AUTENTICACIÓN BÁSICA

Hasta el momento hemos visto que conociendo las rutas adecuadas podemos acceder a la lista de tareas, acceder a una en concreto, borrar las tareas o modificarlas. El problema es que cualquiera que tenga acceso a dichas rutas podría hacer estas mismas operaciones.

DRF proporciona una serie de herramientas de autenticación para controlar quién puede hacer y el qué en cada momento. Hay dos tipos de autenticación, la básica y la autenticación a través de token.

La autenticación se configura dentro del fichero DRF→ settings.py

Los manuales de referencia para conocer todo lo referente a autenticación y permisos en Django son:

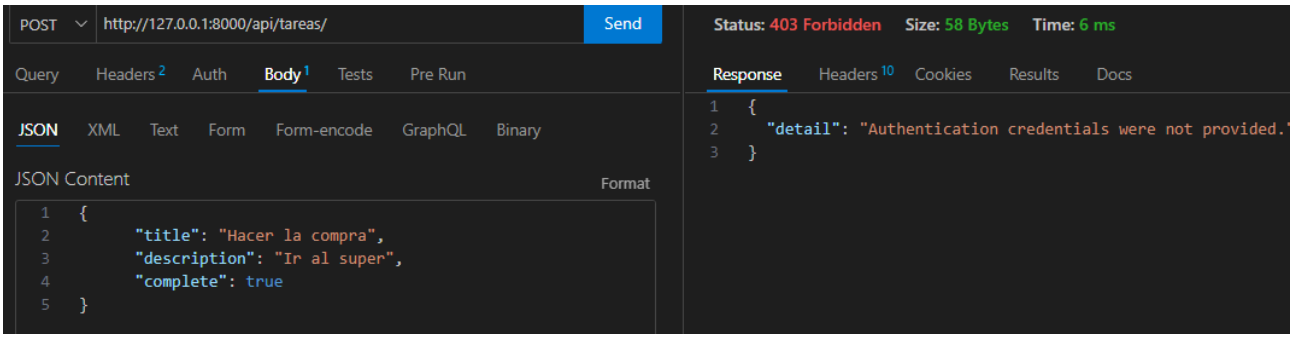
Authentication - Django REST framework (django-rest-framework.org)
Permissions - Django REST framework (django-rest-framework.org)

La documentación es bastante extensa aunque en esta parte de la práctica sólo configuraremos una autenticación básica y te indicaré cómo directamente:

Usaremos inicialmente a modo de demostración un permiso básico que concede a cualquier usuario la posibilidad sólo de hacer consultas con el método GET. No se autorizará ninguna consulta que modifique los datos. Añadimos el código de la imagen en el fichero settings.py de nuestro proyecto.

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',
        #'rest_framework.permissions.IsAuthenticated',
    ]
}
```

Y comprobamos a través del cliente Thunder que podemos hacer peticiones get usando la ruta de la vista que habíamos creado: api/tareas, pero que no podemos hacer peticiones POST ni ninguna otra que suponga modificación alguna:

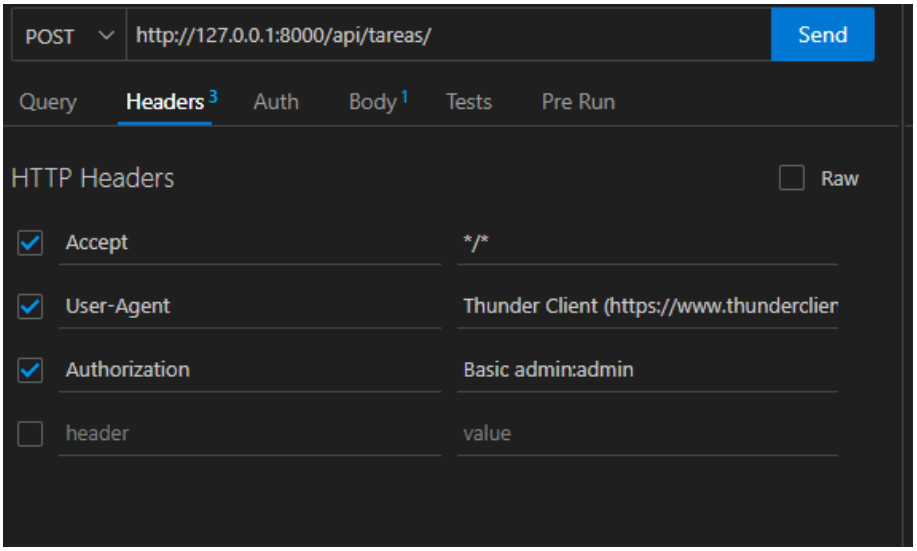


¿Cómo enviamos la información de autenticación?

AUTENTICACIÓN BÁSICA: Debemos enviar la información de usuario (usuario y contraseña) a través de las cabeceras de los archivos que se envían. Para hacer la prueba usamos el cliente Thunder con el usuario admin y la contraseña que hayas añadido cuando se configuró Django.

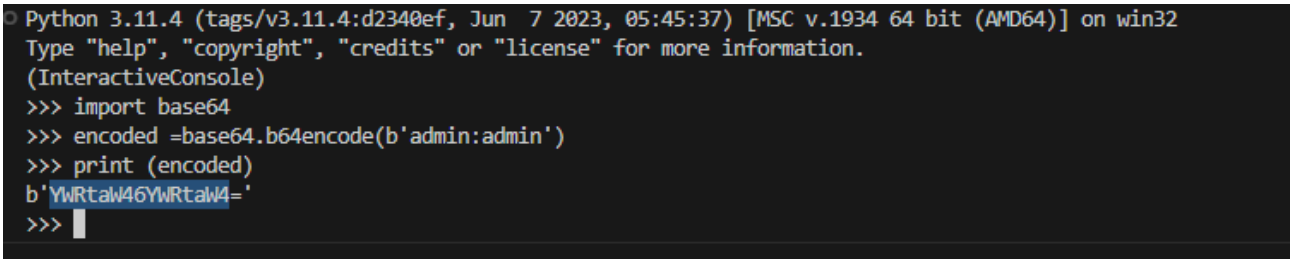
Modo 1: A través de la pestaña Headers

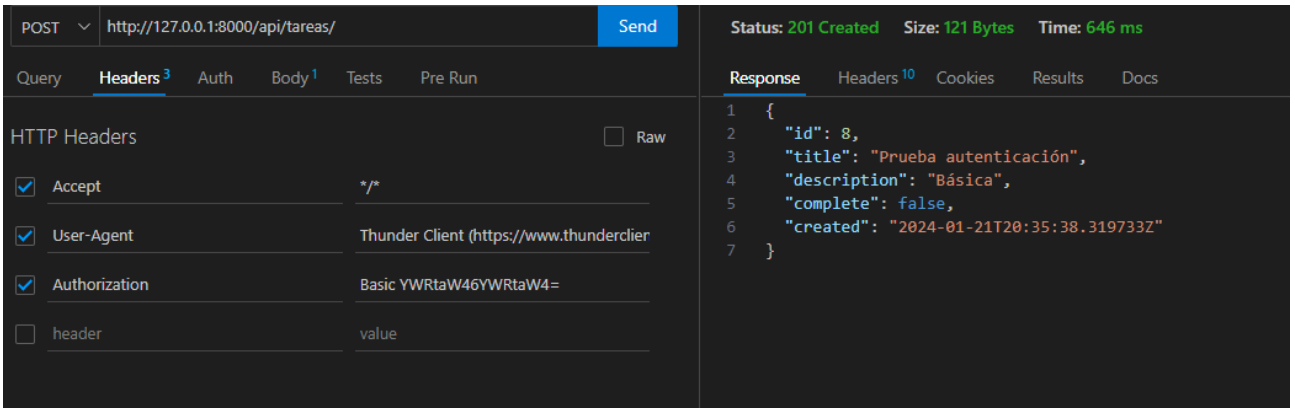
Al hacer la petición, como decíamos, debemos hacer uso de las cabeceras como se muestra en la imagen. Fíjate que debes añadir un nuevo elemento en dicha cabecera que será **"Authorization"** y con la información **Basic usuario:contraseña** (en mi caso admin:admin). La cuestión es que esa información (admin:admin) no se puede enviar así, como texto, sino codificado en base 64. Para hacer esto fíjate en la segunda imagen.



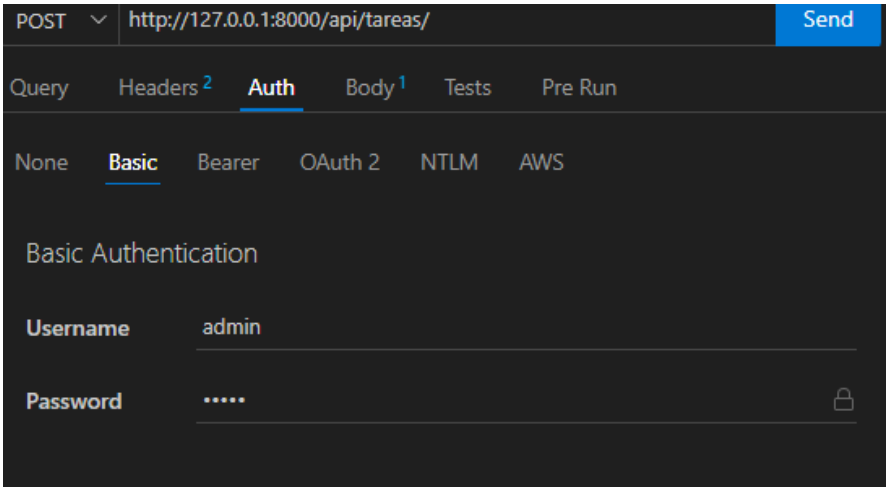
- Apaga el servidor y entra en la shell: *(tu_entorno) … : python manage.py shell*
- Importa el módulo base64

- codifica como se muestra en la imagen tu usuario y contraseña.
- Imprime la variable que contiene ya la codificación.
- Copia toda parte que se muestra sombreada en la imagen incluyendo el signo = (en la imagen no está sombreado)
- Copia el código como se muestra en la siguiente imagen.





Modo 2: Usando la pestaña Auth en combinación con Basic



Supongamos que incluso para las peticiones GET queremos que el cliente esté autenticado:

El fichero settings.py quedaría modificado de la siguiente manera:

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        #'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',
        'rest_framework.permissions.IsAuthenticated',
    ],
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.BasicAuthentication',
    ]
}
```

Si ahora intentas hacer una petición GET sin enviar al servidor los parámetros de autenticación obtendrás un mensaje 401 (Unauthorized) del servidor.

AUTENTICACIÓN PERSONALIZADA (SÓLO EN ALGUNAS VISTAS)

Modificamos el fichero views.py añadiendo, por un lado, los import correspondientes a la autenticación que vamos a añadir en las vistas:

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.authentication import SessionAuthentication, BasicAuthentication
from rest_framework.response import Response
```

Y por otro lado añadimos en la vista que queramos configurar lo siguiente (en mi caso la vista TaskViewSet que es la que está funcionando por defecto).

```
class TaskViewSet (viewsets.ModelViewSet):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer

    authentication_classes = [SessionAuthentication, BasicAuthentication]
    permission_classes = [IsAuthenticated]

    def get (self, request, format = None):
        content = {
            'user': str (request.user),
            'auth': str (request.auth),
        }
        return Response(content)
```

En este momento, aunque en el fichero setting estableciéramos como al principio que las consultas GET si se pueden ejecutar sin autenticación; estamos explicitando en la vista en el propio método GET que obligatoriamente debe autenticarse para acceder y por tanto el servidor volverá a enviar un error 401.