Servidor Django Completo

Explicando el código

- El ejercicio ha sido realizado como una continuación del servidor básico de Django, por lo que la creación de la app y todo el resto de las acciones ya están realizadas, así que explicaremos únicamente la realización de la gestión de usuarios, el testeo, el viewset y la migración de sqlite a MySQL.

```
from rest_framework.viewsets import ModelViewSet
from .models import Pokemon
```

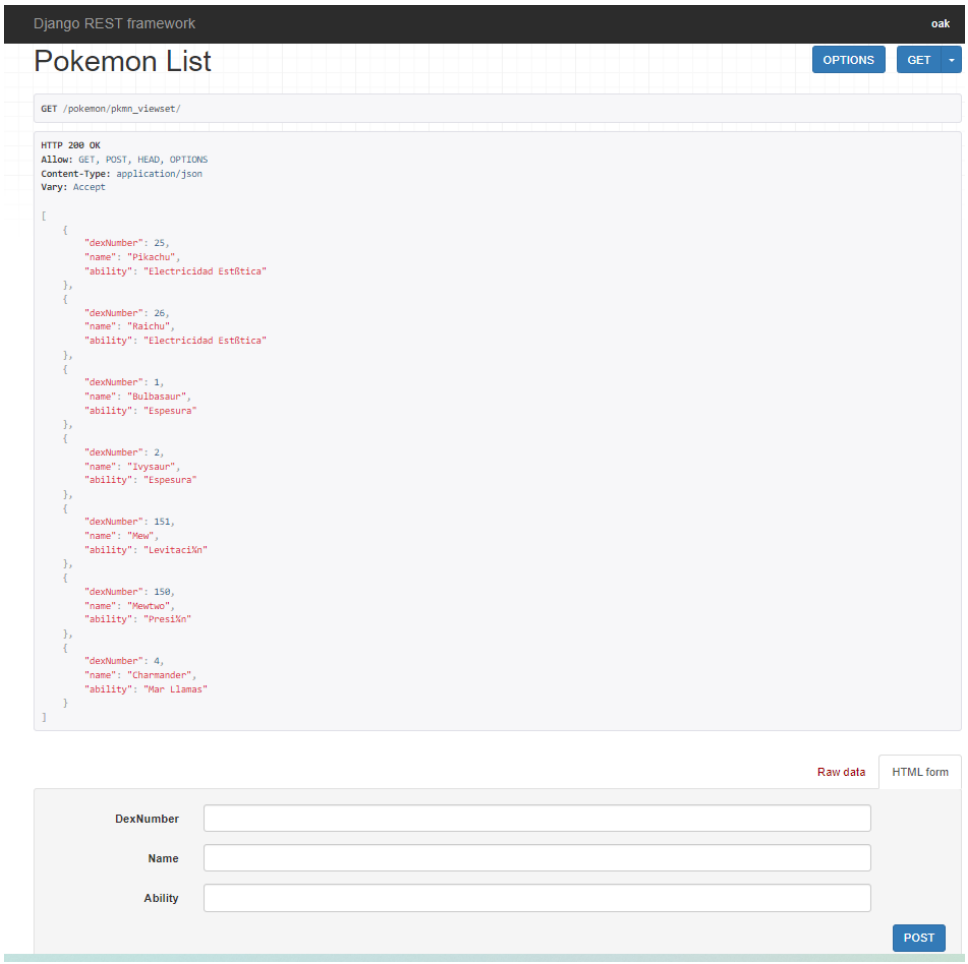
```
class PokemonViewSet(ModelViewSet):
    queryset = Pokemon.objects.all()
    serializer_class = PokemonSerializer
    lookup_field = 'dexNumber'
    authentication_classes = [authentication.SessionAuthentication]
    permission_classes = [permissions.DjangoModelPermissions]
```

- Lo primero que hemos hecho ha sido crear el PokemonViewSet en el archivo views.py de la carpeta pokemon. Le hemos añadido el campo lookup_field para poder buscar pokemons mediante su número de pokedex, es decir si ponemos '/url_del_viewset/1' nos aparecerá el pokemon con dexNumber = 1.
- El authentication_classes y permission_classes los hemos establecido para que solo puedan acceder al viewset los usuarios logados, los cuales explicaremos más adelante en la gestión de usuarios.

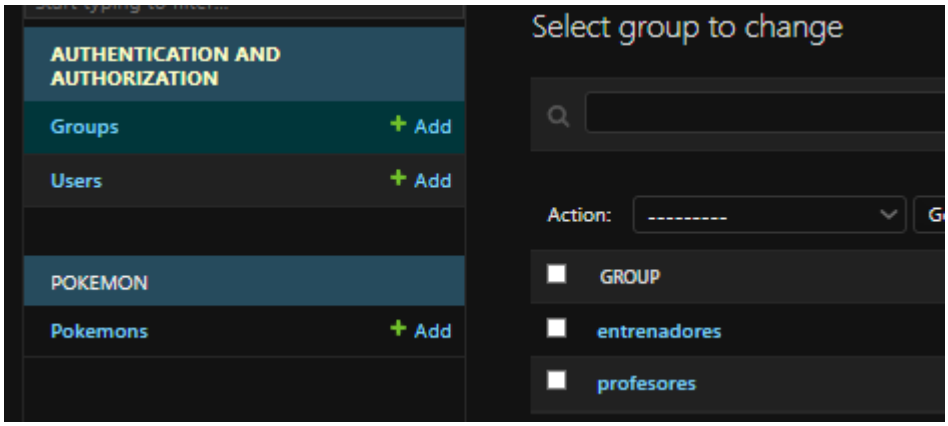
```
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register('pkmn_viewset', views.PokemonViewSet)

urlpatterns = [
    path('<int:pk>', views.PokemonRetrieveAPIView.as_view()),
    path('create', views.PokemonListCreateAPIView.as_view()),
    path('<int:pk>/delete', views.PokemonDestroyAPIView.as_view()),
    path('<int:pk>/update', views.PokemonUpdateAPIView.as_view())
] + router.urls
```

- Después de haber creado el viewset, establecemos la ruta que tendrá para poder acceder a él, y como ya teníamos establecidas unas cuantas rutas dentro de las views de pokemon, hemos establecido una nueva.
- Para crearlo nos tenemos que traer DefaultRouter de rest_framework.router y lo declaramos como router y con router.register establecemos la ruta que queremos darle, en este caso, 'pkmn_viewset' y la añadimos en los urlpatterns.

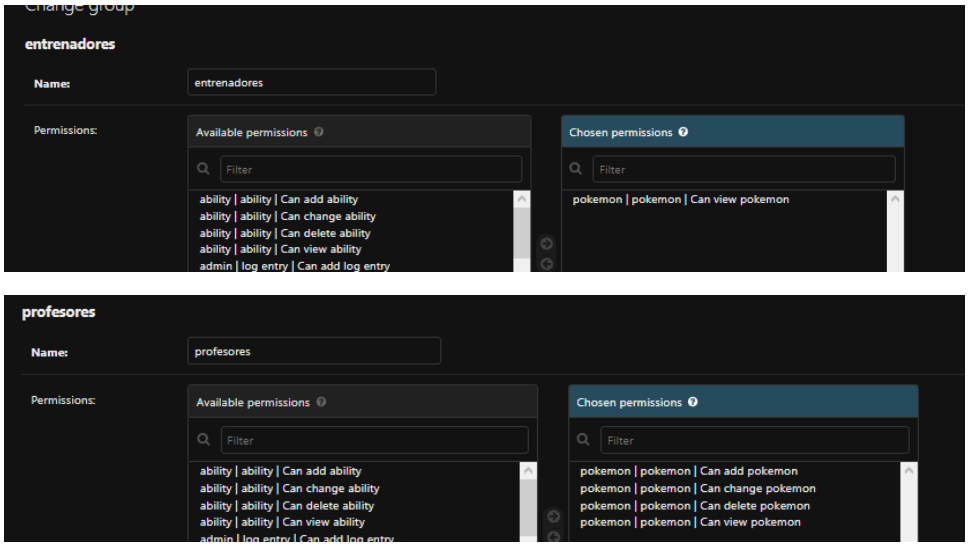
- Como las rutas del archivo views.py de pokemon ya estaban activadas en el archivo views.py de la carpeta principal del proyecto, no tenemos que modificar nada ahí, pero es interesante que sepamos que para acceder al viewset tendremos que seguir la ruta 'localhost:8000/pokemon/pkmn_viewset'.
- Ahora procedemos a la gestión de usuarios. Lo primero que hemos hecho ha sido establecer que al viewset solo podemos acceder si estamos logados y para ello hemos creado un superuser llamado 'oak' con su respectiva contraseña.



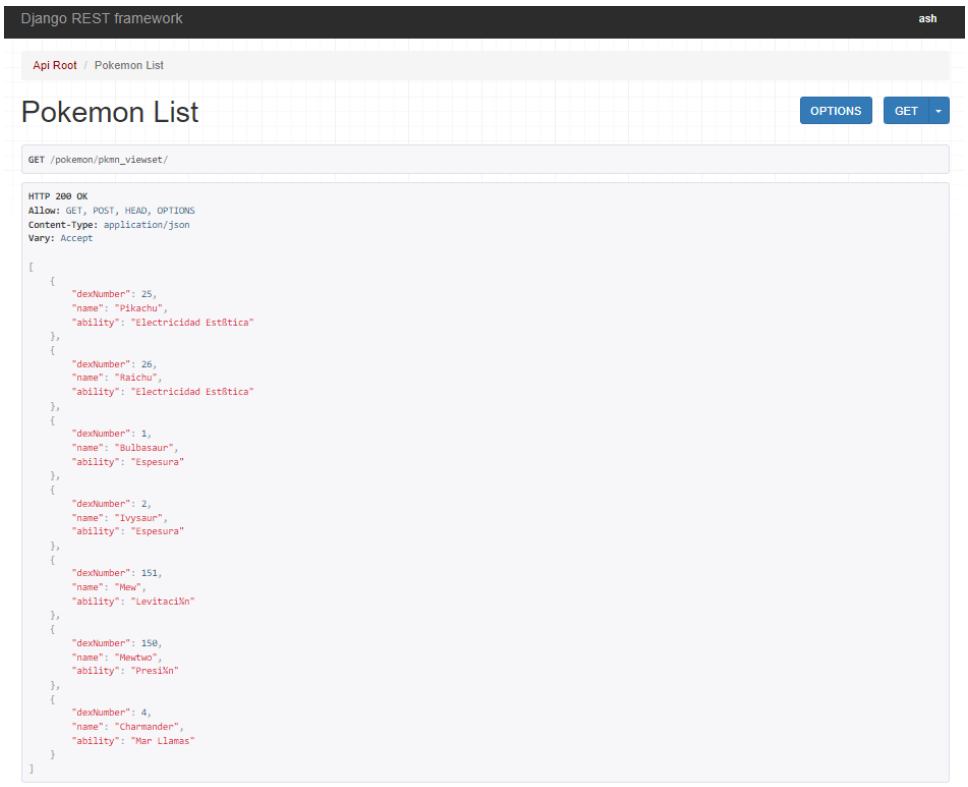
- Con el usuario logado podemos comprobar que accedemos al viewset y podemos realizar las acciones, pero ahora queremos crear grupos de usuarios y restringir su acceso a ciertas acciones dependiendo del grupo.



- Aquí podemos ver como en este caso hemos creado el grupo ‘profesores’ y el grupo ‘entrenadores’.



- Podemos observar cómo se ha establecido que los únicos capaces de hacer cambios sobre los datos de los pokémon, es decir, crearlos, borrarlos o cambiarlos son los profesores, y los entrenadores solo pueden consultarlos.



- Antes hemos podido ver como ‘oak’, que es un profesor podía realizar todas las acciones y aquí vemos como el usuario ‘ash’ que es un entrenador solo puede consultar los datos de los pokemon.

```python
from django.contrib import admin
from .models import Pokemon

admin.site.register(Pokemon)
```

- También dentro del archivo admin.py de la carpeta pokemon hemos registrado a Pokemon para que pueda verse dentro del sitio admin de la app para poder realizar acciones con él y según seamos entrenador o profesor, podemos gestionar los datos en ese apartado también.
- Ahora debemos realizar la migración de sqlite a MySQL, por lo que en la terminal ejecutamos en la ruta del proyecto el comando 'mysql -u root' (si tiene contraseña, como el mio, le añadimos -p al final para poder decir la contraseña del usuario root).
- Después de esto, tenemos que crear la base de datos de nuestros pokemon asi que ejecutamos 'CREATE DATABASE pokedex CHARACTER SET UTF8;', 'GRANT ALL PRIVILEGES ON productos.* TO root@localhost;' y por último 'FLUSH PRIVILEGES;'. Con esto ya está la BBDD creada y tenemos que realizar la migración.
- Para que podamos hacer la migración tenemos que ejecutar dentro de la carpeta del proyecto el comando 'python manage.py dumpdata --natural-foreign --natural-primary -e contenttypes -e auth.permission --indent 4 > datadump.json', y con esto se nos crea el datadump.json con todos los datos.
- Ahora tenemos que instalar con pip install mysqlclient en el proyecto o bien introducimos en requirements.txt mysqlclient e instalamos todo el contenido con 'pip install –r requirements.txt'.

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'pokedex',
        'USER': 'root',
        'PASSWORD': [redacted]
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
```

- Ahora en el archivo settings.py de la carpeta principal del proyecto tenemos que buscar DATABASES y establecer nuestro engine como mysql, poner la base de datos pokedex y los datos para poder acceder.
- Después de esto para que nuestros datos sean introducidos en la base de datos, tenemos que ejecutar el comando 'python manage.py loaddata datadump.json' para que se carguen nuestros datos ya creados en la base de datos que acabamos de crear.
- Con esto ya creado, pasamos a la realización de tests.

```python
from rest_framework import status
from rest_framework.test import APITestCase
from django.contrib.auth.models import User

class PokemonCreationTestCase(APITestCase):
    def setUp(self):
        self.admin = User.objects.create_superuser(username='admin', password='admin')
        self.user = User.objects.create_user(username='user', password='user')

    def test_pokemon_creation(self):
        self.client.force_authenticate(user=self.admin)
        data = {'dexNumber': '4', 'name': 'Charmander', 'ability': 'Mar Llamas'}
        res = self.client.post(f'/pokemon/create', data)
        self.assertEqual(res.status_code, status.HTTP_201_CREATED)

    def test_pokemon_creation_no_authorized(self):
        self.client.force_authenticate(user=self.user)
        data = {'dexNumber': '4', 'name': 'Charmander', 'ability': 'Mar Llamas'}
        res = self.client.post(f'/pokemon/create', data)
        self.assertEqual(res.status_code, status.HTTP_403_FORBIDDEN)
```

- En nuestro caso, como habiamos establecido que los usuarios admin, que son los profesores, podían crear pokemon y los usuarios normales, los entrenadores, no podian hacerlo, hemos optado por testear la creacion de los pokemon según el tipo de user.
- Primero hemos establecido en el setUp de PokemonCreationTestCase ambos tipos de user, y en test_pokemon_create establecemos la autenticacion del usuario admin y hacemos un post de un pokemon, esperando el codigo 201.
- Lo mismo hemos hecho en test_pokemon_create_no_authorized, pero con el usuario normal, esperando un codigo 403.

```python
        data = { 'dexNumber': '4', 'name': 'Charmander', 'ability': 'Mar Llamas' }
        res = self.client.post(f'/pokemon/create', data)
        self.assertEqual(res.status_code, status.HTTP_403_FORBIDDEN)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

[21/Jun/2023 02:48:39] "GET /static/admin/img/icon-viewlink.svg HTTP/1.1" 200 581
[21/Jun/2023 02:48:45] "GET /admin/ HTTP/1.1" 200 5060
[21/Jun/2023 02:48:48] "GET /pokemon/ HTTP/1.1" 200 5376
[21/Jun/2023 02:48:53] "GET /pokemon/pkmn_viewset/ HTTP/1.1" 200 6430
(env) PS C:\Users\Alvaro\Desktop\backend\python\servidor_basico_django> python manage.py test
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.690s

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

- Podemos comprobar que los test han terminado correctamente y que han sido bien realizados.

Conclusiones y observaciones

Después de haber tenido la base del servidor básico ya realizada, me ha resultado bastante fluido la realización del resto de cosas que debíamos añadir, además de lo práctico que me ha resultado el viewset.

Me hubiera gustado realizar los test sobre las url del viewset, ya que era el aspecto principal de este tema, pero después de haber realizado varios intentos no he sabido sacarlo. Al menos si lo he podido realizar sobre las vistas genéricas que ya estaban creadas de antes.

Este ejercicio me ha ayudado a poder ver una unión más clara de las bases de datos con los servidores Django y creo que me va a ayudar bastante con los proyectos que tenemos entre manos para los próximos meses.