



SneakersMarket



MANUEL TECHNIQUE

*Travail Pratique Individuel
CFPT Informatique
Triano Léo*

2024

Table des matières

Introduction	5
Versions.....	5
Mandataire	5
Mandants.....	5
Installation	5
Résumé du cahier des charges	6
Matériel à disposition.....	6
Description du projet.....	6
Pages.....	6
Fonctionnalités	7
Points spécifiques.....	9
Livrables	9
Organisation	10
Méthodologie	10
S'informer	10
Planifier	10
Décider.....	10
Réaliser.....	10
Contrôler	10
Évaluer.....	10
Product Backlog.....	11
Analyse fonctionnelle	14
Use Case.....	14
Plan de navigation	15
Description des fonctionnalités majeures	15
Consulter la liste des Sneakers en vente	15
Filtrer les paires par marques.....	15
Pagination.....	16
Barre de recherche	16
Réserver une paire.....	16
Mettre une paire en favoris	16
Profil	17

Gérer les utilisateurs.....	17
Maquettes.....	18
Analyse organique.....	22
Modèle conceptuel de données.....	22
Modèle logique de données	23
Dictionnaire de données	24
USERS.....	24
SNEAKERS.....	24
MEETPOINT	24
TRANSACTIONS	25
FAVORITE.....	25
BRANDS.....	25
SSTATUT	25
TSTATUT.....	25
USTATUT	26
MODELS.....	26
ECOLE	26
VISIBILITY.....	26
ROLES	26
DATEMEETPLACE.....	26
Structure du projet	27
Modèle	27
Vue	27
Contrôleur.....	27
Description des méthodes et classes principales	28
Classe Home.....	28
Classe Gestion	28
Classe Admin.....	29
Classe Profil	29
Analyse des fonctionnalités majeures	29
Mettre une paire en vente	29
Réserver une paire.....	30
Ajouter une paire en favoris.....	30

Programmes utilisés.....	31
Visual Studio Code – 1.88.....	31
DBeaver – 23.1.5	31
Gitlab – 16.10.3.....	31
Copilot – Free	31
Pixlr – Free.....	31
Remove.bg – Free	31
Codepen - Free	31
Canva – Free	31
Tests	32
Plan de tests	32
Déroulement des tests	36
Planning	37
Planning prévisionnel	37
Planning effectif.....	38
Analyse de similitudes entre les deux plannings	38
Conclusion	39
Difficultés rencontrées.....	39
Améliorations possibles	39
Bilan personnel.....	40
Bibliographie.....	40
Glossaire	41
Annexes	41

Introduction

Ce document détaille les informations importantes qui ont permis de réaliser le projet SneakersMarket. Ce projet est un travail de diplôme, dans le cadre du TPI. Celui-ci me permet de valider les compétences et le savoir acquis lors de ces 4 années de CFC informatique section développement d'application.

SneakersMarket est un site internet d'achat-revente de chaussure. Celui-ci met en relation deux catégories d'utilisateurs, les utilisateurs qui souhaitent revendre leurs chaussures inutilisées, et les utilisateurs qui souhaitent acheter de nouvelles chaussures. L'utilisateur non connecté peut voir la liste des paires mises en vente, mais doit se connecter pour pouvoir mettre en favoris ou réserver la paire. Seules les paires marquées comme visibles et n'ayant pas encore été réservées sont disponibles à la réservation.

Versions

VERSION N°	1.0
AUTEUR	Léo Triano
DATE	15.05.24
DESCRIPTION	Version finale pour le rendu du TPI

Mandataire

ÉLÈVE	Léo Triano - leo.trn@eduge.ch
MAITRE D'APPRENTISSAGE	M. Dido Malambu - edu-malambud@eduge.ch

Mandants

EXPERT N°1	M. Daniel Vanini - daniel.vanini@skyguide.ch
EXPERT N°2	M. Borys Folomietow - borys@folomietow.ch

Installation

Pour pouvoir utiliser SneakersMarket, vous aurez besoin de :

- Visual studio code 1.88
- DBeaver 23.1.5
- Git et un accès à Gitlab 16.10.3
- Une configuration wsl avec apache2 et mysql intallé

L'installation du site se fait comme ceci :

- Clonez le projet depuis Gitlab dans votre projet /var/www/html
- Importez la dernière version du script SQL dans DBeaver
- Renommez le fichier configSample.php situé dans src/Model en config.php et modifiez les informations pour la connexion à la base de données.
- Tapez « localhost » dans votre navigateur et sélectionnez SneakersMarket

Résumé du cahier des charges

Matériel à disposition

- Un PC standard école
- 1 écran
- Visual studio code
- Suite Office
- MySQL
- PHP

Description du projet

Réaliser un site web pour la vente et l'achat de sneakers entre étudiants du CFPT.

Pages

L'application comporte :

L'application comporte :

- Une page d'accueil contient un listing des différentes chaussures ajoutées
- La possibilité de chercher une chaussure par :
 - Son nom
 - Par marque
- Une page avec les détails d'une paire de chaussure
- Une page de connexion
- Une page de profil pour l'étudiant
- Une page contenant les chaussures favorites de l'étudiant
- Une page de gestion contenant les chaussures achetées et celles que souhaite vendre l'étudiant

Un tableau de bord pour l'administrateur contenant les fonctionnalités de gestion :

- Une page de connexion
- Une page de gestion des utilisateurs
- Avec le listing des chaussures qu'ils souhaitent vendre
- Avec le listing des chaussures qu'ils ont acheté
- Une page de gestion des points de rencontre (cafétéria Butin, cafétéria Ternier, cafétéria Bertrand)
- Une page de gestion des établissements (CFPT Mécanique, CFPT Construction, CFPT Informatique)

Fonctionnalités

3.3.1 Description des actions réalisables en tant qu'utilisateur anonyme :

- L'utilisateur visualise depuis la page d'accueil les chaussures disponibles :
- Seulement 10 paires de chaussure sont visibles, les autres sont accessibles via une pagination
- Une paire de chaussures est uniquement visible si son statut est "visible" L'utilisateur peut chercher une paire de chaussure :
 - En entrant son nom
- L'utilisateur peut filtrer les chaussures par :
- Marque
- L'utilisateur visualise le détail d'une paire de chaussure

3.3.2 Description des actions réalisables en tant qu'utilisateur connecté (étudiant) :

- L'utilisateur connecté a les mêmes possibilités qu'un utilisateur anonyme

Gestion des connexions :

- L'utilisateur s'inscrit sur l'application en indiquant son nom, son prénom, un mot de passe ainsi que son adresse mail
- Le mot de passe doit avoir au minimum 8 caractères contenir un chiffre, une lettre majuscule et un caractère spécial
- Le mot de passe doit être crypté au sein de la base de donnée
- L'utilisateur ne peut s'inscrire avec une adresse mail déjà existante dans le système
- L'utilisateur se connecte en indiquant son email et son mot de passe
- L'utilisateur ne peut pas inscrire deux fois la même adresse mail
- L'utilisateur se déconnecte
- L'utilisateur peut changer ses informations

Gestion des chaussures :

- L'utilisateur peut visualiser tous ses chaussures qui sont en vente avec leurs statuts (vendu, disponible) et celles qu'il a acheté
- L'acheteur peut réserver une paire de chaussures auprès d'un vendeur
 - Une paire de chaussures ne peut pas être réservée plus d'une fois auprès d'un vendeur
 - Une paire de chaussures est considérée comme vendu lorsque son statut est « vendu » ☐ L'acheteur et le vendeur peuvent supprimer la réservation d'une paire de chaussures
 - Une réservation peut être supprimée à tout moment
 - Une réservation annulée rend la paire de chaussures de nouveau disponible à la vente
 - L'acheteur peut supprimer uniquement une réservation qu'il a initiée
 - Le vendeur peut annuler les réservations de ses chaussures

3.3.3 Description des actions réalisables en tant qu'administrateur :

- L'administrateur a les mêmes possibilités qu'un utilisateur connecté ☐
 - L'administrateur peut visualiser, modifier, supprimer un utilisateur
 - On peut bloquer un utilisateur
 - On peut débloquer un utilisateur
 - On ne peut pas supprimer un utilisateur qui a vendu des chaussures ☐ L'administrateur peut ajouter un utilisateur
 - Un utilisateur possède un nom, un prénom, un mot de passe ainsi que son adresse mail
 - Le mot de passe doit avoir au minimum 8 caractères contenir un chiffre, une lettre majuscule et un caractère spécial
 - Le mot de passe doit être crypté au sein de la base de donnée
 - L'adresse mail ne doit pas être déjà présente dans le système
 - L'administrateur peut supprimer une paire de chaussure. On peut supprimer une paire de chaussure uniquement s'il n'a pas été réservée ou vendue.

Points spécifiques

A14 : On ne peut pas inscrire un étudiant avec une adresse mail déjà existante

A15 : On ne peut pas réserver une paire de chaussure qui est déjà réservé

A16 : Le mot de passe doit avoir au minimum 8 caractères avec un chiffre, une lettre majuscule et un caractère spécial

A17 : Un étudiant qui a un statut bloqué n'apparaît plus en tant que vendeur

A18 : Une paire de chaussure qui n'a plus le statut réservé est de nouveau disponible à la vente

A19 : Lorsqu'on filtre des chaussures par marque, uniquement les modèles de la marque apparaissent

A20 : On ne peut pas supprimer un étudiant qui a effectué des ventes

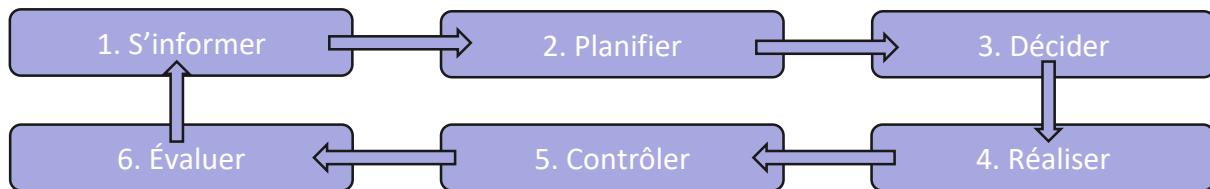
Livrables

- Planning prévisionnel fourni le premier jour du TPI
- Manuel technique (y. compris code)
- Manuel utilisateur
- Journal de travail
- Résumé du TPI

Organisation

Méthodologie

Pour ce projet individuel, j'ai décidé d'opter pour la méthode en 6 étapes. Je trouve cette méthode pertinente, car elle permet de découper le travail en 6 étapes distinctes et représentatives de la partie du travail à faire.



S'informer

J'ai commencé par soigneusement lire la grille d'évaluation qui est universelle à tous les TPI, cela me permet d'être attentif aux différents points importants dans le cadre de mon travail. J'ai également lu le cahier des charges du projet, afin de me faire une idée générale du projet à réaliser.

Planifier

Après avoir pris connaissance des limites qui encadrent mon travail, j'ai réalisé la planification en découplant les grandes tâches en sous-sections et en y attribuant un nombre d'heures que je jugeais pertinent par rapport à la difficulté de cette dernière.

Décider

Pendant la réalisation de ce travail, j'ai dû prendre différentes décisions afin de progresser tout en restant dans les limites d'heures que je me suis imposé dans la planification. Toutes les décisions auxquelles j'ai dû faire face sont inscrites dans mon journal de bord.

Réaliser

Toujours en respectant les délais imposés, j'ai continuellement œuvré à la réalisation des tâches que j'avais au préalable défini. La progression, les tests réalisés, l'avancée de la documentation, les problèmes rencontrés et les solutions trouvées sont notés dans mon journal de bord.

Contrôler

Afin de contrôler les fonctionnalités produites, j'ai réalisé un protocole de tests. Il m'a continuellement servi à contrôler les fonctionnalités déjà notées comme terminées, qu'elles soient fonctionnelles jusqu'au rendu du projet.

Évaluer

À la fin du projet, j'ai évalué l'état final de ce dernier. Cette étape sert à recenser si le projet est complet ou non, les difficultés, les solutions et les appréciations.

Product Backlog

Voici le Product Backlog du projet. Il comporte 7 points techniques évalués spécifiques au projet.

NOM	Inscription
USER STORY	En tant qu'utilisateur, je peux créer un compte afin de profiter pleinement des fonctionnalités du site
TEST A VALIDER	1
PRIORITÉ	Prioritaire

NOM	Connexion
USER STORY	En tant qu'utilisateur, je peux me connecter au compte que j'ai créé afin de continuer de profiter du site
TEST A VALIDER	2
PRIORITÉ	Prioritaire

NOM	Profil
USER STORY	En tant qu'utilisateur, je peux accéder à ma page de profil afin de modifier les informations de mon compte
TEST A VALIDER	5
PRIORITÉ	Prioritaire

NOM	Mettre en favoris
USER STORY	En tant qu'utilisateur, je peux mettre une chaussure qui me plaît en favoris afin de la retrouver plus facilement plus tard
TEST A VALIDER	3
PRIORITÉ	Prioritaire

NOM	Gestion
USER STORY	En tant qu'utilisateur, je peux accéder à ma page de gestion afin de voir et de gérer les chaussures que j'ai achetées et que j'ai à vendre
TEST A VALIDER	6
PRIORITÉ	Prioritaire

NOM	Mettre en vente
USER STORY	En tant qu'utilisateur, je peux mettre en vente une paire de chaussures depuis ma page de gestion afin que les autres utilisateurs puissent la réserver
TEST A VALIDER	4
PRIORITÉ	Prioritaire

NOM	Page admin
USER STORY	En tant qu'administrateur, je peux accéder à ma page admin afin de gérer les utilisateurs, les points de rencontre et les établissements
TEST A VALIDER	De 7 à 9
PRIORITÉ	Prioritaire

NOM	Gestion des utilisateurs
USER STORY	En tant qu'administrateur, je peux gérer les différents utilisateurs ainsi que les paires qu'ils ont achetées et vendues afin de garantir le bon fonctionnement du site
TEST A VALIDER	10
PRIORITÉ	Prioritaire

NOM	Gestion des points de rencontres
USER STORY	En tant qu'administrateur, je peux gérer les différents points de rencontres
TEST A VALIDER	8
PRIORITÉ	Prioritaire

NOM	Gestion des établissements
USER STORY	En tant qu'administrateur, je peux gérer les différents établissements
TEST A VALIDER	9
PRIORITÉ	Prioritaire

NOM	Barre de recherche
USER STORY	En tant qu'utilisateur, je peux rechercher une paire que je souhaite afin de la trouver plus facilement
TEST A VALIDER	11
PRIORITÉ	Prioritaire

NOM	Email déjà existant (A14)
USER STORY	En tant qu'utilisateur, je ne peux pas créer un compte avec un email déjà utilisé
TEST A VALIDER	12
PRIORITÉ	Prioritaire

NOM	Paire déjà réservée (A15)
USER STORY	En tant qu'utilisateur, je ne peux pas réserver une paire ayant déjà été réservée
TEST A VALIDER	13
PRIORITÉ	Prioritaire

NOM Sécurité du mot passe (A16)

USER STORY	En tant qu'utilisateur, mon mot de passe doit contenir au moins 8 caractères, un chiffre, une lettre majuscule et un caractère spécial afin de pouvoir créer mon compte
TEST A VALIDER	14
PRIORITÉ	Prioritaire

NOM	Visibilité du vendeur (A17)
USER STORY	En tant qu'utilisateur, je ne peux plus vendre de paires si mon statut est bloqué afin de préserver la sécurité des acheteurs
TEST A VALIDER	15
PRIORITÉ	Prioritaire

NOM	Disponibilité de la paire (A18)
USER STORY	En tant qu'utilisateur, je peux de nouveau réserver une paire qui n'a plus le statut réservé
TEST A VALIDER	16
PRIORITÉ	Prioritaire

NOM	Filtrage des marques (A19)
USER STORY	En tant qu'utilisateur, je peux filtrer les paires par marque afin de mieux choisir la paire que je souhaite
TEST A VALIDER	17
PRIORITÉ	Prioritaire

NOM	Suppression d'un vendeur (A20)
USER STORY	En tant qu'administrateur, je ne peux pas supprimer un utilisateur qui a déjà effectué des ventes afin de préserver une trace des actions
TEST A VALIDER	18
PRIORITÉ	Prioritaire

Analyse fonctionnelle

Use Case

Voici le use case prévu pour SneakersMarket. Les possibilités de l'utilisateur sont découpées en 3 :

Un utilisateur anonyme ne peut que voir la liste des paires en ventes et créer un compte ou se connecter.

Un utilisateur connecté a accès aux mêmes fonctionnalités qu'un utilisateur anonyme, mais il a également accès à toute la partie gestion, liste des favoris et réservation.

Un administrateur a accès aux mêmes fonctionnalités qu'un utilisateur connecté, et il a également accès à la partie admin.

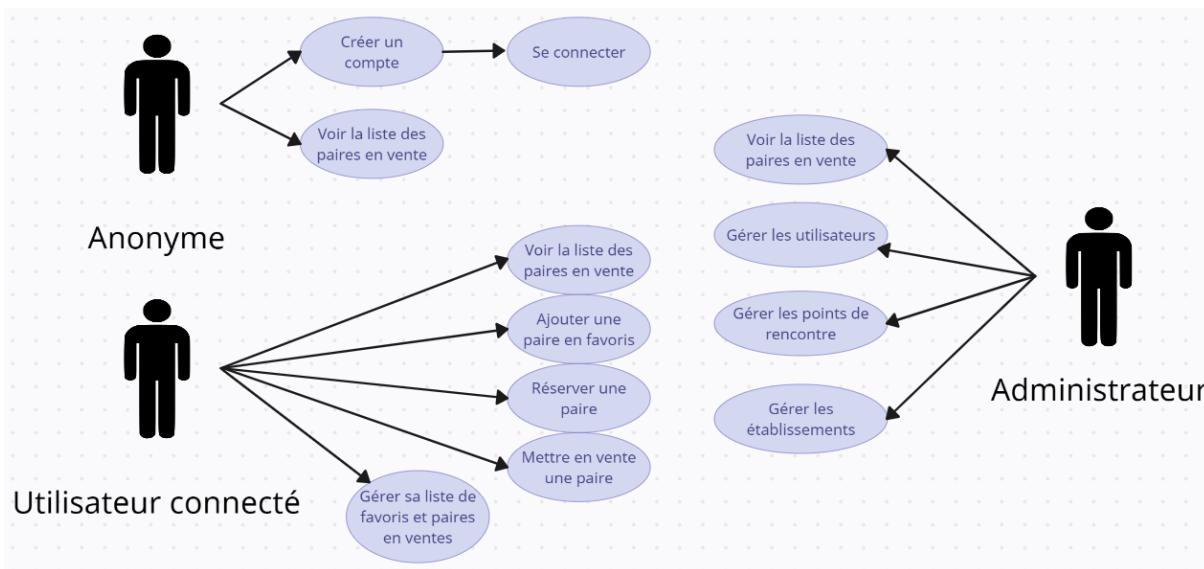


Figure 1 : Use case de SneakersMarket

Plan de navigation

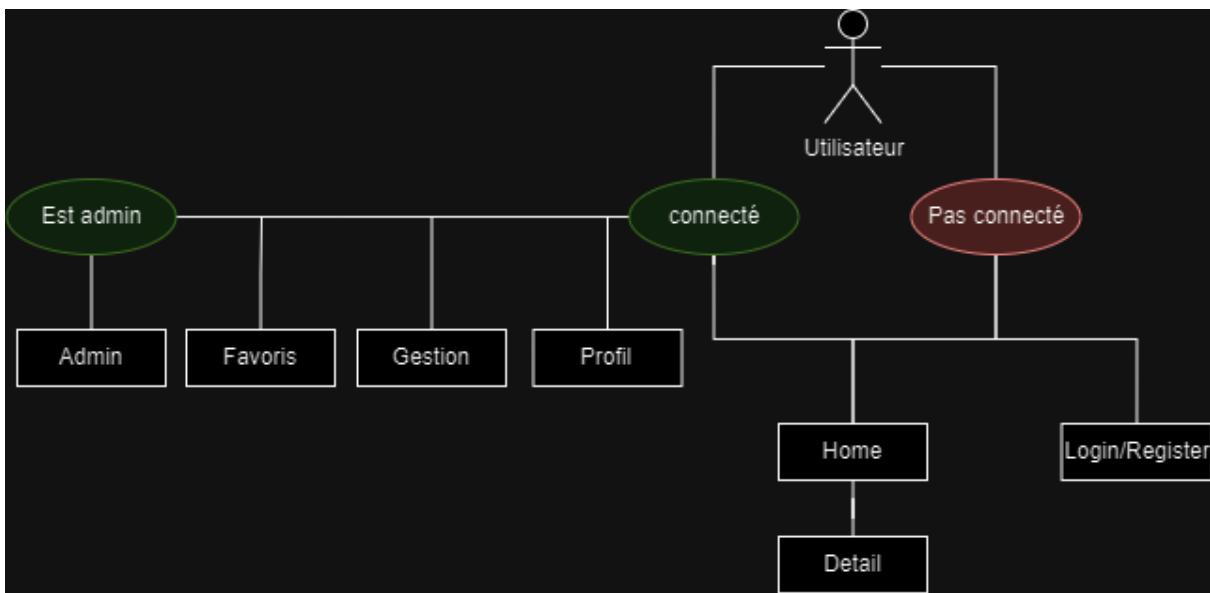


Figure 2 : Plan de navigation de SneakersMarket

Description des fonctionnalités majeures

Consulter la liste des Sneakers en vente

Critères du cahier des charges :

- L'utilisateur visualise depuis la page d'accueil les chaussures disponibles :
- Une paire de chaussures est uniquement visible si son statut est "visible"

La page d'accueil contient la liste de toutes les paires en vente :

- Qui n'ont pas été réservé
- Ou l'utilisateur n'est pas défini comme « bloqué »
- Ou la paire est définie comme « Visible »
- Ou l'utilisateur ayant posté la paire n'est pas l'utilisateur étant connecté

Les paires sont affichées avec comme informations l'image, la marque, le modèle, le prix, la taille, la description et l'utilisateur ayant posté la paire.

Filtrer les paires par marques

Critère du cahier des charges :

- L'utilisateur peut filtrer les chaussures par marque

L'utilisateur peut sélectionner une marque spécifique pour afficher uniquement les paires de cette dernière. Si l'utilisateur ne sélectionne pas de marque, toutes les paires sont affichées.

Pagination

Critère du cahier des charges :

- Seulement 10 paires de chaussures sont visibles, les autres sont accessibles via une pagination

En dessous de la liste des paires de chaussures, l'utilisateur dispose d'une ligne avec des numéros qui représente le nombre de pages disponibles. S'il clique sur un numéro, les paires affichées correspondront à la page sélectionnée. Pour déterminer quelles paires prendre de la base, un LIMIT a été ajouté à toutes les requêtes.

Barre de recherche

Critère du cahier des charges :

- L'utilisateur peut chercher une paire de chaussures en entrant son nom

Dans le header, une barre de recherche est disponible. Si l'utilisateur recherche quelque chose, il est renvoyé sur la page « Search Result » qui affiche la liste des paires recherchées. Si aucune paire ne correspond à la recherche, un message est affiché signalant qu'aucune paire n'a été trouvée.

Réserver une paire

Critère du cahier des charges :

- L'acheteur peut réserver une paire de chaussures auprès d'un vendeur

Dans la page détail, si l'utilisateur est connecté, il peut réserver une paire n'ayant pas déjà été réservée en choisissant un point de rendez-vous. Il peut annuler à tout moment sa réservation dans sa page détail. L'acheteur, de son côté, dans sa page de gestion, reçoit la demande d'achat, et peut valider ou rejeter la demande.

Mettre une paire en favoris

Critère du cahier des charges :

- Liste des favoris : Les chaussures que l'étudiant a marquées comme favoris.

Dans la page détail, l'utilisateur peut mettre en favoris la paire. Celle-ci apparaît ensuite dans la page Favoris. L'utilisateur a la possibilité d'enlever la paire des favoris. L'ajout en favoris n'influence pas la visibilité de la paire, plusieurs personnes peuvent avoir la même paire en favoris. En revanche, quand une paire est validée à l'achat, la paire est enlevée des favoris de tous les utilisateurs.

Profil

Critère du cahier des charges :

- L'utilisateur peut changer ses informations

L'utilisateur connecté dispose d'une page profil, accessible en cliquant sur son nom sur le header. Sur cette page, il y figure un formulaire contenant les informations de l'utilisateur. L'utilisateur a la possibilité de changer ses informations. Si l'utilisateur soumet le formulaire, les changements se font en base et dans la session.

Gérer les utilisateurs

Critères du cahier des charges :

- L'administrateur peut visualiser, modifier, supprimer un utilisateur
- On peut bloquer un utilisateur
- On peut débloquer un utilisateur
- On ne peut pas supprimer un utilisateur qui a vendu des chaussures

Dans sa page Admin, une liste contenant tous les utilisateurs est affichée. Pour chaque utilisateur, l'administrateur à la possibilité de :

- Supprimer l'utilisateur s'il n'a concrétisé aucune vente
- Bloquer l'utilisateur s'il a déjà concrétisé une ou plusieurs ventes
- Modifier les informations de l'utilisateur
- Voir les paires que l'utilisateur a en vente et a vendues

Quand l'administrateur clique sur « Modifier », il est redirigé sur la page « Modify User », similaire à la page profil mais contenant les informations de l'utilisateur voulant être modifié. Une fois le formulaire soumis, les modifications se font dans la base de données.

Maquettes

Pour l'affichage des paires de chaussures, j'ai décidé de partir sur ce design. Les cartes permettent de mieux séparer les différentes paires, ainsi que de mettre en avant les informations importantes, comme par exemple le nom ou le prix de la paire de chaussures.

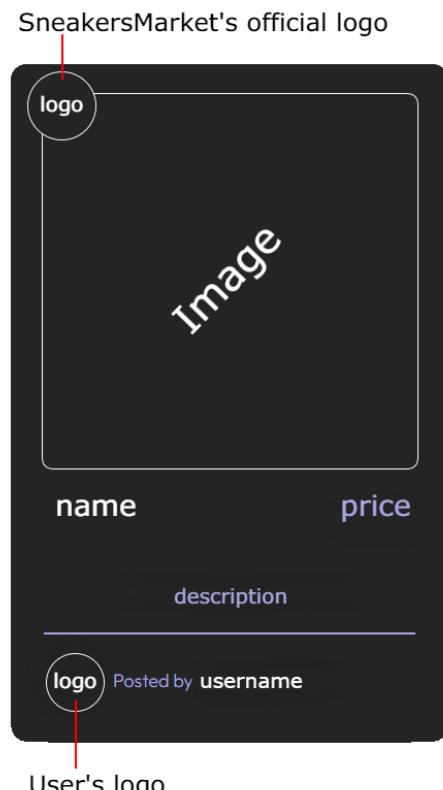


Figure 3 : Maquettes des cards

Pour le design de la page détail, j'ai adoré la disposition d'une template trouvée (voir bibliographie), j'ai donc repris la disposition des éléments et adapté pour y ajouter les informations supplémentaires, comme la sélection de la taille.

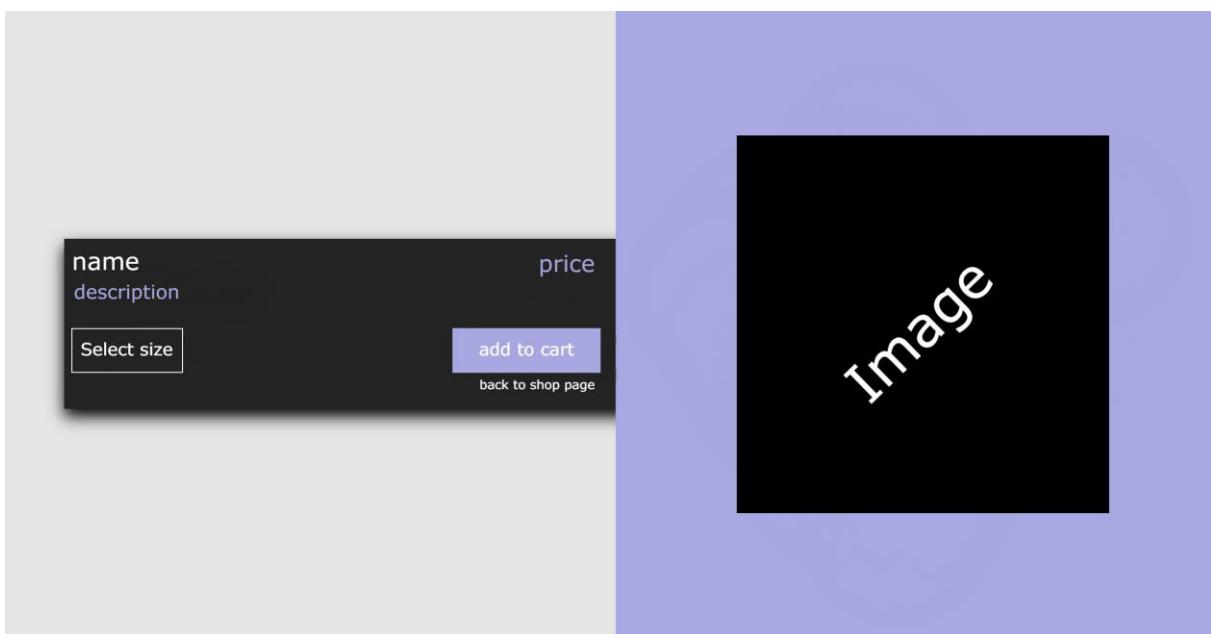


Figure 4 : Maquettes de la page détail

En ce qui concerne mon login et register, j'ai décidé de reprendre le design de ma page détail et de l'adapter en y ajoutant les formulaires nécessaires. Etant donné que la page telle quelle faisait vide, j'ai décidé qu'au lieu d'avoir une page login et une page register distinctes, je combinerai les deux. Ainsi, si on clique sur le lien « register », la partie violette se déplacera vers la droite pour cacher le formulaire de login et faire apparaître celui de register.

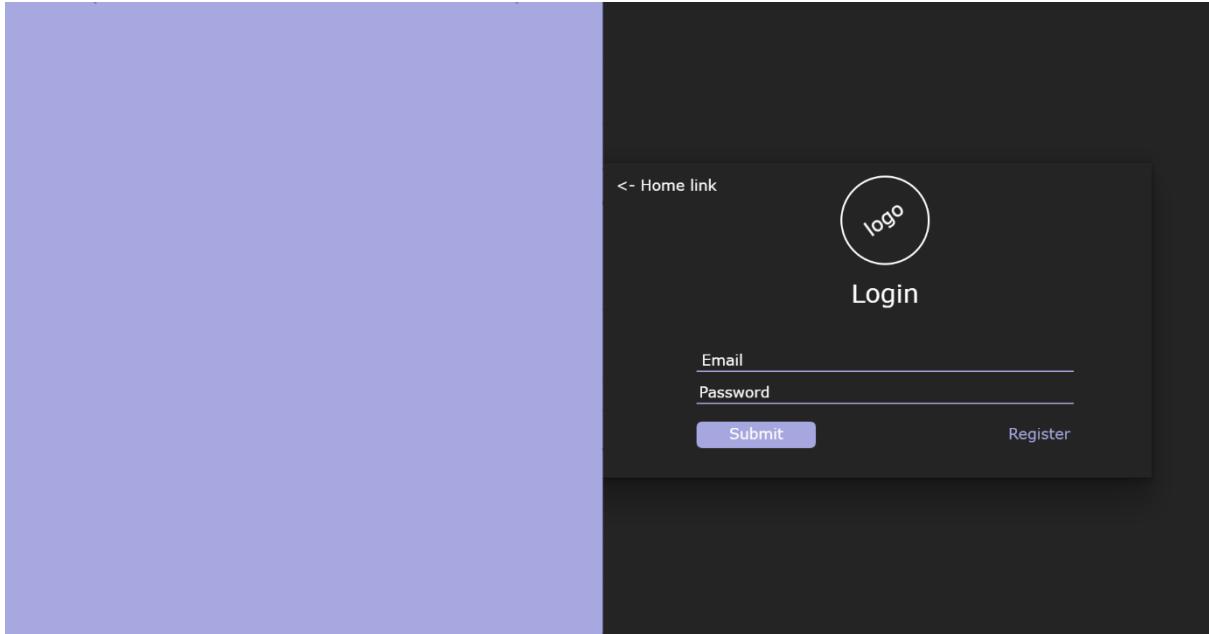


Figure 5 : Maquettes de la page auth (partie login)

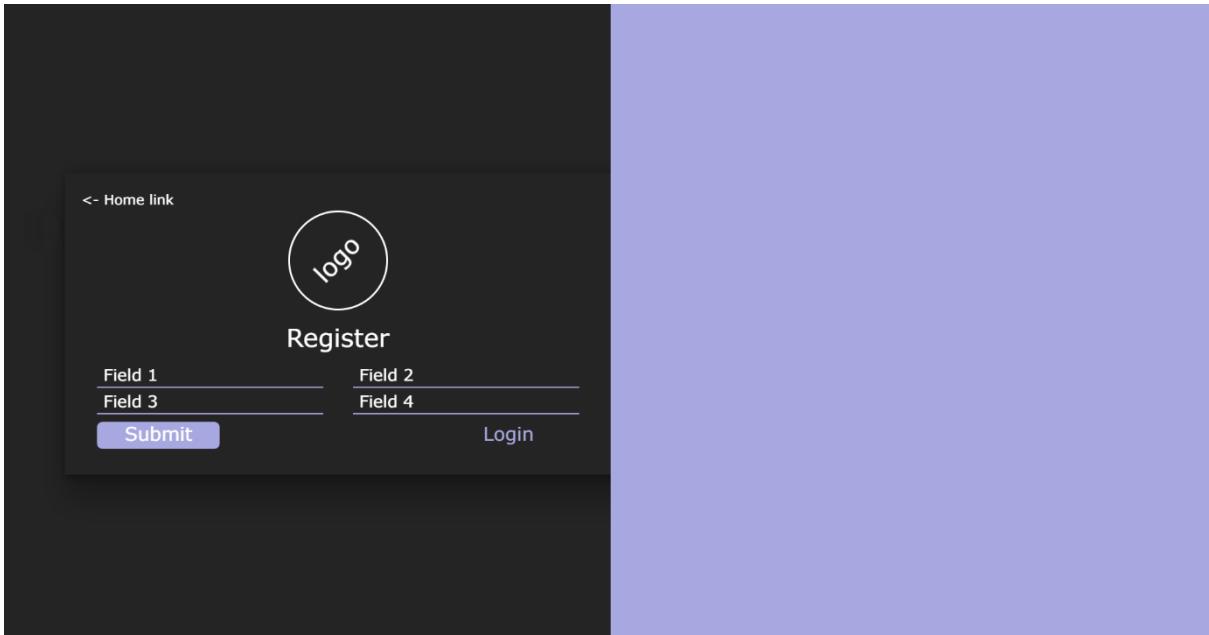


Figure 6 : Maquettes de la page auth (partie register)

Pour la page home, il n'y aura qu'un simple listing des paires de chaussures. Il y aura également un select tout en haut pour filtrer par marques comme ceci :

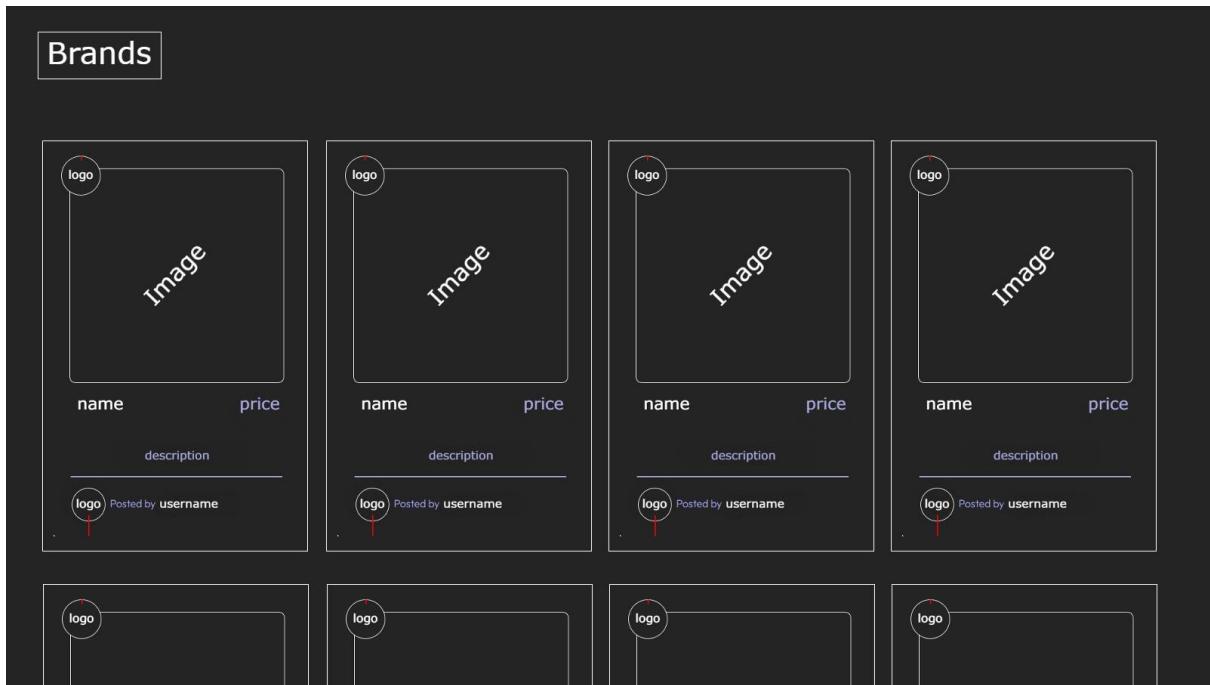


Figure 7 : Maquette de la page home

La page profile sera similaire à celle de connexion/inscription, mais en plus de cela, je rajouterai une preview pour que l'utilisateur voie l'aperçu de son profil.

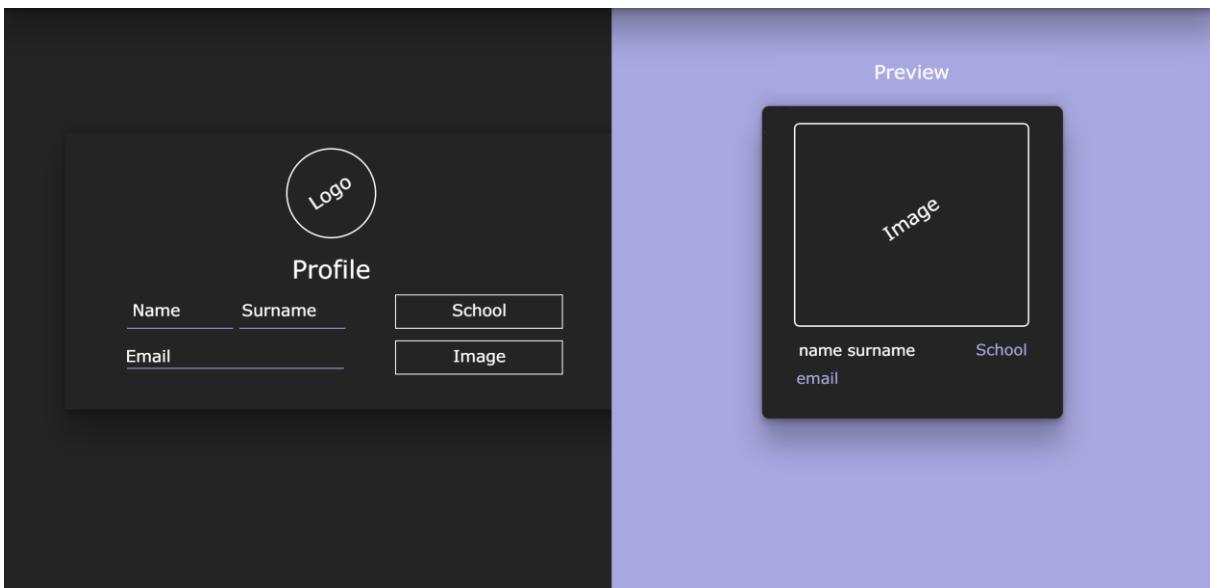


Figure 8: Maquette de la page profil

Ma page à propos se découperait en deux parties. La première étant simplement le nom du projet pour faire page de présentation, et la seconde, accessible en scrollant, contient la description du projet ainsi que l'auteur du projet. Cette page n'étant pas demandée dans le cahier des charges, je ne la ferai que si le temps me le permet. Cette page est utile pour faciliter la tâche des personnes ne connaissant pas le site, qu'ils puissent directement lire de quoi ce dernier traite.



Figure 9 : Maquettes de la page à propos (partie 1)

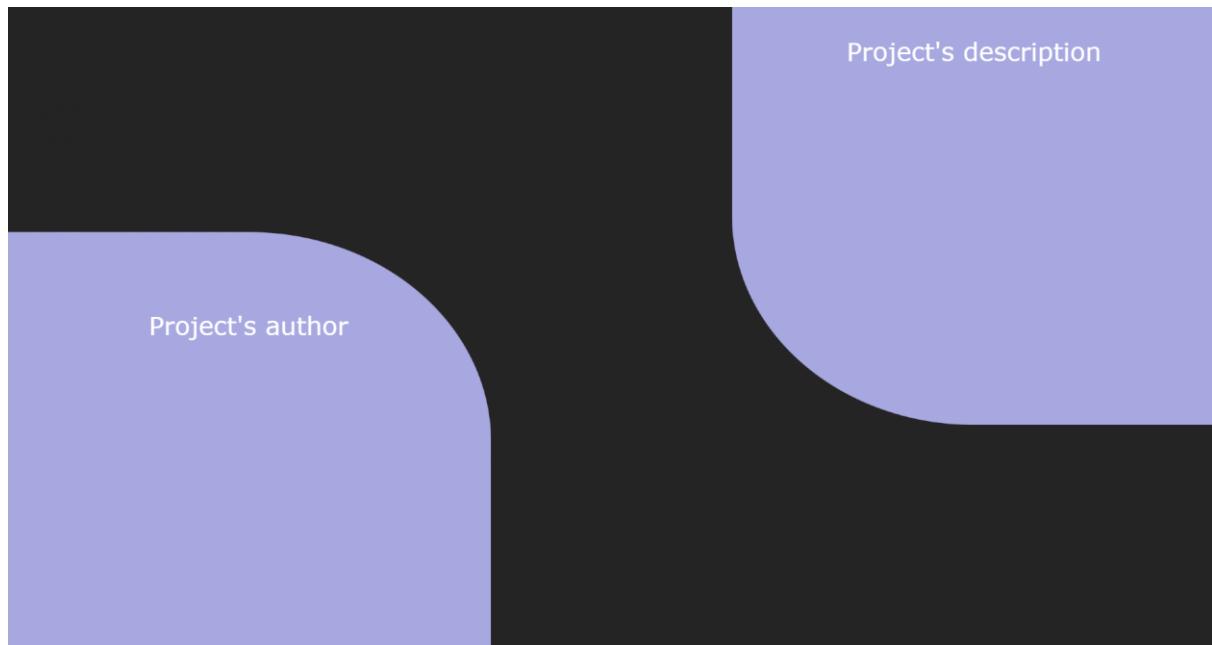


Figure 10 : Maquettes de la page à propos (partie 2)

Analyse organique

Modèle conceptuel de données

Voici le modèle conceptuel de données (MCD) de SneakersMarket. Elle a été réalisée en 3^{ème} forme normale et comporte 14 tables, dont 2 de liaisons.

Ce dernier a beaucoup évolué au fil du temps. La première version était en français, mais afin de garder une uniformité de la langue entre le code et la base de données, je l'ai passé en anglais.

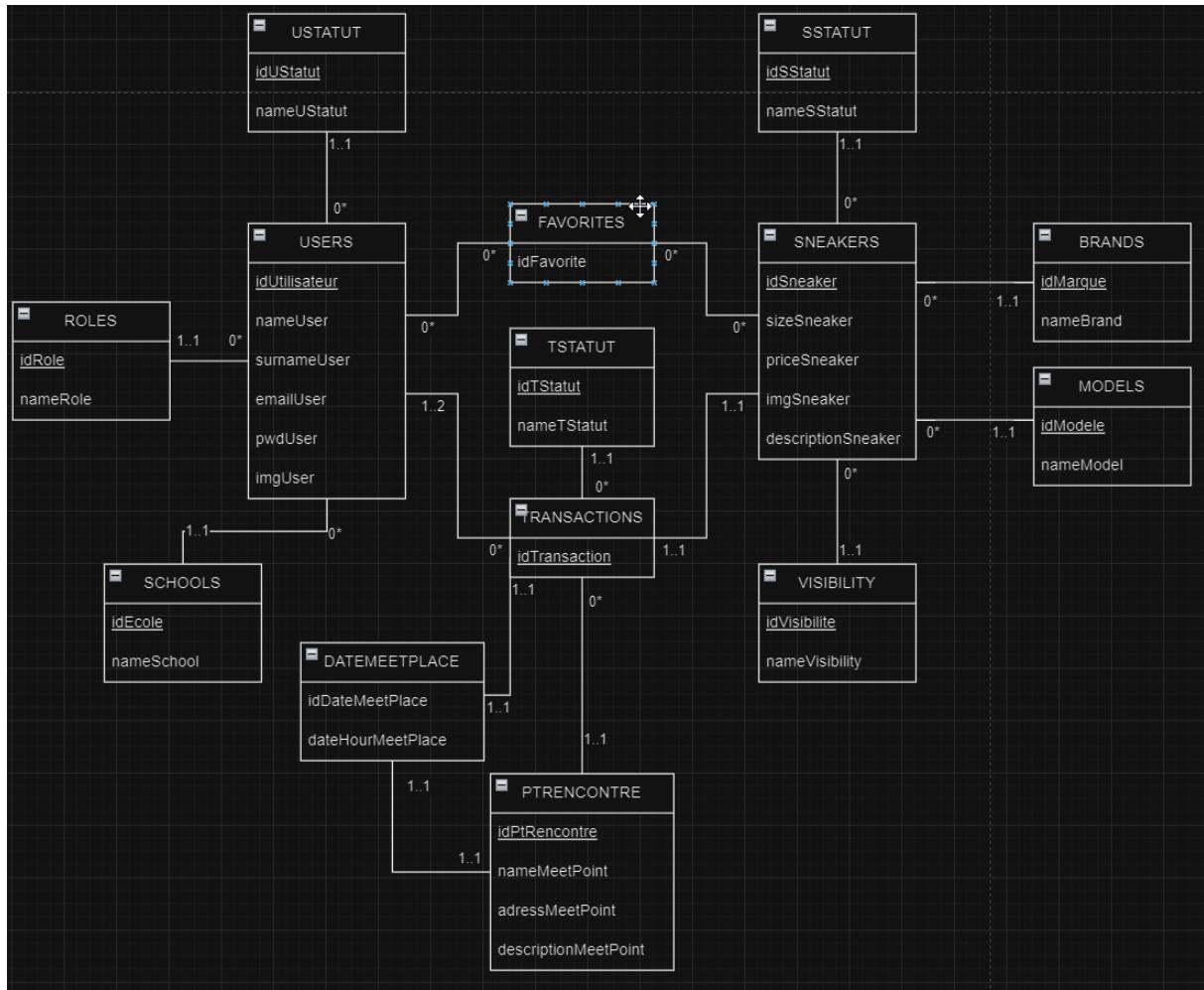


Figure 11 : Modèle conceptuel de données de SneakersMarket

Modèle logique de données

↳

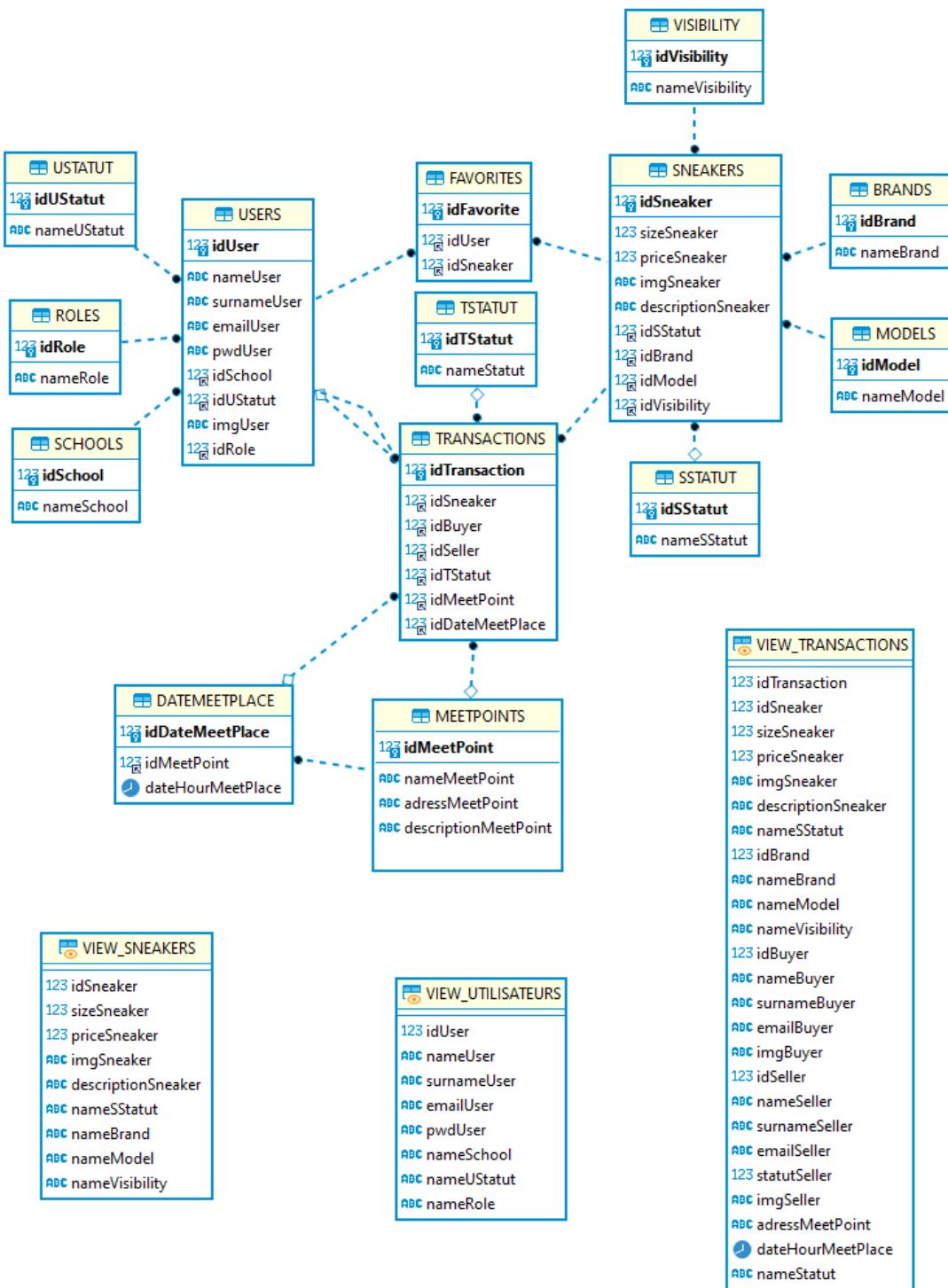


Figure 12 : Modèle logique de données de SneakersMarket

Dictionnaire de données

USERS

Table stockant les utilisateurs qui s'inscrivent sur le site

Nom	Type	NULL ?	Valeur par défaut	Extra
idUser	Int	Non	/	PRIMARY AUTO INCREMENT
nameUser	Varchar	Non	/	/
surnameUser	Varchar	Non	/	/
emailUser	Varchar	Non	/	/
pwdUser	Varchar	Non	/	/
idSchool	Int	Non	/	FOREIGN KEY
IdUStatut	Int	Non	1	FOREIGN KEY

SNEAKERS

Table stockant les paires de chaussures du site

Nom	Type	NULL ?	Valeur par défaut	Extra
idSneaker	Int	Non	/	PRIMARY AUTO INCREMENT
sizeSneaker	Int	Non	/	/
priceSneaker	Double	Non	0.0	/
imgSneaker	Longtext	Non	« notFound »	/
descriptionSneaker	Varchar	Non	/	/
idSStatut	Int	Non	/	FOREIGN KEY
idBrand	Int	Non	/	FOREIGN KEY
idModel	Int	Non	/	FOREIGN KEY
idVisibility	Int	Non	/	FOREIGN KEY

MEETPOINT

Table stockant les points de rencontres

Nom	Type	NULL ?	Valeur par défaut	Extra
IdMeetPoint	Int	Non	/	PRIMARY AUTO INCREMENT
nameMeetPoint	Varchar	Non	/	/
adressMeetPoint	Varchar	Non	/	/
descriptionMeetPoint	Varchar	Non	/	/

TRANSACTIONS

Table associative stockant les transactions entre utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
idTransaction	Int	Non	/	PRIMARY AUTO INCREMENT
idSneaker	Int	Non	/	FOREIGN KEY
idBuyer	Int	Oui	NULL	FOREIGN KEY
idSeller	Int	Non	/	FOREIGN KEY
idTStatut	Int	Non	/	FOREIGN KEY
idMeetPoint	Int	Non	/	FOREIGN KEY
idDateMeetPlace	Int	Oui	NULL	FOREIGN KEY

FAVORITE

Table associative les sneakers favoris des utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
IdFavorite	Int	Non	/	PRIMARY AUTO INCREMENT
IdUser	Int	Non	/	FOREIGN KEY
idSneaker	Int	Non	/	FOREIGN KEY

BRANDS

Table stockant les marques des Sneakers

Nom	Type	NULL ?	Valeur par défaut	Extra
IdBrand	Int	Non	/	PRIMARY AUTO INCREMENT
nameBrand	Varchar	Non	/	/

SSTATUT

Table stockant les statuts des sneakers

Nom	Type	NULL ?	Valeur par défaut	Extra
IdSStatut	Int	Non	/	PRIMARY AUTO INCREMENT
nameSStatut	Varchar	Non	/	/

TSTATUT

Table stockant les statuts des transactions

Nom	Type	NULL ?	Valeur par défaut	Extra
IdTStatut	Int	Non	/	PRIMARY AUTO INCREMENT
nameTStatut	Varchar	Non	/	/

USTATUT

Table stockant les statuts des utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
IdUStatut	Int	Non	/	PRIMARY AUTO INCREMENT
nameUStatut	Varchar	Non	/	/

MODELS

Table stockant les modèles des sneakers

Nom	Type	NULL ?	Valeur par défaut	Extra
IdModel	Int	Non	/	PRIMARY AUTO INCREMENT
nameModel	Varchar	Non	/	/

ECOLE

Table stockant les écoles des utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
idEcole	Int	Non	/	PRIMARY AUTO INCREMENT
nomEcole	Varchar	Non	/	/

VISIBILITY

Table stockant la visibilité des utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
idVisibility	Int	Non	/	PRIMARY AUTO INCREMENT
nameVisibility	Varchar	Non	/	/

ROLES

Table stockant le rôle des utilisateurs

Nom	Type	NULL ?	Valeur par défaut	Extra
idRole	Int	Non	/	PRIMARY AUTO INCREMENT
nameRole	Varchar	Non	/	/

DATETIMEETPLACE

Table stockant la date pour chaque point de rendez-vous

Nom	Type	NULL ?	Valeur par défaut	Extra
idDateMeetPlace	Int	Non	/	PRIMARY AUTO INCREMENT
idMeetPoint	Int	Non	/	FOREIGN KEY
dateHourMeetPlace	Datetime	Non	/	/

Structure du projet

J'ai décidé d'adopter la structure MVC pour l'ensemble de mon projet.

Modèle

La partie modèle contient la connexion à la base en PDO, les objets utilisés ainsi que toutes les requêtes, que ce soit insertion, modification, suppression et sélection. Chaque page possède sa classe avec les fonctions qui lui sont propres.

Vue

La partie vue concerne uniquement ce qui sera visible sur le site, c'est-à-dire toute la partie HTML-CSS. Aucune logique n'est faite dans ces pages. Les seuls moments où on y met du PHP, c'est pour afficher la valeur d'une variable, faire une boucle pour afficher certains éléments ou faire une condition pour afficher ou ne pas afficher certains éléments.

Contrôleur

Il y a un contrôleur par page. Le contrôleur sert à faire la liaison entre l'affichage et la logique, ainsi que pour récupérer les données passées en formulaires.

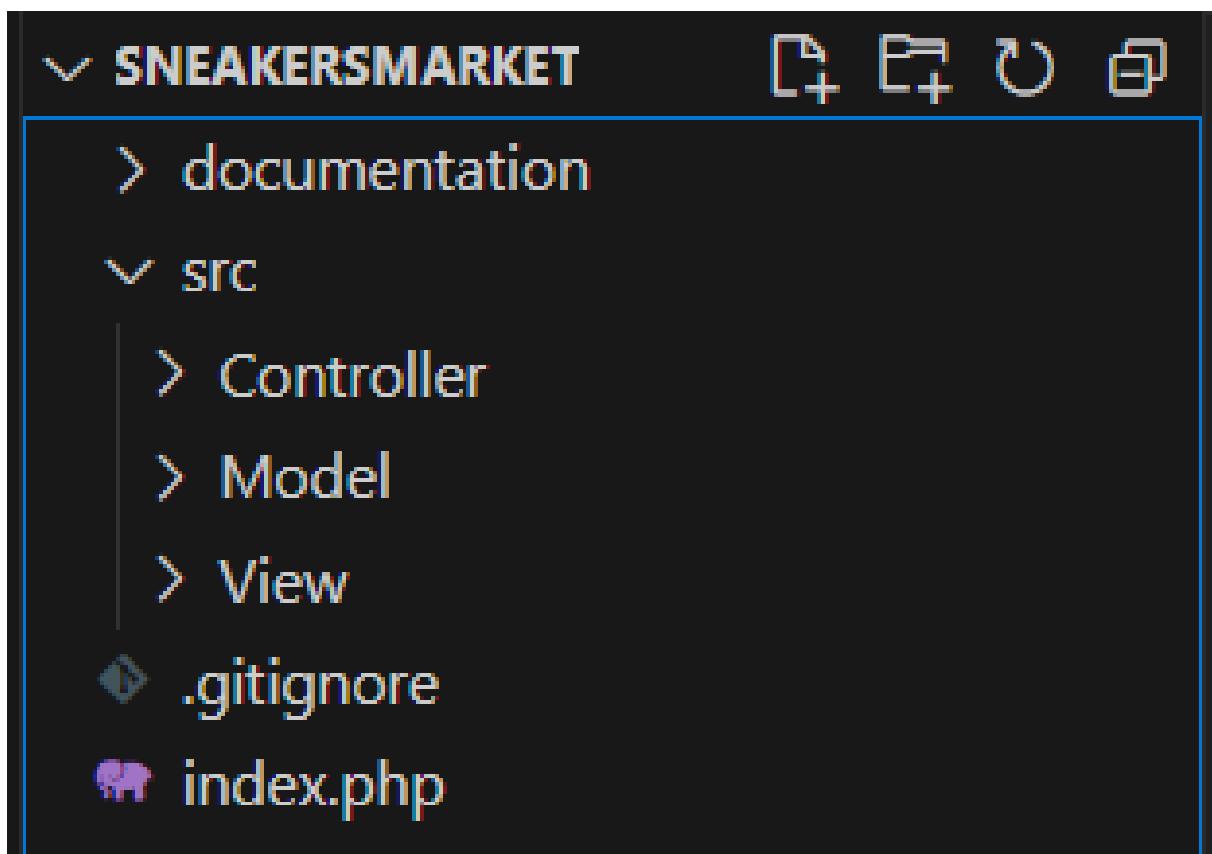


Figure 13 : Arborescence du projet

Description des méthodes et classes principales

Classe Home

Cette classe est dédiée à la page home. Elle gère le listing des paires ainsi que les différents filtres disponibles.

GetAllShoesCard

Cette méthode permet de récupérer les informations de toutes les paires afin de les afficher sous formes de cartes dans la page Home. Elle a comme paramètres :

- idSeller qui correspond à l'id de l'utilisateur ayant posté la paire. On n'affiche pas la paire si le vendeur est l'utilisateur connecté.
- statutSeller qui doit être égal à 1 pour que la paire soit sélectionnée. 1 correspond à « Actif » dans la base
- indexFirst qui correspond à l'index de la première paire sélectionnée pour la pagination
- indexLast qui correspond à l'index de la dernière paire sélectionnée pour la pagination.

FilterByBrand

Cette méthode sert à récupérer les Sneakers correspondant à la marque souhaitée. Elle a comme paramètres :

- idBrand qui correspond à l'id de la marque stockée dans la base
- indexFirst qui correspond à l'index de la première paire sélectionnée pour la pagination
- indexLast qui correspond à l'index de la dernière paire sélectionnée pour la pagination.

Classe Gestion

Cette méthode est dédiée à la page gestion. Elle gère l'ajout d'une paire, les réservations et les achats.

CreateTransaction

Cette méthode ajoute la paire que l'utilisateur et par extension la transaction qu'il souhaite vendre en base si toutes les informations sont correctes. Dans un premier temps, elle vérifie si le modèle entré existe dans la base, si c'est le cas, il récupère son id, sinon, elle l'insère en base et récupère son id. Ensuite, elle insère la paire en base et pour finir récupère l'id de cette dernière pour créer la transaction. Elle a comme paramètres :

- size qui correspond à la taille de la paire
- price qui correspond au prix de la paire
- img qui correspond à l'image de la paire
- nameModel qui correspond au modèle entré
- idBrand qui correspond à la marque sélectionnée

Classe Admin

Cette classe est dédiée à la page admin. la classe admin dispose de fonctions plutôt basiques pour gérer les différents CRUD des utilisateurs, paires, établissements et points de rencontre.

BlockUser

Cette méthode change le statut de l'utilisateur concerné. Il passe de « Active » à « Blocked ». Elle a comme paramètre l'id de l'utilisateur à changer.

Classe Profil

UpdateProfile

Cette méthode met à jour les données dans la base de données de l'utilisateur en fonction de ce qu'il a modifié sur la page profil. Elle commence par vérifier les données, puis change les données dans la base. Elle a comme paramètres :

- user qui correspond à un objet User contenant toutes les informations de l'utilisateur
- img qui correspond à la potentielle image passée, s'il l'image est vide, l'image en base ne change pas

Analyse des fonctionnalités majeures

Mettre une paire en vente

Pour un utilisateur souhaitant devenir vendeur, mettre en vente une paire est la fonctionnalité essentielle. Disponible dans la page gestion, pour pouvoir mettre en vente une paire, l'utilisateur doit renseigner une image, la marque, le modèle, le prix, la taille et la description de cette dernière. Une fois les informations correctes et validées, la paire est disponible à la vente pour tous les autres utilisateurs.

```
try {
    // Début de la transaction
    Database::beginTransaction();

    // Si l'insertion du modèle se fait correctement
    if(self::InsertModele($nameModel) !== false){
        // Récupère l'id du dernier modèle inséré
        $idModel = Database::lastInsertId();
    }
    else{
        // Sinon, récupère l'id du modèle
        $idModel = self::DoesModeleExist($nameModel)->idModel;
    }

    // Insère la sneaker en base
    self::InsertSneaker($size, $price, $img, $desc, $idModel, $idBrand);

    $idSneaker = Database::lastInsertId();

    // Insère la transaction
    $s = "INSERT INTO TRANSACTIONS(idSneaker, idSeller) VALUES(:idSneaker, :idSeller)";
    $statement = Database::prepare($s);

    $statement->bindParam(':idSneaker', $idSneaker);
    $statement->bindParam(':idSeller', $_SESSION['idUser']);

    $statement->execute();
    Database::commit();
} catch (PDOException $e) {
```

Figure 15 : Méthode pour mettre une paire en vente

Réserver une paire

Sur la page détail, un formulaire de réservation est disponible pour l'utilisateur. S'il y renseigne correctement le point de rendez-vous, la date et l'heure souhaitée, la réservation est transmise sur la page gestion du vendeur, et c'est à lui de décider s'il accepte ou non l'offre.

```
try {
    // Commence la transaction
    Database::beginTransaction();

    // Insère la date de rendez-vous
    self::InsertDateMeetPoint($idMeetPoint, $meetingDate);
    $idDateMeetPlace = Database::lastInsertId();

    // Met à jour le point de rendez-vous et le statut
    $s = "UPDATE TRANSACTIONS SET idBuyer = :idUser, idMeetPoint = :idMeetPoint, idStatut = 3, idDateMeetPlace = :idDateMeetPlace";
    $statement = Database::prepare($s);
    [
    $statement->bindParam(':idSneaker', $idSneaker);
    $statement->bindParam(':idMeetPoint', $idMeetPoint);
    $statement->bindParam(':idDateMeetPlace', $idDateMeetPlace);
    $statement->bindParam(':idUser', $_SESSION['idUser']);

    $statement->execute();

    Database::commit();
} catch (PDOException $e) {
```

Figure 14 : Méthode pour réserver une paire

Ajouter une paire en favoris

Également sur la page détail, une petite étoile figure à côté du bouton réserver. Si l'utilisateur clique dessus, la paire est ajoutée à sa liste de favoris. Si l'utilisateur a déjà la paire en favoris et clique à nouveau sur l'étoile, la paire est enlevée de la liste des favoris.

```
try {
    $s = "INSERT INTO FAVORITES(idUser, idSneaker) VALUES(:idUser, :idSneaker)";
    $statement = Database::prepare($s);

    $statement->bindParam(':idSneaker', $idSneaker);
    $statement->bindParam(':idUser', $idUser);

    $statement->execute();
```

Figure 16 : Méthode pour mettre une paire en favoris

```
try {
    $s = "DELETE FROM FAVORITES WHERE idSneaker = :idSneaker AND idUser = :idUser";
    $statement = Database::prepare($s);

    $statement->bindParam(':idSneaker', $idSneaker);
    $statement->bindParam(':idUser', $idUser);

    $statement->execute();
```

Figure 17 : Méthode pour enlever une paire des favoris

Programmes utilisés

Visual Studio Code – 1.88

Visual Studio Code est un éditeur de code développé par Microsoft. J'ai utilisé ce dernier pour écrire et corriger tout le code présent dans mon travail.



DBeaver – 23.1.5

DBeaver est un logiciel qui m'a servi à gérer la base de données du projet, que ce soit l'insertion des données, la création des tables ou les requêtes.



Gitlab – 16.10.3

Gitlab est un logiciel intégré à Git. Il m'a servi pour le suivi des versions de mon projet.



Copilot – Free

Copilot est un chatbot développé par Microsoft, qui permet, à l'image de ChatGPT, de converser et créer des images générées artificiellement. Je l'ai utilisé pour la création de toutes mes images sur le site.



Pixlr – Free

Pixlr est un outil de retouche photo possédant une version gratuite se limitant à 3 retouches par jour. Il m'a servi à modifier mon logo comme je le souhaitais.



Remove.bg – Free

Remove.bg est également un outil de retouche photo, mais permettant d'automatiquement retirer le fond des images, permettant ainsi de les rendre transparentes. Je l'ai utilisé afin de créer une deuxième version aux paires de chaussures, que j'affiche ensuite dans ma page détail.



Codepen - Free

Codepen est un site internet qui regroupe les projets et exemples programmés de personnes du monde entier. Il m'a été utile pour me donner des idées de design ou pour m'aider à comprendre comment réaliser la disposition des éléments que je souhaitais.



Canva – Free

Canva est un site internet de design. Je l'ai utilisé pour créer les différents designs des pages de couvertures présentes sur ce document et le manuel utilisateur.



Tests

Plan de tests

Utilisateur Anonyme : A accès à la page d'accueil et la page détail, mais ne peut pas réserver.

Utilisateur Connecté : A accès à toutes les fonctionnalités sauf celles présentes sur Admin.

Administrateur : A accès à la page admin en plus.

ID TEST	1
USER STORY	En tant qu'utilisateur, je peux créer une compte afin de profiter pleinement des fonctionnalités du site
SITUATION	L'utilisateur entre les informations correctement, et le compte se crée
RESULTAT OBTENU	L'inscription se fait correctement, il y a les bonnes données en session et l'utilisateur est redirigé sur home
STATUT	OK

ID TEST	2
USER STORY	En tant qu'utilisateur, je peux me connecter au compte que j'ai créé afin de continuer de profiter du site
SITUATION	L'utilisateur entre les informations correctement, et la connexion est autorisée
RESULTAT OBTENU	La connexion se fait correctement, les données en session sont correctes
STATUT	OK

ID TEST	3
USER STORY	En tant qu'utilisateur, je peux me connecter au compte que j'ai créé afin de continuer de profiter du site
SITUATION	L'utilisateur entre les informations correctement, et la connexion est autorisée
RESULTAT OBTENU	La connexion se fait correctement, les données en session sont correctes
STATUT	OK

ID TEST	4
USER STORY	En tant qu'utilisateur, je peux mettre en vente une paire de chaussures depuis ma page de gestion afin que les autres utilisateurs puissent la réserver
SITUATION	L'utilisateur rentre les bonnes informations et la paire apparaît dans la page home chez les autres utilisateurs
RESULTAT OBTENU	La paire est correctement ajoutée en base, la paire apparaît dans la page home des autres utilisateurs
STATUT	OK

ID TEST	5
USER STORY	En tant qu'utilisateur, je peux accéder à ma page de profil afin de modifier les informations de mon compte
SITUATION	L'utilisateur change les informations de son compte, les informations en base sont changées
RESULTAT OBTENU	Les informations sont correctement changées
STATUT	OK

ID TEST	6
USER STORY	En tant qu'utilisateur, je peux accéder à ma page de gestion afin de voir et de gérer les chaussures que j'ai achetées et que j'ai à vendre
SITUATION	L'utilisateur voit la liste des paires achetées et à vendre
RESULTAT OBTENU	Les listes sont complètes et correctes
STATUT	OK

ID TEST	7
USER STORY	En tant qu'administrateur, je peux accéder à ma page admin afin de gérer les utilisateurs, les points de rencontre et les établissements
SITUATION	L'administrateur peut bloquer et débloquer un utilisateur
RESULTAT OBTENU	Le blocage s'effectue correctement et l'utilisateur n'apparait plus en tant que vendeur
STATUT	OK

ID TEST	8
USER STORY	En tant qu'administrateur, je peux accéder à ma page admin afin de gérer les utilisateurs, les points de rencontre et les établissements
SITUATION	L'administrateur peut modifier et ajouter un point de rencontre
RESULTAT OBTENU	La modification / L'ajout s'effectue correctement
STATUT	OK

ID TEST	9
USER STORY	En tant qu'administrateur, je peux accéder à ma page admin afin de gérer les utilisateurs, les points de rencontre et les établissements
SITUATION	L'administrateur peut modifier et ajouter un établissement
RESULTAT OBTENU	La modification / L'ajout s'effectue correctement
STATUT	OK

ID TEST	10
USER STORY	En tant qu'administrateur, je peux gérer les différents utilisateurs ainsi que les paires qu'ils ont achetées et vendues afin de garantir le bon fonctionnement du site
SITUATION	L'administrateur peut modifier les paires à vendre d'un utilisateur
RESULTAT OBTENU	La modification s'effectue correctement
STATUT	OK

ID TEST	11
USER STORY	En tant qu'utilisateur, je peux rechercher une paire que je souhaite afin de la trouver plus facilement
SITUATION	L'utilisateur cherche la paire grâce à la barre de recherche
RESULTAT OBTENU	La/Les paire(s) cherchée(s) est/sont trouvée(s) et uniquement elle(s) est/sont affichée(s)
STATUT	OK

ID TEST	12
USER STORY	En tant qu'utilisateur, je ne peux pas créer un compte avec un email déjà utilisé
SITUATION	L'utilisateur tente de créer un compte avec un email déjà existant
RESULTAT OBTENU	L'inscription ne se fait pas et un message d'erreur apparaît
STATUT	OK

ID TEST	13
USER STORY	En tant qu'utilisateur, je ne peux pas réserver une paire ayant déjà été réservée
SITUATION	L'utilisateur recherche une paire déjà réservée par une autre personne
RESULTAT OBTENU	La paire n'apparaît pas
STATUT	OK

ID TEST	14
USER STORY	En tant qu'utilisateur, mon mot de passe doit contenir au moins 8 caractères, un chiffre, une lettre majuscule et un caractère spécial afin de pouvoir créer mon compte
SITUATION	L'utilisateur tente de rentrer un mot de passe qui ne correspond pas à la sécurité demandée
RESULTAT OBTENU	L'inscription ne se fait pas et un message d'erreur apparaît
STATUT	OK

ID TEST	15
USER STORY	En tant qu'utilisateur, je ne peux plus vendre de paires si mon statut est bloqué afin de préserver la sécurité des acheteurs
SITUATION	L'utilisateur tente de vendre une paire en étant bloqué
RESULTAT OBTENU	La catégorie n'apparaît pas dans la page de gestion
STATUT	OK

ID TEST	16
USER STORY	En tant qu'utilisateur, je peux de nouveau réserver une paire qui n'a plus le statut réservé
SITUATION	L'utilisateur ayant réservé la paire annule la réservation
RESULTAT OBTENU	un autre utilisateur cherche la paire et la trouve
STATUT	OK

ID TEST	17
USER STORY	En tant qu'utilisateur, je peux filtrer les paires par marque afin de mieux choisir la paire que je souhaite
SITUATION	L'utilisateur cherche la marque souhaitée
RESULTAT OBTENU	Les paires uniquement de cette marque apparaissent
STATUT	OK

ID TEST	18
USER STORY	En tant qu'administrateur, je ne peux pas supprimer un utilisateur qui a déjà effectué des ventes afin de préserver une trace des actions
SITUATION	L'administrateur veut supprimer un vendeur ayant déjà vendu une ou plusieurs paires
RESULTAT OBTENU	L'utilisateur est bloqué à la place
STATUT	OK

Déroulement des tests

Voici le déroulement de tous mes tests. Comme on peut le voir, la majorité des tests étaient opérationnels dès le 8^e jour, ce qui correspond à la fin du développement de la dernière page qui est la page administrateur.

SNEAKERS MARKET												
N°	1ER JOUR	2E JOUR	3E JOUR	4E JOUR	5E JOUR	6E JOUR	7E JOUR	8E JOUR	9E JOUR	10E JOUR	11E JOUR	
1	X	O	O	O	O	O	O	O	O	O	O	+
2	X	O	O	O	O	O	O	O	O	O	O	
3	X	O	O	O	O	O	O	O	O	O	O	
4	X	O	O	O	O	O	O	O	O	O	O	
5	X	O	O	O	O	O	O	O	O	O	O	
6	X	X	X	X	O	O	O	O	O	O	O	
7	X	X	O	O	O	O	O	O	O	O	O	
8	X	X	O	O	O	O	O	O	O	O	O	
9	X	X	O	O	O	O	O	O	O	O	O	
10	X	X	X	X	O	O	O	O	O	O	O	
11	X	X	X	X	X	O	O	O	O	O	O	
12	X	X	X	X	X	O	O	O	O	O	O	
13	X	X	X	X	X	O	O	O	O	O	O	
14	X	X	X	X	X	X	X	O	O	O	O	
15	X	X	X	X	X	X	X	O	O	O	O	
16	X	X	X	X	X	X	X	O	O	O	O	
17	X	X	X	X	X	X	X	O	O	O	O	
18	X	X	O	O	O	O	O	O	O	O	O	
19	X	X	X	O	O	O	O	O	O	O	O	
20	X	X	X	O	O	O	O	O	O	O	O	
21	X	X	X	O	O	O	O	O	O	O	O	
22	X	X	X	O	O	O	O	O	O	O	O	

Figure 17 : Évolution des tests de SneakersMarket

Planning

Planning prévisionnel

Tâches	Heures	1er Jour	2e Jour	3e Jour	4e Jour	5e Jour	6e Jour	7e Jour	8e Jour	9e Jour	10e Jour	11e Jour
		8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00
1- S' informer	4:00											
Lire la grille d'évaluation du TPI	02:00		01:00	01:00								
Lire le cahier des charges en détail	02:00		02:00									
2- Planifier	07:00											
Découper les tâches	01:40		01:40									
Attribuer un nombre d'heures aux tâches	02:00		02:00									
Product Backlog	02:20		01:00	01:20								
Dessiner le MCD	01:00			01:00								
3- Décider	1:00											
Outils à utiliser	00:30		00:30									
Ressources à utiliser	00:30		00:30									
4- Réaliser	36:00											
Dépôt github	00:10		00:10									
Implémenter le MCD	01:00			01:00								
Structure du projet	01:00			01:00								
Connexion à la base avec PDO	01:00			01:00								
Connexion (fonctionnel + verif + erreurs)	02:00				02:00							
Inscription (fonctionnel + verif + erreurs)	02:00				02:00							
Mot de passe 8 car, maj, chiffre et car spé	00:40				00:40							
Listing des paires	01:00			01:00								
Pagination des paires	02:00				02:00							
Chercher par nom + filtrer par marque	02:20					02:20						
Détail paire de chaussure	01:20				00:20	01:00						
Admin gestion des utilisateurs	03:00				03:00							
Admin gestion points rencontres	01:00				01:00							
Admin gestion établissements	02:00					01:00	01:00					
Ajouter une paire à la liste des favoris	02:10					01:10	01:00					
Voir la liste des paires favorites	02:00						02:00					
Voir la liste des ses paires en vente	02:00						01:00	01:00				
Page profil (pouvoir modifier ses infos)	03:20							02:00	01:20			
Vues	06:00					01:00	01:00	01:00	01:00	01:00	01:00	01:00
5- Documenter	24:00											
Documentation Technique	15:20		03:10	01:40	02:00	00:20	01:10	01:40	01:40	01:20	01:40	00:40
Journal de bord	03:40		00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20
Manuel Utilisateur	03:00											03:00
Rapport de TPI	02:00									01:00	01:00	
6- Contrôler	16:00											
Planification de tests	04:00			03:00						01:00		
Rapport de tests	08:00								02:00	02:00	02:00	02:00
Déroulement de tests	04:00									02:00	02:00	
Rendu de la planification												
Fin de l'analyse												
Fin de l'implémentation												
Rendu final												

Figure 18 : Planning prévisionnel de SneakersMarket

Planning effectif

Tâches	Heures	1er Jour 2e Jour 3e Jour 4e Jour 5e Jour 6e Jour 7e Jour 8e Jour 9e Jour 10e Jour 11e Jour										
		8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00	8:00
1- S' informer	2:30											
Lire la grille d'évaluation du TPI	01:30	01:00	00:30									
Lire le cahier des charges en détail	01:00	01:00										
2- Planifier	03:40											
Découper les tâches	01:00	01:00										
Attribuer nombre d'heures aux tâches	01:00	01:00										
Product Backlog	01:00	01:00										
Dessiner le MCD	00:40	00:40										
3- Décider	0:30											
Outils à utiliser	00:15	00:15										
Ressources à utiliser	00:15	00:15										
4- Réaliser	35:10											
Dépot github	00:10		00:10									
Implémenter le MCD	01:30		01:30									
Structure du projet	00:30		00:30									
Connexion à la base avec PDO	00:10		00:10									
Connexion (fonctionnel + vérif + erreurs)	00:40		00:40									
Inscription (fonctionnel + vérif + erreurs)	01:00		01:00									
Mot de passe 8 car, maj, chiffre et car spé	00:10		00:10									
Listing des paires	00:30		00:30									
Pagination des paires	01:00			01:00								
Chercher par nom + filtrer par marque	01:30		00:30	01:00								
Détail paire de chaussure	01:00		00:30	00:30								
Admin gestion des utilisateurs	02:30				02:00	00:30						
Admin gestion points rencontres	00:00											
Admin gestion établissements	01:30					00:30	01:00					
Admin gestion des paires	01:40					01:20	00:20					
Ajouter une paire à la liste des favoris	00:30		00:30									
Voir la liste des paires favorites	00:30		00:30									
Voir la liste des ses paires en vente	01:30			01:30								
Voir la liste des ses paires achetées	01:00			01:00								
Créer Transaction	02:00		02:00									
Page profil (pouvoir modifier ses infos)	01:00			01:00								
Filtrer par taille et combine avec marque	00:30			00:30								
photo de profil utilisateur	00:40			00:40								
Réglages de petits détails	06:10		00:40	01:00	00:30	01:00	01:00	01:00	01:00	02:00		
Vues	07:30		00:30	02:00	00:30	01:00	01:00	01:00	01:30			
5- Documenter	29:40											
Documentation Technique	14:30	01:00	00:30	00:30	00:40	01:00	02:40	01:30	01:40	01:00	02:00	02:00
Journal de bord	03:40	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20	00:20
Manuel Utilisateur	05:00						01:30	01:00	01:00	01:00	01:00	00:30
Rapport de TPI	02:30									01:30	00:30	00:30
Scénario de démonstration	04:00										02:00	02:00
6- Contrôler	16:30											
Debugging	08:00					01:00	02:00	01:00	01:00	01:00	02:00	
Planification de tests	05:00	00:30	00:30		00:30	01:00			00:30	01:00	00:30	00:30
Rapport de tests	02:00		00:20	00:10	00:30			00:30		00:30		
Déroulement de tests	01:30		00:10	00:10	00:10	00:10	00:10	00:10	00:10	00:10	00:10	00:10
Rendu de la planification												
Fin de l'analyse												
Fin de l'implémentation												
Rendu final												

Figure 19 : Planning effectif de SneakersMarket

Analyse de similitudes entre les deux plannings

Comme nous pouvons le voir, il y a des divergences entre le planning prévisionnel et le planning effectif au niveau du point 4 « Réaliser ». En effet, sur le planning prévisionnel, il avait été convenu que la partie admin soit faite beaucoup plus tôt que ce qui a été réalisé. En revanche, toutes les autres fonctionnalités ont été réalisées avant ces dernières.

Ce qui diverge aussi, ce sont le nombre de tâches. Pendant la réalisation du projet, je me suis rendu compte que certaines tâches manquaient comme le scénario de démonstration. Je les ai donc ajoutées dans le planning effectif.

Les jalons ont été bien respectés, toutes les tâches significatives ont été réalisées dans les temps.

Conclusion

Difficultés rencontrées

Le plus gros problème auquel j'ai été confronté lors de la réalisation de ce projet, est sans doute le fait de travailler et retravailler mon MCD et par extension ma base de données en continu. J'avais dès le départ oublié certaines tables qui étaient essentielles à quelques fonctionnalités, et le fait de devoir la modifier en continu m'obligeait à adapter le code avec les nouvelles modifications.

Améliorations possibles

La plus grosse amélioration possible que j'ai pu penser dans la réalisation de ce projet, est sans doute l'optimisation de la page « Shop ». Ayant des images relativement lourdes, optimiser cela aurait été un bon point à ce projet, j'ai donc pensé à cette potentielle solution :

Lazy loading

Le lazy loading consiste à retarder volontairement le chargement de certaines ressources présentes sur la page, afin de laisser la place aux ressources plus importantes. Cette méthode fonctionnait bien, mais le problème avec cette méthode est qu'on remarquait que le carrousel se chargeait bien après le reste de la page, ce qui ne rendait esthétiquement pas parfait.

Devant consacrer mon temps à la réalisation des autres fonctionnalités, j'ai décidé de passer outre cette gêne. Dans une version future, je me pencherai sur cela, et essaierai de l'améliorer.

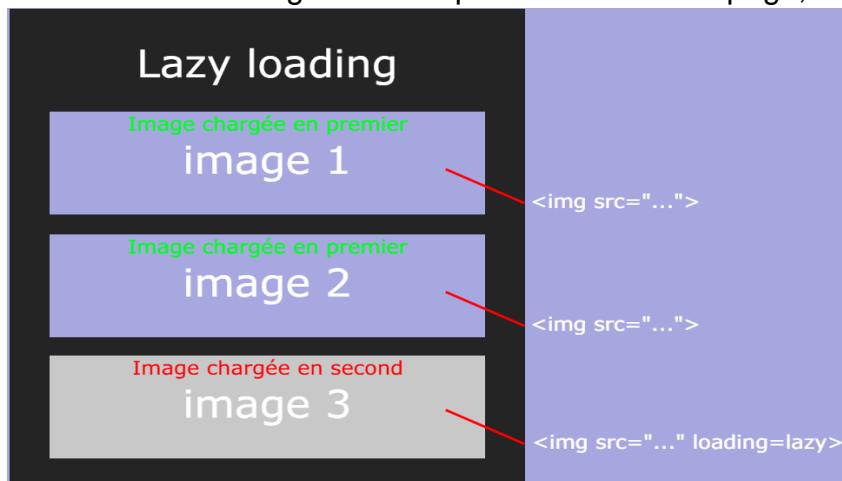


Figure 20 : schéma explicatif du lazy loading

Une autre amélioration possible serait de pouvoir vérifier l'adresse email des utilisateurs quand ils créent leur compte, en vérifiant si l'adresse email correspond bien à l'école souhaitée.

Bilan personnel

J'ai été ravi de travailler sur un projet aligné avec l'un de mes centres d'intérêt et d'avoir eu le maître d'apprentissage que je souhaitais. Bien que le planning prévisionnel n'ait pas été strictement suivi, j'ai finalement créé un site web répondant à toutes les exigences, tant celles du cahier des charges que les miennes. Je suis, dans l'ensemble, satisfait du travail que j'ai produit.

Bibliographie

Logos

Visual Studio Code

<https://uxwing.com/visual-studio-code-icon/>

DBeaver

https://fr.wikipedia.org/wiki/Fichier:DBeaver_logo.svg

Gitlab

<https://seeklogo.com/vector-logo/273186/gitlab>

Copilot

<https://uxwing.com/copilot-icon/>

Pixlr

<https://in.pinterest.com/pin/pixlr-logo-free-download--268456827778839153/>

Remove.bg

<https://marjory.io/partenaires/remove-bg/>

Codepen

<https://seeklogo.com/vector-logo/328201/codepen>

Canva

<https://www.builtinaustin.com/company/canva>

Images

Voici l'outil que j'ai utilisé pour toutes les images figurantes sur le site. Copilot est une intelligence artificielle, j'ai utilisé sa fonctionnalité « Designer » pour générer les images dont j'avais besoin.

<https://copilot.microsoft.com>

Détail des paires de chaussures

Voici une autre template que j'ai utilisé pour le design. Comme pour les cartes, j'ai pris uniquement la disposition des éléments.

<https://codepen.io/JorgeAguilar/pen/poEZWKY>

Formes qui bougent dans le login/register

J'ai utilisé cette template pour combler la zone vide à gauche de mon login et register, j'ai uniquement utilisé le javascript qui permet de dessiner et bouger les formes géométriques

<https://codepen.io/m2creates/pen/EEvGgW>

Pagination

Ce tuto m'a permis de comprendre le fonctionnement d'une pagination et de la réaliser.

<https://nouvelle-techno.fr/articles/mettre-en-place-une-pagination-en-php>

Glossaire

MOT	DEFINITION
TPI	Travail Pratique Individuel
Sneaker	Terme anglais pour les chaussures de sport ou urbaines
Product Backlog	Liste priorisée des tâches à réaliser pour le projet.
Prompt	Conversation avec une intelligence artificielle
CRUD	Représente les 4 actions de base pour interagir et gérer les données en base de données (créer, lire, mettre à jour et supprimer).

Annexes

Vous retrouverez en annexe sous format PDF :

- La Planification prévisionnelle et effective :
- Le Protocole et plan des tests :
- Les prompts générés par intelligence artificielle
- Le Code source

SNEAKERS MARKET					
PROTOCOLE DE TESTS					
ID TEST	PAGE CONCERNEE	DETAIL	LE TEST	JEU DE VALEURS	DATE
1	Login/Register	Inscription	L'utilisateur entre les informations correctement, et le compte se crée	Nom : Triano Prénom : Léo Email : leo.trn@eduge.ch Mot de passe : Super12345@	25.04.2024
2	Login/Register	Inscription	L'utilisateur entre des informations non conformes, des messages d'erreurs apparaissent et le compte n'est pas créé	Nom : [vide] Prénom : [vide] Email : leotrn@edugech Mot de passe : Super	25.04.2024
3	Login/Register	Connexion	L'utilisateur entre ses informations de connexion correctement, la connexion s'effectue et il arrive sur la page d'accueil	Email : leo.trn@eduge.ch Mot de passe : Super12345@	25.04.2024
4	Login/Register	Connexion	L'utilisateur entre un email inexistant, un message d'erreur apparaît et il n'est pas connecté	Email : leo.trn@gmail.com Mot de passe : Super12345@	25.04.2024
5	Login/Register	Connexion	L'utilisateur entre le mauvais mot de passe, un message d'erreur apparaît et il n'est pas connecté	Email : leo.trn@eduge.ch Mot de passe : Super	25.04.2024
6	Profil	Modifier profil	L'utilisateur modifie son email, se déconnecte et se reconnecte avec son nouvel email, la connexion se fait correctement	Nouvel Email : leo.trn2@gmail.com	02.05.2024
7	Accueil/Gestion	Mettre en favoris	L'utilisateur choisit une paire en favoris, va dans sa page gestion et y voit la paire	/	29.04.2024
8	Gestion	Mettre en vente	L'utilisateur entre les informations d'une paire et la met en vente, un autre utilisateur la voit dans la page d'accueil	Marque : Nykee Modèle : 1 Taille : 40 Prix : 150\$ desc : belle paire	29.04.2024
9	Gestion Admin	Gestion des utilisateurs	L'administrateur supprime un utilisateur sans ventes, l'utilisateur n'est plus en base	/	02.05.2024
10	Gestion Admin	Gestion des utilisateurs	L'administrateur bloque un utilisateur ayant fait des ventes, l'utilisateur n'apparaît plus mais est toujours en base	/	02.05.2024
11	Gestion Admin	Gestion des utilisateurs	L'administrateur ajoute un utilisateur, celui-ci apparaît en base	/	06.05.2024
12	Gestion Admin	Gestion des paires	L'administrateur supprime une paire sans réservation ni vente, celle-ci n'apparaît plus en base	/	06.05.2024
13	Gestion Admin	Gestion des paires	L'administrateur veut supprimer une paire avec réservation, la paire n'apparaît pas dans la liste de gestion	/	06.05.2024
14	Gestion Admin	Gestion des points de rencontres	L'administrateur peut ajouter un point de rencontre s'il n'existe pas déjà	Rhône	08.05.2024
15	Gestion Admin	Gestion des points de rencontres	L'administrateur tente d'ajouter un point de rencontre déjà existant, un message d'erreur apparaît et l'action ne se fait pas	Cafétéria Ternier	08.05.2024
16	Gestion Admin	Gestion des établissements	L'administrateur peut ajouter un établissement s'il n'existe pas déjà	CFPC Architecture	08.05.2024
17	Gestion Admin	Gestion des établissements	L'administrateur tente d'ajouter un établissement déjà existant, un message d'erreur apparaît et l'action ne se fait pas	CFPT Informatique	08.05.2024
18	Accueil	Filtrage des paires	L'utilisateur filtre les paires par une marque donnée, uniquement les paires de cette marque s'affichent	Nykee	29.04.2024
19	Détail	Réservation de la paire	L'utilisateur réserve la paire, un autre utilisateur cherche la même paire, elle n'apparaît plus	/	30.04.2024
20	Header	Barre de recherche	L'utilisateur cherche la marque ou modèle d'une paire, il est redirigé sur une page avec le résultat de la recherche et la recherche est correcte	Nykee	30.04.2024
21	Gestion	Paires en ventes	L'utilisateur consulte sa liste de paires en vente, il peut supprimer les paires et elles n'apparaîtront plus	/	30.04.2024
22	Gestion	Paires achetées	L'utilisateur peut consulter sa liste de paires achetées, elles y figurent toutes	/	30.04.2024



SNEAKERS MARKET					
PLAN DE TESTS					
ID PRODUCT BACKLOG	USER STORY	SITUATION	RESULTAT OBTENU	STATUT	
1	En tant qu'utilisateur, je peux créer une compte afin de profiter pleinement des fonctionnalités du siteisateur	L'utilisateur entre les informations correctement, et le compte se crée	L'inscription se fait correctement, il y a les bonnes données en session et l'utilisateur est redirigé sur home	OK	
2	En tant qu'utilisateur, je peux me connecter au compte que j'ai créé afin de continuer de profiter du site	L'utilisateur entre les informations correctement, et la connexion est autorisée	La connexion se fait correctement, les données en session sont correctes	OK	
3	En tant qu'utilisateur, je peux mettre une chaussure qui me plait en favoris afin de la retrouver plus facilement plus tard	L'utilisateur clique sur l'étoile jaune, la paire apparaît dans sa liste de favoris	Le favoris est correctement ajouté en base et l'utilisateur retrouve la bonne paire dans sa liste de favoris	OK	
4	En tant qu'utilisateur, je peux mettre en vente une paire de chaussure depuis ma page de gestion afin que les autres utilisateurs puissent la réserver	L'utilisateur rentre les bonnes informations et la paire apparaît dans la page home chez les autres utilisateurs	La paire est correctement ajoutée en base, la paire apparaît dans la page home des autres utilisateurs	OK	
5	En tant qu'utilisateur, je peux accéder à ma page de profil afin de modifier les informations de mon compte	L'utilisateur change les informations de son compte, les informations en base sont changées	Les informations sont correctement changées	OK	
6	En tant qu'utilisateur, je peux accéder à ma page de gestion afin de voir et gérer les chaussures que j'ai achetées et que j'ai à vendre	L'utilisateur voit la liste des paires achetées et à vendre	Les listes sont complètes et correctes	OK	
7	En tant qu'administrateur, je peux accéder à ma page admin afin de gérer les utilisateurs, les points de rencontres et les établissements	L'administrateur peut bloquer et débloquer un utilisateur	Le blocage s'effectue correctement et l'utilisateur n'apparaît plus en tant que vendeur	OK	
8		L'administrateur peut modifier et ajouter un point de rencontre	La modification / L'ajout s'effectue correctement	OK	
9		L'administrateur peut modifier et ajouter un établissement	La modification / L'ajout s'effectue correctement	OK	
10	En tant qu'administrateur, je peux gérer les différents utilisateurs ainsi que les paires qu'ils ont achetées et vendues afin de garantir le bon fonctionnement du site	L'administrateur peut modifier les paires à vendre d'un utilisateur	La modification s'effectue correctement	OK	
11	En tant qu'utilisateur, je peux rechercher une paire que je souhaite afin de la trouver plus facilement	L'utilisateur cherche la paire grâce à la barre de recherche	La/Les paire(s) cherchée(s) est/sont trouvée(s) et uniquement elle(s) est/sont affichée(s)	OK	
12	En tant qu'utilisateur, je ne peux pas créer un compte avec un email déjà utilisée	L'utilisateur tente de créer un compte avec un email déjà existant	L'inscription ne se fait pas et un message d'erreur apparaît	OK	
13	En tant qu'utilisateur, je ne peux pas réserver une paire ayant déjà été réversée	L'utilisateur recherche une paire déjà réservée par une autre personne	La paire n'apparait pas	OK	
14	En tant qu'utilisateur, mon mot de passe doit contenir au moins 8 caractères, un chiffre, une lettre majuscule et un caractère spécial afin de pouvoir créer mon compte	L'utilisateur tente de rentrer un mot de passe qui ne correspond pas à la sécurité demandée	L'inscription ne se fait pas et un message d'erreur apparaît	OK	
15	En tant qu'utilisateur, je ne peux plus vendre de paires si mon statut est bloqué afin de préserver la sécurité des acheteurs	L'utilisateur tente de vendre une paire en étant bloqué	La catégorie n'apparait pas dans la page de gestion	OK	
16	En tant qu'utilisateur, je peux de nouveau réserver une paire qui n'a plus le statut réservé	L'utilisateur ayant réservé la paire annule la réservation	un autre utilisateur cherche la paire et la trouve	OK	
17	En tant qu'utilisateur, je peux filtrer les paires par marque afin de mieux choisir la paire que je souhaite	L'utilisateur cherche la marque souhaitée	Les paires uniquement de cette marque apparaissent	OK	
18	En tant qu'administrateur, je ne peux pas supprimer un utilisateur qui a déjà effectué des ventes afin de préserver une trace des actions	L'administrateur veut supprimer un vendeur ayant déjà vendu une ou plusieurs paires	L'utilisateur est bloqué à la place	OK	

**Vous**

donne moi une paire de jordan 1 sur fond gris

**Designer**

"une paire de jordan 1 sur fond gris"

Designer

Optimisé par DALL-E 3

Voilà, j'ai créé une image d'une paire de Jordan 1 sur un fond gris pour vous. J'espère que cela correspond à ce que vous cherchiez ! 😊

**Vous**

donne moi une paire de jordan 1 orange sur fond gris

**Designer**

"une paire de Jordan 1 orange sur fond gris"

Designer

Optimisé par DALL-E 3

Je viens de créer une image pour vous avec une paire de Jordan 1 orange sur un fond gris. J'espère que le design vous plaira ! 😊

**Vous**

donne moi une paire de jordan 1 bleues sur fond gris

**Designer**

"une paire de Jordan 1 bleues sur fond gris"



Optimisé par DALL-E 3

Je viens de créer une image avec une paire de Jordan 1 bleues sur un fond gris. J'espère que vous aimerez cette version ! 😊

**Vous**

donne moi une paire de jordan 1 noires sur fond gris

**Designer**

"une paire de Jordan 1 noires sur fond gris"



Optimisé par DALL-E 3

**Vous**

donne moi une paire de jordan 1 vertes sur fond gris

**Designer**

"une paire de Jordan 1 vertes sur fond gris"

Designer

Optimisé par DALL-E 3

Je viens de créer une image avec une paire de Jordan 1 vertes sur un fond gris. J'espère que cette création correspond à vos attentes ! 😊

**Vous**

donne moi une paire de jordan 1 brunes sur fond gris

**Designer**

"une paire de Jordan 1 brunes sur fond gris"

Designer

Optimisé par DALL-E 3

Je viens de créer une image avec une paire de Jordan 1 brunes sur un fond gris. J'espère que vous trouverez le design à votre goût ! 😊

**Vous**

fais moi une paire de jordan 1 couleur #A8A8E1 (violet clair) sur fond gris

**Designer**

Je viens de créer une image pour vous avec une paire de Jordan 1 de couleur violet clair (#A8A8E1) sur un fond gris. J'espère que vous aimerez cette combinaison de couleurs ! 😊



"une paire de Jordan 1 couleur #A8A8E1 (violet clair) sur fond gris"

Designer

Optimisé par DALL-E 3

**Vous**

crée moi une paire de sneaker pumas sur fond gris

**Copilot**

Je vais essayer de créer cela.



2 sur 30



"une paire de sneakers Puma sur fond gris"

Designer

Optimisé par DALL-E 3

**You**

crée moi une paire de sneakers new balance sur fond gris

**Copilot**

Je vais essayer de créer cela.



3 sur 30



"une paire de sneakers New Balance sur fond gris"

Designer

Optimisé par DALL·E 3

**You**

crée moi une paire de sneakers Nike SB sur fond gris

**Copilot**

Je vais essayer de créer cela.



• 4 sur 30



"une paire de sneakers Nike SB sur fond gris"

Designer

Optimisé par DALL-E 3

**Vous**

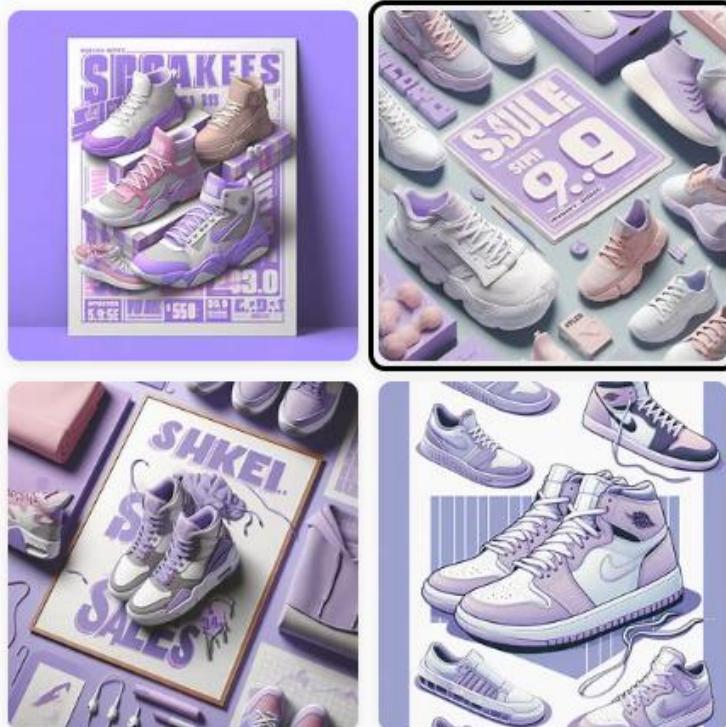
fais moi un poster pour mon site de vente de sneakers avec des couleurs lavande et blanc

**Copilot**

Je vais essayer de créer cela.



1 sur 30



"un poster pour un site de vente de sneakers avec des couleurs I..."

Designer

Optimisé par DALL-E 3

Selected files

60 printable files

index.php
src\View\ResultSearchView.php
src\View\ModifyUserView.php
src\View\ProfileView.php
src\View\ManageUserView.php
src\View\HomeView.php
src\View\HeaderView.php
src\View\GestionView.php
src\View\FavoritesView.php
src\View\DetailView.php
src\View\AuthView.php
src\View\AdminView.php
src\View\AboutView.php
src\View\js\previewImg.js
src\View\js\modalManageUser.js
src\View\js\modal.js
src\View\js\auth.js
src\View\css\profile.css
src\View\css\home.css
src\View\css\header.css
src\View\css\global.css
src\View\css\gestion.css
src\View\css\detail.css
src\View\css\auth.css
src\View\css\admin.css
src\View\css\about.css
src\Model\ProfileModel.php
src\Model\ModifyUserModel.php
src\Model\ManageUserModel.php
src\Model\headerModel.php
src\Model\HomeModel.php
src\Model\GestionModel.php
src\Model\FavoritesModel.php
src\Model\DetailModel.php
src\Model\database.php
src\Model\Constants.php
src\Model\configSample.php
src\Model\config.php
src\Model\AuthModel.php
src\Model\AdminModel.php
src\Model\objects\ViewTransaction.php
src\Model\objects\User.php
src\Model\objects\Sneaker.php
src\Model\objects\Modele.php
src\Model\objects\School.php
src\Model\objects\Brand.php
src\Model\objects\MeetPoint.php
src\Controller\ModifyUserController.php
src\Controller\ProfileController.php
src\Controller\ResultSearchController.php
src\Controller\LogoutController.php
src\Controller\ManageUserController.php
src\Controller\HomeController.php
src\Controller\headerController.php
src\Controller\GestionController.php
src\Controller\FavoritesController.php
src\Controller\DetailController.php
src\Controller\AuthController.php
src\Controller\AdminController.php
src\Controller\AboutController.php

index.php

```
1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Page index qui gère toutes les redirections grâce aux paramètre url passé en get
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10 session_start();
11
12 require_once ("src/Model/database.php");
13
14 // Tableau qui contiendra toutes les redirections possibles
15 $possibleRedirections = [];
16
17 // Tableau qui contient toutes les fichiers à ignorer
18 $ignoredFiles = [/* Ajouter les fichiers à la main */];
19
20 // Récupérer tous les fichiers Controller
21 $files = glob('src/Controller/*Controller.php');
22
23 foreach ($files as $file) {
24
25     // Supprimer le chemin de fichier pour garder que le nom
26     $file = str_replace("src/Controller/", "", $file);
27     $file = str_replace("Controller.php", "", $file);
28
29     // Vérifier si ce n'est pas un fichier à ignorer
30     if (!in_array($file, $ignoredFiles)) {
31         // Ajouter la redirection au tableau
32         $possibleRedirections[] = $file;
33     }
34 }
35
36
37 // Vérifier si on ressoit le paramètre url en get, sinon mettre la redirection par défaut à
38 // Home
39 $redirection = (isset($_GET["url"])) ? filter_input(INPUT_GET, 'url',
40 FILTER_SANITIZE_FULL_SPECIAL_CHARS) : "Home";
41
42 // Vérifier si la redirection est une redirection autorisée, sinon renvoyer à home
43 $redirection = (!in_array($redirection, $possibleRedirections)) ? "Home" : $redirection;
44
45 // Rediriger
46 foreach ($possibleRedirections as $key) {
47     if ($redirection == $key) {
48         require_once ("src/Controller/" . $key . "Controller.php");
49     }
50 }
```

src\View\ResultSearchView.php

```
1 <h1 class="title">Search Result for "<?= $search ?>"</h1>
2 <div class="center">
3     <?php if (!empty($tabPairesCards)) { ?>
4         <?php foreach ($tabPairesCards as $paire) { ?>
5             <div class="card">
```

```

6      <a href="index.php?url=Detail&id=<?= $paire->sneaker->idSneaker ?>">
7          
8          
9          <main class="main">
10              <div class="titre">
11                  <h1 class="marqueModele">
12                      <?= $paire->sneaker->nameBrand ?> <?= $paire->
sneaker->nameModel ?>
13                  </h1>
14                  <h1><span class="prix">
15                      <?= $paire->sneaker->priceSneaker ?>$
16                  </span></h1>
17              </div>
18              <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
19              <p class="description">
20                  <?= $paire->sneaker->descriptionSneaker ?>
21              </p>
22              <br>
23          </main>
24      </a>
25      <div class="postedBy">
26          <?php if ($paire->vendeur->imgUser != "") { ?>
27              
28          <?php } else { ?>
29              
30          <?php } ?>
31          <p>Posted by <span><?= $paire->vendeur->surnameUser ?></span></p>
32      </div>
33  </div>
34
35      <?php }
36  } else { ?>
37      <div class="centerGestion favoritesEmpty">
38          <div class="cardGestion cardGestionFavorites">
39              <h1 class="noFavorites">We're sorry, no Sneaker matches your search.</h1>
40          </div>
41      </div>
42  <?php } ?>
43</div>
44<?php if($pages > 1){ ?>
45<div class="pagination">
46    <?php for ($i = 1; $i <= $pages; $i++) { ?>
47        <?php if (isset($_GET["marques"])) { ?>
48            <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?url=home&marques=<?= $_GET["marques"] ?>&page=<?= $i ?>"><?
= $i ?></a>
49            <?php } else { ?>
50                <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?url=ResultSearch&search=<?= $search ?>&page=<?= $i ?>"><?= $i ?></a>
51            <?php } ?>
52        <?php } ?>
53    <?php } ?>
54</div>
55<?php } ?>
56</body>
57
58</html>

```

src\View\ModifyUserView.php

```

1 <section class="productCard">
2   <div class="container">
3     <div class="animation1 bgPreview">
4       <h1 class="title">Preview</h1>
5       <div class="center">
6         <div class="card">
7           
8           <?php if ($imgUtilisateur != "") { ?>
9             
10            <?php } else { ?>
11              <img class="imgSneaker imgProfile" src=
12 ./src/View/img/anonymous.png" />
13            <?php } ?>
14         <main class="main">
15           <div class="titre">
16             <h1 class="marqueModele">
17               <?= $nameUser ?>
18               <?= $surnameUser ?>
19             </h1>
20             <h1 class="h1Prix"><span class="prix">
21               <?= $modifyUserInstance->GetSchoolFromId($idSchool)->
22 nameSchool ?>
23             </span></h1>
24           </div>
25           <p class="h1Size"><?= $emailUser ?></p>
26         </main>
27         </a>
28       </div>
29     <div class="info">
30       <form action="index.php?url=ModifyUser" method="post" class="formulaire"
31 enctype="multipart/form-data">
32         <input type="hidden" name="idUser" value=<?= $idUser ?>>
33         
34         <h1 class="titleForm">Profile</h1>
35         <div class="ligne">
36           <p class="errorNameRegister">
37             <?= (isset($errorMessages["errorName"])) ? $errorMessages["
38 errorName"] : "" ?>
39             </p>
40           <p class="errorSurnameRegister">
41             <?= (isset($errorMessages["errorSurname"])) ? $errorMessages["
42 errorSurname"] : "" ?>
43             </p>
44           <p class="errorSchoolRegister">
45             <?= (isset($errorMessages["errorSchoolRegister"])) ? $errorMessages["
46 errorSchoolRegister"] : "" ?>
47             </p>
48         </div>
49         <div class="ligne">
50           <input type="text" name="name" class="name" placeholder="Name"
51 autocomplete="off"
52             value="<?= $nameUser ?>">
53           <input type="text" name="surname" class="surname" placeholder="Surname"
54 autocomplete="off"
55             value="<?= $surnameUser ?>">
56           <select name="school" class="school">
57             <?php foreach ($schools as $key => $value) { ?>
58               <option value="<?= $value->idSchool ?>" class="optionSize" <?=
59 ($idSchool == $value->idSchool) ? "selected" : "" ?>><?= $value->nameSchool ?></option>

```

```

53             <?php } ?>
54         </select>
55     </div>
56     <br>
57     <div class="ligne">
58         <p class="errorEmailRegister">
59             <?= (isset($errorMessages["errorEmailRegister"])) ? $errorMessages["errorEmailRegister"] : "" ?>
60         </p>
61
62         <p class="errorPasswordRegister">
63             <?= (isset($errorMessages["errorImg"])) ? $errorMessages["errorImg"]
64 ] : "" ?>
65         </p>
66     </div>
67     <div class="ligne">
68         <input type="email" name="email" class="email" placeholder="Email"
autocomplete="off"
69             value="<?= $emailUser ?>">
70         <div class="imgDiv">
71             <input type="file" name="img" class="selectImg" onchange="
72 previewFile()" id="">
73         </div>
74         <div class="submitDiv">
75             <input type="submit" value="Submit" name="submitRegister" class="submitLogin">
76         </div>
77     </form>
78     <div class="info">
79         </div>
80     </div>
81 </section>
82 </body>
83 <script src=".src/View/js/previewImg.js"></script>
84
85
86 </html>

```

src\View\ProfileView.php

```

1 <section class="productCard">
2     <div class="container">
3         <div class="animation1 bgPreview">
4             <h1 class="title">Preview</h1>
5             <div class="center">
6                 <div class="card">
7                     
8                     <?php if (isset($_SESSION['imgUser']) && $_SESSION["imgUser"] != "") {
?>
9                         
10                    <?php } else { ?>
11                        
12                    <?php } ?>
13                    <main class="main">
14                        <div class="titre">
15                            <h1 class="marqueModele">

```

```

16          <?= $_SESSION['nameUser'] ?>
17          <?= $_SESSION['surnameUser'] ?>
18      </h1>
19      <h1 class="h1Prix"><span class="prix">
20          <?= $profileInstance->GetSchoolFromId($_SESSION['
idSchool'])->nameSchool ?>
21          </span></h1>
22      </div>
23      <p class="h1Size"><?= $_SESSION["emailUser"] ?></p>
24  </main>
25  </a>
26      </div>
27  </div>
28  <div class="info">
29      <form action="index.php?url=Profile" method="post" class="formulaire" enctype="
multipart/form-data">
30          
31          <h1 class="titleForm">Profile</h1>
32          <div class="ligne">
33              <p class="errorNameRegister">
34                  <?= (isset($errorMessages["errorName"])) ? $errorMessages["
errorName"] : "" ?>
35              </p>
36              <p class="errorSurnameRegister">
37                  <?= (isset($errorMessages["errorSurname"])) ? $errorMessages["
errorSurname"] : "" ?>
38              </p>
39              <p class="errorSchoolRegister">
40                  <?= (isset($errorMessages["errorSchoolRegister"])) ? 
$errorMessages["errorSchoolRegister"] : "" ?>
41              </p>
42          </div>
43          <div class="ligne">
44              <input type="text" name="name" class="name" placeholder="Name"
autocomplete="off"
45                  value=<?= $nameUser ?>">
46              <input type="text" name="surname" class="surname" placeholder="Surname"
autocomplete="off"
47                  value=<?= $surnameUser ?>">
48              <select name="school" class="school">
49                  <?php foreach ($schools as $key => $value) { ?>
50                      <option value=<?= $value->idSchool ?> class="optionSize" <?
($idSchool == $value->idSchool) ? "selected" : "" ?><?= $value->nameSchool ?></option>
51                  <?php } ?>
52              </select>
53          </div>
54          <br>
55          <div class="ligne">
56              <p class="errorEmailRegister">
57                  <?= (isset($errorMessages["errorEmailRegister"])) ? 
$errorMessages["errorEmailRegister"] : "" ?>
58              </p>
59
60              <p class="errorPasswordRegister">
61                  <?= (isset($errorMessages["errorImg"])) ? $errorMessages["errorImg"
] : "" ?>
62              </p>
63          </div>
64          <div class="ligne">
65              <input type="email" name="email" class="email" placeholder="Email"
autocomplete="off"
66                  value=<?= $emailUser ?>">

```

```

68         <div class="imgDiv">
69             <input type="file" name="img" class="selectImg" onchange="
70 previewFile()" id="">
71         </div>
72         <div class="imgDiv">
73             <input type="submit" class="submitTransaction" name="
74 submitRegister" value="Delete profile picture">
75         </div>
76         <div class="submitDiv">
77             <input type="submit" value="Submit" name="submitRegister" class=""
78 submitLogin">
79         </div>
80     </form>
81     <div class="info">
82     </div>
83 </section>
84 </body>
85 <script src=".src/View/js/previewImg.js"></script>
86
87 </html>

```

src\View\ManageUserView.php

```

1 <h1 class="title"><?= $currentUser->nameUser ?> <?= $currentUser->surnameUser ?>'s
2 sneakers in sale</h1>
3 <div class="center">
4     <?php if (!empty($shoesInSale)) { ?>
5         <?php foreach ($shoesInSale as $sneaker) { ?>
6             <div class="card">
7                 
8                 
9                 <main class="main">
10                    <div class="titre">
11                        <h1 class="marqueModele">
12                            <?= $sneaker->sneaker->nameBrand ?>           <?= $sneaker->
13                            sneaker->nameModel ?>
14                        </h1>
15                        <h1><span class="prix">
16                            <?= $sneaker->sneaker->priceSneaker ?>$
17                            </span></h1>
18                    </div>
19                    <p class="h1Size">Size <?= $sneaker->sneaker->sizeSneaker ?></p>
20                    <p class="description">
21                        <?= html_entity_decode($sneaker->sneaker->descriptionSneaker) ?>
22                    </p>
23                    <br>
24                </main>
25                <div class="starGestion">
26                    <div class="postedBy postedByGestion">
27                        <?php if ($sneaker->vendeur->imgUser != "") { ?>
28                            
30                        <?php } else { ?>
31                            
33                        <?php } ?>
34                    <p>Posted by <span><?= $sneaker->vendeur->surnameUser ?></span><

```

```

/p>
31         </div>
32     <a class="linkStar linkStarGestion" href="#" data-user-id=<?= $idUser
33             data-sneaker-id=<?= $sneaker->sneaker->idSneaker ?>">
34             <div class="invertedRemove"></div>
35         </a>
36     </div>
37     </div>
38     <?php } ?>
39     <?php if ($pagesSneaker > 1) { ?>
40     <div class="pagination">
41         <?php for ($i = 1; $i <= $pagesSneaker; $i++) { ?>
42             <a class=<?= ($currentPageSneaker == $i) ? "currentPage" : "" ?>
43                 href="index.php?url=ManageUser&idUser=<?= $idUser ?>&pageSneaker=<
44             ?= $i ?>#sneakers"><?= $i ?></a>
45             <?php } ?>
46         </div>
47     <?php } ?>
48     <?php } else { ?>
49         <div class="centerGestion favoritesEmpty">
50             <div class="cardGestion cardGestionFavorites">
51                 <h1 class="noFavorites">There is no Sneaker in sale.</h1>
52             </div>
53         </div>
54     <?php } ?>
55 </div>
56 <div id="confirmModal" class="modal">
57     <div class="modal-content">
58         <p>Are you sure you want to delete this sneaker?</p>
59         <div class="modal-buttons">
60             <button id="confirmDelete">Yes</button>
61             <button id="cancelDelete">No</button>
62         </div>
63     </div>
64 <h1 class="title"><?= $currentUser->nameUser ?> <?= $currentUser->surnameUser ?>'s
65 sneakers sold</h1>
66 <div class="center">
67     <?php if (!empty($sneakersSold)) { ?>
68         <?php foreach ($sneakersSold as $paire) { ?>
69             <div class="card">
70                 
71                 <img class="imgSneaker" src=<?= $paire->sneaker->imgSneaker ?> />
72                 <main class="main">
73                     <div class="titre">
74                         <h1 class="marqueModele">
75                             <?= $paire->sneaker->nameBrand ?> <?= $paire->sneaker->
76 nameModel ?>
77                         </h1>
78                         <h1><span class="prix">
79                             <?= $paire->sneaker->priceSneaker ?>$
80                             <span></span></h1>
81                         </div>
82                         <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
83                         <p class="description">
84                             <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
85                             </p>
86                             <br>
87                         </main>
88                         <div class="starGestion">

```

```

87 <div class="postedBy postedByGestion">
88     <?php if ($paire->acheteur->imgUser != "") { ?>
89         Shop</h1>
2 <div class="filtres">
3     <form style="display: flex;" id="myForm" method="get" action="index.php?url=Home&
marque=<?= $idBrand ?>&size=<?= $idSize ?>">
4         <select name="marques" class="marques" onchange="document.getElementById('myForm').
.submit()">
5             <option value="0" class="optionSize" <?= ($idBrand == 0) ? "selected" : "" ?>>
All brands</option>
6             <?php foreach ($marques as $key => $value) { ?>
7                 <option value="= $value-&gt;idBrand ?&gt;" class="optionSize" &lt;?= ($idBrand ==
$value-&gt;idBrand) ? "selected" : "" ?&gt;&gt;
8                     &lt;?= $value-&gt;nameBrand ?&gt;
9                 &lt;/option&gt;
10            &lt;?php } ?&gt;
11        &lt;/select&gt;
12        &lt;select name="sizes" class="marques" onchange="document.getElementById('myForm').
.submit()"&gt;
13            &lt;option value="0" class="optionSize" &lt;?= ($idSize == 0) ? "selected" : "" ?&gt;&gt;
All sizes&lt;/option&gt;
14            &lt;?php foreach ($sizes as $key =&gt; $value) { ?&gt;
15                &lt;option value="<?= $value-&gt;sizeSneaker ?&gt;" class="optionSize" &lt;?= ($idSize ==
$value-&gt;sizeSneaker) ? "selected" : "" ?&gt;&gt;
16                    &lt;?= $value-&gt;sizeSneaker ?&gt;
17                &lt;/option&gt;
18            &lt;?php } ?&gt;
</pre

```

```

19      </select>
20  </form>
21 </div>
22 <div class="center">
23   <?php if (!empty($tabPairesCards)) { ?>
24     <?php foreach ($tabPairesCards as $paire) { ?>
25       <div class="card">
26         <a href="index.php?url=Detail&id=<?= $paire->sneaker->idSneaker ?>">
27           
28           
29           <main class="main">
30             <div class="titre">
31               <h1 class="marqueModele">
32                 <?= $paire->sneaker->nameBrand ?>          <?= $paire->
sneaker->nameModel ?>
33               </h1>
34               <h1><span class="prix">
35                 <?= $paire->sneaker->priceSneaker ?>$
36               </span></h1>
37             </div>
38             <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
39             <p class="description">
40               <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
41             </p>
42             <br>
43           </main>
44         </a>
45         <div class="postedBy">
46           <?php if ($paire->vendeur->imgUser != "") { ?>
47             
48           <?php } else { ?>
49             
50           <?php } ?>
51           <p>Posted by <span><?= $paire->vendeur->surnameUser ?></span></p>
52         </div>
53       </div>
54
55       <?php }
56   } else { ?>
57     <div class="centerGestion favoritesEmpty">
58       <div class="cardGestion cardGestionFavorites">
59         <h1 class="noFavorites">We're sorry, no Sneaker matches your search.</h1>
60         </div>
61       </div>
62     <?php } ?>
63   </div>
64 <?php if($pages > 1){ ?>
65 <div class="pagination">
66   <?php for ($i = 1; $i <= $pages; $i++) { ?>
67     <?php if (isset($_GET["marques"])) { ?>
68       <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?url=home&marques=<?= $_GET["marques"] ?>&sizes=<?= $_GET["
sizes"] ?>&page=<?= $i ?>"><?= $i ?></a>
69     <?php } else { ?>
70       <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?
url=home&page=<?= $i ?>"><?= $i ?></a>
71     <?php } ?>
72   <?php } ?>
73 </div>
74

```

```

75 <?php } ?>
76 </body>
77
78 </html>

```

src\View\HeaderView.php

```

1  <!DOCTYPE html>
2
3  <html lang="en">
4
5  <head>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Sneakers Market</title>
9      <link rel="stylesheet" href="./src/View/css/global.css">
10     <link rel="stylesheet" href="./src/View/css/home.css">
11     <link rel="stylesheet" href="./src/View/css/detail.css">
12     <link rel="stylesheet" href="./src/View/css/header.css">
13     <link rel="stylesheet" href="./src/View/css/auth.css">
14     <link rel="stylesheet" href="./src/View/css/about.css">
15     <link rel="stylesheet" href="./src/View/css/gestion.css">
16     <link rel="stylesheet" href="./src/View/css/profile.css">
17     <link rel="stylesheet" href="./src/View/css/admin.css">
18     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css"
19         integrity="your-integrity-hash" crossorigin="anonymous">
20     <link rel="icon" type="image/x-icon" href="./src/View/img/logo.png">
21 </head>
22
23 <body>
24     <header>
25         <div class="headerNav">
26             <nav>
27                 <ul>
28                     <li><a href="index.php?url=Home"></a></li>
30                     <li><a class="navA" href="index.php?url=Home">Home</a></li>
31                     <?php if (isset($_SESSION["idUser"])) { ?>
32                         <li><a class="navA" href="index.php?url=Gestion">Dashboard</a></li>
33                         <li><a class="navA" href="index.php?url=Favorites">Favorites</a><
34 /li>
35                     <?php } ?>
36                     <li><a class="navA" href="index.php?url=About">About</a></li>
37
38                     <?php if (isset($_SESSION["idRole"])) {
39                         if ($_SESSION["idRole"] == 2) { ?>
40                             <li><a class="navA" href="index.php?url=Admin">Admin</a></li><?
41                         php } ?>
42                         <?php if (isset($_SESSION['idUser'])) { ?>
43                             <li><a class="navA" href="index.php?url=Logout">Logout</a></li>
44
45                         <?php } else { ?>
46                             <li><a class="navA" href="index.php?url=Auth">Login</a></li>
47
48                         <?php } ?>
49                 </ul>

```

```

50          <li class="searchBarLi">
51              <form action="" method="post">
52                  <input type="text" name="searchBar" class="searchBar"
placeholder="Search by name...""
53                      autocomplete="off">
54                  </form>
55
56          </li>
57      <li>
58          <?php if (isset($_SESSION["surnameUser"])) { ?>
59              <a class="navA linkProfile" href="index.php?url=Profile">
60                  <p><?= $_SESSION["surnameUser"] ?></p>
61                  <img src=<?= (isset($_SESSION['imgUser']) && $_SESSION['
imgUser"] != "") ? $_SESSION['imgUser'] : "./src/View/img/anonymous.png" ?>" class=""
62          avatarHeader" alt="Avatar">
63              </a>
64          <?php } else { ?>
65              <p class="nameUser">Anonymous</p>
66              
68          <?php } ?>
69      </li>
70  </ul>
71 </nav>
72 </div>
73 </header>
74 <p class="customMessages customSuccess"><?= (isset($_SESSION['customMessage'])) ?
$_SESSION['customMessage'] : "" ?>
75 </p>
76 <p class="customMessages customError"><?= (isset($_SESSION['customError'])) ?
$_SESSION['customError'] : "" ?></p>

```

src\View\GestionView.php

```

1 <div class="anchors">
2     <h1>Navigation</h1>
3     <?php if ($gestionIsntance->IsUserNotBlocked()) { ?>
4         <a href="#CTransaction">Sell a sneaker</a>
5         <hr>
6
7         <a href="#sale">My Sneakers</a>
8         <br>
9     <?php } ?>
10    <a href="#reservations">My reservations</a>
11    <br>
12    <a href="#purchases">My purchases</a>
13    <hr>
14    <?php if ($gestionIsntance->IsUserNotBlocked()) { ?>
15        <a href="#reserved">Purchase offers</a>
16        <br>
17    <?php } ?>
18    <a href="#sold">Sneakers sold</a>
19 </div>
20 <?php if ($gestionIsntance->IsUserNotBlocked()) { ?>
21     <a id="CTransaction"></a>
22     <div class="centerGestion">
23         <h1 class="titreTransaction">Sell a sneaker</h1>
24         <div class="sellSneakerCard">
25             <form action="index.php?url=Gestion" method="post" enctype="multipart/form-
data">

```

```

26          
27      />
28      <div class="imgDiv">
29          <p class="errors errorImg"> <?= (isset($errorMessages["errorImg"])) ? 
$errorMessage["errorImg"] : "" ?>
30              </p>
31          <input type="file" name="img" class="selectImg" id="" onchange="
previewFile()">
32      </div>
33      <main class="main">
34          <div class="titre">
35              <div class="marqueModele">
36                  <div class="brandSizeLigne">
37                      <div class="brandDiv">
38                          <p class="errors errorMsgBrand">
39                              <?= (isset($errorMessages["errorBrand"])) ? 
$errorMessage["errorBrand"] : "" ?>
40                          </p>
41                      <select name="brand" class="brand" id="">
42                          <option value="-1">Brand</option>
43                          <?php foreach ($marques as $key => $value) { ?>
44                              <option value="<?= $value->idBrand ?>" class=">
optionSize"
45                                  <?= ($idBrand == $value->idBrand) ? " 
selected" : "" ?>><?= $value->nameBrand ?>
46                                  </option>
47                          <?php } ?>
48                      </select>
49                  </div>
50                  <div class="modeleDiv">
51                      <p class="errors errorMsgModele">
52                          <?= (isset($errorMessages["errorModel"])) ? 
$errorMessage["errorModel"] : "" ?>
53                          </p>
54                      <input type="text" name="modele" class="price white" 
placeholder="Model" id="">
55                          autocomplete="off" value="<?= $modele ?>">
56                  </div>
57                  <div class="prixDiv">
58                      <p class="errors errorMsgPrice">
59                          <?= (isset($errorMessages["errorPrice"])) ? 
$errorMessage["errorPrice"] : "" ?>
60                          </p>
61                      <span class="prixGestion">
62                          <input type="text" name="price" class="price" 
placeholder="price" id="">
63                          autocomplete="off" value="<?= $price ?>">$
64                      </span>
65                  </div>
66
67          </div>
68      </div>
69      <div class="h1Size">
70          <p>Size</p>
71          <input type="number" class="size" name="size" placeholder="size" 
id="" value="<?= $size ?>">
72          <p class="errors"><?= (isset($errorMessages["errorSize"])) ? 
$errorMessage["errorSize"] : "" ?></p>
73          <select name="statut" class="brand">
74              <option value="1">Visible</option>
75              <option value="2">Hidden</option>

```

```

77          </select>
78      </div>
79
80      <div class="descGestion">
81          <p class="errors"><?= ($errorMessages["errorDesc"]) ?>
82          $errorMessages["errorDesc"] : "" ?></p>
83          <input type="text" name="desc" class="desc" placeholder="Description" id="" autocomplete="off"
84              value="<?= $desc ?>">
85      </div>
86      <br>
87      <input type="submit" name="submit" class="submitTransaction" value="Create">
88          </main>
89      </form>
90  </div>
91  <a id="sale"></a>
92  <h1 class="title">My Sneakers in sale</h1>
93  <div class="center">
94      <?php if (!empty($tabPairesEnVente)) { ?>
95      <?php foreach ($tabPairesEnVente as $paire) { ?>
96          <div class="card">
97              
98              
99              <main class="main">
100                  <div class="titre">
101                      <h1 class="marqueModele">
102                          <?= $paire->sneaker->nameBrand ?> <?= $paire->
103 sneaker->nameModel ?>
104                      </h1>
105                      <h1><span class="prix">
106                          <?= $paire->sneaker->priceSneaker ?>$</span></h1>
107                  </div>
108                  <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
109                  <p class="description">
110                      <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
111                  </p>
112                  <br>
113          </main>
114          <div class="starGestion">
115              <div class="postedBy postedByGestion">
116                  <?php if ($paire->vendeur->imgUser != "") { ?>
117                      
119                  <?php } else { ?>
120                      
122                  <?php } ?>
123                  <p>Posted by <span><?= $paire->vendeur->surnameUser ?></span></p>
124          </div>
125          <a class="linkStar linkStarGestion"
126              href="index.php?url=Gestion&remove=yes&idSneaker=<?= $paire->
127 sneaker->idSneaker ?>">
128              <div class="invertedRemove"></div>
129          </a>
130      </div>
131      <?php if($gestionIsntance->IsSneakerInvisible($paire->sneaker->
132 idSneaker)){ ?>
133          <br>

```

```

130
131         <form action="" method="post">
132             <input type="hidden" name="idSneaker" value="<?= $paire->
133             sneaker->idSneaker ?>">
134             <input type="submit" class="submitTransaction" name="submit"
135             value="Set to Visible">
136             </form>
137             <?php } ?>
138         </div>
139
140         <?php }
141     } else { ?>
142         <div class="centerGestion favoritesEmpty">
143             <div class="cardGestion cardGestionFavorites">
144                 <h1 class="noFavorites">You don't have any Sneaker in sale.</h1>
145             </div>
146         <?php } ?>
147         <a id="reservations"></a>
148         <h1 class="title">My reservations</h1>
149         <div class="center">
150             <?php if (!empty($arrSneakersReserved)) { ?>
151                 <?php foreach ($arrSneakersReserved as $paire) { ?>
152                     <div class="card">
153                         
154                         
155                         <main class="main">
156                             <div class="titre">
157                                 <h1 class="marqueModele">
158                                     <?= $paire->sneaker->nameBrand ?>           <?= $paire->sneaker-
159                                     >nameModel ?>
160                                     </h1>
161                                     <h1><span class="prix">
162                                         <?= $paire->sneaker->priceSneaker ?>$
163                                         </span></h1>
164                                     </div>
165                                     <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
166                                     <p class="description">
167                                         <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
168                                         </p>
169                                         <br>
170                                         </main>
171                                         <div class="starGestion">
172                                             <div class="postedBy postedByGestion">
173                                                 <?php if ($paire->vendeur->imgUser != "") { ?>
174                                                     
176                                                 <?php } else { ?>
177                                                     
179                                                 <?php } ?>
180                                                 <p>Posted by <span><?= $paire->vendeur->surnameUser ?></span></p>
181                                         </div>
182                                         </div>
183                                         <form action="" method="post">
184                                             <input type="hidden" name="idSneaker" value="<?= $paire->sneaker->
185                                             idSneaker ?>">
186                                             <input class="submit cancelPurchase" type="submit" name="submit"
187                                             value="Cancel">
188                                         </form>
189                                     </div>

```

```

185
186      <?php ?>
187  } else { ?>
188      <div class="centerGestion favoritesEmpty">
189          <div class="cardGestion cardGestionFavorites">
190              <h1 class="noFavorites">You haven't reserved any Sneaker.</h1>
191          </div>
192      </div>
193      <?php ?>
194  </div>
195  <a id="purchases"></a>
196  <h1 class="title">My purchases</h1>
197  <div class="center">
198      <?php if (!empty($tabPairesAchetees)) { ?>
199          <?php foreach ($tabPairesAchetees as $paire) { ?>
200              <div class="card">
201                  
202                  
203                  <main class="main">
204                      <div class="titre">
205                          <h1 class="marqueModele">
206                              <?= $paire->sneaker->nameBrand ?>           <?= $paire->sneaker-
207 >nameModel ?>
208                          </h1>
209                          <h1><span class="prix">
210                              <?= $paire->sneaker->priceSneaker ?>$
211                          </span></h1>
212                      </div>
213                      <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
214                      <p class="description">
215                          <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
216                      </p>
217                      <br>
218                  </main>
219                  <div class="starGestion">
220                      <div class="postedBy postedByGestion">
221                          <?php if ($paire->vendeur->imgUser != "") { ?>
222                              
224                          <?php } else { ?>
225                              
227                          <?php } ?>
228                          <p>Posted by <span><?= $paire->vendeur->surnameUser ?></span></p>
229                      </div>
230                      <div class="meetPoint">
231                          <?php $dhMeetPoint = new DateTime($paire->meetPoint->
dateHourMeetPlace) ?>
232                          <p><i class="fas fa-map-marker-alt"></i> <?= $paire->meetPoint->
233 adressMeetPoint ?></p>
234                          <p><i class="far fa-calendar-alt"></i> <?= $dhMeetPoint->format('d-m-Y
H:i') ?>
235                          </p>
236                      </div>
237                  </div>
238
239          <?php ?>
240      } else { ?>
241          <div class="centerGestion favoritesEmpty">
242              <div class="cardGestion cardGestionFavorites">

```

```

240             <h1 class="noFavorites">You haven't purchased any Sneaker.</h1>
241         </div>
242     </div>
243     <?php } ?>
244 </div>
245 <?php if ($gestionIsntance->IsUserNotBlocked()) { ?>
246     <a id="reserved"></a>
247     <h1 class="title">Purchase offers</h1>
248     <div class="center">
249         <?php if (!empty($sneakersReserved)) { ?>
250             <?php foreach ($sneakersReserved as $paire) { ?>
251                 <div class="card">
252                     
253                     
254                     <main class="main">
255                         <div class="titre">
256                             <h1 class="marqueModele">
257                                 <?= $paire->sneaker->nameBrand ?> <?= $paire->
258 sneaker->nameModel ?>
259                             </h1>
260                             <h1><span class="prix">
261                                 <?= $paire->sneaker->priceSneaker ?>$
262                                 </span></h1>
263                         </div>
264                         <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
265                         <p class="description">
266                             <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
267                         </p>
268                         <br>
269                     </main>
270                     <div class="starGestion">
271                         <div class="postedBy postedByGestion">
272                             <?php if ($paire->acheteur->imgUser != "") { ?>
273                                 
275                             <?php } else { ?>
276                                 
278                             <?php } ?>
279                         <p>Offered by <span><?= $paire->acheteur->surnameUser ?><
280 /span></p>
281                         </div>
282                     </div>
283                     <div class="meetPoint">
284                         <p><i class="fas fa-map-marker-alt"></i> <?= $paire->meetPoint->
285 adressMeetPoint ?></p>
286                         <?php $dhMeetPoint = new DateTime($paire->meetPoint->
287 dateHourMeetPlace) ?>
288                         <p><i class="far fa-calendar-alt"></i> <?= $dhMeetPoint->format('
289 d-m-Y H:i') ?></p>
290                         </div>
291                         <form action="" method="post" class="acceptDeny">
292                             <input type="hidden" name="idSneaker" value="<?= $paire->sneaker->
293 idSneaker ?>">
294                             <input class="submit acceptPurchase" type="submit" name="submit"
295 value="Accept">
296                             <input class="submit denyPurchase" type="submit" name="submit"
297 value="Deny">
298                         </form>
299                     </div>
300             <?php } ?>

```

```

292     } else { ?>
293         <div class="centerGestion favoritesEmpty">
294             <div class="cardGestion cardGestionFavorites">
295                 <h1 class="noFavorites">You didn't received any purchase offer.</h1>
296             </div>
297         </div>
298     </?php } ?>
299 </div>
300 <?php } ?>
301 <a id="sold"></a>
302 <h1 class="title">Sneakers sold</h1>
303 <div class="center">
304     <?php if (!empty($sneakersSold)) { ?>
305         <?php foreach ($sneakersSold as $paire) { ?>
306             <div class="card">
307                 
308                 
309                 <main class="main">
310                     <div class="titre">
311                         <h1 class="marqueModele">
312                             <?= $paire->sneaker->nameBrand ?>          <?= $paire->sneaker-
313 >nameModel ?>
314                         </h1>
315                         <h1><span class="prix">
316                             <?= $paire->sneaker->priceSneaker ?>$
317                         </span></h1>
318                         <p class="h1Size">Size <?= $paire->sneaker->sizeSneaker ?></p>
319                         <p class="description">
320                             <?= html_entity_decode($paire->sneaker->descriptionSneaker) ?>
321                         </p>
322                         <br>
323                     </main>
324                     <div class="starGestion">
325                         <div class="postedBy postedByGestion">
326                             <?php if ($paire->acheteur->imgUser != "") { ?>
327                                 
329                             <?php } else { ?>
330                                 
332                             <?php } ?>
333                             <p>Offered by <span><?= $paire->acheteur->surnameUser ?></span><
334 /p>
335                         </div>
336                         <div class="meetPoint">
337                             <?php $dhMeetPoint = new DateTime($paire->meetPoint->
dateHourMeetPlace) ?>
338                             <p><i class="fas fa-map-marker-alt"></i> <?= $paire->meetPoint->
339 adressMeetPoint ?></p>
340                             <p><i class="far fa-calendar-alt"></i> <?= $dhMeetPoint->format('d-m-Y
H:i') ?>
341                             </p>
342                         </div>
343                     </div>
344             <?php } ?>
345             <div class="centerGestion favoritesEmpty">
346                 <div class="cardGestion cardGestionFavorites">
347                     <h1 class="noFavorites">You haven't sold any Sneaker.</h1>

```

```

347         </div>
348     </div>
349     <?php } ?>
350 </div>
351 </body>
352
353 <script src=".src/View/js/previewImg.js"></script>
354
355 </html>

```

src\View\FavoritesView.php

```

1 <h1 class="title" style="padding-top:10rem">My favorites</h1>
2 <div class="center">
3     <a id="favorites"></a>
4     <?php if (!empty($userFavList)) { ?>
5         <?php foreach ($userFavList as $key => $value) { ?>
6             <div class="card">
7                 <a href="index.php?url=Detail&id=<?= $value->sneaker->idSneaker ?>">
8                     
9                     
10                <main class="main">
11                    <div class="titre">
12                        <h1 class="marqueModele">
13                            <?= $value->sneaker->nameBrand ?> <?= $value->
sneaker->nameModel ?>
14                            </h1>
15                            <h1><span class="prix">
16                                <?= $value->sneaker->priceSneaker ?>$
17                                </span></h1>
18                            </div>
19                            <p class="h1Size">Size <?= $value->sneaker->sizeSneaker ?></p>
20                            <p class="description">
21                                <?= html_entity_decode($value->sneaker->descriptionSneaker) ?>
22                            </p>
23                            <br>
24                        </main>
25                        <div class="starGestion">
26                            <div class="postedBy postedByGestion">
27                                <?php if ($value->vendeur->imgUser != "") { ?>
28                                    
29                                <?php } else { ?>
30                                    
31                                <?php } ?>
32                                <p>Posted by <span><?= $value->vendeur->surnameUser ?></span><
/p>
33                                </div>
34                                <a class="linkStar linkStarGestion"
35                                    href="index.php?url=Favorites&favorite=yes&idSneaker=<?=
$value->sneaker->idSneaker ?>">
36                                    <div class="invertedStar"></div>
37                                </a>
38                            </div>
39                            </a>
40                        </div>
41                    <?php }
42                } else { ?>
43                    <div class="centerGestion favoritesEmpty">

```

```

44         <div class="cardGestion cardGestionFavorites">
45             <h1 class="noFavorites">You don't have any Sneaker added to your favorites.
46         </div>
47     </div>
48     <?php } ?>
49 </div>
50 <?php if ($pages > 1) { ?>
51
52     <div class="pagination">
53         <?php for ($i = 1; $i <= $pages; $i++) { ?>
54             <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?url=Favorites&page=<?= $i ?>"><?= $i ?></a>
55             <?php } ?>
56         </div>
57     <?php } ?>

```

src\View\DetailView.php

```

1  <!DOCTYPE html>
2
3  <html lang="en">
4
5  <head>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Sneakers Market</title>
9      <link rel="icon" type="image/x-icon" href=".src\View\img\logo.png">
10     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"
11         integrity="sha512-SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJk19QevSCWr3W6A=="
12         crossorigin="anonymous" referrerPolicy="no-referrer" />
13     <script src="https://cdnjs.cloudflare.com/ajax/libs/paper.js/0.12.15/paper-full.min.js"></script>
14     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
15     <link rel="stylesheet" href=".src\View\css\global.css">
16     <link rel="stylesheet" href=".src\View\css\detail.css">
17 </head>
18
19 <body>
20     <section class="productCard">
21         <div class="container">
22             <div class="info">
23                 <a href="index.php?url=Shop" class="goBackLink"><i class="fa fa-arrow-circle-left" aria-hidden="true"></i>Home</a>
24             <div class="namePrice">
25                 <h1>
26                     <?= $paire->nameBrand ?> <?= $paire->nameModel ?>
27                 </h1>
28                 <h1 class="price">
29                     <?= $paire->priceSneaker ?>$
30                 </h1>
31             </div>
32             <h1 class="desc">
33                 Size <?= $paire->sizeSneaker ?> - <?= html_entity_decode($paire->
34 descriptionSneaker) ?>
35                 </h3>
36                 <div class="attribs">
37                     <form method="post" action="" class="form">

```

```

38         <div class="attrib">
39             <?php if (!isset($_SESSION['idUser'])) { ?>
40                 <p class="errorNotConnected"><a class="errorNotConnected" href="index.php?url=Auth">You must be
41                     connected to like or reserve this Sneaker</a></p>
42             <?php } else { ?>
43                 <div style="display:flex; align-items:center">
44                     <div style="display:block">
45                         <p style="width:100%" class="errorMeetPoint">
46                             <?= (isset($errorMessages["meetPoint"])) ? $errorMessages["meetPoint"] : "" ?>
47                         </p>
48                         <select name="meetPoint" <?= ($detailInstance->IsReserved($idSneaker)
49 ) ? "disabled" : "" ?>
50                             class="size" onchange="document.getElementById('myForm').submit()">
51                             <option value="0" class="optionSize" <?= ($idMeetPoint == 0) ? "selected" : "" ?>>Select a meet point
52                             </option>
53                             <?php foreach ($meetPoints as $key => $value) { ?>
54                                 <option value="<?= $value->idMeetPoint ?>" class="optionSize"
55                                     <?= ($idMeetPoint == $value->idMeetPoint) ? "selected" : "" ?>>
56                                     <?= $value->nameMeetPoint ?>
57                                 </option>
58                             <?php } ?>
59                         </select>
60                     </div>
61                     <div style="display:block; margin-left:2%; ">
62                         <p style="width:100%" class="errorMeetPoint">
63                             <?= (isset($errorMessages["meetingDate"])) ? $errorMessages["meetingDate"] : "" ?>
64                         </p>
65                         <input type="datetime-local" style="margin-left:2%; height:33px" id="datetime" name="meetingDate"
66                             <?= ($detailInstance->IsReserved($idSneaker)) ? "disabled" : "" ?>>
67                         </div>
68                     </div>
69                     <div class="moitie">
70                         <a class="linkStar" href="index.php?url=Detail&id=<?= $paire->idSneaker
71 ?>&favorite=yes">
72                             <div class="<?= ($isFavorite) ? "invertedStar" : "star" ?>"></div>
73                         </a>
74                         <?php if (!$detailInstance->IsReserved($paire->idSneaker) &&
75 !$detailInstance->doesUserHasReserved($idSneaker)) { ?>
76                             <input class="submit" type="submit" name="submit" value="Reserve">
77                         <?php } else if ($detailInstance->IsReserved($paire->idSneaker) &&
78 $detailInstance->doesUserHasReserved($idSneaker)) { ?>
79                             <input class="submit" type="submit" name="submit" value="Cancel">
80                         <?php } ?>
81                     </div>
82                 </div>
83                 <div class="colorLayer"></div>
84                 <div class="preview">
85                     <div class="img">
86                         
87                     </div>
88                 </div>
89             </div>
90         </section>

```

```

91 |     <p class="customMessages customSuccess"><?= ($SESSION['customMessage']) ?  

92 |     $SESSION['customMessage'] : "" ?>  

93 |     </p>  

94 |     <p class="customMessages customError"><?= ($SESSION['customError']) ? $SESSION['  

95 |     customError'] : "" ?></p>  

96 |   </body>  

97 | </html>

```

src\View\AuthView.php

```

1 | <!DOCTYPE html>  

2 |  

3 | <html lang="en">  

4 |  

5 | <head>  

6 |   <meta charset="UTF-8">  

7 |   <meta name="viewport" content="width=device-width, initial-scale=1.0">  

8 |   <title>Sneakers Market</title>  

9 |   <link rel="icon" type="image/x-icon" href=".src/View/img/logo.png">  

10 |   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-  

11 |     awesome/6.5.2/css/all.min.css"  

12 |       integrity="sha512-  

13 |       SnH5WK+bZxgPHs44uWIX+LLJAj9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJk19QevSCWr3W6A=="  

14 |         crossorigin="anonymous" referrerpolicy="no-referrer" />  

15 |   <script src="https://cdnjs.cloudflare.com/ajax/libs/paper.js/0.12.15/paper-  

16 |     full.min.js"></script>  

17 |   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><  

18 | /script>  

19 |  

20 |   <link rel="stylesheet" href=".src/View/css/global.css">  

21 |   <link rel="stylesheet" href=".src/View/css/auth.css">  

22 | </head>  

23 |  

24 | <body>  

25 |   <section class="productCard">  

26 |     <div class="container">  

27 |       <div class="animation1 slide-right">  

28 |         <canvas id="canvas" class="canvas-back"></canvas>  

29 |       </div>  

30 |       <div class="info">  

31 |         <a href="index.php?url=Home" class="goBackLink"><i class="fa fa-arrow-  

32 |           circle-left"></i> Home</a>  

33 |  

34 |           <form action="index.php?url=Auth&slide=right" method="post" class="formulaire" enctype="multipart/form-data">  

35 |               

36 |             <h1 class="titleForm">Register</h1>  

37 |             <div class="ligne">  

38 |               <p class="errorNameRegister">  

39 |                 <?= ($errorMessages["errorName"]) ? $errorMessages["  

40 | errorName"] : "" ?>  

41 |               </p>  

42 |               <p class="errorSurnameRegister">  

43 |                 <?= ($errorMessages["errorSurname"]) ? $errorMessages["  

44 | errorSurname"] : "" ?>  

45 |               </p>  

46 |               <p class="errorSchoolRegister">  

47 |                 <?= ($errorMessages["errorSchoolRegister"]) ?  

48 | $errorMessages["errorSchoolRegister"] : "" ?>  

49 |               </p>

```

```

42         </div>
43         <div class="ligne">
44             <input type="text" name="name" class="name" placeholder="Name"
45             autocomplete="off" value="<?= $nameUser ?>">
46             <input type="text" name="surname" class="surname" placeholder="Surname"
47             autocomplete="off" value="<?= $surnameUser ?>">
48             <select name="school" class="school">
49                 <option value="-1" class="optionSize">Select School</option>
50                 <?php foreach ($schools as $key => $value) { ?>
51                     <option value="<?= $value->idSchool ?>" class="optionSize"
52 <?= ($idSchool == $value->idSchool) ? "selected" : "" ?><?= $value->nameSchool ?><
53 /option>
54                     <?php } ?>
55                 </select>
56             <br>
57             <div class="ligne">
58                 <p class="errorEmailRegister">
59                     <?= (isset($errorMessages["errorEmailRegister"])) ?
60 $errorMessages["errorEmailRegister"] : "" ?>
61                 </p>
62                 <p class="errorPasswordRegister">
63                     <?= (isset($errorMessages["errorPasswordRegister"])) ?
64 $errorMessages["errorPasswordRegister"] : "" ?>
65                 </p>
66             </div>
67             <div class="ligne">
68                 <input type="email" name="email" class="email" placeholder="Email"
69             autocomplete="off" value="<?= $emailUser ?>">
70             <input type="password" name="password" class="passwordRegister"
71             placeholder="Password">
72                 <div class="imgDiv">
73                     <input type="file" name="img" class="selectImg" id="">
74                 </div>
75                 <div class="submitDiv">
76                     <input type="submit" value="Submit" name="submitRegister" class="submitLogin">
77                     <a href="#" class="loginLink">Login</a>
78                 </div>
79             </form>
80         </div>
81         <div class="info">
82             <a href="index.php?url=Home" class="goBackLink"><i class="fa fa-arrow-circle-left"></i> Home</a>
83
84             <form action="index.php?url=Auth&slide=left" method="post" class="formulaire">
85                 
86                 <h1 class="titleForm">Login</h1>
87                 <p class="errorLogin">
88                     <?= (isset($errorMessages["errorLogin"])) ? $errorMessages["errorLogin"] : "" ?>
89                 </p>
90                 <input type="email" name="email" class="email" placeholder="Email"
91             autocomplete="off"

```

```

91         value="<?= $emailUser ?>">
92     <input type="password" name="password" class="password" placeholder=""
93     Password">
94     <div class="submitDiv">
95         <input type="submit" value="Submit" name="submitLogin" class="submitLogin">
96         <a href="#" class="registerLink">Register</a>
97     </div>
98     </form>
99   </div>
100 </section>
101
102 <script src=".src/View/js/auth.js"></script>
103
104 </body>
105
106 </html>

```

src\View\AdminView.php

```

1 <div class="anchors">
2   <h1>Navigation</h1>
3   <a href="#users">Manage users</a>
4   <br>
5   <a href="#addU">Add an user</a>
6   <br>
7   <a href="#sneakers">Manage sneakers</a>
8   <hr>
9   <a href="#schools">Manage schools</a>
10  <br>
11  <a href="#meetPoints">Manage meet points</a>
12 </div>
13 <a id="users"></a>
14 <h1 class="title">Manage users</h1>
15 <form action="index.php?url=Admin#users" method="post">
16   <input type="text" name="searchBarUsers" class="searchBar searchBarUsers"
17     placeholder="Search by name, surname or email..." autocomplete="off">
18 </form>
19 <div class="center">
20   <?php if (!empty($users)) { ?>
21     <?php foreach ($users as $user) { ?>
22       <div class="card cardUsers">
23         <a href="index.php?url=ManageUser&idUser=<?= $user->idUser ?>">
24
25           
26           <?php if ($user->imgUser != "") { ?>
27             
28           <?php } else { ?>
29             
30           <?php } ?>
31         <main class="main">
32           <div class="titre">
33             <h1 class="marqueModele">
34               <?= $user->nameUser ?>
35               <?= $user->surnameUser ?>
36             </h1>
37             <h1 class="h1Prix smaller"><span class="prix">

```

```

38 <?php if ($adminInstance->GetSchoolFromId($user->idSchool) ->
39     nameSchool ?>
40         </span></h1>
41     </div>
42     <p class="h1Size"><?= $user->emailUser ?></p>
43     </a>
44     <div class="gestionUsers">
45         <form action="index.php?url=ModifyUser" method="post">
46             <input type="hidden" name="idUser" value="<?= $user->idUser ?>">
47             <input class="submit" type="submit" name="submit" value="Modify">
48         </form>
49         <form action="" class="" method="post">
50             <input type="hidden" name="idUser" value="<?= $user->idUser ?>">
51             <?php if ($adminInstance->CountUsersTransactions($user->idUser) >
52             0) { ?>
53                 <?php if ($adminInstance->IsBlocked($user->idUser)) { ?>
54                     <input class="submit green" type="submit" name="submit"
55                     value="Unblock">
56                 <?php } else { ?>
57                     <input class="submit red" type="submit" name="submit"
58                     value="Block">
59                 <?php } ?>
60             </form>
61         </div>
62         </main>
63     </div>
64     <?php } ?>
65     <?php if ($pages > 1) { ?>
66
67         <div class="pagination">
68             <?php for ($i = 1; $i <= $pages; $i++) { ?>
69                 <a class="<?= ($currentPage == $i) ? "currentPage" : "" ?>" href="index.php?url=Admin&page=<?= $i ?>"><?= $i ?></a>
70             <?php } ?>
71         </div>
72     <?php } ?>
73     <?php } else { ?>
74         <div class="centerGestion favoritesEmpty">
75             <div class="cardGestion cardGestionFavorites">
76                 <h1 class="noFavorites">No user matches your search.</h1>
77             </div>
78         </div>
79     <?php } ?>
80     <?php } ?>
81 </div>
82 <a id="addU"></a>
83 <h1 class="title">Add an user</h1>
84 <div class="centerGestion favoritesEmpty">
85     <div class="cardGestion cardGestionFavorites">
86         <form action="index.php?url=Admin#addU" method="post" class="formulaire" enctype="multipart/form-data">
87             <div class="ligne">
88                 <p class="errorEmailRegister">
89                     <?= (isset($errorMessages["errorName"])) ? $errorMessages["errorName"] : "" ?>
90                 </p>
91                 <p class="errorPasswordRegister">

```

```

92             <?= ($errorMessages["errorSurname"])) ? $errorMessages["  

93         errorSurname"] : "" ?>  

94             </p>  

95         </div>  

96         <div class="ligne">  

97             <input type="text" name="nameUser" class="name" placeholder="Name"  

98         autocomplete="off"  

99             value="<?= $nameUser ?>">  

100            <input type="text" name="surnameUser" class="surname" placeholder="  

101        Surname" autocomplete="off"  

102            value="<?= $surnameUser ?>">  

103        </div>  

104        <br>  

105        <div class="ligne">  

106            <p class="errorEmailRegister">  

107                <?= ($errorMessages["errorEmailRegister"])) ? $errorMessages["  

108        errorEmailRegister"] : "" ?>  

109                </p>  

110                <p class="errorPasswordRegister">  

111                    <?= ($errorMessages["errorPasswordRegister"])) ?  

112                $errorMessages["errorPasswordRegister"] : "" ?>  

113                    </p>  

114                </div>  

115                <div class="ligne">  

116                    <input type="email" name="emailUser" class="email" placeholder="Email"  

117        autocomplete="off"  

118                    value="<?= $emailUser ?>">  

119                    <input type="password" name="pwdUser" class="passwordRegister"  

120        placeholder="Password">  

121                    </div>  

122                    <br>  

123                    <div class="ligne">  

124                        <p class="errorPasswordRegister">  

125                            <?= ($errorMessages["errorSchoolRegister"])) ? $errorMessages["  

126        errorSchoolRegister"] : "" ?>  

127                            </p>  

128                            <p class="errorPasswordRegister">  

129                                <?= ($errorMessages["errorImg"])) ? $errorMessages["errorImg"] :  

130                                "" ?>  

131                                </p>  

132                            </div>  

133                            <div class="ligne">  

134                                <select name="idSchool" class="school">  

135                                    <option value="-1" class="optionSize">Select School</option>  

136                                    <?php foreach ($schools as $key => $value) { ?>  

137                                        <option value="<?= $value->idSchool ?>" class="optionSize" <?=  

138 ($idSchool == $value->idSchool) ? "selected" : "" ?>><?= $value->nameSchool ?></option>  

139                                    <?php } ?>  

140                                </select>  

141                                <input type="file" name="imgUser" class="selectImg">  

142                            </div>  

143  

144                                <input type="submit" value="Register" name="submit" class="submitLogin">  

145                            </form>  

146                        </div>  

147                    </div>  

148                    <a id="sneakers"></a>  

149                    <h1 class="title">Manage Sneakers</h1>  

150                    <div class="center">  

151                        <?php if (!empty($shoesInSale)) { ?>  

152                            <?php foreach ($shoesInSale as $sneaker) { ?>  

153                                <div class="card">

```

```

144         
145         
200    </div>
201  </div>
202  <a id="schools"></a>
203  <h1 class="title">Manage schools</h1>
204  <div class="centerGestion favoritesEmpty">
205    <div class="cardGestion cardGestionFavorites">
206      <div class="errorMsg">
207        <p><?= ($isset($errorMsgCreateSchool["nameSchool"])) ? $errorMsgCreateSchool["<
208          nameSchool"] : "" ?></p>
209        <p></p>
210      </div>
211      <form action="index.php?url=Admin#schools" method="post" class="addMeetPoint">
212        <input type="text" name="nameSchool" placeholder="School's name" class="name" autocomplete="off"
213          value="<?= $nameMeetPoint ?>">
214        <input type="submit" class="submit" name="submit" value="Create school">
215      </form>
216      <table class="tableMeetPoints">
217        <tr>
218          <th>Name</th>
219          <th>Action</th>
220        </tr>
221        <?php foreach ($adminInstance->GetAllSchools() as $key => $value) { ?>
222          <form action="index.php?url=Admin#meetPoints" method="post">
223            <tr>
224              <th><input type="text" name="editNameSchool" class="name editInfoMeetPoint"
225                value="<?= $value->nameSchool ?>" autocomplete="off"></th>
226              <th>
227                <div class="submitMeetPoints">
228                  <input type="hidden" name="idSchool" value="<?= $value->
229                    idSchool ?>">
230                  <?php if (!$adminInstance->DoesSchoolHasUser($value->
231                    idSchool)) { ?>
232                    <input type="submit" name="submit" style="background-
233                      color: #dc3545;" value="Remove school">
234                  <?php } else { ?>
235                    <input type="submit" class="submit removeSubmit" value="Remove school" disabled>
236                  <?php } ?>
237                  <input type="submit" name="submit" class="submit" value="Edit school">
238                </div>
239              </th>
240            </tr>
241          <?php } ?>
242        </table>
243      </div>
244    </div>
245
246  <a id="meetPoints"></a>
247  <h1 class="title">Manage meet points</h1>
248  <div class="centerGestion favoritesEmpty">
249    <div class="cardGestion cardGestionFavorites">
250      <div class="errorMsg">
251        <p><?= ($isset($errorMessages["nameMeetPoint"])) ? $errorMessages["<

```

```

252     <p><?= ($errorMessages["adressMeetPoint"]) ? $errorMessages["  

253     adressMeetPoint"] : "" ?></p>  

254     <p><?= ($errorMessages["descriptionMeetPoint"]) ? $errorMessages["  

255     descriptionMeetPoint"] : "" ?></p>  

256     </div>  

257     <form action="index.php?url=Admin#meetPoints" method="post" class="addMeetPoint">  

258         <input type="text" name="nameMeetPoint" placeholder="Meet point's name" class="name" autocomplete="off"  

259             autocomplete="off" value="<?= $nameMeetPoint ?>">  

260         <input type="text" name="adressMeetPoint" placeholder="Meet point's adress" class="name" autocomplete="off"  

261             autocomplete="off" value="<?= $adressMeetPoint ?>">  

262         <input type="text" name="descriptionMeetPoint" placeholder="Meet point's  

263         description" class="name" autocomplete="off" value="<?= $descriptionMeetPoint ?>">  

264     </form>  

265     <table class="tableMeetPoints">  

266         <tr>  

267             <th>Name</th>  

268             <th>Adress</th>  

269             <th>Description</th>  

270             <th>Action</th>  

271         </tr>  

272         <?php foreach ($adminInstance->GetAllMeetPoints() as $key => $value) { ?>  

273             <form action="index.php?url=Admin#meetPoints" method="post">  

274  

275                 <tr>  

276                     <th><input type="text" name="editNameMeetPoint" class="name  

277                     editInfoMeetPoint" autocomplete="off"  

278                         value="<?= html_entity_decode($value->nameMeetPoint) ?>"><  

279                     /th>  

280                     <th><input type="text" name="editAdressMeetPoint" class="name  

281                     editInfoMeetPoint" autocomplete="off"  

282                         value="<?= html_entity_decode($value->adressMeetPoint) ?>"><  

283                     /th>  

284                     <th><input type="text" name="editDescriptionMeetPoint" class="name  

285                     editInfoMeetPoint" autocomplete="off" value="<?= html_entity_decode($value->  

286                     descriptionMeetPoint) ?>"></th>  

287                     <th>  

288                         <div class="submitMeetPoints">  

289                             <input type="hidden" name="idMeetPoint" value="<?= $value->  

290                             idMeetPoint ?>">  

291                             <?php if (!$adminInstance->  

292                             DoesTransactionHasMeetPoint($value->idMeetPoint)) { ?>  

293                                 <input type="submit" name="submit" style="background-  

294                                 color: #dc3545;"  

295                                 class="submit removeSubmit" value="Remove meet  

296                                 point">  

297                             <?php } else { ?>  

298                                 <input type="submit" class="submit removeSubmit  

299                                 locked" value="Remove meet point" disabled>  

300                             <?php } ?>  

301                             <input type="submit" name="submit" class="submit" value="  

302                             Edit meet point">  

303                         </div>  

304                     </th>  

305                 </tr>  

306             </form>  

307         <?php } ?>

```

```

298     </table>
299     </div>
300 </div>
301 <script src=".src/View/js/modal.js"></script>

```

src\View\AboutView.php

```

1 <link rel="stylesheet" href=".src/View/css/home.css">
2
3 <div class="firstPart">
4     <h1 class="titleHome">Sneakers Market.</h1>
5
6     <div class="circleTop"></div>
7     <div class="circleBottom"></div>
8 </div>
9 <div class="secondPart">
10    <div class="circleRight">
11        <div class="content">
12            <h1 class="titleDesc titleHome">What is Sneakers Market ?</h1>
13            <div class="ligneDesc">
14                
15                <p class="textDesc">Sneakers Market: Your go-to platform for buying and
16 selling kicks. Explore, buy, and sell the latest sneaker releases hassle-free. Join our
17 community of sneaker enthusiasts today!</p>
18        </div>
19    </div>
20 <div class="circleLeft">
21    <div class="content">
22        <h1 class="titleDesc titleHome">Who created Sneakers Market ?</h1>
23        <div class="ligneDesc">
24            
25            <p class="textDesc">LEO.TRN, a 4th year CFPT Informatique student, carried
26 out this project as part of
27            <the TPI.</p>
28        </div>
29    </div>
30 </body>
31
32 </html>

```

src\View\js\previewImg.js

```

1 function previewFile() {
2     var preview = document.querySelector('.imgSneaker');
3     var fileInput = document.querySelector('.selectImg');
4     var file = fileInput.files[0];
5     var reader = new FileReader();
6
7     if (file && isFileExtensionAllowed(file.name)) {
8         reader.addEventListener("load", function () {
9             preview.src = reader.result;
10        }, false);
11
12        reader.readAsDataURL(file);
13

```

```

14     }
15 }
16
17 function isFileExtensionAllowed(filename) {
18     var allowedExtensions = ['jpg', 'jpeg', 'png', 'gif'];
19     var ext = filename.split('.').pop().toLowerCase();
20     return allowedExtensions.includes(ext);
21 }
22

```

src\View\js\modalManageUser.js

```

1 // Récupérez le modal
2 var modal = document.getElementById("confirmModal");
3
4 // Récupérez le lien de suppression
5 var deleteLink = document.querySelectorAll(".linkStarGestion");
6
7 // Bouclez sur tous les liens de suppression pour ajouter un événement de clic
8 deleteLink.forEach(function (link) {
9     link.addEventListener("click", function (event) {
10         event.preventDefault(); // Empêcher le lien de se comporter par défaut
11         var userId = this.dataset.userId; // Récupérer l'ID de l'utilisateur
12         var sneakerId = this.dataset.sneakerId; // Récupérer l'ID de la sneaker
13
14         // Afficher le modal de confirmation
15         modal.style.display = "block";
16
17         // Confirmer la suppression
18         document.getElementById("confirmDelete").onclick = function () {
19             // Rediriger vers la page de suppression avec les IDs de l'utilisateur et de la
sneaker
20             window.location.href = "index.php?url=ManageUser&remove=yes&idUser=" + userId +
"&idSneaker=" + sneakerId;
21         }
22
23         // Annuler la suppression
24         document.getElementById("cancelDelete").onclick = function () {
25             modal.style.display = "none"; // Cacher le modal
26         }
27     });
28 });
29

```

src\View\js\modal.js

```

1 var modal = document.getElementById("confirmModal");
2 var deleteLink = document.querySelectorAll(".linkStarGestion");
3
4 deleteLink.forEach(function (link) {
5     link.addEventListener("click", function (event) {
6         event.preventDefault();
7         var idSneaker = this.dataset.id;
8         modal.style.display = "block";
9
10         document.getElementById("confirmDelete").onclick = function () {
11             window.location.href = "index.php?url=Admin&remove=yes&idSneaker=" + idSneaker;
12         }
13
14         document.getElementById("cancelDelete").onclick = function () {
15             modal.style.display = "none";
16         }
17     });
18 });
19

```

```
13     document.getElementById("cancelDelete").onclick = function () {
14         modal.style.display = "none";
15     }
16 });
17 });
18 });
19
20 window.onclick = function (event) {
21     if (event.target == modal) {
22         modal.style.display = "none";
23     }
24 }
```

src\View\js\auth.js

```
1 /**
2  * Bout de code issu d'une template - Dessine et anime les formes géométriques dans le
3  * login
4  * https://codepen.io/m2creates/pen/EEvGgW
5  */
6
7 // Initialisation de Paper.js sur la fenêtre et configuration du canvas HTML - paper.js
8 // sert à animer des vecteurs graphiques sur des canvas html
9 paper.install(window);
10 paper.setup(document.getElementById("canvas"));
11
12 // Déclaration des variables Paper.js
13 var canvasWidth,
14     canvasHeight,
15     canvasMiddleX,
16     canvasMiddleY;
17
18 // Groupe pour contenir les formes
19 var shapeGroup = new Group();
20
21 // Tableau pour stocker les positions des formes
22 var positionArray = [];
23
24 // Fonction pour obtenir les dimensions du canvas
25 function getCanvasBounds() {
26     // Obtention des dimensions actuelles du canvas
27     canvasWidth = view.size.width;
28     canvasHeight = view.size.height;
29     canvasMiddleX = canvasWidth / 2;
30     canvasMiddleY = canvasHeight / 2;
31
32     // Définition des positions des formes
33     var position1 = {
34         x: (canvasMiddleX / 2) + 100,
35         y: 100,
36     };
37
38     var position2 = {
39         x: 200,
40         y: canvasMiddleY,
41     };
42
43     var position3 = {
44         x: (canvasMiddleX - 50) + (canvasMiddleX / 2),
```

```

43     y: 150,
44   };
45
46   var position4 = {
47     x: 0,
48     y: canvasMiddleY + 100,
49   };
50
51   var position5 = {
52     x: canvasWidth - 130,
53     y: canvasHeight - 75,
54   };
55
56   var position6 = {
57     x: canvasMiddleX + 80,
58     y: canvasHeight - 50,
59   };
60
61   var position7 = {
62     x: canvasWidth + 60,
63     y: canvasMiddleY - 50,
64   };
65
66   var position8 = {
67     x: canvasMiddleX + 100,
68     y: canvasMiddleY + 100,
69   };
70
71   // Ajout des positions au tableau
72   positionArray = [position3, position2, position5, position4, position1, position6,
73   position7, position8];
74
75   /* ===== */
76   * Cr ation des formes
77   * ===== */
78   function initializeShapes() {
79     // Obtention des dimensions du canvas
80     getCanvasBounds();
81
82     // Donn es des chemins des formes
83     /*
84       Les lettres et les chiffres ont des significations sp cifiques :
85
86       'M' le point de d part du chemin
87       'L' une ligne qui se dessine absolument, en utilisant les coordonn es absolues
88       'z' la fermeture du chemin, en reliant le dernier point au premier
89
90       'M 100 100 L 200 100 L 200 200 L 100 200 Z', // Forme 1: exemple de carr 
91       'M 300 300 L 400 400 L 300 400 Z', // Forme 2: exemple de triangle
92     */
93     var shapePathData = [
94       'M0,0l64,219L29,343l535,30L478,371-133,4L0,0z',
95       'M0,65l16,138l196,107l270-2L470,0L337,4L0,65z',
96       'M331.9,3.6l-331,45l231,304l445-156l-76-196l-148,54L331.9,3.6z',
97       'M389,352l192-113l195-43l0,0l0,0L445,48l-80,1L122.7,0L0,275.2L162,297L389,352',
98       'M 50 100 L 300 150 L 550 50 L 750 300 L 500 250 L 300 450 L 50 100',
99       'M 700 350 L 500 350 L 700 500 L 400 400 L 200 450 L 250 350 L 100 300 L 150 50 L 350
100      100 L 250 150 L 450 150 L 400 50 L 550 150 L 350 250 L 650 150 L 650 50 L 700 150 L 600
250 L 750 250 L 650 300 L 700 350'
100    ];

```

```

101 // Création de chaque forme
102 for (var i = 0; i < shapePathData.length; i++) {
103     var headerShape = new Path({
104         strokeColor: 'rgba(36, 36, 36, 0.5)',
105         strokeWidth: 2,
106         parent: shapeGroup,
107     });
108     // Configuration des données du chemin SVG et positionnement de la forme
109     headerShape.pathData = shapePathData[i];
110     headerShape.scale(2);
111     headerShape.position = positionArray[i];
112 }
113 };
114
115 // Initialisation des formes
116 initializeShapes();
117
118 /* ===== */
119 * Animation
120 * ===== */
121 // Animation des formes en rotation
122 view.onFrame = function paperOnFrame(event) {
123     if (event.count % 4 === 0) {
124         for (var i = 0; i < shapeGroup.children.length; i++) {
125             if (i % 2 === 0) {
126                 shapeGroup.children[i].rotate(-0.1);
127             } else {
128                 shapeGroup.children[i].rotate(0.1);
129             }
130         }
131     }
132 };
133
134 // Réorganisation des formes lors du redimensionnement de la fenêtre
135 view.onResize = function paperOnResize() {
136     getCanvasBounds();
137
138     for (var i = 0; i < shapeGroup.children.length; i++) {
139         shapeGroup.children[i].position = positionArray[i];
140     }
141
142     // Ajustement de l'opacité en fonction de la largeur du canvas
143     if (canvasWidth < 700) {
144         shapeGroup.children[3].opacity = 0;
145         shapeGroup.children[2].opacity = 0;
146         shapeGroup.children[5].opacity = 0;
147     } else {
148         shapeGroup.children[3].opacity = 1;
149         shapeGroup.children[2].opacity = 1;
150         shapeGroup.children[5].opacity = 1;
151     }
152 }
153
154 // Sélection des éléments HTML pour les liens d'enregistrement et de connexion
155 var registerLink = document.querySelector('.registerLink');
156 var loginLink = document.querySelector('.loginLink');
157
158 // Animation de l'élément HTML lors du clic sur le lien d'enregistrement
159 registerLink.addEventListener('click', function () {
160

```

```

161  $('.animation1').animate({
162    'marginLeft': '50%' // Déplacer vers la droite
163  }, 0);
164 );
165
166 // Animation de l'élément HTML lors du clic sur le lien de connexion
167 loginLink.addEventListener('click', function () {
168   $('.animation1').animate({
169     'marginLeft': '0%' // Déplacer vers la gauche
170   }, 0);
171 });
172
173 // Récupérer le paramètre get pour afficher l'animation à gauche ou droite après le click
174 // des submit
175 // J'ai suivi cette documentation pour récupérer un paramètre en get en javascript
176 // https://www.sitepoint.com/get-url-parameters-with-javascript/
177 const urlParams = new URLSearchParams(window.location.search);
178
179 const slide = urlParams.get('slide')
180
181 if (slide === 'right') {
182   $('.animation1').css({
183     'marginLeft': '50%',
184     'transition': 'none'
185   });
186
187   setTimeout(() => {
188     $('.animation1').css('transition', 'ease all 0.3s 0.2s');
189   }, 100);
190 }

```

src\View\css\profile.css

```

1 .bgPreview{
2   margin-left:50%;
3   align-items: center;
4   justify-content: center;
5 }
6
7 .bgPreview div{
8   width: 50%;
9 }
10
11 .bgPreview h1{
12   padding-top: 15%;
13   padding-bottom: 0;
14   width: 100%;
15 }
16
17 .bgPreview .center{
18   width: 100%;
19   display: flex;
20   justify-content: center;
21 }
22
23 .bgPreview .titre{
24   width: 100%;
25 }

```

```
27 .bgPreview .titre h1{  
28     padding-top: 3%;  
29 }  
30  
31 .bgPreview .titre .marqueModele{  
32     text-align: left;  
33 }  
34  
35  
36 .bgPreview .titre .h1Prix{  
37     text-align: right;  
38 }  
39  
40 .bgPreview *{  
41     font-size: 2rem;  
42 }  
43  
44 .bgPreview .title{  
45     font-size: 4rem;  
46 }  
47  
48 .imgProfile {  
49     width: 100%;  
50     height: 32rem;  
51     border-radius: 15px;  
52     display: block;  
53 }  
54  
55 @media (max-width: 1980px) {  
56     .bgPreview div{  
57         width: 70%;  
58     }  
59  
60     .bgPreview *{  
61         font-size: 1.5rem;  
62     }  
63 }  
64 }
```

src\View\css\home.css

```
1 .center {  
2     text-align: center;  
3 }  
4  
5 .postedBy {  
6     display: flex;  
7     align-items: center;  
8     padding-top: 1em;  
9     border-top: 2px solid var(--mainColor);  
10 }  
11  
12 .avatar {  
13     outline: none;  
14     width: 70px;  
15     height: 70px;  
16     border-radius: 50%;  
17     object-fit: cover;  
18 }
```

```
19     box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
20     margin-right: 0;
21 }
22
23 .card {
24     margin: 2rem;
25     background-color: var(--backColor);
26     border-radius: 15px;
27     padding: 2rem;
28     display: inline-block;
29     vertical-align: top;
30     box-shadow: var(--boxShadow);
31     position: relative;
32     z-index: 1;
33 }
34
35 .logo {
36     position: absolute;
37     margin-top: 4%;
38     margin-left: 4%;
39     top: 0;
40     left: 0;
41     width: 15%;
42     z-index: 1;
43 }
44
45 .imgSneaker {
46     width: 100%;
47     height: 23rem;
48     border-radius: 15px;
49     display: block;
50 }
51
52 @media (min-width: 600px) {
53     .card {
54         width: calc(20% - 5rem);
55     }
56 }
57
58 .titre {
59
60     display: flex;
61     justify-content: space-between;
62 }
63
64 .title {
65     padding-top: 10%;
66     text-align: center;
67     font-size: 70px;
68 }
69
70 .filtres {
71     display: flex;
72     font-size: 30px;
73     justify-content: left;
74     margin-left: 2%;
75 }
76
77 .prix {
78     color: var(--mainColor);
```

```
79 }
80
81 .marques {
82   margin: .5%;
83   background-color: var(--backColor);
84   padding: calc(.5% - 1px) calc(.9% - 1px) calc(.5% - 1px) calc(.5% - 1px);
85   color: white
86 }
87
88 .marques:hover {
89   border-color: var(--mainColor);
90 }
91
92 .marques:focus {
93   outline: none;
94 }
95
96 .titre h1 {
97   padding-bottom: 0;
98   margin-bottom: 0;
99 }
100
101 .h1Size{
102   text-align: left;
103 }
104
105 .description{
106   margin-top: 5%;
107 }
108
109 .pagination{
110   width: 100%;
111   text-align: center;
112   justify-content: center;
113   font: var(--mainFont);
114   font-size: 3rem;
115   padding-top: 5%;
116   padding-bottom: 5%;
117 }
118
119 .pagination a{
120   padding: 2%;
121   border: 1px solid var(--mainColor);
122 }
123
124 .currentPage{
125   color: var(--mainColor);
126 }
127
128 .anchors{
129   display: column;
130   position: fixed;
131   top: 10%;
132   left: 3%;
133   background-color: var(--backColor);
134   border-radius: 15px;
135   padding: 2rem;
136   box-shadow: var(--boxShadow);
137   z-index: 1;
138 }
```

```
139
140 .anchors h1{
141   color: var(--mainColor);
142   font: var(--mainFont);
143   font-size: 2rem;
144   padding: 0;
145   margin-bottom: 5%;
146   width: 100%;
147   text-align: center;
148 }
149
150 @media (max-width: 1980px) {
151   .card {
152     width: calc(30% - 4rem);
153   }
154
155 }
156
157
158 @media (max-width: 1200px) {
159   .card {
160     width: calc(50% - 4rem);
161   }
162
163 }
164
165 @media (max-width: 600px) {
166
167   h1{
168     font-size: 1rem;
169   }
170
171   .card {
172     width: calc(100% - 2rem);
173     margin: 1rem;
174   }
175
176   .filtres {
177     justify-content: center;
178     margin-left: 0;
179   }
180
181   .marques {
182     font-size: 20px;
183   }
184
185   .center {
186     text-align: left;
187   }
188
189   .imgSneaker {
190     width: auto;
191   }
192
193   .titre {
194     flex-direction: column;
195     align-items: flex-start;
196   }
197 }
198 }
```

src\View\css\header.css

```
1 header {
2     width: 100%;
3     position: fixed;
4     z-index: 2;
5     background-color: var(--mainColor);
6     padding: 1em 0;
7 }
8
9 .headerNav {
10    width: 90%;
11    margin: 0 auto;
12 }
13
14 .navA {
15     font: var(--mainFont);
16     color: □ #ffffff;
17     text-decoration: none;
18     padding: 1em;
19     border-radius: 5px;
20     transition: background-color 0.3s ease;
21 }
22
23 .navA:hover {
24     background-color: rgba(255, 255, 255, 0.1);
25 }
26
27 nav {
28
29     display: flex;
30     justify-content: space-between;
31     align-items: center;
32 }
33
34 ul {
35     display: flex;
36     margin: 0;
37     list-style-type: none;
38     align-items: center;
39 }
40
41 li {
42     margin-right: 10px;
43     display: flex;
44     align-items: center;
45 }
46
47 .avatarHeader {
48     outline: none;
49     width: 5rem;
50     height: 5rem;
51     border-radius: 50%;
52     object-fit: cover;
53     box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
54     aspect-ratio: 1 / 1;
55 }
56
57 }
```

```
58 .logoHeader {  
59   border: none;  
60   width: auto;  
61   height: 70px;  
62   border-radius: 50%;  
63   object-fit: cover;  
64 }  
65  
66 .searchBar {  
67   border: none;  
68   border-bottom: 1px solid white;  
69   background-color: var(--mainColor);  
70   font: var(--mainFont);  
71   color: white  
72 }  
73  
74 .searchBar::placeholder {  
75   color: white  
76 }  
77  
78 .searchBar:focus {  
79   outline: none;  
80 }  
81  
82 .customMessages {  
83   color: white  
84   padding-left: 3%;  
85   font: var(--mainFont);  
86   font-size: 2rem;  
87   width: 100%;  
88   position: fixed;  
89   bottom: 0;  
90   z-index: 3;  
91 }  
92  
93 .green{  
94   background-color: var(--success);  
95 }  
96  
97 .red{  
98   background-color: var(--error);  
99 }  
100  
101 .searchBarLi {  
102   display: flex;  
103   align-items: center  
104 }  
105  
106 .linkProfile{  
107   display: flex;  
108   align-items: center;  
109   padding: 0;  
110 }  
111  
112 }  
113  
114 .linkProfile p{  
115   font: var(--secondFont);  
116   color: white  
117   margin-left: 10%;
```

```
118     margin-right: 10%;  
119 }  
120  
121 .nameUser{  
122     font: var(--mainFont);  
123     color: □ #ffffff;  
124     text-decoration: none;  
125     padding: 1em;  
126     border-radius: 5px;  
127 }
```

src\View\css\global.css

```
1 @import url('https://fonts.googleapis.com/css2?family=Outfit:wght@300;400;600&display=swap');  
2  
3 :root {  
4     --mainColor: rgb(168, 168, 225);  
5     --backColor: rgb(36, 36, 36);  
6  
7     --mainFont: normal normal 600 1.3em/1.6em 'Outfit', sans-serif;  
8     --secondaryFont: normal normal 400 1em/1em 'Outfit', sans-serif;  
9     --thirdFont: normal normal 300 1em/1.55em 'Outfit', sans-serif;  
10  
11    --error: ▢ #dc3545;  
12    --success: ▢ #00bb64;  
13  
14    /* https://getcssscan.com/css-box-shadow-examples - Box shadow n°10 */  
15    --boxShadow: rgba(0, 0, 0, 0.25) 0px 54px 55px,  
16        rgba(0, 0, 0, 0.12) 0px -12px 30px,  
17        rgba(0, 0, 0, 0.12) 0px 4px 6px,  
18        rgba(0, 0, 0, 0.17) 0px 12px 13px,  
19        rgba(0, 0, 0, 0.09) 0px -3px 5px;  
20  
21 }  
22  
23 html {  
24     box-sizing: border-box;  
25 }  
26  
27 *,  
28 *::before,  
29 *::after {  
30     box-sizing: inherit;  
31     margin: 0;  
32 }  
33  
34 body {  
35     font-size: 18px;  
36     background-color: var(--backColor);  
37     font-family: 'Outfit';  
38 }  
39  
40  
41 a {  
42     text-decoration: none;  
43     color: □ white  
44 }
```

```

45
46 a:hover {
47   opacity: .8;
48 }
49
50 h1,
51 input[type="submit"],
52 select {
53   font: var(--mainFont);
54 }
55
56 h1 {
57   color: white
58   padding: 1.2em 0;
59 }
60
61 h2 {
62   font: var(--secondaryFont);
63   color: white
64 }
65
66 p {
67   font: var(--thirdFont);
68   color: var(--mainColor);
69 }
70
71 span {
72   color: white
73 }
74
75 .customSuccess {
76   background-color: var(--success);
77 }
78
79 .customError {
80   background-color: var(--error);
81 }

```

src\View\css\gestion.css

```

1 .productcardGestion {
2   display: flex;
3   flex-direction: column;
4   flex-wrap: nowrap;
5   align-items: center;
6   justify-content: center;
7   align-content: center;
8   width: 100%;
9   min-height: 100vh;
10  position: relative;
11 }
12
13 .container {
14   width: 100%;
15   margin: 0 auto;
16   padding: 50px;
17   background-color: var(--backColor);
18   box-sizing: border-box;

```

```
19     box-shadow: var(--boxShadow);
20     display: flex;
21     justify-content: end;
22 }
23
24 .formulaire {
25     display: flex;
26     flex-direction: column;
27     align-items: center;
28     justify-content: center;
29     width: 100%;
30 }
31
32 .size,
33 .price,
34 .desc {
35     margin: 2%;
36     font: var(--mainFont);
37     padding: .5%;
38     width: 30%;
39 }
40
41 .size,
42 .price,
43 .desc {
44     color: □ white
45     background-color: transparent;
46     border: none;
47     border-bottom: 1px solid var(--mainColor);
48     width: 75%;
49 }
50
51 .size:focus,
52 .price:focus,
53 .desc:focus {
54     border-bottom: 1px solid white;
55     outline: none;
56 }
57
58 .brand,
59 .modele {
60     background-color: var(--backColor);
61     padding: calc(.5% - 1px) calc(.9% - 1px) calc(.5% - 1px) calc(.5% - 1px);
62     color: □ white
63 }
64
65
66 .brand:hover,
67 .modele:hover {
68     border-color: var(--mainColor);
69 }
70
71 .brand:focus,
72 .modele:focus {
73     outline: none;
74 }
75
76 .brand,
77 .modele,
78 .prixGestion {
```

```
79     font-size: 1.5rem;
80     margin: 1%;
81 }
82
83
84 .brandSizeLigne {
85     display: flex;
86     justify-content: space-around;
87 }
88
89 .cardGestion, .sellSneakerCard {
90     width: 25%;
91     margin: 2rem;
92     background-color: var(--backColor);
93     border-radius: 15px;
94     padding: 2rem;
95     display: inline-block;
96     vertical-align: top;
97     box-shadow: var(--boxShadow);
98     position: relative;
99 }
100
101 .centerGestion {
102     text-align: center;
103     padding-top: 10rem;
104 }
105
106 .titreTransaction {
107     padding-top: 10%;
108     text-align: center;
109     font-size: 70px;
110     padding: 0;
111 }
112
113 .price,
114 .size,
115 .desc {
116     color: var(--mainColor);
117     font-size: 1.5rem;
118 }
119
120
121 .brand {
122     margin-left: 0;
123 }
124
125 .prixGestion {
126     color: var(--mainColor);
127 }
128
129 .price {
130     color: var(--mainColor);
131     background-color: transparent;
132     border: none;
133     border-bottom: 1px solid var(--mainColor);
134     width: 75%;
135 }
136
137 .size {
138     width: 20%;
```

```
139 }
140
141 .descGestion {
142   margin-top: 2%;
143   text-align: left;
144 }
145
146 .desc {
147   text-align: left;
148   width: 100%;
149   padding: 0;
150   margin: 0;
151 }
152
153 .selectImg {
154   color: var(--mainColor);
155   font: var(--mainFont);
156   font-size: 1rem;
157 }
158
159 .submitTransaction {
160   width: 100%;
161   margin-bottom: 3%;
162
163   background-color: var(--mainColor);
164   color: white;
165   border-radius: 15px;
166   border: none;
167 }
168
169
170
171 .submitTransaction:hover {
172   cursor: pointer;
173   opacity: .9;
174 }
175
176 .errors {
177   margin-top: 1%;
178   color: var(--error);
179   font: var(--mainFont);
180   font-size: 100%;
181 }
182
183 .ligneErreurs {
184   display: flex;
185   width: 100%;
186   justify-content: space-around;
187   font-size: 1rem;
188 }
189
190 .brandDiv,
191 .modeleDiv,
192 .prixDiv {
193   width: 33%;
194   display: grid;
195   align-items: center;
196 }
197
198 .sizeDiv {
```

```
199     display: flex;
200     align-items: center;
201     justify-content: left;
202 }
203
204 .h1Size {
205     display: flex;
206     align-items: center;
207     z-index: 10;
208 }
209
210
211 .postedByGestion{
212     border: none;
213 }
214
215 .favoritesEmpty{
216     justify-content: center;
217     width: 100%;
218     padding-top: 0;
219 }
220
221 .noFavorites{
222     color: var(--mainColor);
223     font: var(--mainFont);
224     font-size: 3rem;
225 }
226
227 .cardGestionFavorites{
228     padding-top: 5%;
229     padding-bottom: 5%;
230     width: 50%;
231 }
232
233 .logo{
234     margin: 0;
235 }
236
237 .remove,
238 .invertedRemove {
239     background-size: contain;
240     background-repeat: no-repeat;
241     background-position: center;
242     width: 100%;
243     height: 100%;
244 }
245
246 .remove,
247 .invertedRemove:hover {
248     background-image: url("../img/removeSelected.png");
249 }
250
251 .remove:hover,
252 .invertedRemove {
253     background-image: url("../img/removeNotSelected.png");
254 }
255
256 .linkStar {
257     margin-left: 10%;
258     padding-top: 2%;
```

```
259     width: 20%;  
260 }  
261  
262 .starGestion{  
263     border-top: 1px solid var(--mainColor);  
264     display: flex;  
265     align-items: center;  
266 }  
267  
268 .cancelPurchase{  
269     margin-top: 5%;  
270     width: 100%;  
271 }  
272  
273 .acceptPurchase, .denyPurchase{  
274     width: 46%;  
275 }  
276  
277 .acceptPurchase{  
278     background-color: #00bb64;  
279     margin-right: 2%;  
280 }  
281  
282 .denyPurchase{  
283     background-color: #dc3545;  
284     margin-left: 2%;  
285 }  
286  
287 .acceptDeny{  
288     display: flex;  
289     margin-top: 5%;  
290     width: 100%;  
291 }  
292  
293 .meetPoint{  
294     margin-top: 3%;  
295     margin-bottom: 3%;  
296     text-align: left;  
297 }  
298  
299 .white{  
300     color: white  
301 }  
302  
303 .anchors{  
304     z-index: 10;  
305 }  
306  
307 .imgSellSneaker {  
308     width: 100%;  
309     height: auto;  
310     border-radius: 15px;  
311     display: block;  
312 }  
313  
314  
315  
316 @media (max-width: 1980px) {  
317     .card {  
318         width: calc(30% - 4rem);  
319     }  
320 }
```

```
319     }
320
321     .cardGestion{
322         width: 30%;
323     }
324
325     .cardGestionFavorites{
326         width: 70%;
327     }
328
329 }
330
331 @media (max-width: 600px) {
332     .container {
333         padding: 20px;
334     }
335     .cardGestion {
336         width: 90%;
337         margin: 1rem auto;
338     }
339     .brandSizeLigne {
340         flex-direction: column;
341         align-items: center;
342     }
343     .size, .price, .desc {
344         width: 100%;
345     }
346 }
347
348 @media (max-width: 1280px) {
349     .cardGestion {
350         width: 45%;
351     }
352 }
353
354 @media (max-width: 768px) {
355     .size, .price, .desc {
356         font-size: 14px;
357     }
358     .logo {
359         width: 30%;
360     }
361     .titreTransaction {
362         font-size: 2rem;
363     }
364     .submitTransaction {
365         width: 50%;
366     }
367     .starGestion {
368         flex-direction: column;
369         align-items: center;
370     }
371     .postedByGestion {
372         margin-top: 1rem;
373     }
374     .cancelPurchase {
375         margin-top: 2rem;
376     }
377     .acceptDeny {
378         flex-direction: column;
```

```
379     align-items: center;
380 }
381 .meetPoint {
382     margin-top: 2rem;
383 }
384 }
385 }
```

src\View\css\detail.css

```
1 .productCard {
2     display: flex;
3     flex-direction: column;
4     flex-wrap: nowrap;
5     align-items: center;
6     justify-content: center;
7     align-content: center;
8     width: 100%;
9     min-height: 100vh;
10    position: relative;
11    perspective: 100px;
12 }
13
14 .productCard .container {
15     width: 90%;
16     margin: 0 auto;
17     padding: 50px;
18     background-color: var(--backColor);
19     box-sizing: border-box;
20     box-shadow: var(--boxShadow);
21 }
22
23
24
25 .productCard .container .info .attribs .attrib .options {
26     display: flex;
27     flex-direction: row;
28     flex-wrap: wrap;
29     align-items: center;
30     justify-content: flex-start;
31     align-content: center;
32 }
33
34 .productCard .container .info .attribs .attrib .options .option {
35     display: flex;
36     flex-direction: row;
37     flex-wrap: nowrap;
38     align-items: center;
39     justify-content: center;
40     align-content: center;
41     width: 35px;
42     height: 35px;
43     margin: 10px 10px 0 0;
44     font-size: 14px;
45     font-weight: 600;
46     color: # darkgray
47     border-radius: 5px;
48     border: 1px solid darkgray;
49     cursor: pointer;
```

```
50 user-select: none;
51 transition: ease all 0.3s;
52 }
53
54
55
56 .productCard .container .colorLayer {
57 position: absolute;
58 top: 0;
59 right: 0;
60 width: 50%;
61 height: 100%;
62 background-color: var(--mainColor);
63 transform: rotateY(-8deg);
64 transform-origin: right;
65 perspective: 100px;
66 transition: ease all 0.3s 0.2s;
67 }
68
69 .productCard .container .preview {
70 display: flex;
71 flex-direction: row;
72 flex-wrap: nowrap;
73 align-items: center;
74 justify-content: center;
75 align-content: center;
76 position: absolute;
77 top: 0;
78 right: 0;
79 width: 50%;
80 height: 100%;
81 user-select: none;
82 overflow: hidden;
83 transition: ease all 0.3s;
84 }
85
86 .productCard .container .preview .brand {
87 position: absolute;
88 top: 0;
89 width: 90%;
90 height: 200px;
91 font-size: 230px;
92 text-align: center;
93 color: rgba(255, 255, 255, 0.2);
94 text-transform: uppercase;
95 overflow: hidden;
96 transition: ease all 0.3s;
97 }
98
99 .productCard .container .preview .img {
100 display: flex;
101 flex-direction: row;
102 flex-wrap: nowrap;
103 align-items: center;
104 justify-content: center;
105 align-content: center;
106 width: 100%;
107 height: 100%;
108 }
109 }
```

```
110 .productCard .container .preview .img img {
111   position: absolute;
112   top: 0;
113   width: 60%;
114   height: 100%;
115   object-fit: contain;
116   transform: translate(50%, -10%) rotate(-20deg);
117   opacity: 0;
118   pointer-events: none;
119   transition: ease all 0.3s;
120 }
121
122 .productCard .container .preview .img img.activ {
123   opacity: 1;
124   transform: none;
125 }
126
127 .price,
128 .desc {
129   color: var(--mainColor);
130 }
131
132 .namePrice {
133   margin-top: .5%;
134   font-size: 30px;
135   display: flex;
136   width: 50%;
137   justify-content: space-between;
138 }
139
140 .namePrice * {
141   padding-top: 0;
142   margin-right: 5%;
143   padding-bottom: 0;
144 }
145
146 .desc {
147   margin: 0;
148   padding: 0;
149 }
150
151 .submit {
152   float: right;
153   background-color: var(--mainColor);
154   color: □ white
155   padding: .5%;
156   width: 30%;
157   border-radius: 15px;
158   border: none;
159 }
160
161 .submit:hover {
162   cursor: pointer;
163   opacity: .9;
164 }
165
166 .size {
167   margin-bottom: .5%;
168   background-color: var(--backColor);
169   padding: calc(.5% - 1px) calc(.9% - 1px) calc(.5% - 1px) calc(.5% - 1px);
```

```
170  color: □ white
171 }
172
173 .size:hover {
174   border-color: var(--mainColor);
175 }
176
177 .size:focus {
178   outline: none;
179 }
180
181 .form {
182   width: 50%;
183 }
184
185 .back {
186   margin-top: 2%;
187   margin-right: .5%;
188   width: 100%;
189   float: right;
190   text-align: right;
191 }
192
193 .outOfStock {
194   font-size: 35px;
195   text-align: center;
196 }
197
198 .aOut {
199   text-align: center;
200 }
201
202 .msgOut {
203   text-align: center;
204   width: 50%;
205 }
206
207 .goBackLink {
208   margin-left: 0;
209   font: var(--mainFont);
210   color: □ white
211   text-decoration: none;
212   z-index: 2;
213 }
214
215 .fa {
216   margin-right: .3%;
217 }
218
219 .star,
220 .invertedStar {
221   background-size: contain;
222   background-repeat: no-repeat;
223   width: 100%;
224   height: 100%;
225 }
226
227 .star,
228 .invertedStar:hover {
229   background-image: url("../img/starNotSelected.png");
```

```
230 }
231
232 .star:hover,
233 .invertedStar {
234   background-image: url("../img/starSelected.png");
235 }
236
237 .linkStar {
238   margin-right: 2%;
239   width: 10%;
240   height: 3rem;
241 }
242 .attrib {
243   display: flex;
244   width: 100%;
245   align-items: center;
246 }
247
248 .errorNotConnected, .errorMeetPoint{
249   width: 70%;
250   color: #dc3545;
251   font: var(--mainFont);
252   font-size: 100%;
253 }
254
255 .customMessages {
256   color: white
257   padding-left: 3%;
258   font: var(--mainFont);
259   font-size: 2rem;
260   width: 100%;
261   position: fixed;
262   bottom: 0;
263   z-index: 3;
264 }
265
266 .customSuccess{
267   background-color: #00bb64;
268 }
269
270 .customError{
271   background-color: #dc3545;
272 }
273
274 .meetPoint{
275   float: left;
276 }
277
278 .moitie{
279   width: 70%;
280   justify-content: end;
281   display: flex;
282   align-items: center ;
283 }
284
285 .errorMeetPoint{
286   margin-top: 5%;
287 }
```

```
1 .productCard {
2   display: flex;
3   flex-direction: column;
4   flex-wrap: nowrap;
5   align-items: center;
6   justify-content: center;
7   align-content: center;
8   width: 100%;
9   min-height: 100vh;
10  position: relative;
11 }
12
13 .container {
14   width: 90%;
15   margin: 0 auto;
16   padding: 50px;
17   background-color: var(--backColor);
18   box-sizing: border-box;
19   box-shadow: var(--boxShadow);
20   display: flex;
21   justify-content: end;
22 }
23
24 .info{
25   width: 50%;
26 }
27
28 .formulaire {
29   display: flex;
30   flex-direction: column;
31   align-items: center;
32   justify-content: center;
33   width: 100%;
34 }
35
36
37 .titleForm{
38   padding: 0;
39   margin-bottom: .5%;
40   font-size: 55px;
41 }
42
43 .email, .password, .passwordRegister, .confirmPasswordRegister, .submitLogin, .name,
44 .surname{
45   margin: 2%;
46   font: var(--mainFont);
47   padding: .5%;
48   width: 30%;
49 }
50 .email, .password, .passwordRegister, .confirmPasswordRegister, .name, .surname{
51   color: □ white
52   background-color: transparent;
53   border:none;
54   border-bottom: 1px solid var(--mainColor);
55   width: 70%;
56 }
```

```
57
58 .email:focus, .password:focus, .name:focus, .passwordRegister:focus,
59 .confirmPasswordRegister:focus, .surname:focus{
60   border-bottom: 1px solid white;
61   outline: none;
62 }
63
64 .submitLogin{
65   margin-left: -.5%;
66   background-color: var(--mainColor);
67   color: □ white
68   border-radius: 15px;
69   border: none;
70 }
71 .submitLogin:hover{
72   cursor: pointer;
73   opacity: .9;
74 }
75
76 .submitDiv{
77   margin-top: 1%;
78   display: flex;
79   justify-content: space-between;
80   align-items: center;
81   width: 70%;
82 }
83
84 .logoLogin{
85   width: 20%;
86 }
87
88 .registerLink, .loginLink{
89   font: var(--mainFont);
90   color: var(--mainColor);
91 }
92
93 .goBackLink {
94   margin-top: 5%;
95   margin-left: 3%;
96   font: var(--mainFont);
97   color: □ white
98   text-decoration: none;
99   z-index: 2;
100 }
101
102 .canvas-back {
103   position: absolute;
104   top: 0;
105   left: 0;
106   width: 100%;
107   height: 100%;
108   z-index: 10;
109 }
110
111 .productCard .container .animation1 {
112   position: absolute;
113   top: 0;
114   left: 0;
115   width: 50%;
```

```
116  height: 100%;  
117  background-color: var(--mainColor);  
118  transform: rotateY(-8deg);  
119  transform-origin: left;  
120  perspective: 100px;  
121  transition: ease all 0.3s 0.2s;  
122 }  
123  
124 .ligne{  
125   width: 70%;  
126   display: flex;  
127   justify-content: space-around;  
128 }  
129  
130 .ligne .name, .errorNameRegister, .passwordRegister{  
131   margin: 0;  
132   margin-right: 2%;  
133 }  
134  
135 .ligne .email{  
136   margin: 0;  
137   margin-left: 2%;  
138 }  
139  
140 .email, .surname{  
141   margin-top: 0;  
142   margin-bottom: 0;  
143 }  
144  
145 .errorLogin, .errorEmailRegister, .errorNameRegister, .errorPasswordRegister,  
.errorSchoolRegister, .errorSurnameRegister{  
146   width: 70%;  
147   color: #dc3545;  
148   font: var(--mainFont);  
149   font-size: 100%;  
150 }  
151  
152 .ligne .email{  
153   margin-left: 0;  
154   margin-right: 2%;  
155 }  
156  
157 .ligne .password{  
158   margin-left: 2%;  
159   margin-right: 0;  
160 }  
161  
162 .school {  
163   background-color: var(--backColor);  
164   padding: calc(.5% - 1px) calc(.9% - 1px) calc(.5% - 1px) calc(.5% - 1px);  
165   color: white  
166 }  
167  
168 .school:hover {  
169   border-color: var(--mainColor);  
170 }  
171  
172 .school:focus {  
173   outline: none;  
174 }
```

```
175  
176 .school, .errorSurnameRegister{  
177   margin-left: 2%;  
178   margin-right: 2%;  
179 }  
180  
181 .errorSchoolRegister{  
182   margin-right: 2%;  
183 }  
184  
185 .imgDiv{  
186   display: flex;  
187   align-items: center;  
188   color: var(--mainColor);  
189   font: var(--mainFont);  
190   font-size: 1rem;  
191 }
```

src\View\css\admin.css

```
1 .imgUsers {  
2   height: 25rem;  
3   width: 100%;  
4 }  
5  
6 .cardUsers {  
7   width: 20%;  
8 }  
9  
10 .searchBarUsers {  
11   background-color: var(--backColor);  
12   margin-left: 6%;  
13   width: 20%;  
14 }  
15  
16 .gestionUsers {  
17   margin-top: 5%;  
18   display: flex;  
19   width: 100%;  
20   justify-content: space-around;  
21 }  
22  
23 .gestionUsers form input[type="submit"] {  
24   width: 100%;  
25 }  
26  
27 .gestionUsers form {  
28   width: 45%;  
29 }  
30  
31 .tableSchools,  
32 .tableMeetPoints {  
33   font: var(--mainFont);  
34   color: □ white  
35   width: 100%;  
36   border-collapse: collapse;  
37 }  
38 }  
39 }
```

```
40 .tableSchools tr:first-child,
41 .tableMeetPoints tr:first-child {
42     background-color: var(--mainColor);
43 }
44
45 .tableSchools th,
46 .tableMeetPoints th {
47     margin: 0;
48     padding: 0;
49     border: 2px solid var(--mainColor);
50 }
51
52 .tableSchools th:first-child {
53     width: 75%;
54 }
55
56 .tableMeetPoints th:last-child {
57     width: 30%;
58 }
59
60 .tableSchools .submitMeetPoints,
61 .submitMeetPoints .submit,
62 .tableMeetPoints .submitMeetPoints,
63 .tableMeetPoints .submitMeetPoints .submit {
64     text-align: center;
65     display: flex;
66     width: 100%;
67 }
68
69 .tableSchools th .submit,
70 .tableMeetPoints th .submit {
71     justify-content: center;
72     margin: 2%;
73 }
74
75 .addMeetPoint {
76     width: 100%;
77     display: flex;
78     align-items: center;
79     justify-content: space-between;
80     margin-bottom: 2%;
81 }
82
83 .errorMsg {
84     display: flex;
85     width: 100%;
86     justify-content: space-around;
87 }
88
89 .errorMsg * {
90
91     color: #dc3545;
92     font: var(--mainFont);
93     font-size: 100%;
94 }
95
96
97 .tableMeetPoints th .locked,
98 .tableSchools th .locked {
99     background-color: gray;
```

```
100 }
101
102 .editInfoMeetPoint {
103     width: 90%;
104 }
105
106
107 .locked:hover {
108     cursor: default;
109     opacity: 1;
110 }
111
112 .cardGestion {
113     width: 75%;
114 }
115
116 .tableMeetPoints th:nth-child(2){
117     width: 40%;
118 }
119
120 #confirmModal {
121     display: none;
122     position: fixed;
123     top: 0;
124     left: 0;
125     width: 100%;
126     height: 100%;
127     background-color: rgba(0, 0, 0, 0.5);
128     z-index: 1000;
129 }
130
131 .modal-content {
132     position: absolute;
133     top: 50%;
134     left: 50%;
135     transform: translate(-50%, -50%);
136     background-color: var(--backColor);
137     padding: 20px;
138     border-radius: 10px;
139 }
140
141 .modal-buttons {
142     text-align: center;
143     margin-top: 20px;
144 }
145
146 #confirmDelete {
147     background-color: var(--error);
148     color: white;
149     border: none;
150     padding: 10px 20px;
151     border-radius: 5px;
152     cursor: pointer;
153     margin-right: 10px;
154 }
155
156 #cancelDelete {
157     background-color: #6c757d;
158     color: white;
159     border: none;
```

```
160     padding: 10px 20px;
161     border-radius: 5px;
162     cursor: pointer;
163 }
164
165
166 @media (max-width: 1980px) {
167     .cardUsers {
168         width: calc(30% - 4rem);
169     }
170
171     .anchors {
172         top: 15%;
173     }
174
175     .title {
176         padding-top: 15%;
177     }
178
179     .tableSchools th .submitMeetPoints,
180     th .submitMeetPoints .submit,
181     .tableMeetPoints th .submitMeetPoints,
182     .tableMeetPoints th .submitMeetPoints .submit {
183         flex-wrap: wrap;
184     }
185 }
186
187 @media (max-width: 1200px) {
188     .card {
189         width: calc(50% - 4rem);
190     }
191 }
192
193 @media (max-width: 600px) {
194
195     h1 {
196         font-size: 1rem;
197     }
198
199     .card {
200         width: calc(100% - 2rem);
201         margin: 1rem;
202     }
203
204     .filtres {
205         justify-content: center;
206         margin-left: 0;
207     }
208
209     .marques {
210         font-size: 20px;
211     }
212
213     .center {
214         text-align: left;
215     }
216
217     .imgUsers {
218         width: auto;
219     }
```

```
220
221     .titre {
222         flex-direction: column;
223         align-items: flex-start;
224     }
225 }
```

src\View\css\about.css

```
1 body, html {
2     margin: 0;
3     padding: 0;
4     height: 100%;
5 }
6
7 .firstPart, .secondPart{
8     display: flex;
9     width: 100%;
10    height: 100%;
11 }
12
13 .firstPart {
14     justify-content: center;
15     align-items: center;
16 }
17
18 .secondPart{
19     background-color: var(--backColor);
20 }
21
22 .titleContainer {
23     position: relative;
24     text-align: center;
25 }
26
27 .titleHome {
28     z-index: 1;
29     font-size: 160px;
30     color: var(--mainColor);
31     text-shadow: 0 0 30px var(--backColor);
32 }
33
34 .circleTop, .circleBottom, .circleLeft, .circleRight{
35     position: absolute;
36     background-color: var(--mainColor);
37     display: flex;
38     justify-content: center;
39     align-items: center;
40     text-align: center;
41     width: 40%;
42     height: 65%;
43 }
44
45 .circleTop {
46     top: 0;
47     left: 0;
48     border-bottom-right-radius: 50%;
49 }
50 }
```

```
51 .circleBottom{  
52     bottom: 0px;  
53     right: 0px;  
54     border-top-left-radius: 50%;  
55 }  
56  
57 .circleRight {  
58     top: 100%;  
59     right: 0;  
60     border-bottom-left-radius: 50%;  
61 }  
62  
63 .circleLeft {  
64     top: 135%;  
65     left: 0;  
66     border-top-right-radius: 50%;  
67 }  
68  
69 header {  
70     box-shadow: var(--boxShadow);  
71 }  
72  
73 .imgDesc{  
74     width: 20%;  
75 }  
76  
77 .pfp{  
78     border-radius: 50%;  
79 }  
80  
81 .content{  
82     width: 80%;  
83     display: flex;  
84     flex-direction: column;  
85     text-align: justify;  
86     align-items: center;  
87 }  
88  
89 .titleDesc{  
90     font-size: 300%;  
91     text-shadow: 0 0 15px var(--backColor);  
92 }  
93  
94 .ligneDesc{  
95     display: flex;  
96     justify-content: center;  
97     align-items: center;  
98 }  
99  
100 .textDesc{  
101     font-weight: bold;  
102     margin-left: 2%;  
103     color: white  
104     font-size: 170%;  
105 }  
106  
107 }  
108  
109 @media (max-width: 1980px) {  
110     .titleDesc{
```

```

111     font-size: 200%;  

112 }  

113  

114 .textDesc{  

115     font-size: 130%;  

116 }  

117  

118 }
```

src\Model\ProfileModel.php

```

1 <?php  

2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  

3 Projet : SneakersMarket (TPI)  

4 Auteur : Léo Triano  

5 Desc. : Site d'achat-revente de sneakers  

6 Version : 1.0  

7 Date : Mai 2024  

8 Page : Modèle de la page profil  

9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  

10 class Profile  

11 {  

12  

13  

14     /**  

15      * Récupère toutes les écoles de la base de données  

16      *  

17      * @return array // tableau des écoles  

18      */  

19     function GetAllSchools() : array  

20     {  

21  

22         $arr = array();  

23  

24         $s = "SELECT * FROM SCHOOLS";  

25         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))  

;  

26         try {  

27             $statement->execute();  

28         } catch (PDOException $e) {  

29             echo 'Problème de lecture de la base de données: ' . $e->getMessage();  

30         }  

31  

32  

33  

34         while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {  

35             $u = new School($row["idSchool"], $row["nameSchool"]);  

36             array_push($arr, $u);  

37         }  

38  

39         return $arr;  

40     }  

41  

42     /**  

43      * Récupère l'utilisateur qui correspond à l'email donné  

44      *  

45      * @param string $email // l'email à vérifier  

46      * @return User|false // Retourne l'utilisateur s'il existe, false s'il n'existe pas  

47      */
```

```
48     function GetUserByEmail(string $email): User|false
49     {
50         $s = "SELECT * FROM USERS WHERE emailUser = :email";
51         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
52
53         try {
54             $statement->execute(array(':email' => $email));
55
56             $row = $statement->fetch(PDO::FETCH_ASSOC);
57
58             if ($row) {
59                 return new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
60 $row["emailUser"], $row["pwdUser"], $row["idSchool"], $row["imgUser"], $row["idRole"]);
61             }
62         } catch (PDOException $e) {
63             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
64         }
65
66         return false;
67     }
68
69     /**
70      * Vérifie si l'email donné est déjà utilisé par un utilisateur
71      *
72      * @param string $email // email à vérifier
73      * @return boolean // oui ou non
74      */
75     function CheckEmailAlreadyUsed(string $email): bool
76     {
77         $email = self::GetUserByEmail($email);
78
79         return ($email !== false);
80     }
81
82     /**
83      * Vérifie si l'école donnée existe
84      *
85      * @param integer $idSchool // l'id de l'école à vérifier
86      * @return boolean // oui ou non
87      */
88     function DoesSchoolExist(int $idSchool): bool
89     {
90         $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";
91         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
92
93         try {
94             $statement->bindParam(':idSchool', $idSchool);
95             $statement->execute();
96
97             $row = $statement->fetch(PDO::FETCH_ASSOC);
98
99             if ($row) {
100                 return true;
101             } else {
102                 // School not found
103                 return false;
104             }
105         } catch (PDOException $e) {
106             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
107             return false;
108         }
109     }
110 }
```

```

107     }
108 }
109
110 /**
111 * Vérifie si le mot de passe correspond avec le mot de passe hashé dans la base
112 *
113 * @param User $user // utilisateur à vérifier
114 * @return boolean
115 */
116 function CheckPassword(string $email, $password): bool
117 {
118     $Hashedpassword = self:: GetUserByEmail($email)->pwdUser;
119
120     return password_verify($password, $Hashedpassword);
121 }
122
123 /**
124 * Vérifie les informations de modification de compte
125 *
126 * @param User $user // Utilisateur à vérifie
127 * @param integer $idSchool // l'id de l'école
128 * @param array $img // la potentielle nouvelle image de l'utilisateur
129 * @return array // Tableau des potentielles erreurs
130 */
131 function ErrorMessagesRegister(User $user, int $idSchool, array $img): array
132 {
133     $expressionEmail = "/[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}/"; // Reprise
d'un ancien projet, trouvée à l'époque sur internet
134
135
136     // Variables
137     $errorMessages = [];
138
139
140     if ($img["type"] != "image/png" && $img["type"] != "image/jpeg" && $img["type"] !=
"image/jpg" && $img["name"] != "") {
141         $errorMessages["errorImg"] = Constants::MUST_BE_IMG;
142     }
143     if($img["size"] > Constants::IMG_SIZE){
144         $errorMessages["errorImg"] = Constants::ERROR_IMG_SIZE;
145     }
146
147     if(strlen($user->nameUser) > 15){
148         $errorMessages["errorName"] = Constants::MAX_CAR_NAME;
149     }
150
151     // Erreur si le nom est vide
152     if ($user->nameUser == "") {
153         $errorMessages["errorName"] = Constants::NOT_EMPTY;
154     }
155
156     if(strlen($user->surnameUser) > 10){
157         $errorMessages["errorSurname"] = Constants::MAX_CAR_SURNAME;
158     }
159
160     // Erreur si le prénom est vide
161     if ($user->surnameUser == "") {
162         $errorMessages["errorSurname"] = Constants::NOT_EMPTY;
163     }
164 }
```

```

165
166
167     if(!preg_match($expressionEmail, $user->emailUser)){
168         $errorMessages["errorEmailRegister"] = Constants::EMAIL;
169     }
170
171     // Erreur si l'email est vide
172     if ($user->emailUser == "") {
173         $errorMessages["errorEmailRegister"] = Constants::NOT_EMPTY;
174     }
175
176
177     // Erreur si le mot de passe est vide
178     if (!self::DoesSchoolExist($idSchool)) {
179         $errorMessages["errorSchoolRegister"] = Constants::SCHOOL_NOT_EXIST;
180     }
181
182     return $errorMessages;
183 }
184
185 /**
186 * Récupère l'école en fonction de son id donné
187 *
188 * @param int $idSchool // l'id de l'école
189 * @return School // L'école récupérée
190 */
191 function GetSchoolFromId(int $idSchool): School
192 {
193
194     $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";
195     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
196 ;
197     try {
198         $statement->bindParam(':idSchool', $idSchool, PDO::PARAM_INT);
199         $statement->execute();
200     } catch (PDOException $e) {
201         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
202     }
203
204     $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
205
206     $u = new School($row["idSchool"], $row["nameSchool"]);
207
208     return $u;
209 }
210
211 /**
212 * Met à jour le profil de l'utilisateur grâce aux nouvelles informations données
213 *
214 * @param User $user // Les nouvelles informations de l'utilisateur
215 * @param array $img // La potentielle nouvelle image
216 * @return boolean // L'opération a réussi ou non
217 */
218 function UpdateProfile(User $user, array $img) : bool
219 {
220     try {
221         if($img["name"] == ""){
222             $img = $_SESSION["imgUser"];
223         }
224     } else{

```

```

224         $img = Constants::ConvertImgTo64($img);
225     }
226     $nameUser = Constants::AntiXSS($user->nameUser);
227     $surnameUser = Constants::AntiXSS($user->surnameUser);
228     $emailUser = Constants::AntiXSS($user->emailUser);
229     $idSchool = Constants::AntiXSS($user->idSchool);
230
231     $s = "UPDATE USERS SET nameUser = :nameUser, surnameUser = :surnameUser,
232 emailUser = :emailUser, idSchool = :idSchool, imgUser = :imgUtilisateur
233 WHERE idUser = :idUser";
234     $statement = Database::prepare($s);
235
236     $statement->bindParam(':nameUser', $nameUser);
237     $statement->bindParam(':surnameUser', $surnameUser);
238     $statement->bindParam(':emailUser', $emailUser);
239     $statement->bindParam(':idSchool', $idSchool, PDO::PARAM_INT);
240     $statement->bindParam(':imgUtilisateur', $img);
241     $statement->bindParam(':idUser', $user->idUser);
242
243     $statement->execute();
244
245     self::ValuesSession(new User($_SESSION['idUser'], $nameUser, $surnameUser,
246 $emailUser, "", $idSchool, $img, $_SESSION['idRole']));
247
248     return true;
249
250 } catch (PDOException $e) {
251     echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
252     return false;
253 }
254
255 /**
256 * Met à jour les valeurs de la session après la mise à jour du profil
257 *
258 * @param User $user // Les informations à changer
259 * @return void
260 */
261
262 function ValuesSession(User $user): void
263 {
264     $_SESSION["idUser"] = $user->idUser;
265     $_SESSION["nameUser"] = $user->nameUser;
266     $_SESSION["surnameUser"] = $user->surnameUser;
267     $_SESSION["emailUser"] = $user->emailUser;
268     $_SESSION["idSchool"] = $user->idSchool;
269     $_SESSION["imgUser"] = $user->imgUser;
270     $_SESSION['idRole'] = $user->idRole;
271     $_SESSION["customMessage"] = "Welcome " . $user->surnameUser . " !";
272 }
273
274 function DeleteProfilePicture(){
275     try {
276
277         $s = "UPDATE USERS SET imgUser = NULL WHERE idUser = :idUser";
278         $statement = Database::prepare($s);
279
280         $statement->bindParam(':idUser', $_SESSION['idUser']);
281

```

```

282     $statement->execute();
283
284     $_SESSION['imgUser'] = "";
285     return true;
286
287
288 } catch (PDOException $e) {
289     echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
290     return false;
291 }
292 }
293 }
```

src\Model\ModifyUserModel.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modèle de la page modify user
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10
11 class ModifyUser
12 {
13
14     /**
15      * vérifie si l'utilisateur existe avec son id
16      *
17      * @param integer $idUser // id de l'utilisateur à vérifier
18      * @return boolean // il existe ou non
19      */
20     function DoesUserExist(int $idUser): bool
21     {
22         $s = "SELECT * FROM USERS WHERE idUser = :idUser";
23         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
24
25         try {
26             $statement->execute(array(':idUser' => $idUser));
27
28             $row = $statement->fetch(PDO::FETCH_ASSOC);
29
30             if ($row) {
31                 return true;
32             }
33         } catch (PDOException $e) {
34             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
35         }
36
37         return false;
38     }
39
40     /**
41      * Récupère toutes les écoles dans la base de données
42      *
43      * @return array // le tableau avec toutes les écoles
44      */

```

```

45     function GetAllSchools(): array
46     {
47
48         $arr = array();
49
50         $s = "SELECT * FROM SCHOOLS";
51         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
52     ;
53     try {
54         $statement->execute();
55     } catch (PDOException $e) {
56         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
57     }
58
59     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
60         $u = new School($row["idSchool"], $row["nameSchool"]);
61         array_push($arr, $u);
62     }
63
64     return $arr;
65 }
66 /**
67 * Récupère l'utilisateur avec l'id donné
68 *
69 * @param integer $idUser // l'id à vérifier
70 * @return User|false // L'utilisateur s'il existe, false s'il existe pas
71 */
72 function GetUserById(int $idUser): User|false
73 {
74     $s = "SELECT * FROM USERS WHERE idUser = :idUser";
75     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
76
77     try {
78         $statement->execute(array(':idUser' => $idUser));
79
80         $row = $statement->fetch(PDO::FETCH_ASSOC);
81
82         if ($row) {
83             return new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
84             $row["emailUser"], $row["pwdUser"], $row["idSchool"], $row["imgUser"], $row["idRole"]);
85         }
86     } catch (PDOException $e) {
87         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
88     }
89
90     return false;
91 }
92 /**
93 * Récupère l'utilisateur avec l'email donné
94 *
95 * @param string $emailUser // l'email à vérifier
96 * @return User|false // L'utilisateur s'il existe, false s'il existe pas
97 */
98 function GetUserByEmail(string $emailUser): User|false
99 {
100    $s = "SELECT * FROM USERS WHERE emailUser = :emailUser";
101    $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
102

```

```

103     try {
104         $statement->execute(array(':emailUser' => $emailUser));
105
106         $row = $statement->fetch(PDO::FETCH_ASSOC);
107
108         if ($row) {
109             return new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
110 $row["emailUser"], $row["pwdUser"], $row["idSchool"], $row["imgUser"], $row["idRole"]);
111         }
112     } catch (PDOException $e) {
113         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
114     }
115
116     return false;
117 }
118 /**
119 * Vérifie si l'email a déjà été utilisé grâce à l'id de l'utilisateur
120 *
121 * @param integer $id // l'id à vérifier
122 * @return boolean // Oui ou non
123 */
124 function CheckEmailAlreadyUsed(int $id): bool
125 {
126     $id = self::GetUserById($id);
127
128     return ($id !== false);
129 }
130
131 /**
132 * Vérifie si l'école existe grâce à l'id donné
133 *
134 * @param integer $idSchool // l'id de l'école à vérifier
135 * @return boolean // Oui ou non
136 */
137 function DoesSchoolExist(int $idSchool): bool
138 {
139     $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";
140     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
141
142     try {
143         $statement->bindParam(':idSchool', $idSchool);
144         $statement->execute();
145
146         $row = $statement->fetch(PDO::FETCH_ASSOC);
147
148         if ($row) {
149             return true;
150         } else {
151             // School not found
152             return false;
153         }
154     } catch (PDOException $e) {
155         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
156         return false;
157     }
158 }
159
160 /**
161 * Vérifie si le mot de passe correspond avec le mot de passe hashé dans la base

```

```

162  *
163  * @param User $user // utilisateur à vérifier
164  * @return boolean // oui ou non
165  */
166  function CheckPassword(string $email, $password): bool
167  {
168      $Hashedpassword = self::GetUserById($email)->pwdUser;
169
170      return password_verify($password, $Hashedpassword);
171  }
172
173 /**
174  * Vérifie si les informations données sont conformes
175  *
176  * @param User $user // informations de l'utilisateur à vérifier
177  * @param integer $idSchool // id de l'école
178  * @param array $img // potentielle nouvelle image
179  * @return array // Tableau des potentielles erreurs
180  */
181  function ErrorMessagesRegister(User $user, int $idSchool, array $img): array
182  {
183      $expressionEmail = "/[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}/"; // Reprise
d'un ancien projet, trouvée à l'époque sur internet
184
185
186      // Variables
187      $errorMessages = [];
188
189      if ($img["type"] != "image/png" && $img["type"] != "image/jpeg" && $img["type"] !=
"image/jpg" && $img["name"] != "") {
190          $errorMessages["errorImg"] = Constants::MUST_BE_IMG;
191      }
192      if ($img["size"] > Constants::IMG_SIZE) {
193          $errorMessages["errorImg"] = Constants::ERROR_IMG_SIZE;
194      }
195
196      // Erreur si le nom est vide
197      if ($user->nameUser == "") {
198          $errorMessages["errorName"] = Constants::NOT_EMPTY;
199      }
200
201      // Erreur si le prénom est vide
202      if ($user->surnameUser == "") {
203          $errorMessages["errorSurname"] = Constants::NOT_EMPTY;
204      }
205
206      if (!preg_match($expressionEmail, $user->emailUser)) {
207          $errorMessages["errorEmailRegister"] = Constants::EMAIL;
208      }
209
210      // Erreur si l'email est vide
211      if ($user->emailUser == "") {
212          $errorMessages["errorEmailRegister"] = Constants::NOT_EMPTY;
213      }
214
215
216      // Erreur si le mot de passe est vide
217      if (!self::DoesSchoolExist($idSchool)) {
218          $errorMessages["errorSchoolRegister"] = Constants::SCHOOL_NOT_EXIST;
219      }

```

```

220
221     return $errorMessages;
222 }
223
224 /**
225 * Récupère l'école grâce à son id
226 *
227 * @param integer $idSchool // id de l'école à vérifier
228 * @return School // l'école
229 */
230 function GetSchoolFromId(int $idSchool): School
231 {
232
233     $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";
234     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
235 ;
236     try {
237         $statement->bindParam(':idSchool', $idSchool);
238         $statement->execute();
239     } catch (PDOException $e) {
240         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
241     }
242
243     $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
244
245     $u = new School($row["idSchool"], $row["nameSchool"]);
246
247     return $u;
248 }
249 /**
250 * Met à jour le profil de l'utilisateur
251 *
252 * @param User $user // Nouvelles informations de l'utilisateur
253 * @param array $img // Potentielle nouvelle image de l'utilisateur
254 * @return boolean // L'opération a réussi ou non
255 */
256 function UpdateProfile(User $user, array $img): bool
257 {
258     try {
259         if ($img["name"] == "") {
260             $img = self::GetUserById($user->idUser)->imgUser;
261         } else {
262             $img = Constants::ConvertImgTo64($img);
263         }
264         $nameUser = Constants::AntiXSS($user->nameUser);
265         $surnameUser = Constants::AntiXSS($user->surnameUser);
266         $emailUser = Constants::AntiXSS($user->emailUser);
267         $idSchool = Constants::AntiXSS($user->idSchool);
268
269         $s = "UPDATE USERS SET nameUser = :nameUser, surnameUser = :surnameUser,
emailUser = :emailUser, idSchool = :idSchool, imgUser = :imgUtilisateur
270             WHERE idUser = :idUser";
271         $statement = Database::prepare($s);
272
273         $statement->bindParam(':nameUser', $nameUser);
274         $statement->bindParam(':surnameUser', $surnameUser);
275         $statement->bindParam(':emailUser', $emailUser);
276         $statement->bindParam(':idSchool', $idSchool, PDO::PARAM_INT);
277         $statement->bindParam(':imgUtilisateur', $img);

```

```

278     $statement->bindParam(':idUser', $user->idUser);
279
280     $statement->execute();
281     return true;
282
283
284     } catch (PDOException $e) {
285         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
286         return false;
287     }
288 }
289 }
```

src\Model\ManageUserModel.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : SneakersMarket (TPI)
5 Auteur : Léo Triano
6 Desc. : Site d'achat-revente de sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Modèle de la page manage user
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10
11 class ManageUser{
12     /**
13      * Récupère l'utilisateur avec l'id donné
14      *
15      * @param integer $idUser // l'id à vérifier
16      * @return User|false // L'utilisateur s'il existe, false s'il existe pas
17      */
18     function GetUserById(int $idUser): User|false
19     {
20         $s = "SELECT * FROM USERS WHERE idUser = :idUser";
21         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
22
23         try {
24             $statement->execute(array(':idUser' => $idUser));
25
26             $row = $statement->fetch(PDO::FETCH_ASSOC);
27
28             if ($row) {
29                 return new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
30 $row["emailUser"], $row["pwdUser"], $row["idSchool"], $row["imgUser"], $row["idRole"]);
31             }
32         } catch (PDOException $e) {
33             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
34         }
35
36         return false;
37     }
38
39     /**
40      * Récupère toutes les sneakers en vente
41      *
42      * @param integer $indexFirst // [Pagination] index du premier utilisateur
43      * @param integer $indexLast // [Pagination] index du dernier utilisateur
44      * @return array // Tableau de ViewTransaction
45 }
```

```

44     */
45     function GetAllShoesInSale(int $idSeller, int $indexFirst, int $indexLast): array
46     {
47         $arr = array();
48
49         $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress' AND
50 idSeller = :idSeller LIMIT :indexFirst, :indexLast";
51         $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
52 ;
53     try {
54         $statement->bindParam(':idSeller', $idSeller, PDO::PARAM_INT);
55         $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
56         $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
57
58         $statement->execute();
59     } catch (PDOException $e) {
60         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
61     }
62
63         while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
64             $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row[
65 priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],
66 $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
67             $acheteur = new User($row["idUser"], $row["nameBuyer"], $row["surnameBuyer"],
68 $row["emailBuyer"], "", 0, $row["imgBuyer"]);
69             $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],
70 ], $row["emailSeller"], "", 0, $row["imgSeller"]);
71
72             $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,
73 $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row[
74 dateHourMeetPlace"]));
75             array_push($arr, $u);
76         }
77
78         return $arr;
79     }
80
81 /**
82 * Compte le nombre de sneakers dans la base de données
83 *
84 * @return integer // Le nombre de sneakers
85 */
86
87 function CountSneakers($idUser): int
88 {
89     $s = "SELECT COUNT(*) FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress'
90 AND idSeller = :idUser";
91     $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
92 ;
93     try {
94         $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
95         $statement->execute();
96         $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
97         return (int) $count["COUNT(*)"];
98
99     } catch (PDOException $e) {
100         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
101         return 0;
102     }
103 }
104
105 /**

```

```
96     * Vérifie si une sneaker existe dans la base de données
97     *
98     * @param integer $idSneaker // l'id de la sneaker à vérifier
99     * @return boolean // Oui ou non
100    */
101   function DoesSneakerExist(int $idSneaker): bool
102   {
103
104
105     try {
106       $s = "SELECT * FROM TRANSACTIONS WHERE idSneaker = :idSneaker";
107       $statement = Database::prepare($s);
108
109       $statement->bindParam(':idSneaker', $idSneaker, PDO::PARAM_INT);
110
111       $statement->execute();
112
113       $row = $statement->fetch(PDO::FETCH_ASSOC);
114
115       if ($row) {
116         return true;
117       }
118     } catch (PDOException $e) {
119       echo 'Problème de lecture de la base de données: ' . $e->getMessage();
120     }
121
122     return false;
123   }
124
125 /**
126 * Supprime la sneaker en vente de la base
127 *
128 * @param integer $idSneaker // l'id de la sneaker
129 * @return void
130 */
131 function RemoveSneakerInSale(int $idSneaker): void
132 {
133   if (self::DoesSneakerExist($idSneaker)) {
134     try {
135       $s = "DELETE FROM TRANSACTIONS WHERE idSneaker = :idSneaker";
136       $statement = Database::prepare($s);
137
138       $statement->bindParam(':idSneaker', $idSneaker, PDO::PARAM_INT);
139
140       $statement->execute();
141
142     } catch (PDOException $e) {
143       echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
144     }
145   }
146 }
147
148 /**
149 * Récupère toutes les paires que l'utilisateur a vendu
150 *
151 * @return array // tableau des paires
152 */
153 function SneakersSold($idUser): array
154 {
155   $arr = array();
```

```

156
157     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'Sold' AND idSeller =
158 :idSeller";
159     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
160 ;
161     try {
162         $statement->bindParam(':idSeller', $idUser);
163
164         $statement->execute();
165     } catch (PDOException $e) {
166         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
167     }
168
169     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
170         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"],
171 $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],
172 $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
173         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"],
174 $row["emailBuyer"], "", 0, $row["imgBuyer"]);
175         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],
176 $row["emailSeller"], "", 0, $row["imgSeller"]);
177
178         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,
179 $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
180         array_push($arr, $u);
181     }
182
183     return $arr;
184 }
185
186 }
```

src\Model\headerModel.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : SneakersMarket (TPI)
5 Auteur : Léo Triano
6 Desc. : Site d'achat-revente de sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Modele du header
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
11 class Header
12 {
13
14     public $idUser;
15
16     function __construct()
17     {
18         // Initialisation de $idUser en fonction de $_SESSION
19         $this->idUser = isset($_SESSION['idUser']) ? $_SESSION['idUser'] : 0;
20     }
21
22     /**
23      * Sélectionne toutes les paires en fonction de la marque ou du modèle cherché
24      *
25      * @param string $input // marque ou modèle cherché
26     }
```

```

26     * @param integer $indexFirst // [Pagination] index du premier utilisateur
27     * @param integer $indexLast // [Pagination] index du dernier utilisateur
28     * @return array // tableau des paires
29   */
30   function SearchBar(string $input, int $indexFirst, int $indexLast) : array
31   {
32     $arr = array();
33
34     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE (nameBrand LIKE CONCAT('%', :input, '%'
35 ) OR nameModel LIKE CONCAT('%', :input, '%')) AND idSeller != :idSeller AND nameStatut LIKE
36     'In progress' AND nameVisibility LIKE 'Visible' LIMIT :indexFirst, :indexLast";
37     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
38     try {
39
40       $statement->bindParam(':input', $input);
41       $statement->bindParam(':idSeller', $this->idUser);
42       $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
43       $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
44
45       $statement->execute();
46     } catch (PDOException $e) {
47       echo 'Problème de lecture de la base de données: ' . $e->getMessage();
48     }
49
50     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
51
52       $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["
53 priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"], $row["
54 nameBrand"], $row["nameModel"], $row["nameVisibility"]);
55       $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"],
56       $row["emailBuyer"], "", 0, $row["imgBuyer"]);
57       $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"]
58 , $row["emailSeller"], "", 0, $row["imgSeller"]);
59
60       $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,
61       $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["
62 dateHourMeetPlace"]));
63       array_push($arr, $u);
64     }
65
66     return $arr;
67   }
68
69 /**
70  * Compte le nombre de paire trouvées via la recherche pour la pagination
71  *
72  * @param string $search // recherche
73  * @return integer // nombre de paires trouvées
74  */
75   function CountSneakers(string $search): int
76   {
77     $s = "SELECT COUNT(*) FROM VIEW_TRANSACTIONS WHERE nameBrand LIKE CONCAT('%',
78 :input, '%') OR nameModel LIKE CONCAT('%', :input, '%') AND idSeller != :idSeller AND
79     nameVisibility LIKE 'Visible' AND nameStatut LIKE 'In progress'";
80     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
81     try {
82       $statement->bindParam(':input', $search);
83       $statement->bindParam(':idSeller', $this->idUser);
84       $statement->execute();
85       $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
86       return (int) $count["COUNT(*)"];
87   }

```

```

78     } catch (PDOException $e) {
79         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
80         return 0;
81     }
82 }
83 }
84
85

```

src\Model\HomeModel.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modele de la page home
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10 class Home
11 {
12     public $idUser;
13
14     function __construct()
15     {
16         // Initialisation de $idUser en fonction de $_SESSION
17         $this->idUser = isset($_SESSION['idUser']) ? $_SESSION['idUser'] : 0;
18     }
19
20     /**
21      * Récupère toutes les transactions pour les cards
22      *
23      * @param integer $indexFirst // index de la première paire pour pagination
24      * @param integer $indexLast // index de la dernière paire pour pagination
25      *
26      * @return array // Tableau de paires
27     */
28     function GetAllSneakers(int $indexFirst, int $indexLast): array
29     {
30         $statutSeller = 1;
31         $nameVisibility = "Visible";
32         $arr = array();
33
34         $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress' AND
35         idSeller != :idSeller AND statutSeller = :statutSeller AND nameVisibility LIKE
36         :nameVisibility ORDER BY nameBrand LIMIT :indexFirst, :indexLast";
37         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
38
39         $statement->bindParam(':idSeller', $this->idUser, PDO::PARAM_INT);
40         $statement->bindParam(':statutSeller', $statutSeller, PDO::PARAM_INT);
41         $statement->bindParam(':nameVisibility', $nameVisibility, PDO::PARAM_STR);
42         $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
43         $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
44
45         $statement->execute();

```

```

46     } catch (PDOException $e) {
47         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
48     }
49
50     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
51         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["  
priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameStatut"],  
$row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
52         $acheteur = new User($row["idUser"], $row["nameBuyer"], $row["surnameBuyer"],  
$row["emailBuyer"], "", 0, $row["imgBuyer"]);
53         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],  
$row["emailSeller"], "", 0, $row["imgSeller"]);
54
55         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,  
$row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["  
dateHourMeetPlace"]));
56         array_push($arr, $u);
57     }
58
59     return $arr;
60 }
61
62 /**
63 * Récupère toutes les marques de la base
64 *
65 * @return array
66 */
67 function GetAllBrands(): array
68 {
69     $arr = array();
70
71     $s = "SELECT * FROM BRANDS";
72     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
73 ;
74     try {
75         $statement->execute();
76     } catch (PDOException $e) {
77         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
78     }
79
80     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
81         $u = new Brand($row["idBrand"], $row["nameBrand"]);
82         array_push($arr, $u);
83     }
84
85     return $arr;
86 }
87
88 /**
89 * Filtre les paires du listring par marque et / ou taille
90 *
91 * @param integer|null $idBrand // id de la marque de la paire
92 * @param integer|null $indexFirst // [Pagination] index de la première paire
93 * @param integer|null $indexLast // [Pagination] index de la dernière paire
94 * @param integer|null $idSize // id de la taille de la paire
95 * @return array // tableau des paires
96 */
97 function FilterByBrand(int|null $idBrand, int|null $indexFirst, int|null $indexLast,  
int|null $idSize) : array
98 {
99     $arr = array();

```

```

100     $query = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress' AND
101 idSeller != :idSeller AND statutSeller = 1";
102     $params = array(':idSeller' => $this->idUser);
103
104     if ($idBrand != 0 && $idBrand != -1 && $idBrand != null) {
105         $query .= " AND idBrand = :idBrand";
106         $params[':idBrand'] = $idBrand;
107     }
108
109     if ($idSize != 0 && $idSize != -1 && $idSize != null) {
110         $query .= " AND sizeSneaker = :idSize";
111         $params[':idSize'] = $idSize;
112     }
113
114     $query .= " LIMIT :indexFirst, :indexLast";
115     $params[':indexFirst'] = $indexFirst;
116     $params[':indexLast'] = $indexLast;
117
118     $statement = Database::prepare($query, array(PDO::ATTR_CURSOR =>
119 PDO::CURSOR_FWDONLY));
120
121     try {
122         foreach ($params as $param => $value) {
123             // Réglage d'erreur demandé à chatgpt, vu qu'il fallait impérativement
124             // spécifier que l'index first et last soient des int ça faisait une erreur
125             // https://chat.openai.com/share/d9c40bf0-5969-4163-807c-5f3303e8f1fd
126             $statement->bindValue($param, $value, is_int($value) ? PDO::PARAM_INT :
127 PDO::PARAM_STR);
128         }
129
130         $statement->execute();
131     } catch (PDOException $e) {
132         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
133     }
134
135     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
136         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row[
137 priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],
138 $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
139         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"],
140 $row["emailBuyer"], "", 0, $row["imgBuyer"]);
141         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],
142 $row["emailSeller"], "", 0, $row["imgSeller"]);
143
144         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,
145 $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
146         array_push($arr, $u);
147     }
148
149     return $arr;
150 }
151
152 /**
153 * Compte le nombre de paires, par marque si indiqué
154 *
155 * @param integer|null $idBrand // id de la marque s'il y en a une, sinon null
156 * @return integer // Le nombre de paires
157 */
158 function CountSneakers(int|null $idBrand): int
159 {
160 }
```

```

152     if ($idBrand != -1 && $idBrand != 0) {
153         $s = "SELECT COUNT(*) FROM TRANSACTIONS
154             INNER JOIN SNEAKERS ON TRANSACTIONS.idSneaker = SNEAKERS.idSneaker
155             INNER JOIN USERS ON TRANSACTIONS.idSeller = USERS.idUser
156             WHERE idTStatut = 1 AND SNEAKERS.idBrand = :idBrand AND idSeller != :idSeller
157             AND idUStatut = 1";
158         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR =>
159                                         PDO::CURSOR_FWDONLY));
160         try {
161             $statement->bindParam(':idSeller', $this->idUser);
162             $statement->bindParam(':idBrand', $idBrand);
163             $statement->execute();
164             $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
165             return (int) $count["COUNT(*)"];
166         } catch (PDOException $e) {
167             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
168             return 0;
169         }
170     } else {
171         $s = "SELECT COUNT(*) FROM TRANSACTIONS INNER JOIN USERS ON
172             TRANSACTIONS.idSeller = USERS.idUser WHERE idTStatut = 1 AND idSeller != :idSeller AND
173             idUStatut = 1";
174         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR =>
175                                         PDO::CURSOR_FWDONLY));
176         try {
177             $statement->bindParam(':idSeller', $this->idUser);
178             $statement->execute();
179             $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
180             return (int) $count["COUNT(*)"];
181         } catch (PDOException $e) {
182             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
183             return 0;
184         }
185     }
186     /**
187      * Récupère le nombre de tailles disponibles des paires
188      *
189      * @return array // Le tableau des paires contenant uniquement leur tailles
190      */
191     function GetAllSizes() : array
192     {
193         $arr = array();
194
195         $s = "SELECT sizeSneaker FROM VIEW_TRANSACTIONS WHERE idSeller != :idUser AND
196             statutSeller = 1 AND nameStatut LIKE 'in progress' GROUP BY sizeSneaker ORDER BY
197             sizeSneaker";
198         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
199     ;
200         try {
201             $statement->bindParam(":idUser", $this->idUser);
202             $statement->execute();
203         } catch (PDOException $e) {
204             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
205         }
206
207         while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
208             $u = new Sneaker(0, $row["sizeSneaker"]);
209             array_push($arr, $u);

```

```

206     }
207
208     return $arr;
209 }
210 }
```

src\Model\GestionModel.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modele de la page gestion
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10
11 class Gestion
12 {
13     /**
14      * Vérifie si les données pour créer une nouvelle paire sont conformes
15      *
16      * @param integer $size // taille de la paire
17      * @param float|integer $price // prix de la paire
18      * @param array $img // image de la paire
19      * @param string $desc // description de la paire
20      * @param string $model // modèle de la paire
21      * @param integer $idBrand // id de la marque de la paire
22      * @return array // tableau des potentielles erreurs
23      */
24     function CheckErrorsCreateTransaction(int|string $size, float|int|string $price, array
25     $img, string $desc, string $model, int $idBrand) : array
26     {
27         $errorMessages = [];
28
29         if (strlen($desc) > Constants::MAX_DESC_CAR) {
30             $errorMessages["errorDesc"] = Constants::DESC_MAX_CAR;
31         }
32         if (!preg_match("/[0-9]/", $price)) {
33             $errorMessages["errorPrice"] = Constants::MUST_BE_NUMBER;
34         }
35
36         if ($img["type"] != "image/png" && $img["type"] != "image/jpeg" && $img["type"] !=
37         "image/jpg") {
38             $errorMessages["errorImg"] = Constants::MUST_BE_IMG;
39         }
40
41         if($img["size"] > Constants::IMG_SIZE){
42             $errorMessages["errorImg"] = Constants::ERROR_IMG_SIZE;
43         }
44
45         if ($size == "") {
46             $errorMessages["errorSize"] = Constants::NOT_EMPTY;
47         }
48         if ($size < Constants::SIZE_MIN || $size > Constants::SIZE_MAX) {
49             $errorMessages["errorSize"] = Constants::SIZE_MIN_MAX;
50         }
51         if ($price == "") {
```

```

50     $erorrMessages["errorPrice"] = Constants::NOT_EMPTY;
51 }
52 if ($price < Constants::PRICE_MIN) {
53     $erorrMessages["errorPrice"] = Constants::PRICE_MIN_MAX;
54 }
55 if (empty($img["name"])) {
56     $erorrMessages["errorImg"] = Constants::NOT_EMPTY;
57 }
58 if ($desc == "") {
59     $erorrMessages["errorDesc"] = Constants::NOT_EMPTY;
60 }
61 if ($model == "") {
62     $erorrMessages["errorModel"] = Constants::NOT_EMPTY;
63 }
64 if ($idBrand == -1) {
65     $erorrMessages["errorBrand"] = Constants::SELECT_BRAND;
66 }

68     return $erorrMessages;
69 }

70
71
72
73
74 /**
75 * Récupère toutes les marques de la base
76 *
77 * @return array // tableau des marques
78 */
79 function GetAllBrands(): array
80 {
81     $arr = array();

82
83     $s = "SELECT * FROM BRANDS ORDER BY nameBrand ASC";
84     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
85 ;
86     try {
87         $statement->execute();
88     } catch (PDOException $e) {
89         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
90     }

91     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
92         $u = new Brand($row["idBrand"], $row["nameBrand"]);
93         array_push($arr, $u);
94     }

95     return $arr;
96 }
97
98
99 /**
100 * Récupère tous les modèles de la base
101 *
102 * @return array // tableau des modèles
103 */
104 function GetAllModeles(): array
105 {
106     $arr = array();

107     $s = "SELECT * FROM MODELS";

```

```

109     ;
110
111     $statement = Database:::prepare($s, array(PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY));
112
113     try {
114         $statement->execute();
115     } catch (PDOException $e) {
116         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
117     }
118
119     while ($row = $statement->fetch(PDO:::FETCH_ASSOC, PDO:::FETCH_ORI_NEXT)) {
120         $u = new Modele($row["idModel"], $row["nameModel"]);
121         array_push($arr, $u);
122     }
123
124     return $arr;
125 }
126
127 /**
128 * Insère la paire dans la base de données
129 *
130 * @param integer $size // taille de la paire
131 * @param float|integer $price // prix de la paire
132 * @param array $img // image de la paire
133 * @param string $desc // description de la paire
134 * @param integer $idModel // id du modèle de la paire
135 * @param integer $idBrand // id de la marque de la paire
136 * @return void
137 */
138 function InsertSneaker(int $size, float|int $price, array $img, string $desc, int
139 $idModel, int $idBrand, int $statut): void
140 {
141     try {
142         $size = Constants:::AntiXSS($size);
143         $price = Constants:::AntiXSS($price);
144         $desc = Constants:::AntiXSS($desc);
145         $idModel = Constants:::AntiXSS($idModel);
146         $idBrand = Constants:::AntiXSS($idBrand);
147         $img = Constants:::ConvertImgTo64($img);
148
149         $s = "INSERT INTO SNEAKERS(sizeSneaker, priceSneaker, imgSneaker,
150 descriptionSneaker, idModel, idBrand, idVisibility)
151 VALUES(:size, :price, :img, :desc, :idModel, :idBrand, :statut)";
152         $statement = Database:::prepare($s);
153
154         $statement->bindParam(':size', $size);
155         $statement->bindParam(':price', $price);
156         $statement->bindParam(':img', $img);
157         $statement->bindParam('desc', $desc);
158         $statement->bindParam('idModel', $idModel);
159         $statement->bindParam('idBrand', $idBrand);
160         $statement->bindParam('statut', $statut);
161
162         $statement->execute();
163
164     } catch (PDOException $e) {
165         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
166     }
167
168 /**
169 * Vérifie si le modèle existe à partir de son nom

```

```

167     *
168     * @param string $nameModel // nom du modèle
169     * @return Modele|false // le modèle ou false s'il n'existe pas
170     */
171     function DoesModeleExist(string $nameModel) : Modele|false
172     {
173
174         $s = "SELECT * FROM MODELS WHERE nameModel = :nameModel";
175         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
176
177         try {
178             $statement->execute(array(':nameModel' => $nameModel));
179
180             $row = $statement->fetch(PDO::FETCH_ASSOC);
181
182             if ($row) {
183                 return new Modele($row["idModel"], $row["nameModel"]);
184             }
185         } catch (PDOException $e) {
186             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
187         }
188
189         return false;
190     }
191
192
193     /**
194     * Insère le modèle dans la base de données
195     *
196     * @param string $nameModel // nom du modèle
197     * @return boolean // l'opération a réussi ou non
198     */
199     function InsertModele(string $nameModel) : bool
200     {
201         if (!self::DoesModeleExist($nameModel)) {
202             try {
203                 $nameModel = Constants::AntiXSS($nameModel);
204                 $s = "INSERT INTO MODELS(nameModel) VALUES(:nameModel)";
205                 $statement = Database::prepare($s);
206
207                 $statement->bindParam(':nameModel', $nameModel);
208
209                 $statement->execute();
210                 return true;
211
212             } catch (PDOException $e) {
213                 echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
214                 return false;
215             }
216         }
217     }
218     else{
219         return false;
220     }
221 }
222
223
224     /**
225     * Insère la paire et la transaction affiliée à cette dernière en base de données
226     *

```

```

227     * @param integer $size // taille de la paire
228     * @param integer $price // prix de la paire
229     * @param array $img // image de la paire
230     * @param string $desc // description de la paire
231     * @param string $nameModel // nom du modèle de la paire
232     * @param integer $idBrand // id de la marque de la paire
233     * @return void
234     */
235     function CreateTransaction(int $size, int $price, array $img, string $desc, string
236     $nameModel, int $idBrand, int $statut): void
237     {
238         try {
239             // Début de la transaction
240             Database::beginTransaction();
241
242             // Si l'insertion du modèle se fait correctement
243             if(self::InsertModele($nameModel) !== false){
244                 // Récupère l'id du dernier modèle inséré
245                 $idModel = Database::lastInsertId();
246             }
247             else{
248                 // Sinon, récupère l'id du modèle
249                 $idModel = self::DoesModeleExist($nameModel)->idModel;
250             }
251
252             // Insère la sneaker en base
253             self::InsertSneaker($size, $price, $img, $desc, $idModel, $idBrand, $statut);
254
255             $idSneaker = Database::lastInsertId();
256
257             // Insère la transaction
258             $s = "INSERT INTO TRANSACTIONS(idSneaker, idSeller) VALUES(:idSneaker,
259             :idSeller)";
260             $statement = Database::prepare($s);
261
262             $statement->bindParam(':idSneaker', $idSneaker);
263             $statement->bindParam(':idSeller', $_SESSION['idUser']);
264
265             $statement->execute();
266             Database::commit();
267         } catch (PDOException $e) {
268             // Annule la transaction
269             Database::rollBack();
270             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
271         }
272     }
273
274     /**
275      * Récupère toutes les paires en vente
276      *
277      * @return array // tableau des paires
278      */
279     function GetAllShoesInSale(): array
280     {
281         $arr = array();
282
283         $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress' AND
idSeller = :idSeller";
284         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
;

```

```

284     try {
285         $statement->bindParam(':idSeller', $_SESSION['idUser']);
286
287         $statement->execute();
288     } catch (PDOException $e) {
289         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
290     }
291
292     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
293         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"], $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
294         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"], $row["emailBuyer"], "", 0, $row["imgBuyer"]);
295         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"], $row["emailSeller"], "", 0, $row["imgSeller"]);
296
297         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur, $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
298         array_push($arr, $u);
299     }
300
301     return $arr;
302 }
303
304 /**
305 * Vérifie si la paire existe dans la base
306 *
307 * @param integer $idSneaker // id de la paire à vérifier
308 * @return boolean // oui ou non
309 */
310 function DoesSneakerExist(int $idSneaker) : bool
311 {
312     $s = "SELECT * FROM SNEAKERS WHERE idSneaker = :idSneaker";
313     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
314
315     try {
316         $statement->execute(array(':idSneaker' => $idSneaker));
317
318         $row = $statement->fetch(PDO::FETCH_ASSOC);
319
320         if ($row) {
321             return true;
322         }
323     } catch (PDOException $e) {
324         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
325         return false;
326     }
327
328     return false;
329 }
330
331 /**
332 * Vérifie si l'utilisateur actuel n'est pas bloqué ou non
333 *
334 * @return boolean // Oui ou non
335 */
336 function IsUserNotBlocked() : bool
337 {
338     $s = "SELECT * FROM USERS WHERE idUser = :idUser AND idUStatut = 1";

```

```

339     $statement = Database:::prepare($s, [PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY]);
340
341     try {
342         $statement->execute(array(':idUser' => $_SESSION['idUser']));
343
344         $row = $statement->fetch(PDO:::FETCH_ASSOC);
345
346         if ($row) {
347             return true;
348         }
349     } catch (PDOException $e) {
350         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
351         return false;
352     }
353
354     return false;
355 }
356
357 /**
358 * Supprime une paire de la base de données
359 *
360 * @param integer $idSneaker // id de la paire à supprimer
361 * @return void
362 */
363 function RemoveSneakerInSale(int $idSneaker)
364 {
365     $idSneaker = Constants:::AntiXSS($idSneaker);
366     if (self:::DoesSneakerExist($idSneaker)) {
367         try {
368             $s = "DELETE FROM TRANSACTIONS WHERE idSneaker = :idSneaker AND idSeller = :idUser";
369             $statement = Database:::prepare($s);
370
371             $statement->bindParam(':idSneaker', $idSneaker);
372             $statement->bindParam(':idUser', $_SESSION['idUser']);
373
374             $statement->execute();
375
376         } catch (PDOException $e) {
377             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
378         }
379     }
380 }
381
382 /**
383 * Récupère toutes les paires que l'utilisateur a acheté
384 *
385 * @return array // tableau des paires
386 */
387 function GetAllShoesPurchased(): array
388 {
389     $arr = array();
390
391     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'Sold' AND idBuyer = :idUser";
392     $statement = Database:::prepare($s, array(PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY));
393 ;
394     try {
395         $statement->bindParam(':idUser', $_SESSION['idUser']);
396
397         $statement->execute();

```

```

397     } catch (PDOException $e) {
398         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
399     }
400
401     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
402         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["  

priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],  

$row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
403         $acheteur = new User($row["idUser"], $row["nameBuyer"], $row["surnameBuyer"],  

$row["emailBuyer"], "", 0, $row["imgBuyer"]);
404         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],  

$row["emailSeller"], "", 0, $row["imgSeller"]);
405
406         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,  

$row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["  

dateHourMeetPlace"]));
407         array_push($arr, $u);
408     }
409
410     return $arr;
411 }
412
413 /**
414 * Récupère toutes les paires que l'utilisateur a réservé
415 *
416 * @return array // tableau des paires
417 */
418 function GetAllShoesReserved(): array
419 {
420     $arr = array();
421
422     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'Reserved' AND idBuyer  

= :idUser";
423     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
424 ;
425     try {
426         $statement->bindParam(':idUser', $_SESSION['idUser']);
427
428         $statement->execute();
429     } catch (PDOException $e) {
430         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
431     }
432
433     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
434         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["  

priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],  

$row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
435         $acheteur = new User($row["idUser"], $row["nameBuyer"], $row["surnameBuyer"],  

$row["emailBuyer"], "", 0, $row["imgBuyer"]);
436         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],  

$row["emailSeller"], "", 0, $row["imgSeller"]);
437
438         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,  

$row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["  

dateHourMeetPlace"]));
439         array_push($arr, $u);
440     }
441
442     return $arr;
443 }
444
445 /**
446 * Annule la réservation de l'utilisateur

```

```

446     *
447     * @param integer $idSneaker // id de la paire à annuler
448     * @return boolean // l'opération a réussi ou non
449     */
450     function CancelReserve(int $idSneaker) : bool
451     {
452         try {
453             $s = "UPDATE TRANSACTIONS SET idBuyer = NULL, idTStatut = 1 WHERE idSneaker =
454 :idSneaker";
455             $statement = Database::prepare($s);
456
457             $statement->bindParam(':idSneaker', $idSneaker);
458
459             $statement->execute();
460
461             return true;
462         } catch (PDOException $e) {
463             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
464             return false;
465         }
466     }
467
468 /**
469 * Récupère toutes les paires que l'utilisateur vend et son réservées
470 *
471 * @return array // tableau des paires
472 */
473 function SneakersReserved(): array
474 {
475     $arr = array();
476
477     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'Reserved' AND
478 idSeller = :idSeller";
479     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
480 ;
481     try {
482         $statement->bindParam(':idSeller', $_SESSION['idUser']);
483
484         $statement->execute();
485     } catch (PDOException $e) {
486         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
487     }
488
489     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
490         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row[
491 priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],
492 $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
493         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"],
494 $row["emailBuyer"], "", 0, $row["imgBuyer"]);
495         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"],
496 $row["emailSeller"], "", 0, $row["imgSeller"]);
497
498         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur,
499 $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row[
500 dateHourMeetPlace"]));
501         array_push($arr, $u);
502     }
503
504     return $arr;
505 }

```

```

499  /**
500   * Accepte la réservation d'un utilisateur et finalise la transaction
501   *
502   * @param int $idSneaker // id de la paire
503   * @return boolean // l'opération a réussi ou non
504   */
505  function AcceptReservation(int $idSneaker) : bool
506  {
507      try {
508          $s = "UPDATE TRANSACTIONS SET idTStatut = 2 WHERE idSneaker = :idSneaker";
509          $statement = Database::prepare($s);
510
511          $statement->bindParam(':idSneaker', $idSneaker);
512
513          $statement->execute();
514
515          return true;
516
517      } catch (PDOException $e) {
518          echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
519          return false;
520      }
521  }
522
523 /**
524  * Supprime les favoris d'une paire si cette dernière a été vendue
525  *
526  * @param integer $idSneaker // id de la paire
527  * @return boolean // l'opération a réussi ou non
528  */
529  function RemoveFavoritesSneakerSold(int $idSneaker) : bool
530  {
531      try {
532          $s = "DELETE FROM FAVORITES WHERE idSneaker = :idSneaker";
533          $statement = Database::prepare($s);
534
535          $statement->bindParam(':idSneaker', $idSneaker);
536
537          $statement->execute();
538
539          return true;
540
541      } catch (PDOException $e) {
542          echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
543          return false;
544      }
545  }
546
547 /**
548  * Refuse la demande d'achat d'un utilisateur
549  *
550  * @param integer $idSneaker // l'id de la paire
551  * @return boolean // l'opération a réussi ou non
552  */
553  function DenyReservation(int $idSneaker) : bool
554  {
555      try {
556          $s = "UPDATE TRANSACTIONS SET idTStatut = 1, idBuyer = NULL, idMeetPoint =
NULL, idDateMeetPlace = NULL WHERE idSneaker = :idSneaker";
557          $statement = Database::prepare($s);

```

```

558
559     $statement->bindParam(':idSneaker', $idSneaker);
560
561     $statement->execute();
562
563     return true;
564
565 } catch (PDOException $e) {
566     echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
567     return false;
568 }
569 }
570
571 /**
572 * Récupère toutes les paires que l'utilisateur a vendu
573 *
574 * @return array // tableau des paires
575 */
576 function SneakersSold(): array
577 {
578     $arr = array();
579
580     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'Sold' AND idSeller = :idSeller";
581     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
582 ;
583     try {
584         $statement->bindParam(':idSeller', $_SESSION['idUser']);
585
586         $statement->execute();
587     } catch (PDOException $e) {
588         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
589     }
590
591     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
592         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"], $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
593         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"], $row["emailBuyer"], "", 0, $row["imgBuyer"]);
594         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"], $row["emailSeller"], "", 0, $row["imgSeller"]);
595
596         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur, $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
597         array_push($arr, $u);
598     }
599
600     return $arr;
601 }
602
603 function IsSneakerInvisible($idSneaker){
604     $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameVisibility LIKE 'hidden' AND idSneaker = :idSneaker";
605     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
606
607     try {
608         $statement->execute(array(':idSneaker' => $idSneaker));
609
610         $row = $statement->fetch(PDO::FETCH_ASSOC);

```

```

611     if ($row) {
612         return true;
613     }
614 } catch (PDOException $e) {
615     echo 'Problème de lecture de la base de données: ' . $e->getMessage();
616     return false;
617 }
618
619     return false;
620 }
621
622 function SetSneakerToVisible($idSneaker){
623     try {
624         $s = "UPDATE SNEAKERS SET idVisibility = 1 WHERE idSneaker = :idSneaker";
625         $statement = Database::prepare($s);
626
627         $statement->bindParam(':idSneaker', $idSneaker);
628
629         $statement->execute();
630
631         return true;
632
633     } catch (PDOException $e) {
634         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
635         return false;
636     }
637 }
638 }
```

src\Model\FavoritesModel.php

```

1 <?php
2 /**
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Trianos
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modele de la page favoris
9 */
10 class Favorites
11 {
12     /**
13     * Sélectionne les paires favorites de l'utilisateur
14     *
15     * @return array // Le tableau des paires
16     */
17     function UserFavList() : array
18     {
19         $arr = array();
20
21         $s = "SELECT
22             DISTINCT VIEW_TRANSACTIONS.*, FAVORITES.*, SNEAKERS.*, MODELS.*, BRANDS.*
23             FROM VIEW_TRANSACTIONS
24             JOIN FAVORITES ON VIEW_TRANSACTIONS.idSneaker = FAVORITES.idSneaker
25             JOIN SNEAKERS ON FAVORITES.idSneaker = SNEAKERS.idSneaker
26             JOIN MODELS ON SNEAKERS.idModel = MODELS.idModel
27             JOIN BRANDS ON SNEAKERS.idBrand = BRANDS.idBrand
28             WHERE FAVORITES.idUser = :idUser

```

```

29     GROUP BY VIEW_TRANSACTIONS.idSneaker;";
30
31     $statement = Database:::prepare($s, [PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY]);
32
33     try {
34         $statement->bindParam(":idUser", $_SESSION['idUser']);
35         $statement->execute();
36     } catch (PDOException $e) {
37         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
38     }
39
40     while ($row = $statement->fetch(PDO:::FETCH_ASSOC, PDO:::FETCH_ORI_NEXT)) {
41         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameStatut"], $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
42         $acheteur = new User($row["idUser"], $row["nameBuyer"], $row["surnameBuyer"], $row["emailBuyer"], "", 0, $row["imgBuyer"]);
43         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"], $row["emailSeller"], "", 0, $row["imgSeller"]);
44
45         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur, $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
46         array_push($arr, $u);
47     }
48
49     return $arr;
50 }
51 /**
52 * Renvoie true ou false si la paire sélectionnée est déjà dans les FAVORITES de l'utilisateur
53 */
54 * @param integer $idSneaker // Paire
55 * @param integer $idUser // Utilisateur
56 * @return boolean
57 */
58 function CheckIfFavorite(int $idSneaker, int $idUser): bool
59 {
60     $s = "SELECT * FROM FAVORITES WHERE idSneaker = :idSneaker AND idUser = :idUser";
61     $statement = Database:::prepare($s, [PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY]);
62
63     try {
64         $statement->execute(array(':idSneaker' => $idSneaker, ':idUser' => $idUser));
65
66         $row = $statement->fetch(PDO:::FETCH_ASSOC);
67
68         if ($row) {
69             return true;
70         }
71     } catch (PDOException $e) {
72         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
73     }
74
75     return false;
76 }
77
78 /**
79 * Vérifie si la paire indiquée existe dans la base de données
80 */
81 * @param integer $idSneaker // Paire à vérifier
82 * @return boolean // Oui ou non

```

```

83     */
84     function DoesSneakerExist(int $idSneaker) : bool
85     {
86         $s = "SELECT * FROM SNEAKERS WHERE idSneaker = :idSneaker";
87         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
88
89         try {
90             $statement->execute(array(':idSneaker' => $idSneaker));
91
92             $row = $statement->fetch(PDO::FETCH_ASSOC);
93
94             if ($row) {
95                 return true;
96             }
97         } catch (PDOException $e) {
98             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
99         }
100
101         return false;
102     }
103
104 /**
105 * Supprime la paire de détail dans les FAVORITES
106 *
107 * @param integer $idSneaker // Paire
108 * @param integer $idUser // Utilisateur
109 * @return void
110 */
111 function RemoveFromFavorite(int $idSneaker, int $idUser): void
112 {
113     if ($self::DoesSneakerExist($idSneaker)) {
114         try {
115             $s = "DELETE FROM FAVORITES WHERE idSneaker = :idSneaker AND idUser =
116 :idUser";
117             $statement = Database::prepare($s);
118
119             $statement->bindParam(':idSneaker', $idSneaker);
120             $statement->bindParam(':idUser', $idUser);
121
122             $statement->execute();
123
124         } catch (PDOException $e) {
125             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
126         }
127     }
128
129 /**
130 * Compte le nombre de paires favorites dans la base
131 *
132 * @return integer // le nombre de paires favorites
133 */
134 function CountSneakers(): int
135 {
136     $s = "SELECT COUNT(*) FROM FAVORITES WHERE idUser = :idUser";
137     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
138
139     try {
140         $statement->bindParam(":idUser", $_SESSION['idUser']);
141         $statement->execute();

```

```

141     $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
142     return (int) $count["COUNT(*)"];
143
144 } catch (PDOException $e) {
145     echo 'Problème de lecture de la base de données: ' . $e->getMessage();
146     return 0;
147 }
148
149 }
150
151 }
```

src\Model\DetailModel.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modele de la page détail
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10 class Detail
11 {
12
13     /**
14      * Récupère la paire correspondant à l'id reçu en paramètre
15      *
16      * @param integer $idSneaker // id de la paire désirée
17      * @return Sneaker|false // Retourne la paire ou faux si erreur
18      */
19     function GetSneakerById(int $idSneaker): Sneaker|false
20     {
21         $s = "SELECT * FROM VIEW_SNEAKERS WHERE idSneaker = :idSneaker";
22         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
23
24         try {
25             $statement->execute(array(':idSneaker' => $idSneaker));
26
27             $row = $statement->fetch(PDO::FETCH_ASSOC);
28
29             if ($row) {
30                 return new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"],
31                 $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"],
32                 $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
33             }
34         } catch (PDOException $e) {
35             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
36         }
37
38         return false;
39     }
40
41     /**
42      * Renvoie true ou false si la paire sélectionnée est déjà dans les favoris de l'
43      * utilisateur
44      *
45      * @param integer $idSneaker // Paire
46  
```

```
43     * @param integer $idUser // Utilisateur
44     * @return boolean
45     */
46     function CheckIfFavorite(int $idSneaker, int $idUser): bool
47     {
48         $s = "SELECT * FROM FAVORITES WHERE idSneaker = :idSneaker AND idUser = :idUser";
49         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
50
51         try {
52             $statement->execute(array(':idSneaker' => $idSneaker, ":idUser" => $idUser));
53
54             $row = $statement->fetch(PDO::FETCH_ASSOC);
55
56             if ($row) {
57                 return true;
58             }
59         } catch (PDOException $e) {
60             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
61         }
62
63         return false;
64     }
65
66 /**
67 * Ajoute la paire de détail dans les favoris
68 *
69 * @param integer $idSneaker // Paire
70 * @param integer $idUser // Utilisateur
71 * @return void
72 */
73     function AddToFavorite(int $idSneaker, int $idUser): void
74     {
75         try {
76             $s = "INSERT INTO FAVORITES(idUser, idSneaker) VALUES(:idUser, :idSneaker)";
77             $statement = Database::prepare($s);
78
79             $statement->bindParam(':idSneaker', $idSneaker);
80             $statement->bindParam(':idUser', $idUser);
81
82             $statement->execute();
83
84         } catch (PDOException $e) {
85             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
86         }
87     }
88
89 /**
90 * Supprime la paire de détail dans les favoris
91 *
92 * @param integer $idSneaker // Paire
93 * @param integer $idUser // Utilisateur
94 * @return void
95 */
96     function RemoveFromFavorite(int $idSneaker, int $idUser): void
97     {
98         try {
99             $s = "DELETE FROM FAVORITES WHERE idSneaker = :idSneaker AND idUser =
:idUser";
100            $statement = Database::prepare($s);
101        }
```

```

102         $statement->bindParam(':idSneaker', $idSneaker);
103         $statement->bindParam(':idUser', $idUser);
104
105         $statement->execute();
106
107     } catch (PDOException $e) {
108         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
109     }
110 }
111
112 /**
113 * Vérifie si l'utilisateur souhaité a réservé une paire
114 *
115 * @param integer $idSneaker // l'utilisateur à vérifier
116 * @return boolean // oui ou non
117 */
118 function doesUserHasReserved(int $idSneaker) : bool
119 {
120     $s = "SELECT * FROM TRANSACTIONS WHERE idSneaker = :idSneaker AND idBuyer =
121 :idUser";
122     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
123
124     try {
125         $statement->execute(array(':idSneaker' => $idSneaker, ':idUser' =>
126 $_SESSION['idUser']));
127
128         $row = $statement->fetch(PDO::FETCH_ASSOC);
129
130         if ($row) {
131             return true;
132         }
133     } catch (PDOException $e) {
134         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
135     }
136
137     return false;
138 }
139
140 /**
141 * Vérifie si la paire souhaitée est réservée
142 *
143 * @param integer $idSneaker // paire à vérifier
144 * @return boolean // oui ou non
145 */
146 function IsReserved(int $idSneaker) : bool
147 {
148     $s = "SELECT * FROM TRANSACTIONS WHERE idSneaker = :idSneaker AND idBuyer IS NOT
149 NULL";
150     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
151
152     try {
153         $statement->execute(array(':idSneaker' => $idSneaker));
154
155         $row = $statement->fetch(PDO::FETCH_ASSOC);
156
157         if ($row) {
158             return true;
159         }
160     } catch (PDOException $e) {
161         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
162     }
163 }
```

```

160
161     return false;
162 }
163
164 /**
165 * Vérifie si les données passées sont conformes pour réserver
166 *
167 * @param integer $meetPoint // id du point de rendez vous
168 * @param string|DateTime $meetingDate // date et heure de rendez vous
169 * @return array // tableau des potentielles erreurs
170 */
171 function CheckErrorReserve(int $meetPoint, string|DateTime $meetingDate) : array
172 {
173     $errorMessages = [];
174     $currentDate = new DateTime();
175     $meetingDate = new DateTime($meetingDate);
176
177     if ($meetPoint == 0 || $meetPoint == -1) {
178         $errorMessages["meetPoint"] = "*Select a meet point";
179     }
180
181
182     if($meetingDate == "" || $meetingDate == NULL || $currentDate > $meetingDate){
183         $errorMessages["meetingDate"] = "*Select a valid date";
184     }
185
186
187     return $errorMessages;
188 }
189
190 /**
191 * Insère la date et l'heure du rendez vous en base de données
192 *
193 * @param integer $idMeetPoint // id du point de rendez vous
194 * @param string|DateTime $meetingDate // date et heure de rendez vous
195 * @return boolean // L'opération a réussi ou non
196 */
197 function InsertDateMeetPoint(int $idMeetPoint, string|DateTime $meetingDate) : bool
198 {
199     try {
200         $s = "INSERT INTO DATEMEETPLACE(idMeetPoint, dateHourMeetPlace)
VALUES(:idMeetPoint, :dateHourMeetPlace)";
201         $statement = Database::prepare($s);
202
203         $statement->bindParam(':idMeetPoint', $idMeetPoint);
204         $statement->bindParam(':dateHourMeetPlace', $meetingDate);
205
206         $statement->execute();
207
208         return true;
209     } catch (PDOException $e) {
210         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
211         return false;
212     }
213 }
214
215
216 /**
217 * Résèrve une paire dans la base de données
218 *

```

```
219 * @param integer $idSneaker // L'utilisateur souhaitant réserver
220 * @param integer $idMeetPoint // L'id du point de rendez vous
221 * @param string|DateTime $meetingDate // La date et l'heure de rendez vous
222 * @return void
223 */
224 function Reserve(int $idSneaker, int $idMeetPoint, string|DateTime $meetingDate)
225 {
226     try {
227         // Commence la transaction
228         Database::beginTransaction();
229
230         // Insère la date de rendez-vous
231         self::InsertDateMeetPoint($idMeetPoint, $meetingDate);
232         $idDateMeetPlace = Database::lastInsertId();
233
234         // Met à jour le point de rendez-vous et le statut
235         $s = "UPDATE TRANSACTIONS SET idBuyer = :idUser, idMeetPoint = :idMeetPoint,
236 idTStatut = 3, idDateMeetPlace = :idDateMeetPlace WHERE idSneaker = :idSneaker";
237         $statement = Database::prepare($s);
238
239         $statement->bindParam(':idSneaker', $idSneaker);
240         $statement->bindParam(':idMeetPoint', $idMeetPoint);
241         $statement->bindParam(':idDateMeetPlace', $idDateMeetPlace);
242         $statement->bindParam(':idUser', $_SESSION['idUser']);
243
244         $statement->execute();
245
246         Database::commit();
247     } catch (PDOException $e) {
248         // Annule la transaction
249         Database::rollBack();
250
251         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
252     }
253 }
254 /**
255 * Annule la réservation de la paire souhaitée
256 *
257 * @param integer $idSneaker // La paire à annuler
258 * @return void
259 */
260 function CancelReserve(int $idSneaker)
261 {
262     try {
263         $s = "UPDATE TRANSACTIONS SET idBuyer = NULL, idTStatut = 1, idDateMeetPlace =
264 NULL WHERE idSneaker = :idSneaker";
265         $statement = Database::prepare($s);
266
267         $statement->bindParam(':idSneaker', $idSneaker);
268
269         $statement->execute();
270
271     } catch (PDOException $e) {
272         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
273     }
274 }
275 /**
276 * Récupère tous les points de rencontres de la
```

```
277     *
278     * @return array
279     */
280     function GetAllMeetPoint(): array
281     {
282         $arr = array();
283
284         $s = "SELECT * FROM MEETPOINTS";
285         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
286 ;
287         try {
288             $statement->execute();
289         } catch (PDOException $e) {
290             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
291         }
292
293         while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
294             $u = new MeetPoint($row["idMeetPoint"], $row["nameMeetPoint"], $row["adressMeetPoint"], $row["descriptionMeetPoint"], "");
295             array_push($arr, $u);
296         }
297
298         return $arr;
299     }
299 }
```

src\Model\database.php

```
1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : SneakersMarket (TPI)
5 Auteur : Léo Triano
6 Desc. : Site d'achat-revente de sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Classe qui connecte la base de données
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
11
12 require_once('./src/Model/config.php');
13
14 class Database
15 {
16
17     private static $objInstance;
18
19     private static function getInstance()
20     {
21         if (!self::$objInstance) {
22             try {
23                 $dsn = EDB_DBTYPE . ':host=' . EDB_HOST . ';port=' . EDB_PORT . ';dbname='
24 . EDB_DBNAME;
25                 self::$objInstance = new PDO($dsn, EDB_USER, EDB_PASS, array('charset' =>
26 utf8'));
27                 self::$objInstance->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)
28 ;
29             } catch (PDOException $e) {
30                 echo "Database error : " . $e;
31             }
32         }
33     }
34 }
```

```

30         return self::$objInstance;
31     }
32
33     final public static function __callStatic($chrMethod, $arrArguments)
34     {
35         $objInstance = self::getInstance();
36         return call_user_func_array($array[$objInstance], $chrMethod), $arrArguments);
37     }
38 }
```

src\Model\Constants.php

```

1 <?php
2 /**
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Constantes utilisées dans le site
9 */
10 class Constants
11 {
12     // Constantes
13     const MIN_PWD_CAR = 8; // Nombre de caractères minimum pour le mot de passe
14     const MAX_SURNAME_CAR = 15; // Nombre de caractères maximum pour le prénom
15     const MAX_NAME_CAR = 10; // Nombre de caractères maximum pour le nom
16     const MAX_DESC_CAR = 30; // Nombre de caractères maximum pour la description
17     const IMG_SIZE = 3000000; // Taille max d'une image
18     const SIZE_MIN = 0; // Taille min d'une image
19     const SIZE_MAX = 60; // Taille max d'une paire
20     const PRICE_MIN = 1; // Prix minimum d'une paire
21
22
23     // Messages d'erreur
24     const ERROR_MIN_PWD_CAR = "*Must be at least " . self::MIN_PWD_CAR . " characters long";
25     const NOT_EMPTY = "*Must not be empty";
26     const SCHOOL_NOT_EXIST = "*Must select a school";
27     const PWD_CAPS_SPE = "*Must contain 1 caps, digit and special caracter";
28     const MUST_BE_IMG = "*Must be an image";
29     const EMAIL = "*Must be a valid email";
30     const ERROR_EMAIL_TAKEN = "*Email already taken";
31     const MAX_CAR_SURNAME = "*Max " . self::MAX_SURNAME_CAR . " characters";
32     const MAX_CAR_NAME = "*Max " . self::MAX_NAME_CAR . " characters";
33     const SELECT_BRAND = "*Select a brand";
34     const SELECT_MODEL = "*Select a model";
35     const MUST_BE_NUMBER = "*Must be a number";
36     const DESC_MAX_CAR = "*Max " . self::MAX_DESC_CAR . " characters";
37     const WRONG_EMAIL_PWD = "*Informations are incorrect or your account has been blocked";
38     const ERROR_IMG_SIZE = "*Image is too big";
39     const SIZE_MIN_MAX = "*Must be between " . self::SIZE_MIN . " and " . self::SIZE_MAX;
40     const PRICE_MIN_MAX = "*can't be lower than " . self::PRICE_MIN . "$";
41     const MEET_POINT_EXIST = "*Meet point already exist";
42     const SCHOOL_EXIST = "*School already exist";
43
44     /**
45      * Empêche les injections SQL et XSS
46      *
47      * @param string $string // la chaîne de caractère à sécuriser

```

```
48     * @return string // la chaîne de caractère sécurisée
49     */
50     static function AntiXSS($string): string
51     {
52         $string = htmlspecialchars($string, ENT_QUOTES);
53         $string = strip_tags($string);
54         $string = addslashes($string);
55
56         return $string;
57     }
58
59 /**
60 * Convertit l'image donnée en base 64
61 *
62 * @param array $file // Image à convertir
63 * @return string // Chaîne en base 64
64 */
65 static function ConvertImgTo64($file)
66 {
67     $src = "";
68
69     if($file["name"] != ""){
70         $data = file_get_contents($file['tmp_name']);
71         $finfo = new finfo(FILEINFO_MIME_TYPE);
72         $mime = $finfo->file($file['tmp_name']);
73         $src = 'data:' . $mime . ';base64,' . base64_encode($data);
74     }
75     return $src;
76 }
77
78 }
```

src\Model\configSample.php

```
1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Exemple des informations requises pour la connexion à la base de données
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10 define('EDB_DBTYPE', "mysql");
11 define('EDB_DBNAME', "xxx");
12 define('EDB_HOST', "127.0.0.1");
13 define('EDB_PORT', "3306");
14 define('EDB_USER', "xxx");
15 define('EDB_PASS', "xxx");
```

src\Model\config.php

```
1 <?php
2 define('EDB_DBTYPE', "mysql");
3 define('EDB_DBNAME', "SneakersMarket");
4 define('EDB_HOST', "127.0.0.1");
5 define('EDB_PORT', "3306");
```

```
6 | define('EDB_USER', "dbeaver_admin");
7 | define('EDB_PASS', "Super");
```

src\Model\AuthModel.php

```
1 | <?php
2 | /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 | Projet : SneakersMarket (TPI)
4 | Auteur : Léo Triano
5 | Desc. : Site d'achat-revente de sneakers
6 | Version : 1.0
7 | Date : Mai 2024
8 | Page : Modele de la page login/register (auth)
9 | * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10| class Auth
11| {
12|     /**
13|      * Vérifie les données envoyées en POST du register
14|      * et retourne un tableau d'erreur en fonction de s'il y en a ou pas
15|      *
16|      * @param User $user // Utilisateur à vérifier
17|      * @param integer $idSchool // école sélectionnée
18|      * @return array // Tableau d'erreurs
19|      */
20|     function ErrorMessagesRegister(User $user, int $idSchool, $img): array
21|     {
22|         // Variables
23|         $errorMessages = [];
24|
25|         // Expressions régulières
26|         $expressionCarSpe = "/[!@#$%^&*()_+{}\\[\\]:;<>, .?~\\\\\\\\\\\\\\\\-\\\"\\'\\\\|\\\\s]/"; // Reprise
d'un ancien projet, trouvée à l'époque sur internet
27|         $expressionEmail = "/[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}/"; // Reprise
d'un ancien projet, trouvée à l'époque sur internet
28|         $expressionCaps = "/[A-Z]/"; // écrite pour ce projet
29|         $expressionDigit = "/[0-9]/"; // écrite pour ce projet
30|
31|         // Erreur si l'email est déjà utilisé
32|         if ($self::CheckEmailAlreadyUsed($user->emailUser)) {
33|             $errorMessages["errorEmailRegister"] = Constants::ERROR_EMAIL_TAKEN;
34|         }
35|
36|         if ($img["type"] != "image/png" && $img["type"] != "image/jpeg" && $img["type"] !=
"image/jpg" && $img["name"] != "") {
37|             $errorMessages["errorImg"] = Constants::MUST_BE_IMG;
38|         }
39|         if ($img["size"] > Constants::IMG_SIZE) {
40|             $errorMessages["errorImg"] = Constants::ERROR_IMG_SIZE;
41|         }
42|
43|         // Erreur si le nom est vide
44|         if ($user->nameUser == "") {
45|             $errorMessages["errorName"] = Constants::NOT_EMPTY;
46|         }
47|
48|         if(strlen($user->nameUser) > Constants::MAX_NAME_CAR){
49|             $errorMessages["errorName"] = Constants::MAX_CAR_NAME;
50|         }
51|     }
```

```

52     // Erreur si le prénom est vide
53     if ($user->surnameUser == "") {
54         $errorMessages["errorSurname"] = Constants::NOT_EMPTY;
55     }
56
57     if(strlen($user->surnameUser) > Constants::MAX_SURNAME_CAR){
58         $errorMessages["errorSurname"] = Constants::MAX_CAR_SURNAME;
59     }
60
61     if (!preg_match($expressionEmail, $user->emailUser)) {
62         $errorMessages["errorEmailRegister"] = Constants::EMAIL;
63     }
64
65     // Erreur si l'email est vide
66     if ($user->emailUser == "") {
67         $errorMessages["errorEmailRegister"] = Constants::NOT_EMPTY;
68     }
69
70     // Erreur si le mot de passe ne contient pas de caractère spécial, de majuscule ou
de chiffre
71     if (!preg_match($expressionCarSpe, $user->pwdUser) || !preg_match($expressionCaps,
$user->pwdUser) || !preg_match($expressionDigit, $user->pwdUser)) {
72         $errorMessages["errorPasswordRegister"] = Constants::PWD_CAPS_SPE;
73     }
74
75     // Erreur si le nom ne convient pas à la plage de nombre de caractères donné
76     if (strlen($user->pwdUser) < Constants::MIN_PWD_CAR) {
77         $errorMessages["errorPasswordRegister"] = Constants::ERROR_MIN_PWD_CAR;
78     }
79
80     // Erreur si le mot de passe est vide
81     if (!self::DoesSchoolExist($idSchool)) {
82         $errorMessages["errorSchoolRegister"] = Constants::SCHOOL_NOT_EXIST;
83     }
84
85     // Erreur si le mot de passe est vide
86     if ($user->pwdUser == "") {
87         $errorMessages["errorPasswordRegister"] = Constants::NOT_EMPTY;
88     }
89
90     return $errorMessages;
91 }
92
93 /**
94 * Vérifie si l'utilisateur actuel n'est pas bloqué ou non
95 *
96 * @return boolean // Oui ou non
97 */
98 function IsUserNotBlocked($emailUser) : bool
99 {
100     $s = "SELECT * FROM USERS WHERE emailUser = :emailUser AND idUstatut = 1";
101     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
102
103     try {
104         $statement->execute(array(':emailUser' => $emailUser));
105
106         $row = $statement->fetch(PDO::FETCH_ASSOC);
107
108         if ($row) {
109             return true;

```

```

110     }
111 } catch (PDOException $e) {
112     echo 'Problème de lecture de la base de données: ' . $e->getMessage();
113     return false;
114 }
115
116     return false;
117 }
118
119 /**
120 * Vérifie les données envoyées en POST du login
121 * et retourne un tableau d'erreur en fonction de s'il y en a ou pas
122 *
123 * @param User $user // Utilisateur à vérifier
124 * @return array // Tableau des potentielles erreurs
125 */
126 function ErrorMessagesLogin(User $user): array
127 {
128     $errorMessages = [];
129
130     // Erreur si l'email n'existe pas
131     if (!self::CheckEmailAlreadyUsed($user->emailUser) ||
132 !self::IsUserNotBlocked($user->emailUser)) {
133         $errorMessages["errorLogin"] = Constants::WRONG_EMAIL_PWD;
134     } else {
135         if (!self::CheckPassword($user->emailUser, $user->pwdUser)) {
136             $errorMessages["errorLogin"] = Constants::WRONG_EMAIL_PWD;
137         }
138     }
139
140     return $errorMessages;
141 }
142
143 /**
144 * récupère l'utilisateur avec l'email correspondant
145 *
146 * @param string $email // email à vérifier
147 * @return User|false // Retourne l'utilisateur s'il y en a un, false si non
148 */
149 function GetUserByEmail(string $email): User|false
150 {
151     $s = "SELECT * FROM USERS WHERE emailUser = :email";
152     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
153
154     try {
155         $statement->execute([':email' => $email]);
156
157         $row = $statement->fetch(PDO::FETCH_ASSOC);
158
159         if ($row) {
160             return new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
161 $row["emailUser"], $row["pwdUser"], $row["idSchool"], $row["imgUser"], $row["idRole"]);
162         }
163     } catch (PDOException $e) {
164         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
165     }
166
167     return false;
168 }

```

```

168     /**
169      * Vérifie si l'email entré correspond déjà à un utilisateur dans la base, renvoie
170      * true ou false en fonction
171      * @param string $email email à vérifier
172      * @return boolean
173      */
174     function CheckEmailAlreadyUsed(string $email): bool
175     {
176         $email = self:: GetUserByEmail($email);
177
178         return ($email !== false);
179     }
180
181     /**
182      * Vérifie si le mot de passe correspond avec le mot de passe hashé dans la base
183      *
184      * @param User $user // utilisateur à vérifier
185      * @return boolean
186      */
187     function CheckPassword(string $email, $password): bool
188     {
189         $Hashedpassword = self:: GetUserByEmail($email)->pwdUser;
190
191         return password_verify($password, $Hashedpassword);
192     }
193
194
195     /**
196      * Insère dans la base les données de l'utilisateur
197      *
198      * @param User $user // L'utilisateur à insérer
199      * @param integer $idSchool // l'id de l'école choisie
200      * @return void
201      */
202     function Register(User $user, int $idSchool, $img): void
203     {
204         try {
205             // Sécurisation des données
206             $name = Constants::AntiXSS($user->nameUser);
207             $surname = Constants::AntiXSS($user->surnameUser);
208             $email = Constants::AntiXSS($user->emailUser);
209             $password = Constants::AntiXSS($user->pwdUser);
210             $password = password_hash($password, PASSWORD_DEFAULT);
211             $idSchool = Constants::AntiXSS($idSchool);
212             if ($img["name"] != "") {
213                 $img = Constants::ConvertImgTo64($img);
214             } else {
215                 $img = "";
216             }
217
218             $s = "INSERT INTO USERS(nameUser, surnameUser, emailUser, pwdUser, idSchool,
219 imgUser) VALUES(:name, :surname, :email, :password, :idSchool, :img)";
220             $statement = Database::prepare($s);
221
222             $statement->bindParam(':name', $name);
223             $statement->bindParam(':surname', $surname);
224             $statement->bindParam(':email', $email);
225             $statement->bindParam('password', $password);
226             $statement->bindParam('idSchool', $idSchool);

```

```

226         $statement->bindParam('img', $img);
227
228     $statement->execute();
229
230 } catch (PDOException $e) {
231     echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
232 }
233 }
234
235 /**
236 * Connecte l'utilisateur
237 *
238 * @param User $user // L'utilisateur à connecter
239 * @return void
240 */
241 function Login($email): void
242 {
243     $user = self:: GetUserByEmail($email);
244
245     if ($user !== false) {
246         self::ValuesSession(new User($user->idUser, $user->nameUser, $user->
surnameUser, $user->emailUser, "", $user->idSchool, $user->imgUser, $user->idRole));
247     }
248 }
249
250 /**
251 * Rentre les données peu importantes de l'utilisateur en session
252 *
253 * @param User $user
254 * @return void
255 */
256 function ValuesSession(User $user): void
257 {
258     $_SESSION["idUser"] = $user->idUser;
259     $_SESSION["nameUser"] = $user->nameUser;
260     $_SESSION["surnameUser"] = $user->surnameUser;
261     $_SESSION["emailUser"] = $user->emailUser;
262     $_SESSION["idSchool"] = $user->idSchool;
263     $_SESSION["imgUser"] = $user->imgUser;
264     $_SESSION['idRole'] = $user->idRole;
265     $_SESSION["customMessage"] = "Welcome " . $user->surnameUser . " !";
266 }
267
268 /**
269 * Vérifie si l'école sélectionnée est en base
270 *
271 * @param integer $idSchool // id à vérifier
272 * @return bool // Vrai ou faux
273 */
274 function DoesSchoolExist(int $idSchool): bool
275 {
276     $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";
277     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
278
279     try {
280         $statement->bindParam(':idSchool', $idSchool);
281         $statement->execute();
282
283         $row = $statement->fetch(PDO::FETCH_ASSOC);
284     }

```

```

285     if ($row) {
286         return true;
287     } else {
288         // School not found
289         return false;
290     }
291 } catch (PDOException $e) {
292     echo 'Problème de lecture de la base de données: ' . $e->getMessage();
293     return false;
294 }
295 }

297 /**
298 * Récupère toutes les écoles de la base
299 *
300 * @return array // Tableau des écoles
301 */
302 function GetAllSchools(): array
303 {
304     $arr = array();
305
306     $s = "SELECT * FROM SCHOOLS";
307     $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
308 ;
309     try {
310         $statement->execute();
311     } catch (PDOException $e) {
312         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
313     }
314
315     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
316         $u = new School($row["idSchool"], $row["nameSchool"]);
317         array_push($arr, $u);
318     }
319
320     return $arr;
321 }
322 }
```

src\Model\AdminModel.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : SneakersMarket (TPI)
4 Auteur : Léo Triano
5 Desc. : Site d'achat-revente de sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Modele de la page admin
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10 class Admin
11 {
12     /**
13      * Récupère tous les utilisateurs de la base de données. Si on a recherché quelque chose, récupère tous les utilisateurs correspondant à la recherche
14      *
15      * @param string|null $searchBarUsers // string recherchée dans la barre de recherche
16      * @param integer $indexFirst // [Pagination] index du premier utilisateur

```

```

17     * @param integer $indexLast // [Pagination] index du dernier utilisateur
18     * @return array // Tableau de User
19     */
20     function GetAllUsers(string|null $searchBarUsers, int $indexFirst, int $indexLast): array
21     {
22         $arr = array();
23
24         // SI on a recherché quelque chose
25         if ($searchBarUsers != "") {
26             $s = "SELECT * FROM USERS WHERE nameUser LIKE CONCAT('%', :search, '%') OR
surnameUser LIKE CONCAT('%', :search, '%') OR emailUser LIKE CONCAT('%', :search, '%') AND
idRole = 1 ORDER BY idUser DESC LIMIT :indexFirst, :indexLast";
27             $statement = Database::prepare($s, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
28             try {
29                 $statement->bindParam(':search', $searchBarUsers, PDO::PARAM_STR);
30                 $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
31                 $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
32                 $statement->execute();
33             } catch (PDOException $e) {
34                 echo 'Problème de lecture de la base de données: ' . $e->getMessage();
35             }
36
37             while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
38                 $u = new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
$row["emailUser"], "", $row["idSchool"], $row["imgUser"]);
39                 array_push($arr, $u);
40             }
41         } else {
42             $s = "SELECT * FROM USERS WHERE idRole = 1 ORDER BY idUser DESC LIMIT
:indexFirst, :indexLast";
43             $statement = Database::prepare($s, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
44             try {
45                 $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
46                 $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
47                 $statement->execute();
48             } catch (PDOException $e) {
49                 echo 'Problème de lecture de la base de données: ' . $e->getMessage();
50             }
51
52             while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
53                 $u = new User($row["idUser"], $row["nameUser"], $row["surnameUser"],
$row["emailUser"], "", $row["idSchool"], $row["imgUser"]);
54                 array_push($arr, $u);
55             }
56         }
57
58         return $arr;
59     }
60
61     /**
62     * Récupère l'école en fonction de son id
63     *
64     * @param integer $idSchool // id de l'école voulue
65     * @return School // l'école en entier
66     */
67     function GetSchoolFromId(int $idSchool): School
68     {
69         $s = "SELECT * FROM SCHOOLS WHERE idSchool = :idSchool";

```

```

70     ; $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
71
72     try {
73         $statement->bindParam(':idSchool', $idSchool, PDO::PARAM_INT);
74         $statement->execute();
75     } catch (PDOException $e) {
76         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
77     }
78
79     $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
80
81     $u = new School($row["idSchool"], $row["nameSchool"]);
82
83     return $u;
84 }
85 /**
86 * Compte le nombre d'utilisateurs
87 *
88 * @param string|null $searchBarUsers // Si l'utilisateur a recherché quelque chose on prend uniquement les utilisateurs correspondant à la recherche
89 * @return integer // Le nombre d'utilisateurs
90 */
91 function CountUsers(string|null $searchBarUsers): int
92 {
93     if ($searchBarUsers != "") {
94         $s = "SELECT COUNT(*) FROM USERS WHERE idRole = 1 AND nameUser LIKE CONCAT('%', :search, '%') OR surnameUser LIKE CONCAT('%', :search, '%') OR emailUser LIKE CONCAT('%', :search, '%')";
95         $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
96     try {
97         $statement->bindParam(':search', $searchBarUsers, PDO::PARAM_STR);
98         $statement->execute();
99         $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
100        return (int) $count["COUNT(*)"];
101    } catch (PDOException $e) {
102        echo 'Problème de lecture de la base de données: ' . $e->getMessage();
103        return 0;
104    }
105 } else {
106     $s = "SELECT COUNT(*) FROM USERS WHERE idRole = 1";
107     $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
108     try {
109         $statement->execute();
110         $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
111         return (int) $count["COUNT(*)"];
112     } catch (PDOException $e) {
113         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
114         return 0;
115     }
116 }
117 }
118 }
119 }
120
121 /**
122 * Compte le nombre de sneakers dans la base de données
123 *
124 */

```

```

125     * @return integer // Le nombre de sneakers
126     */
127     function CountSneakers(): int
128     {
129         $s = "SELECT COUNT(*) FROM TRANSACTIONS INNER JOIN USERS ON TRANSACTIONS.idSeller
= USERS.idUser WHERE idTStatut = 1 AND idSeller != :idSeller AND idUStatut = 1";
130         $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
;
131         try {
132             $statement->bindParam(':idSeller', $_SESSION['idUser'], PDO::PARAM_INT);
133             $statement->execute();
134             $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
135             return (int) $count["COUNT(*)"];
136
137         } catch (PDOException $e) {
138             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
139             return 0;
140         }
141     }
142
143 /**
144 * Compte le nombre de transactions de l'utilisateur voulu
145 *
146 * @param integer $idUser // L'id de l'utilisateur souhaité
147 * @return integer // Le nombre de transaction
148 */
149 function CountUsersTransactions(int $idUser): int
150 {
151     $s = "SELECT COUNT(*) FROM TRANSACTIONS WHERE idSeller = :idUser OR idBuyer =
:idUser AND idTStatut = 2";
152     $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
;
153     try {
154         $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
155         $statement->execute();
156         $count = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
157         return (int) $count["COUNT(*)"];
158
159     } catch (PDOException $e) {
160         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
161         return 0;
162     }
163 }
164
165 /**
166 * Bloque l'utilisateur en changeant son statut dans la base de données
167 *
168 * @param integer $idUser // L'id de l'utilisateur souhaité
169 * @return bool // si l'opération a réussi ou non
170 */
171 function BlockUser(int $idUser): bool
172 {
173     try {
174         // 1 - Active / 2 - Blocked
175         $s = "UPDATE USERS SET idUStatut = 2 WHERE idUser = :idUser";
176         $statement = Database:::prepare($s);
177
178         $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
179
180         $statement->execute();
181

```

```

182         return true;
183
184     } catch (PDOException $e) {
185         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
186         return false;
187     }
188 }
189
190 /**
191 * Vérifie si l'utilisateur est bloqué
192 *
193 * @param integer $idUser // l'id de l'utilisateur à vérifier
194 * @return bool // Oui ou non
195 */
196 function IsBlocked(int $idUser): bool
197 {
198     $s = "SELECT * FROM USERS WHERE idUser = :idUser AND idUStatut = 2";
199     $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
200 ;
201     try {
202         $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
203         $statement->execute();
204     } catch (PDOException $e) {
205         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
206     }
207
208     $row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT);
209     if ($row) {
210         return true;
211     } else {
212         return false;
213     }
214 }
215 /**
216 * Débloque un utilisateur
217 *
218 * @param integer $idUser // L'id de l'utilisateur souhaité
219 * @return boolean // si l'opération a réussi ou non
220 */
221 function UnblockUser(int $idUser): bool
222 {
223     try {
224         // 1 - Active / 2 - Blocked
225         $s = "UPDATE USERS SET idUStatut = 1 WHERE idUser = :idUser";
226         $statement = Database:::prepare($s);
227
228         $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
229
230         $statement->execute();
231
232         return true;
233
234     } catch (PDOException $e) {
235         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
236         return false;
237     }
238 }
239
240 /**

```

```
241     * Supprime l'utilisateur de la base de données
242     *
243     * @param integer $idUser // l'id de l'utilisateur à supprimer
244     * @return boolean // si l'opération a réussi ou non
245     */
246     function DeleteUser(int $idUser): bool
247     {
248         try {
249             $s = "DELETE FROM USERS WHERE idUser = :idUser";
250             $statement = Database::prepare($s);
251
252             $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
253
254             $statement->execute();
255
256             return true;
257
258         } catch (PDOException $e) {
259             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
260             return false;
261         }
262     }
263
264 /**
265 * Supprime les offres d'achats de l'utilisateur
266 *
267 * @param integer $idUser // L'id de l'utilisateur souhaité
268 * @return boolean // si l'opération a réussi ou non
269 */
270     function RemoveBuyingOffers(int $idUser): bool
271     {
272         try {
273             $s = "UPDATE TRANSACTIONS SET idBuyer = NULL AND idMeetPoint = NULL AND
274 idDateMeetPlace = NULL WHERE idBuyer = :idUser";
275             $statement = Database::prepare($s);
276
277             $statement->bindParam(':idUser', $idUser, PDO::PARAM_INT);
278
279             $statement->execute();
280
281             return true;
282
283         } catch (PDOException $e) {
284             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
285             return false;
286         }
287
288 /**
289 * Récupère toutes les sneakers en vente
290 *
291 * @param integer $indexFirst // [Pagination] index du premier utilisateur
292 * @param integer $indexLast // [Pagination] index du dernier utilisateur
293 * @return array // Tableau de ViewTransaction
294 */
295     function GetAllShoesInSale(int $indexFirst, int $indexLast): array
296     {
297         $arr = array();
298
299         $s = "SELECT * FROM VIEW_TRANSACTIONS WHERE nameStatut LIKE 'In progress' AND
```

```

  idSeller != :idSeller LIMIT :indexFirst, :indexLast";
      $statement = Database:::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY))
;
301     try {
302         $statement->bindParam(':idSeller', $_SESSION['idUser']);
303         $statement->bindParam(':indexFirst', $indexFirst, PDO::PARAM_INT);
304         $statement->bindParam(':indexLast', $indexLast, PDO::PARAM_INT);
305
306
307         $statement->execute();
308     } catch (PDOException $e) {
309         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
310     }
311
312     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
313         $sneaker = new Sneaker($row["idSneaker"], $row["sizeSneaker"], $row["priceSneaker"], $row["imgSneaker"], $row["descriptionSneaker"], $row["nameSStatut"], $row["nameBrand"], $row["nameModel"], $row["nameVisibility"]);
314         $acheteur = new User($row["idBuyer"], $row["nameBuyer"], $row["surnameBuyer"], $row["emailBuyer"], "", 0, $row["imgBuyer"]);
315         $vendeur = new User($row["idSeller"], $row["nameSeller"], $row["surnameSeller"], $row["emailSeller"], "", 0, $row["imgSeller"]);
316
317         $u = new ViewTransaction($row["idTransaction"], $sneaker, $acheteur, $vendeur, $row["nameStatut"], new MeetPoint(0, "", $row["adressMeetPoint"], "", $row["dateHourMeetPlace"]));
318         array_push($arr, $u);
319     }
320
321     return $arr;
322 }
323
324 /**
325 * Vérifie si une sneaker existe dans la base de données
326 *
327 * @param integer $idSneaker // l'id de la sneaker à vérifier
328 * @return boolean // Oui ou non
329 */
330 function DoesSneakerExist(int $idSneaker): bool
331 {
332
333     try {
334         $s = "SELECT * FROM TRANSACTIONS WHERE idSneaker = :idSneaker";
335         $statement = Database:::prepare($s);
336
337         $statement->bindParam(':idSneaker', $idSneaker, PDO::PARAM_INT);
338
339         $statement->execute();
340
341         $row = $statement->fetch(PDO::FETCH_ASSOC);
342
343         if ($row) {
344             return true;
345         }
346     } catch (PDOException $e) {
347         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
348     }
349
350     return false;
351 }
352
353

```

```

354     /**
355      * Supprime la sneaker en vente de la base
356      *
357      * @param integer $idSneaker // l'id de la sneaker
358      * @return void
359      */
360     function RemoveSneakerInSale(int $idSneaker): void
361     {
362         if ($self::DoesSneakerExist($idSneaker)) {
363             try {
364                 $s = "DELETE FROM TRANSACTIONS WHERE idSneaker = :idSneaker";
365                 $statement = Database::prepare($s);
366
367                 $statement->bindParam(':idSneaker', $idSneaker, PDO::PARAM_INT);
368
369                 $statement->execute();
370
371             } catch (PDOException $e) {
372                 echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
373             }
374         }
375     }
376
377     /**
378      * Récupère toutes les écoles de la base de données
379      *
380      * @return array
381      */
382     function GetAllSchools(): array
383     {
384         $arr = array();
385
386         $s = "SELECT * FROM SCHOOLS ORDER BY idSchool DESC";
387         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
388 ;
389         try {
390             $statement->execute();
391         } catch (PDOException $e) {
392             echo 'Problème de lecture de la base de données: ' . $e->getMessage();
393         }
394
395         while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
396             $u = new School($row["idSchool"], $row["nameSchool"]);
397             array_push($arr, $u);
398         }
399
400         return $arr;
401     }
402
403     /**
404      * Récupère tout les points de rendez-vous de la base
405      *
406      * @return array
407      */
408     function GetAllMeetPoints(): array
409     {
410         $arr = array();
411
412         $s = "SELECT * FROM MEETPOINTS ORDER BY idMeetPoint DESC";
413         $statement = Database::prepare($s, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));

```

```

;
413     try {
414         $statement->execute();
415     } catch (PDOException $e) {
416         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
417     }
418
419     while ($row = $statement->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
420         $u = new MeetPoint($row["idMeetPoint"], $row["nameMeetPoint"], $row["adressMeetPoint"], $row["descriptionMeetPoint"]);
421         array_push($arr, $u);
422     }
423
424     return $arr;
425 }
426
427 /**
428 * Vérifie si le point de rendez-vous existe
429 *
430 * @param MeetPoint $meetPoint // Le point de rendez vous à vérifier
431 * @return boolean // oui ou non
432 */
433 function DoesMeetPointExist(MeetPoint $meetPoint): bool
434 {
435     $nameMeetPoint = Constants::AntiXSS($meetPoint->nameMeetPoint);
436
437     $s = "SELECT * FROM MEETPOINTS WHERE nameMeetPoint = :nameMeetPoint";
438     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
439
440     try {
441         $statement->bindParam(':nameMeetPoint', $nameMeetPoint);
442         $statement->execute();
443
444         $row = $statement->fetch(PDO::FETCH_ASSOC);
445
446         if ($row) {
447             return true;
448         } else {
449             return false;
450         }
451     } catch (PDOException $e) {
452         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
453         return false;
454     }
455 }
456
457 /**
458 * Vérifie les données envoyées pour créer un point de rendez vous
459 *
460 * @param MeetPoint $meetPoint // Le point de rendez vous souhaitant être créé
461 * @return array // tableau des potentielles erreurs
462 */
463 function ErrorCreateMeetPoint(MeetPoint $meetPoint): array
464 {
465     $errorMsg = [];
466     if (self::DoesMeetPointExist($meetPoint)) {
467         $errorMsg["nameMeetPoint"] = Constants::MEET_POINT_EXIST;
468     }
469     if ($meetPoint->nameMeetPoint == "") {
470         $errorMsg["nameMeetPoint"] = Constants::NOT_EMPTY;

```

```

471     }
472     if ($meetPoint->adressMeetPoint == "") {
473         $errorMsg["adressMeetPoint"] = Constants::NOT_EMPTY;
474     }
475     if ($meetPoint->descriptionMeetPoint == "") {
476         $errorMsg["descriptionMeetPoint"] = Constants::NOT_EMPTY;
477     }
478
479     return $errorMsg;
480 }
481
482 /**
483 * Vérifie les données envoyées pour modifier un point de rendez vous
484 *
485 * @param MeetPoint $meetPoint // Le point de rendez vous souhaitant être modifié
486 * @return array // tableau des potentielles erreurs
487 */
488 function ErrorEditMeetPoint(MeetPoint $meetPoint): array
489 {
490     $errorMsg = [];
491     if ($meetPoint->nameMeetPoint == "") {
492         $errorMsg["nameMeetPoint"] = Constants::NOT_EMPTY;
493     }
494     if ($meetPoint->adressMeetPoint == "") {
495         $errorMsg["adressMeetPoint"] = Constants::NOT_EMPTY;
496     }
497     if ($meetPoint->descriptionMeetPoint == "") {
498         $errorMsg["descriptionMeetPoint"] = Constants::NOT_EMPTY;
499     }
500
501     return $errorMsg;
502 }
503
504 /**
505 * Crée le point de rendez vous dans la base de données
506 *
507 * @param MeetPoint $meetPoint // Point de rendez vous à ajouter
508 * @return boolean // L'opération a réussi ou non
509 */
510 function CreateMeetPoint(MeetPoint $meetPoint) : bool
511 {
512     try {
513         $nameMeetPoint = Constants::AntiXSS($meetPoint->nameMeetPoint);
514         $adressMeetPoint = Constants::AntiXSS($meetPoint->adressMeetPoint);
515         $descriptionMeetPoint = Constants::AntiXSS($meetPoint->descriptionMeetPoint);
516
517         $s = "INSERT INTO MEETPOINTS(nameMeetPoint, adressMeetPoint,
518 descriptionMeetPoint) VALUES(:nameMeetPoint, :adressMeetPoint, :descriptionMeetPoint)";
519         $statement = Database::prepare($s);
520         $statement->bindParam(':nameMeetPoint', $nameMeetPoint);
521         $statement->bindParam(':adressMeetPoint', $adressMeetPoint);
522         $statement->bindParam(':descriptionMeetPoint', $descriptionMeetPoint);
523
524         $statement->execute();
525
526         return true;
527     } catch (PDOException $e) {
528         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
529         return false;

```

```
530     }
531 }
532 /**
533 * Vérifie si une transaction a le point de rendez-vous indiqué
534 *
535 * @param integer $idMeetPoint // Point de rendez vous à vérifier
536 * @return boolean // Oui ou non
537 */
538 function DoesTransactionHasMeetPoint(int $idMeetPoint) : bool
539 {
540
541     $s = "SELECT * FROM TRANSACTIONS WHERE idMeetPoint = :idMeetPoint";
542     $statement = Database:::prepare($s, [PDO:::ATTR_CURSOR => PDO:::CURSOR_FWDONLY]);
543
544     try {
545         $statement->bindParam(':idMeetPoint', $idMeetPoint);
546         $statement->execute();
547
548         $row = $statement->fetch(PDO:::FETCH_ASSOC);
549
550         if ($row) {
551             return true;
552         } else {
553             return false;
554         }
555     } catch (PDOException $e) {
556         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
557         return false;
558     }
559 }
560
561 /**
562 * Supprime le point de rendez vous dans la base de données
563 *
564 * @param integer $idMeetPoint // id du point de rendez vous à supprimer
565 * @return boolean // L'opération a réussi ou non
566 */
567 function RemoveMeetPoint(int $idMeetPoint) : bool
568 {
569     if (!self::DoesTransactionHasMeetPoint($idMeetPoint)) {
570         try {
571             $s = "DELETE FROM MEETPOINTS WHERE idMeetPoint = :idMeetPoint";
572             $statement = Database:::prepare($s);
573
574             $statement->bindParam(':idMeetPoint', $idMeetPoint);
575
576             $statement->execute();
577
578             return true;
579
580         } catch (PDOException $e) {
581             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
582             return false;
583         }
584     } else {
585         return false;
586     }
587 }
588 }
589 }
```

```

590     /**
591      * Modifie le point de rendez vous souhaité dans la base de données
592      *
593      * @param MeetPoint $meetPoint // Point de rendez vous modifié
594      * @return boolean // L'opération a réussi ou non
595      */
596     function EditMeetPoint(MeetPoint $meetPoint) : bool
597     {
598         try {
599             $idMeetPoint = Constants::AntiXSS($meetPoint->idMeetPoint);
600             $nameMeetPoint = Constants::AntiXSS($meetPoint->nameMeetPoint);
601             $adressMeetPoint = Constants::AntiXSS($meetPoint->adressMeetPoint);
602             $descriptionMeetPoint = Constants::AntiXSS($meetPoint->descriptionMeetPoint);
603
604             $s = "UPDATE MEETPOINTS SET nameMeetPoint = :nameMeetPoint, adressMeetPoint =
605 :adressMeetPoint, descriptionMeetPoint = :descriptionMeetPoint WHERE idMeetPoint =
606 :idMeetPoint";
607             $statement = Database::prepare($s);
608
609             $statement->bindParam(':nameMeetPoint', $nameMeetPoint);
610             $statement->bindParam(':adressMeetPoint', $adressMeetPoint);
611             $statement->bindParam(':descriptionMeetPoint', $descriptionMeetPoint);
612             $statement->bindParam(':idMeetPoint', $idMeetPoint);
613
614             $statement->execute();
615
616         } catch (PDOException $e) {
617             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
618             return false;
619         }
620     }
621
622     /**
623      * Vérifie si les données pour modifier une école sont correctes
624      *
625      * @param School $school // L'école à modifier
626      * @return array // Le tableau des potentielles erreurs
627      */
628     function ErrorEditSchool(School $school) : array
629     {
630         $errorMsg = [];
631         if ($school->nameSchool == "") {
632             $errorMsg["nameSchool"] = Constants::NOT_EMPTY;
633         }
634
635         return $errorMsg;
636     }
637
638     /**
639      * Vérifie si l'école indiquée est affiliée à un utilisateur ou plus
640      *
641      * @param integer $idSchool // id de l'école à vérifier
642      * @return boolean // Oui ou non
643      */
644     function DoesSchoolHasUser(int $idSchool) : bool
645     {
646         $s = "SELECT * FROM USERS WHERE idSchool = :idSchool";
647         $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);

```

```

648
649     try {
650         $statement->bindParam(':idSchool', $idSchool);
651         $statement->execute();
652
653         $row = $statement->fetch(PDO::FETCH_ASSOC);
654
655         if ($row) {
656             return true;
657         } else {
658             return false;
659         }
660     } catch (PDOException $e) {
661         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
662         return false;
663     }
664 }
665
666 /**
667 * Supprime l'école souhaitée de la base
668 *
669 * @param integer $idSchool // id de l'école à supprimer
670 * @return boolean // L'opération a réussi ou non
671 */
672 function RemoveSchool(int $idSchool) : bool
673 {
674     if (!self::DoesSchoolHasUser($idSchool)) {
675         try {
676             $s = "DELETE FROM SCHOOLS WHERE idSchool = :idSchool";
677             $statement = Database::prepare($s);
678
679             $statement->bindParam(':idSchool', $idSchool);
680
681             $statement->execute();
682
683             return true;
684
685         } catch (PDOException $e) {
686             echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
687             return false;
688         }
689     } else {
690         return false;
691     }
692 }
693
694 /**
695 * Modifie l'école souhaitée dans la base de données
696 *
697 * @param School $school // école modifiée
698 * @return boolean // L'opération a réussi ou non
699 */
700 function EditSchool(School $school) : bool
701 {
702     try {
703         $idSchool = Constants::AntiXSS($school->idSchool);
704         $nameSchool = Constants::AntiXSS($school->nameSchool);
705
706         $s = "UPDATE SCHOOLS SET nameSchool = :nameSchool WHERE idSchool = :idSchool";
707

```

```

708     $statement = Database::prepare($s);
709
710     $statement->bindParam(':nameSchool', $nameSchool);
711     $statement->bindParam(':idSchool', $idSchool);
712
713     $statement->execute();
714
715     return true;
716
717 } catch (PDOException $e) {
718     echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
719     return false;
720 }
721 }
722
723 /**
724 * Vérifie si l'école indiquée existe dans la base de données
725 *
726 * @param School $school // école à vérifier
727 * @return boolean // oui ou non
728 */
729 function DoesSchoolExist(School $school) : bool
730 {
731     $nameSchool = Constants::AntiXSS($school->nameSchool);
732
733     $s = "SELECT * FROM SCHOOLS WHERE nameSchool = :nameSchool";
734     $statement = Database::prepare($s, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
735
736     try {
737         $statement->bindParam(':nameSchool', $nameSchool);
738         $statement->execute();
739
740         $row = $statement->fetch(PDO::FETCH_ASSOC);
741
742         if ($row) {
743             return true;
744         } else {
745             return false;
746         }
747     } catch (PDOException $e) {
748         echo 'Problème de lecture de la base de données: ' . $e->getMessage();
749         return false;
750     }
751 }
752
753 /**
754 * Vérifie si l'école voulant être créée est correcte
755 *
756 * @param School $school // école voulant être créée
757 * @return array // tableau des potentielles erreurs
758 */
759 function ErrorCreateSchool(School $school) : array
760 {
761     $errorMsg = [];
762     if ($self::DoesSchoolExist($school)) {
763         $errorMsg["nameSchool"] = Constants::SCHOOL_EXIST;
764     }
765     if ($school->nameSchool == "") {
766         $errorMsg["nameSchool"] = Constants::NOT_EMPTY;
767     }

```

```

768     return $errorMsg;
769 }
770
771 /**
772 * Crée l'école dans la base de données
773 *
774 * @param School $school // école à créer
775 * @return boolean // L'opération a réussi ou non
776 */
777 function CreateSchool(School $school) : bool
778 {
779     try {
780         $nameSchool = Constants::AntiXSS($school->nameSchool);
781
782         $s = "INSERT INTO SCHOOLS(nameSchool) VALUES(:nameSchool)";
783         $statement = Database::prepare($s);
784         $statement->bindParam(':nameSchool', $nameSchool);
785
786
787         $statement->execute();
788
789         return true;
790
791     } catch (PDOException $e) {
792         echo 'Problème d\'insertion en base de données: ' . $e->getMessage();
793         return false;
794     }
795 }
796 }
```

src\Model\objects\ViewTransaction.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet User
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
11
12 class ViewTransaction
13 {
14     // Constructeur de l'objet
15     public function __construct(int|null $idTransaction = 0, Sneaker|null $sneaker = new
16     Sneaker(), User|null $acheteur = new User(), User|null $vendeur = new User(), string|null
17     $nameStatut = "", MeetPoint|null $meetPoint = new MeetPoint())
18     {
19         $this->idTransaction = $idTransaction;
20         $this->sneaker = $sneaker;
21         $this->acheteur = $acheteur;
22         $this->vendeur = $vendeur;
23         $this->nameStatut = $nameStatut;
24         $this->meetPoint = $meetPoint;
25     }
26 }
```

```

26  /**
27   * Id de la transaction
28   *
29   * @var integer
30   */
31 public $idTransaction;
32
33 /**
34  * Paire concernée
35  *
36  * @var Sneaker
37  */
38 public $sneaker;
39
40 /**
41  * Acheteur de la paire
42  *
43  * @var User
44  */
45 public $acheteur;
46
47 /**
48  * Vendeur de la paire
49  *
50  * @var User
51  */
52 public $vendeur;
53
54 /**
55  * statut de la transaction
56  *
57  * @var string
58  */
59 public $nameStatut;
60
61 /**
62  * point de rencontre
63  *
64  * @var string
65  */
66 public $meetPoint;
67
68 }

```

src\Model\objects\User.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet User
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
11
12 class User
13 { // Constructeur de l'objet

```

```
14  public function __construct(int|null $idUser = 0, string|null $nameUser = "",  
15  string|null $surnameUser = "", string|null $emailUser = "", string|null $pwdUser = "",  
16  int|null $idSchool = 0, string|null $imgUser = "", int|null $idRole = 1)  
17  {  
18      $this->idUser = $idUser;  
19      $this->nameUser = $nameUser;  
20      $this->surnameUser = $surnameUser;  
21      $this->emailUser = $emailUser;  
22      $this->pwdUser = $pwdUser;  
23      $this->idSchool = $idSchool;  
24      $this->imgUser = $imgUser;  
25      $this->idRole = $idRole;  
26  
27  /**  
28  * Id de l'utilisateur  
29  *  
30  * @var integer  
31  */  
32  public $idUser;  
33  
34  /**  
35  * Nom de l'utilisateur  
36  *  
37  * @var string  
38  */  
39  public $nameUser;  
40  
41  /**  
42  * prénom de l'utilisateur  
43  *  
44  * @var string  
45  */  
46  public $surnameUser;  
47  
48  /**  
49  * email de l'utilisateur  
50  *  
51  * @var string  
52  */  
53  public $emailUser;  
54  
55  /**  
56  * mot de passe hashé de l'utilisateur  
57  *  
58  * @var string  
59  */  
60  public $pwdUser;  
61  
62  /**  
63  * id de l'école de l'utilisateur  
64  *  
65  * @var integer  
66  */  
67  public $idSchool;  
68  
69  /**  
70  * Photo de profil de l'utilisateur  
71  */
```

```

72     * @var string
73     */
74     public $imgUser;
75
76     /**
77      * id du role de l'utilisateur
78      *
79      * @var integer
80      */
81     public $idRole;
82
83 }
```

src\Model\objects\Sneaker.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet Sneaker
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
11
12 class Sneaker
13 {
14     // Constructeur de l'objet
15     public function __construct(int|null $idSneaker = 0, int|null $sizeSneaker = 0,
16     float|null $priceSneaker = 0.0, string|null $imgSneaker = "", string|null
17     $descriptionSneaker = "", string|null $nameSStatut = "", string|null $nameBrand = "",
18     string|null $nameModel = "", string|null $nameVisibility = "")
19     {
20         $this->idSneaker = $idSneaker;
21         $this->sizeSneaker = $sizeSneaker;
22         $this->priceSneaker = $priceSneaker;
23         $this->imgSneaker = $imgSneaker;
24         $this->descriptionSneaker = $descriptionSneaker;
25         $this->nameSStatut = $nameSStatut;
26         $this->nameBrand = $nameBrand;
27         $this->nameModel = $nameModel;
28         $this->nameVisibility = $nameVisibility;
29     }
30
31     /**
32      * Id de la sneaker
33      *
34      * @var integer
35      */
36     public $idSneaker;
37
38     /**
39      * taille de la sneaker
40      *
41      * @var integer
42      */
43     public $sizeSneaker;
```

```

42
43 /**
44 * prix de la sneaker
45 *
46 * @var float
47 */
48 public $priceSneaker;
49
50 /**
51 * image de la sneaker
52 *
53 * @var string
54 */
55 public $imgSneaker;
56
57 /**
58 * description de la sneaker
59 *
60 * @var string
61 */
62 public $descriptionSneaker;
63
64 /**
65 * nom du statut de la sneaker
66 *
67 * @var string
68 */
69 public $nameSStatut;
70 /**
71 * nom de la marque de la sneaker
72 *
73 * @var string
74 */
75 public $nameBrand;
76
77 /**
78 * nom du modèle de la sneaker
79 *
80 * @var string
81 */
82 public $nameModel;
83
84 /**
85 * nom de la visibilité de la sneaker
86 *
87 * @var string
88 */
89 public $nameVisibility;
90
91 }

```

src\Model\objects\Modele.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers

```

```

7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet Modèle
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
11
12 class Modele
13 {
14     // Constructeur de l'objet
15     public function __construct(int|null $idModel = 0, string|null $nameModel = "")
16     {
17         $this->idModel = $idModel;
18         $this->nameModel = $nameModel;
19     }
20
21     /**
22      * Id du modèle
23      *
24      * @var integer
25      */
26     public $idModel;
27
28     /**
29      * Nom du modèle
30      *
31      * @var string
32      */
33     public $nameModel;
34 }
35 }
```

src\Model\objects\School.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet Ecole
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
11
12 class School
13 {
14     // Constructeur de l'objet
15     public function __construct(int|null $idSchool = 0, string|null $nameSchool = "")
16     {
17         $this->idSchool = $idSchool;
18         $this->nameSchool = $nameSchool;
19     }
20
21     /**
22      * Id de le l'école
23      *
24      * @var integer
25      */
26     public $idSchool;
27 }
```

```

28 /**
29  * Nom de l'école
30  *
31  * @var string
32  */
33 public $nameSchool;
34
35 }
```

src\Model\objects\Brand.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
9 Page : Objet Marque
10 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
11
12 class Brand
13 {
14     // Constructeur de l'objet
15     public function __construct(int $idBrand = 0, string $nameBrand = "")
16     {
17         $this->idBrand = $idBrand;
18         $this->nameBrand = $nameBrand;
19     }
20
21     /**
22      * Id de la marque
23      *
24      * @var integer
25      */
26     public $idBrand;
27
28     /**
29      * Nom de la marque
30      *
31      * @var string
32      */
33     public $nameBrand;
34
35 }
```

src\Model\objects\MeetPoint.php

```

1 <?php
2
3 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
4 Projet : TPI
5 Auteur : Léo Triano
6 Desc. : Site de Sneakers
7 Version : 1.0
8 Date : Mai 2024
```

```

9 | Page : Objet Point de rendez vous
10| * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
11|
12| class MeetPoint
13| {
14|     // Constructeur de l'objet
15|     public function __construct(int|null $idMeetPoint = 0, string|null $nameMeetPoint = "",  

16|     string|null $adressMeetPoint = "", string|null $descriptionMeetPoint = "",  

17|     DateTime|string|null $dateHourMeetPlace = new DateTime())
18|     {
19|         $this->idMeetPoint = $idMeetPoint;
20|         $this->nameMeetPoint = $nameMeetPoint;
21|         $this->adressMeetPoint = $adressMeetPoint;
22|         $this->descriptionMeetPoint = $descriptionMeetPoint;
23|         $this->dateHourMeetPlace = $dateHourMeetPlace;
24|     }
25|
26| /**
27| * Id du point de rencontre
28| *
29| * @var integer
30| */
31| public $idMeetPoint;
32|
33| /**
34| * Nom du point de rendez vous
35| *
36| * @var string
37| */
38| public $nameMeetPoint;
39|
40| /**
41| * Adresse du point de rendez vous
42| *
43| * @var string
44| */
45| public $adressMeetPoint;
46|
47| /**
48| * Description du point de rendez vous
49| *
50| * @var string
51| */
52| public $descriptionMeetPoint;
53|
54| /**
55| * Date et heure du rendez vous
56| *
57| * @var DateTime
58| */
59| public $dateHourMeetPlace;

```

src\Controller\ModifyUserController.php

```

1| <?php
2|
3| if (!isset($_SESSION['idUser']) || $_SESSION['idRole'] != 2) {

```

```

4     header("Location: index.php?url=Home");
5     exit;
6 }
7
8 // Récupère le modèle
9 require("./src/Model/Constants.php");
10
11 require ('./src/Model/ModifyUserModel.php');
12 require ('./src/Model/objects/School.php');
13 require ('./src/Model/objects/User.php');
14
15
16
17 $modifyUserInstance = new ModifyUser();
18 $schools = $modifyUserInstance->GetAllSchools();
19 $idUser = filter_input(INPUT_POST, 'idUser', FILTER_SANITIZE_NUMBER_INT);
20 $submit = filter_input(INPUT_POST, 'submitRegister', FILTER_SANITIZE_STRING);
21 $errorMessages = [];
22
23
24 if ($idUser == "" || !$modifyUserInstance->DoesUserExist($idUser)) {
25     header("Location: index.php?url=Home");
26 } else {
27     $idSchool = $modifyUserInstance->GetUserById($idUser)->idSchool;
28     $idUser = $modifyUserInstance->GetUserById($idUser)->idUser;
29     $nameUser = $modifyUserInstance->GetUserById($idUser)->nameUser;
30     $emailUser = $modifyUserInstance->GetUserById($idUser)->emailUser;
31     $surnameUser = $modifyUserInstance->GetUserById($idUser)->surnameUser;
32     $imgUtilisateur = $modifyUserInstance->GetUserById($idUser)->imgUser;
33 }
34
35
36 if ($submit == "Submit") {
37     $idSchool = filter_input(INPUT_POST, 'school', FILTER_VALIDATE_INT);
38     $nameUser = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
39     $emailUser = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_STRING);
40     $surnameUser = filter_input(INPUT_POST, 'surname', FILTER_SANITIZE_STRING);
41
42     $user = new User($idUser, $nameUser, $surnameUser, $emailUser, "", $idSchool);
43     $errorMessages = $modifyUserInstance->ErrorMessagesRegister($user, $idSchool, $_FILES["img"]);
44     if (empty($errorMessages)) {
45         if ($modifyUserInstance->UpdateProfile($user, $_FILES["img"])) {
46             $_SESSION["customMessage"] = "User's profile updated successfully";
47             header("Location: index.php?url=Admin");
48             exit;
49         }
50     }
51 }
52
53
54 // Récupère la vue
55 require ('./src/Controller/headerController.php');
56 require ('./src/View/ModifyUserView.php');
57

```

src\Controller\ProfileController.php

1 | <?php

```

2
3 if (!isset($_SESSION['idUser'])) {
4     header("Location: index.php?url=Home");
5     exit;
6 }
7
8 require ("./src/Model/Constants.php");
9
10 require ('./src/Model/ProfileModel.php');
11 require ('./src/Model/objects/School.php');
12 require ('./src/Model/objects/User.php');
13
14
15 $profileInstance = new Profile();
16 $schools = $profileInstance->GetAllSchools();
17 $submitRegister = filter_input(INPUT_POST, 'submitRegister', FILTER_SANITIZE_STRING);
18 $errorMessages = [];
19
20 $idSchool = $_SESSION['idSchool'];
21 $nameUser = $_SESSION['nameUser'];
22 $surnameUser = $_SESSION['surnameUser'];
23 $emailUser = $_SESSION['emailUser'];
24
25 if ($submitRegister == "Submit") {
26     $idSchool = filter_input(INPUT_POST, 'school', FILTER_VALIDATE_INT);
27     $nameUser = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
28     $emailUser = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_STRING);
29     $surnameUser = filter_input(INPUT_POST, 'surname', FILTER_SANITIZE_STRING);
30
31
32     $user = new User($_SESSION['idUser'], $nameUser, $surnameUser, $emailUser, "", $idSchool);
33
34     $errorMessages = $profileInstance->ErrorMessagesRegister($user, $idSchool, $_FILES['img']);
35     if (empty($errorMessages)) {
36         if ($profileInstance->UpdateProfile($user, $_FILES["img"])) {
37             $_SESSION["customMessage"] = "Profile updated successfully";
38             header("Location: index.php?url=Profile");
39         }
40     }
41 } else if ($submitRegister == "Delete profile picture") {
42     $profileInstance->DeleteProfilePicture();
43
44     $_SESSION["customMessage"] = "Profile updated successfully";
45     header("Location: index.php?url=Profile");
46 }
47
48 require ('./src/Controller/headerController.php');
49
50 require ('./src/View/ProfileView.php');
51
52 if (isset($_SESSION["customMessage"]) && $submitRegister == "") {
53     $_SESSION["customMessage"] = "";
54 }
55
56

```

src\Controller\ResultSearchController.php

```

1 <?php
2
3 require("./src/Model/Constants.php");
4 include('./src/Controller/headerController.php');
5
6 $headerInstance = new Header();
7 $search = "";
8
9 if(isset($_GET["search"])){
10     $search = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_STRING);
11     $search = Constants::AntiXSS($search);
12
13     if(isset($_GET['page']) && !empty($_GET['page'])){
14         $currentPage = (int)$_GET['page'];
15     }else{
16         $currentPage = 1;
17     }
18
19     $nbArticles = $headerInstance->CountSneakers($search);
20     $parPage = 10;
21     $pages = ceil($nbArticles / $parPage);
22     $premier = ($currentPage * $parPage) - $parPage;
23
24     $tabPairesCards = $headerInstance->SearchBar($search, $premier, $parPage);
25 }
26 else{
27     header("Location: index.php?url=Home");
28 }
29
30 include('./src/View/ResultSearchView.php');
```

src\Controller\LogoutController.php

```

1 <?php
2
3
4 // Détruit la session
5 session_start();
6 $_SESSION = [];
7 if (ini_get("session.use_cookies")) {
8     setcookie(session_name(), '', 0);
9 }
10 session_destroy();
11
12 header("Location: index.php?url=Login");
13
14
```

src\Controller\ManageUserController.php

```

1 <?php
2
3 if (!isset($_SESSION['idUser']) || $_SESSION['idRole'] != 2) {
4     header("Location: index.php?url=Home");
5     exit;
6 }
7
```

```

8 require("./src/Model/Constants.php");
9
10 require ('./src/Model/ManageUserModel.php');
11 require ('./src/Model/objects/Sneaker.php');
12 require ('./src/Model/objects/User.php');
13 require ('./src/Model/objects/ViewTransaction.php');
14 require ('./src/Model/objects/MeetPoint.php');
15
16
17
18 $idUser = filter_input(INPUT_GET, 'idUser', FILTER_SANITIZE_NUMBER_INT);
19 $manageUserInstance = new ManageUser();
20
21 $currentPageSneaker = isset($_GET['page']) ? (int) $_GET['page'] : 1;
22 $parPageSneaker = 10;
23 $currentPageSneaker = isset($_GET['pageSneaker']) ? (int) $_GET['pageSneaker'] : 1;
24 $nbSneakers = $manageUserInstance->CountSneakers($idUser);
25 $pagesSneaker = ceil($nbSneakers / $parPageSneaker);
26 $premierSneaker = ($currentPageSneaker * $parPageSneaker) - $parPageSneaker;
27
28 $currentUser = $manageUserInstance->GetUserById($idUser);
29 $shoesInSale = $manageUserInstance->GetAllShoesInSale($idUser, $premierSneaker,
30 $parPageSneaker);
31 $sneakersSold = $manageUserInstance->SneakersSold($idUser);
32
33 if (!$currentUser || $idUser == ""){
34     header("Location: index.php?url=Home");
35     exit;
36 }
37
38 if (isset($_GET["remove"]) && isset($_GET["idSneaker"])) {
39     $idSneaker = filter_input(INPUT_GET, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
40     $manageUserInstance->RemoveSneakerInSale($idSneaker);
41     $_SESSION["customMessage"] = "aaaaaaSneaker successfully removed";
42     header("Location: index.php?url=ManageUser&idUser=".$idUser);
43     exit;
44 }
45
46
47 // Récupère la vue
48 require ('./src/Controller/headerController.php');
49 require ('./src/View/ManageUserView.php');
50

```

src\Controller\HomeController.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : TPI
4 Auteur : Léo Triano
5 Desc. : Site de Sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Controller de la page Shop
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
10
11
12

```

```

13 // Récupère le modèle
14 require("./src/Model/Constants.php");
15
16 require ('./src/Model/HomeModel.php');
17 require ('./src/Model/objects/Sneaker.php');
18 require ('./src/Model/objects/User.php');
19 require ('./src/Model/objects/ViewTransaction.php');
20 require ('./src/Model/objects/Brand.php');
21 require ('./src/Model/objects/MeetPoint.php');
22
23
24
25
26 // Instancie un nouvel objet shop
27 $homeInstance = new Home();
28
29 // Récupère toutes les paires
30 $marques = $homeInstance->GetAllBrands();
31 $sizes = $homeInstance->GetAllSizes();
32
33
34 $idBrand = filter_input(INPUT_GET, 'marques', FILTER_SANITIZE_NUMBER_INT);
35 $idSize = filter_input(INPUT_GET, 'sizes', FILTER_SANITIZE_NUMBER_INT);
36
37
38 // Pagination des paires, tuto suivi et adapté au besoin
39 // https://nouvelle-techno.fr/articles/mettre-en-place-une-pagination-en-php
40 if(isset($_GET['page']) && !empty($_GET['page'])){
    $currentPage = (int)$_GET['page'];
41 }else{
    $currentPage = 1;
42 }
43
44
45
46 $nbArticles = $homeInstance->CountSneakers($idBrand);
47 $parPage = 10;
48 $pages = ceil($nbArticles / $parPage);
49 $premier = ($currentPage * $parPage) - $parPage;
50
51
52 $tabPairesCards = $homeInstance->GetAllSneakers($premier, $parPage);
53
54 if($idBrand != ""){
    $tabPairesCards = $homeInstance->FilterByBrand($idBrand, $premier, $parPage, $idSize);
55 }
56
57
58 // Récupère la vue
59 require('./src/Controller/headerController.php');
60 require ('./src/View/HomeView.php');
61
62 if(isset($_SESSION["customMessage"])){
    $_SESSION["customMessage"] = "";
63 }
64 }
```

src\Controller\headerController.php

```

1 <?php
2 if ($redirection == "ResultSearch") {
3     require ('./src/Model/objects/Sneaker.php');
4     require ('./src/Model/objects/User.php');
```

```

5   require ('./src/Model/objects/ViewTransaction.php');
6   require ('./src/Model/objects/Brand.php');
7   require ('./src/Model/objects/MeetPoint.php');
8
9 }
10
11 include ('./src/Model/headerModel.php');
12
13
14 $headerInstance = new Header();
15
16 if (isset($_POST["searchBar"])) {
17     var_dump($_POST["searchBar"]);
18     $search = filter_input(INPUT_POST, 'searchBar', FILTER_SANITIZE_STRING);
19     $search = Constants::AntiXSS($search);
20
21     header("Location: index.php?url=ResultSearch&search=" . $search);
22 }
23
24 include ('./src/View/HeaderView.php');
```

src\Controller\GestionController.php

```

1 <?php
2
3 if (!isset($_SESSION['idUser'])) {
4     header("Location: index.php?url=Home");
5     exit;
6 }
7 require ("./src/Model/Constants.php");
8
9 require ('./src/Model/GestionModel.php');
10
11 require ('./src/Model/objects/Brand.php');
12 require ('./src/Model/objects/Modele.php');
13 require ('./src/Model/objects/Sneaker.php');
14 require ('./src/Model/objects/User.php');
15 require ('./src/Model/objects/MeetPoint.php');
16 require ('./src/Model/objects/ViewTransaction.php');
17
18
19
20
21 $gestionIsntance = new Gestion();
22
23 $marques = $gestionIsntance->GetAllBrands();
24 $modeles = $gestionIsntance->GetAllModeles();
25
26 $size = filter_input(INPUT_POST, 'size', FILTER_SANITIZE_NUMBER_INT);
27 $price = filter_input(INPUT_POST, 'price', FILTER_SANITIZE_NUMBER_INT);
28 $desc = filter_input(INPUT_POST, 'desc', FILTER_SANITIZE_STRING);
29 $idBrand = filter_input(INPUT_POST, 'brand', FILTER_SANITIZE_NUMBER_INT);
30 $modele = filter_input(INPUT_POST, 'modele', FILTER_SANITIZE_STRING);
31 $submit = filter_input(INPUT_POST, 'submit', FILTER_SANITIZE_STRING);
32 $idSneaker = filter_input(INPUT_GET, 'idSneaker', FILTER_VALIDATE_INT);
33 $statut = filter_input(INPUT_POST, 'statut', FILTER_VALIDATE_INT);
34
35 $errorMessages = [];
36
```

```

37 $tabPairesEnVente = $gestionIsntance->GetAllShoesInSale();
38 $tabPairesAchetees = $gestionIsntance->GetAllShoesPurchased();
39 $arrSneakersReserved = $gestionIsntance-> GetAllShoesReserved();
40 $sneakersReserved = $gestionIsntance-> SneakersReserved();
41 $sneakersSold = $gestionIsntance-> SneakersSold();

42
43 switch ($submit) {
44     case "Create":
45         $errorMessages = $gestionIsntance->CheckErrorsCreateTransaction($size, $price,
46 $_FILES["img"], $desc, $modele, $idBrand);
47         if (empty($errorMessages)) {
48             $gestionIsntance->CreateTransaction($size, $price, $_FILES["img"], $desc,
49 $modele, $idBrand, $statut);
50             $_SESSION["customMessage"] = "Transaction added with success";
51             header("Location: index.php?url=Gestion");
52             exit;
53         }
54         break;
55     case "Cancel":
56         $idSneaker = filter_input(INPUT_POST, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);

57         if ($gestionIsntance->CancelReserve($idSneaker)) {
58             $_SESSION["customMessage"] = "Reservation successfully removed";
59             header("Location: index.php?url=Gestion#reservations");
60             exit;
61         }
62         break;

63     case "Accept":
64         $idSneaker = filter_input(INPUT_POST, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
65         if ($gestionIsntance->AcceptReservation($idSneaker)) {
66             $_SESSION["customMessage"] = "Sneaker successfully sold";
67             header("Location: index.php?url=Gestion#reserved");
68             exit;
69         }
70         break;

71     case "Deny":
72         $idSneaker = filter_input(INPUT_POST, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
73         if ($gestionIsntance->DenyReservation($idSneaker)) {
74             $gestionIsntance->RemoveFavoritesSneakerSold($idSneaker);
75             $_SESSION["customMessage"] = "purchase offer successfully canceled";
76             header("Location: index.php?url=Gestion#reserved");
77             exit;
78         }
79         break;

80     case "Set to Visible":
81         $idSneaker = filter_input(INPUT_POST, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
82         $gestionIsntance->SetSneakerToVisible($idSneaker);

83         $_SESSION["customMessage"] = "Sneaker successfully set to visible";
84         header("Location: index.php?url=Gestion#sale");
85         exit;
86     }

87 if (isset($_GET["remove"]) && isset($_GET["idSneaker"])) {
88     $idSneaker = filter_input(INPUT_GET, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
89
90     $gestionIsntance->RemoveSneakerInSale($idSneaker);
91 }
```

```

95     $_SESSION["customMessage"] = "Sneaker successfully removed";
96     header("Location: index.php?url=Gestion#sale");
97     exit;
98 }
99
100 require ('./src/Controller/headerController.php');
101 include ('./src/View/GestionView.php');
102
103 if ($_SESSION["customMessage"] != "") {
104     $_SESSION["customMessage"] = "";
105 }
106

```

src\Controller\FavoritesController.php

```

1 <?php
2
3 if (!isset($_SESSION['idUser'])) {
4     header("Location: index.php?url=Home");
5     exit;
6 }
7 require("./src/Model/Constants.php");
8
9 require ('./src/Model/FavoritesModel.php');
10
11 require ('./src/Model/objects/Brand.php');
12 require ('./src/Model/objects/Modele.php');
13 require ('./src/Model/objects/Sneaker.php');
14 require ('./src/Model/objects/User.php');
15 require ('./src/Model/objects/MeetPoint.php');
16 require ('./src/Model/objects/ViewTransaction.php');
17
18
19 $favoritesInstance = new Favorites();
20 $userFavList = $favoritesInstance->UserFavList();
21 $idSneaker = filter_input(INPUT_GET, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
22
23 if (isset($_GET["favorite"])) {
24     $favoritesInstance->RemoveFromFavorite($idSneaker, $_SESSION['idUser']);
25     $_SESSION["customMessage"] = "Sneaker removed from favorites successfully";
26     header("Location: index.php?url=Favorites");
27     exit;
28 }
29
30 // Pagination des paires, tuto suivi et adapté au besoin
31 // https://nouvelle-techno.fr/articles/mettre-en-place-une-pagination-en-php
32 if (isset($_GET['page']) && !empty($_GET['page'])) {
33     $currentPage = (int) $_GET['page'];
34 } else {
35     $currentPage = 1;
36 }
37
38 $nbArticles = $favoritesInstance->CountSneakers();
39 $parPage = 10;
40 $pages = ceil($nbArticles / $parPage);
41 $premier = ($currentPage * $parPage) - $parPage;
42
43 require ('./src/Controller/headerController.php');
44 include ('./src/View/FavoritesView.php');

```

```

45
46 if(isset($_SESSION["customMessage"])){
47     $_SESSION["customMessage"] = "";
48 }

```

src\Controller\DetailController.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : TPI
4 Auteur : Léo Triano
5 Desc. : Site de Sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Controller de la page Detail
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10
11 // Récupère le modèle
12 require("./src/Model/Constants.php");
13
14 require ('./src/Model/DetailModel.php');
15 require ('./src/Model/objects/Sneaker.php');
16 require ('./src/Model/objects/MeetPoint.php');
17
18
19 // Récupère les paramètres envoyés en get
20 if (filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT)) {
21
22     $idSneaker = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
23     $addToFavorite = filter_input(INPUT_GET, 'favorite', FILTER_SANITIZE_STRING);
24     $submit = filter_input(INPUT_POST, 'submit', FILTER_SANITIZE_STRING);
25     $idMeetPoint = filter_input(INPUT_POST, 'meetPoint', FILTER_SANITIZE_NUMBER_INT);
26     $meetingDate = filter_input(INPUT_POST, 'meetingDate', FILTER_SANITIZE_STRING);
27     $formatted_datetime = date("Y-m-d H:i", strtotime($meetingDate));
28
29     // Instancie un nouvel objet shop
30     $detailInstance = new Detail();
31
32     // Récupère toutes les paires
33     $paire = $detailInstance->GetSneakerById($idSneaker);
34     $isFavorite = "";
35     $meetPoints = $detailInstance->GetAllMeetPoint();
36     $errorMessages = [];
37
38
39     // Si l'id ne correspond pas à une paire, retourner sur shop
40     if (!$paire) {
41         header("Location: index.php?url=Shop");
42     }
43
44     if (isset($_SESSION["idUser"])) {
45         // Vérifie si la paire est en favorite ou non
46         $isFavorite = $detailInstance->CheckIfFavorite($idSneaker, $_SESSION['idUser']);
47
48         if ($addToFavorite == "yes") {
49             if (!$isFavorite) {
50                 $detailInstance->AddToFavorite($idSneaker, $_SESSION['idUser']);
51                 $_SESSION["customMessage"] = "Sneaker added to favorites successfully";
52             } else {

```

```

53             $detailInstance->RemoveFromFavorite($idSneaker, $_SESSION['idUser']);
54             $_SESSION["customMessage"] = "Sneaker removed from favorites successfully"
55         ;
56     }
57     header("Location: index.php?url=Detail&id=" . $idSneaker);
58     exit;
59 }
60
61 if ($submit == "Reserve") {
62     $errorMessages = $detailInstance->CheckErrorReserve($idMeetPoint,
63     $formatted_datetime);
64     if (empty($errorMessages)) {
65         if (!$detailInstance->IsReserved($idSneaker)) {
66             $detailInstance->Reserve($idSneaker, $idMeetPoint, $formatted_datetime);
67             $_SESSION["customMessage"] = "Sneaker successfully reserved";
68         }
69     else{
70         $_SESSION["customError"] = "We're sorry, another user already reserved
this pair";
71     }
72     header("Location: index.php?url=Detail&id=" . $idSneaker);
73     exit;
74 }
75 else if ($submit == "Cancel") {
76     if ($detailInstance->IsReserved($idSneaker)) {
77         if($detailInstance->doesUserHasReserved($idSneaker)){
78             $detailInstance->CancelReserve($idSneaker);
79             $_SESSION["customMessage"] = "Reservation successfully canceled";
80         }
81     }
82     header("Location: index.php?url=Detail&id=" . $idSneaker);
83     exit;
84 }
85 // Récupère la vue
86 require ('./src/View/DetailView.php');
87 } else {
88     header("Location: index.php?url=Shop");
89 }
90
91
92 if (isset($_SESSION["customMessage"])) {
93     $_SESSION["customMessage"] = "";
94 }
95
96 if (isset($_SESSION["customError"])) {
97     $_SESSION["customError"] = "";
98 }
99
100
101

```

src\Controller\AuthController.php

```

1 <?php
2 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3 Projet : TPI
4 Auteur : Léo Triano

```

```
5 Desc. : Site de Sneakers
6 Version : 1.0
7 Date : Mai 2024
8 Page : Controller de la page Login
9 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10
11
12
13 // Récupère le modèle
14 require("./src/Model/Constants.php");
15
16 require("./src/Model/objects/User.php");
17 require("./src/Model/objects/School.php");
18 require ('./src/Model/AuthModel.php');
19
20
21 // Instancie un nouvel objet shop
22 $authInstance = new Auth();
23
24 $nameUser = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_STRING);
25 $emailUser= filter_input(INPUT_POST, 'email', FILTER_SANITIZE_STRING);
26 $pwdUser = filter_input(INPUT_POST, 'password', FILTER_SANITIZE_STRING);
27 $surnameUser = filter_input(INPUT_POST, 'surname', FILTER_SANITIZE_STRING);
28 $idSchool = filter_input(INPUT_POST, 'school', FILTER_VALIDATE_INT);
29
30 $submitRegister = filter_input(INPUT_POST, 'submitRegister', FILTER_SANITIZE_STRING);
31 $submitLogin = filter_input(INPUT_POST, 'submitLogin', FILTER_SANITIZE_STRING);
32
33 $user = new User();
34 $errorMessages = [];
35 $schools = $authInstance->GetAllSchools();
36
37
38 if($submitRegister == "Submit"){
39     $user = new User(0, $nameUser, $surnameUser, $emailUser, $pwdUser);
40
41     $errorMessages = $authInstance->ErrorMessagesRegister($user, $idSchool, $_FILES['img'])
42 ;
43
44     if(empty($errorMessages)){
45         $authInstance->Register($user, $idSchool, $_FILES["img"]);
46         $img = Constants::ConvertImgTo64($_FILES["img"]);
47         $authInstance->ValuesSession(new User(Database::lastInsertId(), $nameUser,
48 $surnameUser, $emailUser, "", $idSchool, $img));
49         header("Location: index.php?url=Home");
50     }
51 }
52 else if ($submitLogin == "Submit"){
53     $user = new User(0, "", "", $emailUser, $pwdUser);
54
55     $errorMessages = $authInstance->ErrorMessagesLogin($user);
56
57     if(empty($errorMessages)){
58         $authInstance->login($user->emailUser);
59         header("Location: index.php?url=Home");
60     }
61 }
62 // Récupère la vue
63 require ('./src/View/AuthView.php');
```

63
64
65
66
67
68

src\Controller\AdminController.php

```
1 <?php
2
3 if (!isset($_SESSION['idUser']) || $_SESSION['idRole'] != 2) {
4     header("Location: index.php?url=Home");
5     exit;
6 }
7 require ("./src/Model/Constants.php");
8
9 require ('./src/Model/AdminModel.php');
10 require ('./src/Model/objects/User.php');
11 require ('./src/Model/objects/Sneaker.php');
12 require ('./src/Model/objects/ViewTransaction.php');
13 require ('./src/Model/objects/MeetPoint.php');
14
15 require ('./src/Model/AuthModel.php');
16 require ('./src/Model/objects/School.php');
17
18
19 // Validation des données d'entrée
20 $searchBarUsers = filter_input(INPUT_POST, 'searchBarUsers', FILTER_SANITIZE_STRING);
21 $idUser = filter_input(INPUT_POST, 'idUser', FILTER_SANITIZE_NUMBER_INT);
22 $submit = filter_input(INPUT_POST, 'submit', FILTER_SANITIZE_STRING);
23 $nameUser = filter_input(INPUT_POST, 'nameUser', FILTER_SANITIZE_STRING);
24 $surnameUser = filter_input(INPUT_POST, 'surnameUser', FILTER_SANITIZE_STRING);
25 $emailUser = filter_input(INPUT_POST, 'emailUser', FILTER_SANITIZE_STRING);
26 $pwdUser = filter_input(INPUT_POST, 'pwdUser', FILTER_SANITIZE_STRING);
27 $idSchool = filter_input(INPUT_POST, 'idSchool', FILTER_SANITIZE_NUMBER_INT);
28 $nameMeetPoint = filter_input(INPUT_POST, 'nameMeetPoint', FILTER_SANITIZE_STRING);
29 $adressMeetPoint = filter_input(INPUT_POST, 'adressMeetPoint', FILTER_SANITIZE_STRING);
30 $descriptionMeetPoint = filter_input(INPUT_POST, 'descriptionMeetPoint',
31 FILTER_SANITIZE_STRING);
32 $idMeetPoint = filter_input(INPUT_POST, 'idMeetPoint', FILTER_SANITIZE_NUMBER_INT);
33
34 $editNameMeetPoint = filter_input(INPUT_POST, 'editNameMeetPoint', FILTER_SANITIZE_STRING);
35 $editAdressMeetPoint = filter_input(INPUT_POST, 'editAdressMeetPoint',
36 FILTER_SANITIZE_STRING);
37 $editDescriptionMeetPoint = filter_input(INPUT_POST, 'editDescriptionMeetPoint',
38 FILTER_SANITIZE_STRING);
39
40
41 // Création d'instances
42 $adminInstance = new Admin();
43 $authInstance = new Auth();
44
45 $errorMessages = [];
46 $schools = $authInstance->GetAllSchools();
```

```

48 // Logique de pagination
49 $currentPage = isset($_GET['page']) ? (int) $_GET['page'] : 1;
50 $parPage = 8;
51
52 // Logique de pagination pour les utilisateurs
53 $nbArticles = $adminInstance->CountUsers($searchBarUsers);
54 $pages = ceil($nbArticles / $parPage);
55 $premier = ($currentPage * $parPage) - $parPage;
56
57 $users = $adminInstance-> GetAllUsers($searchBarUsers, $premier, $parPage);
58
59 // Logique de pagination pour les sneakers
60 $parPageSneaker = 10;
61 $currentPageSneaker = isset($_GET['pageSneaker']) ? (int) $_GET['pageSneaker'] : 1;
62 $nbSneakers = $adminInstance->CountSneakers();
63 $pagesSneaker = ceil($nbSneakers / $parPageSneaker);
64 $premierSneaker = ($currentPageSneaker * $parPageSneaker) - $parPageSneaker;
65
66 $shoesInSale = $adminInstance-> GetAllShoesInSale($premierSneaker, $parPageSneaker);
67
68 switch ($submit) {
69     case "Block":
70         if ($adminInstance->BlockUser($idUser)) {
71             $_SESSION["customMessage"] = "User successfully blocked";
72             header("Location: index.php?url=Admin#users");
73             exit;
74         }
75         break;
76
77     case "Delete":
78         if ($adminInstance->RemoveBuyingOffers($idUser)) {
79             if ($adminInstance->DeleteUser($idUser)) {
80                 $_SESSION["customMessage"] = "User successfully deleted";
81                 header("Location: index.php?url=Admin#users");
82                 exit;
83             }
84         }
85         break;
86
87     case "Unblock":
88         if ($adminInstance->UnblockUser($idUser)) {
89             $_SESSION["customMessage"] = "User successfully unblocked";
90             header("Location: index.php?url=Admin#users");
91             exit;
92         }
93         break;
94
95     case "Register":
96         $user = new User(0, $nameUser, $surnameUser, $emailUser, $pwdUser);
97         $errorMessages = $authInstance->ErrorMessagesRegister($user, $idSchool, $_FILES["imgUser"]);
98         if (empty($errorMessages)) {
99             $authInstance->Register($user, $idSchool, $_FILES["imgUser"]);
100            $_SESSION["customMessage"] = "User created successfully";
101            header("Location: index.php?url=Admin#addU");
102            exit;
103        }
104        break;
105
106    case "Create meet point":
```

```
107
108     $meetPoint = new MeetPoint(0, $nameMeetPoint, $adressMeetPoint,
$descriptionMeetPoint);
109
110     $errorMessages = $adminInstance->ErrorCreateMeetPoint($meetPoint);
111
112     if (empty($errorMessages)) {
113         if ($adminInstance->CreateMeetPoint($meetPoint)) {
114             $_SESSION["customMessage"] = "Meet point created successfully";
115             header("Location: index.php?url=Admin#meetPoints");
116             exit;
117         }
118     }
119     break;
120
121 case "Remove meet point":
122     if ($adminInstance->RemoveMeetPoint($idMeetPoint)) {
123         $_SESSION["customMessage"] = "Meet point removed Successfully";
124         header("Location: index.php?url=Admin#meetPoints");
125         exit;
126     }
127     break;
128
129 case "Edit meet point":
130     $meetPoint = new MeetPoint($idMeetPoint, $editNameMeetPoint, $editAdressMeetPoint,
$editDescriptionMeetPoint);
131     $erorrMsg = $adminInstance->ErrorEditMeetPoint($meetPoint);
132     if (empty($erorrMsg)) {
133         if ($adminInstance->EditMeetPoint($meetPoint)) {
134             $_SESSION["customMessage"] = "Meet point edited Successfully";
135             header("Location: index.php?url=Admin#meetPoints");
136             exit;
137         }
138     } else {
139         $_SESSION["customError"] = "Meet points' informations are incorrect";
140         header("Location: index.php?url=Admin#meetPoints");
141         exit;
142     }
143
144     break;
145
146 case "Remove school":
147     if ($adminInstance->RemoveSchool($idSchool)) {
148         $_SESSION["customMessage"] = "School removed Successfully";
149         header("Location: index.php?url=Admin#schools");
150         exit;
151     }
152     break;
153
154 case "Edit school":
155     $school = new school($idSchool, $editNameSchool);
156     $erorrMsgEditSchool = $adminInstance->ErrorEditSchool($school);
157     if (empty($erorrMsgEditSchool)) {
158         if ($adminInstance->EditSchool($school)) {
159             $_SESSION["customMessage"] = "School edited Successfully";
160             header("Location: index.php?url=Admin#schools");
161             exit;
162         }
163     } else {
164         $_SESSION["customError"] = "School' informations are incorrect";
```

```

165         header("Location: index.php?url=Admin#schools");
166         exit;
167     }
168     break;
169
170 case "Create school":
171     $school = new school(0, $nameSchool);
172     $errorMsgCreateSchool = $adminInstance->ErrorCreateSchool($school);
173
174     if (empty($errorMsgCreateSchool)) {
175         if ($adminInstance->CreateSchool($school)) {
176             $_SESSION["customMessage"] = "School created successfully";
177             header("Location: index.php?url=Admin#schools");
178             exit;
179         }
180     }
181     break;
182 }
183
184 if (isset($_GET["remove"]) && isset($_GET["idSneaker"])) {
185     $idSneaker = filter_input(INPUT_GET, 'idSneaker', FILTER_SANITIZE_NUMBER_INT);
186     $adminInstance->RemoveSneakerInSale($idSneaker);
187     $_SESSION["customMessage"] = "Sneaker successfully removed";
188     header("Location: index.php?url=Admin#sneakers");
189     exit;
190 }
191
192 require ('./src/Controller/headerController.php');
193 require ('./src/View/AdminView.php');
194
195 if (isset($_SESSION["customMessage"])) {
196     $_SESSION["customMessage"] = "";
197 }
198
199 if (isset($_SESSION["customError"])) {
200     $_SESSION["customError"] = "";
201 }
```

src\Controller\AboutController.php

```

1 <?php
2
3 // Récupère la vue
4 require("./src/Model/Constants.php");
5 require('./src/Controller/headerController.php');
6 require('./src/View/AboutView.php');
7
```