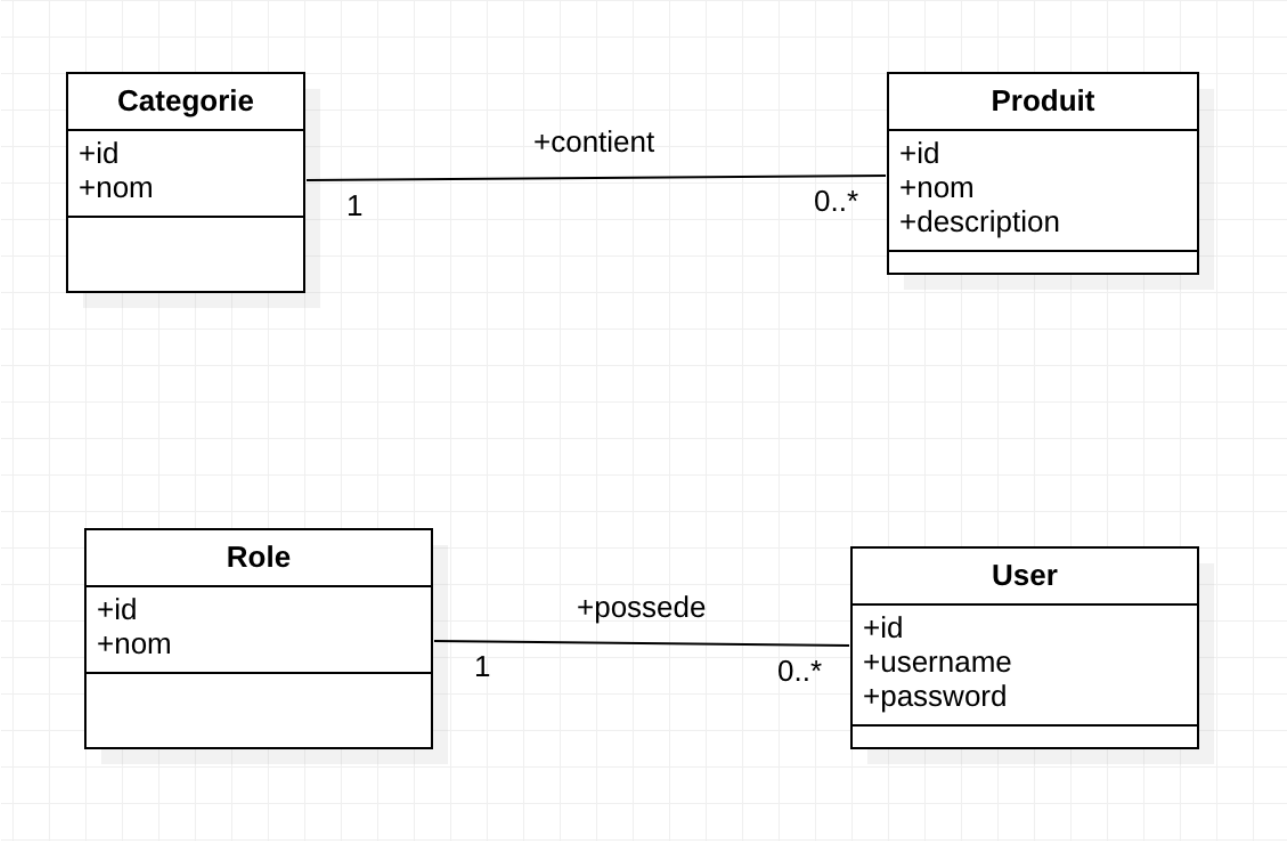
Description du projet

Ce projet consiste à concevoir, développer, sécuriser et documenter une API RESTful complète en utilisant Spring Boot pour la gestion d'un catalogue de produits organisés en catégories. L'application devra gérer deux types d'utilisateurs avec des rôles distincts : les administrateurs, qui auront la capacité de gérer l'intégralité du catalogue (création, lecture, mise à jour, suppression des produits et des catégories), et les utilisateurs, qui pourront uniquement consulter les informations des produits. La sécurité de l'API sera assurée par l'implémentation d'un système d'authentification et d'autorisation basé sur JWT (JSON Web Tokens). Un système de journalisation quotidien sera également mis en place pour le suivi et l'audit des opérations de l'application,

En termes de structure globale nous avons les packages suivants :

- **config** qui contient une configuration des urls autorisés à attaquer l'api ;
- **abstract** qui contient une classe abstraite réutilisable ;
- **controller** qui contient le controller des categories,le controller des produits, le controller des rôles et les controller des users ;
- **dto** qui contient les dto des produits, catégories, users et rôles pour le mapping des données ;
- **model** qui contient les models , catégorie, produit, user et rôle qui nous servirons de tables dans la base de données ;
- **repository** dans lequel nous avons les différents repository ou nous pouvons déclarer des méthodes à utiliser depuis les controller ;
- **securite** qui contient les configuration de sécurité pour l'accès aux endpoints, dans ce package nous gérons le type de token, la réponse que nous voulons lors de la génération du token, aussi nous configurons l'accès aux endpoint selon les différents rôles que nous avons ;
- **service** qui contient tous les services des différents models pour leur utilisation au niveau des controllers ;
- **serviceImpl** qui contient l'implémentation des tous les services qui sont dans le package service ;
- **utiles** qui contient une classe de configuration qui initialise dans la bd un admin avec les accès admin:admin et le rôle ADMIN dès le lancement du projet ;

Modélisation des Données



En termes de relation coté produit-catégorie, on peut dire qu'une catégorie contient zéro ou plusieurs produits tandis qu'un produit appartient à une et une seule catégorie

En termes de relation coté user-rôle, on peut dire qu'un user possède un et un seul rôle tandis qu'un rôle est détenu par zéro ou plusieurs users

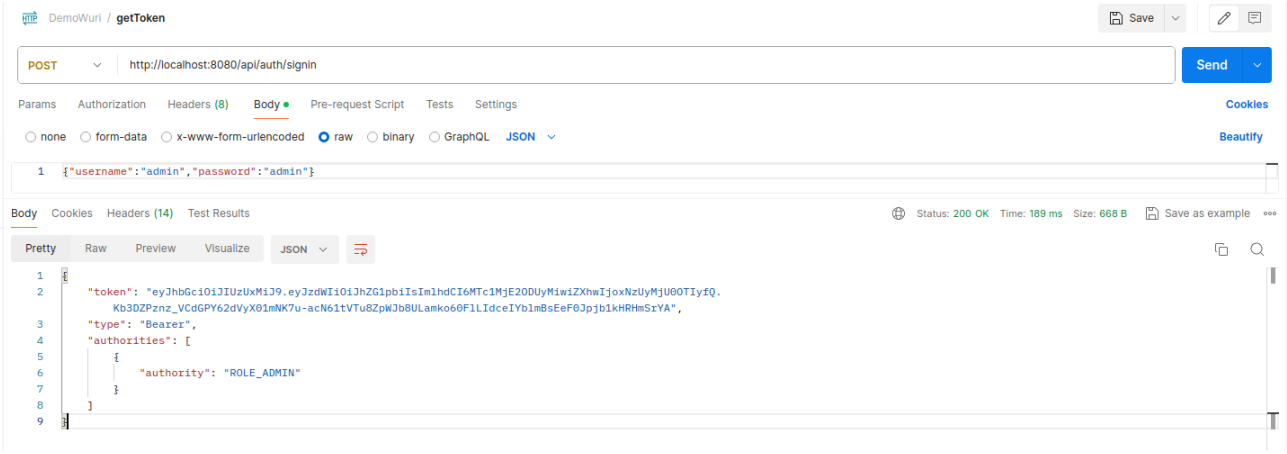
Configuration de l'Application

En terme de configuration nous avons déjà évoqué la partie sécurité plus haut, maintenant nous au niveau de ressources notre application.properties dans lequel nous avons spécifier la source des données qui est le chemin de la base de données (jdbc:postgresql://localhost:5432/demowuri
), ensuite nous avons spécifier le username (postgres) et le mot de passe (2885351) pour acceder à la base de données, ensuite nous avons specifier le Dialect (org.hibernate.dialect.PostgreSQLDialect
) parce que nous utilisons postgres comme base de données.
Nous avons par la suite déclaré notre clé secrète pour la génération du token, et nous terminons par définir les logs de sécurité en mode DEBUG.

Implémentation des Endpoints de l'API RESTful

- getToken

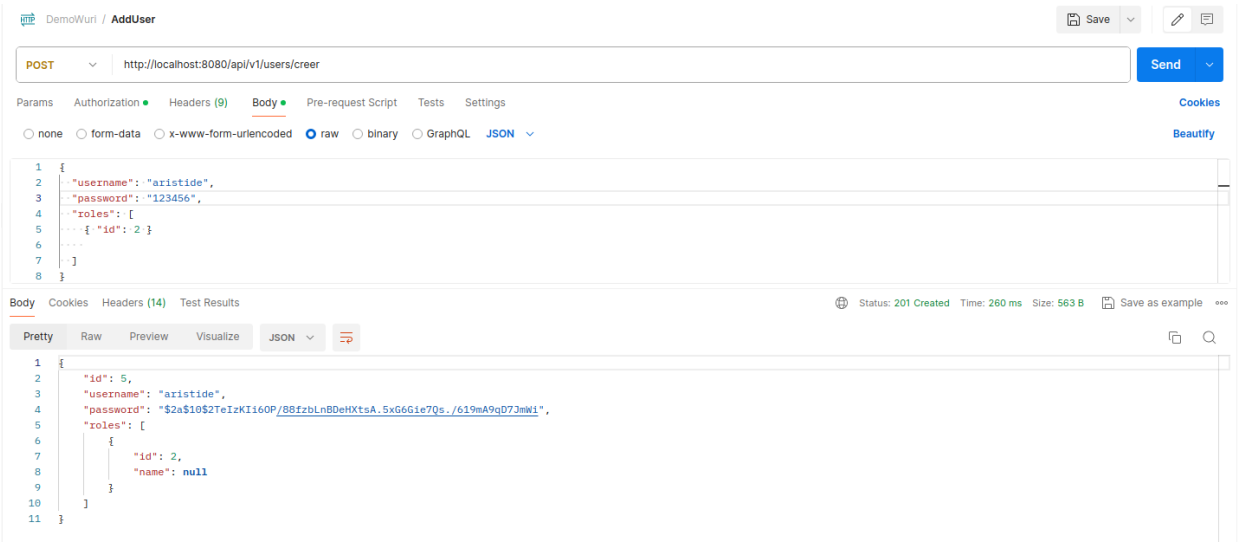
Cet endpoint authentifie un user et retourne un token et le rôle du user, ce token qu'il peut maintenant utiliser pour acceder aux ressources auxquelles il est autorisé.



- addUser

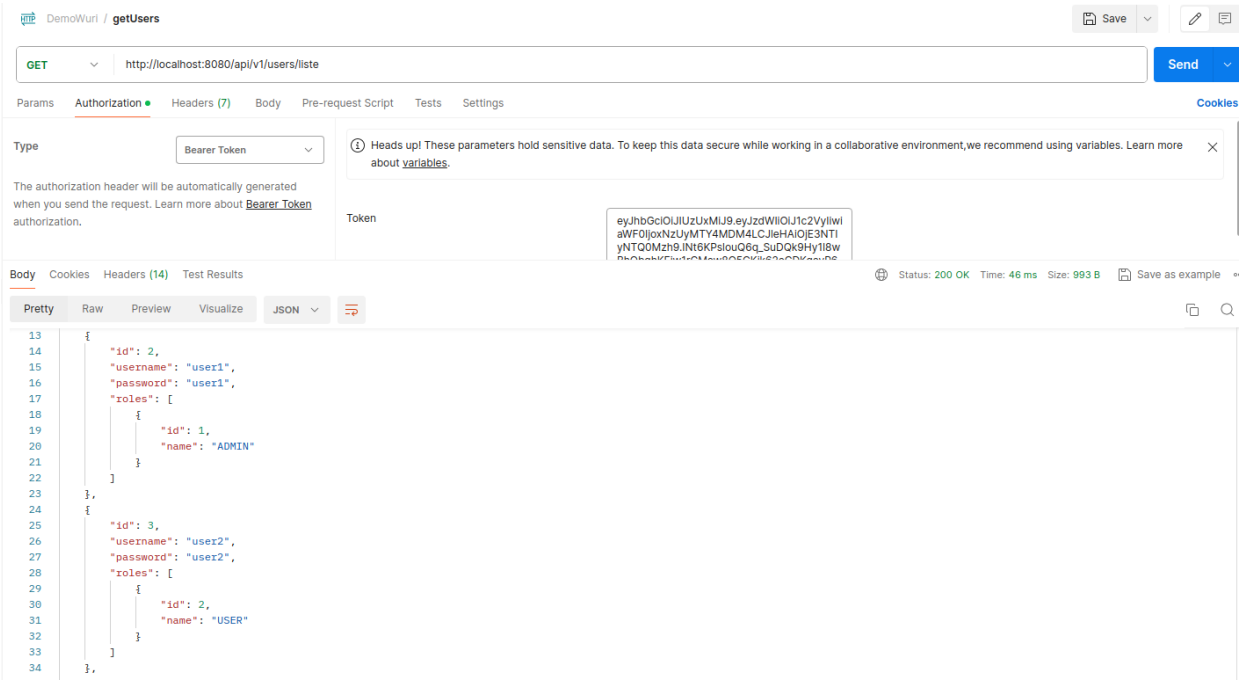
Cet endpoint permet de créer un utilisateur

-



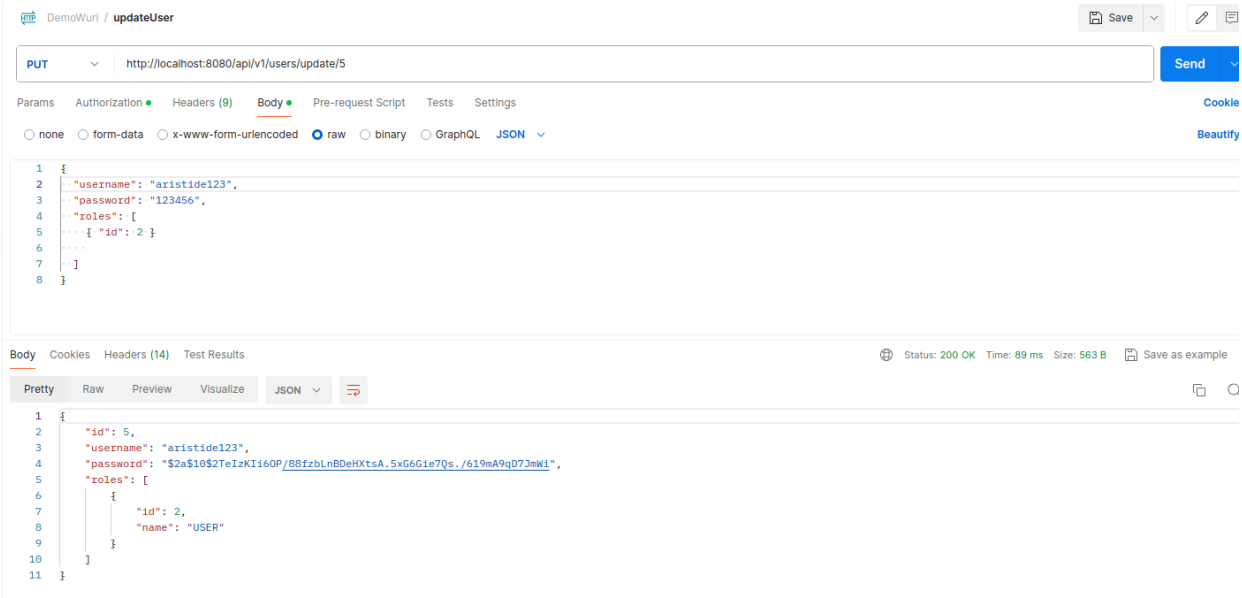
- getUser

Cet endpoint retourne la liste des utilisateurs



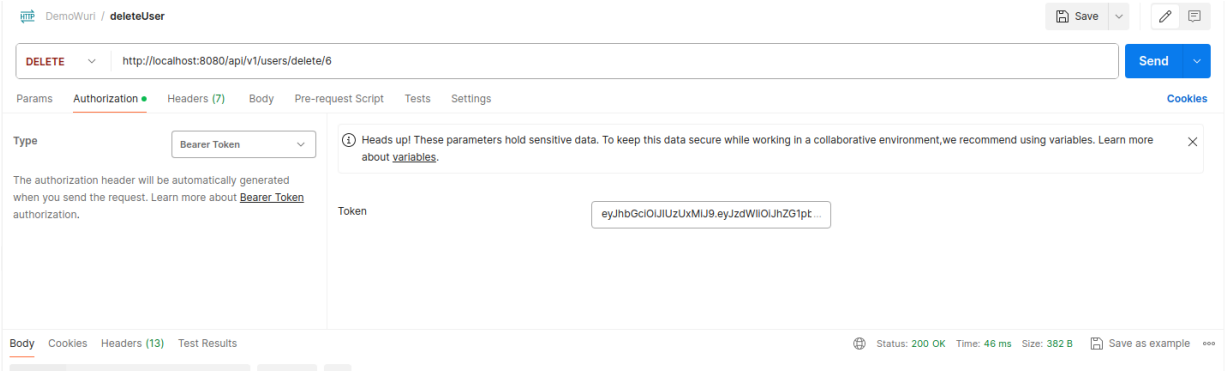
- updateUser

Cet endpoint permet la modification des informations d'un utilisateur



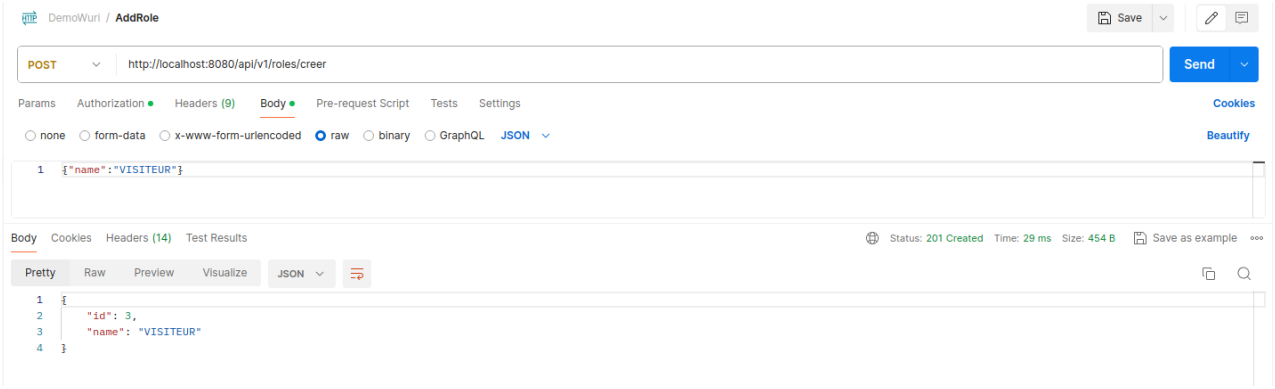
- deleteUser

Cet endpoint permet de supprimer un utilisateur



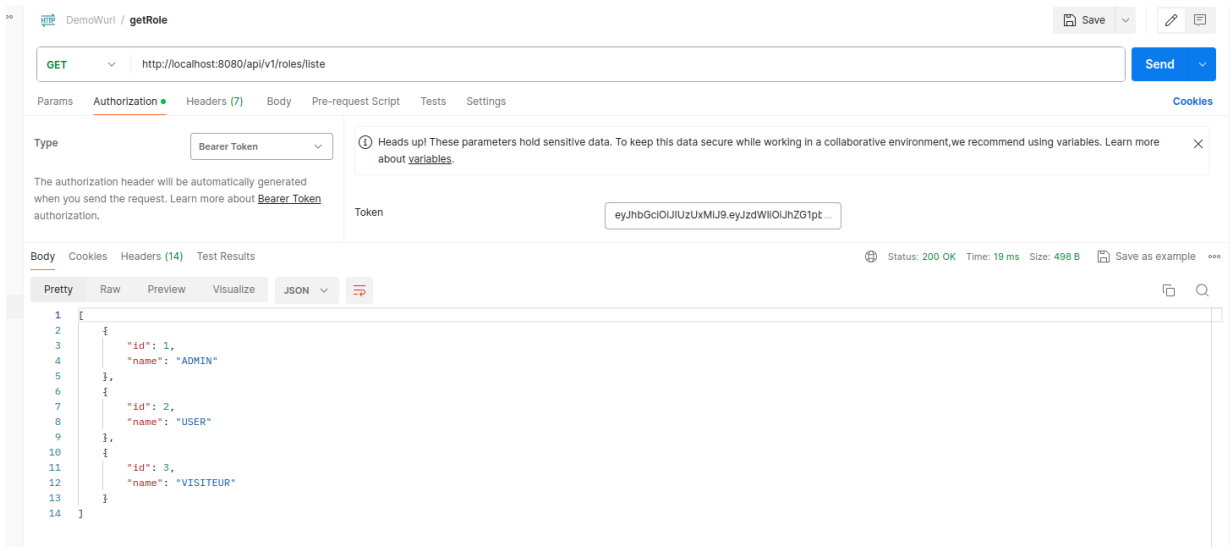
- addRole

Cet endpoint permet d’ajouter un rôle



- getRoles

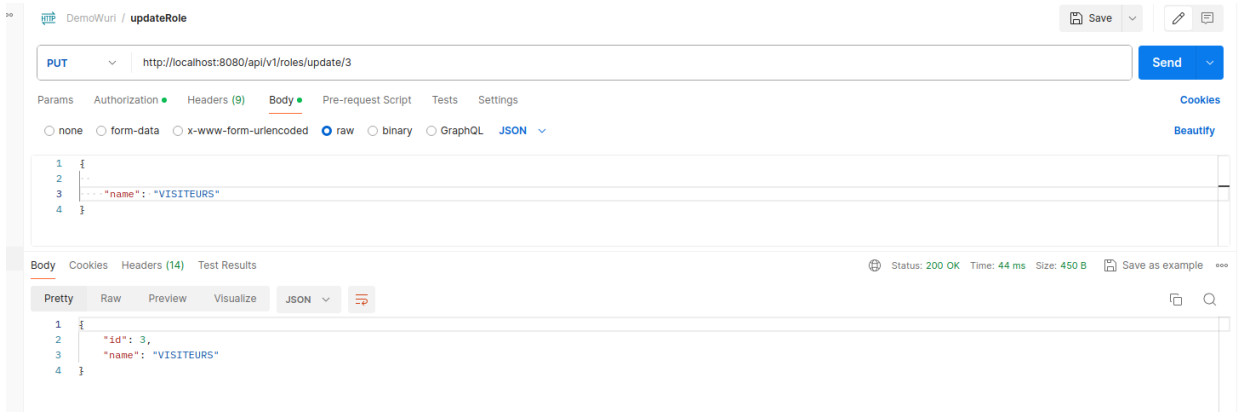
Cet endpoint envoie la liste des rôles



- updateRole

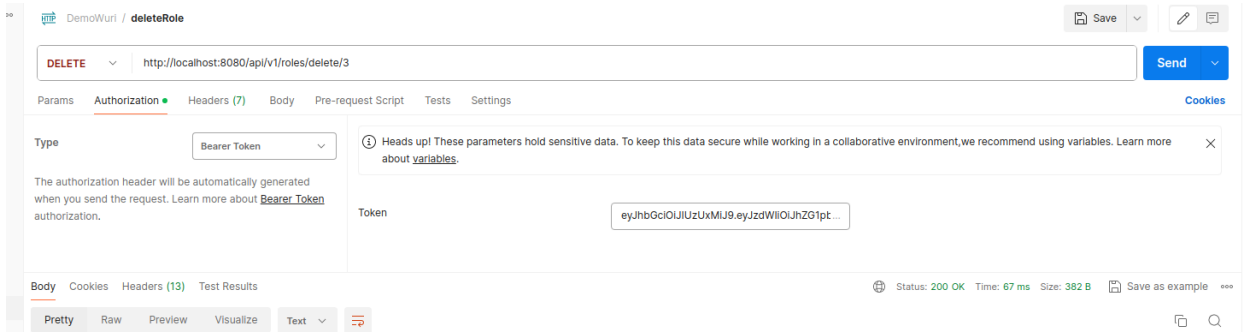
Cet endpoint permet de modifier un rôle

-



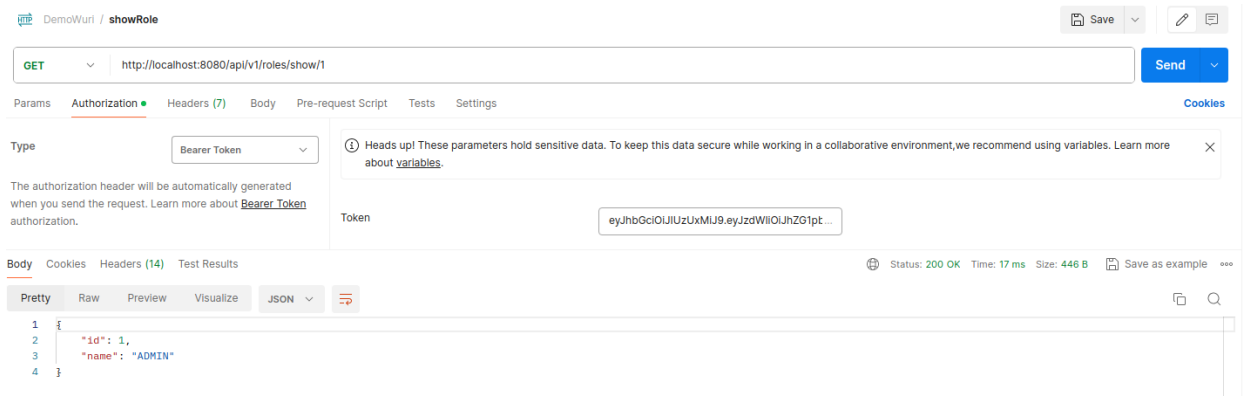
- deleteRole

Cet endpoint permet de supprimer un rôle



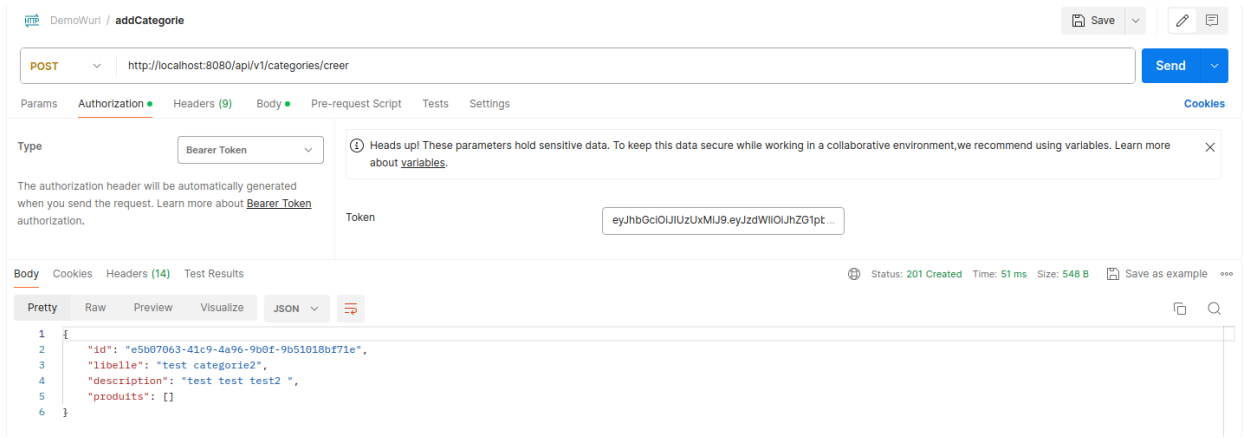
- showRole

Cet endpoint permet de récupérer le rôle selon l'id



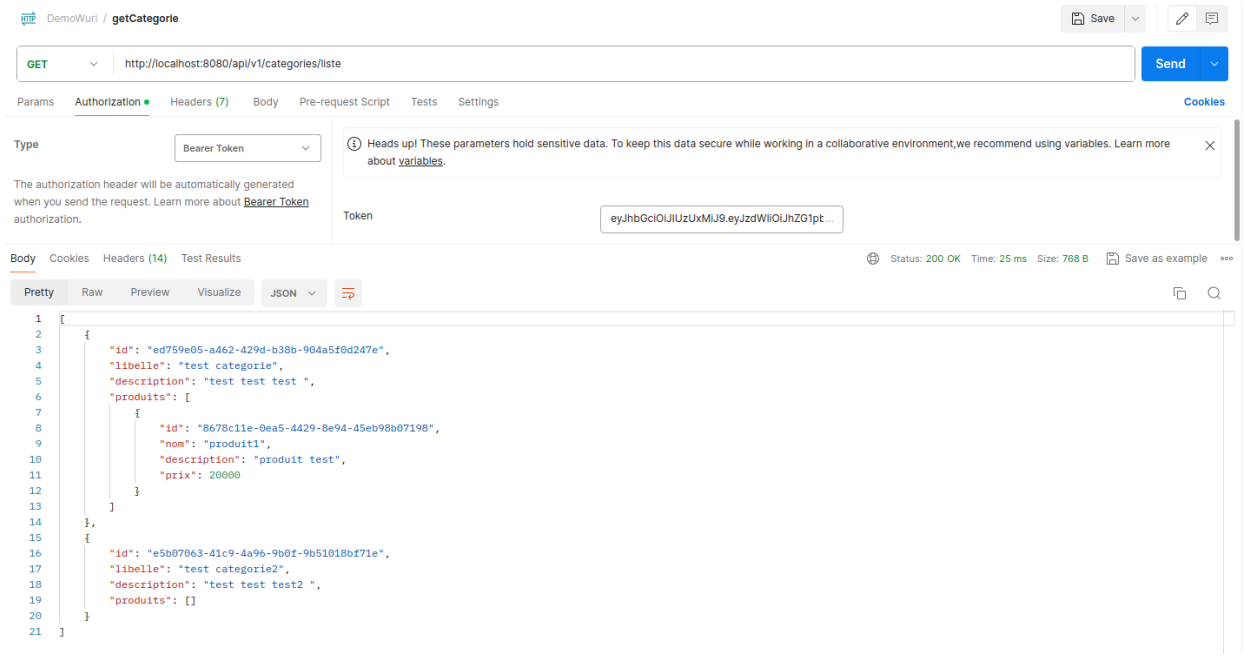
- addCategorie

Cet endpoint permet de créer une nouvelle catégorie



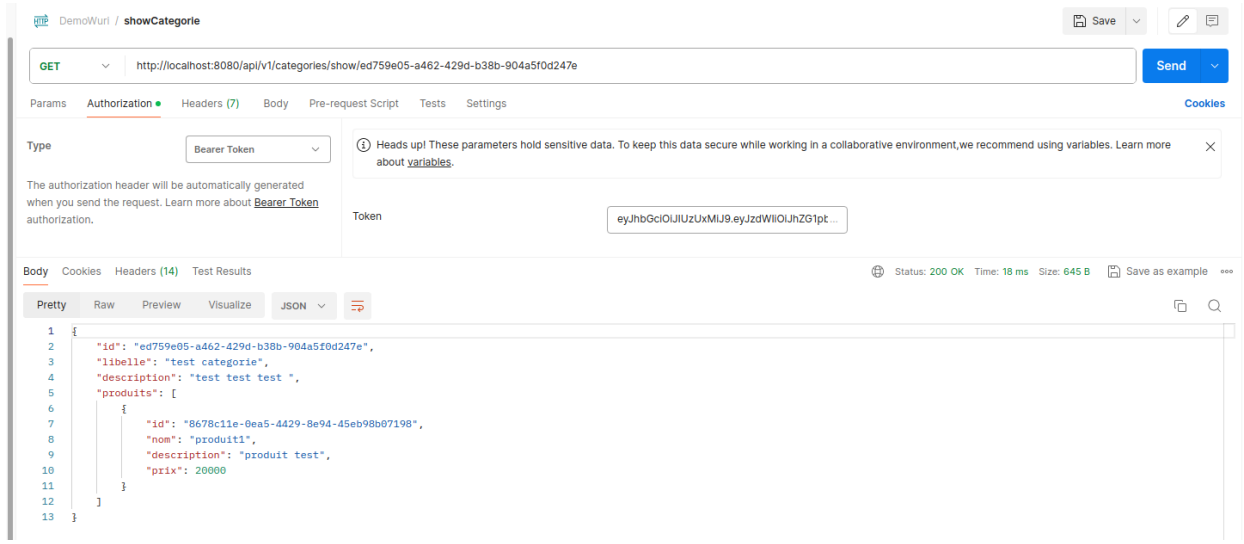
- getCategorie

Cet endpoint créer de lister toutes les catégories et leurs produits



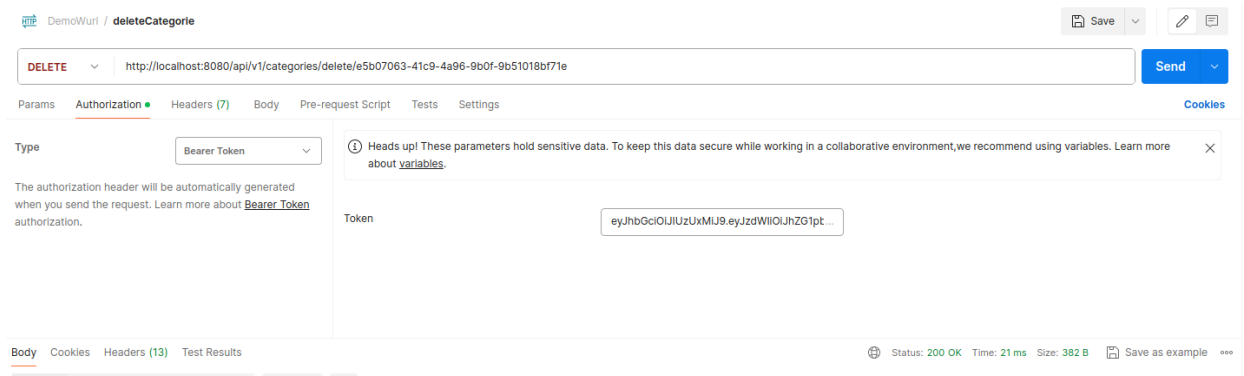
- showCategorie

Cet endpoint permet récupérer une catégorie et ses produits via son id



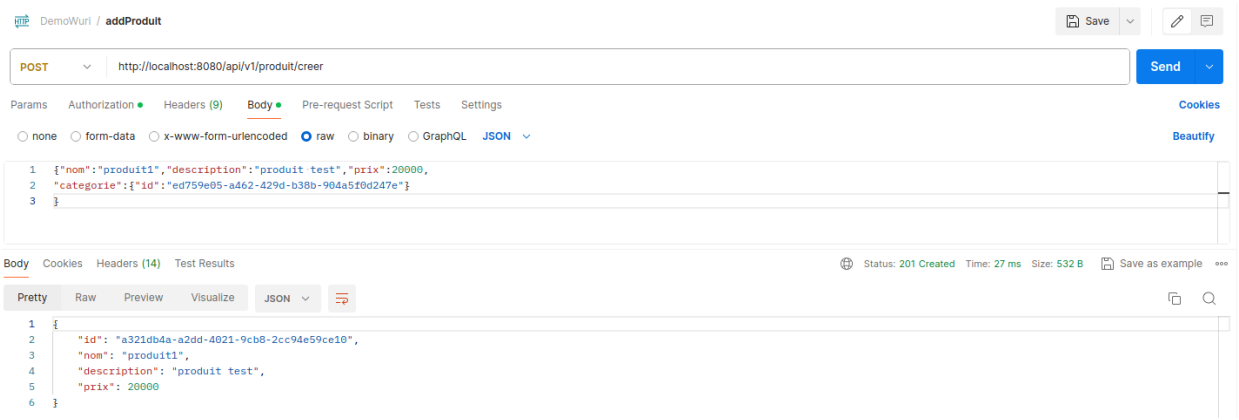
- deleteCategorie

Cet endpoint permet de supprimer une catégorie



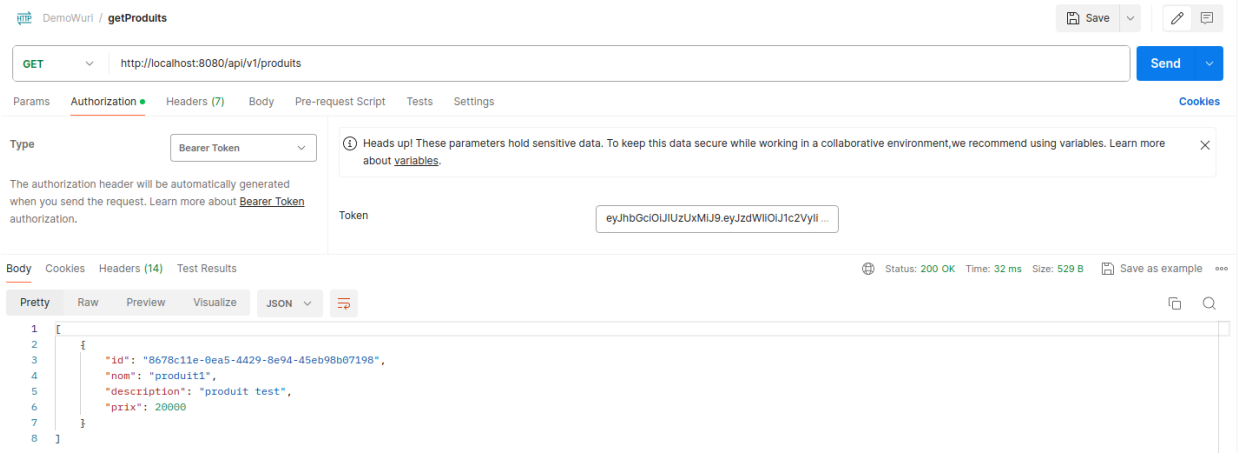
- addProduit

Cet endpoint permet d’ajouter un produit



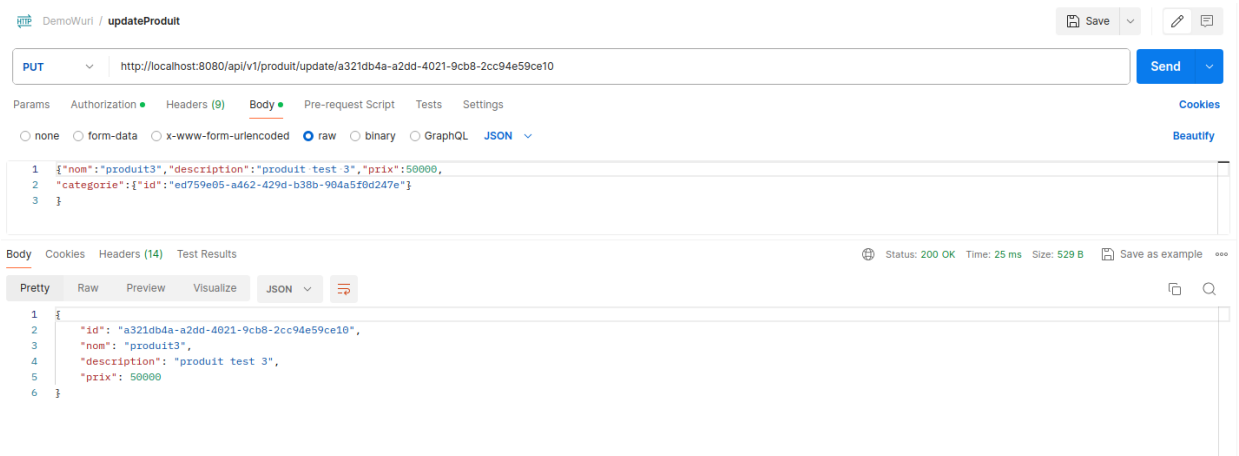
- getProduit

Cet endpoint permet de lister tous les produits



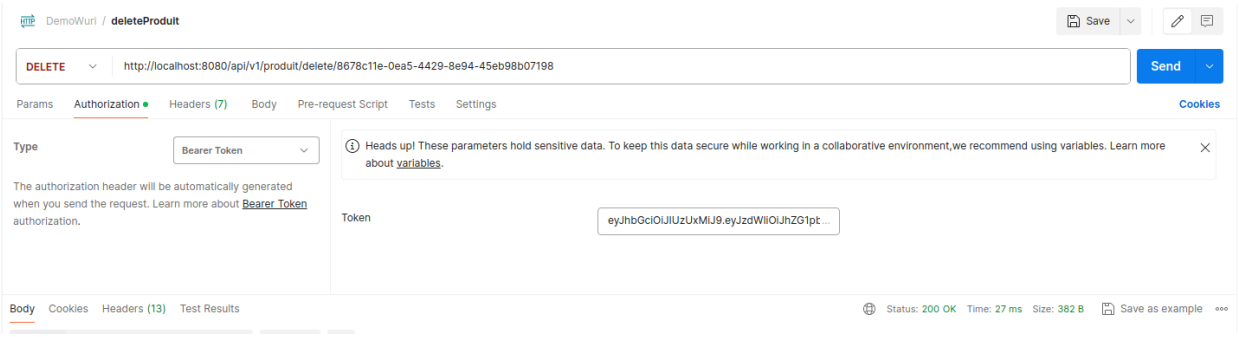
- updateProduit

Cet endpoint permet de modifier un produit



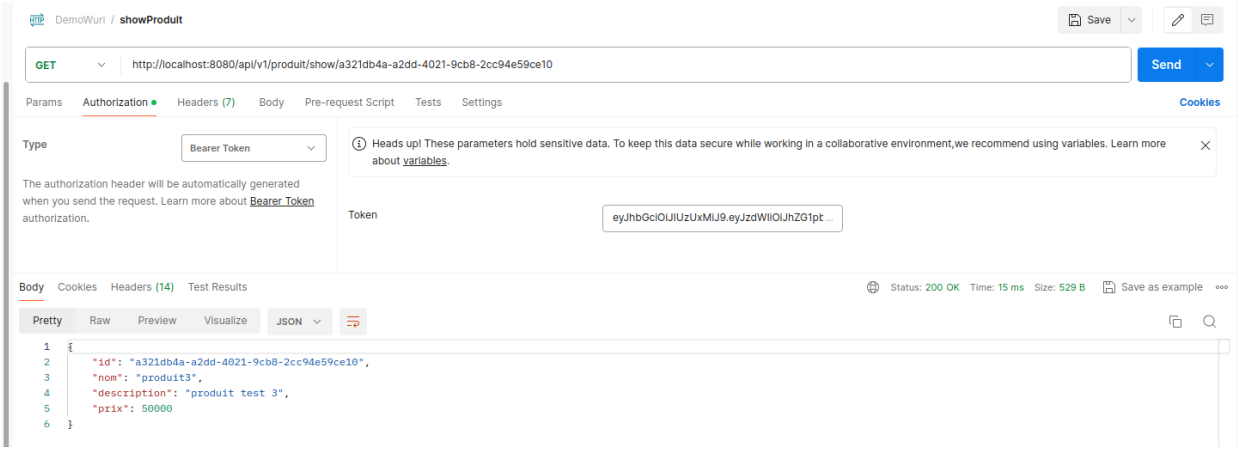
- deleteProduit

Cet endpoint permet de supprimer un produit



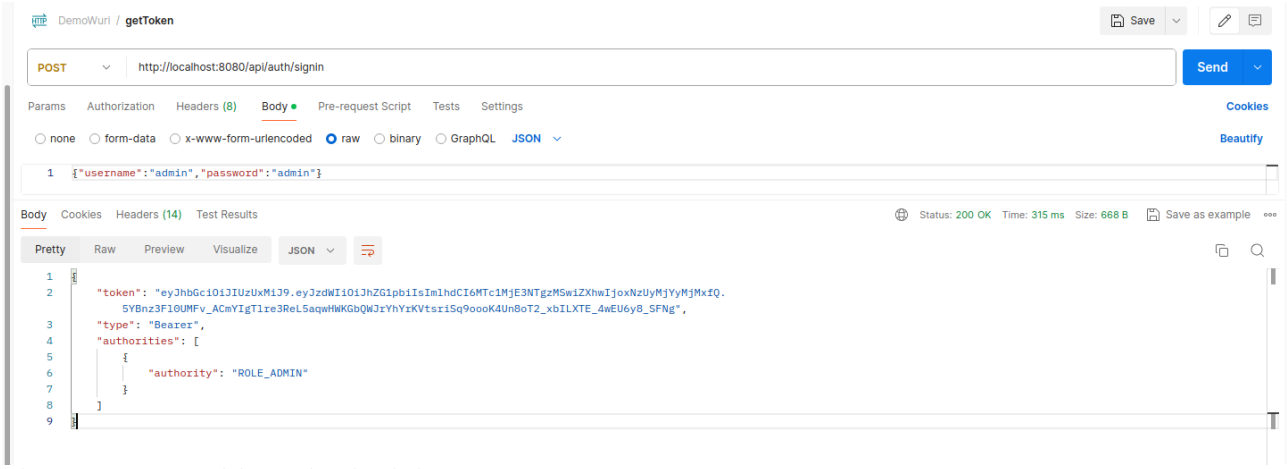
- showProduit

Cet endpoint permet de récupérer un produit via son id

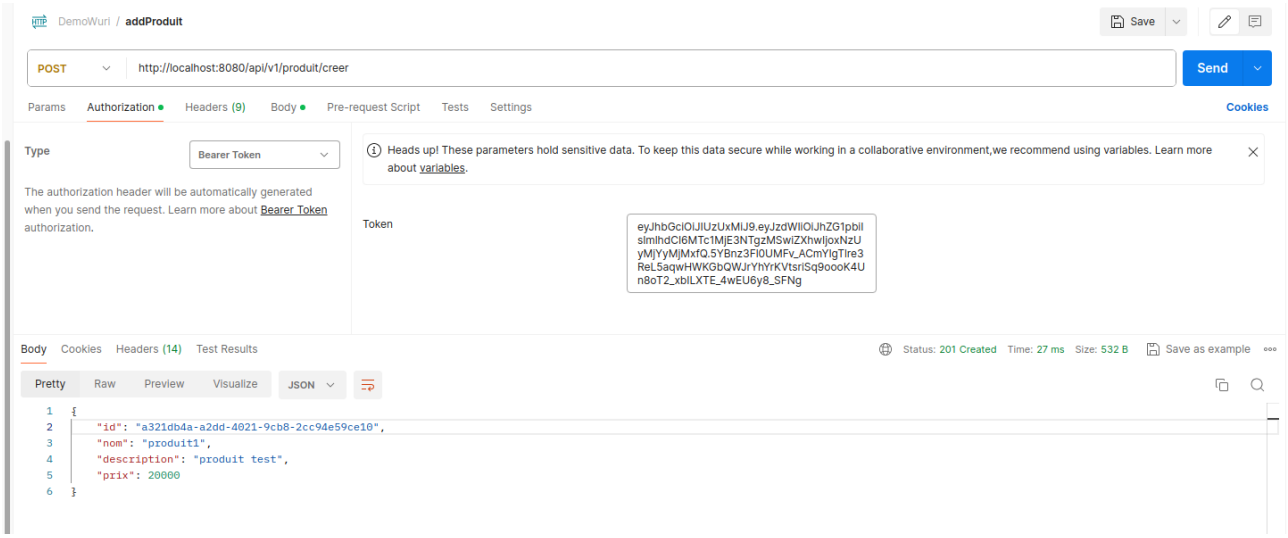


Sécurité de l'API

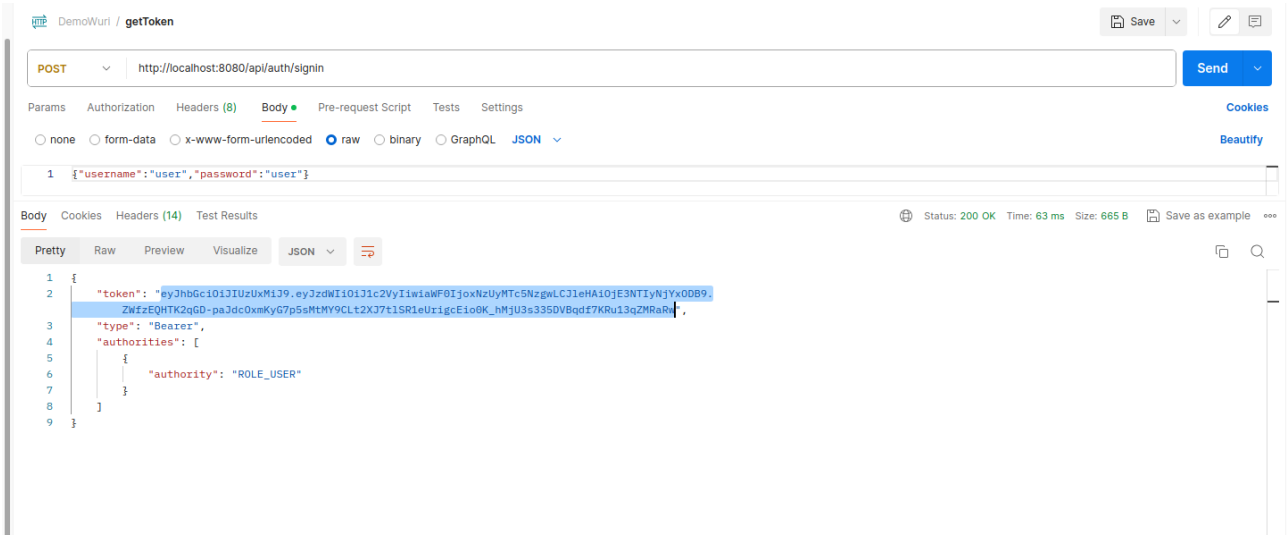
Pour la sécurité nous avons le signin qui permet d'authentifier un user et de retourner le son token d'accès aux endpoint



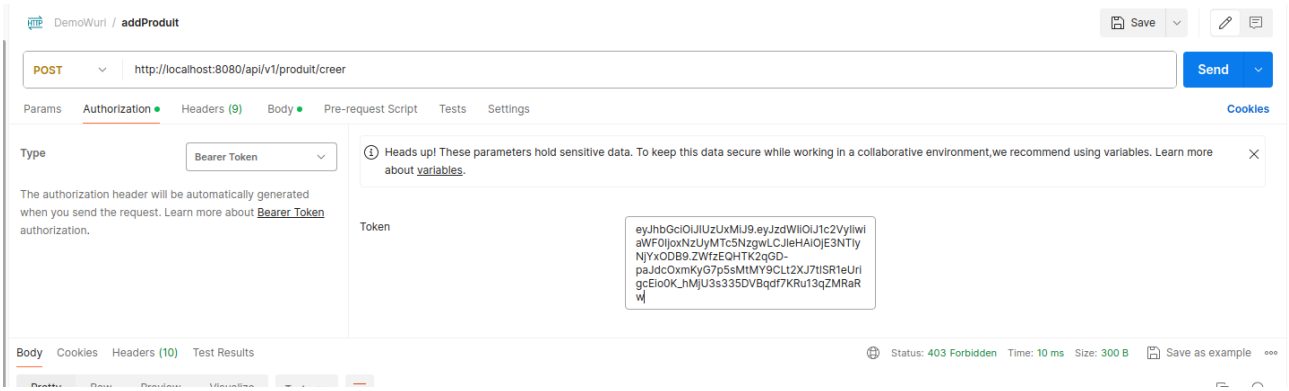
ici nous créons un produit avec le rôle admin



Et ici nous essayons de créer un produit avec un token avec le rôle user qui n'est pas autorisé à créer un produit en commençant par la génération du token user



récupération du produit avec le token user

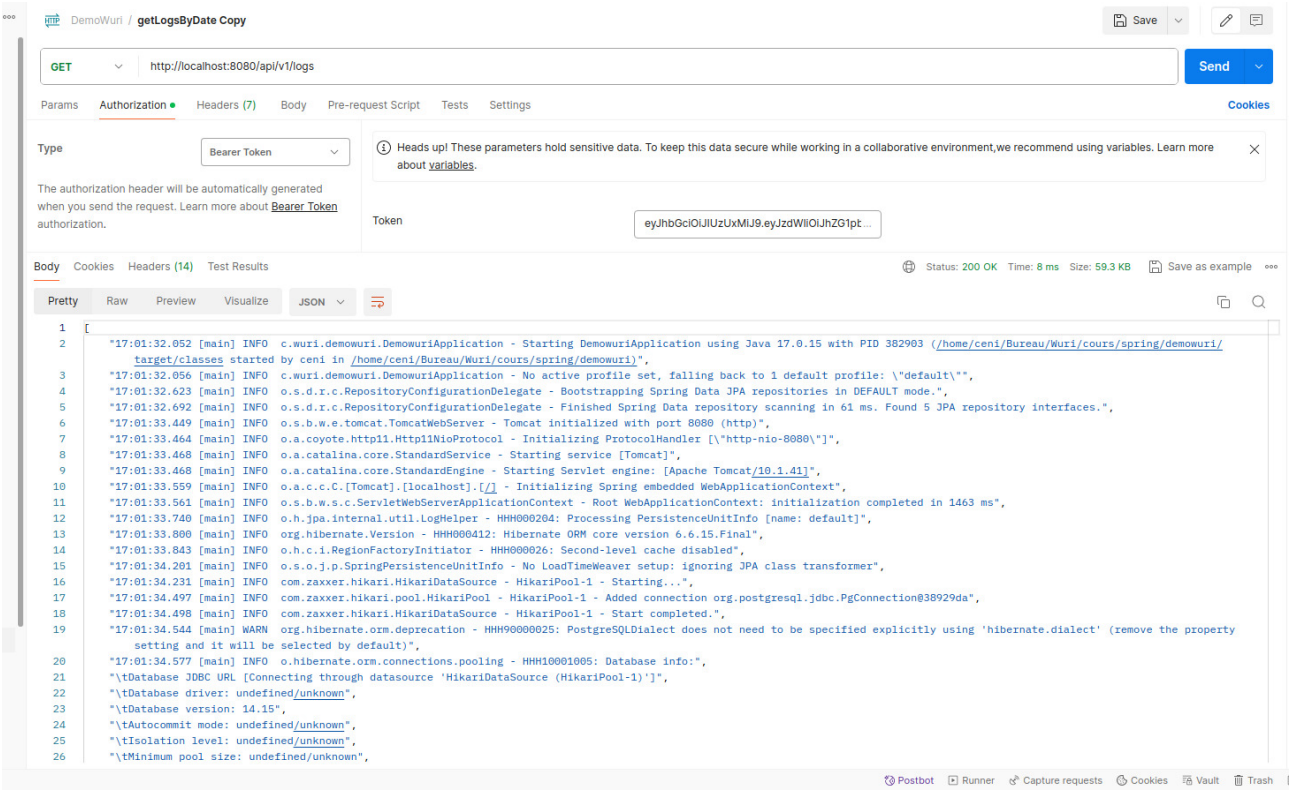


Journalisation

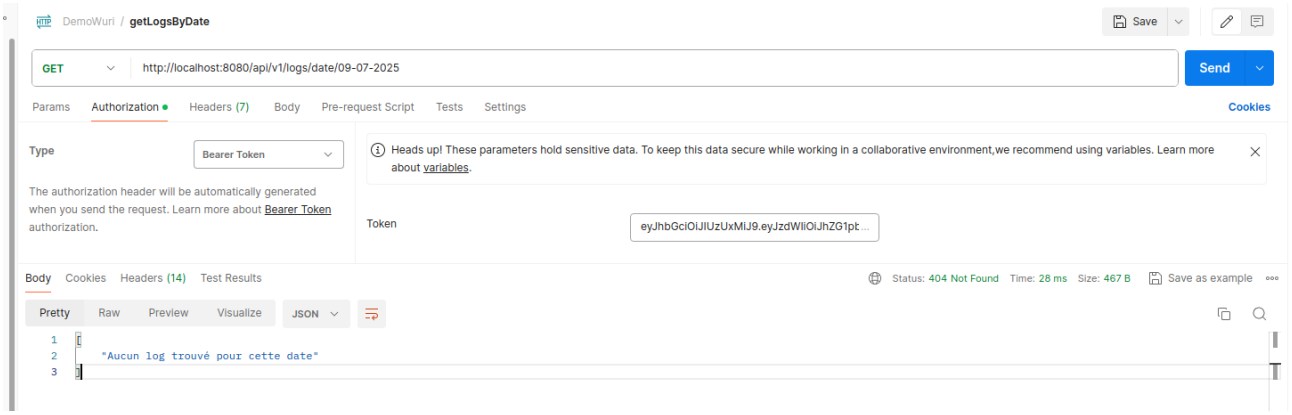
Cette partie consiste à gérer les logs, pour ce faire nous avons défini dans ressources, un fichier logback .xml dans lequel nous spécifions le dossier logs comment chemin du stockage des logs. Nous

avons spécifié demowuri.log qui va stocker les logs courant et demowuri.{dd-MM-yyyy}.log qui va stocker les logs par date, ensuite nous avons deux enpoint logs et logs/date/{date} qui respectivement retourne les logs courent et les logs selon la date.

logs courant :



log par date



Conclusion

Ce projet a permis de bien comprendre la technologie springboot et surtout maitriser la sécurisation des endpoint.

En termes de difficulté était au niveau des endpoints sur lesquels le user et l'admin ont accès, j'avais utilisé hasRole et ça donnait le rôle à l'admin seulement après recherche j'ai vu qu'il fallait hasAnyRole.

En termes de perspectives c’est de garder ce projet comme socle de base pour les futurs projets plus conséquents