Autenticación y autorización

Autenticación y autorización

- Django REST Framework nos proporciona una serie de utilidades para trabajar con la autenticación y autorización (permisos)

- Autenticación: nos permite saber quién está usando el API

- Autorización: nos permite definir si el usuario que está usando el API, está autorizado a hacer la acción que solicita o no

Autenticación

- Con Django, sabemos qué usuario está autenticado gracias al atributo **user** del objeto request que recibe cada vista.

Autorización (permisos)

- En nuestros controladores podemos añadir una clase para gestionar la autorización (o permisos) de cada operación.

- Podemos utilizar clases predefinidas o bien crear nuestra propia clase para gestionar la autorización.

Autorizaciones genéricas

- **AllowAny:** permite todo a todos
- **IsAuthenticated:** permite todo a los usuarios autenticados
- **IsAdminUser:** sólo se permite hacer cosas a usuarios admin
- **IsAuthenticatedOrReadOnly:** permite operaciones de lectura a todos y de escritura/actualización/borrado a usuarios autenticados
- **DjangoModelPermissions:** utiliza los modelos de permisos de Django
- **DjangoModelPermissionsOrAnonReadOnly:** utiliza los modelos de permisos de Django para operaciones de escritura/actualización/borrado con usuarios autenticados y permite operaciones de lectura a cualquiera
- **DjangoObjectPermissions:** utiliza el framework de django de objects permissions.
- **TokenHasReadWriteScope:** para utilizar con autenticación OAuth u OAuth2

Autorizaciones personalizadas

- Debemos crear una clase que hereda de
  - rest_framework.permissions.BasePermission
- Implementar los métodos has_permission y has_object_permission que deben devolver True o False
- El orden de ejecución es:
  1. has_permission(self, request, view)
  2. has_object_permission(self, request, view, obj)

Autorizaciones personalizadas

```
from rest_framework import permissions

class PlanetPermissions(permissions.BasePermission):

    def has_object_permission(self, request, view, obj):
        # se ejecuta sólo en peticiones de DETALLE

        return request.user.is_superuser

def has_permission(self, request, view):

    return request.user.is_authenticated()
```

Autorizaciones personalizadas

IMPORTANTE

- Si nuestra clase es APIView o GenericAPIView, la autorización a nivel de objeto deberemos ejecutarla a mano con: **self.check_object_permissions(request, <objeto>)**

- En las clases con queryset, funciona automáticamente