Un API es un conjunto de funcionalidades o recursos que nos expone un sistema para poder interactuar con él desde otro sistema.App web que realiza peticiones a diferentes APIs de diferentes servicios de streaming de música.

**REST** es un acrónimo que significa *Representational State Transfer*, y se refiere a una arquitectura de software para crear servicios web. Para que todos los componentes de una API Web se comuniquen sin ningún problema, establece algunas reglas. Entre ellas se encuentra que utilicen el mismo tipo de dato para la comunicación de la información. Así, el estándar para devolver la información será por lo general o en formato xml o en formato JSON, este último el más usado. Los servicios RESTful emplean el protocolo HTTP(como GET, POST, PUT, DELETE) para realizar operaciones en los recursos que se encuentran en el servidor. aplicaciones de cliente y servidor deben ser completamente independientes entre sí. La única información que la aplicación de cliente debe conocer es el URI del recurso solicitado; no puede interactuar con la aplicación de servidor de ninguna otra forma. De forma similar, una aplicación de servidor no debe modificar la aplicación de cliente sino para pasarle los datos solicitados a través de HTTP.

Sin estado. Las API REST son API sin estado, lo que significa que cada solicitud debe incluir toda la información necesaria para procesarla. Es decir, las API REST no requieren ninguna sesión del lado del servidor. Las aplicaciones de servidor no pueden almacenar datos relacionados con una solicitud de cliente.

Capacidad de almacenamiento en memoria caché. Siempre que sea posible, los recursos deben poder almacenarse en la memoria caché en el lado del cliente o el servidor.Arquitectura del sistema en capas. En las API REST, las llamadas y las respuestas pasan por diferentes capas. Como regla general, no suponga que las aplicaciones de cliente y servidor se conectan directamente entre sí. Puede que haya varios intermediarios diferentes en el bucle de comunicación.**REST es una buena opción cuando se quiere crear una API para un sistema que será accedido por diferentes tipos de dispositivos o aplicaciones, como móviles, web o de escritorio. Si deseas permitir que tus servicios web sean consumidos con facilidad por otras aplicaciones y que sean escalables.**

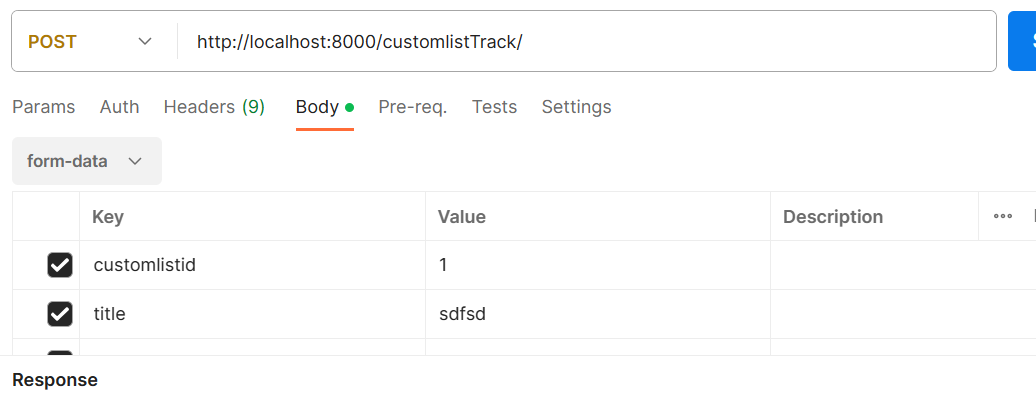
**Django Rest Framework** es una herramienta para la creación de API REST Web escrito en python en el que en servidor se almacena la lógica y en el cliente estaría la interfaz.

El marco REST proporciona varios esquemas de autenticación listos para usar y también le permite implementar esquemas personalizados.

Autenticación, conjunto de mecanismos para asociar la solicitud entrante con un conjunto de credenciales de identificación que puede contener el usuario del cual proviene la petición o el token perteneciente a un usuario, dependiendo los mecanismos que se hayan definido, se valida si la solicitud puede ser permitida o no. Autenticación siempre se ejecuta al comienzo de la vista antes de que se produzcan las comprobaciones de permisos (autorizacion, [La **autenticación** es el proceso de verificar la identidad de un usuario que intenta obtener acceso a un sistema](https://www.bing.com/ck/a?!&&p=d460e064b1cd0b3aJmltdHM9MTY5NDk5NTIwMCZpZ3VpZD0yY2MwOTM3Ny1mN2M3LTY5YmEtMmQ3ZS04MDNlZjNjNzZiNDImaW5zaWQ9NTcxMQ&ptn=3&hsh=3&fclid=2cc09377-f7c7-69ba-2d7e-803ef3c76b42&psq=autenticacion+vs+autorizacion&u=a1aHR0cHM6Ly9zaW1wbHlmYXEuY29tL2RpZmVyZW5jaWEtZW50cmUtYXV0ZW50aWNhY2lvbi15LWF1dG9yaXphY2lvbi8&ntb=1)[**1**](https://www.bing.com/ck/a?!&&p=a13bccb2b9ca16f2JmltdHM9MTY5NDk5NTIwMCZpZ3VpZD0yY2MwOTM3Ny1mN2M3LTY5YmEtMmQ3ZS04MDNlZjNjNzZiNDImaW5zaWQ9NTcxMg&ptn=3&hsh=3&fclid=2cc09377-f7c7-69ba-2d7e-803ef3c76b42&psq=autenticacion+vs+autorizacion&u=a1aHR0cHM6Ly9zaW1wbHlmYXEuY29tL2RpZmVyZW5jaWEtZW50cmUtYXV0ZW50aWNhY2lvbi15LWF1dG9yaXphY2lvbi8&ntb=1)[**2**](https://www.bing.com/ck/a?!&&p=060683dcadc388cfJmltdHM9MTY5NDk5NTIwMCZpZ3VpZD0yY2MwOTM3Ny1mN2M3LTY5YmEtMmQ3ZS04MDNlZjNjNzZiNDImaW5zaWQ9NTcxMw&ptn=3&hsh=3&fclid=2cc09377-f7c7-69ba-2d7e-803ef3c76b42&psq=autenticacion+vs+autorizacion&u=a1aHR0cHM6Ly9kaWdpdGFsaW5ub3ZhdGlvbm5ld3MuZXMvYXV0ZW50aWNhY2lvbi12cy1hdXRvcml6YWNpb24tY3VhbC1lcy1sYS1kaWZlcmVuY2lhLw&ntb=1)[**3**](https://www.bing.com/ck/a?!&&p=695a70320a99bb2fJmltdHM9MTY5NDk5NTIwMCZpZ3VpZD0yY2MwOTM3Ny1mN2M3LTY5YmEtMmQ3ZS04MDNlZjNjNzZiNDImaW5zaWQ9NTcxNA&ptn=3&hsh=3&fclid=2cc09377-f7c7-69ba-2d7e-803ef3c76b42&psq=autenticacion+vs+autorizacion&u=a1aHR0cHM6Ly9hdXRoMC5jb20vZXMvaW50cm8tdG8taWFtL2F1dGhlbnRpY2F0aW9uLXZzLWF1dGhvcml6YXRpb24jIQ&ntb=1). [La **autorización** es un método que se utiliza para determinar los recursos a los que puede acceder un usuario](https://www.bing.com/ck/a?!&&p=4681e3966ebd8120JmltdHM9MTY5NDk5NTIwMCZpZ3VpZD0yY2MwOTM3Ny1mN2M3LTY5YmEtMmQ3ZS04MDNlZjNjNzZiNDImaW5zaWQ9NTcxNQ&ptn=3&hsh=3&fclid=2cc09377-f7c7-69ba-2d7e-803ef3c76b42&psq=autenticacion+vs+autorizacion&u=a1aHR0cHM6Ly9zaW1wbHlmYXEuY29tL2RpZmVyZW5jaWEtZW50cmUtYXV0ZW50aWNhY2lvbi15LWF1dG9yaXphY2lvbi8&ntb=1) AUTENTICADO) y limitaciones y antes de cualquier procesamiento de datos, por ello Django Rest Framework, después de validar la petición, guardará la instancia de la clase User en request.

request.user una instancia del usuario con los datos que le envíe

request.auth token



Hay diferentes métodos de autenticación

1. básica

Se basa principalmente en un nombre de usuario y una contraseña para identificarte.Cualquiera que intercepte la trasmisión de datos puede decodificar fácilmente esta información. Esto se denomina [ataque Man-In-The-Middle (MiTM)](https://es.wikipedia.org/wiki/Ataque_de_intermediario). Se llama asi porque normalmente el cibercriminal o un software malicioso, se incrusta entre la víctima y la fuente de datos (cuentas bancaas, email…etc). Un hacker interviene en el tráfico de datos de dos partes vinculadas entre sí en una comunicación haciéndose pasar por cualquiera de ellas, haciéndoles creer que se están comunicando entre ellos cuando en realidad es el intermediario quien recibe la comunicación.

-Ataques basados en servidores DHCP

Lo que buscan los hackers por esta vía es controlar las direcciones IP locales mediante el servidor DHCP simulado, para utilizar en su favor las puertas de acceso y el servidor DNS en los ordenadores víctimas y poder desviar el tráfico de datos saliente para interceptar y manipular su contenido.

-ARP cache poisoning

dar respuestas falsas en el proceso para lograr que el atacante use su ordenador como punto de acceso inalámbrico o entrada a Internet. Si es exitoso, el ataque permite leer todos los datos salientes de los ordenadores atacados, aparte de registrarlos o de manipularlos antes de enviarlos al lugar correcto.

-DNS

objetivo manipular las entradas en la caché de un servidor DNS haciendo que den direcciones de destino falsas. Si ha tenido éxito, los hackers pueden mandar a los usuarios de Internet a cualquier página web sin que nadie se dé cuenta.

El proceso se inicia cuando los datos del sistema de nombres de dominio se distribuyen por diferentes ordenadores de la red. Cuando alguien quiere acceder a una web lo suele hacer usando un nombre de dominio. También necesita una dirección IP, determinada por el router que tenga el usuario, para enviar la solicitud. Si hay entradas en la caché, el servidor DNS emite la respuesta a la solicitud con la IP que proceda, y si no las hay el servidor decidirá la IP con ayuda de otros servidores.

### -Ataque Man in the Browser

### Por último, el ataque Man in the Browser consiste en que el atacante instala malware en el navegador de los usuarios de Internet con la finalidad de interceptar sus datos. La principal causa para verse infectado por este ataque es el hecho de tener ordenadores que no están correctamente actualizados y que, por ello, ofrecen brechas de seguridad muy visibles que dan camino libre para infiltrarse en el sistema.

Para proteger tu API mediante la autenticación básica debes configurar que las conexiones entre los clientes y tu servicio API funcionen únicamente mediante una conexión TLS/HTTPS, nunca sobre HTTP.

1. basada en token

usuario y contraseña. Pero en este caso, con la primera petición de autentificación, el servidor generará un token basado en esas credenciales.El servidor guarda en base de datos este registro y lo devuelve al usuario para que a partir de ese momento no envíe más credenciales de inicio de sesión en cada petición HTTP.  En lugar de las credenciales, simplemente se debe enviar el token codificado en cada petición HTTP.Por norma general, los tokens están codificados con la fecha y la hora para que en caso de que alguien intercepte el token con un ataque MiTM, no pueda utilizarlo pasado un tiempo establecido. Además de que el token se puede configurar para que caduque después de un tiempo definido, por lo que los usuarios deberán iniciar sesión de nuevo.

1. basada en clave API

en este caso primero debes configurar el acceso a los recursos de tu API. Tu sistema API debe generar una clave (*key*) y un *secret key* para cada cliente que requiera acceso a tus servicios. Cada vez que una aplicación necesite consumir los datos de tu API, deberás enviar tanto la *key* como la *secret key*.Este sistema es más seguro que los métodos anteriores, pero la generación de credenciales debe ser manual y esto dificulta la escalabilidad de tu API.Puede ser complicado poder almacenar y administrar estas credenciales. Es por ello que es imprescindible contar con una [API Gateway](https://www.itdo.com/blog/api-gateway-en-tu-arquitectura-de-microservicios/). Es un sistema intermediario que proporciona una [interfaz API REST o WebSocket](https://www.itdo.com/blog/rest-vs-websocket-que-diferencia-hay/) para hacer de enrutador desde un único punto de entrada, el API Gateway, hacia un grupo de microservicios y/o API de terceros definidos. Interactúa como puerta de enlace “Gateway”.

1. OAuth 2.0 (Autorización abierta)

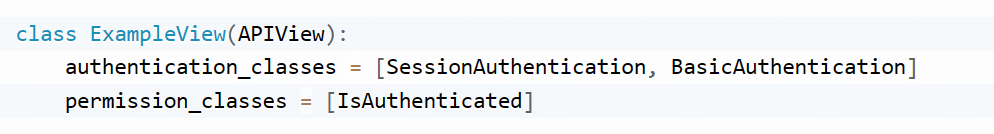
Su propósito es permitir a otros proveedores, servicios o aplicaciones, el acceso a la información sin facilitar directamente las credenciales de los usuarios.accederán bajo la confirmación del usuario

Los esquemas de autenticación predeterminados se pueden establecer globalmente, utilizando la configuración. Por ejemplo.DEFAULT\_AUTHENTICATION\_CLASSES

En settings.py



Esta configuración no sólo se puede indicar de forma global, podemos utilizarlas para que sólo se apliquen a una vista en específico.



Este tipo de autenticación utiliza un esquema HTTP simple basado en Token, un Token es un secuencia de caracteres especiales encriptados que representan información del usuario al cual se asocian. Este tipo de autenticación es ideal para configuraciones a proyectos cliente-servidor, donde hay clientes de varios tipos, puede ser escritorio o móvil.

Esta clase, incorporada en Django Rest Framework puede utilizarse como Autenticación Global, para ello sólo debemos aplicarla de la misma forma que SessionAuthentication y BasicAuthentication, adicionalmente debemos agregar rest\_framework.authtoken a nuestro archivo de configuración.

Cuando se realiza este tipo de Autenticación, utilizando esta clase, la clave del token debe incluirse en el header de la petición HTTP, debe tener como prefijo la palabra Token seguido de un espacio y luego el Token en cuestión.

Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b

Si la autenticación es realizada correctamente, colocará las siguientes credenciales en el request:

request.user será una instancia del modelo User que se halla definido.

request.auth será será una instancia de rest\_framework.authtoken.models.Token.

Toda respuesta que no se ha autenticado se les negará el permiso y darán como resultado una respuesta HTTP 401 Unauthorized .

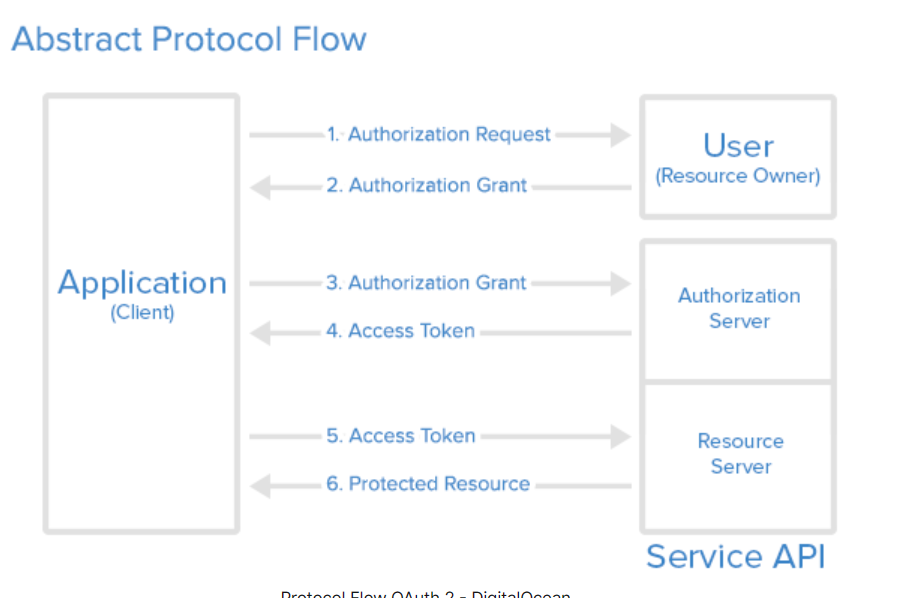
si configuro OAuth como autenticación en mi API, eso le permitiría delegar a un tercero para que use sus datos.

App x quiere acceder a los datos de tu cuenta de google

Le permitirá a la app...allow o cancel

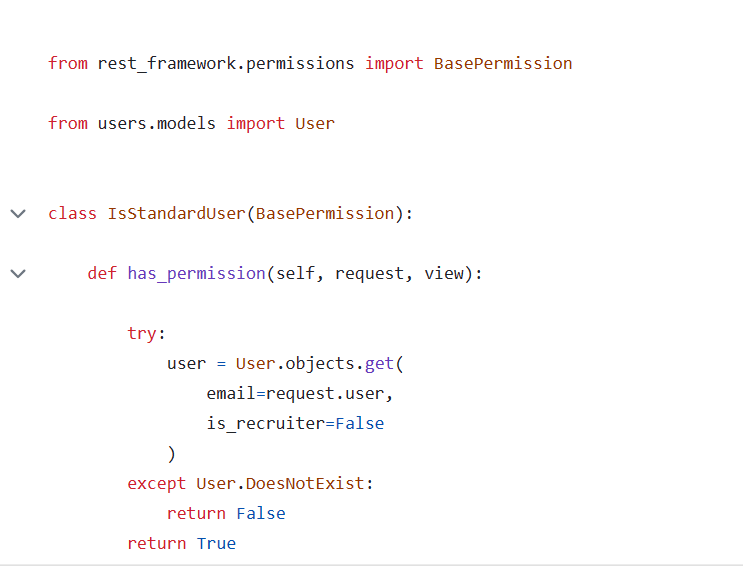
solicitando acceso a la cuenta de usuario de Google.

se solicita el acceso a la gestión de Google Calendar. Una vez aprobada la autorización, esta aplicación de terceros podrá acceder a la información permitida mediante una autentificación con un token de acceso.



Adecuado a mis necesidades

Django Rest Framework tiene por defecto algunos paquetes de permisos que permiten saber por ejemplo si el usuario está autenticado, mediante la clase IsAuthenticated, pero además en esta aplicación se añade una restricción nueva para evitar que el administrador modifique o añada accidentalmente datos de los usuarios desde la url y pueda hacerlo sólo desde el panel de administrador para mayor seguridad. Se creó un archivo permissions.py en el que se añadió el siguiente código

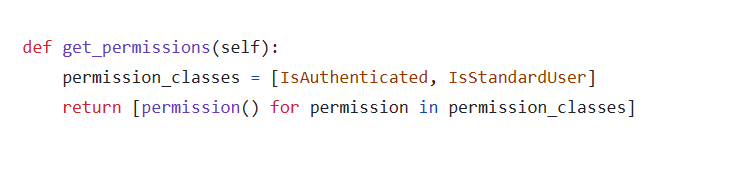


el método authenticate que necesitará como parámetros obligatorios username y password

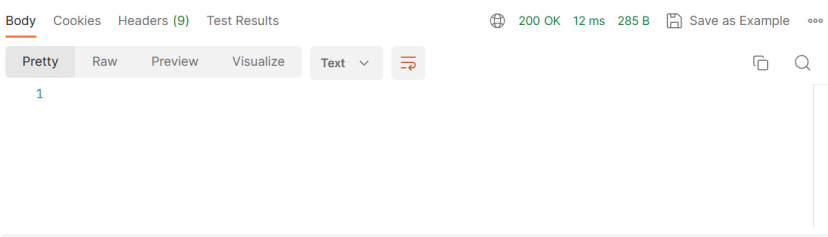
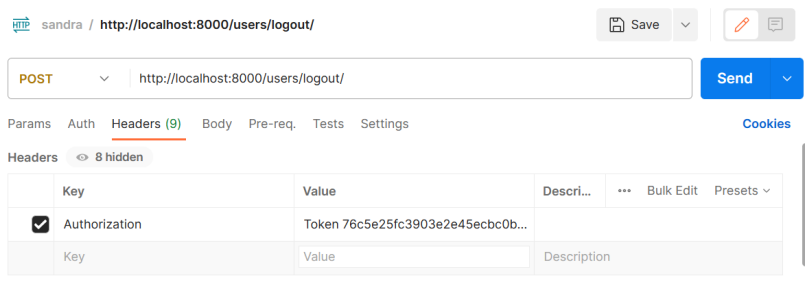


En el login se opta por sustituir el username obligatorio por email, este cambio se refleja en UserLoginSerializer y además hay que especificarle en models.py que el campo username sería el equivalente al email:

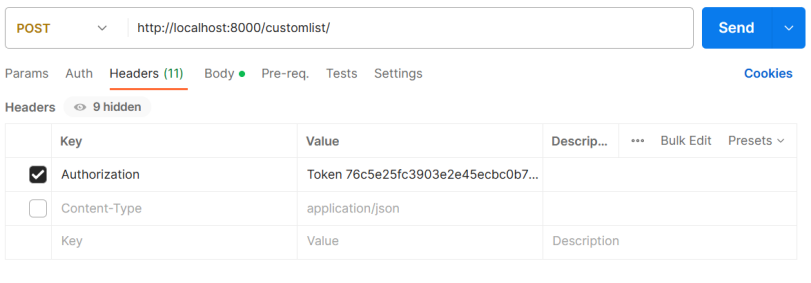
método get\_permissions para verificar que quien requiera una acción de listado, modificación,creación o borrado desde la url sea el usuario estándar y no el administrador, que sólo podrá realizar esta acción desde el panel de administración. En views.py de customlisttrack



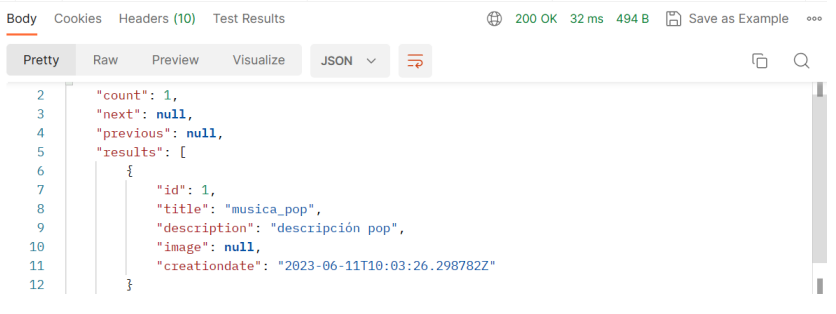
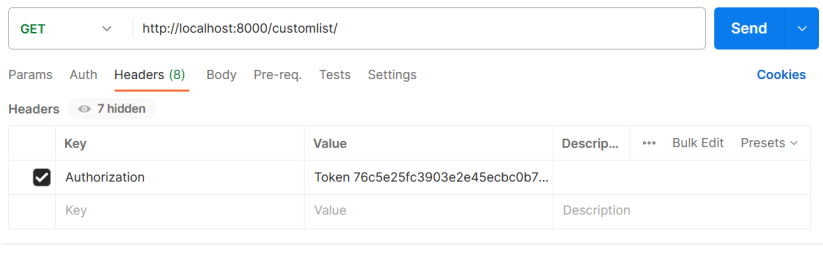
Para cerrar la sesión de usuario hay que pasarle por cabecera a http://localhost:8000/users/logout/ la variable Authentication en la que especificamos Token seguido de un espacio y el token asociado al usuario que quiere cerrar sesión.



Para la creación de una lista de usuario a través de http://localhost:8000/customlist/ necesitamos pasarle como cabecera la clave authorization junto con el token del usuario.



Con el método GET y especificando en la cabecera el token del usuario, el usuario accederá a <http://localhost:8000/customlist>



para poder borrarla se hace de la siguiente manera:

