SHOPCONNECT – Prise de notes

# Liste des fonctions – GESTION DES ERREURS

## Avis

### AvisController :

**Gestion des erreurs** : Les erreurs sont gérées de manière appropriée avec les statuts HTTP (NOT\_FOUND pour un avis non trouvé, BAD\_REQUEST pour une mauvaise requête lors de la création).

#### getAllAvis() :

Récupère et renvoie la liste de tous les avis présents dans la base de données.

Méthode HTTP : GET

URL : /api/avis

#### getAvisById(Long id) :

Récupère un avis spécifique par son ID.

Méthode HTTP : GET

URL : /api/avis/{id}

#### createAvis(Avis avis) :

Crée un nouvel avis dans la base de données avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/avis

#### updateAvis(Long id, Avis updatedAvis) :

Met à jour un avis existant avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/avis/{id}

#### deleteAvis(Long id) :

Supprime un avis spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/avis/{id}

### AvisService :

* getAllAvis() : Récupère tous les avis de la base de données et les renvoie.
* getAvisById(Long id) : Récupère un avis spécifique selon l'ID fourni.
* saveAvis(Avis avis) : Sauvegarde un nouvel avis dans la base de données.
* updateAvis(Long id, Avis updatedAvis) : Met à jour un avis existant en utilisant l'ID et les nouvelles informations.
* deleteAvis(Long id) : Supprime un avis spécifique de la base de données en fonction de l'ID fourni.

## Boutique

### BoutiqueController

#### getAllBoutiques() :

Récupère et renvoie la liste de toutes les boutiques présentes dans la base de données.

Méthode HTTP : GET

URL : /api/boutiques

#### getBoutiqueById(Long id) :

Récupère une boutique spécifique par son ID.

Méthode HTTP : GET

URL : /api/boutiques/{id}

#### createBoutique(Boutique boutique) :

Crée une nouvelle boutique avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/boutiques

#### updateBoutique(Long id, Boutique updatedBoutique) :

Met à jour une boutique existante avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/boutiques/{id}

#### deleteBoutique(Long id) :

Supprime une boutique spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/boutiques/{id}

### BoutiqueService

* getAllBoutiques() : Récupère toutes les boutiques de la base de données.
* getBoutiqueById(Long id) : Récupère une boutique spécifique selon l'ID fourni.
* saveBoutique(Boutique boutique) : Sauvegarde une nouvelle boutique dans la base de données.
* updateBoutique(Long id, Boutique updatedBoutique) : Met à jour une boutique existante avec les nouvelles informations.
* deleteBoutique(Long id) : Supprime une boutique spécifique de la base de données en fonction de l'ID fourni.

## Categorie

### CategorieController :

#### getAllCategories() :

Récupère et renvoie la liste de toutes les catégories présentes dans la base de données.

Méthode HTTP : GET

URL : /api/categories

#### getCategorieById(Long id) :

Récupère une catégorie spécifique par son ID.

Méthode HTTP : GET

URL : /api/categories/{id}

#### createCategorie(Categorie categorie) :

Crée une nouvelle catégorie avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/categories

#### updateCategorie(Long id, Categorie updatedCategorie) :

Met à jour une catégorie existante avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/categories/{id}

#### deleteCategorie(Long id) :

Supprime une catégorie spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/categories/{id}

### CategorieService :

* getAllCategories() : Récupère toutes les catégories de la base de données.
* getCategorieById(Long id) : Récupère une catégorie spécifique selon l'ID fourni.
* saveCategorie(Categorie categorie) : Sauvegarde une nouvelle catégorie dans la base de données.
* updateCategorie(Long id, Categorie updatedCategorie) : Met à jour une catégorie existante avec les nouvelles informations.
* deleteCategorie(Long id) : Supprime une catégorie spécifique de la base de données en fonction de l'ID fourni.

## Commande

### CommandeController :

#### getAllCommandes() :

Récupère et renvoie la liste de toutes les commandes présentes dans la base de données.

Méthode HTTP : GET

URL : /api/commandes

#### getCommandeById(Long id) :

Récupère une commande spécifique par son ID.

Méthode HTTP : GET

URL : /api/commandes/{id}

#### createCommande(Commande commande) :

Crée une nouvelle commande avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/commandes

#### updateCommande(Long id, Commande updatedCommande) :

Met à jour une commande existante avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/commandes/{id}

#### deleteCommande(Long id) :

Supprime une commande spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/commandes/{id}

### CommandeService :

* getAllCommandes() : Récupère toutes les commandes de la base de données.
* getCommandeById(Long id) : Récupère une commande spécifique selon l'ID fourni.
* saveCommande(Commande commande) : Sauvegarde une nouvelle commande dans la base de données.
* updateCommande(Long id, Commande updatedCommande) : Met à jour une commande existante avec les nouvelles informations.
* deleteCommande(Long id) : Supprime une commande spécifique de la base de données en fonction de l'ID fourni.

## Paiement

### PaiementController :

#### getAllPaiements() :

Récupère et renvoie la liste de tous les paiements présents dans la base de données.

Méthode HTTP : GET

URL : /api/paiements

#### getPaiementById(Long id) :

Récupère un paiement spécifique par son ID.

Méthode HTTP : GET

URL : /api/paiements/{id}

#### createPaiement(Paiement paiement) :

Crée un nouveau paiement avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/paiements

#### updatePaiement(Long id, Paiement updatedPaiement) :

Met à jour un paiement existant avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/paiements/{id}

#### deletePaiement(Long id) :

Supprime un paiement spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/paiements/{id}

### PaiementService :

* getAllPaiements() : Récupère tous les paiements de la base de données.
* getPaiementById(Long id) : Récupère un paiement spécifique selon l'ID fourni.
* savePaiement(Paiement paiement) : Sauvegarde un nouveau paiement dans la base de données.
* updatePaiement(Long id, Paiement updatedPaiement) : Met à jour un paiement existant avec les nouvelles informations.
* deletePaiement(Long id) : Supprime un paiement spécifique de la base de données en fonction de l'ID fourni.

## Panier

### PanierController :

#### getAllPaniers() :

Récupère et renvoie la liste de tous les paniers présents dans la base de données.

Méthode HTTP : GET

URL : /api/paniers

#### getPanierById(Long id) :

Récupère un panier spécifique par son ID.

Méthode HTTP : GET

URL : /api/paniers/{id}

#### createPanier(Panier panier) :

Crée un nouveau panier avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/paniers

#### updatePanier(Long id, Panier updatedPanier) :

Met à jour un panier existant avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/paniers/{id}

#### deletePanier(Long id) :

Supprime un panier spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/paniers/{id}

### PanierService :

* getAllPaniers() : Récupère tous les paniers de la base de données.
* getPanierById(Long id) : Récupère un panier spécifique selon l'ID fourni.
* savePanier(Panier panier) : Sauvegarde un nouveau panier dans la base de données.
* updatePanier(Long id, Panier updatedPanier) : Met à jour un panier existant avec les nouvelles informations.
* deletePanier(Long id) : Supprime un panier spécifique de la base de données en fonction de l'ID fourni.

## Produit

### ProduitController :

#### getAllProduits() :

Récupère et renvoie la liste de tous les produits présents dans la base de données.

Méthode HTTP : GET

URL : /api/produits

#### getProduitById(Long id) :

Récupère un produit spécifique par son ID.

Méthode HTTP : GET

URL : /api/produits/{id}

#### createProduit(Produit produit) :

Crée un nouveau produit avec les informations fournies dans la requête.

Méthode HTTP : POST

URL : /api/produits

#### updateProduit(Long id, Produit updatedProduit) :

Met à jour un produit existant avec les nouvelles informations fournies.

Méthode HTTP : PUT

URL : /api/produits/{id}

#### deleteProduit(Long id) :

Supprime un produit spécifique de la base de données selon son ID.

Méthode HTTP : DELETE

URL : /api/produits/{id}

### ProduitService :

* getAllProduits() : Récupère tous les produits de la base de données.
* getProduitById(Long id) : Récupère un produit spécifique selon l'ID fourni.
* saveProduit(Produit produit) : Sauvegarde un nouveau produit dans la base de données.
* updateProduit(Long id, Produit updatedProduit) : Met à jour un produit existant avec les nouvelles informations.
* deleteProduit(Long id) : Supprime un produit spécifique de la base de données en fonction de l'ID fourni.

## Promotion

### PromotionController :

#### getAllPromotions() :

Récupère et renvoie une liste de toutes les promotions.

Méthode HTTP : GET

URL : /api/promotions

#### getPromotionById(Long id) :

Récupère une promotion spécifique par son ID.

Méthode HTTP : GET

URL : /api/promotions/{id}

#### createPromotion(Promotion promotion) :

Crée une nouvelle promotion avec les détails fournis dans le corps de la requête.

Méthode HTTP : POST

URL : /api/promotions

#### updatePromotion(Long id, Promotion updatedPromotion) :

Met à jour une promotion existante en utilisant son ID et les nouvelles données fournies.

Méthode HTTP : PUT

URL : /api/promotions/{id}

#### deletePromotion(Long id) :

Supprime une promotion spécifique par son ID.

Méthode HTTP : DELETE

URL : /api/promotions/{id}

### PromotionService :

#### getAllPromotions() :

Récupère toutes les promotions de la base de données.

Retourne une liste de toutes les promotions enregistrées.

#### getPromotionById(Long id) :

Récupère une promotion spécifique depuis la base de données selon l'ID fourni.

Retourne un objet Promotion correspondant à cet ID, ou null si non trouvé.

#### savePromotion(Promotion promotion) :

Sauvegarde une nouvelle promotion dans la base de données.

Retourne la promotion créée.

#### updatePromotion(Long id, Promotion updatedPromotion) :

Met à jour une promotion existante dans la base de données, en utilisant l'ID et les nouvelles informations.

Retourne la promotion mise à jour ou null si elle n'existe pas.

#### deletePromotion(Long id) :

Supprime une promotion spécifique de la base de données en fonction de l'ID fourni.

## Utilisateur

### UtilisateurController :

Contient les endpoints REST pour gérer les utilisateurs. Voici un résumé des fonctionnalités et des vérifications pour chaque méthode :

#### getAllUtilisateurs() :

Récupère tous les utilisateurs.

Méthode HTTP : GET

URL : /api/utilisateurs

#### getUtilisateurById(@PathVariable Long id) :

Récupère un utilisateur par son ID.

Méthode HTTP : GET

URL : /api/utilisateurs/{id}

#### createUtilisateur(@RequestBody Utilisateur utilisateur) :

Crée un nouvel utilisateur.

Méthode HTTP : POST

URL : /api/utilisateurs

#### deleteUtilisateur(@PathVariable Long id) :

Supprime un utilisateur.

Méthode HTTP : DELETE

URL : /api/utilisateurs/{id}

### UtilisateurService :

* findAllUtilisateurs() : Récupère tous les utilisateurs de la base de données.
* findUtilisateurById(Long id) : Récupère un utilisateur spécifique par son ID.
* saveUtilisateur(Utilisateur utilisateur) : Sauvegarde ou crée un nouvel utilisateur dans la base de données.
* deleteUtilisateur(Long id) : Supprime un utilisateur par son ID.
* updateUtilisateur(Long id, Utilisateur updatedUtilisateur) : Met à jour les informations d'un utilisateur.

## PROCEDURE DUMB DE LA DB

### Juste le schéma

PS C:\Program Files\PostgreSQL\16\bin> .\pg\_dump.exe -U postgres -W --schema-only ShopConnect > C:\Users\vince\dump.sql

Mot de passe :

PS C:\Program Files\PostgreSQL\16\bin>