Bienvenido al API CRUD de eventos para la prueba técnica de SIER

Requisitos

Antes de iniciar el programa recuerde instalar la dependencia flask de python
Ejecute el archivo main.py y consuma desde POSTMAN

Donde:

Tipo de entrada/output	**Descripción**
id	Es el parámetro que diferencia los eventos, es un identificador único, numérico y mayor a 0. Nota: cuando se desea obtener ciertos event se puede indicar el número en los params, la cadena “all” para retornar todos los eventos.
name_event	Es un parámetro de entrada de event el cual es una cadena alfanumérica se recomienda poner un nombre descriptivo para el evento.
type_event	Es un parámetro de entrada de event y event_gestion el cual es una cadena alfanumérica.
date_event	Es un parámetro de entrada de evento el cual es una cadena que simboliza la fecha, DD-MM-YYYY.
status_event	Es un parámetro de entrada de event el cual simboliza una transición de estados al crear un evento se debe de ingresar obligatoriamente en el estado: “`PendingRevision`” y luego de ser creada puede ser modificada a “Revised” es sensible a mayúsculas.
200	código de estado que refleja toda transacción exitosa.
401	código de estado que refleja toda transacción no exitosa.
402	código de estado que refleja un error en la solicitud por parte del usuario.

Endpoints

Para insertar nuevos eventos:

La solicitud post a /event creará un nuevo event y retornará un código de estado más un mensaje que indicará cómo fue procesada la solicitud por el servidor.

Dicha solicitud tiene que ir acompañada de un archivo Json el cual contiene los parámetros de creación.

Ejemplo de Json:

```
{
    "name_event" : "test 01",
    "type_event" : "event A",
    "description" : "this is a test",
    "date_event" : "10-02-2023",
    "status_event" : "PendingRevision"
}
```

Al momento de crear el evento el servidor retorna un archivo Json informando del estado de la creación.

Ejemplo de creación exitosa:

```
{
    "message": "Event ID: 16 insert successfull",
    "status": 200
}
```

Ejemplo de creación no exitosa:

```
{
    "message": "Not found 'name_event',  Total errors: 1",
    "status": 401
}
```

Para obtener los Eventos:

la consulta get de /event retornara los resultados requeridos en los filtros enviados en los params, siempre retornara 1 archivo Json con la información correspondiente y los eventos indicados en los parámetros enviados.

El Json que retorna el servidor contiene 3 valores

1. data: un vector de eventos con toda la información requerida por el usuario: identificador, nombre, tipo, descripción, fecha.
2. message: Son metadatos que se obtienen a través de las consultas en el servidor.
3. satus: código de estado de la solicitud.

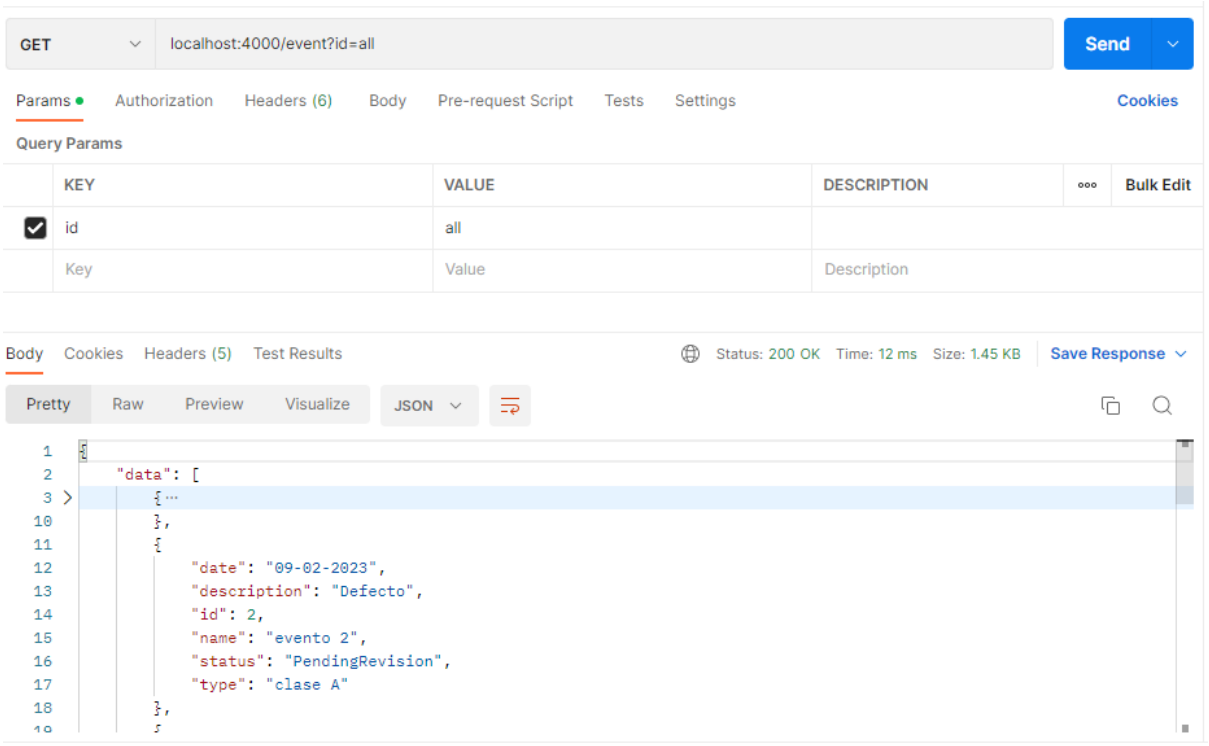
Filtros:

- id: es el parámetro a filtrar por identificador de evento.

Ejemplos:

Para obtener todos los event

url: localhost:4000/event
params:
id=all



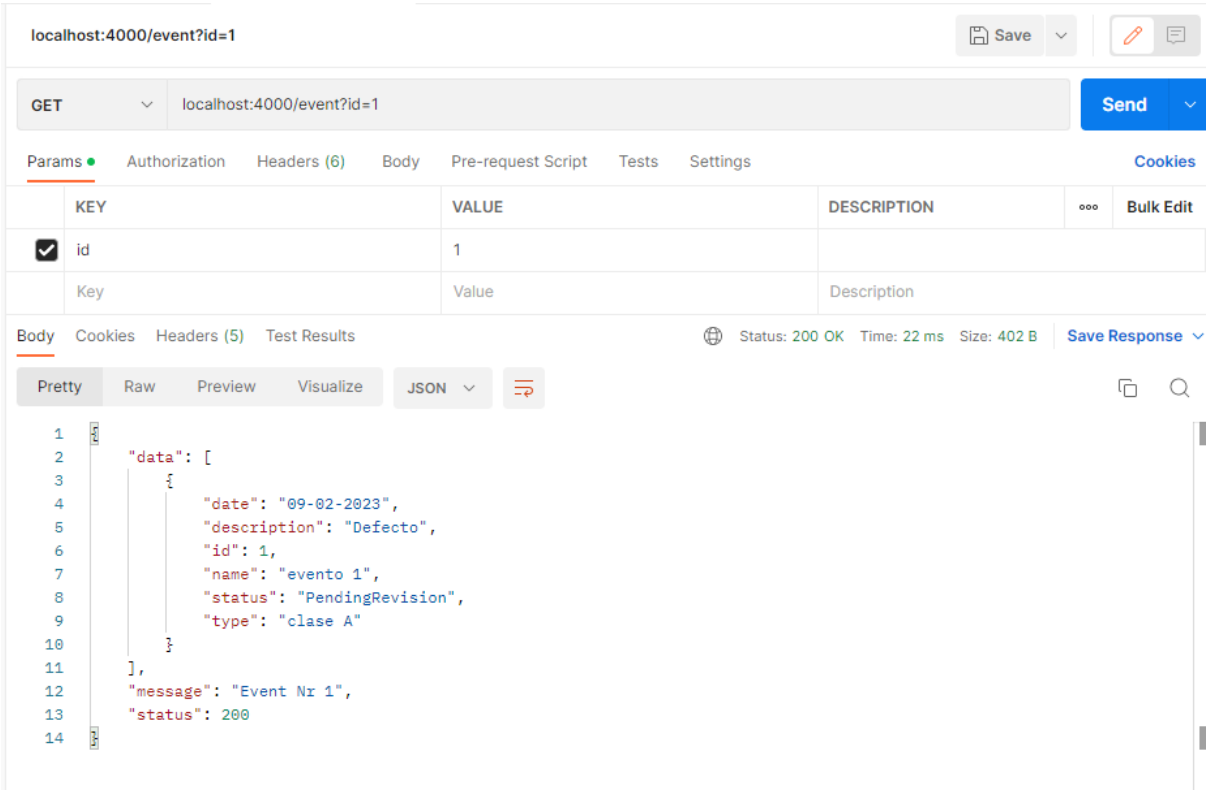
Resultado:


```
{
    "data": [
                …
            ],
    "message": "All information of events > total: 7",
    "status": 200
}
```


Para obtener el evento con identificador igual a 1:

url: localhost:4000/event
params:
id=1

Resultado:



Para obtener los eventos borrados:

localhost:4000/event?id=delete

Para obtener el contador total de eventos:

localhost:4000/event?id=delete

Para eliminar un evento:

La solicitud delete a /event eliminará el evento según su identificador, para ello debemos de enviar un archivo Json que contendrá el id del evento que se desea eliminar. Una vez enviada la petición el servidor responderá con un archivo Json que contiene la información acerca de la eliminación del evento.

El Json retornado es “status” donde se observará el código que depende de la solicitud y “message” que corresponde al mensaje que da el servidor.

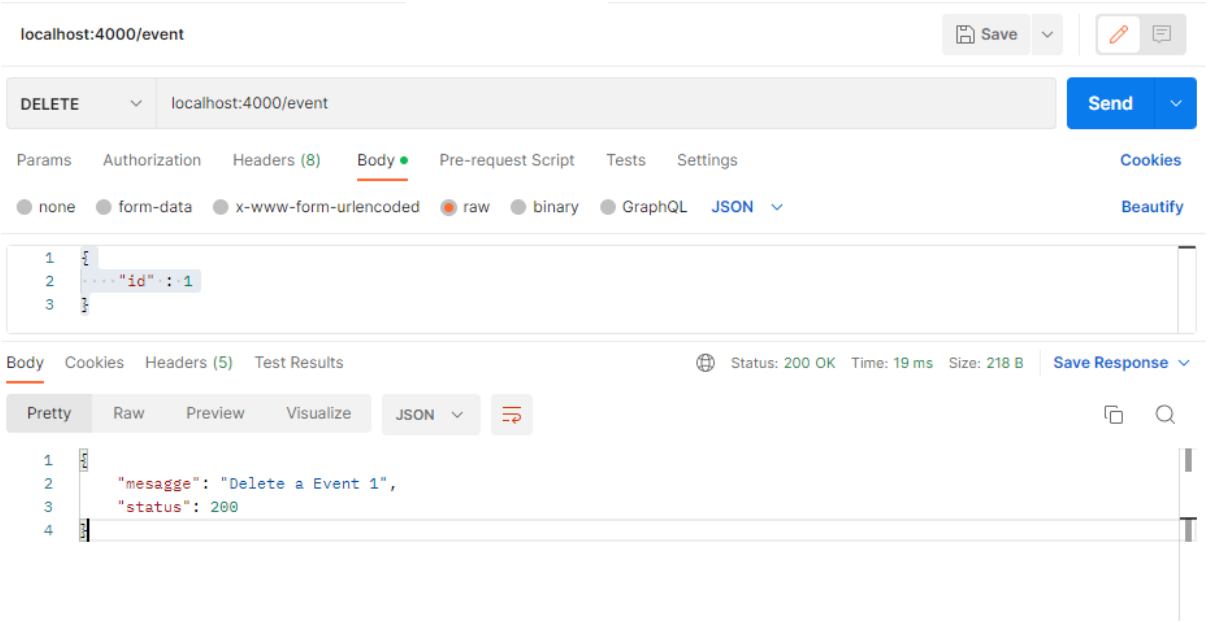
Por ejemplo:

Si usted desea eliminar el evento con identificador 1 basta con enviar el siguiente Json:

```
{
    "id" : 1
}
```

Y esto retornará el siguiente Json:

```
{
    "message": "Delete a Event 1",
    "status": 200
}
```



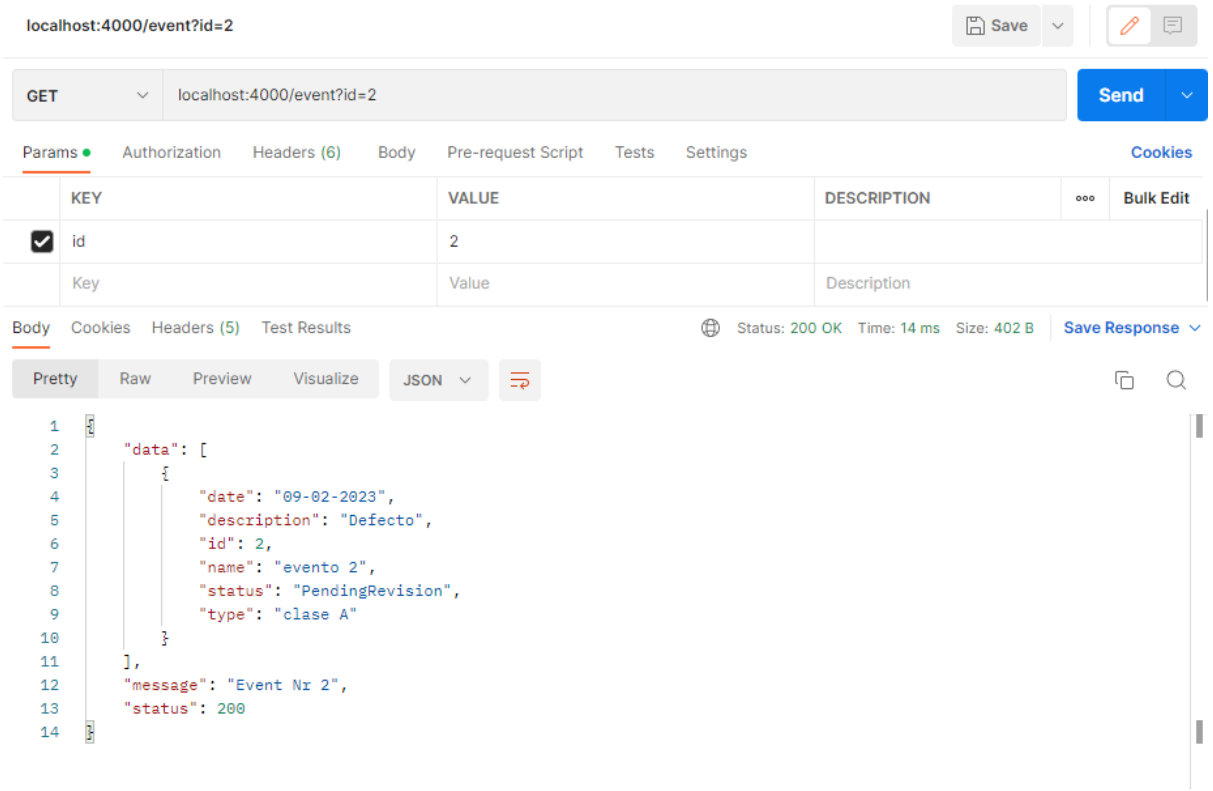
Para modificar un evento:

La solicitud patch a /event es capaz de modificar los atributos de event, para ellos los atributos que quieren ser modificados deben enviarse como Json. Nota: No se necesita enviar todos los parámetros sólo aquellos que el usuario esté interesado en modificar. Cuando se envíe la solicitud al servidor recibiremos un Json con la información correspondiente a la modificación.

El Json retornado contiene: “status” es el código de la solicitud, “message”: es el mensaje para el usuario.

Ejemplo:

Se tiene el evento 2



Nota: esta imagen es solo un GET y no tiene nada que ver con la modificación.

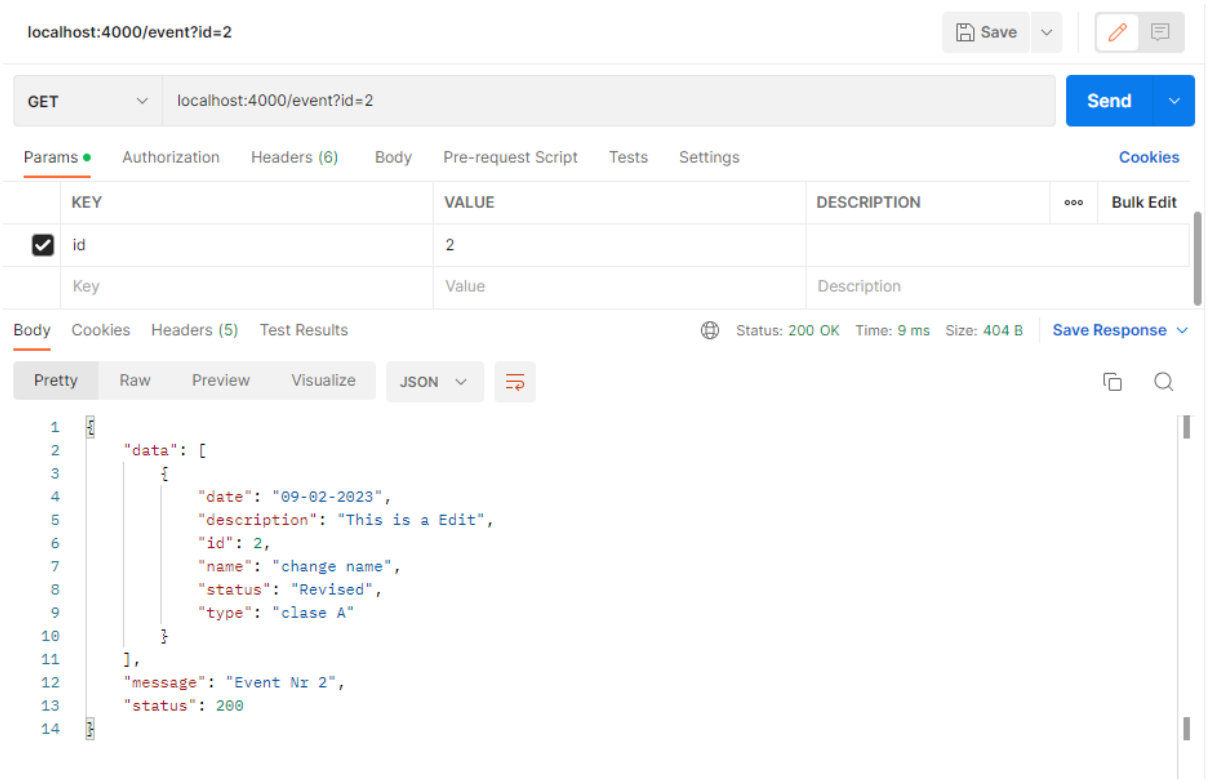
Dicho evento se desea modificar su nombre, descripción y estado entonces procederemos a enviar dichos parámetros a través de un Json de la siguiente manera:

```
{
    "id" : 2,
    "name_event" : "change name",
    "description" : "This is a Edit",
    "status_event" : "Revised"
}
```

Luego de ello el servidor retornará un Json con el estado de la modificación.

```
{
    "gestion": 1,
    "message": "Edit a Event 2, Note: Requires GESTION. ",
    "status": 200
}
```

Y así quedará nuestro evento despues de ser modificado:



Gestión de eventos

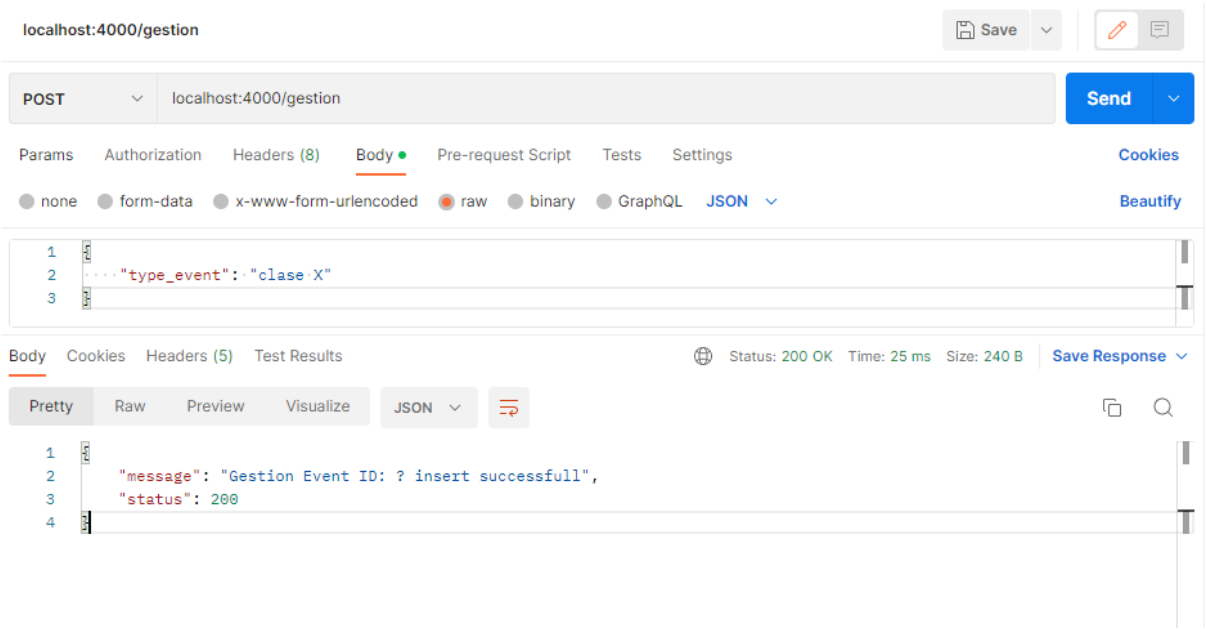
Los eventos pueden necesitar o no ser gestionados después de modificar su estatus, para determinar cuáles eventos necesitan gestión hay que crear un event_gestion a través de la solicitud post de /gestion y enviar un Json con el tipo de evento.

Ejemplo:

Todos los eventos de “clase X” requieren gestión se genera el siguiente Json y se envia:

```
{
    "type_event": "clase X"
}
```

luego el servidor responde con un estado de la inserción.


localhost:4000/gestion
Save
POST
localhost:4000/gestion
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "type_event": "clase X"
3 }
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 25 ms
Size: 240 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "message": "Gestion Event ID: ? insert successfull",
3 "status": 200
4 }