



Challenge technique

Rédiger par :

- Mahamdi Mohammed

Alger le : 14/01/2021

Table des matières

Questions.....	3
Question 1	3
Question 2	3
Réponses	4
Réponse 1	4
Réponse 2	7
Les technologies utilisées	7
Les modèles créés	7
Plan des tests.....	8
Démonstration	9
Les étapes nécessaires pour exécuter le code source	11

Table de figures

Figure 1 diagramme cas d'utilisation n° 1	4
Figure 2 diagramme cas d'utilisation n° 2	4
Figure 3 diagramme cas d'utilisation n° 3	5
Figure 4 diagramme cas d'utilisation n° 4	5
Figure 5 diagramme cas d'utilisation n° 5	6
Figure 6 Exécution des tests	8
Figure 7 Figure 6 Exécution des tests détaillée	9
Figure 8 Création du client C1	9
Figure 9 Création du produit P1	10
Figure 10 Création du produit P2	10
Figure 11 vérification du prix total	11

Questions

Question 1

Faites la conception des use-cases suivants :

1. Permettre à l'administrateur de gérer une liste de produits qui ont des prix.
2. Permet à l'administrateur de fixer des prix concrets (tels que 1000 da) et des remises sur les prix, soit d'un montant concret (-200 da), soit d'un pourcentage (-10%).
3. Permet à l'administrateur de regrouper les produits pour former des offres groupées à des prix indépendants.
4. Permettre aux clients d'obtenir la liste des produits et leurs prix respectifs.
5. Permettre aux clients de passer une commande pour un ou plusieurs produits et fournir aux clients la liste des produits et le prix total.

Question 2

A l'aide des technologies choisies, développez les interfaces REST suivantes :

1. Permettre à l'administrateur de gérer une liste de produits qui ont des prix.
2. Permettre aux clients de passer une commande pour un ou plusieurs produits et fournir aux clients la liste des produits et le prix total.

Réponses

Réponse 1

1. Permettre à l'administrateur de gérer une liste de produits qui ont des prix.

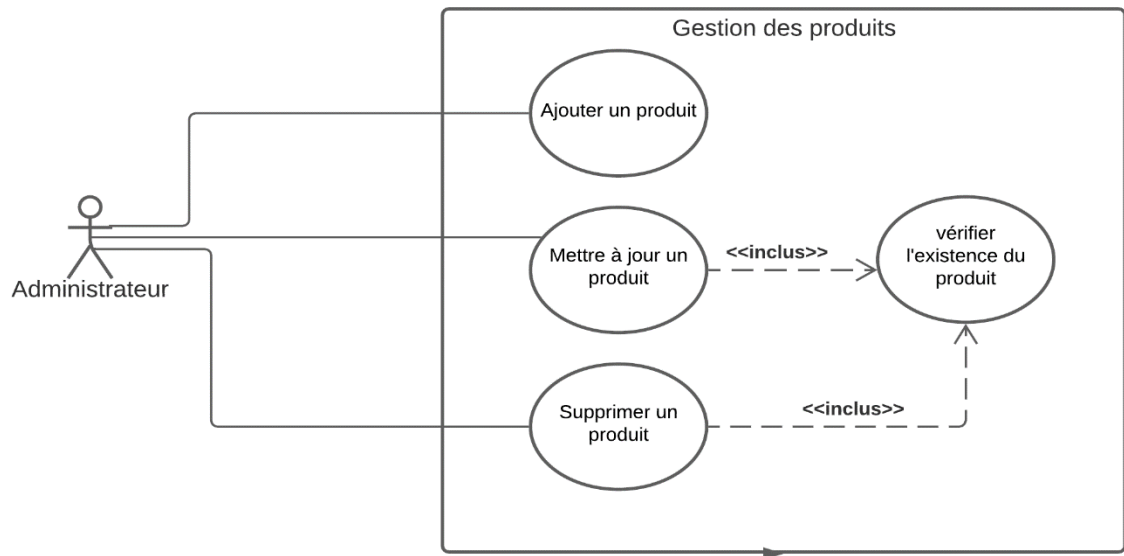


Figure 1 diagramme cas d'utilisation n° 1

2. Permet à l'administrateur de fixer des prix concrets (tels que 1000 da) et des remises sur les prix, soit d'un montant concret (-200 da), soit d'un pourcentage (-10%).

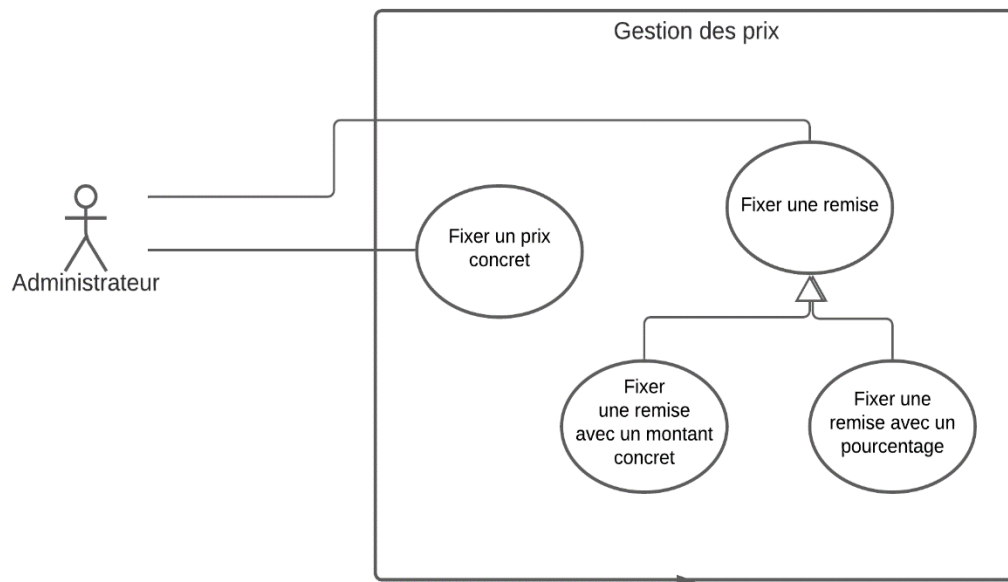


Figure 2 diagramme cas d'utilisation n° 2

3. Permet à l'administrateur de regrouper les produits pour former des offres groupées à des prix indépendants.

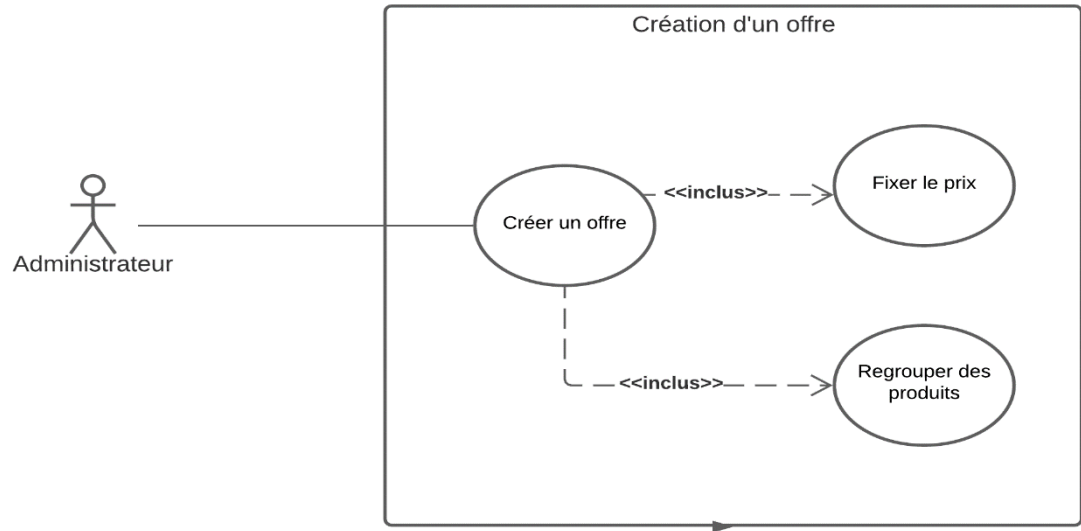


Figure 3 diagramme cas d'utilisation n° 3

4. Permettre aux clients d'obtenir la liste des produits et leurs prix respectifs.

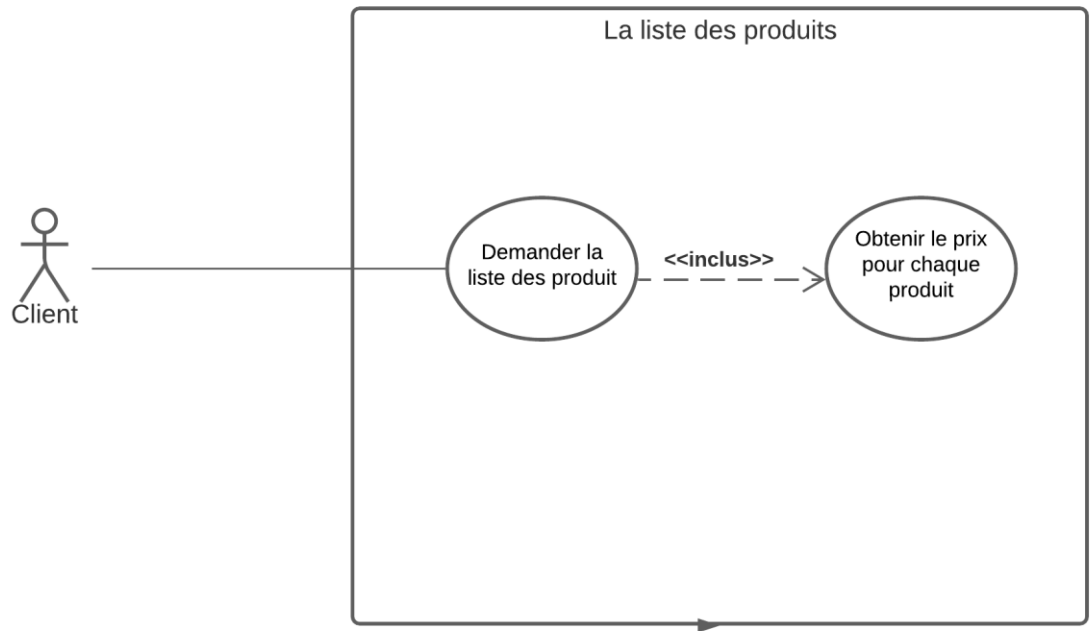


Figure 4 diagramme cas d'utilisation n° 4

5. Permettre aux clients de passer une commande pour un ou plusieurs produits et fournir aux clients la liste des produits et le prix total.

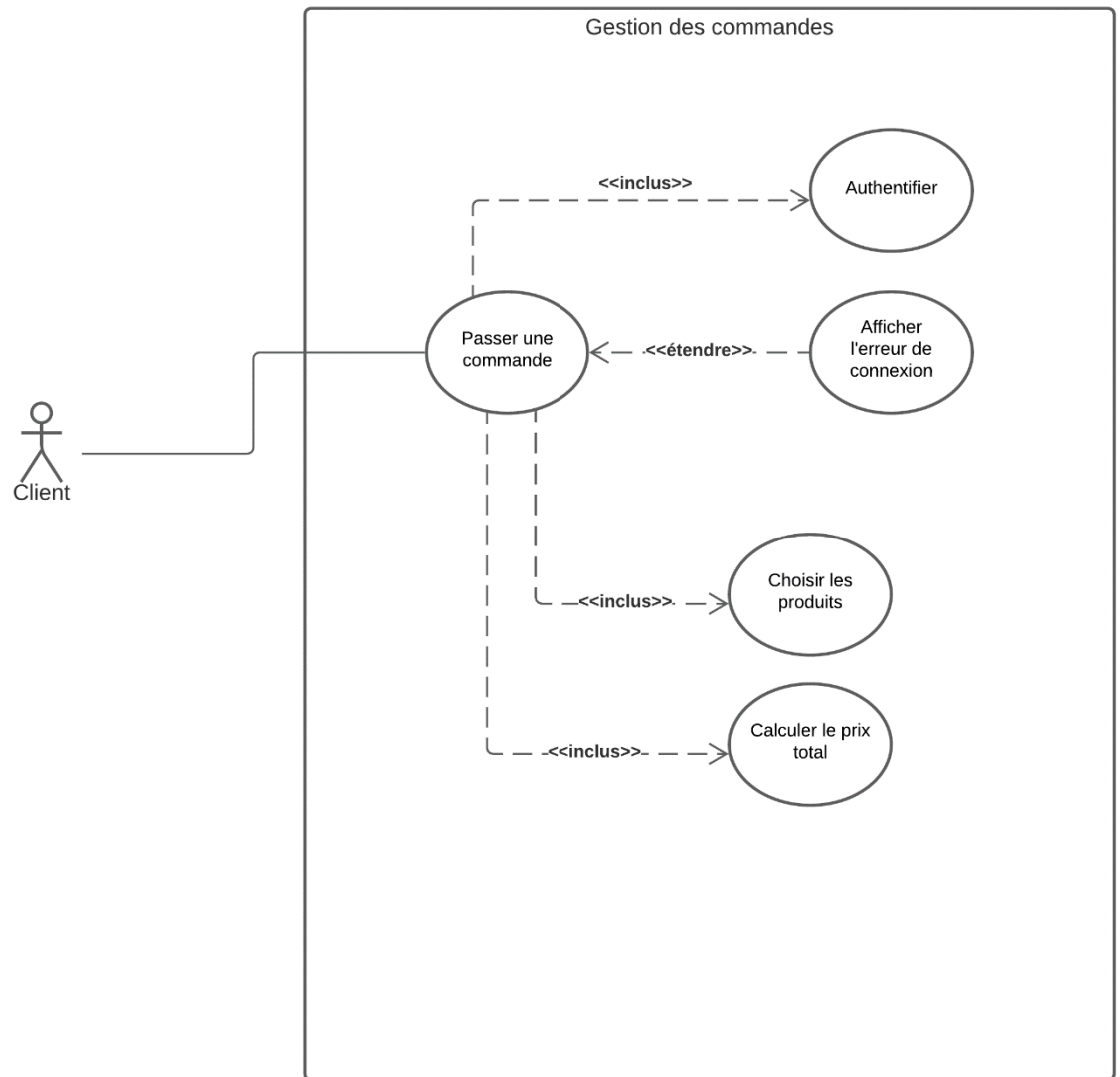


Figure 5 diagramme cas d'utilisation n° 5

Réponse 2

Les technologies utilisées

- Python.
- Django.
- DjangoRestFramework.
- Git.

Les modèles créés

Product

Champs	Type	La signification	Champ obligatoire ?
code	Chaîne de caractère	Code produit	Oui
name	Chaîne de caractère	Le nom du produit	Oui
family	Chaîne de caractère	La famille de produit	Non
price	Numérique	Le prix	Oui
remark	Chaîne de caractère	Les remarques	Non

Client

Champs	Type	La signification	Champ obligatoire ?
code	Chaîne de caractère	Code client	Oui
first_name	Chaîne de caractère	Le nom	Oui
last_name	Chaîne de caractère	Le prénom	Oui
address	Chaîne de caractère	L'adresse	Oui
date_of_birth	Date	La date de naissance	Oui
mobile_phone1	Chaîne de caractère	Numéro de téléphone 1	Oui
mobile_phone2	Chaîne de caractère	Numéro de téléphone 2	Non
email	Email	L'adresse mail du client	Non
company	Chaîne de caractère	La société du client	Non

Order

Champs	Type	La signification	Champ obligatoire ?
code	Chaîne de caractère	Code de la commande	Oui
date	Date	La date	Oui
client	Référence un client	Le client	Oui
products	Reference une liste des produit	La liste des produits	Oui

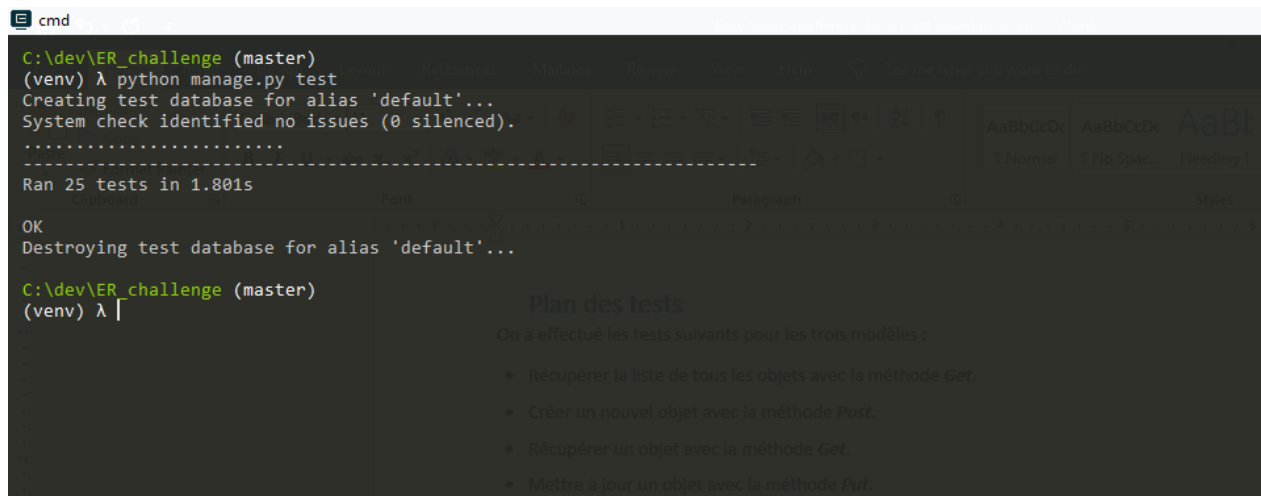
Plan des tests

On a effectué les tests suivants pour les trois modèles :

- Récupérer la liste de tous les objets avec la méthode **Get**.
- Créer un nouvel objet avec la méthode **Post**.
- Récupérer un objet avec la méthode **Get**.
- Mettre à jour un objet avec la méthode **Put**.
- Supprimer un objet avec la méthode **Delete**.

Finalement on tester que le montant total à payer est égale à la somme des prix des produits présent dans la commande.

Tous les tests sont passés correctement.



```
cmd
C:\dev\ER_challenge (master)
(venv) λ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 25 tests in 1.801s

OK
Destroying test database for alias 'default'...

C:\dev\ER_challenge (master)
(venv) λ |
```

Figure 6 Exécution des tests


```

System check identified no issues (0 silenced).
test_create_client (api.tests.ClientApiTest) ... ok
test_retrieve_clients_list (api.tests.ClientApiTest) ... ok
test_invalid_delete_client (api.tests.DeleteSingleClientTest) ... ok
test_valid_delete_client (api.tests.DeleteSingleClientTest) ... ok
test_invalid_delete_order (api.tests.DeleteSingleOrderTest) ... ok
test_valid_delete_order (api.tests.DeleteSingleOrderTest) ... ok
test_invalid_delete_product (api.tests.DeleteSingleProductTest) ... ok
test_valid_delete_product (api.tests.DeleteSingleProductTest) ... ok
test_get_invalid_single_client (api.tests.GetSingleClientTest) ... ok
test_get_valid_single_client (api.tests.GetSingleClientTest) ... ok
test_get_invalid_single_order (api.tests.GetSingleOrderTest) ... ok
test_get_valid_single_order (api.tests.GetSingleOrderTest) ... ok
test_get_invalid_single_product (api.tests.GetSingleProductTest) ... ok
test_get_valid_single_product (api.tests.GetSingleProductTest) ... ok
test_total_price (api.tests.GetTotalPriceTest) ... ok
test_create_order (api.tests.OrderApiTest) ... ok
test_retrieve_orders_list (api.tests.OrderApiTest) ... ok
test_create_product (api.tests.ProductApiTest) ... ok
test_retrieve_products_list (api.tests.ProductApiTest) ... ok
test_invalid_update_client (api.tests.UpdateSingleClientTest) ... ok
test_valid_update_client (api.tests.UpdateSingleClientTest) ... ok
test_invalid_update_order (api.tests.UpdateSingleOrderTest) ... ok
test_valid_update_order (api.tests.UpdateSingleOrderTest) ... ok
test_invalid_update_product (api.tests.UpdateSingleProductTest) ... ok
test_valid_update_product (api.tests.UpdateSingleProductTest) ... ok

-----
Ran 25 tests in 0.985s

OK
Destroying test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...

C:\dev\ER_challenge (master)
(venv) λ

```

Figure 6 Exécution des tests

Figure 7 Figure 6 Exécution des tests détaillée

Démonstration

Dans le suivant, on va créer un client *C1*, deux produits *P1* et *P2* et une commande pour le client *C1* qui contient les produits *P1* et *P2*.

Method	URL	Status	Time	Size
POST	http://127.0.0.1:8000/client/	201 Created	548 ms	201 B

JSON	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
<pre> 1 { 2 "code": "C1", 3 "first_name": "client 1", 4 "last_name": "client 1", 5 "address": "Batna, Algeria", 6 "date_of_birth": "1990-01-01", 7 "mobile_phone1": "0123456789" 8 } </pre>					<pre> 1 { 2 "id": 4, 3 "code": "C1", 4 "first_name": "client 1", 5 "last_name": "client 1", 6 "address": "Batna, Algeria", 7 "date_of_birth": "1990-01-01", 8 "mobile_phone1": "0123456789", 9 "mobile_phone2": null, 10 "email": null, 11 "company": null 12 } </pre>			

Figure 8 Création du client C1

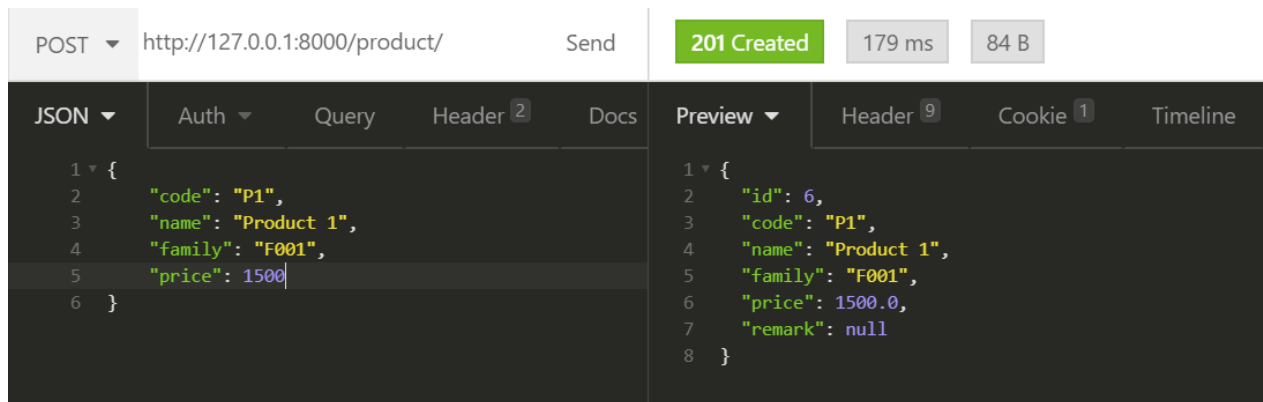


Figure 9 Création du produit P1

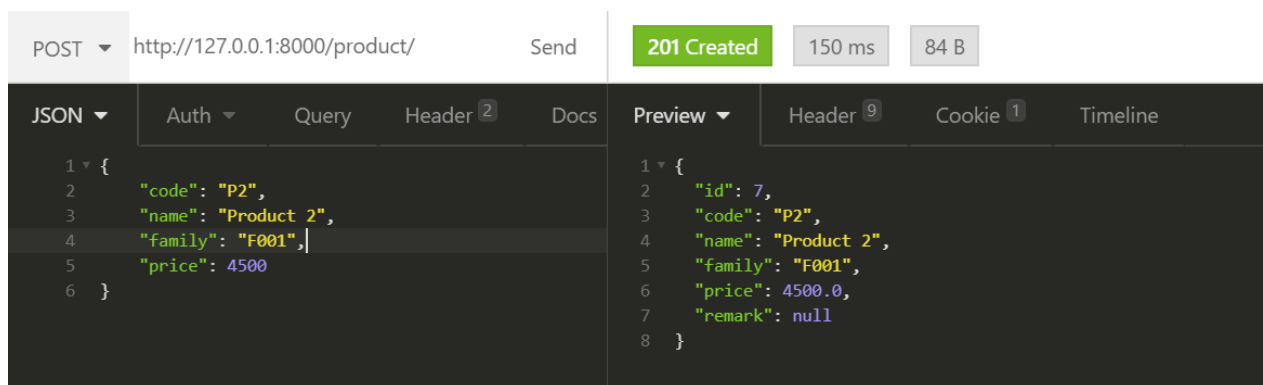
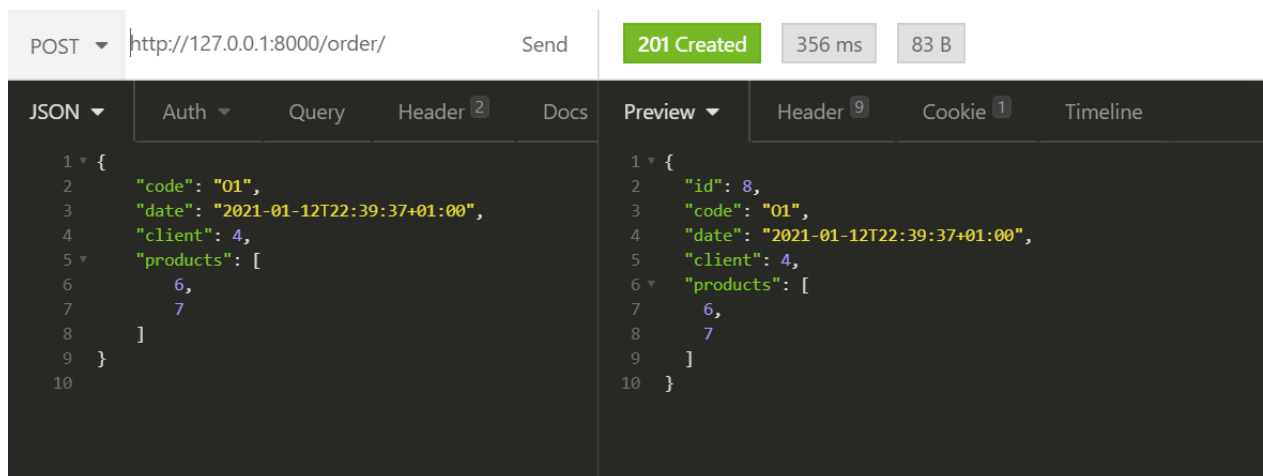


Figure 10 Création du produit P2

Maintenant on va créer une commande O1, on rappelle que l'id du client C1 est 4 et les ids du P1 et P2 sont 6 et 7 respectivement.



Une fois la commande est créée avec succès, on vérifie que le prix total doit être $1500 + 4500 = 6000$.

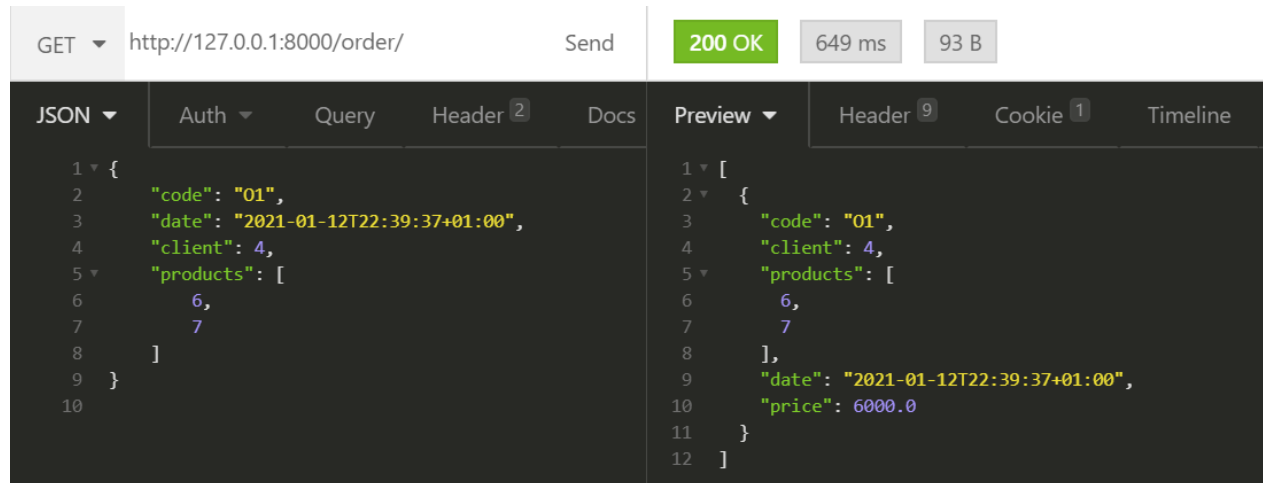


Figure 11 vérification du prix total

Les étapes nécessaires pour exécuter le code source

- Créer un environnement virtuel avec venv et l'activer.
- Installer les dépendances : `pip install -r requirements.txt`
- Makemigrations : `python manage.py makemigrations`
- Migrate : `python manage.py migrate`
- Création d'un super utilisateur : `python manage.py createsuperuser`
- Vérifier les tests : `python manage.py test`
- Lancer le server : `python manage.py runserver`