**wipe**

[**1. Conceptos principales 5**](#_mbtbmpmxyy7k)

[a. Estructura de los principales bloques de datos 5](#_ipg0h5qv67kp)

[b. Esquema simplificado de la BBDD 7](#_eg4ffkdhiyra)

[c. Glosario de atributos 8](#_aaeg3usu5i3b)

[i. Corporate 8](#_m13fcokp08ah)

[ii. Company 8](#_ac9y0lqju6z)

[iii. Area 8](#_a7m552vpdl7l)

[iv. Campaign 8](#_4vfkrjtzdtou)

[v. Plot 9](#_5irpu5yw9tp1)

[vi. Lot 9](#_ns6azqvg10df)

[d. Permisos de usuarios, roles y límites 10](#_j179pqhj00e5)

[i. Admin 10](#_k4kebka3ayb)

[ii. Basic 10](#_pxqoamz7m218)

[iii. Funcionamiento de los límites en la API 10](#_szgm2lxi5n5n)

[e. Estándares de atributos 11](#_gp03gxjfra50)

[f. Una vista en detalle a campaigns 12](#_70ufbtmibs2u)

[i. Tipos de cultivo admitidos 12](#_fn1smwxlzay2)

[g. Una vista en detalle a operations 12](#_duzy613g5y9t)

[i. Tipos de operation admitidos 13](#_u2z6w549dgkr)

[ii. Unidades de medida admitidas 13](#_x38oj7kuvxmf)

[h. Una vista en detalle a extradata 13](#_fc2vicboud5j)

[i. Tipos de extradata 13](#_cp4lyjfrfkk8)

[i. Introducción al concepto API 13](#_wccpcs8g79wn)

[i. Inicio de sesión 14](#_kcf597rbhn7s)

[j. Que es una petición API 17](#_ydydhfmclfrh)

[i. URL 17](#_iaqhdweaig0q)

[ii. Request Body 17](#_uak8m0com917)

[iii. Response 17](#_a2js0ptyzg0b)

[k. Tipos de peticiones 18](#_98p5kan8gm1y)

[i. GET 18](#_lkfysg9vmx9h)

[ii. POST 18](#_ixoop0t97mmh)

[iii. PUT/PATCH 18](#_luij6f8wiqmz)

[iv. DELETE 18](#_ph55c4rnuclx)

[l. Uso de Swagger 19](#_f1ec9atmu0zp)

[m. Importación de la especificación OpenAPI en Postman 21](#_bl51e9oqkos0)

[**2. Implementación básica recomendada 24**](#_5fjimqburabm)

[a. Implementaciones básicas 24](#_evsw5i6ahavw)

[i. Obtener token de acceso 24](#_7l41172xtkxo)

[ii. Obtener ids de entidades 25](#_p4mf1obasxtt)

[b. Pasos a seguir 28](#_gib8fn46367t)

[i. Actualizar datos de cultivo 28](#_kx5zjhgmz3as)

[ii. Atributos datos de cultivo  
 Atributos generales de Datos del cultivo 30](#_69bhdpbzbkpw)

[Unidades de agrupación en Datos del cultivo 31](#_m20uk82x3pt2)

[Atributos específicos de caña de azúcar 32](#_j1l002eapuym)

[Atributos específicos de arroz 32](#_ioew6oertixu)

[iii. Subida de operaciones 32](#_xlj9bezdxw6z)

[iv. Atributos operaciones 34](#_896rd9tmbdm5)

[Atributos generales de Operaciones 34](#_70uk8aevml4w)

[v. Subida de resultados de muestreos inteligentes 35](#_849omxvdmkl9)

[vi. Atributos muestreos inteligentes 38](#_jiwaxzl2e27q)

[Atributos generales de muestreos inteligentes 38](#_cra7a9v8hx59)

[vii. subida de datos climáticos 39](#_l1iloqcjn0dn)

[viii. Atributos datos climáticos 42](#_ntpnzncvey2z)

[Atributos generales de Registros climatológicos 42](#_pzcl5i8mxry9)

[**3. Uso de la API 43**](#_ams42zoq7osy)

[a. Peticiones básicas a la API NAX 43](#_3xfy89tymph2)

[i. Corporate 43](#_k291t5hjrrfx)

[1. GET 43](#_13vi40soctd2)

[2. PUT 43](#_wwu7auutbvk8)

[3. GET Users from company 43](#_u6kqcdintot4)

[ii. Company 43](#_lwzp5tggpqlc)

[1. GET 43](#_j2r1naoa0hbo)

[2. POST 44](#_k5rkdh5x5z3s)

[3. PUT 45](#_buvph8jnkz6m)

[4. DELETE 45](#_x4jihw622l5w)

[5. GET areas from company 45](#_arii5x5embp3)

[6. GET lots from company 45](#_fe1k10158exk)

[7. GET plots from company 45](#_g09oavvkj52)

[iii. Area 45](#_9vblkucsmxw8)

[1. GET 45](#_9rtpeniu7tj6)

[2. POST 45](#_p4kjzag7h7bf)

[3. PATCH 46](#_66rzjzkbdozr)

[4. DELETE 46](#_vg3zf22zefl4)

[5. GET Campaigns from areas 46](#_c5uk2aj55sie)

[iv. Campaign 46](#_q3kp2tlzdde0)

[1. GET 46](#_2cnlogfqid72)

[2. POST 46](#_eq7uqsi0f9w4)

[3. PATCH 46](#_1xol90mdt3n6)

[4. DELETE 47](#_cxh65bcmi8pg)

[5. GET lots from campaign 47](#_vrjx9odnuy0y)

[6. GET plots from campaign 47](#_5w9j8ccjvlbf)

[7. GET products from campaign 47](#_tgz34e68um75)

[v. Lot 47](#_6tcqptycn226)

[1. GET 47](#_9sydifn1yl3h)

[2. PUT 47](#_n2y9t99xv6p)

[3. DELETE 48](#_9v2apf72drx4)

[4. Subida de lots 48](#_qogwdid381mo)

[5. Actualización de lots por nombre 49](#_d6dp7ae08tsy)

[6. Actualización de lots 50](#_rza1okazl7w4)

[7. Actualizar lots high frequency 51](#_aps15h91tnsc)

[8. Subida de operaciones de lots 52](#_tlqtxcozgiir)

[vi. Plot 52](#_w0kq5uvgg5fp)

[1. GET curvas de un plot 52](#_kkt9pormj44w)

[2. GET stats de un plot 53](#_z96rcm7w5p9w)

[vii. Users 53](#_gdrn4ai953b9)

[1. GET mis areas 53](#_pro3wbdgc2s7)

[2. GET mis datos 53](#_ouk3zqa5ai3f)

[3. PUT mi configuración 53](#_dj0f5z6hc87b)

[**4. Anexo I: Ejemplos de subida de CSV 54**](#_xlmfem6g94yu)

[**5. Anexo II: Tabla de permisos por rol 55**](#_1ne6yamo86g)

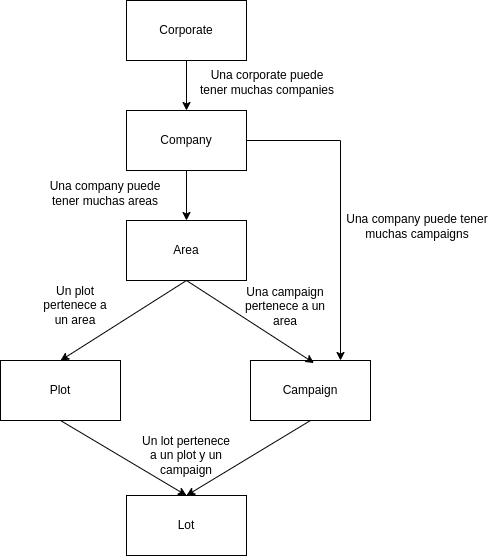
[a. Admin 55](#_hj2hh0arnm97)

[b. Basic 56](#_le3379jura9y)

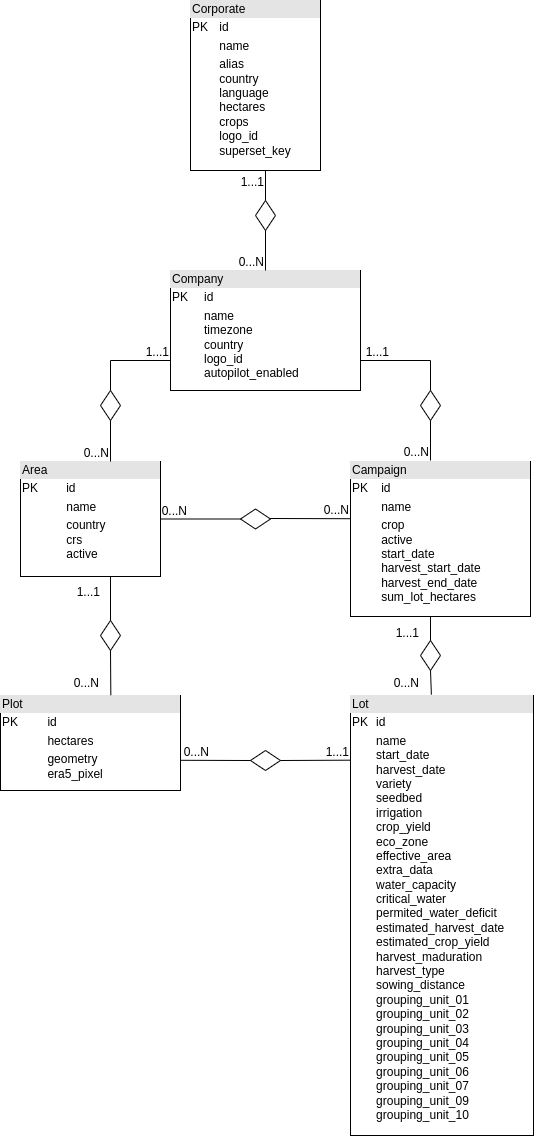
# Conceptos principales

## Estructura de los principales bloques de datos

* **Plots**: Un plot representa una parcela o un terreno. En un plot se almacenan los atributos mínimos principales que describen las características de cada parcela y definen su geometría. Equivalencia: *Parcela.*
* **Area**: Consiste en una agrupación de plots, es decir, los plots pertenecen a un área. A nivel conceptual, se entiende como una zona o región geográfica donde se ubica una agrupación de plots, Importante: La distancia máxima entre los plots de los extremos vertical y horizontal no debe exceder los 100 km. Equivalencia: *Área, Ingenio.*
* **Campaign**: Incluye los datos relacionados con las actividades necesarias para un ciclo concreto de siembra, cultivo y cosecha, en un area determinada. Define las condiciones de producción de un ciclo concreto. Equivalencia: Ciclo de cultivo, *Campaña*, en caña de azúcar corresponde a un periodo de *Zafra.*
* **Lot**: Información de un **plot** para un **campaign** concreto, es decir, un lot almacena los datos de un plot para una campaña puntual (ej. variedad, fecha de cosecha, etc.) Equivalencia: Atributos o datos de un polígono.
* **Company**: Empresa que posee una o varias areas.
* **Corporate**: Entidad que engloba varias **companies**. Puede aplicarse a un usuario que distribuye a varias empresas separadas, o a varias compañías que forman parte de un mismo corporativo.



## Esquema simplificado de la BBDD



## Glosario de atributos

### Corporate

* **id:** identificador de la corporate.
* **name:** Nombre de la corporate.
* **alias:** Alias de la corporate.
* **country:** País de la corporate en código de dos letras, ej:”ES”.
* **language:** idioma de la corporate, en el siguiente formato, ej:”es-ES”.
* **hectares:** hectáreas totales de la corporate.
* **crops:** cultivos a los que se dedica la corporate.
* **logo\_id:** UUID del logo de la corporate.
* **superset\_key:** UUID del superset de la corporate.

### Company

* **id:** identificador de la company.
* **name:** nombre de la company.
* **timezone:** huso horario de la company.
* **country:** país de la company en código de dos letras, ej:”ES”.
* **logo\_id:** UUID del logo de la company.
* **autopilot\_enabled:** indica si la company tiene activado el “autopilot”.

### Area

* **id:** Identificador del area.
* **name:** nombre del area.
* **country:** país del area en código de dos letras, ej:”ES”. Se puede consultar una lista de códigos de todos los países en el siguiente [enlace](https://www.ssb.no/en/klass/klassifikasjoner/100)
* **crs:** (Coordinate Reference System) es un sistema que determina cómo se ubican y representan los puntos en la Tierra, ej:”EPSG 4326”.
* **active:** indica si el area está activa

### Campaign

* **id:** identificador del campaign.
* **name:** nombre del campaign.
* **crop:** cultivo del campaign.
* **active**: indica si el campaign está activo.
* **start\_date:** fecha de inicio del campaign.
* **harvest\_start\_date:** fecha de inicio de la cosecha en ese campaign.
* **harvest\_end\_date:** fecha de fin de la cosecha en ese campaign.
* **sum\_lot\_hectares:** suma de hectáreas de todos los lots de ese campaign.

### Plot

* **id:** identificador del plot.
* **hectares:** hectáreas del plot. Calculado por NAX a partir del polígono.
* **geometry:**  geometría del plot.
* **era5\_pixel:** numero de pixel de era5. Asignado por NAX automáticamente.

### Lot

* **id**: identificador del lot.
* **name**: nombre del lot.
* **start\_date**: fecha de inicio del lot.
* **harvest\_date**: fecha de la cosecha.
* **variety**: variedad del lot.
* **seedbed**: tipo de cama de siembra utilizada.
* **irrigation**: método de riego empleado.
* **crop\_yield**: rendimiento esperado del lot.
* **eco\_zone**: zona ecológica donde se ubica el lot.
* **effective\_area**: area efectiva destinada al lot.
* **extra\_data**: datos adicionales relevantes para el lot definidos por el usuario.
* **water\_capacity**: capacidad de retención de agua del suelo en el lot.
* **critical\_water**: cantidad mínima de agua necesaria para el lot.
* **permitted\_water\_deficit**: déficit de agua permitido durante el cultivo del lot.
* **estimated\_harvest\_date**: fecha estimada de cosecha del lot.
* **estimated\_crop\_yield**: rendimiento estimado del lot.
* **harvest\_duration**: duración de la cosecha del lot.
* **harvest\_type**: tipo de cosecha (manual o mecánica) del lot.
* **sowing\_distance**: distancia entre siembras en el lot.
* **grouping\_unit\_01**: unidad de agrupamiento 01.
* **grouping\_unit\_02**: unidad de agrupamiento 02.
* **grouping\_unit\_03**: unidad de agrupamiento 03.
* **grouping\_unit\_04**: unidad de agrupamiento 04.
* **grouping\_unit\_05**: unidad de agrupamiento 05.
* **grouping\_unit\_06**: unidad de agrupamiento 06.
* **grouping\_unit\_07**: unidad de agrupamiento 07.
* **grouping\_unit\_09**: unidad de agrupamiento 09.
* **grouping\_unit\_10**: unidad de agrupamiento 10.

## Permisos de usuarios, roles y límites

Un usuario siempre pertenecerá a uno de dos roles disponibles. Los roles son:

### Admin

Un usuario con rol Admin opera a nivel de company y estará limitado a una o varias companies. Podrá ver todas las geometrías y realizar cualquier acción dentro de las mismas

### Basic

Un usuario con rol Basic estará limitado a una sola compañía. Dentro de esta compañía se podrá elegir si se quiere limitar al usuario o darle acceso a todos los plots de la compañía.

En caso de elegir limitar el usuario se podrá limitar en base a uno de los siguientes casos:

* Limitado en base a una o varias areas dentro de la compañía
* Limitado a solo algunos plots en concreto dentro de la compañía
* Limitado a solo los plots que pertenecen a una determinada grouping\_unit de la compañía

| **ATENCIÓN: Cuando se añadan o modifiquen los limites de un usuario en base a una grouping\_unit, los cambios pueden tardar hasta 24h en verse reflejados en la plataforma.** |
| --- |

***En el Anexo II se encuentran las tablas con todos los permisos que existen en la aplicación especificados por rol.***

### Funcionamiento de los límites en la API

Es posible restringir el acceso de los usuarios a ciertas partes de la API según su rol. Una vez implementadas estas restricciones para un usuario, éste sólo podrá acceder a los contenidos y funcionalidades para los que esté autorizado. Las posibilidades de limitación según el rol son las siguientes.

* **Admin**: Un admin opera a nivel de company. Se le podrá dar acceso a una o varias companies y tendrá acceso a todos los datos que se encuentren dentro de las mismas.
* **Basic**: Los usuarios Basic operan dentro de una compañía. A un usuario Basic se le podrá limitar el acceso a una o varias areas o a uno o varios plots.

Es importante tener en cuenta que no es posible aplicar restricciones más específicas de las que corresponden a un rol (por ejemplo, no se puede restringir a un admin a una sola area). Si se desea un nivel de acceso más restringido, será necesario cambiar el rol del usuario (por ejemplo, cambiar a un admin a basic para limitar su acceso a una única area).

Además, es importante destacar que las restricciones se escalan automáticamente. Esto significa que, al limitar a un basic a un plot específico, automáticamente se restringe su acceso al area y a la company a la que pertenece ese plot, garantizando así un control jerárquico de los permisos.

## Estándares de atributos

Algunos atributos en la API siguen estándares uniformes en toda su estructura. A continuación, se presentan algunos ejemplos:

* **country**: Los países siempre se representarán utilizando su código de dos letras, por ejemplo: "ES" para España, "MX" para México.
* **language**: Los idiomas se indicarán con el formato de código de idioma y región, por ejemplo: "es-ES" para español de España, "en-EN" para inglés.
* **dates**: Las fechas seguirán el formato **yyyy-mm-dd**, por ejemplo: "2024-03-31" para el 31 de marzo de 2024.

## Una vista en detalle a campaigns

Una campaña está definida por varios elementos clave: el **nombre**, el **cultivo**, las **áreas** de trabajo, y las **fechas** importantes, que incluyen el **inicio de la campaña**, el **inicio de la cosecha**, y el **fin de la cosecha**. Estas parcelas seguirán sus ciclos de cosecha basados en dichas fechas.

El nombre de la campaña se asigna automáticamente y sigue una convención específica, que consiste en indicar el mes y el año de la fecha de inicio de la cosecha, seguido por el tipo de cultivo entre paréntesis. Por ejemplo, para una campaña con fecha de inicio de cosecha el 12 de abril de 2024 y con arroz como cultivo, el nombre será “04/2024 (RICE)”.

Es importante tener en cuenta que un área de trabajo no puede estar asociada a dos campañas con el mismo tipo de cultivo si las fechas de inicio de cosecha de ambas coinciden en el mismo mes y año. En otras palabras, un área no puede vincularse a dos campañas distintas que tengan la misma fecha de inicio de cosecha y el mismo tipo de cultivo, ya que no serían consideradas campañas diferentes.

En cuanto a la edición de la información de una campaña, la única fecha que puede modificarse es la de fin de cosecha, y solo si la nueva fecha es posterior a la fecha de fin de cosecha original. Además, es posible modificar las áreas de trabajo asociadas a la campaña, pero no se permite editar el resto de las fechas ni los atributos previamente establecidos de la campaña.

### Tipos de cultivo admitidos

***[‘sugarcane’, ’rice’, ’soya’, ’wheat’, ’corn’, ’chikpea’, ’sunflower’, ’cotton’, ’dam’, ’horticulture’, ’fruitgrowing’, ’tree’, ’other’, ’walnut’, ’pecan’, ’barley’, ’pasture’, ’bean’, ’fallowing’, ’avocado’, ’peanut’]***

## Una vista en detalle a operations

Las operaciones representan los trabajos realizados o que se realizarán sobre los cultivos de una parcela. El objetivo de cada operación varía según el tipo de operación que se esté llevando a cabo.

Cada operación está vinculada a un lote, su área, y la campaña en la que se realiza o se llevará a cabo. Además, las operaciones tienen un tipo específico, un identificador único, y están asociadas a un producto utilizado, junto con la cantidad de dicho producto y su unidad de medida. También incluyen la superficie en hectáreas sobre la que se ha realizado o se realizará la operación.

Entre los atributos clave, las operaciones cuentan con una fecha de inicio y, opcionalmente, una fecha de fin. Si la fecha de fin no se especifica, se considerará que la operación tiene una duración de un solo día, coincidiendo con el día de inicio en los gráficos.

### Tipos de operation admitidos

***["irrigation", "fertilization", "maturing\_agent", "others", "potassium", "fungicides", "insecticides", "herbicides", "resowing"]***

### Unidades de medida admitidas

***[‘ton’, ’l’, ’ml’, ’kg’, ’g’, ’u’, ’ha’, ’m3’, ’cm3’]***

## Una vista en detalle a extradata

En algunas entidades de la aplicación, como lots y operations, los usuarios tienen la posibilidad de definir sus propios campos y atributos personalizados, conocidos como “extradata”.

El extradata se almacena en la base de datos en formato JSON. Al recuperar un lot o una operation de la base de datos, se recibirá un campo llamado extradata, que contiene estos datos personalizados en formato JSON. Este JSON únicamente incluirá los atributos extradata que hayan sido completados. Si un lot tiene un atributo extradata que no ha sido rellenado, la clave correspondiente no aparecerá en el JSON.

Cada extradata puede tener sólo ciertos tipos predefinidos, y se validará según el tipo elegido cuando se carguen los datos.

### Tipos de extradata

***[‘integer’, ‘string’, ‘bool’, ‘choices’]***

## Introducción al concepto API

La API es el conjunto de servicios que se ofrecen para trabajar con tablas, modelos y sus atributos. Podemos interactuar con ella mediante código o desde aplicaciones de testing como Postman. En el siguiente apartado explicaremos cómo usar la API desde [Swagger](https://beta-api.naxsolutions.com/api/v1/docs), que ofrece una mejor visualización gráfica, permitiendo conocer las peticiones disponibles.

Las peticiones vienen tituladas por su tipo de petición (GET, POST, DELETE, PUT) y su url de acceso. Cuando desplegamos una petición podremos observar una breve descripción de la misma, los parámetros de la url, el request body y el response (la respuesta que recibiremos al hacer la petición). Todo esto lo explicaremos en la siguiente sección, por ahora nos centraremos en iniciar sesión para obtener el token de acceso.

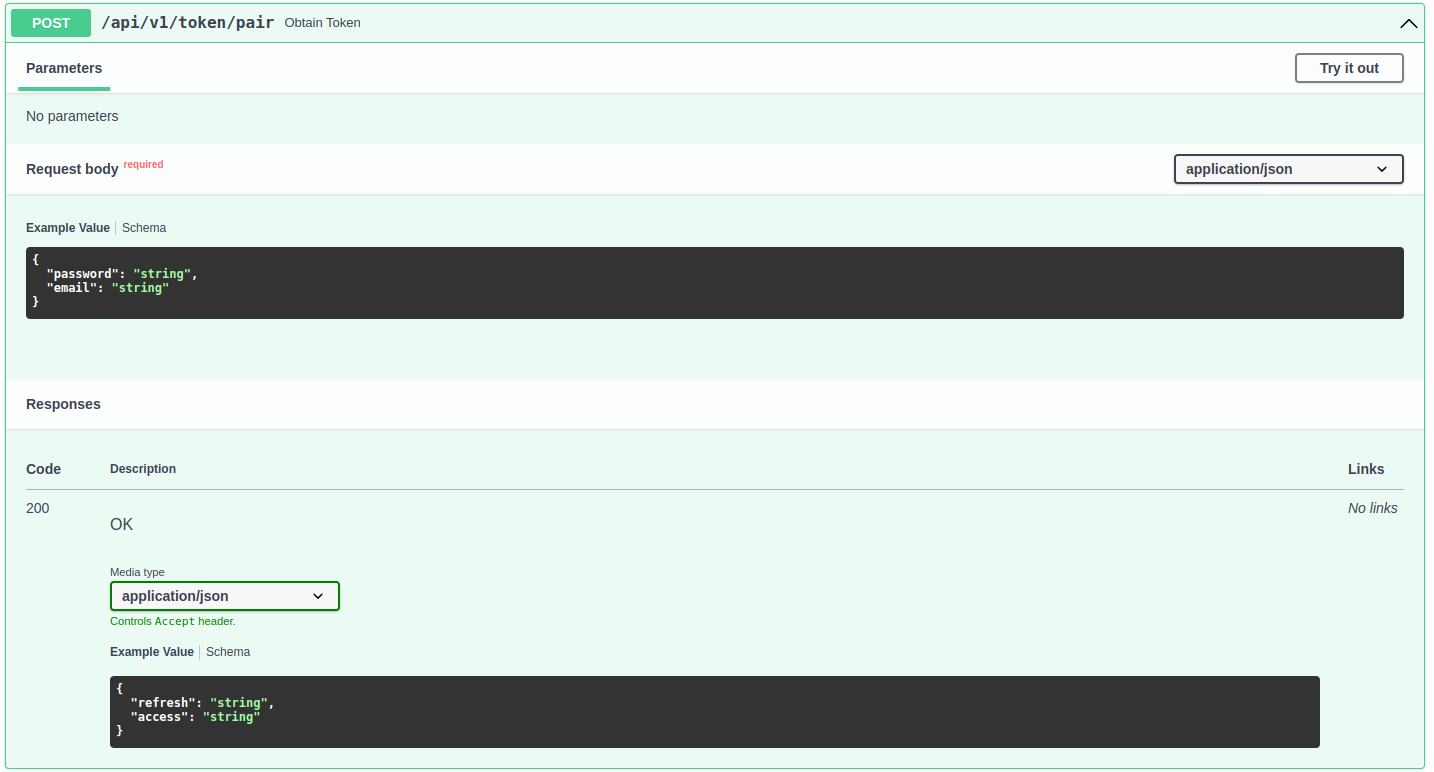
Para comenzar, es necesario contar con un usuario registrado. Si no tienes uno, debes solicitar a alguien que ya posea una cuenta con permisos iguales o superiores a los que se necesitan para la nueva cuenta que cree el usuario, siguiendo las instrucciones que se detallan un poco más adelante.

### Inicio de sesión

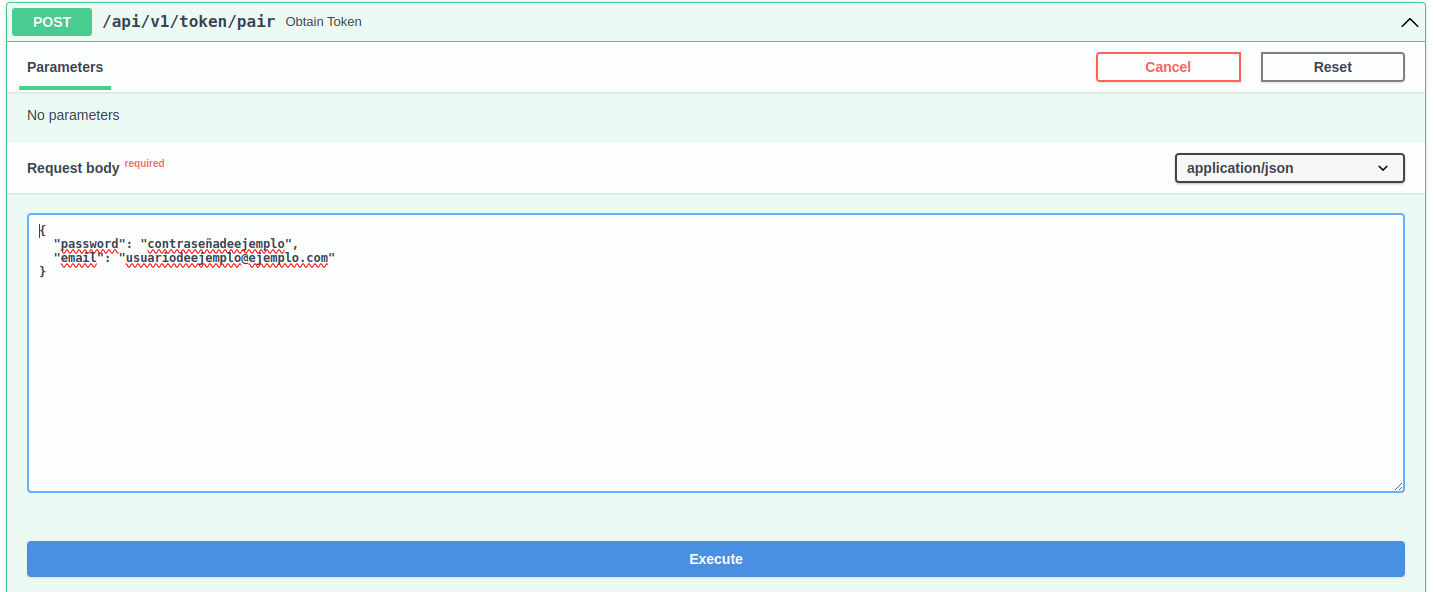
Para iniciar sesión con un usuario ya registrado debemos hacer uso de la petición **/api/v1/docs/token/pair** :



Al hacer clic sobre la petición se nos abrirá un desplegable similar a este:



Lo primero que debemos hacer es hacer clic en el botón “try it out” situado en la parte superior derecha, una vez hecho esto se nos desplegará un cuadro de texto donde podremos introducir nuestro usuario y contraseña.



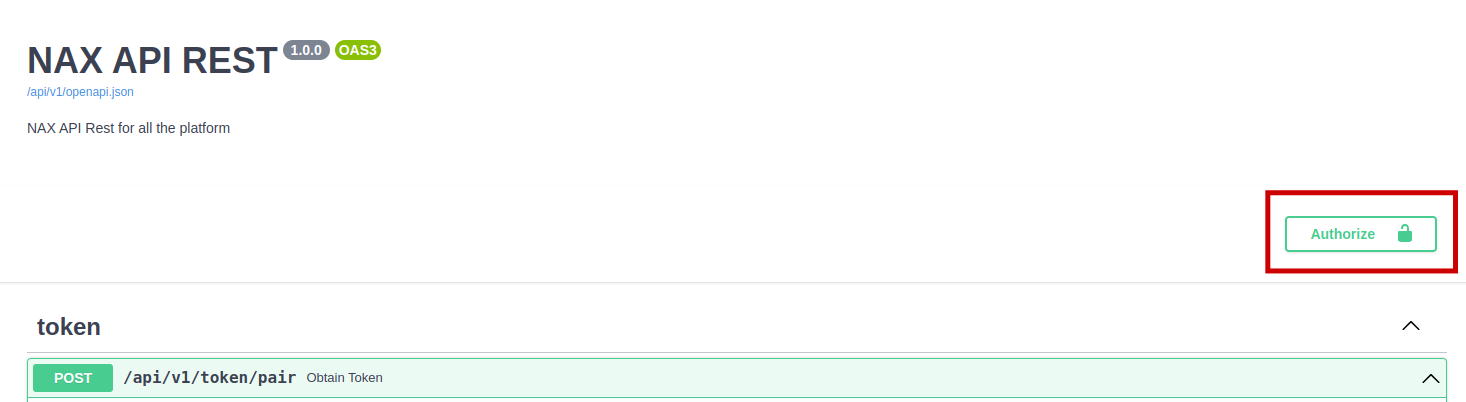
Una vez que hayamos ingresado nuestro usuario y contraseña, hacemos clic en el botón “Execute”. Debería aparecer el token en la sección de respuesta.

  
  
Podemos ver que aparecen dos tokens diferentes, uno llamado refresh y otro llamado access. El token refresh sirve para refrescar la sesión y obtener un nuevo token de acceso, ya que el token de acceso caduca cada 10 minutos. Podemos utilizar la siguiente petición para ello, o simplemente volver a iniciar sesión.

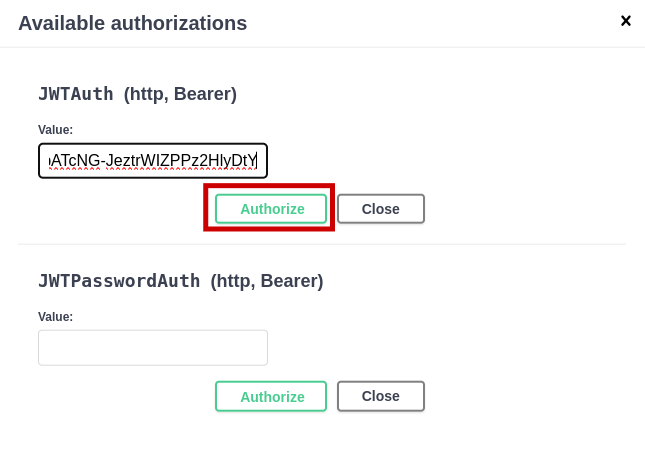


El token refresh dura 24h, una vez pasado ese tiempo debemos volver a iniciar sesión y obtener un nuevo token access y refresh.

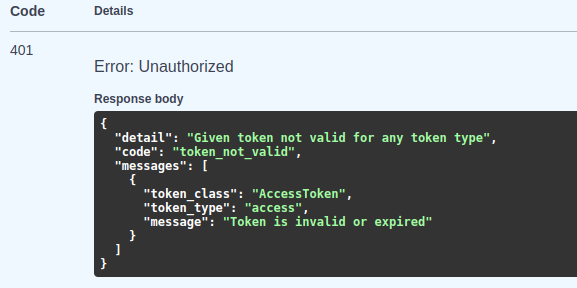
De momento, el token que nos interesa es el de acceso. En la parte superior de la página, o bien en la parte derecha de cada endpoint que requiera autorización, encontraremos un botón llamado “Authorize”, representado por un candado.



Al pulsar el botón se nos abrirá una ventana emergente donde podremos introducir el token de acceso que habremos copiado **sin las comillas** de la respuesta de la petición, posteriormente pulsamos el botón “authorize”.



| **ATENCIÓN: esto no implica que estemos autorizados, solo hemos introducido el token que utilizarán las peticiones a partir de ahora. Si hemos introducido mal el token o ha caducado, cuando intentemos hacer cualquier otro tipo de petición nos dará un error con un estado de prohibido como el siguiente:** |
| --- |

****

## Que es una petición API

Una **petición** es un servicio que se ofrece para interactuar con una base de datos, permitiendo a los usuarios realizar diversas operaciones sobre la información almacenada. Para facilitar este proceso, las peticiones se estructuran en tres partes fundamentales: la **URL**, el **Request Body** y la **Response**.

### URL

Es la dirección única que define el servicio al que se está accediendo. Esta dirección es crucial, ya que indica al servidor el recurso específico con el que se desea trabajar. Las URLs pueden contener parámetros que permiten a la petición identificar de manera precisa los objetos que se van a manipular.

Por ejemplo, en una URL como **/companies/{company\_id}/areas**, "company\_id" es un parámetro que representa el identificador único de una company específica. Es importante destacar que todas las peticiones requieren de una URL, ya que esta es la base para que el servidor entienda qué operación se está solicitando.

### Request Body

Es la parte de la petición donde se incluye información adicional que es necesaria en algunas solicitudes, especialmente cuando se quiere introducir o actualizar datos en el objeto identificado por la URL. Este cuerpo de la petición está en formato JSON. Por ejemplo, si se desea crear un nuevo registro en la base de datos, el Request Body puede contener datos como el nombre, la descripción y otros atributos relevantes del objeto que se está creando.

### Response

Es la respuesta que el servidor envía al cliente después de procesar la petición. Esta respuesta proporciona información sobre el resultado de la operación solicitada. Puede incluir un código de estado HTTP que indica si la petición fue exitosa (por ejemplo, un código 200) o si ocurrió un error (como un código 404 para "Not found"). Además, el cuerpo de la respuesta puede contener datos útiles, como el objeto que se ha creado o actualizado, o un mensaje de error detallando el problema encontrado.

## Tipos de peticiones

### GET

* Suelen usarse para recuperar información de la bbdd
* No utilizan un request body.
* Su respuesta es la información pedida para visualizar.

### POST

* Suelen utilizarse para crear un objeto
* Utilizan un Request Body necesario para introducir los datos que se desean crear.
* Muchas peticiones POST requieren rellenar el Request Body con un mínimo de atributos necesarios para crear un nuevo elemento en la base de datos.
* Su respuesta suele ser el objeto creado.

### PUT/PATCH

* Suelen usarse para editar un objeto
* Utilizan un Request Body necesario para introducir los datos que se desean editar.
* Su respuesta suele ser el objeto editado.

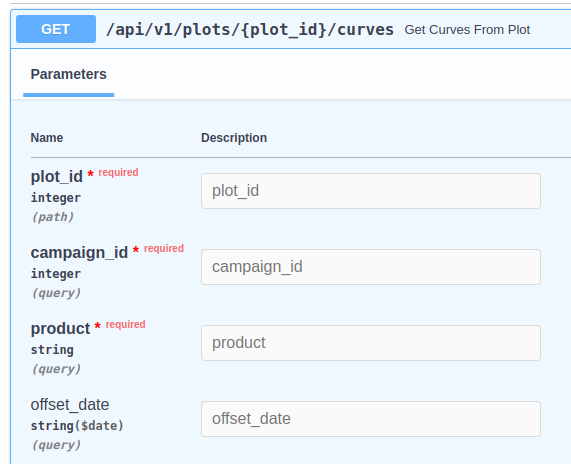
### DELETE

* Suelen usarse para eliminar un objeto
* Pueden no requerir de Request Body.
* Su respuesta es un estado HTTP 204 o 200 para comunicar que se eliminó el contenido.

## Uso de Swagger

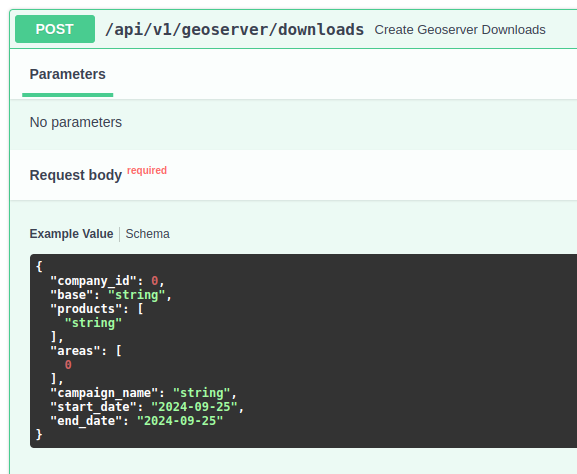
[Swagger](https://beta-api.naxsolutions.com/api/v1/docs) es una herramienta que facilita la interacción con APIs al proporcionar una interfaz gráfica para probar y documentar las solicitudes y respuestas. En swagger, además de conocer de qué endpoints dispone la API podemos además conocer qué estructura debe tener el payload de cada endpoint.

Encontramos dos formas principales para introducir los datos necesarios para un endpoint. El primero de ellos nos permite introducir los PathParams (parámetros que se envían en la url) así como los QueryParams.

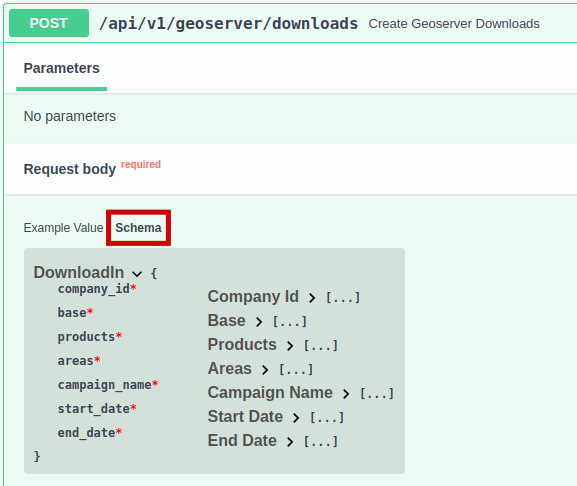


Como podemos ver en el ejemplo anterior en este tipo de parámetros podemos ver directamente cuales son obligatorios y el tipo de dato al que deben pertenecer. Se nos indica además si dicho parámetro va a ser un Query o un Path param.

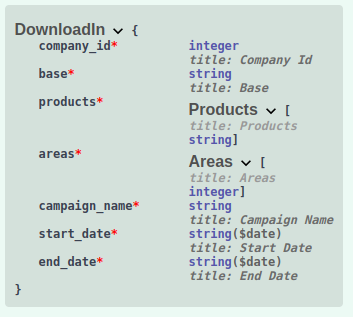
La segunda forma en la que puede ser necesario introducir datos es mediante el uso de un payload en formato JSON.



En este caso, podemos conocer qué atributos son obligatorios pulsando en el botón “schema” como se muestra en la siguiente imagen.



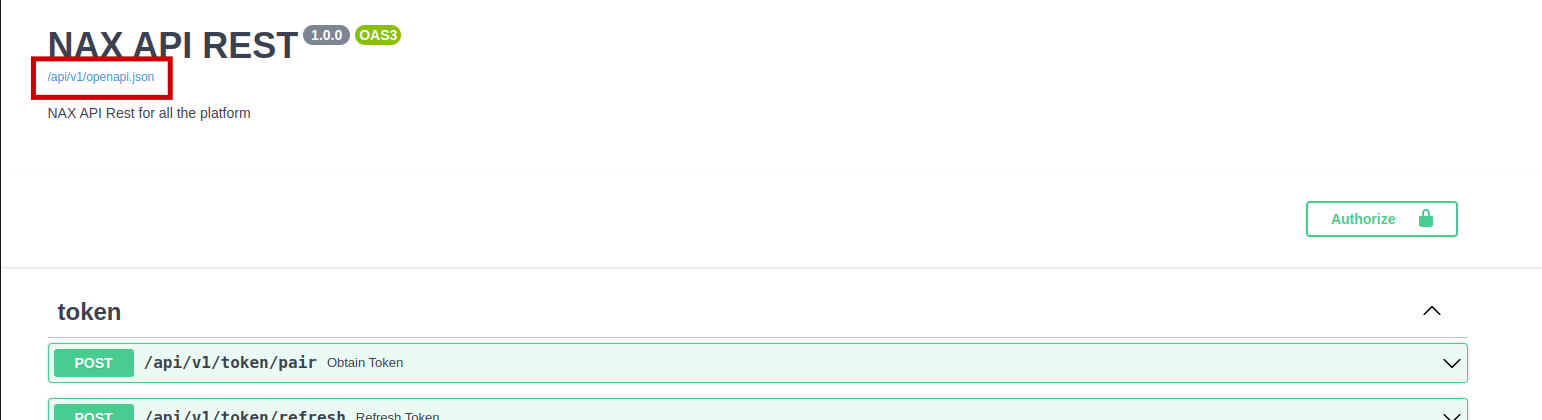
Podemos además hacer click en los desplegables que se muestran a la derecha de cada atributo en el esquema y ver de qué tipo deben ser y qué estructura deben tener los datos.



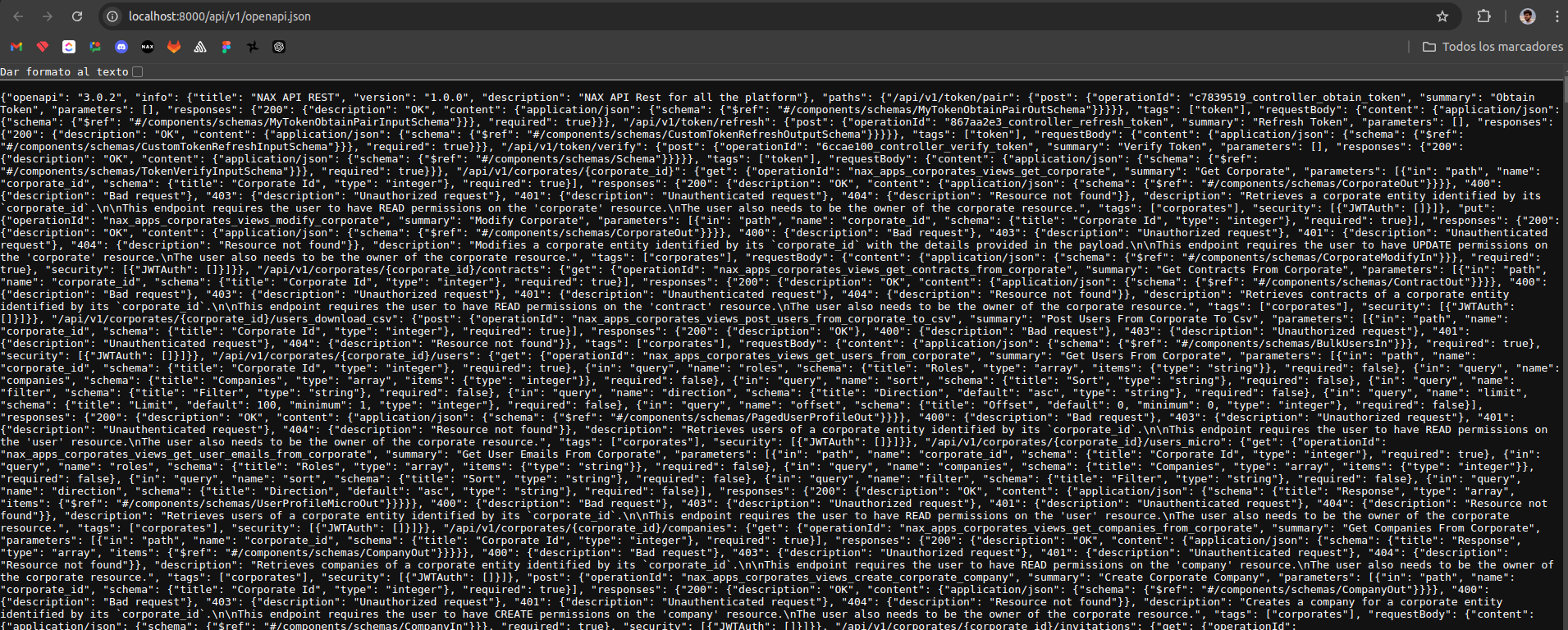
## Importación de la especificación OpenAPI en Postman

Existe la posibilidad en caso de que así se desee de importar la api en postman mediante su especificación de OpenAPI. Para ello debemos seguir unos sencillos pasos.

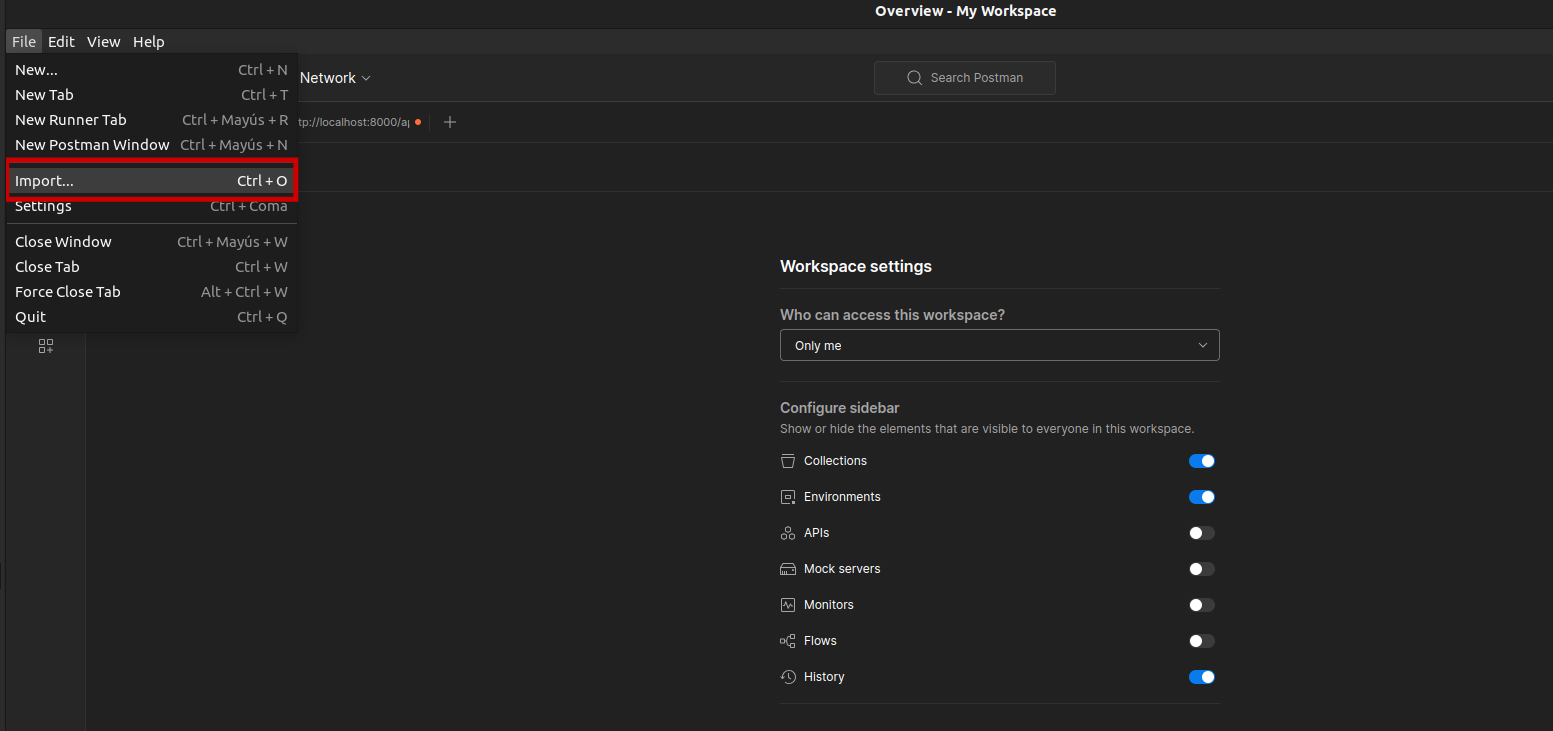
Primero debemos obtener la especificación OpenAPI de la API desde swagger en formato JSON, para ello hacemos click en api/v1/openapi.json, justo debajo del título de la página como se indica en la siguiente imagen.



Esto nos abrirá un archivo como el siguiente en el navegador, que debemos copiar íntegramente.



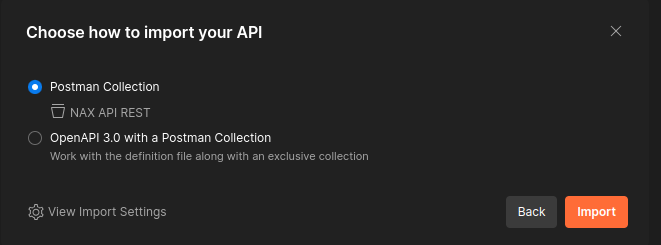
A continuación, ya en postman, abrimos el desplegable “file” de la barra superior y hacemos click en “Importar…”



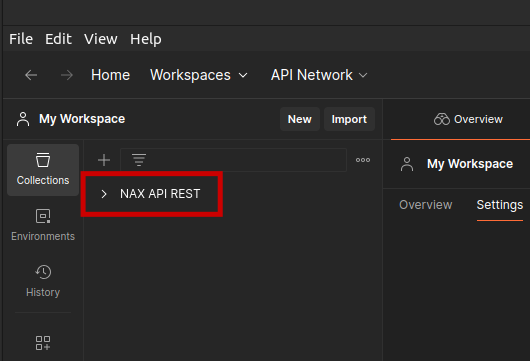
Esto nos abrirá una ventana emergente, donde deberemos introducir el texto copiado anteriormente en el siguiente cuadro de texto



Al pegar el texto nos preguntara si deseamos importar la API como una colección de Postman, seleccionamos dicha opción y hacemos click en “Import”



Y finalmente, en cuanto finalice el proceso de importación, se nos habrá creado una nueva colección en postman con la API.



# Implementación básica recomendada

A continuación se describe la implementación básica recomendada para el funcionamiento de la aplicación, y que podrá servir como base para la futura ampliación de la implementación.

Para cualquier trabajo sobre el desarrollo de las conexiones se recomienda trabajar inicialmente en el ambiente de desarrollo, para posteriormente migrar el desarrollo al ambiente de producción:

* Ambiente de desarrollo: <https://stage-api.naxsolutions.com/>
* Ambiente de producción: <https://beta-api.naxsolutions.com>

## Implementaciones básicas

Estas funciones básicas consisten principalmente en la actualización de datos en tiempo real esenciales para la funcionalidad de la plataforma. Las principales operaciones son las siguientes.

| Objetivo | Tabla | URL |
| --- | --- | --- |
| Actualizar datos de cultivo | Lots | <https://beta-api.naxsolutions.com/api/v1/lots/validate_csv>  <https://beta-api.naxsolutions.com/api/v1/lots/bulk_update_csv> |
| Subida de operaciones | Operations | <https://beta-api.naxsolutions.com/api/v1/operations/validate_csv>  <https://beta-api.naxsolutions.com/api/v1/operations/bulk_upload_csv> |
| Subida de resultados de muestreos inteligentes | Smart\_samplings | <https://beta-api.naxsolutions.com/api/v1/smart_samplings/samplings/validate_csv>  <https://beta-api.naxsolutions.com/api/v1/smart_samplings/samplings/bulk_upload_csv> |
| Subida de datos climáticos | Climate vez\_records | <https://beta-api.naxsolutions.com/api/v1/climate_stations/validate_csv>  <https://beta-api.naxsolutions.com/api/v1/climate_stations/bulk_upload_csv> |

### Obtener token de acceso

Sea cual sea la operación que queremos realizar, el primer paso siempre será común, y consistirá en obtener el token de acceso a la API. Podemos hacerlo de la siguiente manera.

| import json import requests   payload = json.dumps(  {  "password": "example",  "email": "example@example.com",  } )  headers = {  'Content-Type': 'application/json' }  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/token/pair",   headers=headers,   data=payload )  token = response.json()['access'] |
| --- |

A partir de ahora, para realizar cualquier petición debemos incluir el header de autorización en el que se incluirá el token que hemos obtenido. Vamos a probar por ejemplo a obtener nuestros datos de usuario.

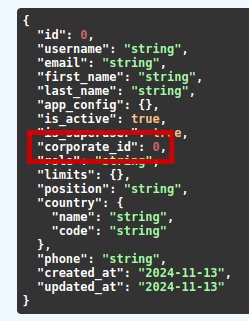
| import json import requests   headers = {  'Authorization': f'Bearer {token}', }  response = requests.request(  method="GET",  url="https://beta-api.naxsolutions.com/api/v1/users/me",  headers=headers, )  data = response.json() |
| --- |

### Obtener ids de entidades

Para poder trabajar con la API necesitaremos conocer los identificadores únicos (id) que se corresponden con las entidades que nos pertenecen. Podemos obtener dichos id siguiendo una serie de pasos que se detallan a continuación.

Primero, debemos conocer el id de la corporate a la que pertenecemos, para ello podemos hacer uso del siguiente endpoint.



A continuación, podemos ver la respuesta donde se indica el id de nuestra corporate 

Una vez hayamos obtenido nuestro corporate id podemos obtener los ids de las companies que posea nuestra corporate.



Este endpoint nos devolverá una lista con todas las companies que nos perteneces, donde podremos encontrar sus respectivos ids.



Ahora podemos conocer los ids de las areas que pertenecen a una company, para ello debemos hacer uso del siguiente endpoint   
  
El que, indicando el id de la companie de la que deseamos obtener las areas, nos devolverá la lista de areas que pertenecen a esa company indicando su id



Necesitaremos además para poder realizar las subidas de datos que se explican en los siguientes apartados obtener los ids de los tableros de smart\_samplings y de las estaciones climatológicas. Podemos obtenerlos de igual manera a la explicada anteriormente en los siguientes endpoints respectivamente.





## Pasos a seguir

### Actualizar datos de cultivo

El primer paso para subir nuestro archivo CSV a la plataforma es generarlo, y para ello recomendamos utilizar la librería pandas.

Una vez generado el archivo CSV, necesitamos obtener un token de acceso, tal como se explicó previamente, para poder cargarlo en la plataforma. Antes de hacer la carga definitiva de los datos, es aconsejable validar el archivo CSV, de modo que podamos corregir cualquier error que contenga. Las filas con datos erróneos serán omitidas durante el proceso de carga final.

En el payload se enviarán los siguientes atributos:

| **Nombre** | **Descripción** | **Obligatorio** |
| --- | --- | --- |
| company\_id | Id de la compañía | Si |
| campaign\_name | Nombre de la campaña que se va a actualizar | Si |
| area\_ids | Lista de ids de las áreas a actualizar, si no se envia se actualizarán todas las áreas de la campaña | No |
| upload\_field | Además, se permite cargar datos de cultivo en función de las unidades de agrupación; para ello, utilizaremos el atributo "upload\_field" para seleccionar el campo a emplear en la carga de datos. Por defecto se elige el atributo “name” | No |
| delimiter | Delimitador del archivo CSV, por defecto se elige “,” | No |
| columns | En el payload de la solicitud, el atributo "columns" permite especificar qué columnas deben actualizarse. Puedes encontrar ejemplos de cómo funciona este proceso en el “Anexo I: Ejemplos de carga de CSV”. | Si |

| from io import StringIO import json import requests   # df = El csv que hemos creado  headers = {  'Authorization': f'Bearer {token}', }  payload = {  "company\_id": company\_id,  "campaign\_name": campaign\_name,  "area\_ids": area\_ids,  "upload\_field": "name",  "delimiter": ",",  "columns": columns, }  csv\_file = StringIO() df.to\_csv(csv\_file, index=False, sep=",") csv\_file.seek(0)  files = {  'file': ('data.csv', csv\_file, 'text/csv') }  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/lots/validate\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

En la respuesta obtendremos un resumen de las filas que contienen errores o warnings, así como de las filas que son válidas.

Una vez nuestro archivo CSV no contenga errores, realizaremos la subida definitiva de los datos.

| response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/lots/bulk\_upload\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

### Atributos datos de cultivo **Atributos generales de Datos del cultivo**

| **Nombre atributo** | **Descripción** | **Tipo de dato** | ***Nombre anterior*** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| area\_name | Nombre del área a la que pertenece el polígono | Texto | - | Si |
| name | Nombre o identificador de cada polígono. | Texto | *id* | Depende de la unidad de carga de datos seleccionada. |
| start\_date | Fecha de inicio del ciclo, puede ser por siembra, o por corte (en el caso de la caña de azúcar). En esta fecha se comienza a medir el polígono. | Fecha (aaaa-mm-dd) | *fs* | No |
| harvest\_date | Fecha de cosecha o fin del ciclo. En esta fecha se deja de medir el polígono. | Fecha (aaaa-mm-dd) | *fc* | No |
| variety | Variedad del cultivo. | Texto | *variedad* | No |
| crop | Cultivo | Texto | *cultivo* | No |
| irrigation | Tipo de riego, tal cual está especificado en las notas a pie de tabla. \* | Numérico | *riego* | No |
| crop\_yield | Rendimiento normalizado por hectárea (ej kg/ha, toneladas/ha). En caña de azúcar el TCH. | Decimal | *rend* | No |
| eco\_zone | Zona agroecológica, tal cual está especificado actualmente en el manual de gestión de datos. | Numérico | *zona\_eco* | No |
| effective\_area | Área efectiva de cultivo | Decimal | *area\_ef* | No |
| water\_capacity | Capacidad total de campo, en milímetros de agua. | Decimal | *agua\_disp* | No |
| permited\_water\_deficit | Déficit hídrico permitido, o valor umbral de agua del cual no se quiere bajar. | Decimal | *-* | No |
| critical\_water | Punto de marchitez permanente, en milímetros de agua. | Decimal | *agua\_aprov* | No |
| seedbed | Definir si el cultivo se destina a semillero. | Booleano (True/False) | *semillero* | No |
| estimated\_harvest\_date | Fecha de cosecha estimada durante el ciclo (planificación de cosecha). | Fecha (aaaa-mm-dd) | *fc\_est* | No |
| estimated\_crop\_yield | Rendimiento estimado del cultivo. | Decimal | *rend\_est* | No |
| harverst\_type | Tipo de cosecha: ‘mechanical’ o ‘manual’ | Texto | *-* | No |
| sowing\_distance | Distancia entre surcos | Decimal | *-* | No |

\*Tipos de riego: Este atributo nos determina el tipo de riego, se deberá poner un número según la siguiente lista, en caso de no encontrar el tipo ponga el más similar: 0 = Sin riego; 1 = Con riego; 2 = Goteo; 3 = Gravedad; 4 = Mecanizado; 5 = Pivote; 6 = Pivote móvil; 7 = Frontal 1; 8 = Frontal 2; 9 = Aspersión.

También hay 10 atributos destinados a utilizarse como unidades de agrupación.

#### Unidades de agrupación en Datos del cultivo

| **Nombre atributo** | **Descripción** | **Tipo de dato** | **Obligatorio** |
| --- | --- | --- | --- |
| grouping\_unit\_01 | Unidad de agrupación 1 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_02 | Unidad de agrupación 2 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_03 | Unidad de agrupación 3 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_04 | Unidad de agrupación 4 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_05 | Unidad de agrupación 5 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_06 | Unidad de agrupación 6 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_07 | Unidad de agrupación 7 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_08 | Unidad de agrupación 8 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_09 | Unidad de agrupación 9 | Texto | Depende de la unidad de carga de datos seleccionada. |
| grouping\_unit\_10 | Unidad de agrupación 10 | Texto | Depende de la unidad de carga de datos seleccionada. |

Por último, además de los atributos generales anteriores, dependiendo del cultivo con el que se trabaje se podrán subir otros atributos adicionales

#### Atributos específicos de caña de azúcar

| **Nombre atributo** | **Descripción** | **Tipo de dato** | ***Nombre anterior*** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| soca | Ciclo del cultivo. En la caña de azúcar, el número de cortes, siendo 0 las nuevas siembras. | Numérico | *soca* | No |
| tsh | En la caña de azúcar, toneladas de azúcar por hectárea. | Decimal | *tah* | No |
| harvest\_burning | Cosecha con o sin quema | Booleano (True/False) | *-* | No |

#### Atributos específicos de arroz

| **Nombre atributo** | **Descripción** | **Tipo de dato** | ***Nombre anterior*** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| start\_flooding\_date | En arroz, fecha en la que la parcela se considera inundada | Fecha (aaaa-mm-dd) | *-* | No |
| flooding\_date | En arroz, fecha de comienzo de inundación | Fecha (aaaa-mm-dd) | *-* | No |
| emergence\_date | En arroz, fecha de emergencia | Fecha (aaaa-mm-dd) | *-* | No |

### Subida de operaciones

El primer paso será construir nuestro csv para posteriormente subirlo a la plataforma. Para ello recomendamos el uso de la librería pandas.

Una vez tengamos el csv debemos subirlo a la plataforma, y para ello debemos obtener primero nuestro token de acceso como se ha explicado anteriormente.

Se recomienda primero validar el archivo CSV, de forma que podamos subsanar los errores que contenga antes de subir los datos definitivamente. Las filas que contengan datos erróneos serán ignoradas a la hora de subir definitivamente los cambios. A continuación se muestra un ejemplo de como subir un archivo CSV.   
  
En la subida de datos de operaciones siempre es obligatorio subir todas las columnas.

Además, la subida de datos siempre se realizará en base al atributo “name” del lote en el que se realizó la operación

En el payload se enviarán los siguientes atributos:

| **Nombre** | **Descripción** | **Obligatorio** |
| --- | --- | --- |
| campaign\_name | Nombre de la campaña que se va a actualizar | Si |
| area\_ids | Lista de ids de las áreas a actualizar, si no se envia se actualizarán todas las áreas de la campaña | No |
| delimiter | Delimitador del archivo CSV, por defecto se elige “,” | No |

| from io import StringIO import json import requests   # df = El csv que hemos creado  headers = {  'Authorization': f'Bearer {token}', }  payload = {  "campaign\_name": campaign\_name,  "area\_ids": area\_ids,  "delimiter": ",", }  csv\_file = StringIO() df.to\_csv(csv\_file, index=False, sep=",") csv\_file.seek(0)  files = {  'file': ('data.csv', csv\_file, 'text/csv') }  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/operations/validate\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

En la respuesta obtendremos un resumen de las filas que contienen errores o warnings, así como de las filas que son válidas.

Una vez nuestro archivo CSV no contenga errores, realizaremos la subida definitiva de los datos.

| response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/operations/bulk\_upload\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

### Atributos operaciones

#### Atributos generales de Operaciones

| **Nombre atributo** | **Descripción** | **Tipo de dato** | **Nombre anterior** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| lot\_name | Nombre o identificador de cada polígono. | Texto | id | Si |
| area\_name | Nombre del área a la que pertenece el polígono | Texto | - | Si |
| start\_date | Fecha de inicio de operación | Fecha (aaaa-mm-dd) | - | Si |
| end\_date | Fecha de fin de operación | Fecha (aaaa-mm-dd) | - | No |
| type | Tipo de operación: Seleccionar entre: irrigation, fertilization, maturing\_agent, potassium, fungicides, insecticides, herbicides, resowing, others | Texto |  | Si |
| application\_type | Tipo de aplicación. **Para operaciones de tipo ‘fertilization’, seleccionar entre: ‘foliar’, ‘al suelo’ o ‘fraccionamiento’** | Texto |  | No |
| amount | Cantidad aplicada | Decimal | cantidad | No |
| measurement\_unit | Unidad de medida para la cantidad | Texto | - | Solo cuando se indica un valor en el campo ‘amount’ |
| product | Producto aplicado | Texto | producto | No |
| hectares | Hectáreas aplicadas (en caso de ser todo el polígono dejar vacío) | Decimal | area | No |
| identifier | Identificador propio de la operación | Texto | codigo | No |

### Subida de resultados de muestreos inteligentes

El primer paso será crear un muestreo, para ello necesitamos primero obtener el id del tablero en el que queremos crear el muestreo. Podemos obtener dichos ids haciendo uso de la siguiente petición



Este endpoint nos devolverá una lista de donde podremos obtener el id.

Se debe obtener también el id del tile en el que se desean realizar los samplings. lo podemos obtener mediante la siguiente petición.



Una vez obtenido el tile id podemos proceder a generar los marcadores para el sampling, que serán los puntos donde se realicen las muestras, y que deberemos utilizar también a la hora de crear el sampling.



Cuando se hayan obtenido los datos necesarios podemos proceder a crear el sampling propiamente dicho, para lo que haremos uso de la siguiente petición.



Una vez tengamos nuestro sampling creado, el siguiente paso será construir nuestro csv para posteriormente subirlo a la plataforma. Para ello recomendamos el uso de la librería pandas.

Una vez tengamos el csv debemos subirlo a la plataforma, y para ello debemos obtener primero nuestro token de acceso como se ha explicado anteriormente.

Se recomienda primero validar el archivo CSV, de forma que podamos subsanar los errores que contenga antes de subir los datos definitivamente. Las filas que contengan datos erróneos serán ignoradas a la hora de subir definitivamente los cambios. A continuación se muestra un ejemplo de como subir un archivo CSV.

La subida de datos siempre se realizará en base al nombre del marcador al que corresponden los resultados de esa fila.

En el payload se enviarán los siguientes atributos:

| **Nombre** | **Descripción** | **Obligatorio** |
| --- | --- | --- |
| board\_id | Id del tablero | Si |
| company\_id | Id de la compañía | Si |
| delimiter | Delimitador del archivo CSV, por defecto se elige “,” | No |
| columns | En el payload de la solicitud, el atributo "columns" permite especificar qué columnas deben actualizarse. Puedes encontrar ejemplos de cómo funciona este proceso en el “Anexo I: Ejemplos de carga de CSV”. | Si |

| from io import StringIO import json import requests   # df = El csv que hemos creado  headers = {  'Authorization': f'Bearer {token}', }  payload = {  "company\_id": company\_id,  "board\_id": board\_id,  "delimiter": ",",  "columns": columns, }  csv\_file = StringIO() df.to\_csv(csv\_file, index=False, sep=",") csv\_file.seek(0)  files = {  'file': ('data.csv', csv\_file, 'text/csv') }  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/smart\_samplings/validate\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

En la respuesta obtendremos un resumen de las filas que contienen errores o warnings, así como de las filas que son válidas.

Una vez nuestro archivo CSV no contenga errores, realizaremos la subida definitiva de los datos.

| response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/smart\_samplings/bulk\_upload\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

### Atributos muestreos inteligentes

#### Atributos generales de muestreos inteligentes

| **Nombre atributo** | **Descripción** | **Tipo de dato** | **Nombre anterior** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| name | Nombre o identificador de cada marcador. | Texto | - | Si |
| brix | Porcentaje de sólidos aparentes que contiene una solución de azúcar, según se determina por el uso del hidrómetro Brix. | Decimal | - | No |
| pol | la cantidad aparente de SACAROSA (porcentaje en masa) presente en una solución de azúcares. | Decimal | - | No |
| sucrose | Valor determinado por polarización del peso normal de un producto azucarado, usando un sacarímetro con escala internacional. El término pol es usado como una entidad real para todos los efectos de cálculo. | Decimal | - | No |
| reducing\_sugar | Porcentaje de azúcar e invertidos presentes en la sección 8-10 en el jugo de caña. | Decimal | - | No |
| humidity | Contenido de agua y de aquellas sustancias susceptibles de ser eliminadas con el agua por el procedimiento de secado. | Decimal | - | No |
| purity | Porcentaje que expresa la calidad del jugo analizado. | Decimal | - | No |
| weight | Cantidad de jugo extraído (peso) de una muestra de caña picada, por medio de un equipo de alta presión, expresado en porcentaje. | Decimal | - | No |
| trash | Contenido de materiales indeseables (masa) adheridos o mezclados con los tallos o trozos de caña de azúcar. | Decimal | - | No |
| fiber | Materia insoluble en agua. | Decimal | - | No |
| identifier | identificador | Texto | - | No |

### subida de datos climáticos

El primer paso será construir nuestro csv para posteriormente subirlo a la plataforma. Para ello recomendamos el uso de la librería pandas.

Una vez tengamos el csv debemos subirlo a la plataforma, y para ello debemos obtener primero nuestro token de acceso como se ha explicado anteriormente.

Se recomienda primero validar el archivo CSV, de forma que podamos subsanar los errores que contenga antes de subir los datos definitivamente. Las filas que contengan datos erróneos serán ignoradas a la hora de subir definitivamente los cambios. A continuación se muestra un ejemplo de como subir un archivo CSV.

La subida de datos siempre se realizará en base al nombre del marcador al que corresponden los resultados de esa fila.

En el payload se enviarán los siguientes atributos:

| **Nombre** | **Descripción** | **Obligatorio** |
| --- | --- | --- |
| company\_id | Id de la compañía | Si |
| climate\_station\_ids | Lista de ids de las estaciones climáticas a actualizar | Si |
| delimiter | Delimitador del archivo CSV, por defecto se elige “,” | No |
| columns | En el payload de la solicitud, el atributo "columns" permite especificar qué columnas deben actualizarse. Puedes encontrar ejemplos de cómo funciona este proceso en el “Anexo I: Ejemplos de carga de CSV”. | Si |

| from io import StringIO import json import requests   # df = El csv que hemos creado  headers = {  'Authorization': f'Bearer {token}', }  payload = {  "company\_id": company\_id,  "climate\_station\_ids": climate\_station\_ids,  "delimiter": ",",  "columns": columns, }  csv\_file = StringIO() df.to\_csv(csv\_file, index=False, sep=",") csv\_file.seek(0)  files = {  'file': ('data.csv', csv\_file, 'text/csv') }  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/climate\_stations/validate\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

En la respuesta obtendremos un resumen de las filas que contienen errores o warnings, así como de las filas que son válidas.

Una vez nuestro archivo CSV no contenga errores, realizaremos la subida definitiva de los datos.

| response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/climate\_stations/bulk\_upload\_csv",   headers=headers,  data={  "payload": json.dumps(payload),  "file": files  },  files=files, ) |
| --- |

### 

### Atributos datos climáticos

#### Atributos generales de Registros climatológicos

| **Nombre atributo** | **Descripción** | **Tipo de dato** | **Nombre anterior** | **Obligatorio** |
| --- | --- | --- | --- | --- |
| climate\_station\_id | Identificador único de la estación climatológica. Visible en el apartado ‘Estaciones climatológicas’. | Texto | *id* | Si |
| timestamp | Fecha en la que se obtuvo la información (formato ‘AAAA-MM-DD’) | Fecha (aaaa-mm-dd) | *fecha* | Si |
| med\_temperature | Temperatura media diaria en ºC | Decimal | *tmed* | No |
| min\_temperature | Temperatura mínima diaria en ºC | Decimal | *tmin* | No |
| max\_temperature | Temperatura máxima diaria en ºC | Decimal | *tmax* | No |
| precipitation | Precipitación diaria en mm2 | Decimal | *precip* | No |
| radiation | Radiación diaria | Decimal | *radiation* | No |
| et0 | mm de agua evapotranspirados diarios | Decimal | *cantidad* | No |

# Uso de la API

## Peticiones básicas a la API NAX

A continuación, revisaremos las acciones fundamentales necesarias para interactuar con las entidades básicas de la API. Estas acciones cubren los aspectos esenciales como la creación, lectura, actualización y eliminación de datos (CRUD), permitiendo gestionar de manera eficiente la información almacenada en la plataforma. A través de estas operaciones, podremos manipular entidades clave como areas, campaigns y otras, asegurando un correcto flujo de datos y facilitando la integración con distintos servicios y aplicaciones.

### Corporate

#### GET

Si deseamos obtener los datos de una corporate en concreto podemos usar el endpoint para la petición GET de la entidad corporate.



#### PUT

Si queremos modificar los datos de una corporate podemos hacer uso de la petición PUT de la entidad corporate.



#### GET Users from company

Para obtener todos los usuarios que pertenecen a una company haremos uso de la siguiente petición GET.



### Company

#### GET

Si deseamos obtener los datos de una company podemos usar la petición GET de la entidad company



#### POST

Si queremos crear una nueva company debemos utilizar el endpoint habilitado para ello, se trata de una petición POST en el entidad corporate.



#### PUT

Para poder modificar una company debemos hacer uso de la siguiente petición PUT que nos permitirá modificar los atributos de la company



#### DELETE

Para eliminar una company utilizaremos la petición DELETE



#### GET areas from company

Si deseamos obtener todas las areas de una company debemos utilizar la siguiente petición GET.



#### GET lots from company

Si deseamos obtener todos los lots de una company debemos utilizar la siguiente petición GET.



#### GET plots from company

Si deseamos obtener todos los plots de una company debemos utilizar la siguiente petición.



### Area

#### GET

Para obtener la información referente a un area dispones de la siguiente petición GET.



#### POST

Si deseamos crear una nueva area debemos realizarlo desde la entidad companies, donde encontraremos una petición POST que nos permitirá crear una nueva area.



#### PATCH

Para modificar un area debemos hacer uso del endpoint que proporciona la petición PATCH para la entidad area.



#### DELETE

Disponemos de una petición DELETE para eliminar areas.



#### GET Campaigns from areas

En caso de querer obtener todos los campaigns que pertenecen a un mismo area, se proporciona la siguiente petición GET.



### Campaign

#### GET

Si deseamos acceder a los datos de un campaign disponemos del siguiente endpoint.



#### POST

Para crear un nuevo campaign se dispone de la siguiente petición POST.

#### 

#### PATCH

Para modificar un campaign existente se debe utilizar esta petición POST. En caso de modificar el end\_harvest\_date de un campaign debe introducirse un valor posterior al actual.



#### DELETE

Para eliminar un campaign se puede utilizar el endpoint que contiene la petición DELETE.



#### GET lots from campaign

Si se desean obtener todos los lots que pertenecen a un campaign se puede utilizar el

siguiente endpoint. 

#### GET plots from campaign

Se pueden obtener todos los plots que pertenecen a un campaign haciendo uso de la petición GET que se muestra a continuación

#### 

#### GET products from campaign

Si se quieren obtener los productos que pertenecen a un campaign, se dispone de la siguiente petición GET

#### 

### Lot

#### GET

Para obtener un lot se debe utilizar la siguiente petición GET.



#### PUT

En caso de querer modificar un lot se debe utilizar el siguiente endpoint. Posteriormente conoceremos como actualizar más de un lot mediante subida de csv.



#### DELETE

En caso de querer borrar un lot podemos hacerlo mediante la correspondiente petición DELETE.



#### Subida de lots

A la hora de realizar la subida de lots tenemos varias opciones. La primera de ellas consiste en realizar la subida de lots mediante un shapefile, que realizaremos desde el siguiente endpoint.



O bien también se permite la subida de lots mediante un geojson, para lo que existe la siguiente petición.



Este endpoint admite los siguientes flags:

* **review\_context** a False hace que solo se revisen los plots que se envían. Si se pone a True se revisan los que se envían con los que existen (se comprobarán intersecciones y coincidencias con los que ya están subidos previamente). Si *review\_context* no está a True, *wipe\_existing* y *replacing* no aplican.
* **wipe\_existing** a True ignora *replacing* y lo que hace es que se reemplazan las geometrías ya subidas por las nuevas. Todas las anteriores se desactivan y las que se suben pasan a ser las únicas activas.
* **replacing** a True hace que si alguna geometría que se sube, intersecta con alguna ya subida, la que estaba subida se desactiva quedando solamente la nueva (reemplaza las que se pisan).

Ambos endpoints devuelven un geojson validado y con correcciones. Se recomienda repetir la activación del flag “review\_context”, que añadirá una propiedad al geojson llamada “status”, y que nos permite conocer si un lot va a ser creado, ya existía o va a ser eliminado. Después de subsanar los errores podemos realizar la petición de validación de nuevo para asegurarnos, y una vez que los datos estén correctos realizamos la petición de subida de datos.



| **ATENCIÓN: si el geojson utilizado para la subida de datos contiene errores, las features con errores serán ignoradas. Solo se persistirán las features que estén completamente correctas.** |
| --- |

| **ATENCIÓN: El endpoint de subida de datos puede generar un error si se superan los límites establecidos por el contrato o los límites configurados a nivel de area o company.** |
| --- |

#### Actualización de lots por nombre

Existe la posibilidad de actualizar lotes mediante nombre, para ello podemos hacer uso del siguiente endpoint:



El primer paso para realizar la actualización será obtener nuestro token de acceso tal y como se indica en el apartado ***2.a.i*** de este documento. Una vez tengamos el token podemos realizar la petición como se muestra en el ejemplo:

| # Headers de la petición  headers = {  'Content-Type': 'application/json',  'Authorization': f'Bearer {token}', }  # Payload de la petición  payload = json.dumps(  {  "name": "Nombre de ejemplo", # Sustituir con el nombre del lote que se quiere actualizar  "variety": "Variedad de ejemplo", # Sustituir con la variedad del lote que se quiere actualizar  "company\_id": 0, # Sustituir con el id de la company a la que pertenece el lote  "campaign\_name": "Camapaña de ejemplo", # Sustituir con el nombre de la campaña a la que pertenece el lote  "area\_ids": [0], # Sustituir con los ids de las areas en las que se quieren actualizar lotes  } )  # Petición POST  response = requests.request(  method="POST",   url="https://beta-api.naxsolutions.com/api/v1/lots/update\_by\_name",   headers=headers,   data=payload )  # Respuesta de la petición  print(response.json()) # Respuesta de la petición, donde podemos comprobar si se ha ejecutado correctamente |
| --- |

#### Actualización de lots

Anteriormente hemos tratado la actualización de un único lot, pero también existe la posibilidad de actualizar más de un lot a la vez mediante la subida de un archivo CSV con los datos que se quieran actualizar.

Si no se desea comenzar la modificación de los lots desde cero, existe la opción de descargar una plantilla en formato CSV con los datos actuales de los lots. Esto facilita considerablemente la actualización y modificación de la información de manera más eficiente. Para obtener esta plantilla se debe hacer uso de la siguiente petición.



| **ATENCIÓN: al momento de descargar la plantilla, es importante tener en cuenta que solo se incluirá la primera instancia de aquellos lots que presenten un valor duplicado en el “upload field“ seleccionado al enviar la solicitud.** |
| --- |

Una vez tengamos nuestro CSV con los datos de los lots actualizados listo para subir procederemos primero a su validación, donde se nos informará de los posibles errores que contenga dicho archivo, mediante el uso de la siguiente petición.



En el Anexo I, se pueden observar algunos ejemplos que ilustran cómo la selección de columnas influye en la validación y subida de datos, dependiendo de las columnas incluidas en el archivo CSV. Estos ejemplos permiten comprender mejor el impacto de elegir determinadas columnas en el proceso de carga de datos.

La respuesta a esta petición nos indicará los lots que contienen datos inválidos. Después de subsanar los errores podemos realizar la petición de validación de nuevo para asegurarnos, y una vez que los datos estén correctos realizamos la petición de subida de datos.



| **ATENCIÓN: si el CSV utilizado para la subida de datos contiene errores, las filas con problemas serán ignoradas. Solo se guardarán los datos correspondientes a las filas que estén completamente correctas.** |
| --- |

#### Actualizar lots high frequency

Es posible modificar varios lots a alta frecuencia mediante la subida de un archivo CSV. Para facilitar este proceso, se ofrece la opción de descargar una plantilla con los datos actuales de los lots, permitiendo realizar las modificaciones necesarias a partir de esa base. Para ello podemos realizar la siguiente petición.



Una vez tengamos nuestro CSV con los lots que queremos añadir a alta frecuencia listo para subir procederemos primero a su validación, donde se nos informará de los posibles errores que contenga dicho archivo, mediante el uso de la siguiente petición.



La respuesta a esta petición nos indicará las filas que contienen datos inválidos. Después de subsanar los errores podemos realizar la petición de validación de nuevo para asegurarnos, y una vez que los datos estén correctos realizamos la petición de subida de datos.



| **ATENCIÓN: si el CSV utilizado para la subida de datos contiene errores, las filas con problemas serán ignoradas. Solo se guardarán los datos correspondientes a las filas que estén completamente correctas.** |
| --- |

#### Subida de operaciones de lots

También se ofrece la opción de subir operaciones para los lots mediante un archivo CSV que contenga los datos correspondientes. Para simplificar este proceso, es posible descargar una plantilla que facilite la carga de información. Para ello, se debe realizar la siguiente solicitud.



Una vez tengamos nuestro CSV con los datos de las operaciones actualizados listo para subir procederemos primero a su validación, donde se nos informará de los posibles errores que contenga dicho archivo, mediante el uso de la siguiente petición.



La respuesta a esta petición nos indicará las operaciones que contienen datos inválidos. Después de subsanar los errores podemos realizar la petición de validación de nuevo para asegurarnos, y una vez que los datos estén correctos realizamos la petición de subida de datos.



| **ATENCIÓN: si el CSV utilizado para la subida de datos contiene errores, las filas con problemas serán ignoradas. Solo se guardarán los datos correspondientes a las filas que estén completamente correctas.** |
| --- |

### Plot

#### GET curvas de un plot

Si queremos obtener las curvas óptimas de un plot podemos hacerlo mediante la siguiente petición GET.



#### GET stats de un plot

En caso de querer obtener los stats de un plot podemos hacerlo mediante la siguiente petición GET.



### Users

#### GET mis areas

En caso de querer conocer las areas a las que tenemos acceso así como sus ids podemos hacerlo mediante la siguiente petición.



#### GET mis datos

En caso de querer conocer nuestros datos en la aplicación o nuestra configuración de la aplicación podemos consultarla mediante el siguiente endpoint.



#### PUT mi configuración

Podemos modificar nuestra configuración actual en la aplicación mediante la siguiente petición PUT.



# Anexo I: Ejemplos de subida de CSV

A continuación, se presenta una tabla que ilustra algunos ejemplos de las opciones disponibles al subir un archivo CSV y seleccionar las columnas que deseamos actualizar. Esta funcionalidad permite una gestión más eficiente de los datos, ya que podemos elegir específicamente qué información modificar.

| **Campo de subida** | **Campos seleccionados** | **Campos en el CSV** | **Resultado esperado** |
| --- | --- | --- | --- |
| name | [variety] | [area\_name, name, variety] | Se actualiza “variety” en los lots en los que coincida el name. |
| name | [variety] | [area\_name, name, variety, irrigation] | Solo se actualiza la columna “variety” en los lots en los que coincida el name. Se ignora la columna “irrigation” en el CSV. |
| name | \* | \* | Se actualizan todos los campos de los lotes en los que coincida el name. |
| name | [variety, irrigation] | [variety] | Se actualiza el campo “variety” en los lots en los que coincida el name. Se actualiza la columna “irrigation” a “null” |
| name | \* | [area\_name, name, variety] | Se actualizan todos los campos de los lotes en los que coincida el name. El campo “variety” se actualiza con los valores del CSV y el resto de campos se actualizan a “null” |
| name | [grouping\_unit\_01] | [grouping\_unit\_01] | Se actualizan las “grouping\_unit\_01” de los lotes en los que coincida el name. |
| grouping\_unit\_01 | [variety] | [variety] | Se actualiza el campo “variety· de los lotes en los que coincida la “grouping\_unit\_01” |

***( “\*” equivale a todas las columnas)***

# Anexo II: Tabla de permisos por rol

## Admin

| **Entidad** | **Crear** | **Leer** | **Actualizar** |
| --- | --- | --- | --- |
| area | ✓ | ✓ | ✓ |
| background\_task | ✓ | ✓ | ✓ |
| board | ✓ | ✓ | ✓ |
| campaign | ✓ | ✓ | ✓ |
| climate\_station | ✓ | ✓ | ✓ |
| climate\_station\_record | ✓ | ✓ | ✓ |
| company | ✓ | ✓ | ✓ |
| contract | ✓ | ✓ | ✓ |
| corporate | ✓ | ✓ | ✓ |
| curve | ✓ | ✓ | ✓ |
| dashboard | ✓ | ✓ | ✓ |
| invitation | ✓ | ✓ | ✓ |
| laboratory | ✓ | ✓ | ✓ |
| legend | ✓ | ✓ | ✓ |
| lot | ✓ | ✓ | ✓ |
| map\_layers | ✓ | ✓ | ✓ |
| marker | ✓ | ✓ | ✓ |
| note | ✓ | ✓ | ✓ |
| operation | ✓ | ✓ | ✓ |
| plot | ✓ | ✓ | ✓ |
| smart\_samplings\_board | ✓ | ✓ | ✓ |
| smart\_samplings\_marker | ✓ | ✓ | ✓ |
| smart\_samplings\_sampling | ✓ | ✓ | ✓ |
| stat | ✓ | ✓ | ✓ |
| task | ✓ | ✓ | ✓ |
| task\_label | ✓ | ✓ | ✓ |
| task\_submission | ✓ | ✓ | ✓ |
| tile | ✓ | ✓ | ✓ |
| user | ✓ | ✓ | ✓ |
| variable\_map | ✓ | ✓ | ✓ |

## Basic

| **Entidad** | **Crear** | **Leer** | **Actualizar** |
| --- | --- | --- | --- |
| area | ✓ | ✓ | ✓ |
| background\_task | ✓ | ✓ | ✓ |
| board | ✓ | ✓ | ✓ |
| campaign | ✓ | ✓ | ✓ |
| climate\_station | ✓ | ✓ | ✓ |
| climate\_station\_record | ✓ | ✓ | ✓ |
| company | ✓ | ✓ | ✓ |
| contract | ✓ | ✓ | ✓ |
| corporate | ✓ | ✓ | ✓ |
| curve | ✓ | ✓ | ✓ |
| dashboard | ✓ | ✓ | ✓ |
| invitation | ✓ | ✓ | ✓ |
| laboratory | ✓ | ✓ | ✓ |
| legend | ✓ | ✓ | ✓ |
| lot | ✓ | ✓ | ✓ |
| map\_layers | ✓ | ✓ | ✓ |
| marker | ✓ | ✓ | ✓ |
| note | ✓ | ✓ | ✓ |
| operation | ✓ | ✓ | ✓ |
| plot | ✓ | ✓ | ✓ |
| smart\_samplings\_board | ✓ | ✓ | ✓ |
| smart\_samplings\_marker | ✓ | ✓ | ✓ |
| smart\_samplings\_sampling | ✓ | ✓ | ✓ |
| stat | ✓ | ✓ | ✓ |
| task | ✓ | ✓ | ✓ |
| task\_label | ✓ | ✓ | ✓ |
| task\_submission | ✓ | ✓ | ✓ |
| tile | ✓ | ✓ | ✓ |
| user | ✓ | ✓ | ✓ |
| variable\_map | ✓ | ✓ | ✓ |