Haute École d'Informatique

AA 2025 - 2026 | Promotion K

UE : PROG3 - Implémentation d'API REST

Enseignant : M. Ryan A.

Exercice TD1 : Java & PostgreSQL (JDBC) - Filtre et pagination

Timing : semaine du 1er décembre

Objectifs académiques :

- Réussir à connecter une base de données PostgreSQL avec Java en utilisant JDBC.
- Comprendre comment doit être exploitée les données d'une base de données avec Java (ou tout autre langage orienté objet)
- Réussir à implémenter la pagination et les recherches multi-critères.

Considérons le schéma de la base de données suivant :

- Product (id: int, name: varchar, price: number, creation_datetime: timestamp)
- Product_category (id: int, name: varchar, #product_id: int)

1. Créez la base de données intitulée "product_management_db" ainsi qu'un nouvel utilisateur (hors ADMIN) qui va s'appeler "product_manager_user" avec un mot de passe "123456". Les privilèges doivent permettre à ce nouvel utilisateur de créer des tables sur cette nouvelle base de données et faire les CRUD sur les tables. Ajouter les scripts SQL utilisés dans un fichier "init_db.sql".

2. Créez les tables correspondantes au schéma fourni plus tôt et ajoutez les scripts SQL dans un nouveau fichier SQL intitulé "schema.sql".

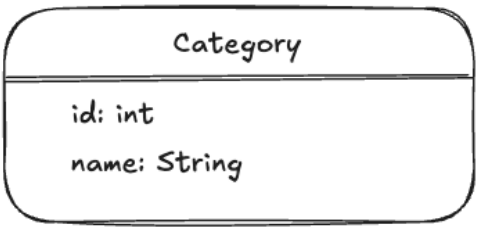
3. Insérez les données suivantes et ajouter les scripts SQL utilisés dans un nouveau fichier intitulé "data.sql" :
    a. Données de la table Product :

id	name	price	creation_datetime
1	Laptop Dell XPS	4500.00	2024-01-15 09:30:00
2	iPhone 13	5200.00	2024-02-01 14:10:00
3	Casque Sony WH1000	890.50	2024-02-10 16:45:00
4	Clavier Logitech	180.00	2024-03-05 11:20:00
5	Ecran Samsung 27"	1200.00	2024-03-18 08:00:00

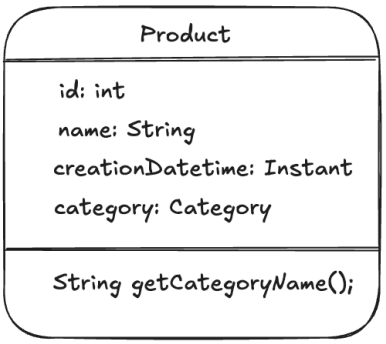
b. Données de la table Product_category :

id	name	product_id
1	Informatique	1
2	Téléphonie	2
3	Audio	3
4	Accessoires	4
5	Informatique	5
6	Bureau	5
7	Mobile	2

4. Créez les classes suivantes :
    a. Classe Category qui permet de spécifier une catégorie de produit :



b. Classe Product qui permet de spécifier un produit.



5. Créez la classe DBConnection qui permet de récupérer une Connection (représentant une connexion active entre l'apps Java et la base de données PostgreSQL) à travers une méthode : Connection getDBConnection();

En particulier, mettez en tant que variable **EN CLAIR** l'URL de la JDBC, le nom d'utilisateur et le mot de passe.

6. Créez une classe intitulée DataRetriever où vous allez récupérer les données Product et Category dans la base de données à travers Java en utilisant la classe DBConnection créée précédemment, et y ajouter les méthodes suivantes :

- **List<Category> getAllCategories()** : lire toutes les catégories enregistrées dans la base de données, les convertir et retourner une liste d'objets Category.
- **List<Product> getProductList (int page, int size)** : lire les produits enregistrés dans la base de données suivant une pagination, les convertir et retourner en liste d'objets Product. Par exemple, si page=1 et size=10, alors on va prendre les 10 premiers éléments; si page=2 et size=5, on va prendre les 6è au 10è éléments (donc 5 éléments du 2è page)
- **List<Product> getProductsByCriteria(String productName, String categoryName, Instant creationMin, instant creationMax)** : lire les produits enregistrés qui respectent les critères suivantes :
    - Si une valeur est fournie pour **productName**, filtrer la liste retournée avec la valeur contenue (opérateur ILIKE et non égale) notamment, ne retourner que les produits contenant un category contenant le nom de category fourni.. Si la valeur fournie est nulle, alors ne pas filtrer avec le nom du produit.
    - Si une valeur est fournie pour **categoryName**, filtrer la liste retournée avec la valeur contenue (opérateur ILIKE et non égale) notamment, ne retourner que les produits contenant un category contenant le nom de category fourni. Si la valeur fournie est nulle, alors ne pas filtrer avec le nom de la category.
    - Si une valeur **creationMin** est donnée, filtrer la liste retournée avec les produits qui ont été créés après la date et heure fournie si une valeur est

fournie. Si aucune valeur n'est fournie (nulle), alors ne pas filtrer avec une date et heure.

- Si une valeur **creationMax** est donnée, filtrer la liste retournée avec les produits qui ont été créés avant la date et heure fournie si une valeur est fournie. Si aucune valeur n'est fournie (nulle), alors ne pas filtrer avec une date et heure.

- **List<Product> getProductsByCriteria(String productName, String categoryName, Instant creationMin, instant creationMax, int page, int size) :** combine les filtres et paginations en même temps, il faut d'abord filtrer et ensuite paginer et non l'inverse.

7. Créez une classe Main, où vous allez invoquer toutes les méthodes de la classe DataRetriever pour les tester et affichez dans la console (avec la méthode statique **System.out.println()**) les résultats. En particulier, voici des données de tests en fonction des méthodes :

a) Pour **List<Category> getAllCategories() :** pas de paramètres alors ça va tout retourner.

b) Pour **List<Product> getProductList (int page, int size)** :

page	size
1	10
1	5
1	3
2	2

c) Pour **List<Product> getProductsByCriteria(String productName, String categoryName, Instant creationMin, instant creationMax) :**

productName	categoryName	creationMin	creationMax
"Dell"	null	null	null
null	"info"	null	null
"iPhone"	"mobile"	null	null
null	null	2024-02-01	2024-03-01
"Samsung"	"bureau"	null	null
"Sony"	"informatique"	null	null
null	"audio"	2024-01-01	2024-12-01
null	null	null	null

d) Pour **List<Product> getProductsByCriteria(String productName, String categoryName, Instant creationMin, instant creationMax, int page, int size)** :

productName	categoryName	creationMin	creationMax	page	size
null	null	null	null	1	10
"Dell"	null	null	null	1	5
null	"informatique"	null	null	1	10

Anticipation pour la suite : Plutôt que de valider visuellement que tout marche sur une classe Main, documentez-vous sur JUnit, et écrivez des tests qui permettent de vérifier par programmation que vos fonctionnalités fonctionnent correctement;