

Créer des API

Avec

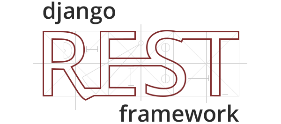


Rapport du Projet :   
**Gestion des Films au cinéma**



**Réalisé par : EZZAHRAOUI HaROUN**

**Encadré par : Mr. HABIB AYAD**



Filière : CI-2IA&GI

Année Universitaire : 2020/2021

INTRODUCTION

I

**Table des Matières**

II

PRESENTATION DU PROJET

* Problématique et contrainte de réalisation

ETAT DE L’ART DJANGO

III

* Méthode REST
* Django
* Le modèle MVT

IV

REALISATION

* Langage de programmation, distributions et environnement
* Installation des exigences
* Créer notre projet
* Créer notre application
* Créer notre base de données et assurer la connexion
* Création de nos modelés et migration
* Administrer notre application via panneau d’administration
* Sérialisation
* Contrôler notre application
* Router l’application

V

TEST

* Tester notre api avec Postman en appliquant le JWTAuthentification
* Test sur FILMS
* Test sur USER
* Test sur CINEMA
* Test sur GENRE
* Test sur TITLE « **Utilisant API externe pour recuperer les titre des films** »
* Join FILM et CINEM
* Join FILM et GENRE

V

CONCLUSION

**INTRODUCTION**

Framework Web (WF) ou une infrastructure d'application Web (WAF) est une infrastructure logicielle conçue pour prendre en charge le développement d'applications Web, notamment des services Web, des ressources Web et des API Web. Les Framework Web fournissent un moyen standard de créer et de déployer des applications Web sur le World Wide Web. Les Framework Web visent à automatiser les frais généraux associés aux activités courantes effectuées dans le développement Web. Par exemple, de nombreux Framework Web fournissent des bibliothèques pour l'accès aux bases de données, les Framework de création de modèles et la gestion de session, et ils encouragent souvent la réutilisation du code. Bien qu'ils visent souvent le développement de sites Web dynamiques, ils s'appliquent également aux sites Web statiques.

Il existe une myriade d'options disponibles lors du choix du Framework backend avec lequel vous souhaitez travailler. Bien que chaque Framework backend ait son propre ensemble d'avantages et d'inconvénients, il y a également quelques autres facteurs que vous voudrez prendre en compte avant de prendre une décision finale. Dans ce guide, a nous le choix des Framework éprouvés et les plus récents pour vous aider à décider quel est le meilleur Framework backend pour vous.

**Le modèle MVT** : Django se base sur le modèle MVT, l´légèrement différent du modèle MVC, le Framework gère lui-même le contrôleur et laisse place au Template.

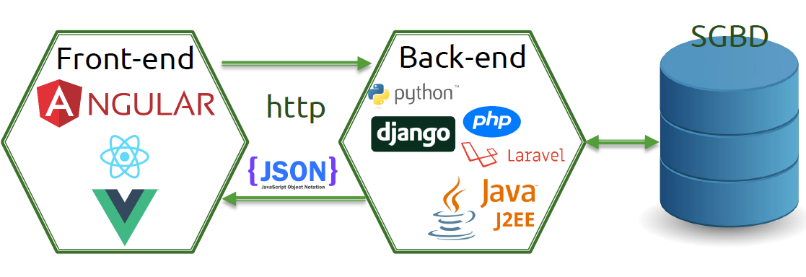
**Méthode REST :** Pour rendre accessible des données via un site, il existe les méthodes dites REST (représentationnel state Transfer). Il s’agit d’un ensemble de conventions et de bonnes pratiques à respecter et non d’une technologie entière. L’information de base, dans une architecture REST, est appelée ressource. Toute information qui peut être nommée est une ressource : la description d’un bâtiment, la liste des arrêts de bus ou n’importe quel concept. Dans un système hypermédia, une ressource est tout ce qui peut être référencé par un lien. L’interface entre les composants est simple et uniforme. En HTTP, cette interface est implantée par les verbes GET, PUT, POST, DELETE, . . . qui permettent aux composants de manipuler les ressources de manière simple. Par exemple quand un agent voudra récupérer la liste des arrêts de bus depuis l’application, il passera par la méthode GET qui lui retournera les ressources voulues.

**ETAT DE L’ART DJANGO**



Lors d’une requête venant de l’utilisateur, le Framework Django gère lui-même, via les règles de routage défini par le développeur, de charger la bonne vue correspondante au résultat voulu. Une Template est un fichier HTML qui sera récupère par la vue pour être envoyer à l’utilisateur, mais entre cette ´étape, Django va exécuter la Template comme si c’était un fichier de code. Inclus dans les Template, le Framework propose l’utilisation des structures conditionnelles, des boucles, des variables... afin d’avoir une grande liberté de d´développement.

**Django** est un Framework web Python de haut niveau qui encourage le développement rapide et propre. Gratuit et open source, Django vous permet d’éviter de réinventer la roue grâce `a toutes les libraires disponibles en Python, mais aussi de tout ce que ce Framework offre d`es son installation. Le fonctionnement d’une application Django se divise en 2 parties. La première section a pour but de préparer l’étape entre l’utilisateur et l’application en elle-même.   
Cette section gère la relation entre les bases de données et les applications Django, mais aussi s’occupe du routage via des règles URL. La seconde section est l’application. Organisée en modèles, vues et Template, l’application se trouve au cœur du projet Django.   
Utilise par la NASA ou le Washington Post, Django prouve sa fiabilité et sa stabilité à travers ce type d’organisme.



**PRESENTATION DU PROJET**

On veut créer une application qui nous aide à gérer les cinémas films et genre de tel façon que les films se récupère automatiquement d’une api externe pour qu’il nous aide pour remplir notre base de données, la table film dispose des colonnes suivant « **ID et TITLE et YEAR\_PRODUCTION et DATE\_PRESENATION et OWNER** », la table cinéma dispose des colonnes suivant « **ID et NAME et CITY et FILMS et OWNER** », la table genre dispose des colonnes suivant «**ID et NAME** ».

Pour assurer le bon fonctionnement la relation entre :

* **FILM** et **CINEMA** est **ManyToMany**
* **FILM** et **Genre** est **OneToMany**

Contrainte :

* **L’utilisation des Apis Génériques « REST FRAMEWORK »**
* **Appliquant l’authentification avec JWT.**
* **Montrer la jointure entre les tables.**

Sans utilisation de l’interface Montrons le fonctionnement de l’api.

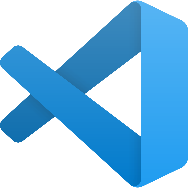
**REALISATION**

1. **Langage de programmation, distributions et environnement**

**Python** est le langage de programmation open source le plus employé par les informaticiens, à cause de sa simplicité et sa facilité d’utilisation, il est simple à coder, vous n’êtes pas obligé d’ajouter des points-virgules ou des accolades n’importe où. En python, vous pouvez écrire de petits codes pour effectuer de grandes taches. Par conséquent, vous gagnez du temps même lors de l’écriture du code



**Anaconda** est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement.

****

**Visual Studio** Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux  
 et MacOs

****

**Postman** est une plateforme de collaboration pour le développement d'API. Les fonctionnalités de Postman simplifient chaque étape de la création d'une API et rationalisent la collaboration afin que vous puissiez créer de meilleures API, plus rapidement.

****

**MySQL** est un système de gestion de bases de données relationnelles (**SGBDR**). Il fait partie des logiciels de gestion de base de données les plus utilisés au monde3, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec *Oracle*, *PostgreSQL* et *Microsoft SQL Server*.

1. **Installation des exigences**

**Django :**

Dans le cadre du processus de développement de Django 3.2, Django 3.2a1 est disponible. Cette version est réservée aux utilisateurs qui souhaitent essayer la nouvelle version et aider à identifier les bogues restants avant la version 3.2.

Pour L’installation du Django tapez la commande

> **pip install --pre django**

**Djangorestframework :**

Le Framework Django REST est une boîte à outils puissante et flexible pour la création d'API Web.

Pour L’installation du djangorestframework tapez la commande

> **pip install djangorestframework**

\*Ajoutez dans **FilmManagement\FilmManagement\settings.py   
dans** INSTALLED\_APP ajouter ‘rest\_framework’,

**MySQLclient :**

Nous utiliserons MySQL comme base de données. Vous pouvez également souhaiter utiliser une autre base de données ou avoir déjà une base de données installée. Cette étape à pour d’assurer la connexion à une base de données MySQL et réaliser des migrations.

Pour L’installation du mysqlclient tapez la commande

> **pip installer** **mysqlclient**

**Djangorestframework\_simplejwt :**

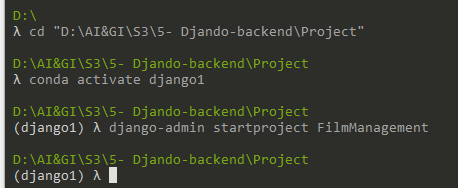
Pour bénéficier la stratégie d'authentification utilisée par les applications client/serveur, json web token nous allons utiliser la djangorestframework\_simplejwt bibliothèque, recommandée par les développeurs DRF.

Pour L’installation du djangorestframework\_simplejwt tapez la commande

> **pip install** **djangorestframework\_simplejwt**

1. **Créer notre projet**

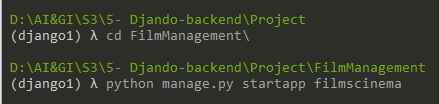
* Pour la création d’un projet dans django on utilise la commande suivante



Dans notre invite de commandes on accède à notre dossier puis activé l’environnement qui contient les packages requis en suite on exécute   
> **django-admin startproject FilmManagement**

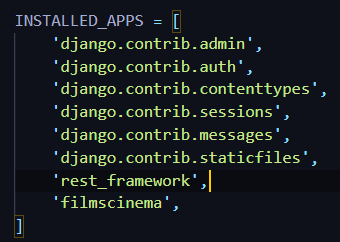
1. **Créer notre application**

* La création d’une application au sein de notre projet FilmManagement



Dans notre invite de commandes on accède à notre projet puis on lance la commande  
> **python manage.py startapp filmscinema**

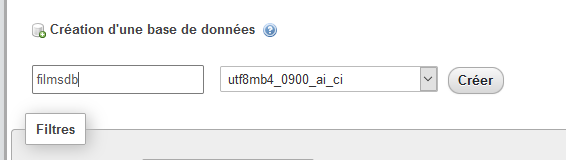
* Ajoutez dans **FilmManagement\FilmManagement\settings.py** la configuration suivante



A la fin de INSTALLED\_APP ajoutons Le nom de notre application

1. **Créer notre base de données et assurer la connexion avec elle**

* La création d’une base de données en MySQL se fait de deux manières via ligne de commande ou via l’interface du navigateur
* Pour accéder à SGBDR MySQL par défaut on a :
* Username : root
* Password :

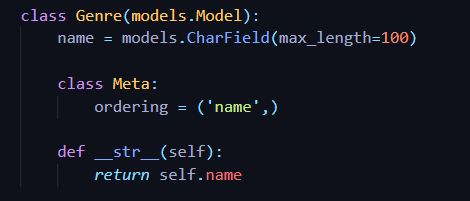




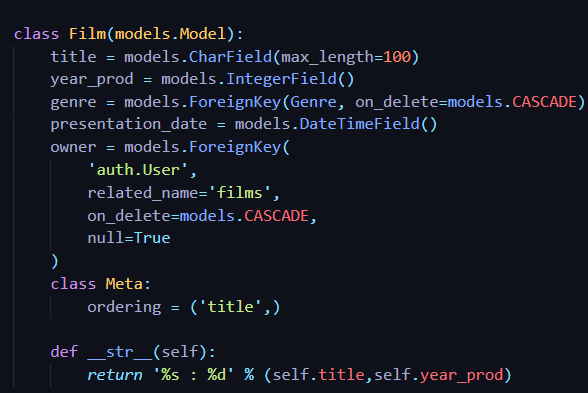
* Dans fichier **FilmManagement\FilmManagement\settings.py** ajoutant la configuration suivante

1. **Création de nos modelés et migration**

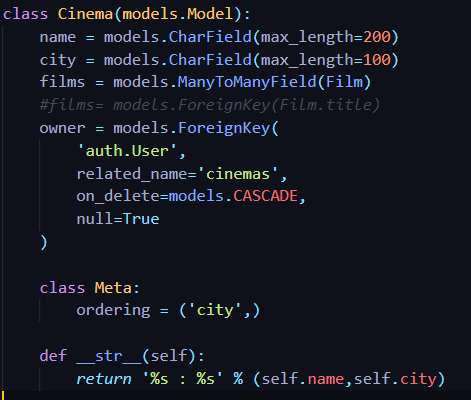
* A l’intérieur de notre fichier **models.py** on va créer :



**Création de notre modèle Genre dans ses colonnes on trouve le nom et on va lister les genres selon le nom**

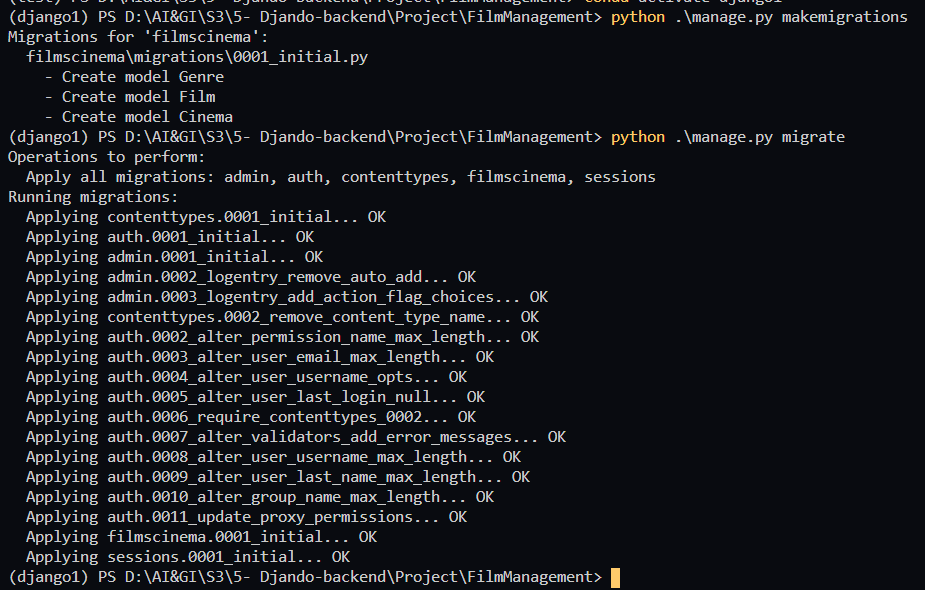


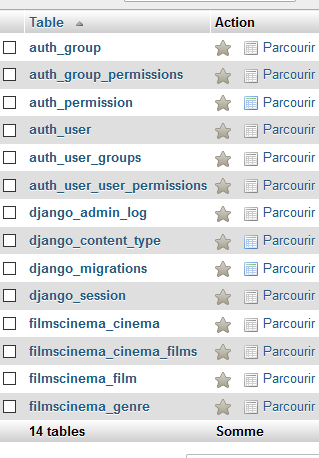
**Création de notre modèle Film dans ses colonnes on trouve le titre et l’année de production et son genre comme clé étrangère, la date de sortie dans le cinéma et on va lister les Films selon le titre.**



**Création de notre modèle Cinéma dans ses colonnes on trouve le nom et la ville, les films dans ce cinéma comme clé étrangère mais on le représente avec la relation many to many aussi les couronnées de super user et on va lister les cinémas selon leur ville.**

* **Pour que nos modelés soient physiques on doit migrer vers la base de données MySQL**





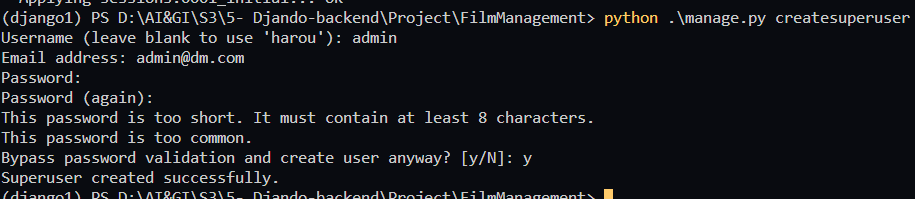
**La migration crée directement les table d’authentification, les groups et user**

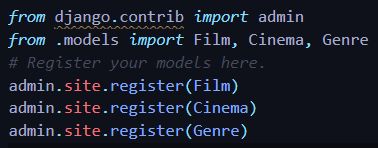
**}**

**La table filmscinema\_cinema\_films est créé automatiquement à cause de notre relation « ManytoMany » dans le modelé**

1. **Administrer notre application via panneau d’administration**

* Maintenant on va créer un Super User pour que nous puisse accéder au panneau

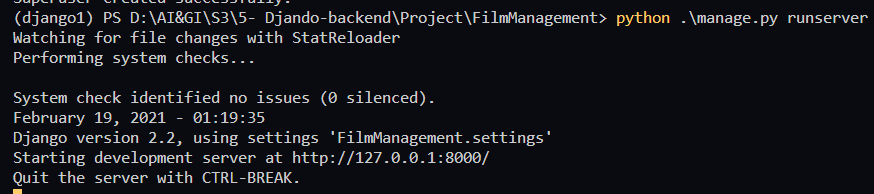




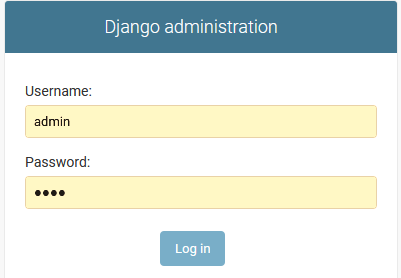
* Dans le fichier **admin.py** on ajoute

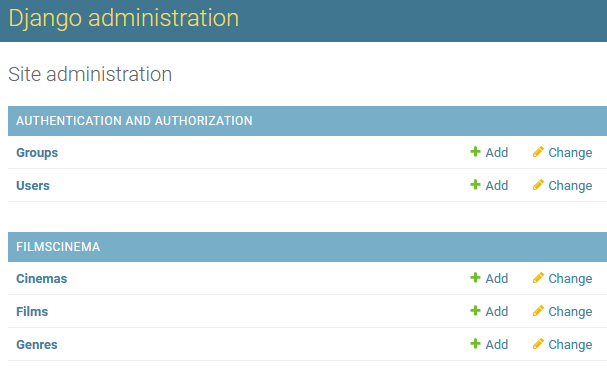
**Pour que les tables se visualise dans l’AdminPanel on doit registrer les modelés qu’on a créé**

* Pour vérifier on doit lancer le serveur avec la commande suivante

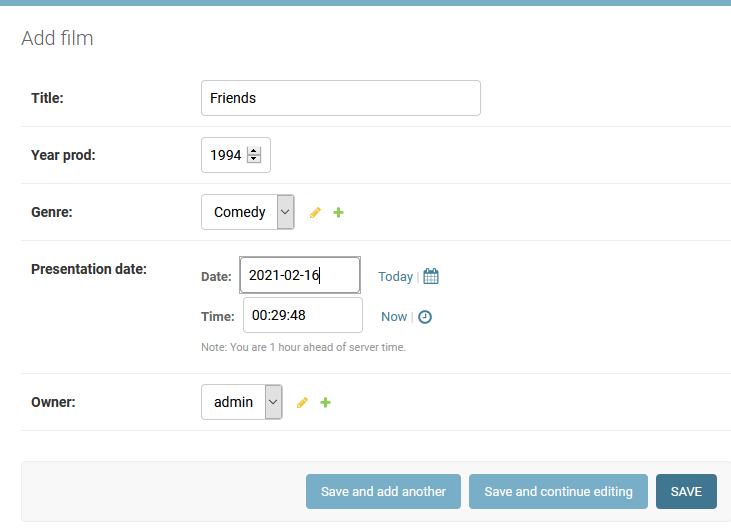


* Si on accède au lien « **http://127.0.0.1:8000/admin/** »

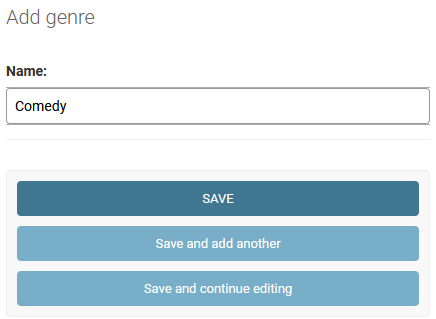




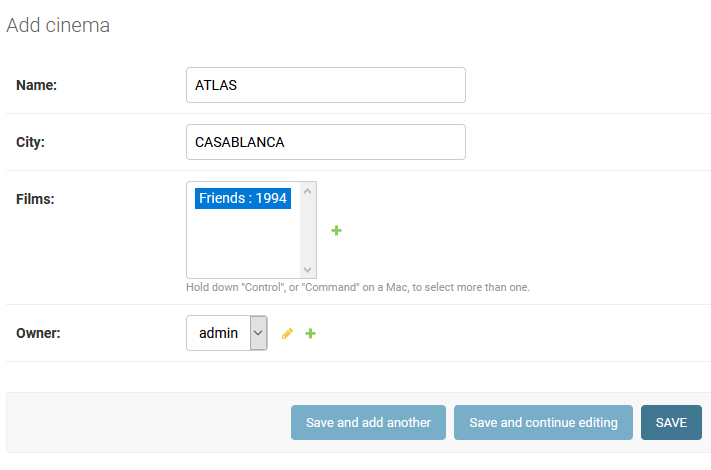
* On peut directement administrer notre application via adminPanel ou Django administration



**Ajouter un Film via DjangoAdmin**



**Ajouter un genre via DjangoAdmin**



**Ajouter un Cinéma via DjangoAdmin**

1. **Sérialisation**

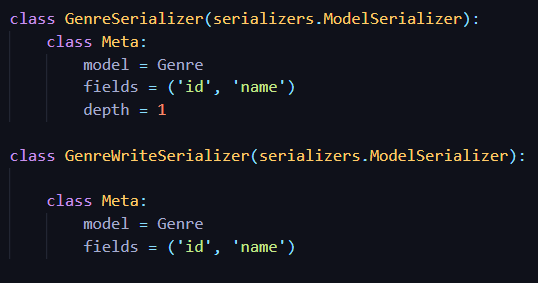
* La sérialisation permet aux données, complexes telles que les modèles, d’être convertis en type de données natives pour Python. Ils peuvent ensuite être facilement rendus dans les formats JSON ou XML. Dans le cas présent, la s´sérialisation permet aussi de s´sélectionner les champs à afficher pour limiter la vue des données.

**On va utiliser le modèle près défini dans Django d’user, puis on définit la relation entre utilisateur et les deux modelé films et cinémas**

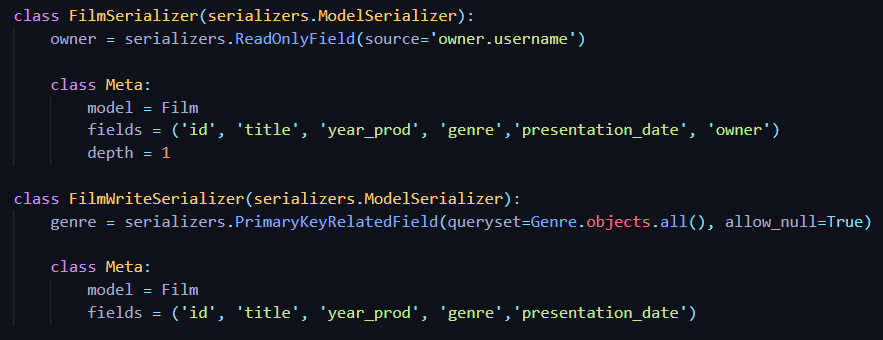


* **On va créer deux sérialiser l’un pour la modification et l’autre pour la visualisation seulement**

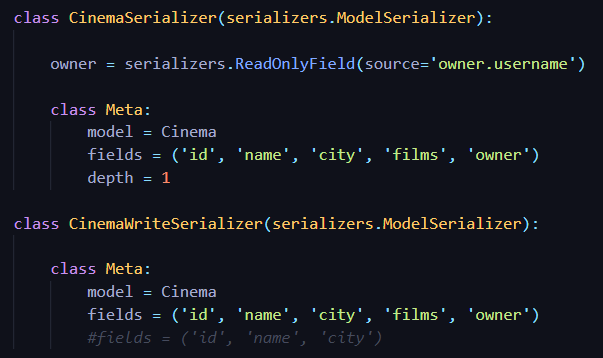
**Sérialisation des genres  
En récupérant ID et NAME**



**Sérialisation des genres  
En récupérant ID et TITLE et YEAR\_PRODUCTION et DATE\_PRESENATION et OWNER**



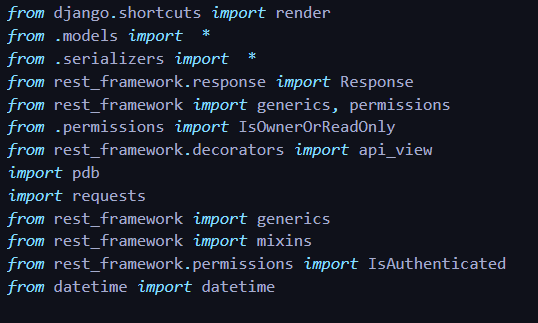
**Sérialisation des genres  
On récupérant ID et NAME et CITY et FILMS et OWNER**



1. **Contrôler notre application**

* Pour contrôler notre application on doit créer des views dans le fichier **views.py**

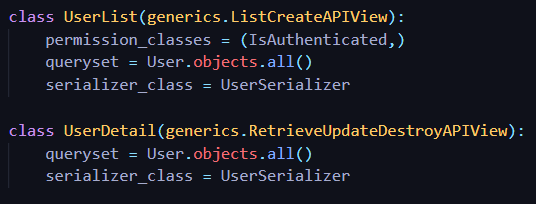
**Ensemble des package qu’on va utiliser**



**Pour récupérer les films d’une autre api externe qui s’appelle omdbapi on parcourt la requête puis on stocke dans un dictionnaire pour assure le bon fonctionnement de api**



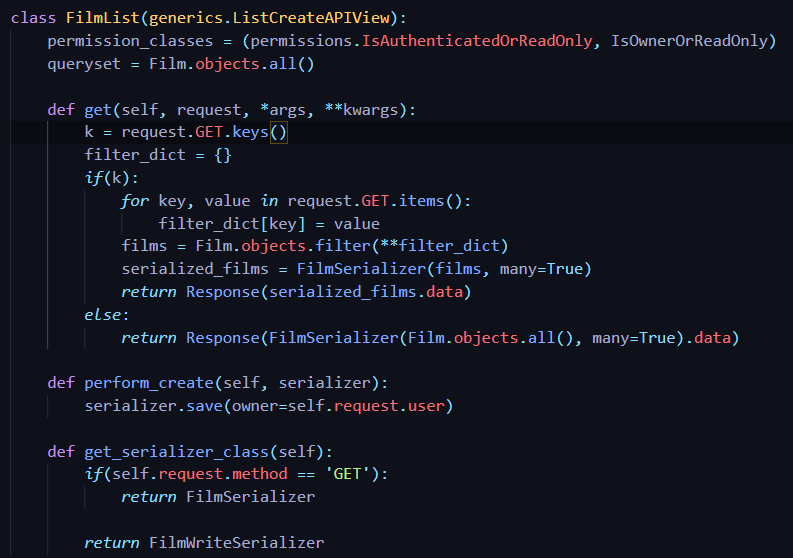
**Récupération et Modification des utilisateur par List ou par id**



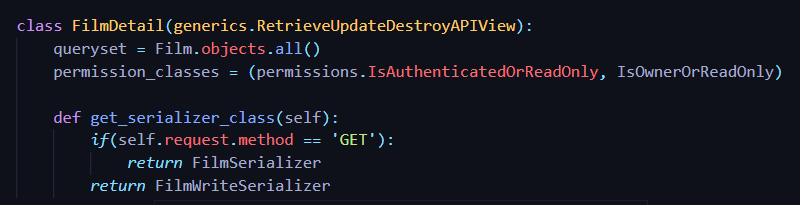
**L’utilisation des api Generics**

**Récupération et Modification des Films par List ou par id**

**Parcourant la requête et récupère les données dans un dictionnaire**



**L’utilisation des api Generics**

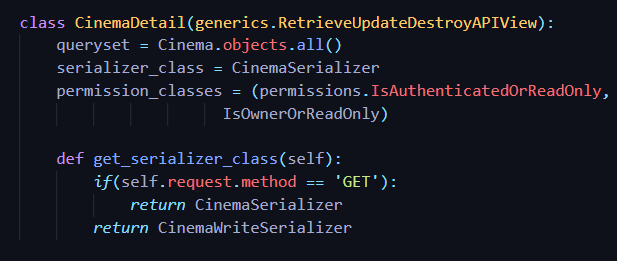


**Récupérer les détails sur un film précis**

**Récupération et Modification des Cinémas par List ou par id**

**Parcourant la requête et récupère les données dans un dictionnaire**

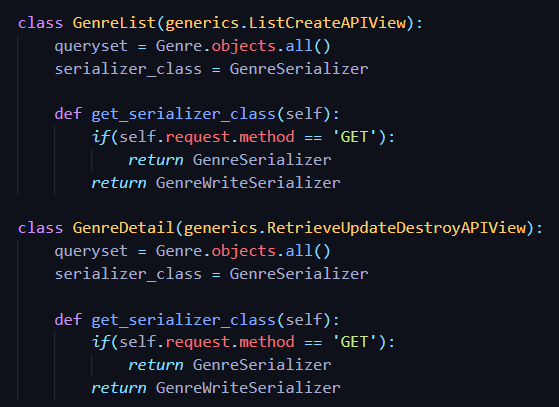




**Récupérer les détails sur un cinema précis**

**Récupération et Modification des Genres par List ou par id**

**Parcourant la requête et récupère les données dans un dictionnaire**



**L’utilisation des api Generics**

1. **Router l’application**

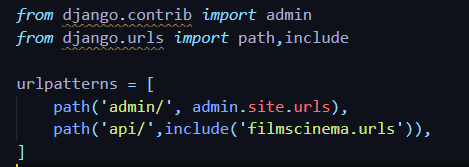
* Dans **filmscinema**\**urls.py** on ajoute des path et des url pour faire interagir notre api



A ce niveau, l’application propose un URI : **http://api.yourcompany.com/films/**

Lorsqu’un client accédera à cette adresse, Django ira lire le fichier **url.py** qui rédigera le client vers la vue : **FilmList**

* Dans **FilmManagement**\**urls.py** on ajoute des path et des url pour faire interagir notre api

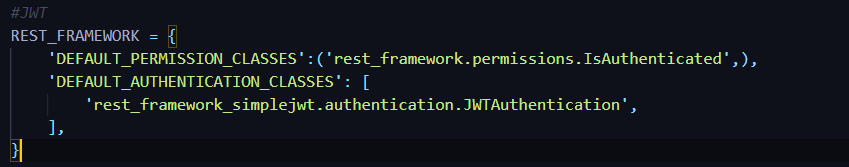


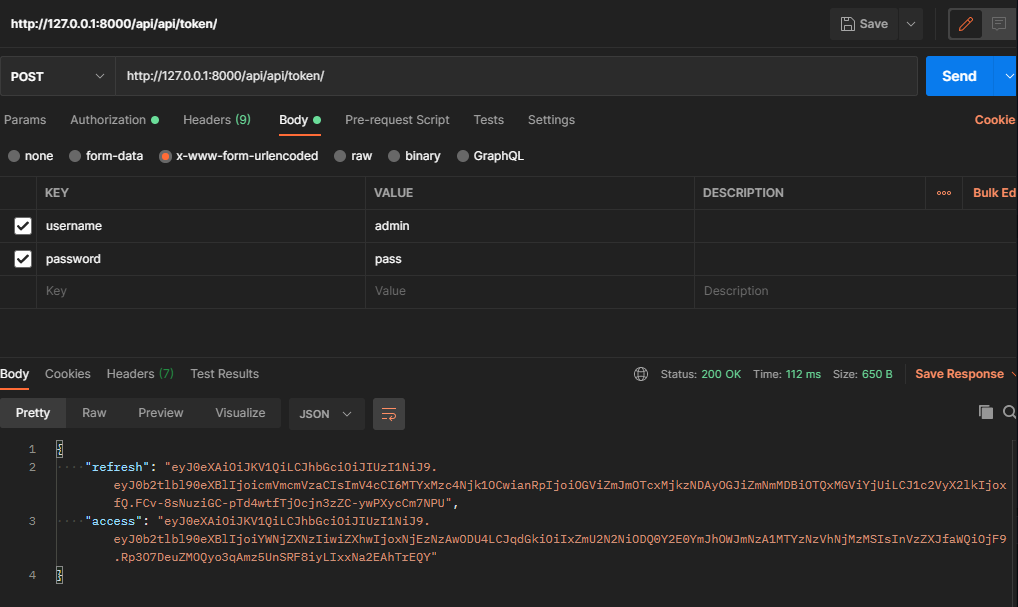
Le fichier **url.py** a pour but de gérer le routage de l’application. Pour faire simple, lorsqu’un client appel une ressource via un URI, ce fichier permet de faire la liaison entre l’extension de l’adresse et la vue adéquate

**TEST**

1. **Tester notre api avec Postman en appliquant le** **JWTAuthentification**

* L’utilisation de JWT nécessite un package
* **pip install djangorestframework\_simplejwt**
* **Pour appliquer le JWTAuthentification on doit ajouter à la fin de notre sittings.py la configuration :**

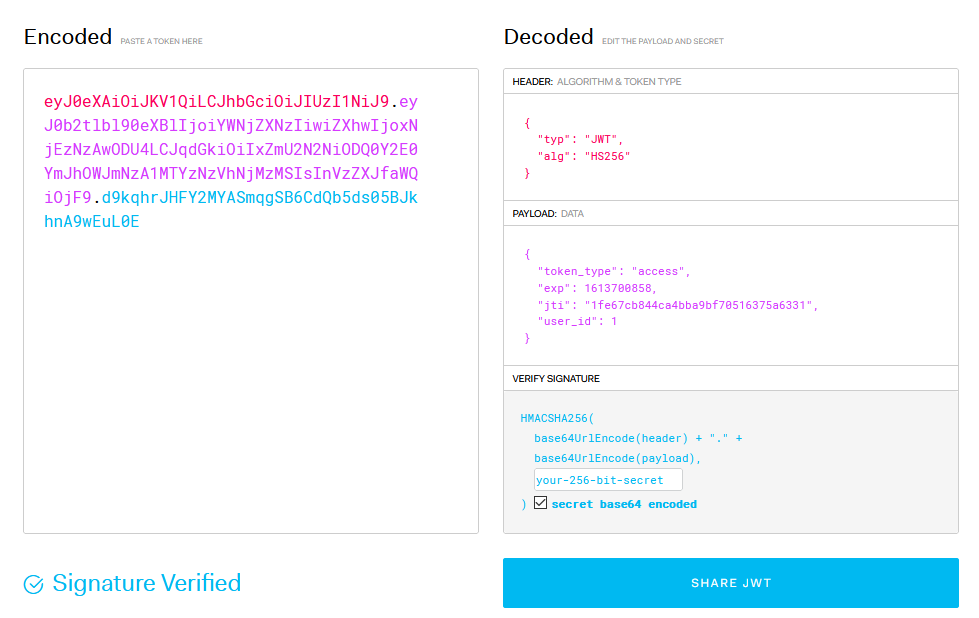




**Cette méthode allow post seulement pour récupérer le Token**

**On passe comme argument le nom d’utilisateur et le mot de passe de notre super User**

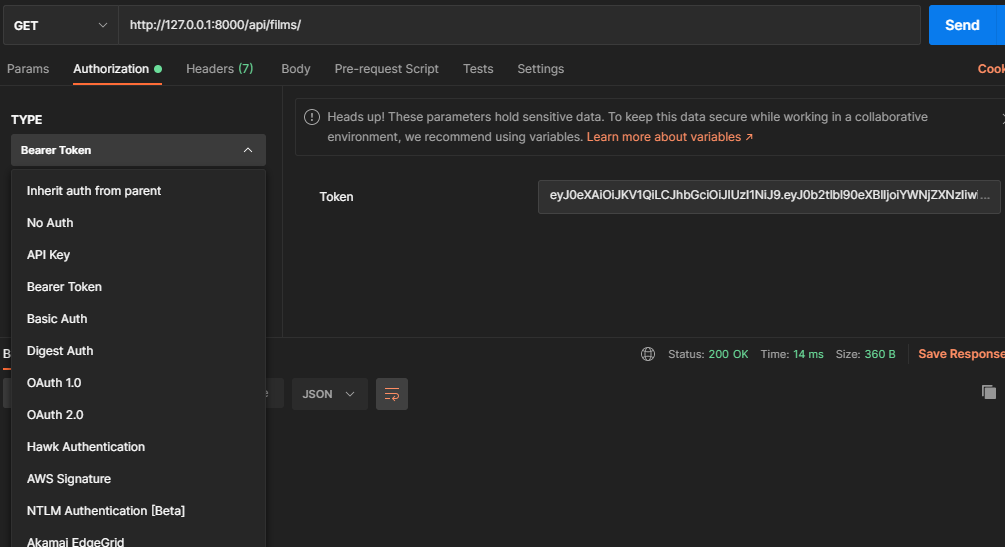
**Le Token Access expire après 5min.**



**Dans le site officiel de jwt.io**

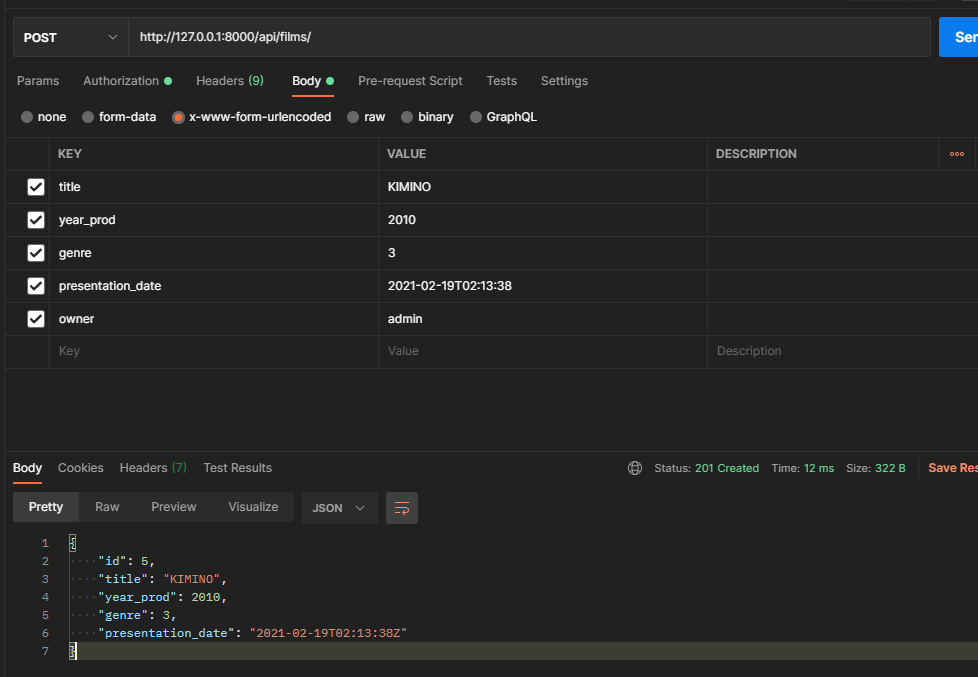
**On peut tester la validité de notre Token qui est encodé sur Base64**

**Pour visualiser nos données ou faire des manipulations on doit**



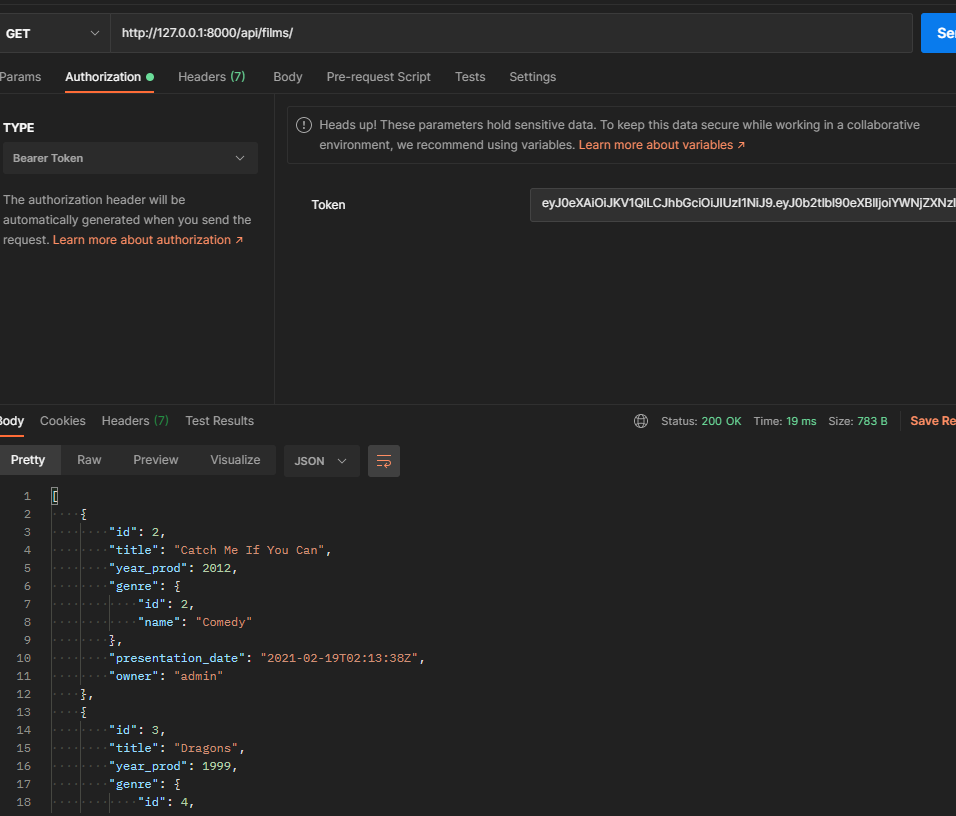
**Puis on le colle dans ce champ**

**Dans Autorisation   
on choisit « Bearer Token »**

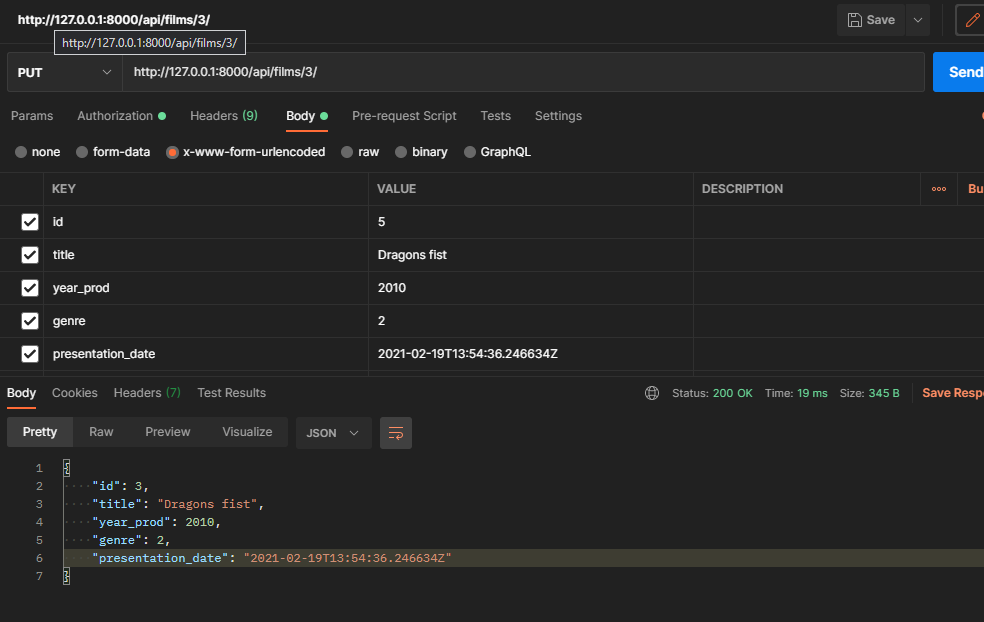
* **Test sur FILMS**

**Testons la méthode POST**

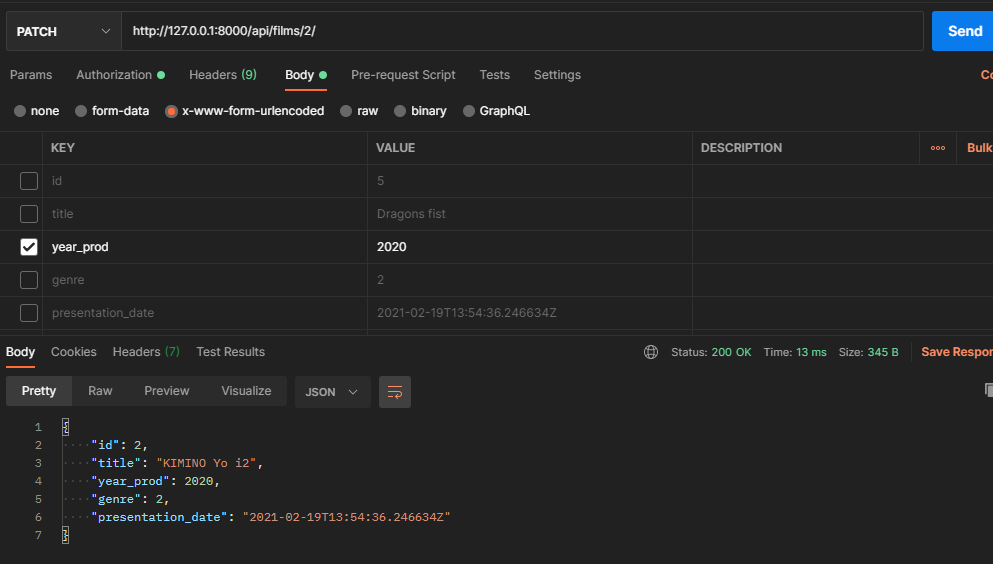
**On peut poster les données par Json ou x-www-form**



**Testons la méthode GET**

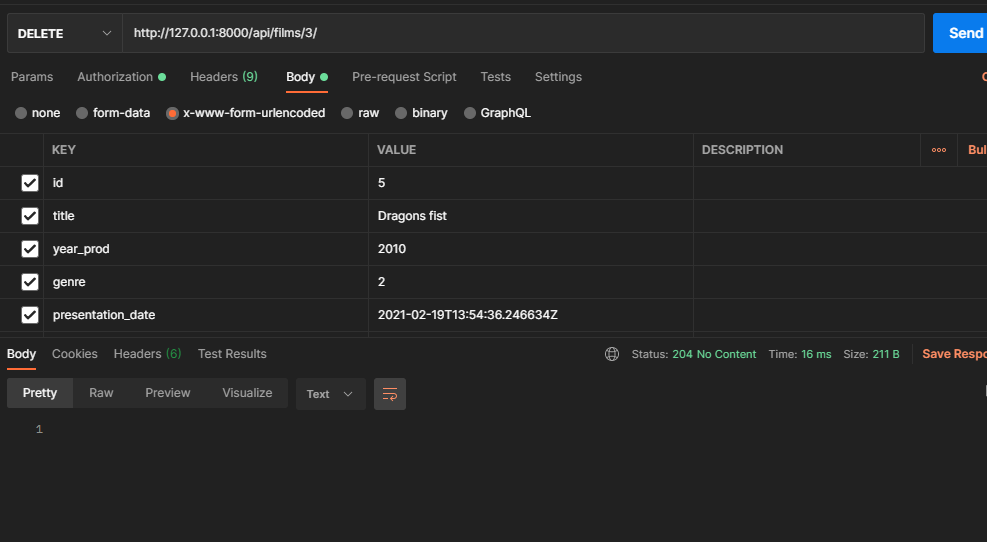


**Testons la méthode PUT**

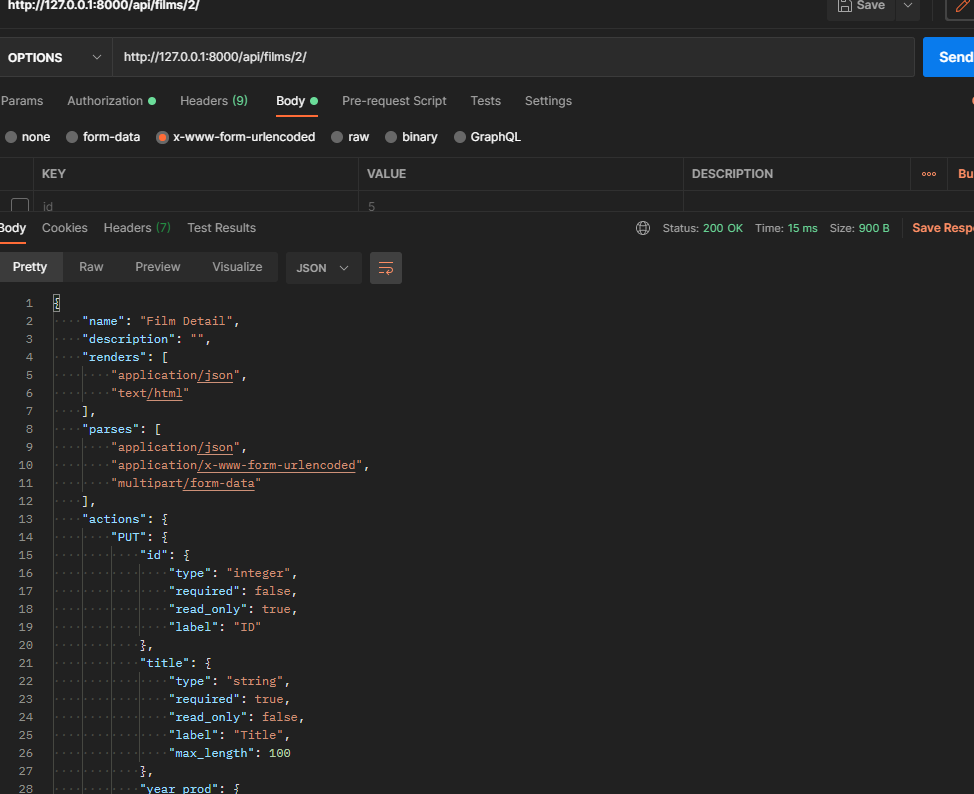


**Testons la méthode PATCH**

* **Permet de modifier un champ précis**

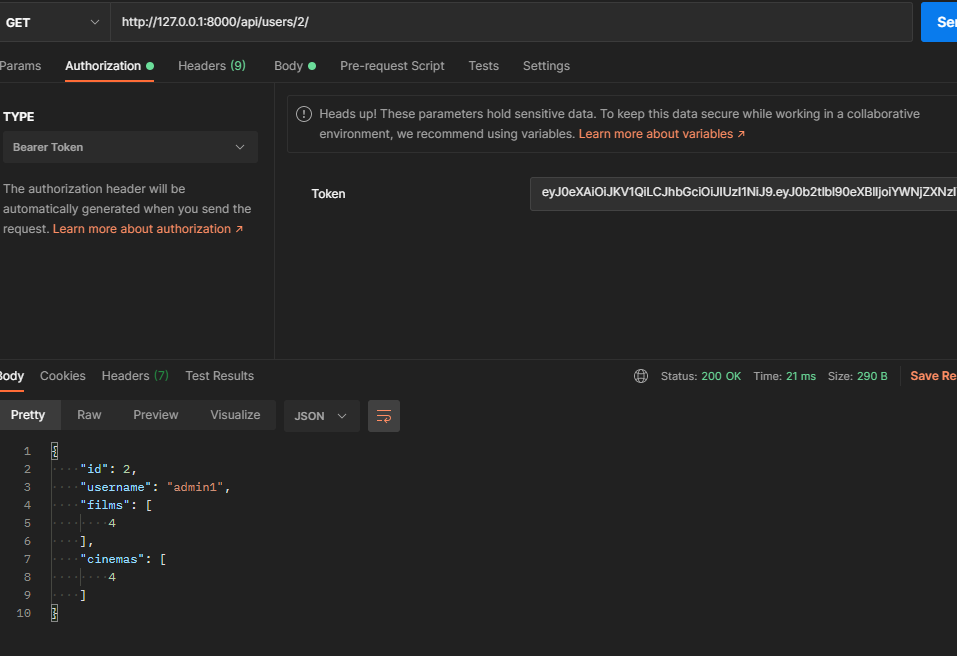


**Testons la méthode DELETEE**

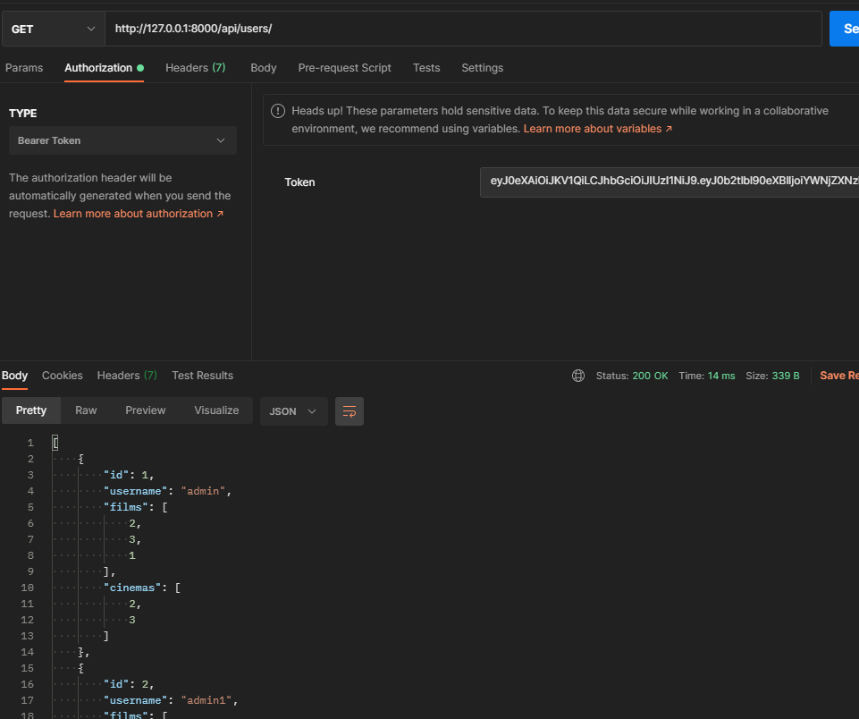


**Testons la méthode OPTIONS**

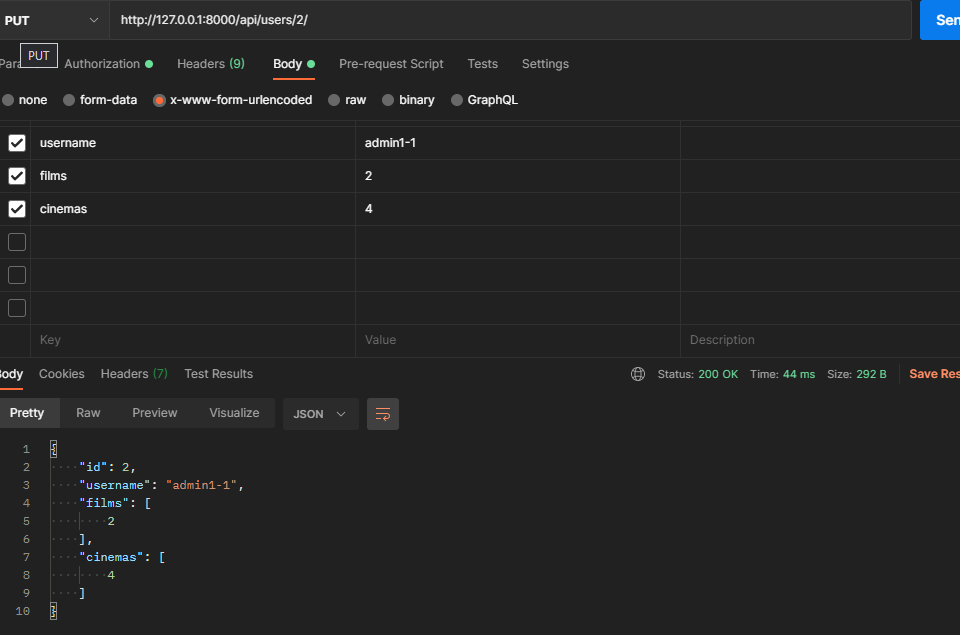
**Pour savoir quel méthodes un serveur support**

* **Test sur User**

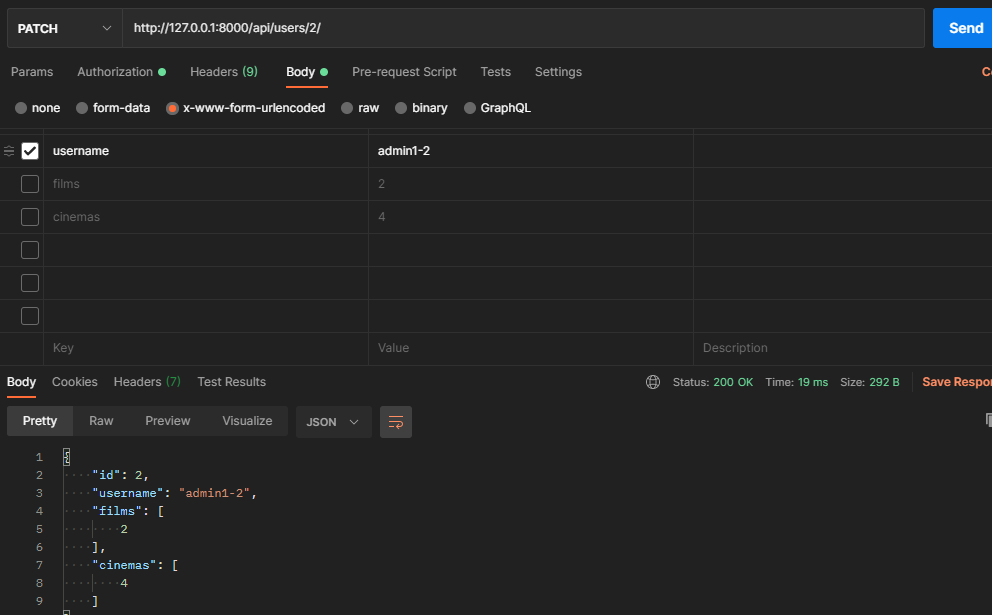
**Testons la méthode GET by ID**



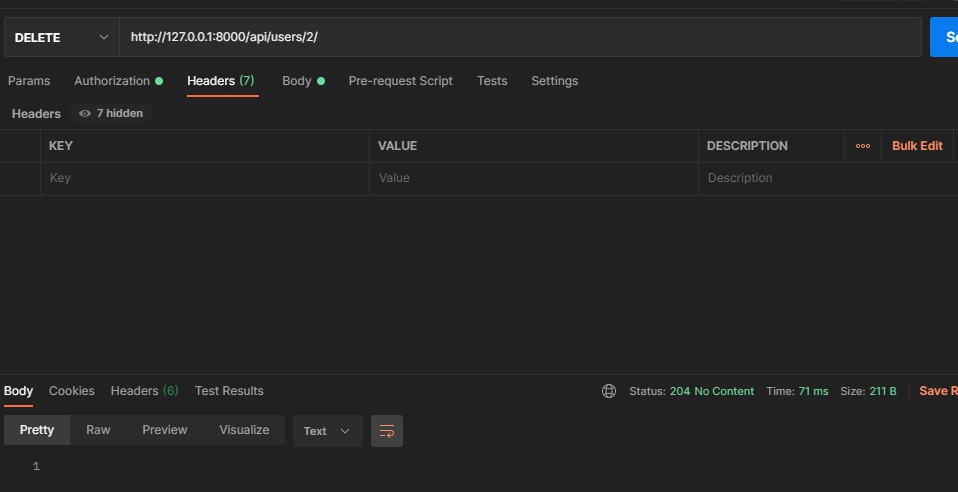
**GET tous les utilisateurs**



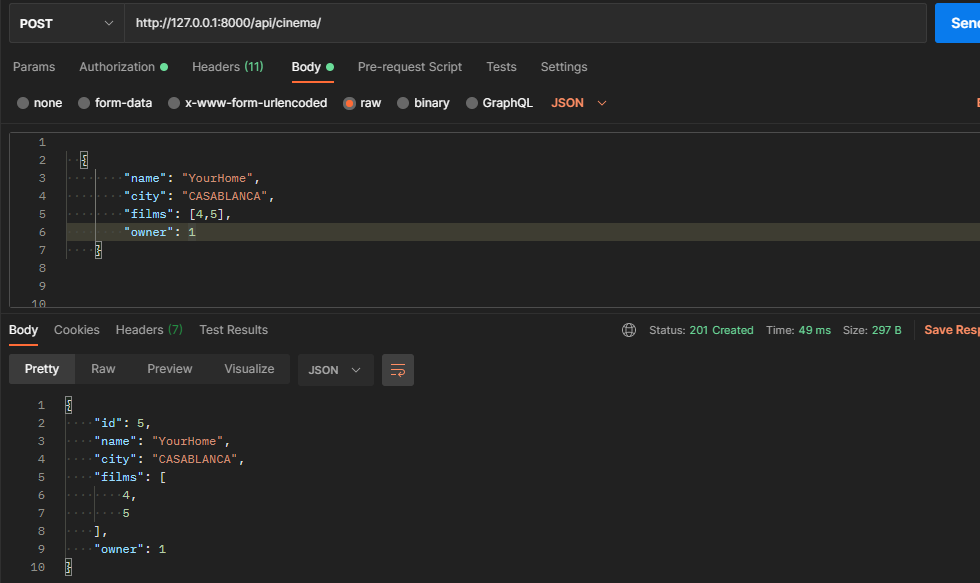
**PUT : modifier user dont id = 2**



**Modifier le nom d’utilisateur par la méthode PATCH**

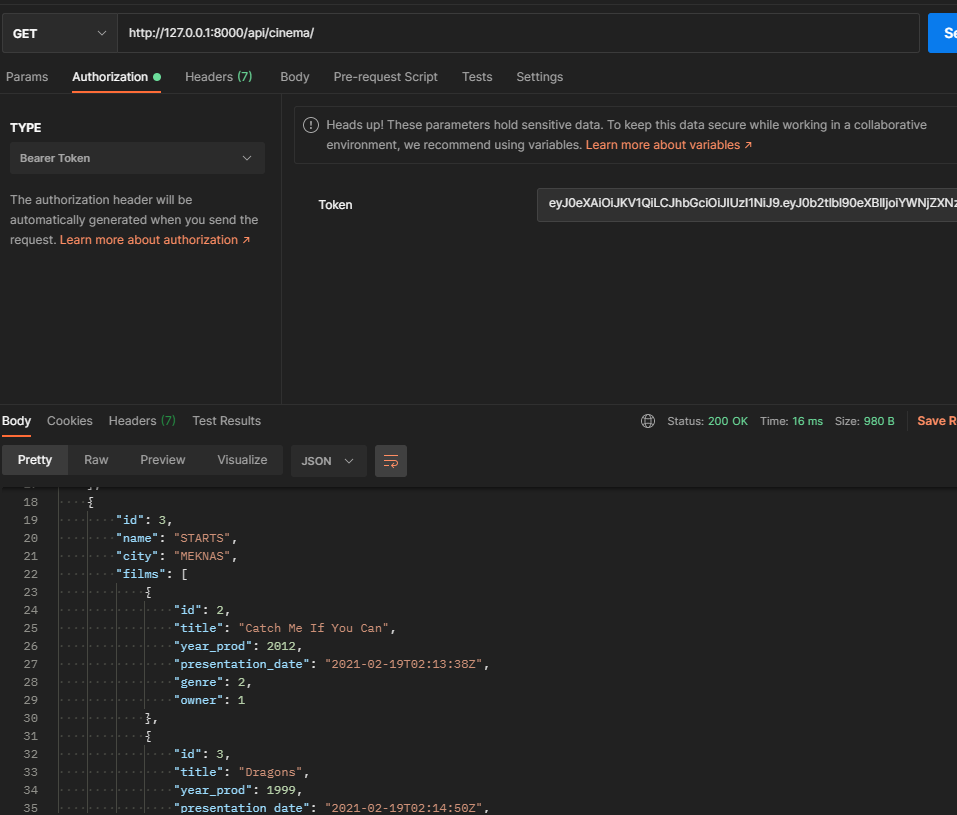


**La méthode DELETEE  
supprimer user avec id :2**

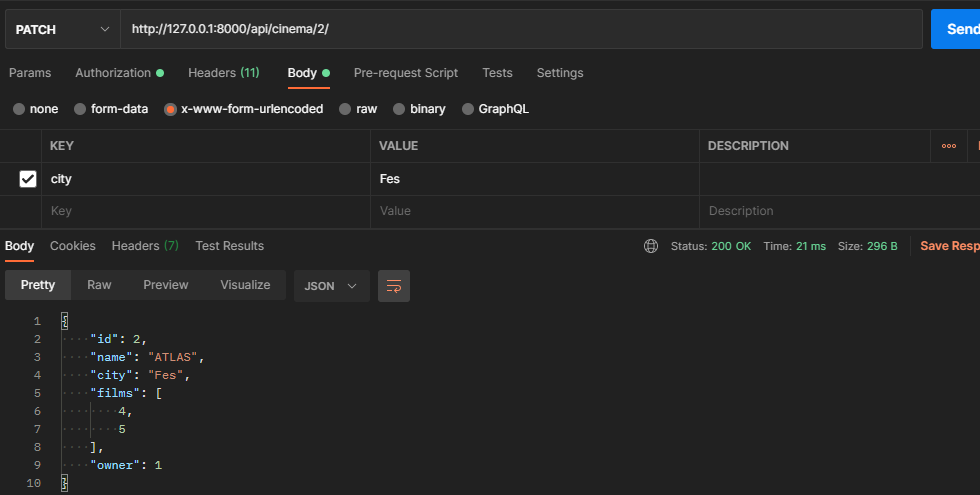
* **Test sur Cinema**

**La méthode POST**

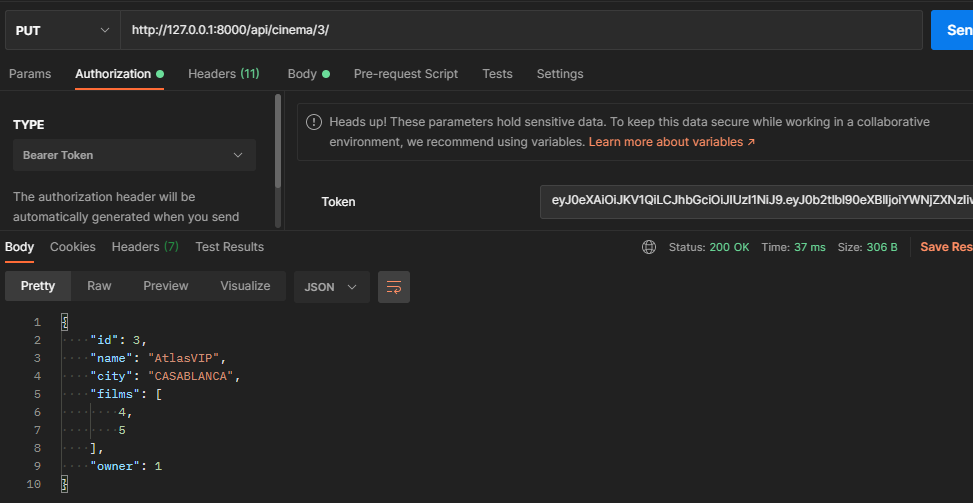
**Ajouter un cinéma avec le nom yourHome et ville : Casablanca et les films avec id : 4, 5 et utilisateur 1**



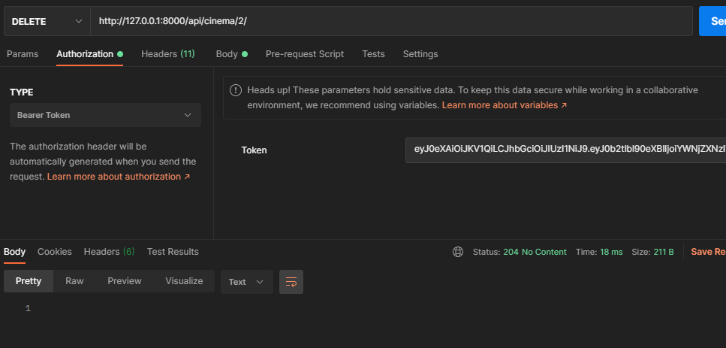
**Lister tous les cinémas**



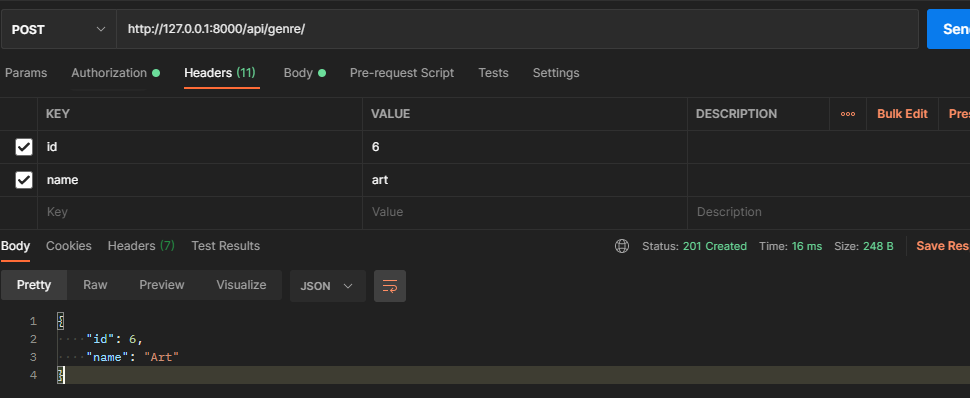
**Modifier la ville à Fès**



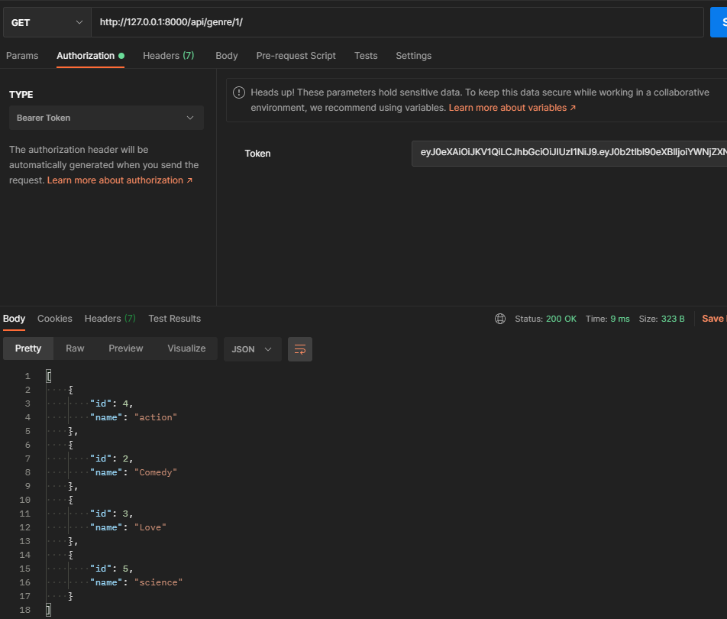
**PUT On passant tous les champs requis**



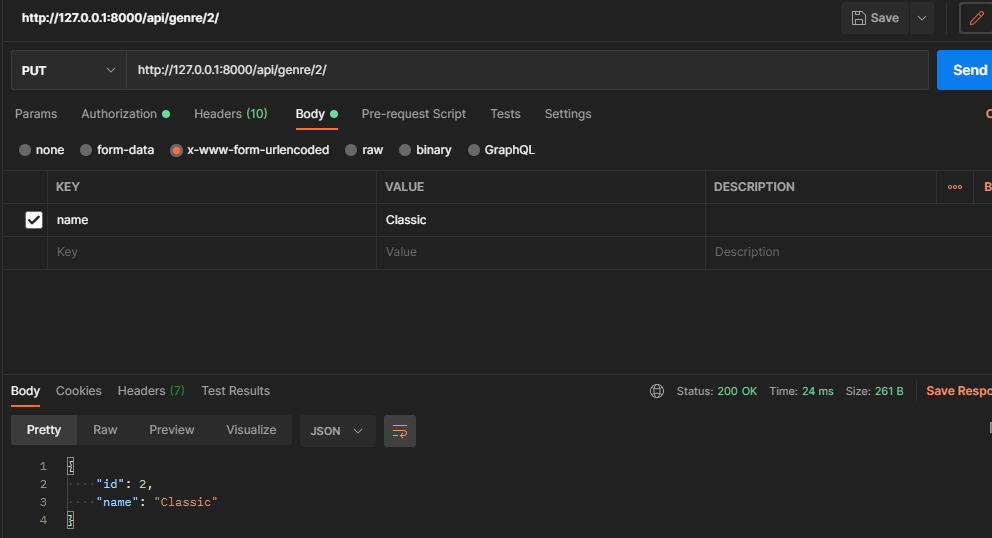
**DELETE le cinéma id : 2**

* **Test sur Genre**

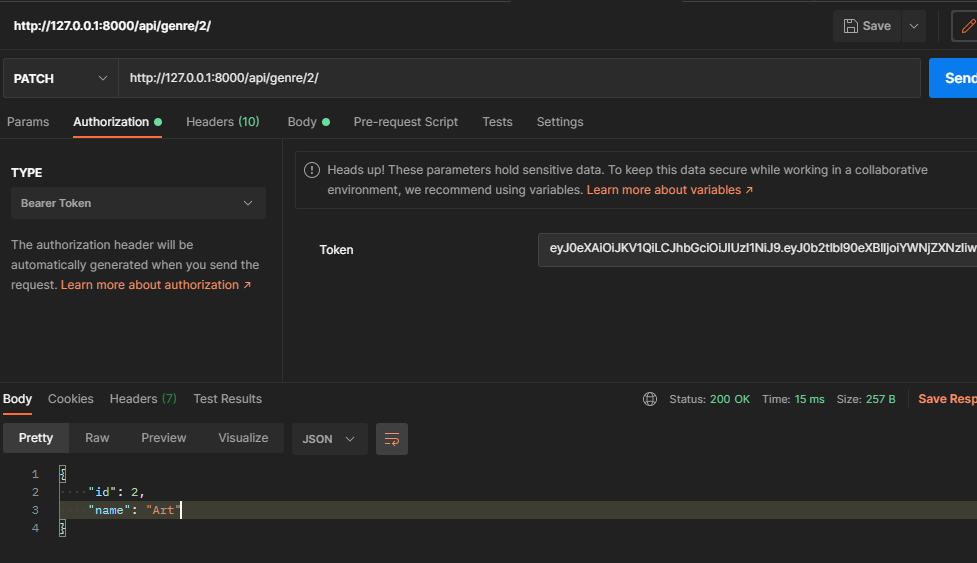
**Passer un genre avec Name : art**



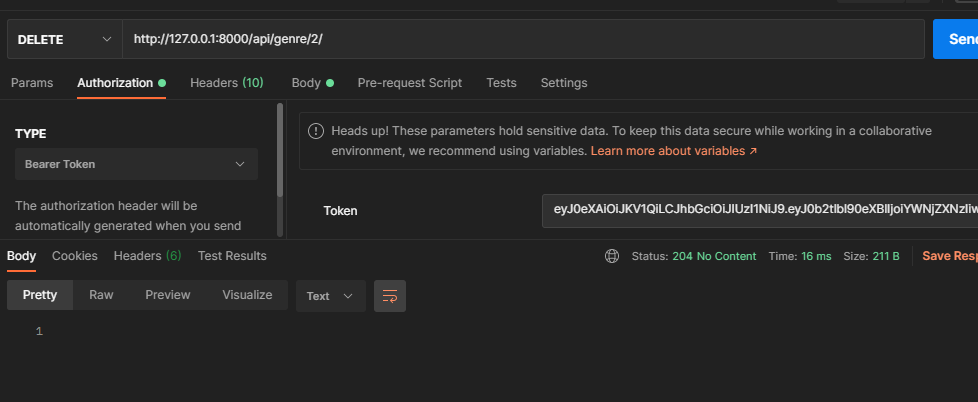
**Lister tous les genres**



**PUT Méthode**

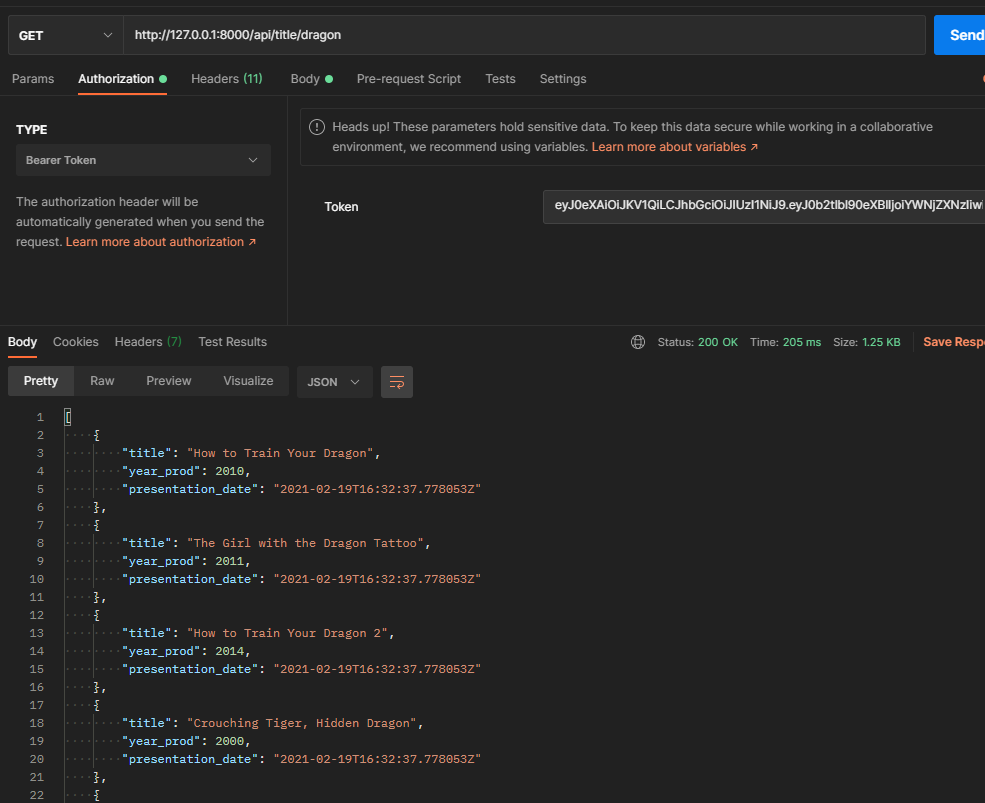


**Modifier un seul champ par PATCH**



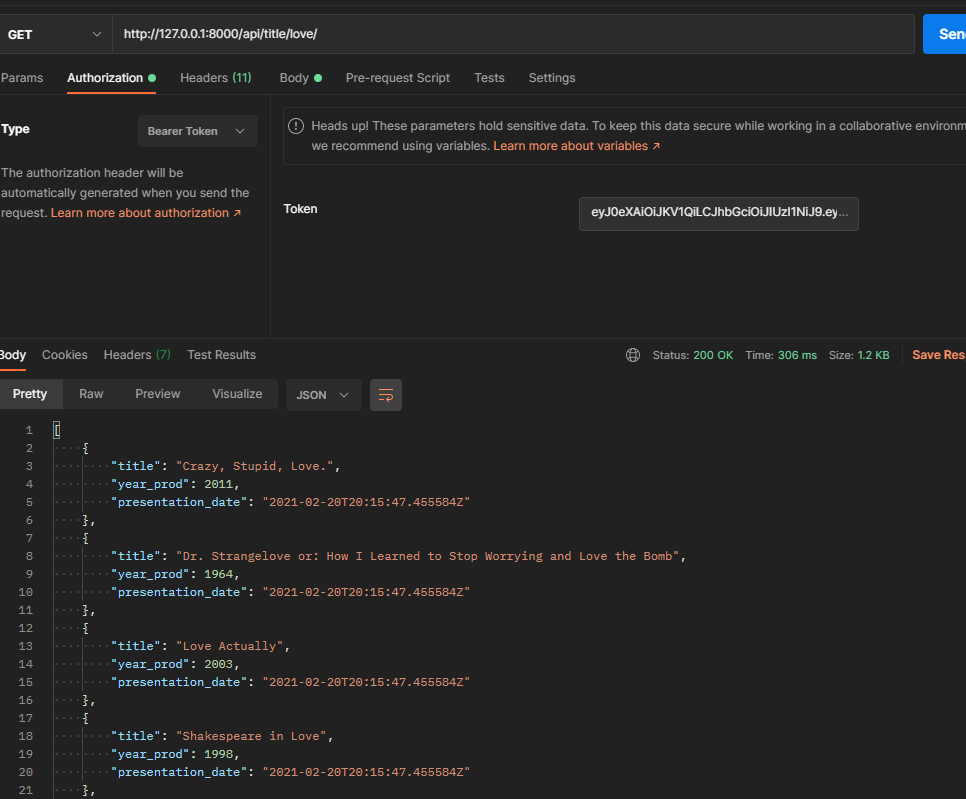
**DELETE :**

**Supprimer le genre a id : 2**

* **Test sur TITLE**

**Faisant appeler à une API externe qui permet de récupérer les titres et l’année de production avec le code dans la partie 9, on peut directement les lister et remplir notre base de données par la suite.**

**Cherchant les films contenant le nom « dragon » dans le titre.**

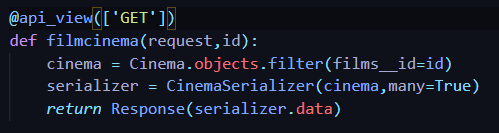


**Cherchant les films contenant le nom « love » dans le titre.**

* **Join FILM et CINEMA**

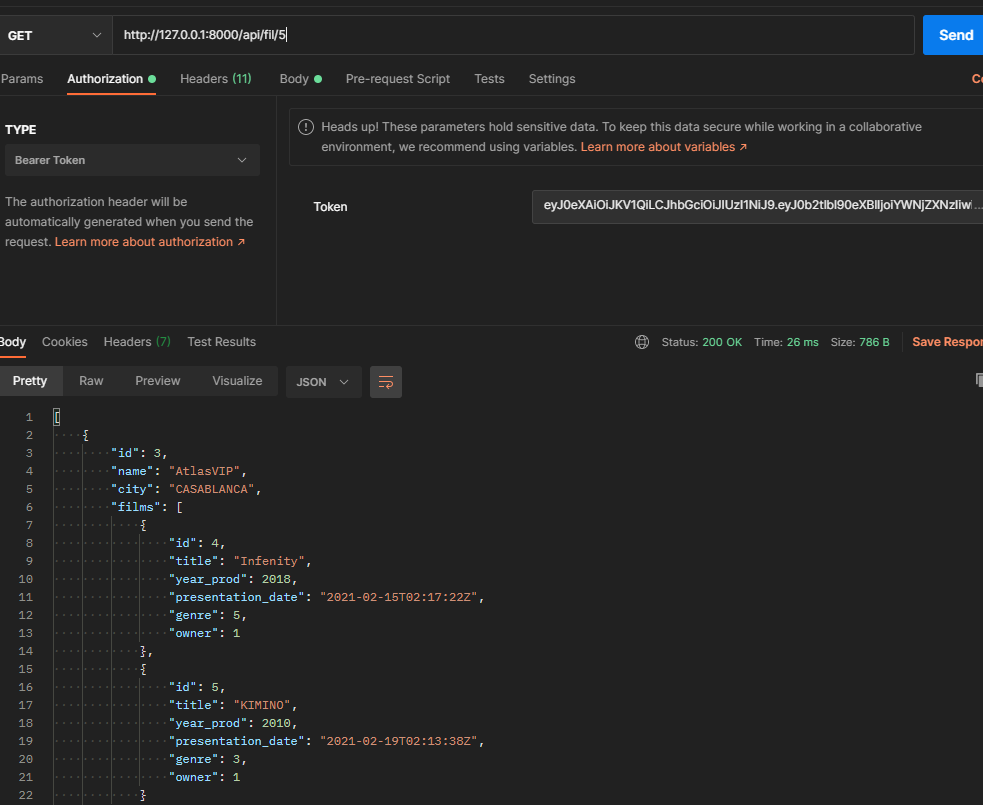
**Affichant tous les cinémas qui vont présenter un film passé par url**

**Pour effectuer la jointure on doit appeler la méthode filter() en passant id de l’autre table et api\_view directement détecte la relation entre les deux tables**





**Pour retourner les résultats on doit créer un url**



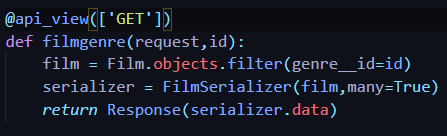
**Les cinémas ayant le film id : 5 « KIMINO »**

* **JOIN FILM et GENRE**

**Pour effectuer la jointure on doit appeler la méthode filter () en passant id de l’autre table et api\_view directement détecte la relation entre les deux tables**

**FILM et GENRE**

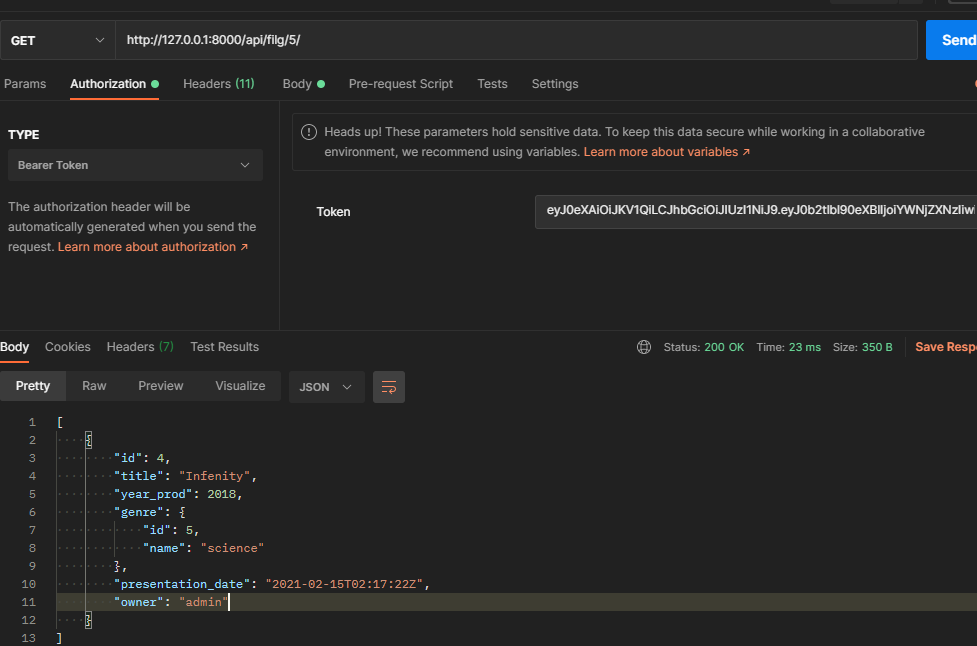
**Affichant tous les films qui appartient à un genre passé par url**



**Urls.py**



**Pour retourner les résultats on doit créer un url**



**Les films ayant le genre id :5 « science »**

**Nous avons vu comment Django REST Framework facilite la création de ressources d'une API REST, augmentant considérablement notre productivité.**

**Django REST Framework n'est pas limité aux seuls sujets traités dans ce rapport, dans la documentation DRF, vous avez de nombreuses autres fonctionnalités liées aux API REST telles que la limitation, la documentation, les validateurs, le contrôle de version et bien plus encore.**

**DRF a également des capacités pour prendre en charge le modèle d'API HATEOAS. Comme Django, Django REST Framework est extrêmement puissant, fiable et peut vous aider à résoudre vos problèmes avec le moins d'effort.**

**CONCLUSION**