SESSION 12-13/06/2024

Groupe 2 : Backend/API/Architecture

TODO : https://chatgpt.com/share/47f34ace-9858-4372-a005-844143ec1ce1

- Reprendre leur modèle de classe
- Jeter un œil sur la création d'API avec Django
- Jeter un œil sur l'approche database first

CREATION D'API

Django REST Framework :

La création d'une API avec Django peut être réalisée efficacement en utilisant le framework Django REST Framework (DRF). Ce framework permet de développer des API RESTful de manière simple et robuste en fournissant des outils et des abstractions pour gérer les différentes opérations courantes d'une API.

Database First

"**Database First**" est une approche de développement où la base de données est créée en premier, puis le code de l'application est généré ou ajusté en fonction de cette base de données existante.

Pour ce faire nous devons :

1) **Configurer la Connexion à la Base de Données** : Modifiez les paramètres de la base de données dans `settings.py` pour vous connecter à la base de données existante.

2) **Utiliser l'Outil « `inspectdb` » :** génère automatiquement des modèles à partir des tables existantes dans la base de données

3) **Vérifier et Ajuster les Modèles** : Passez en revue les modèles générés dans `myapp/models.py` et ajustez-les si nécessaire.

4) **Générer des Migrations Initiales** : Une fois les modèles en place, vous pouvez générer les migrations initiales.

Dans la classe Enseigne :

+ Latitude (décimal): La latitude du magasin

+ Longitude (décimal): La longitude du magasin

+ DateCreation

+ DateModification

Dans la classe Produit :

- Enseignes(int)

+ PrixTTC (decimal): Le prix hors taxe du produit

+ DateCration

+ DateModification

A rajouté dans la classe Utilisateurs :

+ DateCreation(string ?)

+ DateModification(string ?)

A rajouté dans la classe Prix :

+ IdPrix(int)

+ Prix(float)

PROPOSITION :

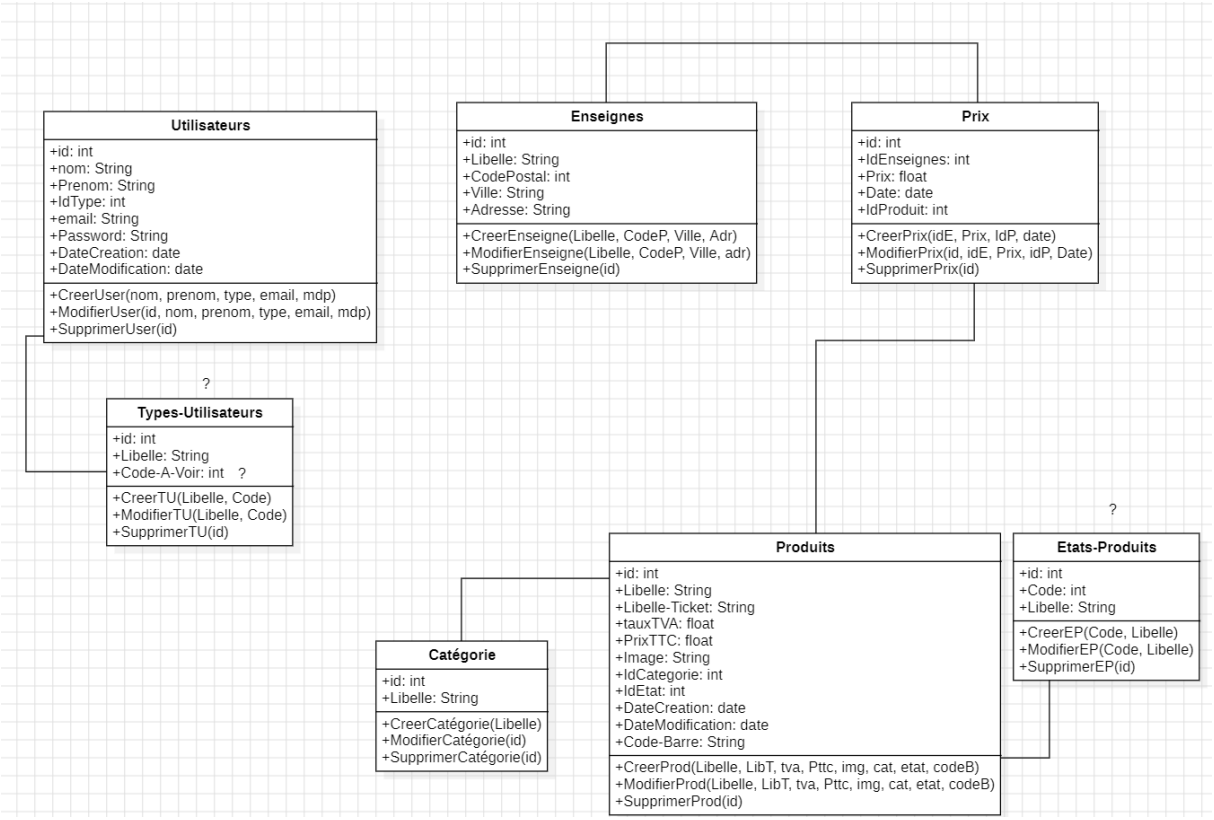
Création d'une table/classe modification pour avoir l'historique des modifications effectuées sur un objet (Produit,Utilisateurs,Enseignes)

Dans la table on trouve soit :

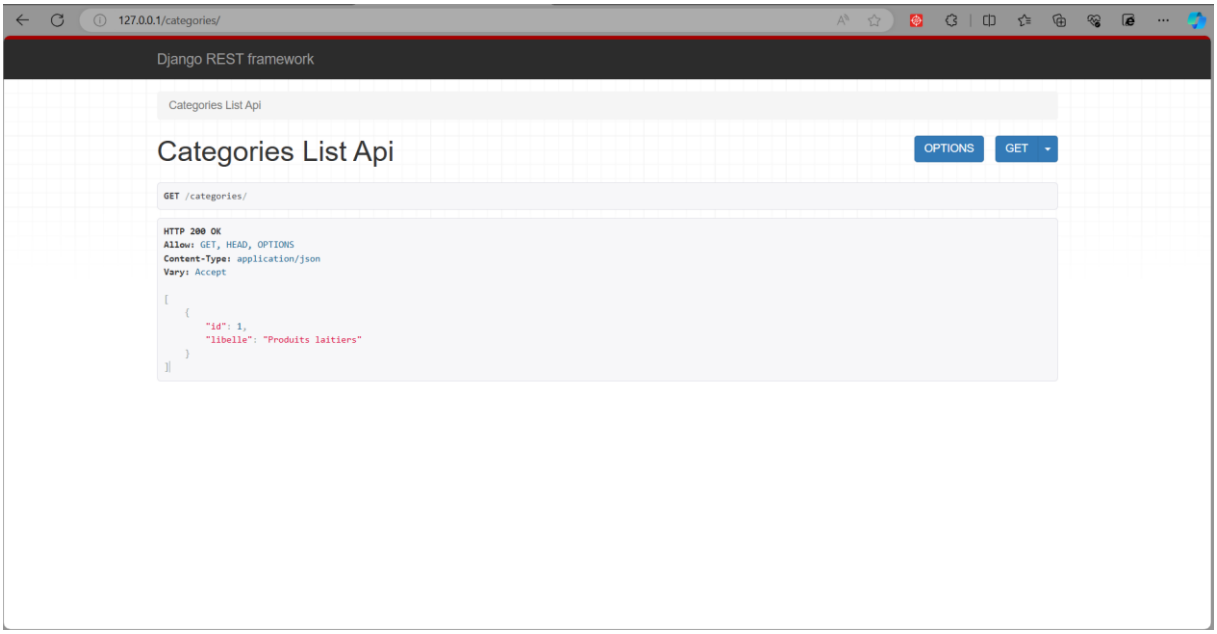
- idUser : int, date, Produit(idProduit : int), Enseignes(idEnseigne : int), Utilisateurs(idUser : int) Effectuer une requête SELECTE avant et après l'update pour savoir quels attributs ont été changés.

- idUser :int, date, Contenu : String(Objet modifié(Enseigne, Utilisateur, Produit), Commentaire : String (Détails modification)

NOUVEAU DIAGRAMME



EXEMPLE CREATION API



19/06/2024

FOCUS : PRODUIT, PRIX

DROITS UTILISATEURS :

- Utilisateur certifié :
- Administrateur :
- Utilisateur lambda certifié :

Méthodes pratique avec django.

[nom_classe].objects.create() : La méthode create() permet de créer un objet.

Il est possible d'utiliser la méthode `create` pour créer et enregistrer un objet en une seule étape. Cependant, utiliser `create` ne permet pas d'appeler explicitement `full_clean()` pour effectuer des validations avant l'enregistrement.

[nom_objet_créé].save() : La méthode save() enregistre l'objet dans la base de données.