Guía paso a paso (Step by Step)

- Autenticación
- Parámetros de consulta global
- Reglas de filtrado
- Acceso a los elementos

Autenticación

Todos los datos dentro de la plataforma son privados de forma predeterminada. El rol público se puede configurar para exponer datos sin autenticación, o puede pasar un token de acceso a la API para acceder a datos privados.

Tokens de acceso

Hay tres tipos de tokens que se pueden usar para autenticarse dentro de Catalogo.

1. **El token temporal (JWT)** es devuelto por el punto de conexión/mutación de inicio de sesión. Estos tokens tienen un tiempo de caducidad relativamente corto y, por lo tanto, son la opción más segura de usar. Los tokens se devuelven con un token de actualización que se puede usar para recuperar un nuevo token de acceso a través del punto de conexión o mutación de actualización.
2. **El token de sesión (JWT)** también puede ser devuelto por el punto de conexión/mutación de inicio de sesión. Los tokens de sesión combinan un token de actualización y un token de acceso en una sola cookie. Estos tokens no deben tener un tiempo de caducidad corto como los tokens temporales, ya que no se pueden actualizar después de que hayan caducado.
3. **El token estático** se puede configurar para cada usuario de la plataforma y nunca caducan. Son menos seguros, pero bastante útiles para la comunicación de servidor a servidor.

Una vez que tenga su token de acceso, hay tres formas de pasarlo a la API: en el encabezado o header `Authorization` de la solicitud, como **cookie de sesión** o a través del parámetro `access_token` de consulta.

Encabezado de autorización

```
Authorization: Bearer <token>
```

Cookie de sesión

```
Cookie: directus_session_token=<token>
```

Parámetro de consulta

```
?access_token=<token>
```

> ADVERTENCIA
>
> La opción de parámetro de consulta no se recomienda en las configuraciones de producción, ya que varios sistemas pueden registrar los parámetros.

Iniciar sesión

Autenticarse como usuario.

Request

```
POST /auth/login
{
"email": user_email,
"password": user_password
}
```

Cuerpo de la solicitud

- `email`: Dirección de correo electrónico requerida del usuario.
- `password`: Contraseña requerida del usuario.
- `otp`: La contraseña de un solo uso del usuario (si MFA está habilitada).
- `mode`: Si se va a recuperar el token de actualización en la respuesta JSON o en una cookie `httpOnly`. Uno de `json`, `cookie` o `session`. El valor predeterminado es `json`.

Respuesta

- `access_token` **string**: Token de acceso temporal que se usará en las solicitudes de seguimiento. Nota: si lo usaste como modo en la solicitud, el token de acceso no se devolverá en el JSON.session
- `expires` **integer**: Cuánto tiempo pasará antes de que caduque el token de acceso. El valor se expresa en milisegundos.
- `refresh_token` **string**: El token que se puede usar para recuperar un nuevo token de acceso a través de /auth/refresh. Nota: si usaste `cookie` o `session` como el modo en la solicitud, el token de actualización no se devolverá en el JSON.

>

> Tiempo de caducidad
>
> El tiempo de caducidad del token se puede configurar a través de la variable de entorno ACCESS_TOKEN_TTL.

Ejemplo

```
POST /auth/login
{
"email": "admin@example.com",
"password": "c4t4l0g0"
}
```

Actualizar token

Recupere un nuevo token de acceso mediante un token de actualización.

Request

```
POST /auth/refresh
{
"refresh_token": refresh_token_string,
"mode": refresh_mode
}
```

Cuerpo de la solicitud

- `refresh_token`: El token de actualización que se va a usar. Si tiene el token de actualización en una cookie a través de `/auth/login`, no es necesario que lo envíe aquí.
- `mode`: Si se debe enviar y recuperar el token de actualización en la respuesta JSON o en una cookie `httpOnly`. Uno de `json`, `cookie` o `session`.

Respuesta

- `access_token` **string**: Token de acceso temporal que se usará en las solicitudes de seguimiento. Nota: si lo usaste como modo en la solicitud, el token de acceso no se devolverá en el JSON.session
- `expires` **integer**: Cuánto tiempo pasará antes de que caduque el token de acceso. El valor se expresa en milisegundos.
- `refresh_token` **string**: El token que se puede usar para recuperar un nuevo token de acceso a través de `/auth/refresh`. Nota: si usaste `cookie` o `session` como el modo en la

solicitud, el token de actualización no se devolverá en el JSON.

Ejemplo

```
POST /auth/refresh
{
"refresh_token": "gmPd...8wuB",
"mode": "json"
}
```

Cerrar sesión

Invalide el token de actualización, destruyendo así la sesión del usuario.

Request

```
POST /auth/logout
{
"refresh_token": refresh_token
}
```

Cuerpo de la solicitud

- `refresh_token`: El token de actualización que se va a invalidar. Si tiene el token de actualización en una cookie a través de /auth/login, no es necesario que lo envíe aquí.
- `mode`: Si el token de actualización se envía en la respuesta JSON o en una cookie `httpOnly`. Uno de `json`, `cookie` o `session`.

Ejemplo

```
POST /auth/logout
{
"refresh_token": "gmPd...8wuB",
"mode": "json"
}
```

Solicitar restablecimiento de contraseña

Solicite que se envíe un correo electrónico de restablecimiento de contraseña al usuario determinado.

Request

```
POST /auth/password/request
{
"email": user_email
}
```

Cuerpo de la solicitud

- `email` **Requerido**: Dirección de correo electrónico del usuario para el que solicitas el restablecimiento de contraseña.
- `reset_url`: Proporcione una URL de restablecimiento personalizada a la que le llevará el enlace del correo electrónico. El token de restablecimiento se pasará como parámetro.

Ejemplo

```
POST /auth/password/request
{
"email": "admin@example.com"
}
```

Restablecer una contraseña

La solicitud de un punto de conexión de restablecimiento de contraseña envía un correo electrónico con un vínculo a la aplicación de administración (o a una ruta personalizada) que, a su vez, usa este punto de conexión para permitir que el usuario restablezca su contraseña.

Request

```
POST /auth/password/reset
{
"token": password_reset_token,
"password": password
}
```

Cuerpo de la solicitud

- `token` **Requerido**: Token de restablecimiento de contraseña, tal y como se proporciona en el correo electrónico enviado por el punto de conexión de la solicitud.

- `password` **Requerido**: Nueva contraseña para el usuario.

Ejemplo

```
POST /auth/password/reset
{
"token": "eyJh...KmUk",
"password": "c4t4l0g0"
}
```

Parámetros de consulta global

La mayoría de las operaciones de punto de conexión de la API de Catalogo se pueden manipular con los siguientes parámetros. Es importante entenderlos para sacar el máximo partido a la plataforma.

Campos

Elija los campos que se devuelven en el conjunto de datos actual. Este parámetro admite la notación de puntos para solicitar campos relacionales anidados. También puede utilizar un comodín (*) para incluir todos los campos a una profundidad específica.

```
GET /items/services
    ?fields=name,description,status_id.name
```

Ejemplos

Campo	Descripción
`name,description`	Devuelve solo los campos `name` y `description`.
`name,status_id.name`	Devuelve `name` y el elemento `name` relacionado con `status_id`
`*`	Devuelve todos los campos.
`*.*`	Devuelve todos los campos de primer y segundo nivel relacionado.
`*,institution_id.*`	Devuelve todos los campos y todos los campos de `institution_id`.

> **Rendimiento y tamaño**
>
> Aunque el comodín `fields` es muy útil para fines de depuración, se recomienda solicitar solo campos específicos para su uso en producción. Al solicitar solo los campos que realmente necesita, puede acelerar la solicitud y reducir el tamaño total de la salida.

Relaciones

En el **Catálogo de Servicios** algunos campos de la colección `services` almacenan datos relacionados con múltiples colecciones mediante campos relacionales o tablas intermedias.

Para solicitar **campos específicos** de cada relación, utiliza la sintaxis:

```
?fields=<relacion>.<subrelacion>.<campo>
```

Esto permite que, al consultar servicios, podamos incluir campos de colecciones relacionadas sin traer datos innecesarios.

Escenario

Sabiendo que tenemos la colección `services`, que tiene relaciones con otras colecciones:

- **status_id** → colección `status`
- **service_type_id** → colección `service_type`
- **addressed_to_id** → colección `addressed_to`

Queremos traer:

- Campos básicos del servicio
- Estado y tipo
- Público objetivo

Quedaría de esta manera:

```
GET /items/services
?fields[]=name
    &fields[]=status_id.name
    &fields[]=service_type_id.type
    &fields[]=addressed_to_id.other_entity
```

Filtro

Se utiliza para buscar elementos en una colección que coincida con las condiciones del filtro. El parámetro filter sigue la especificación Reglas de filtro, que incluye información adicional sobre operadores lógicos (AND/OR), filtrado relacional anidado y variables dinámicas.

Hay dos sintaxis disponibles:

```
GET /items/services
    ?filter[name][_eq]=Renovación pasaporte

// or

GET /items/services
    ?filter={ "name": { "_eq": "Renovación pasaporte" }}
```

Ejemplos

Recupera todos los elementos donde es igual a "Renovación pasaporte" `name`

```
{
  "name": {
    "_eq": "Renovación pasaporte"
  }
}
```

Recupere todos los elementos de uno de los siguientes estados: "Borrador", "Publicado"

```
{
  "status_id": {
    "_in": [1, 2]
  }
}
```

Recuperar todos los elementos que se publican entre dos fechas

```
{
  "date_created": {
    "_between": ["2021-01-24", "2021-02-23"]
  }
}
```

Recupere todos los elementos en los que la marca "`default_variation`" de una variación es verdadera

```
{
  "variation_ids": {
```

```
  "default_variation": {
    "_eq": true
  }
 }
}
```

> **Filtros anidados**
>
> En el ejemplo anterior se filtrarán los elementos de nivel superior en función de una condición del elemento relacionado. Si quieres filtrar los elementos relacionados, ¡echa un vistazo al parámetro deep!

Buscar

El parámetro de búsqueda permite realizar una búsqueda en todos los campos de tipo cadena y texto de una colección. Es una manera fácil de buscar un artículo sin crear filtros de campo complejos, aunque está mucho menos optimizado. Solo busca en los campos del elemento raíz, no se incluyen los campos de elementos relacionados.

Ejemplo

Buscar todos los servicios que mencionan `Pasaporte`

```
GET /items/services
    ?search=Pasaporte
```

Ordenar

La ordenación predeterminada es ascendente, pero se puede usar un signo menos `-` para revertir esto a un orden descendente. Los campos se priorizan según el orden del parámetro. La notación de puntos debe usarse cuando se ordena con valores de campos anidados.

Ejemplos

Campo	Descripción
`date_created`	Ordenar por fecha de creación descendente.

`sort,-date_updated`	Ordene por un campo de "ordenación", seguido de la fecha de actualización descendente.
`sort,-variation_ids.name`	Ordenar por un campo de "ordenación", seguido del nombre de una variación anidada.

```
GET /items/services
    ?sort=sort,-date_updated,author.name

// or

GET /items/services
    ?sort[]=sort
    &sort[]=-date_updated
    &sort[]=-variation_ids.name
```

Límite

Establezca el número máximo de artículos que se devolverán. El límite predeterminado se establece en `100`.

Ejemplos

- Consigue los primeros 200 artículos: `200`
- Obtener el número máximo permitido de artículos: `-1`

```
GET /items/services
    ?limit=200
```

> **Límites elevados y rendimiento**
>
> Dependiendo del tamaño de la colección, obtener la cantidad máxima de elementos puede provocar una disminución del rendimiento o tiempos de espera más largos. Úselo con precaución.

Omitir

Omita los primeros `n` elementos de la respuesta. Este parámetro se puede utilizar para la paginación.

Ejemplos

- Consigue los objetos 101—200

```
GET /items/services
    ?offset=100
```

Página

Una alternativa a `offset`. La página es una forma de establecer `offset` bajo el capó mediante el cálculo de `limit * page`. La primera página que está indexada es `1`.

Ejemplos

- Obtener artículos 101-200

```
GET /items/services
    ?page=2
```

Agregación y agrupación

Agregación

Las funciones de agregado o `aggregate` le permiten realizar cálculos en un conjunto de valores, devolviendo un único resultado.

Las siguientes funciones de agregación están disponibles en Catalogo:

Nombre	Descripción
`count`	Cuenta cuántos elementos hay
`countDistinct`	Cuenta cuántos objetos únicos hay
`sum`	Suma los valores del campo dado
`sumDistinct`	Suma los valores únicos en el campo dado
`avg`	Obtener el valor promedio del campo dado
`avgDistinct`	Obtener el valor promedio de los valores únicos en el campo dado

Nombre	Descripción
`min`	Devuelve el valor más bajo del campo
`max`	Devuelve el valor más alto del campo

```
GET /items/services
    ?aggregate[count]=*
```

Agrupación

De forma predeterminada, las funciones de agregación anteriores se ejecutan en todo el conjunto de datos. Para permitir informes más flexibles, puede combinar la agregación anterior con la agrupación. La agrupación permite ejecutar las funciones de agregación en función de un valor compartido. Esto permite cosas como "Calificación promedio por mes" o "Ventas totales de artículos en la categoría de jeans".

La consulta permite agrupar varios campos simultáneamente. En combinación con las funciones, esto permite la generación de informes agregados por año-mes-fecha.groupBy

```
GET /items/services
    ?aggregate[count]=name,description
    &groupBy[]=institution_id
    &groupBy[]=date_updated
```

Profundo

`deep` le permite establecer cualquiera de los otros parámetros de consulta (excepto `fields` y `deep` en sí) en un conjunto de datos relacional anidado.

Ejemplos

- Limite las variaciones relacionadas anidados a 3

```
{
□"variation_ids": {
□□"_limit": 3
□}
}
```

- Solo obtenga 3 artículos relacionados, con solo el comentario mejor calificado anidado

```
{
  "variation_ids": {
    "_limit": 3,
    "result_ids": {
      "_sort": "name",
      "_limit": 1
    }
  }
}
```

Hay dos sintaxis disponibles:

```
GET /items/services
    ?deep[institution_id][_filter][acronym][_eq]=OGTIC

// or

GET /items/services
    ?deep={ "institution_id": { "_filter": { "acronym": { "_eq": "OGTIC" }}}}
```

Alias

Los `alias` le permiten cambiar el nombre de los campos sobre la marcha y solicitar el mismo conjunto de datos anidados varias veces utilizando diferentes filtros.

> **Campos anidados**
>
> Solo es posible asignar alias a los campos del mismo nivel. El alias de los campos anidados, por ejemplo, `field.nested`, no funcionará.

```
GET /items/services
    ?alias[name]=nombre
    &alias[description]=descripcion
    &deep[institution_id][_filter][acronym][_eq]=OGTIC
```

Exportar

Guarde la respuesta del API actual en un archivo. Acepta un tipo de los siguientes: `csv`, `json`, `xml`, `yaml`.

```
GET /items/services
    ?export=type
```

Ejemplos:

```
?export=csv
?export=json
?export=xml
?export=yaml
```

Funciones

Las funciones permiten la modificación "en vivo" de los valores almacenados en un campo. Las funciones se pueden usar en cualquier parámetro de consulta que normalmente proporcionaría una clave de campo, incluidos los campos, la agregación y el filtro.

Las funciones se pueden usar envolviendo la clave de campo en una sintaxis similar a JavaScript, por ejemplo:

- `timestamp` -> `year(timestamp)`

Funciones de fecha y hora

Filtro	Descripción
`year`	Extraer el año de un campo de fecha y hora/fecha/marca de tiempo
`month`	Extraer el mes de un campo datetime/date/timestamp
`week`	Extraer la semana de un campo datetime/date/timestamp
`day`	Extraer el día de un campo de fecha y hora/fecha/marca de tiempo
`weekday`	Extraer el día de la semana de un campo datetime/date/timestamp

Filtro	Descripción
`hour`	Extraer la hora de un campo datetime/date/timestamp
`minute`	Extraer el minuto de un campo de fecha y hora/fecha/marca de tiempo
`second`	Extraiga el segundo de un campo de fecha y hora/fecha/marca de tiempo

Funciones de matriz

Filtro	Descripción
`count`	Extraer el número de elementos de una matriz JSON o un campo relacional

```
GET /items/services
    ?fields=id,name,weekday(date_created)
    &filter[year(date_created)][_eq]=2024
```

Reglas de filtrado

Los permisos, la validación y el parámetro filter de la API se basan en una estructura JSON específica para definir sus reglas. En esta página se describe la sintaxis para crear reglas de filtro planas, relacionales o complejas.

Sintaxis

- Campo: cualquier campo raíz, campo relacional u operador lógico válido
- Operador: cualquier operador de filtro válido
- Valor: cualquier valor estático válido o variable dinámica

```
{
  <field>: {
    <operator>: <value>
  }
}
```

Ejemplos

```
{
  "title": {
    "_contains": "Catalogo"
  }
}
```

```
{
  "owner": {
    "_eq": "$CURRENT_USER"
  }
}
```

```
{
  "datetime": {
    "_lte": "$NOW"
  }
}
```

```
{
  "category": {
```

```
□□"_null": true
□}
}
```

Operadores de filtro

Nombre del operador (en la aplicación)	Operador	Descripción
Igual a	`_eq`	Igual a
No es igual a	`_neq`	No es igual a
Menor que	`_lt`	Menor que
Menor o igual que	`_lte`	Menor o igual que
Mayor que	`_gt`	Mayor que
Mayor o igual que	`_gte`	Mayor o igual que
Es uno de los	`_in`	Coincide con cualquiera de los valores
No es uno de los	`_nin`	No coincide con ninguno de los valores
Es nulo	`_null`	Es nulo `null`
No es nulo	`_nnull`	No es nulo `null`
Contiene	`_contains`	Contiene la subcadena
Contiene (no distingue entre mayúsculas y minúsculas)	`_icontains`	Contiene la subcadena que no distingue entre mayúsculas y minúsculas
No contiene	`_ncontains`	No contiene la subcadena
Comienza con	`_starts_with`	Comienza con
Comienza con (sin distinción entre mayúsculas y minúsculas)	`_istarts_with`	Comienza con, sin distinción entre mayúsculas y minúsculas
No comienza con	`_nstarts_with`	No comienza con
No comienza con (sin distinción entre mayúsculas y minúsculas)	`_nistants with`	No comienza con, sin distinción entre mayúsculas y minúsculas
Termina con	`_ends_with`	Termina con
Termina con (sin distinción entre mayúsculas y minúsculas)	`_iends_with`	Termina con, sin distinción entre mayúsculas y minúsculas
No termina con	`_nends_with`	No termina con
No termina con (sin distinción entre mayúsculas y minúsculas)	`_niends_with`	No termina con, sin distinción entre mayúsculas y minúsculas
Está entre	`_between`	El valor interseca un punto dado

Nombre del operador (en la aplicación)	Operador	Descripción
No está entre	`_nbetween`	El valor no interseca un punto dado
Está vacío	`_empty`	Está vacío (`null` o falso)
No está vacío	`_nempty`	No está vacío (`null` o falso)

Relacional

Puede apuntar a valores relacionados anidando nombres de campo. Por ejemplo, si tiene un campo relacional Many-to-One, puede establecer una regla para el campo mediante la siguiente sintaxis.authorauthor.name

```
{
□"author": {
□□"name": {
□□□"_eq": "Rijk van Zanten"
□□}
□}
}
```

Cuando se utilizan relaciones M2M, se creará una tabla de uniones y el filtro se aplicará a la propia tabla de uniones. Por ejemplo, si tiene una colección, con una relación M2M con los autores de cada libro, es probable que la colección de cruces tenga un nombre y 3 campos: y . Para filtrar libros específicos en función de sus autores, debe pasar por la tabla de uniones y el campo :books books_authorsid books_id authors_id authors_id

```
{
□"authors": {
□□"authors_id": {
□□□"name": {
□□□□"_eq": "Rijk van Zanten"
□□□}
□□}
□}
}
```

Operadores lógicos

Puede anidar o agrupar varias reglas mediante los operadores lógicos o. Cada operador lógico contiene una matriz de reglas de filtro, lo que permite un filtrado más complejo. Tenga en cuenta también en el ejemplo que los operadores lógicos se pueden subanidar en operadores lógicos. Sin embargo, no se pueden subanidar en reglas de filtro._and_or

```
{
  "_or": [
    {
      "_and": [
        {
          "user_created": {
            "_eq": "$CURRENT_USER"
          }
        },
        {
          "status": {
            "_in": ["published", "draft"]
          }
        }
      ]
    },
    {
      "_and": [
        {
          "user_created": {
            "_neq": "$CURRENT_USER"
          }
        },
        {
          "status": {
            "_in": ["published"]
          }
        }
      ]
    }
  ]
}
```

Algunos vs Ninguno en uno a muchos

Al aplicar filtros a un campo de uno a varios, la plataforma usará de forma predeterminada una búsqueda de "algunos", por ejemplo, en:

```
{
  "categories": {
    "name": {
      "_eq": "Recipe"
    }
  }
}
```

El elemento primario de nivel superior se devolverá si una de las categorías tiene el nombre . Este comportamiento se puede invalidar mediante el uso de los operadores y explícitos, por ejemplo: Recipe_some_none

```
{
  "categories": {
    "_none": {
      "name": {
        "_eq": "Recipe"
      }
    }
  }
}
```

obtendrá todos los elementos principales que no tengan la categoría "Receta".

Acceso a los elementos

Los elementos son datos individuales de la base de datos. Pueden ser cualquier cosa, desde artículos hasta comprobaciones de estado de IoT.

El objeto Item

Los elementos tienen un esquema predefinido. El formato depende completamente de los datos y estructuras definidas.

> Datos relacionales
>
> De forma predeterminada, el objeto item no contiene datos relacionales anidados. Para recuperar datos anidados, consulte Datos relacionales y parámetros de campo.

Obtener items

Enumera todos los elementos que existen en la colección definida.

Request

```
GET /items/:collection


SEARCH /items/:collection
```

Si utiliza SEARCH, puede proporcionar un objeto de consulta como cuerpo de la solicitud. Más información sobre SEARCH.

Parámetros de consulta

Admite todos los parámetros de consulta globales.

>

> Datos relacionales
>
> El parámetro Field es necesario para devolver datos relacionales anidados.

Respuesta

Matriz de objetos de elemento hasta límite. Si no hay elementos disponibles, los datos serán una matriz vacía.

Ejemplo

```
SEARCH /items/services
```

Obtener item por ID

Obtén un elemento que exista en la colección definida.

Request

```
GET /items/:collection/:id
```

Parámetros de consulta

Admite todos los parámetros de consulta globales.

Respuesta

Devuelve un objeto de elemento si se proporcionó una clave principal válida.

Ejemplo

```
GET /items/services/15
```