

# **API REST avec Django REST Framework**

## **V3 : interface d'administration, documentation**

**Prof. Samuel OUYA**

Objectif

**On désire mettre une API REST faisant CRUD sur le compte bancaire des presidents**

### **1- On prepare l'environnement de developpement**

```
apt-get install python3
```

```
apt-get install python3-pip
```

```
pip3 install virtualenv
```

```
root@im:/home/alain/django# python3 --version
```

```
Python 3.8.10
```

```
root@im:/home/alain/django# virtualenv --version
```

```
virtualenv 20.0.17 from /usr/lib/python3/dist-packages/virtualenv/__init__.py
```

### **créons l'environnement de developpement ven**

```
root@im:/home/alain/django# virtualenv venv
```

### **Activons l'environnement de developpement venv**

```
root@im:/home/alain/django# source venv/bin/activate
```

```
(venv) root@im:/home/alain/django#
```

### **Utilisons pip3 pour installer dans notre environnement**

```
(venv) root@im:/home/alain/django# pip3 install django
```

### **Installons le restframework**

```
(venv) root@im:/home/alain/django# pip3 install djangorestframework
```

## 2- Création du projet president

```
(venv) root@im:/home/alain/django# django-admin startproject president
```

```
(venv) root@im:/home/alain/django# ls
```

```
president venv
```

**On constate que le dossier presidentt est crée**

**Allons dans le dossier president pour demarrer le serveur django et le tester**

```
(venv) root@im:/home/alain/django# cd president/
```

```
(venv) root@im:/home/alain/django/president# ./manage.py runserver
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```

**On teste dans un navigateur l'accès à django sur le port 8000**

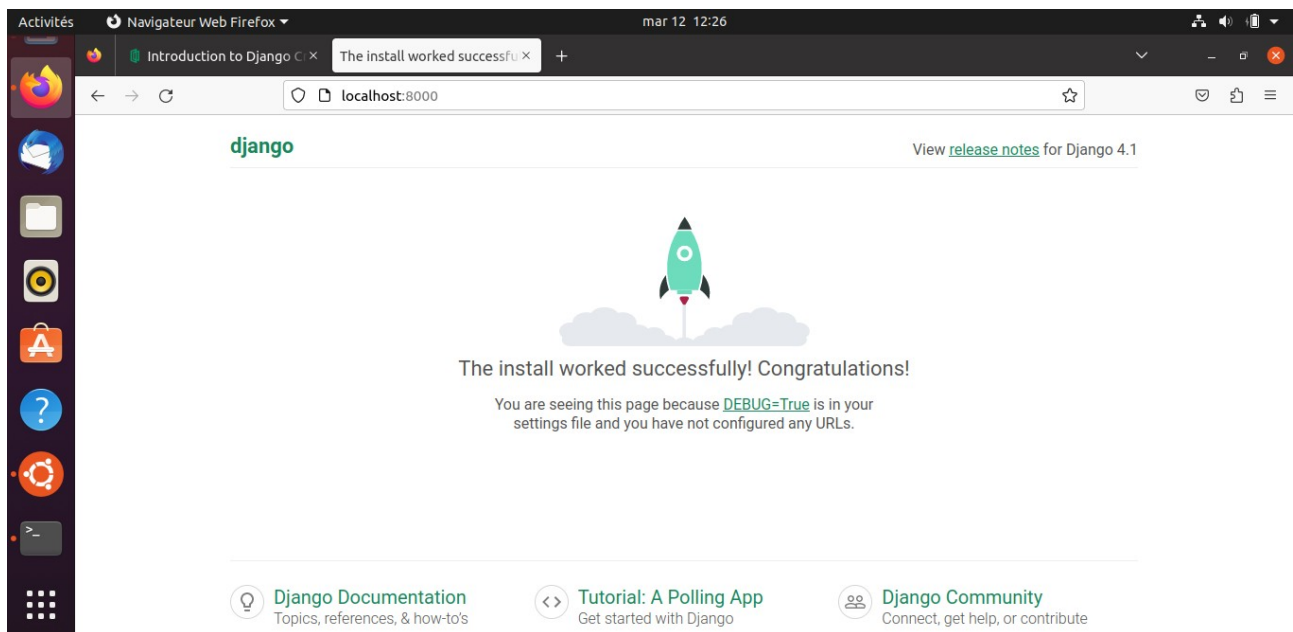


Fig 1 : Test de bon fonctionnement de Django

## 3- Creation de l'application banque

```
root@im:/home/alain/django/president# python3 manage.py startapp banque
```

```
(venv) root@im:/home/alain/django/president# ls
```

```
banque db.sqlite3 manage.py president
```

**On constate que le dossier banque est créé ainsi que la base db.sqlite3**

## 4- Paramétrage de la prise en compte de l'application banque et le framework rest par notre projet president

Pour cela, on édite le fichier du sous\_dossier président dans le répertoire president

```
root@im:/home/alain/django/president# nano president/settings.py
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'banque',  
    'rest_framework',  
]
```

On a ajouté les 2 dernières lignes en gras

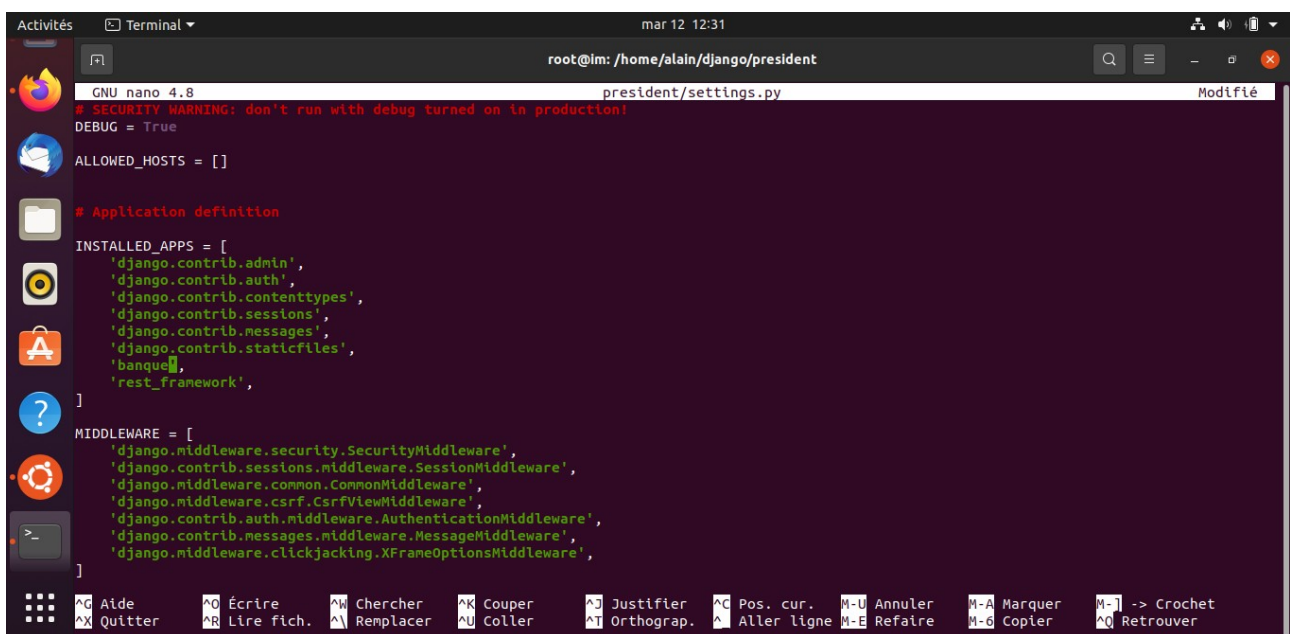


Fig 2 : Paramétrage de la prise en compte de l'application banque et du rest\_framework

## 5- Création du modèle Banque

Dans models.py du sous-dossier banque

```
# Create your models here.
```

```
from django.db import models
```

```
# Create your models here.
```

```
class Banque(models.Model):  
    prenom = models.CharField(max_length = 100)  
    nom = models.CharField(max_length = 100)  
    code = models.CharField(max_length = 100)  
    numcompte= models.CharField(max_length = 100)  
    solde = models.CharField(max_length = 100)
```

```
def __str__(self):  
    return self.prenom
```

## **6- Utilisation de migration pour préparer la création du modele banque**

```
(venv) root@im:/home/alain/django/president# ./manage.py makemigrations
```

## **7- Application de la migration**

```
(venv) root@im:/home/alain/django/president# ./manage.py migrate
```

## **8- Serialisation du modele Banque**

Dans le repertoire de l'application banque

Editer `serializers.py` et y ajouter

```
from rest_framework import serializers  
from banque.models import Banque  
class BanqueSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Banque  
        fields = "__all__"
```

## 9- Creation de la vue de notre Api **BanqueAPIView**

C'est ici, on definit la logique de votre application dans le fichier view.py du repertoire banque

(venv) root@im:/home/alain/django/president# **nano banque/views.py**

```
from rest_framework.generics import ListAPIView
from rest_framework.generics import CreateAPIView
from rest_framework.generics import DestroyAPIView
from rest_framework.generics import UpdateAPIView
from banque.serializers import BanqueSerializer
from banque.models import Banque
```

# Create your views here.

```
class ListBanqueAPIView(ListAPIView):
```

```
    """This endpoint list all of the available todos from the database"""
```

```
    queryset = Banque.objects.all()
```

```
    serializer_class = BanqueSerializer
```

```
class CreateBanqueAPIView(CreateAPIView):
```

```
    """This endpoint allows for creation of a todo"""
```

```
    queryset = Banque.objects.all()
```

```
    serializer_class = BanqueSerializer
```

```
class UpdateBanqueAPIView(UpdateAPIView):
```

```
    """This endpoint allows for updating a specific todo by passing in the id of the todo to update"""
```

```
    queryset = Banque.objects.all()
```

```
    serializer_class = BanqueSerializer
```

```
class DeleteBanqueAPIView(DestroyAPIView):
```

```
    """This endpoint allows for deletion of a specific Todo from the database"""
```

```
    queryset = Banque.objects.all()
```

```
    serializer_class = BanqueSerializer
```

## 10- Creation des endpoints URL pour l'accès à l'API

```
venv) root@im:/home/alain/django/president# nano banque/urls.py

from django.urls import path
from tbanque import views

urlpatterns = [

    path("",views.ListBanqueAPIView.as_view(),name="banque_list"),
    path("create/", views.CreateBanqueAPIView.as_view(),name="banque_create"),
    path("update/<int:pk>/",views.UpdateBanqueAPIView.as_view(),name="update_banque"),
    path("delete/<int:pk>/",views.DeleteBanqueAPIView.as_view(),name="delete_banque")

]
```

## 11- Déclaration des url enpoints de l'application banque dans notre projet president

```
venv) root@im:/home/alain/django/president# nano president/urls.py

from django.contrib import admin
from django.urls import path
from django.urls import include

urlpatterns = [

    path('admin/', admin.site.urls),
    path('api/v1/banque/',include("banque.urls"))

]
```

## 12- Test de notre Api

```
venv) root@im:/home/alain/django/president# ./manage.py runserver
```

Pour créer un compte, on utilise la methode POST

http://localhost:8000/api/v1/banque/create/

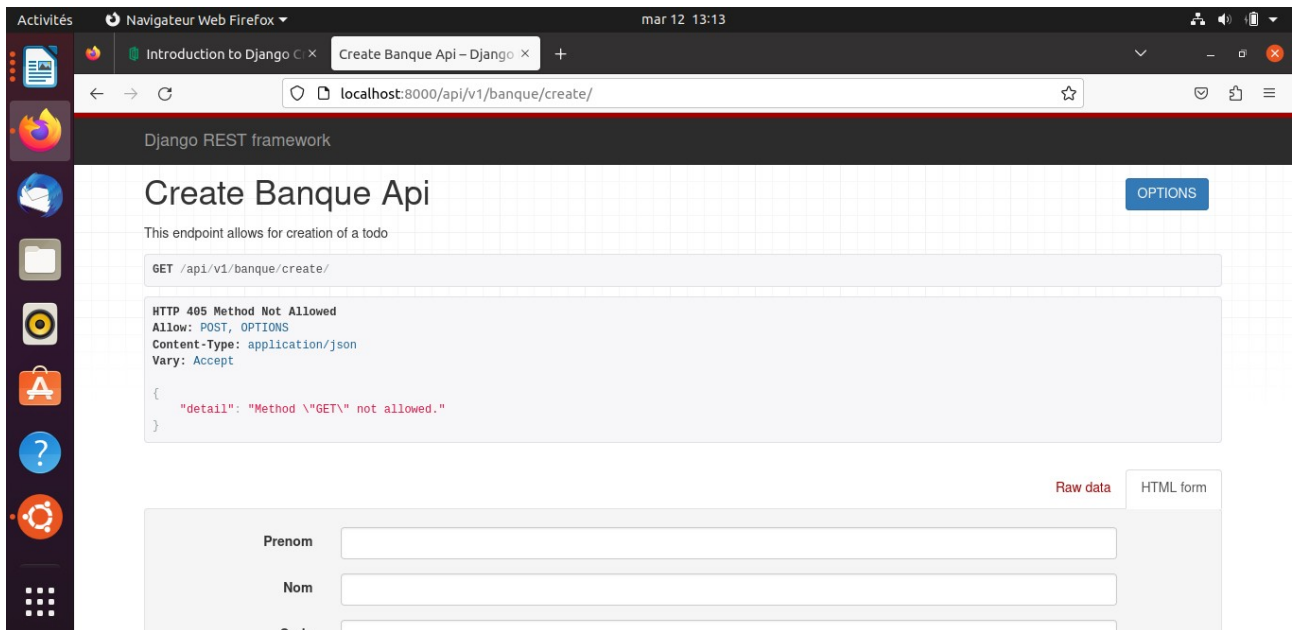


Fig 3 : Interface de création de compte

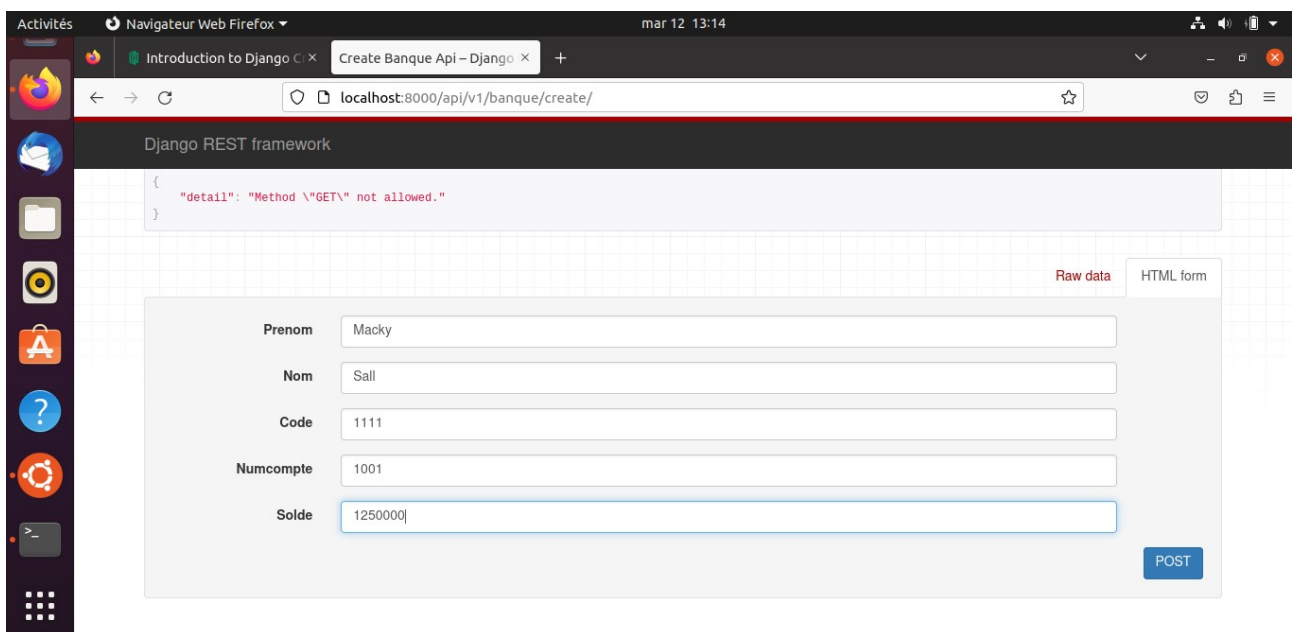


Fig 4 : formulaire de création de compte pour presi Sall

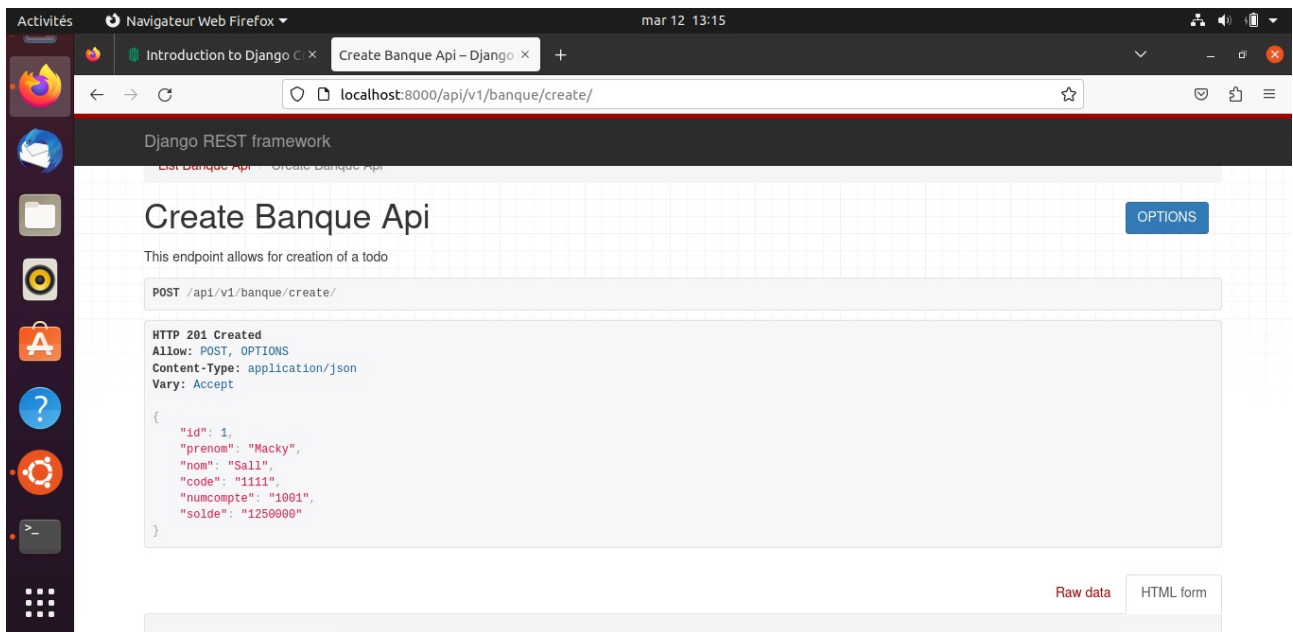


Fig 5 : résultat de la création

Pour lister les comptes, on utilise la methode GET et url `http://localhost:8000/api/v1/banque/`

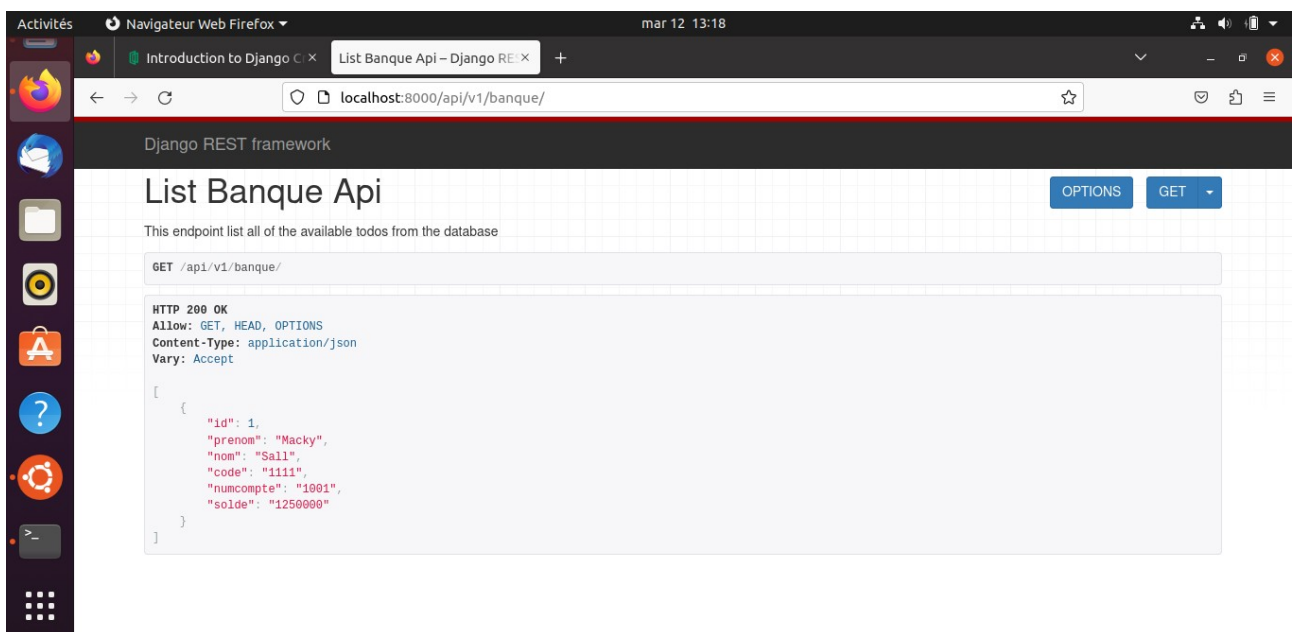


Fig 6 : liste des comptes des presidents

Ajout du président Touadera



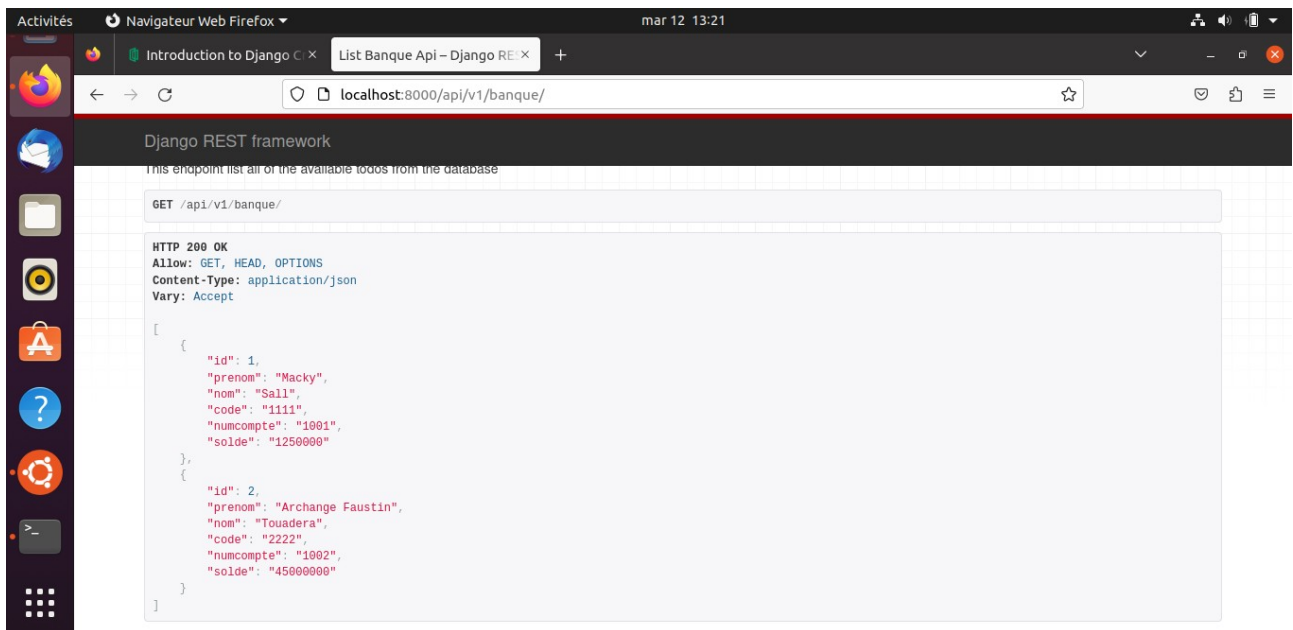


Fig 7 : Résultat de l'ajout du président Touadera

Ajout de Idrissa Seck

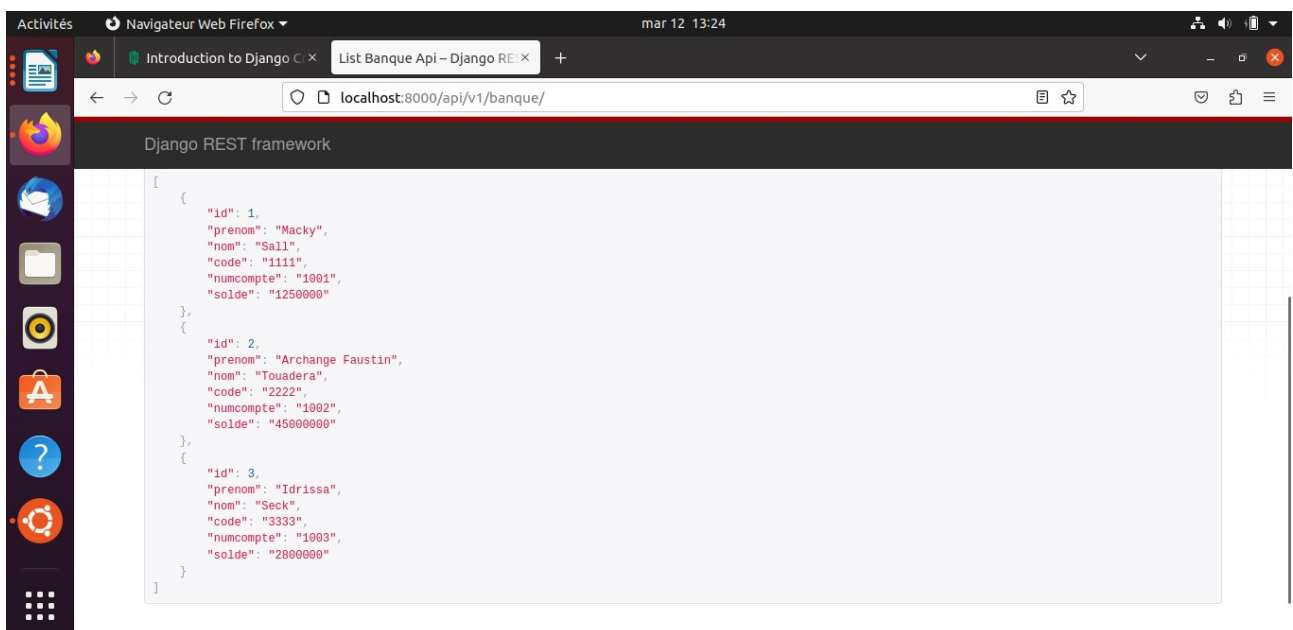


Fig 8: Resultat de l'ajout de Idrissa Seck

Testons la mise à jour via l'api pour le compte d'id 1

On utilise le verbe PUT avec url <http://localhost:8000/api/v1/banque/update/1/>

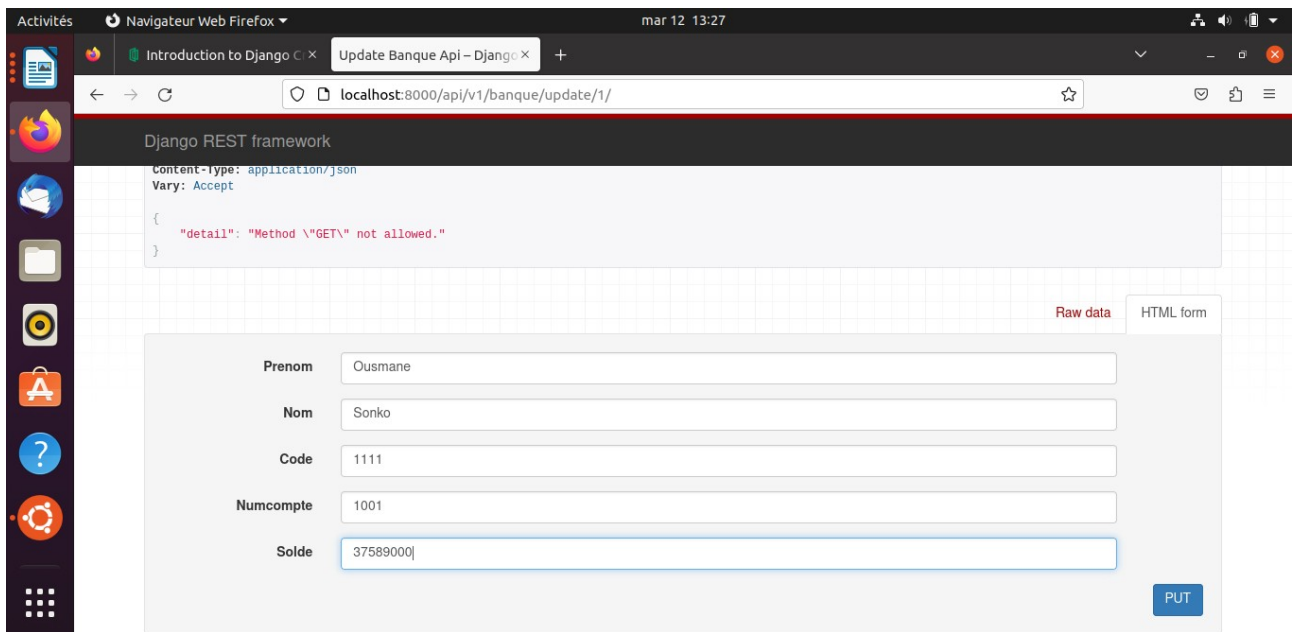


Fig 9 : Mise à jour des infos du compte 1

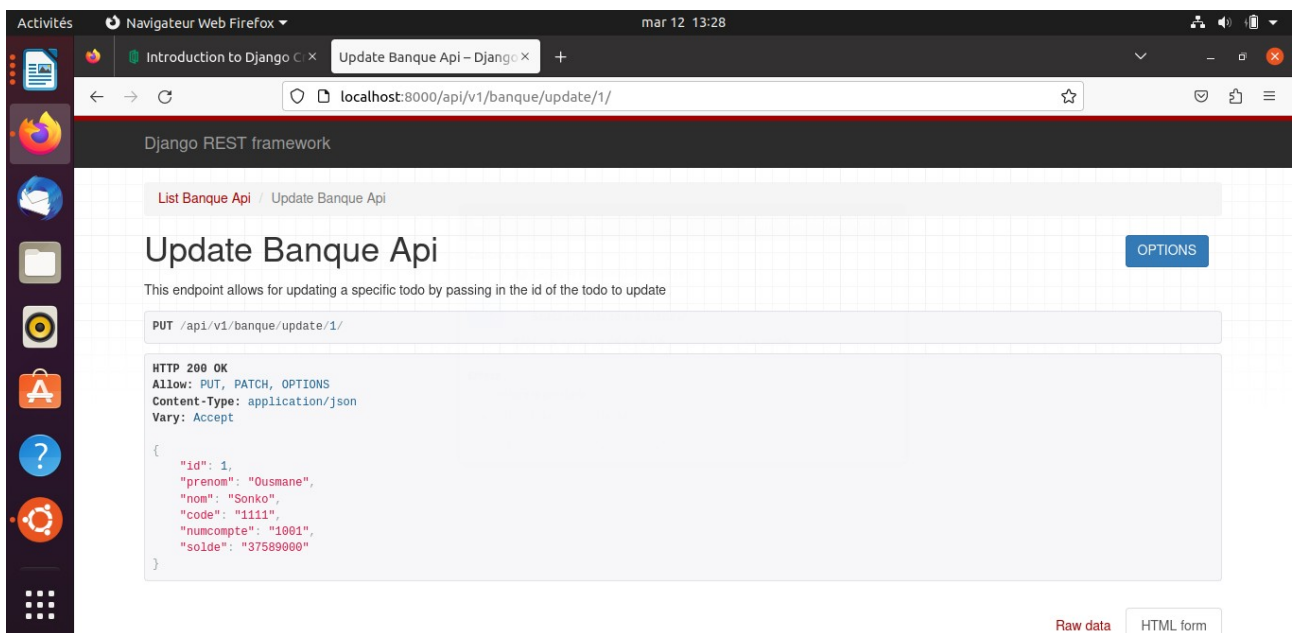


Fig 10 : Résultat de la mise à jour du compte 1

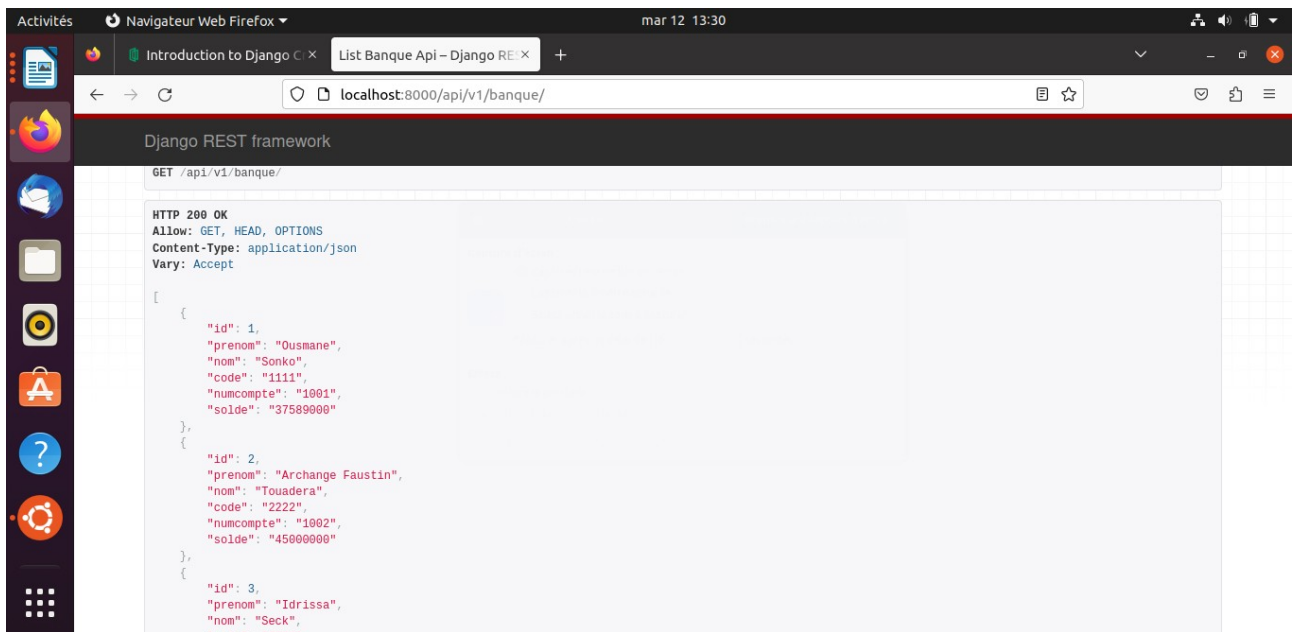


Fig 11 : Resultat de la mise à jour

## Supprimons le compte 3

alain@im:~\$ `curl -X DELETE "http://localhost:8000/api/v1/banque/delete/3/"`

Après

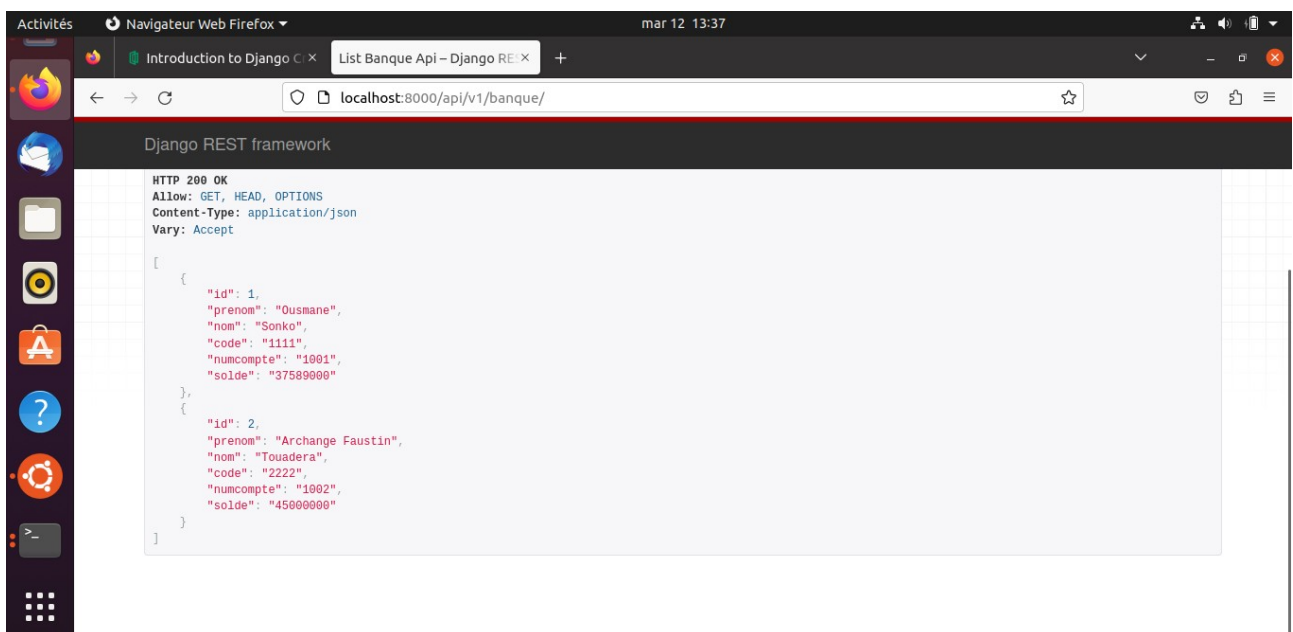


Fig 12: Résultat de suppression

### 13- creation du compte admin pour la gestion du backoffice

(venv) root@im:/home/alain/django/president# ./manage.py createsuperuser

Username (leave blank to use 'root'): admin

Email address: admin@rtn.sn

Password: **passer**

Password (again):

This password is too short. It must contain at least 8 characters.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

### On teste

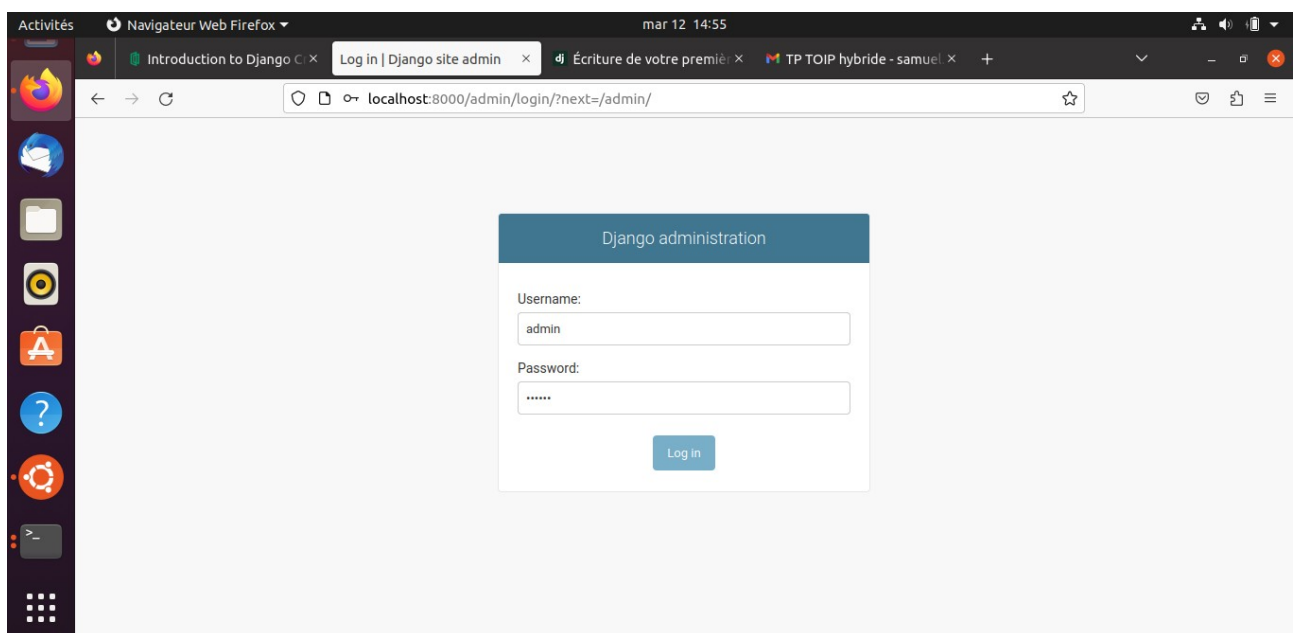


Fig 13: Page d'authentification

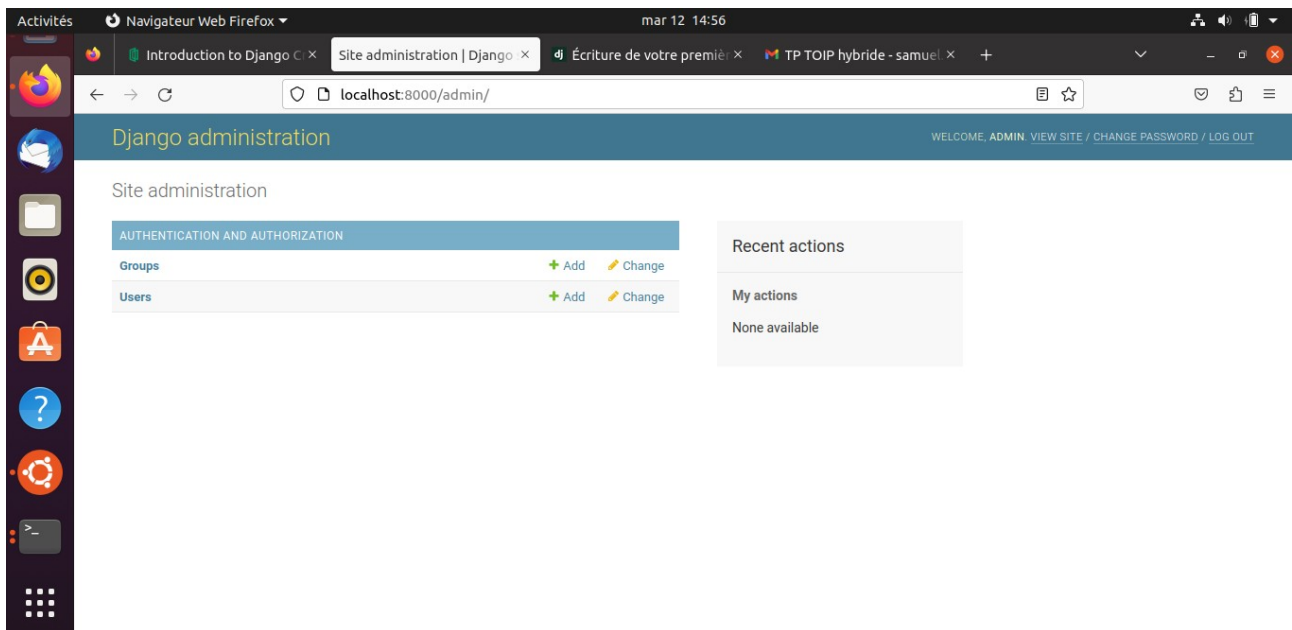


fig 14 : premiere connexion sur l'interface admin de django

15-

root@im:/home/alain/django/president/banque# **nano admin.py**

```
# Register your models here.
```

```
from django.contrib import admin
```

```
from .models import Banque
```

```
admin.site.register(Banque)
```

On redemarre django

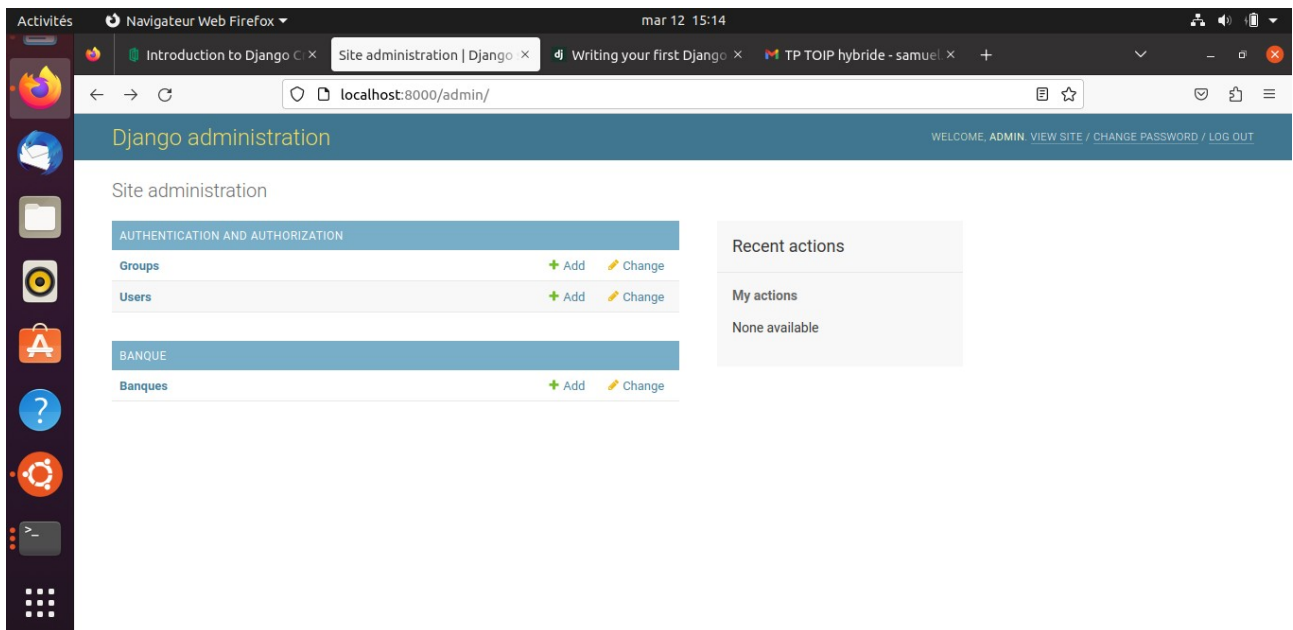


Fig 15 : Apparition de l'application banque dans l'interface admin de django

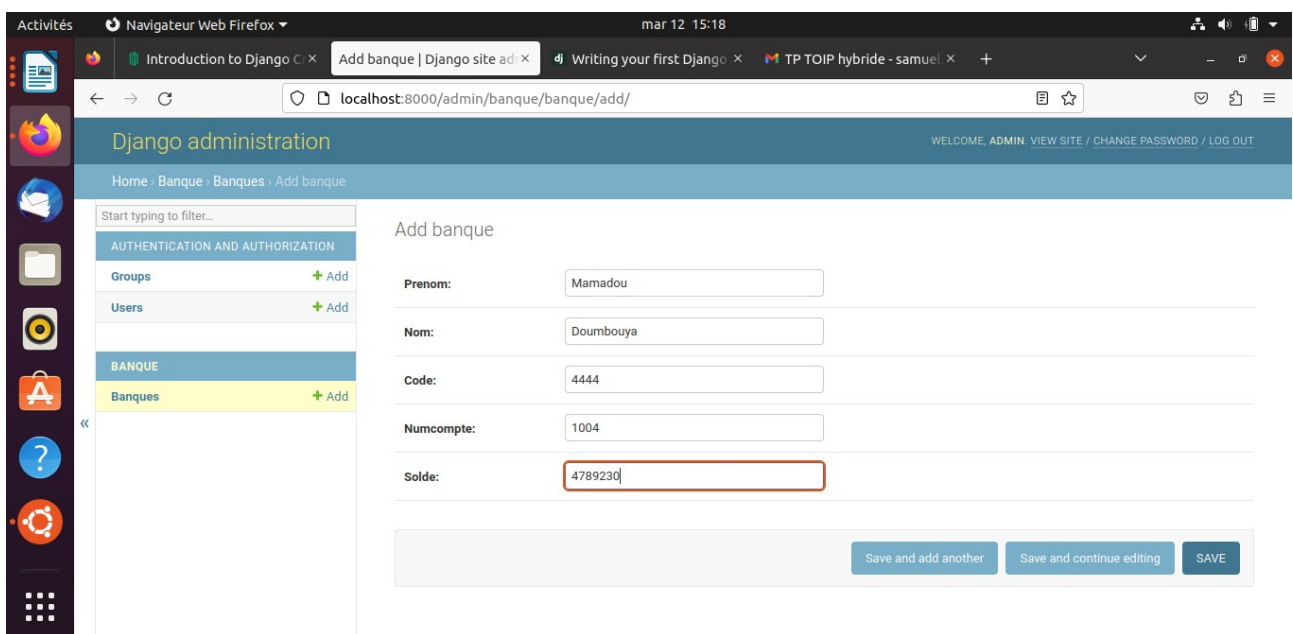


Fig 16 : Ajout de compte via l'interface d'administration de Django

On peut voir l'historique des actions aites par l'administrateur

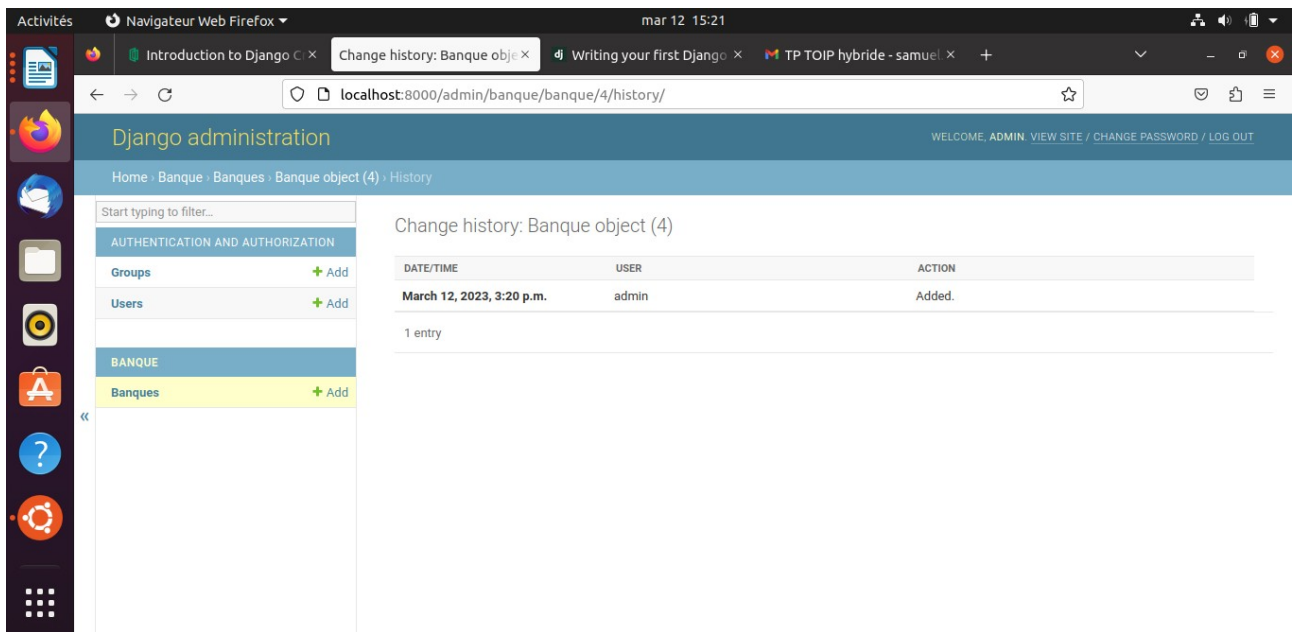


fig 17 : Historique des actions faites par l'administrateur

En allant sur l'interface de consommation de l'API, on constate l'ajout de Doumboua

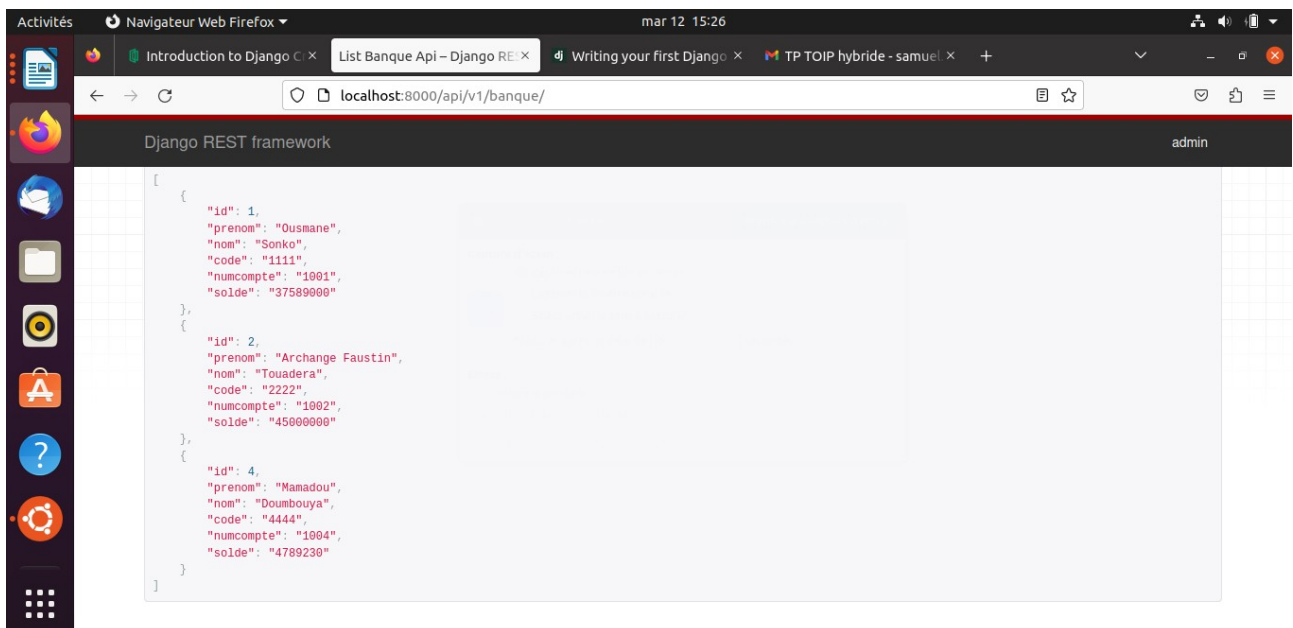


Fig 18 : Résultat d'ajout du président Doumbouya via l'interface d'admin de Django

Documentation de l'API

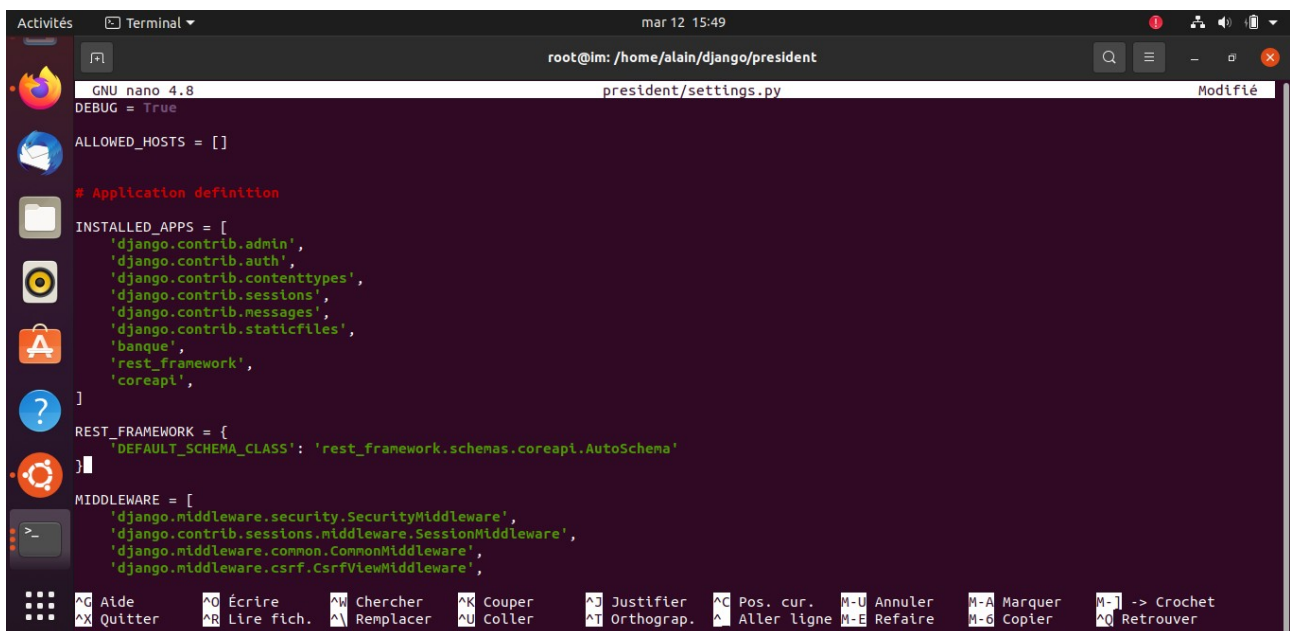
on installe coreapi

(venv) root@im:/home/alain/django/president# pip3 install coreapi

```
(venv) root@im:/home/alain/django/president# nano president/settings.py
```

```
# Application definition
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'banque',  
    'rest_framework',  
    'coreapi',  
]  
  
REST_FRAMEWORK = {  
    'DEFAULT_SCHEMA_CLASS': 'rest_framework.schemas.coreapi.AutoSchema'  
}
```



```
Activités Terminal mar 12 15:49  
root@im: /home/alain/django/president  
GNU nano 4.8 president/settings.py Modifié  
DEBUG = True  
ALLOWED_HOSTS = []  
  
# Application definition  
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'banque',  
    'rest_framework',  
    'coreapi',  
]  
  
REST_FRAMEWORK = {  
    'DEFAULT_SCHEMA_CLASS': 'rest_framework.schemas.coreapi.AutoSchema'  
}  
  
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
]  
  
^G Aide ^O Écrire ^W Chercher ^K Couper ^J Justifier ^C Pos. cur. M-U Annuler M-A Marquer M-] -> Crochet  
^X Quitter ^R Lire fich. ^_ Remplacer ^U Coller ^T Orthograp. ^_ Aller ligne M-E Refaire M-6 Copier M-Q Retrouver
```

Fig 19 : Configuration de la documentation de l'API



```
(venv) root@im:/home/alain/django/president# nano president/urls.py
```

```
from django.contrib import admin
from django.urls import path
from django.urls import include
from rest_framework.documentation import include_docs_urls

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/banque/', include("banque.urls")),
    path('docs/', include_docs_urls(title='Banque Api')),
]
```

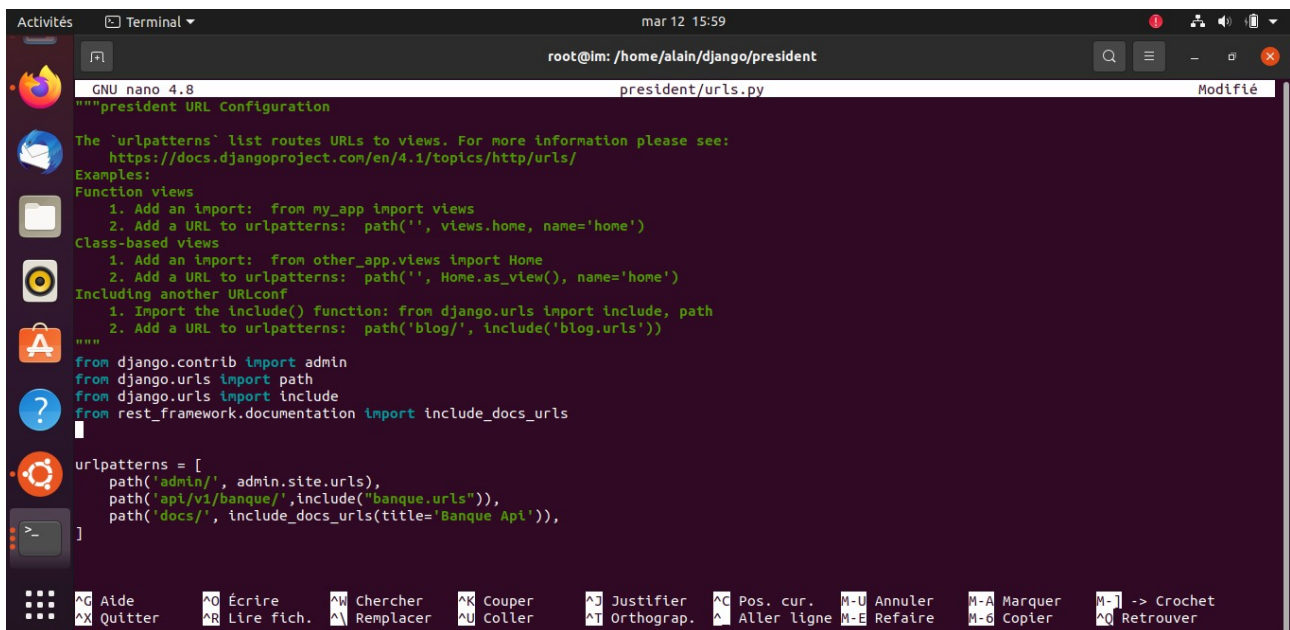


Fig 20 : Paramétrage de la documentation de notre API

On redemarre Django et on teste la documentation

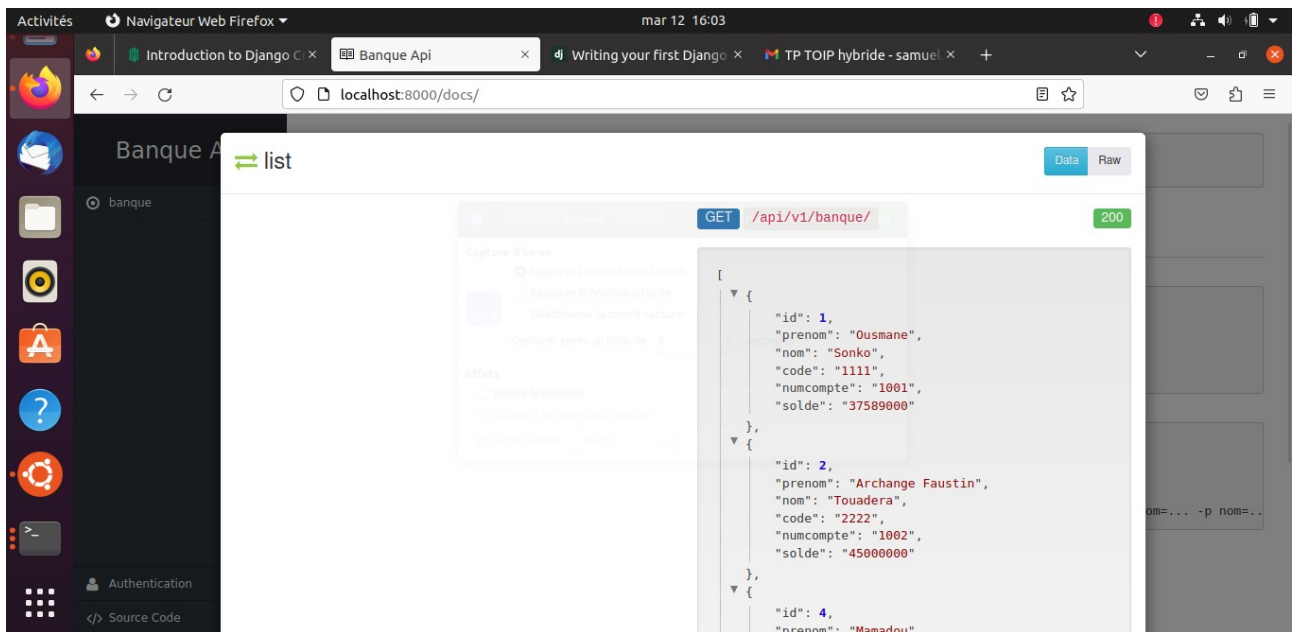


Fig 21 : liste des comptes via docs

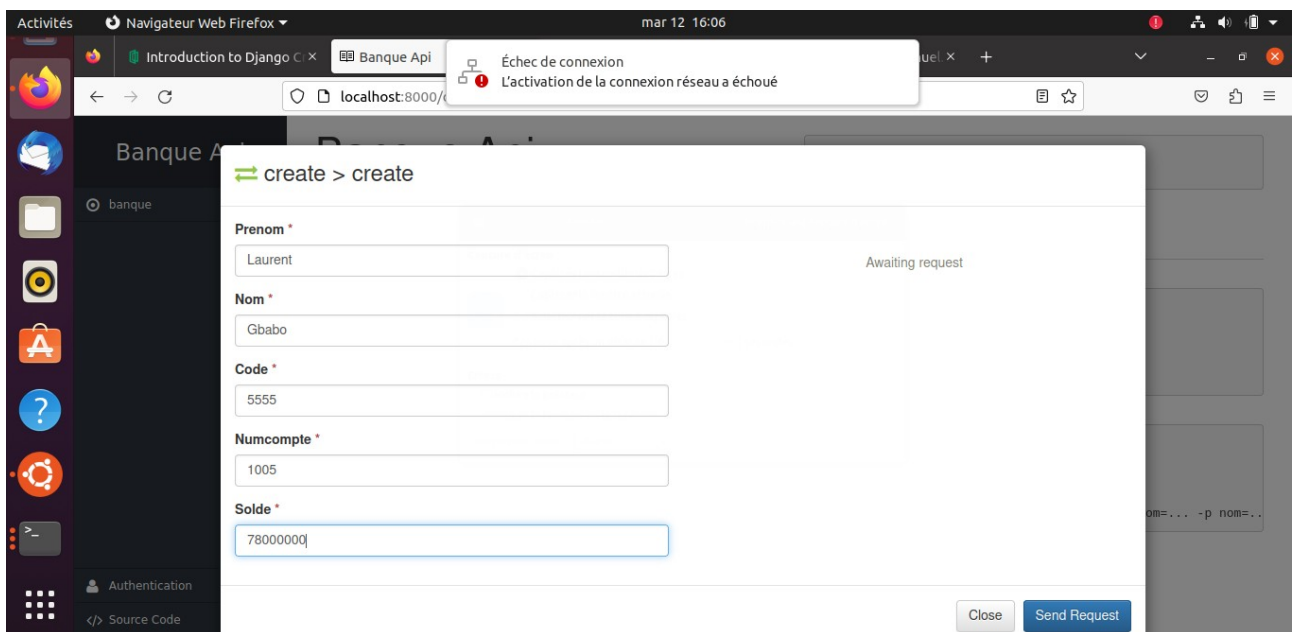


Fig 22 : Ajout de Gbago via docs de Django

**Banque Api**

banque

**list**

GET /api/v1/banque/

This endpoint list all of the available todos from the database

**create > create**

POST /api/v1/banque/create/

This endpoint allows for creation of a todo

**Request Body**

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
prenom	required

```
# Install the command line client
$ pip install coreapi-cli

# Load the schema document
$ coreapi get http://localhost:8000/docs/

# Interact with the API endpoint
$ coreapi action banque list

# Load the schema document
$ coreapi get http://localhost:8000/docs/

# Interact with the API endpoint
$ coreapi action banque create create -p prenom=... -p nom=...
```

Fig 23 : docs 1

**Banque Api**

banque

containing the following items.

Parameter	Description
prenom	required
nom	required
code	required
numcompte	required
solde	required

**delete > delete**

DELETE /api/v1/banque/delete/{id}/

This endpoint allows for deletion of a specific Todo from the database

**Path Parameters**

The following parameters should be included in the URL path.

Parameter	Description
id	required

A unique integer value identifying this banque.

```
# Load the schema document
$ coreapi get http://localhost:8000/docs/

# Interact with the API endpoint
$ coreapi action banque delete delete -p id=...
```

Fig 24 : docs2

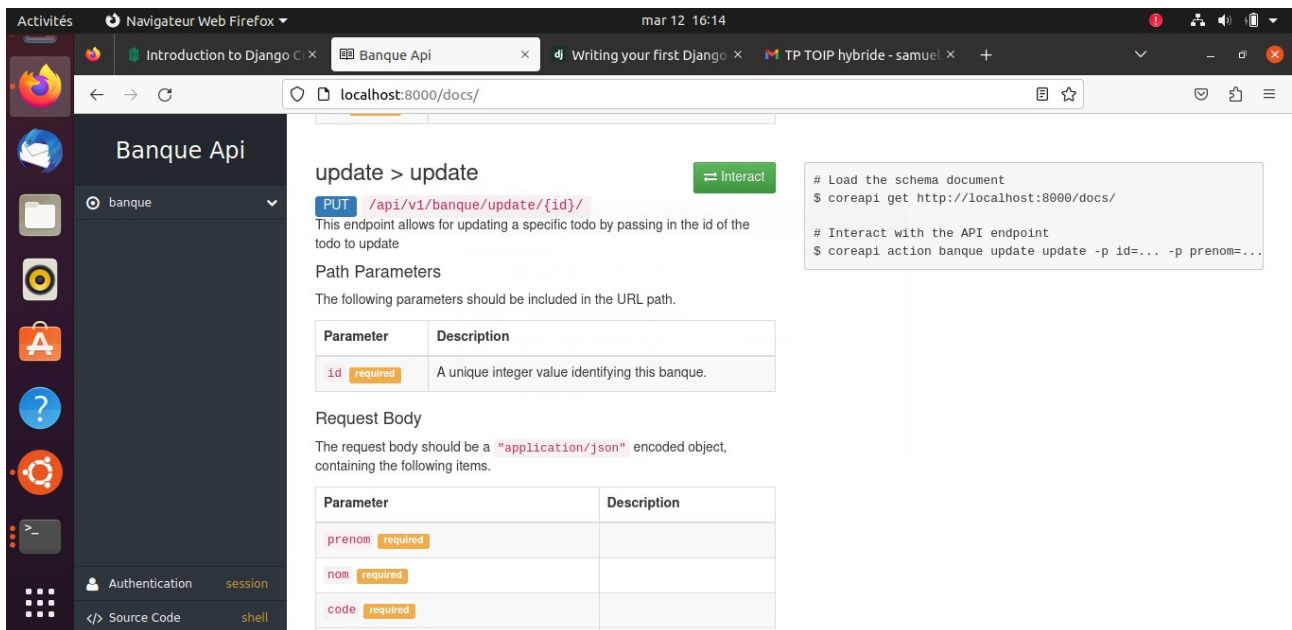


Fig 25 : Docs

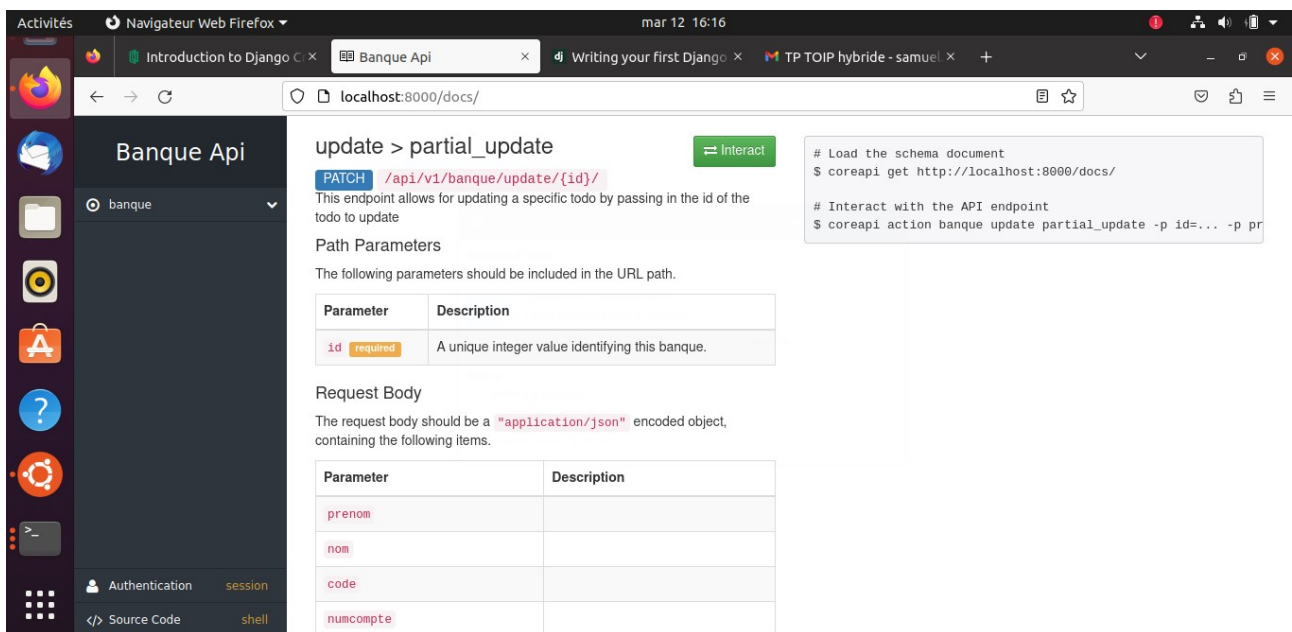


Fig 26 : docs4

## Conclusion

Il est facile de mettre en place un API REST avec Django

La démarche de mise en place d'une API Rest avec Django est claire:

Il faut aller jusqu'à la **documentation pour aider les développeurs frontend** à créer des interfaces d'utilisation de l'Api en passant par l'**interface d'administration** sans négliger la sécurité d'accès de préférence par les jetons JWT.

Dans notre cas, nous avons utilisé la base sqllitedb

On vous demande de paramétrer django ORM pour que l'Api utilise Mysql

Pour respecter la norme, vous devriez aussi la gestion de la sécurité par l'utilisation de JWT