'adresse associée au compte. L'utilisateur doit ensuite entrer ce code dans un champ dédié pour finaliser l'authentification. Ce système permet de limiter les connexions non autorisées même si le mot de passe est compromis, et offre une protection supplémentaire sans nécessiter d'outils externes. Cette fonctionnalité repose sur un service SMTP sécurisé (via Gmail) configuré en backend avec des variables d'environnement pour éviter toute exposition de données sensibles.

Le champ createdAt dans les relations Users, Buys(PurshaseDate) et Reviews(ReviewDate), permet un suivi précis des actions dans le système, ce qui contribue à la traçabilité et à l'analyse de potentiel comportement suspect.L'authentification des utilisateurs est renforcée via la vérification des identifiants *uniques* (Username, Email) et des contraintes d'intégrité strictes.

Les champs sensibles comme les informations d'achat (Buys) et les notes/commentaires (Reviews) sont protégés.Les champs sensibles comme les informations d'achat (Buys) et les notes/commentaires (Reviews) sont protégés. Toute tentative d'insertion multiple par un même utilisateur est bloquée via une contrainte UNIQUE (Author_Id, HomeProduct_Id).

Pour prévenir les injections SQL, nous avons verrouillé les interactions avec la base de données via des requêtes paramétrées. Cela nous protège contre les attaques d'injection SQL, en empêchant l'exécution de code arbitraire dans nos requêtes.Un exemple de ces requêtes est: cursor.execute("SELECT * FROM Users WHERE Email = %s", (email,)) .

Grâce à l'utilisation de triggers automatisés, plusieurs couches de contrôle garantissent l'intégrité du système.Les transactions sont automatiquement rejetées si l'utilisateur a déjà acheté l'article, ou si l'article est en rupture de stock et Les transactions sont automatiquement rejetées si l'utilisateur a déjà acheté l'article, ou si l'article est en rupture de stock.

8 Implémentation de la logique d'affaire :

Le backend de Style&Home repose sur une architecture claire, modulaire et entièrement basée sur Flask. La logique d'affaire est répartie dans trois couches bien distinctes : les routes Flask, les services Python, et les repositories qui exécutent les requêtes SQL.

Les routes, comme user_routes.py, wallet_routes.py, ou homeproduct_routes.py, reçoivent les requêtes HTTP (GET, POST, PUT, DELETE), valident les données reçues, et les transmettent aux services appropriés. Chaque groupe de routes est organisé par entité (utilisateurs, produits, achats, etc.), ce qui facilite leur maintenance. Les réponses retournées sont toujours structurées de façon cohérente, accompagnées de messages explicites et de codes d'état HTTP adaptés aux situations.

La logique métier se trouve dans les fichiers *_service.py, où toutes les règles fonctionnelles sont implémentées. Lors de l'inscription d'un utilisateur, un portefeuille vide est automatiquement généré grâce à un trigger SQL, tandis que l'adresse est renseignée manuellement à partir du formulaire d'inscription. Lors d'un achat, la quantité de stock est vérifiée, le prix TTC est calculé automatiquement (en appliquant une taxe de 15 %), le portefeuille de l'acheteur est débité et celui du vendeur crédité. Toute tentative d'achat sans solde suffisant est interceptée. Pour les avis, un utilisateur ne peut commenter un produit que s'il l'a effectivement acheté. Bien que cette règle soit définie comme exigence fonctionnelle, elle n'est pas encore imposée automatiquement par le backend dans l'état actuel du projet. De plus, il est impossible pour un utilisateur de publier plus d'un avis sur le même produit, grâce à une contrainte UNIQUE dans la base et une gestion d'erreur dans le code Python.

Les fichiers *_repository.py centralisent les requêtes SQL. Cette séparation entre la logique métier et l'accès aux données renforce la clarté et la modularité du projet. Les fonctions SQL sont paramétrées et sécurisées contre les injections, ce qui garantit la fiabilité des opérations.

Toutes les erreurs critiques sont capturées par des blocs try...except, assurant une tolérance aux fautes. En cas d'erreur (produit introuvable, utilisateur inexistant, mot de passe incorrect, etc.), un message structuré est renvoyé au frontend avec un code HTTP cohérent. Ce système est appliqué uniformément dans toutes les routes.

Le système d'authentification est sécurisé et basé sur deux étapes. Lorsqu'un utilisateur se connecte, ses identifiants sont d'abord vérifiés. Ensuite, un code de vérification (OTP) est généré et envoyé à son adresse e-mail. Une fois ce code saisi correctement, un jeton JWT est généré, permettant une session sécurisée.

Le découpage du backend en trois niveaux est respecté rigoureusement. Le frontend (client) déclenche les requêtes, le serveur Flask applique la logique métier et fait le lien entre les couches, tandis que la base de données MySQL assure la gestion sécurisée des données, des contraintes d'intégrité et des automatisations via des triggers.

Cette organisation assure un système robuste, évolutif et professionnel, parfaitement adapté aux exigences du projet Style&Home.

9 Implémentation de l’interface utilisateur :

Pour implémenter l’interface utilisateur, nous avons opté pour la méthode classique sans react: Html/CSS/vanilla javascript. L’interface finale a pour objectif de rendre l'expérience de l’utilisateur agréable: l’affichage doit être clair et compréhensible pour que l’utilisateur puisse naviguer sans problèmes et atteindre sa destination en un temps minimum. Pour garantir la réalisation de cet objectif, notre interface respecte le cheminement suivant:

Page d'accueil: la page de départ comporte un accès à la plupart des autres fonctionnalités(seront mentionnés plus tard) à condition d’être connecté à son compte ou, si on a pas encore de compte, inscrit.

S’inscrire et se connecter: cliquer sur n’importe quelle bouton sur la page d’accueil en étant déconnecté dirige automatiquement sur la page de connexion, si l’utilisateur n’a pas encore de compte, il suffit pour lui de cliquer sur le bouton “j’ai pas encore de compte” et l’utilisateur sera dirigé vers la page d’inscription.L’inscription faite, il sera redirigé vers la page connexion pour se connecter.

Ajout solde: une fois connecté, l’utilisateur est automatiquement redirigé vers la page portefeuille pour ajouter et retirer de l’argent à son portefeuille virtuelle, il peut aussi ignorer cette étape en cliquant “commencer les achats”

Page d'accueil fonctionnalités: La page d'accueil contient des boutons de filtrages pour catégories(amène vers la page home avec le filtre déjà activé selon le bouton cliqué), une icône profil avec un menu déroulant contenant profil et se déconnecter, un bouton “voir mon solde” qui redirige directement vers la page portefeuille et un bouton “commencer achats” pour aller à la page home.

Page Home: c’est la page clé du site, où on trouve les articles affichés par rangées(le nombre par rangée change selon la grandeur de l’écran) de haut en bas. Il y a un filtre pour catégories et marques avec des menus déroulants et pour prix avec une case où on peut limiter le prix affiché et il y a une barre de recherche pour des recherches plus précises.

Voir en détails et acheter:Depuis l’article, l’utilisateur peut cliquer sur “voir en détails” pour que plus d’informations s’affichent et peut aussi cliquer sur acheter pour procéder à l’achat du produit.Le popup qui apparaîtra donnera les informations de l’achat et l’option de commenter sur l’article.

Profil: la page profil comporte des données cruciales de l’utilisateur, et une petite interface pour mettre en vente son article en entrant toutes les données nécessaires.

Quelques conditions sont ajoutées pour garantir qu’il y a que des données justes qui parviennent à la base de données, comme la condition qui vérifie si “mot de passe “ correspond à “vérifier mot de passe”.

10 Test du système:

Tout au long du développement, des tests rigoureux ont été réalisés afin d'assurer le bon fonctionnement global du système. Chaque route définie dans l'API a été testée via Postman, et toutes les requêtes ont répondu avec succès, confirmant ainsi la stabilité des échanges entre le client et le serveur.

Du côté de la base de données, la compilation complète du script SQL ne produit aucun message d'erreur, ce qui garantit une définition correcte de toutes les relations, clés, contraintes et déclencheurs. Chaque structure a été vérifiée individuellement et en cohérence avec l'ensemble du système.

Enfin, une validation stricte des entrées a été mise en place : chaque champ d'entrée, qu'il provienne d'un formulaire ou d'une requête, est contrôlé pour rejeter tout format invalide (par exemple, empêcher une chaîne de caractères dans un champ numérique). Cette couche de validation permet d'éviter les erreurs utilisateurs tout en renforçant la sécurité de l'application.

11 Accessibilité du système:

Comme mentionné plusieurs fois précédemment, un de nos objectifs les plus importants est de rendre notre site le plus compréhensible, accessible et clair possible pour la majorité des utilisateurs.L'aspect général qui permet à accomplir cet objectif est la simplicité de la navigation dans le site:Le nombre de boutons est minime et ils sont assez grands pour attirer l'attention avec une écriture simpliste et lisible.

On a ajouté aussi un facteur majeur qui pourra améliorer l'expérience de l'utilisateur: un guide.nous avons développé et intégré un ChatBot interactif directement sur la page d'accueil. Ce ChatBot est capable de répondre instantanément à des questions fréquentes concernant la livraison, les retours, le paiement, les commandes, le solde du portefeuille, les produits proposés, ou encore la publication d'articles. Il comprend également les messages de type "bonjour" et peut même raconter une blague. Le traitement des messages intègre une normalisation automatique (gestion des majuscules, accents, espaces) afin d'assurer une reconnaissance fiable des mots-clés, même en cas de variations dans la saisie. Cette fonctionnalité renforce l'accessibilité du site et permet aux utilisateurs d'obtenir rapidement des réponses sans devoir consulter une FAQ.

12 Gestion de l'équipe et organisation du travail :

La réussite du projet a reposé sur une organisation structurée et une collaboration active entre les membres de l'équipe dès les premières étapes. L'idée du projet n'a pas été choisie au hasard : avant même le début du développement, nous avons évalué plusieurs propositions en tenant compte de leur faisabilité, de leur pertinence collective et des compétences disponibles. Ce travail préparatoire a permis de lancer le projet sur des bases solides.

Une fois le projet amorcé, nous avons mis en place des réunions hebdomadaires régulières. Celles-ci ont joué un rôle essentiel pour faire le point sur l'état d'avancement, discuter des

problèmes rencontrés et ajuster les priorités. Grâce à cette communication continue, nous avons pu maintenir une vision partagée des objectifs et renforcer la cohésion du groupe.

Pour assurer une coordination technique efficace, GitHub a été utilisé tout au long du projet comme outil de gestion de versions et de collaboration. Cette méthode a renforcé la qualité du développement et réduit les conflits lors des fusions.

La répartition des tâches s'est appuyée à la fois sur les compétences spécifiques de chacun (interface utilisateur, base de données,logique d'affaire ..) et sur les préférences individuelles. Cette approche a non seulement favorisé une meilleure implication des membres de l'équipe, mais aussi une exécution plus fluide des différentes composantes du projet.

En somme, la combinaison de méthodes de coordination claires, d'un outil collaboratif efficace et d'une répartition réfléchie des responsabilités a permis de maintenir une dynamique de travail constructive tout au long du projet.

Déclaration d'utilisation de l'IA :

Dans le cadre de ce projet, nous avons utilisé l'intelligence artificielle de manière éthique et encadrée, conformément aux attentes académiques. L'IA a été employée pour : à peupler la base de données de manière réaliste et cohérente (noms, marques, descriptions), Résoudre certains bugs de logique ou de syntaxe dans le backend, Explorer les meilleures pratiques et outils à adopter pour notre architecture (ex. triggers, structuration des routes, gestion des erreurs). Aucune portion du code ou de la documentation n'a été copiée sans modification ni compréhension. L'IA a été utilisée comme un outil d'assistance complémentaire, au même titre qu'un moteur de recherche ou une documentation technique. Toutes les décisions de conception ont été prises par l'équipe de manière critique, en fonction des besoins spécifiques de notre application.