



Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas



Producto Integrador de Aprendizaje.

Mtro. Gustavo Adolfo Rodríguez Martínez.

Desarrollo Web: Back-end.

Nombre	Matrícula
Félix Yahveh Alanís Candelaria	1947480
Johann Joseph Velázquez Antonio	1969211

Licenciatura en Ciencias Computacionales.

Grupo: 037

San Nicolás de los Garza N.L a 23 de mayo del 2023.

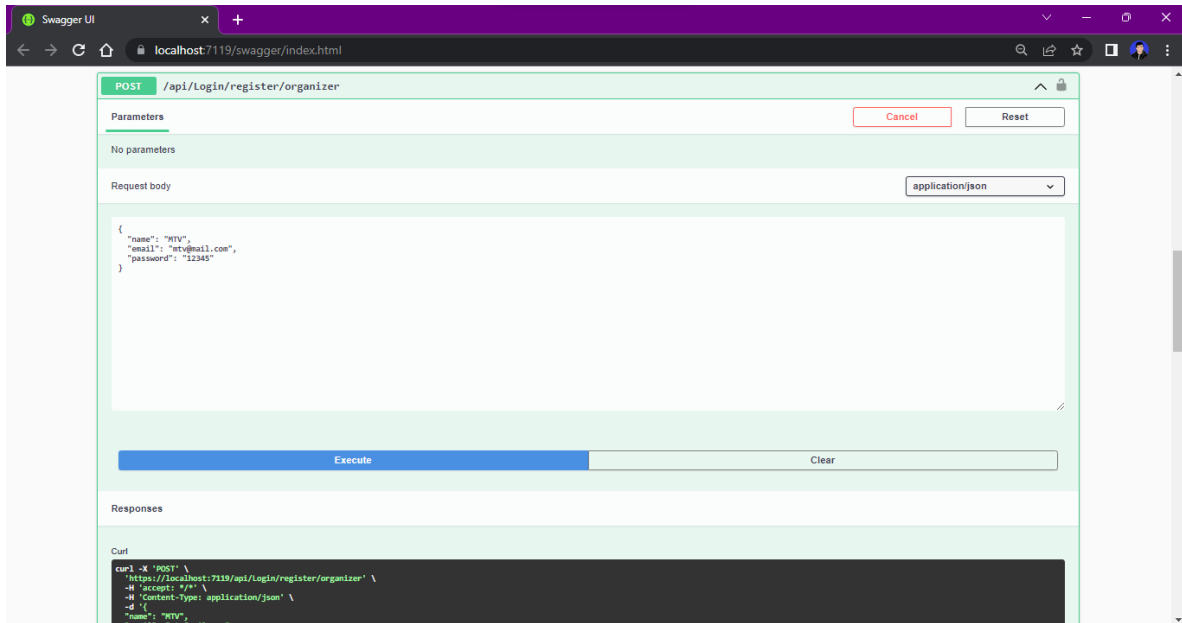
Contenido

Organizadores.....	3
Creación e inicio de sesión.	3
Eventos	7
Cupones.....	14
Usuarios.....	16
Creación e inicio de sesión.	16
Usuarios.....	19
Eventos	22
Asistencia a eventos	23
Comentarios.....	25
Repositorio de github.....	27
Diagrama entidad relación	28

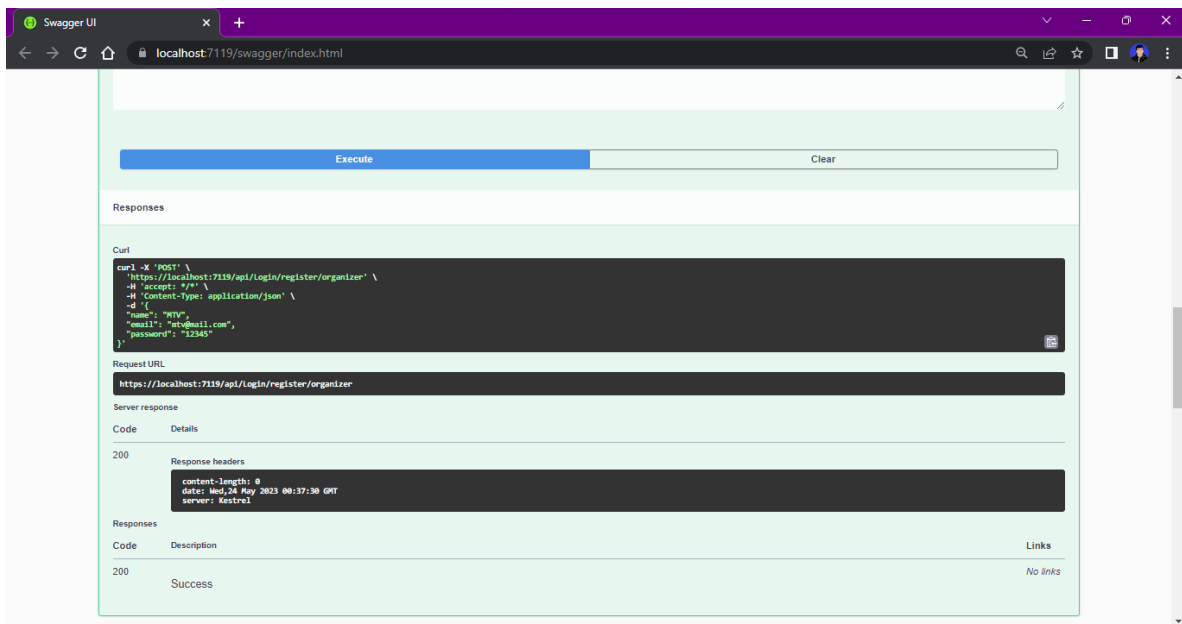
Organizadores

Creación e inicio de sesión.

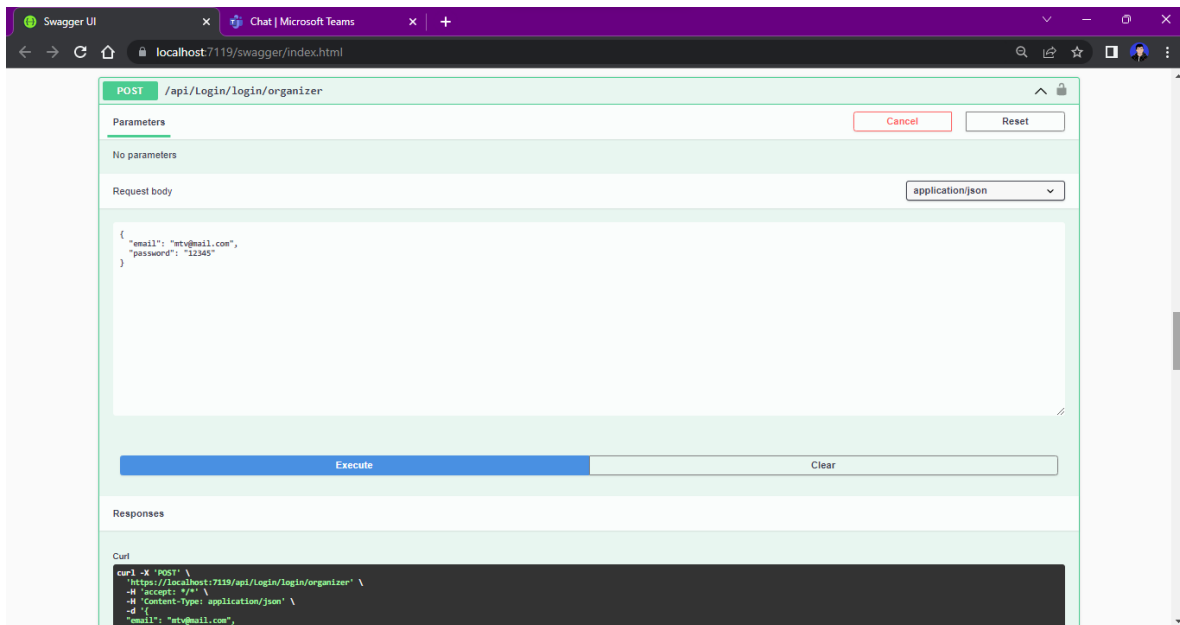
Se accede a la ruta '/api/Login/register/organizer' se ingresan los datos para la creación de un organizador, los datos requeridos son: nombre, correo electrónico (esta valida que la cadena sea un correo) y una contraseña



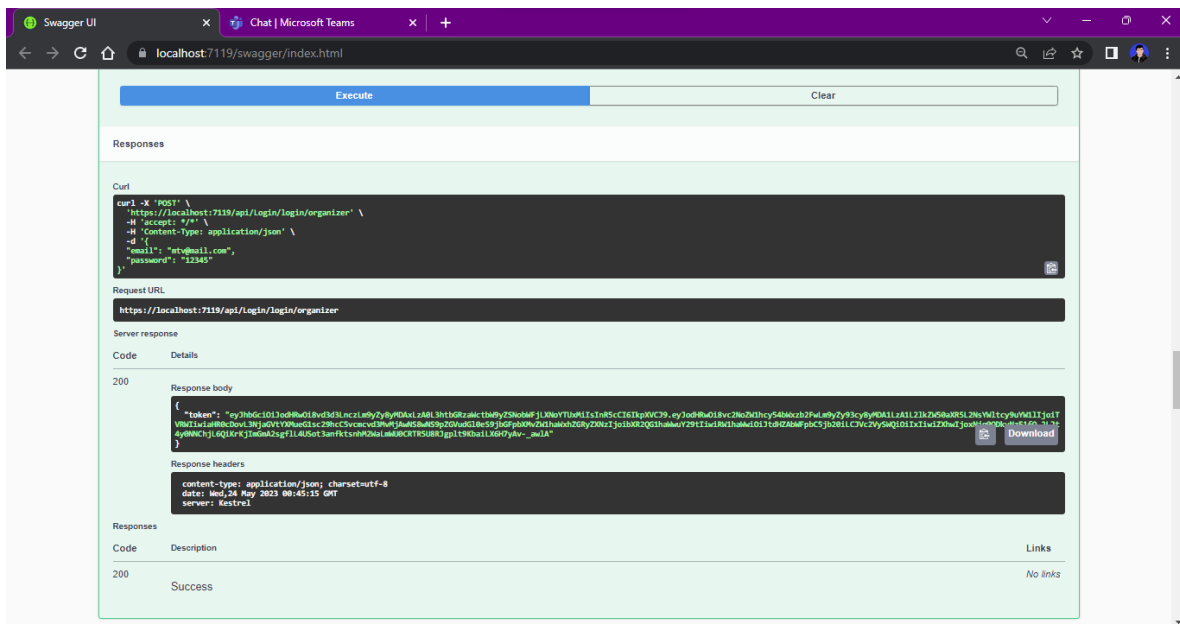
Aquí se muestra el resultado de la petición.



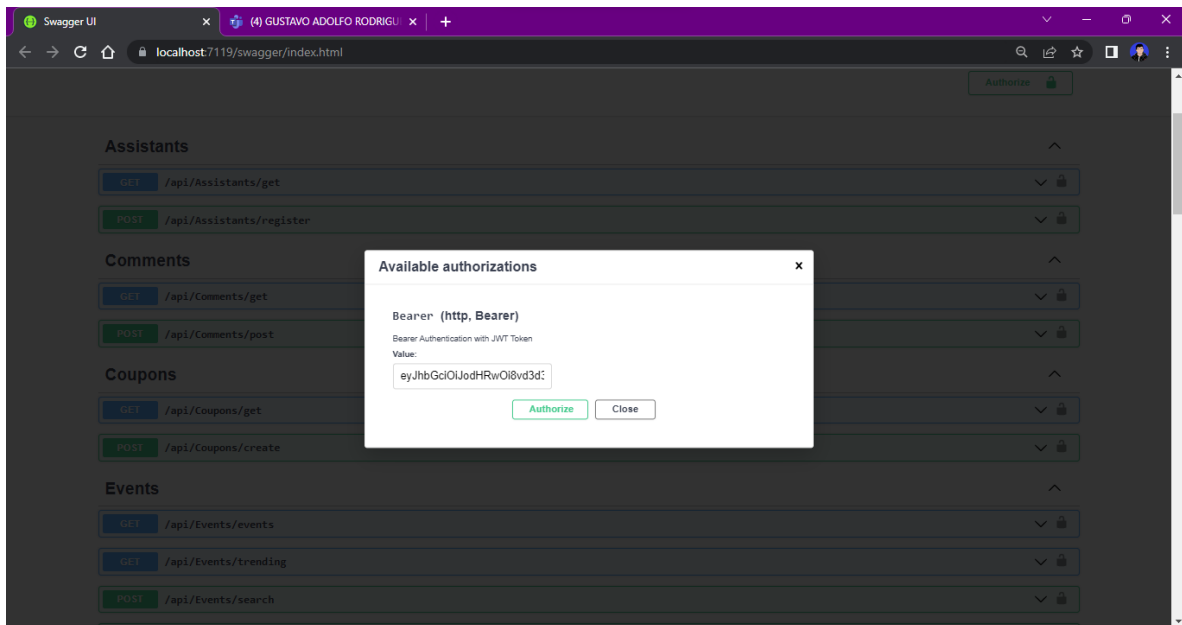
Ahora accediendo a la ruta `‘/api/Login/login/organizer’` por el método POST se puede obtener un token de inicio de sesión, los datos requeridos en esta petición son: correo electrónico (El sistema valida que la cadena ingresada sea un correo electrónico) y la contraseña



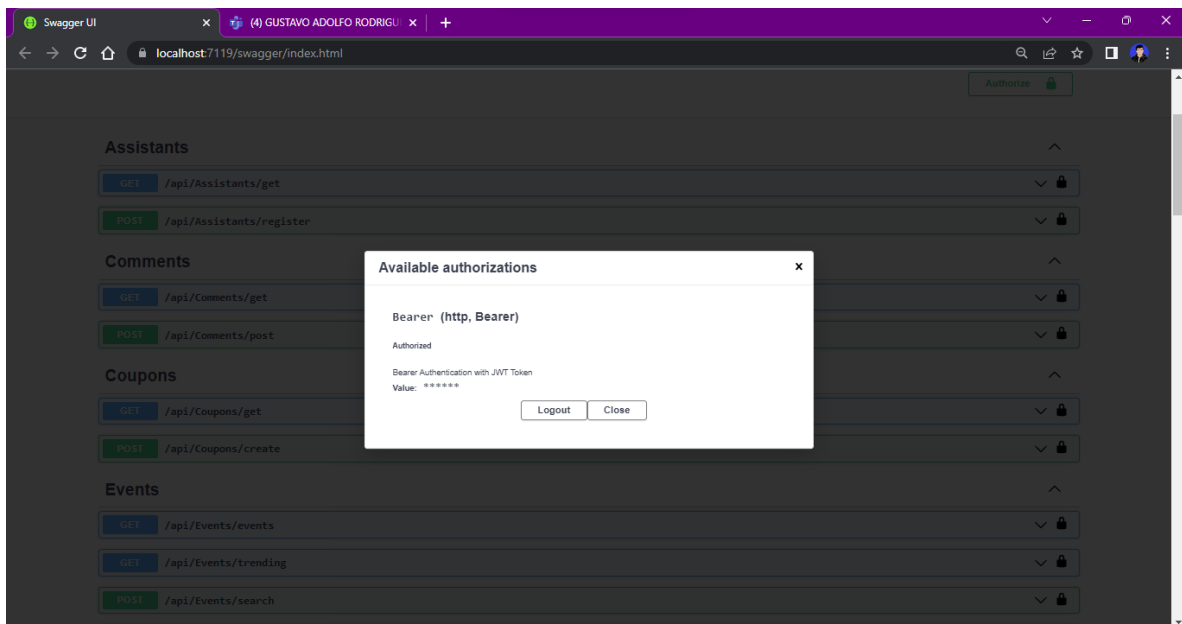
El resultado de esta petición es la obtención de un token



Al dar la autorización se solicita ingresar el token proporcionado por la petición de log in.



Cuando se ingresó y se dio click en el botón autorizar nos aparecerá un cuadro que nos indica que ya hemos sido autorizados.



Una vez autorizados podemos acceder a diferentes enlaces que los organizadores tienen disponible ver.

La ruta organizers 'get' nos da el listado de todos los organizados registrados en nuestro sistema.

The screenshot shows the Swagger UI interface for a REST API. The 'Responses' tab is selected, displaying the details for the 'get' endpoint. The request URL is 'https://localhost:7119/get'. The server response is a 200 status code. The response body is a JSON object representing an event organizer. The response headers include 'content-type: application/json; charset=utf-8', 'date: Wed, 24 May 2023 01:45:18 GMT', and 'server: Kestrel'.

```
curl -X 'GET' \
  'https://localhost:7119/get' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9'

{
  "id": 1,
  "name": "MTV",
  "events": [
    {
      "id": 1,
      "name": "Cafe Tacvba Rock Tour 2023",
      "description": "Concierto de cafe tacvba patrocinado por MTV",
      "date": "2023-05-30T00:00:00",
      "ubicacion": "Arena Monterrey",
      "capacity": 2000,
      "organizers": [
        {
          "id": 1,
          "name": "MTV",
          "type": "Organizador",
          "comment": ""
        }
      ]
    }
  ]
}
```

Code	Description	Links
200	Success	No links

La ruta 'getComments' nos dará un listado de todos los comentarios hechos a los organizadores.

The screenshot shows the Swagger UI interface for a REST API. The 'Responses' tab is selected, displaying the details for the 'get' endpoint of the 'api/Comments' resource. The request URL is 'https://localhost:7119/api/Comments/get'. The server response is a 200 status code. The response body is a JSON array of comment objects. The response headers include 'content-type: application/json; charset=utf-8', 'date: Wed, 24 May 2023 02:43:04 GMT', and 'server: Kestrel'.

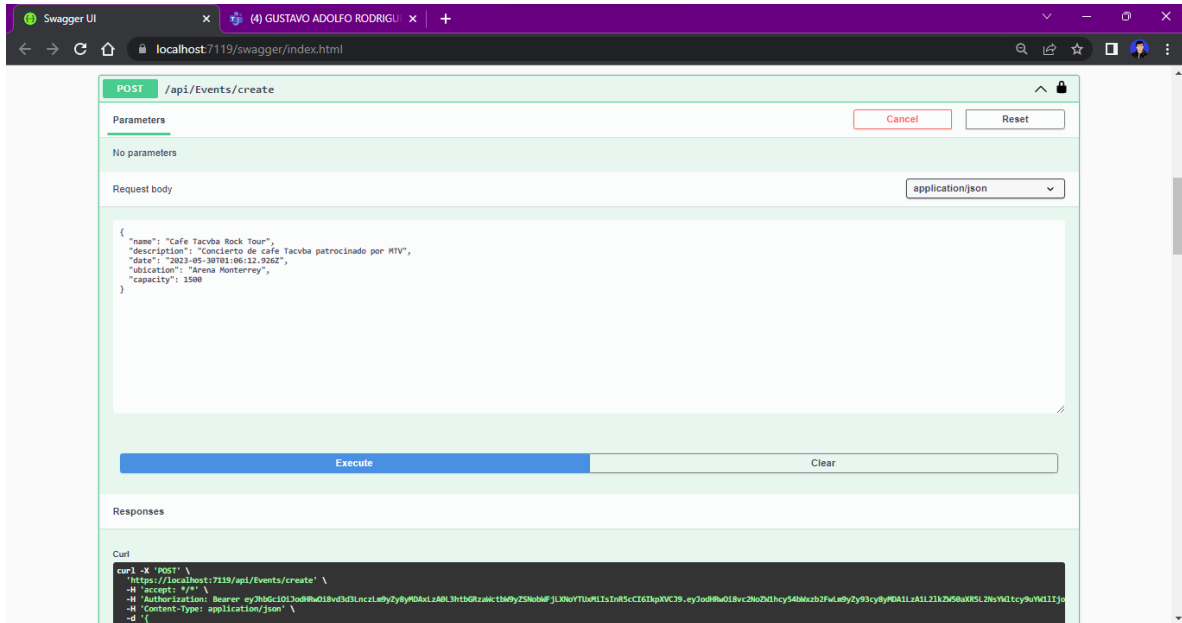
```
curl -X 'GET' \
  'https://localhost:7119/api/Comments/get' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9.eyJ3b290aW4iOiJkaW5kaWkiLCJ1aW50IjoiZm9udCJ9'

[
  {
    "name": "felix",
    "type": "Comentario",
    "comment": "Cafe tacvba es una excelente banda, que bueno que organicen eventos asi",
    "organizer": "MTV"
  },
  {
    "name": "felix",
    "type": "Pregunta",
    "comment": "¿Van a tener mas cupones para el concierto de daniel me estas matando?",
    "organizer": "MTV"
  }
]
```

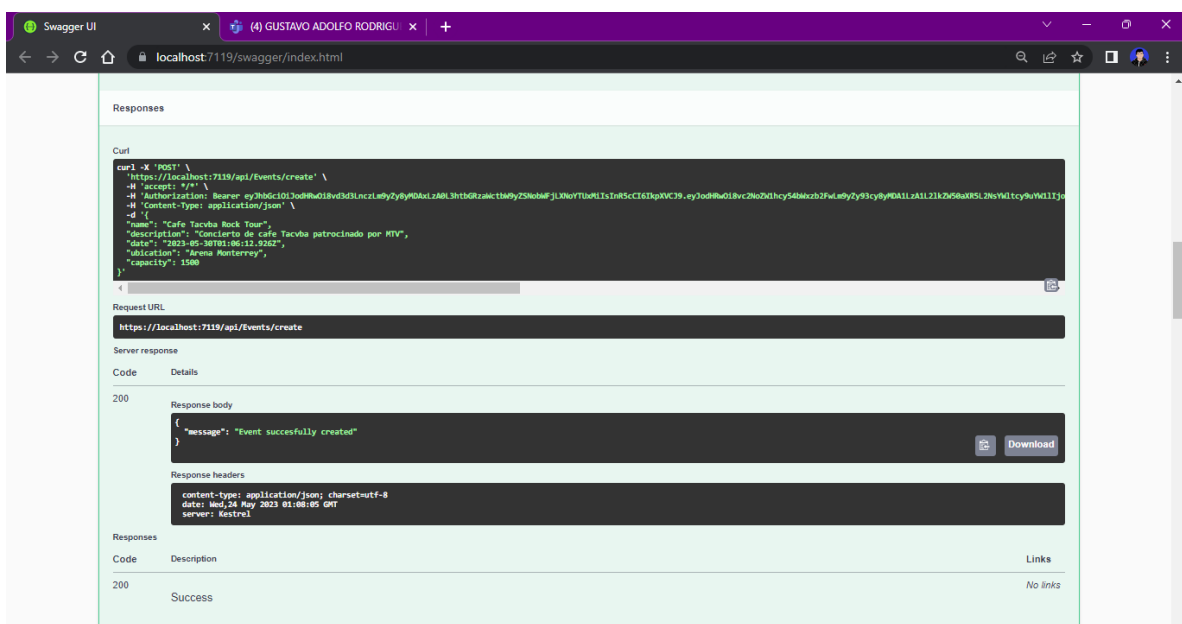
Code	Description	Links
200	Success	No links

Eventos

Accediendo a la ruta '/api/Events/Create' por el método POST podremos ingresar un nuevo evento. Esta ruta nos pide los datos: Nombre del evento, descripción, fecha (verifica que la fecha dada no sea una anterior a la fecha actual), ubicación del evento, capacidad de personas.



Resultado de la petición.



La ruta '/api/Events/update/{id}' por el verbo PUT actualiza un evento ya creado con anterioridad, este controlador verifica que el Id del evento exista, así como de su organizador, solicita los mismos datos que la ruta anterior.

The image shows the Swagger UI interface for the PUT endpoint '/api/Events/update/{id}'. The interface is divided into several sections:

- Parameters:** A table with columns 'Name' and 'Description'. It contains one parameter: 'id' (required), type 'integer(\$int32)', and path location. The value '1' is entered in the input field.
- Request body:** A text area containing a JSON object:

```
{  "id": 1,  "name": "Cafe Tacvba Rock Tour 2023",  "description": "Concierto de cafe tacvba patrocinado por MTV",  "date": "2023-05-30",  "ubicacion": "Arena Monterrey",  "capacidad": 2500,  "organizersId": 1}
```

 The media type is set to 'application/json'.
- Buttons:** 'Execute' and 'Clear' buttons are at the bottom.
- Responses:** An empty section for viewing the response.

Resultado de ejecutar la petición.

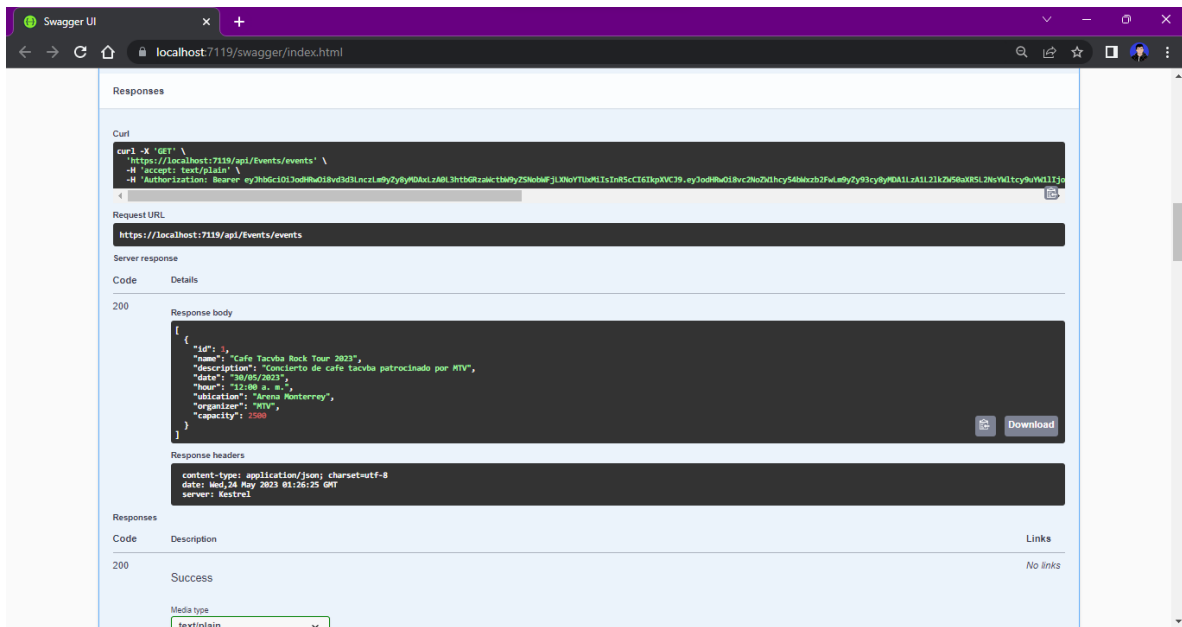
The image shows the Swagger UI interface after executing the PUT request. The 'Responses' section is active, displaying the following information:

- Curl:** A code block showing the curl command used to execute the request.
- Request URL:** 'https://localhost:7119/api/Events/update/1'.
- Server response:** A table with 'Code' and 'Details' columns. It shows a '200' status code.
- Response body:** A JSON object:

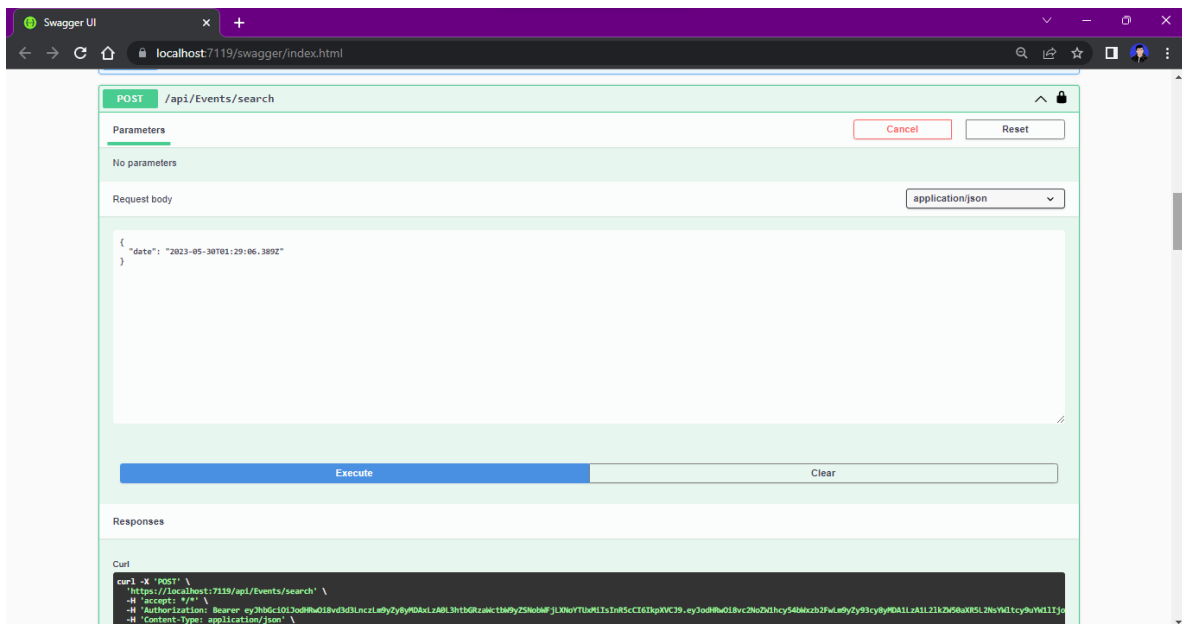
```
{  "message": "Event succesfully updated"}
```

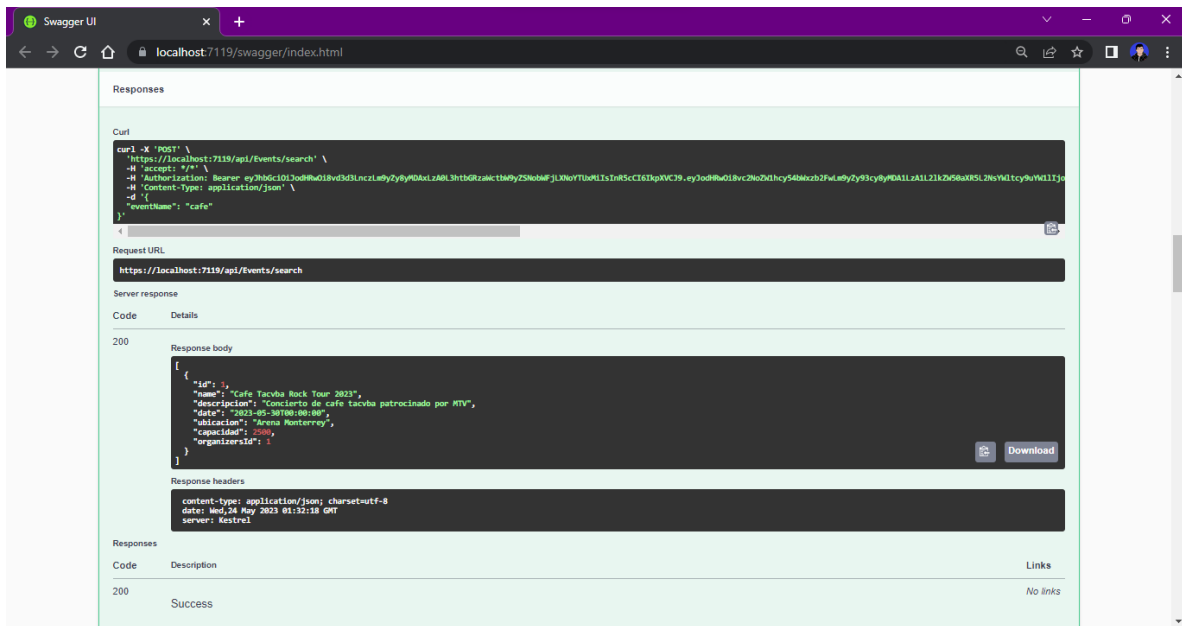
 A 'Download' button is next to it.
- Response headers:** A table with 'content-type: application/json; charset=utf-8', 'date: Wed, 24 May 2023 01:22:39 GMT', and 'server: Kestrel'.
- Responses table:** A table with columns 'Code', 'Description', and 'Links'. It shows a '200' status code with the description 'Success' and 'No links'.

Ahora podemos visualizar nuestros eventos creados. A través de la ruta '/api/Events/events' por medio del verbo GET.

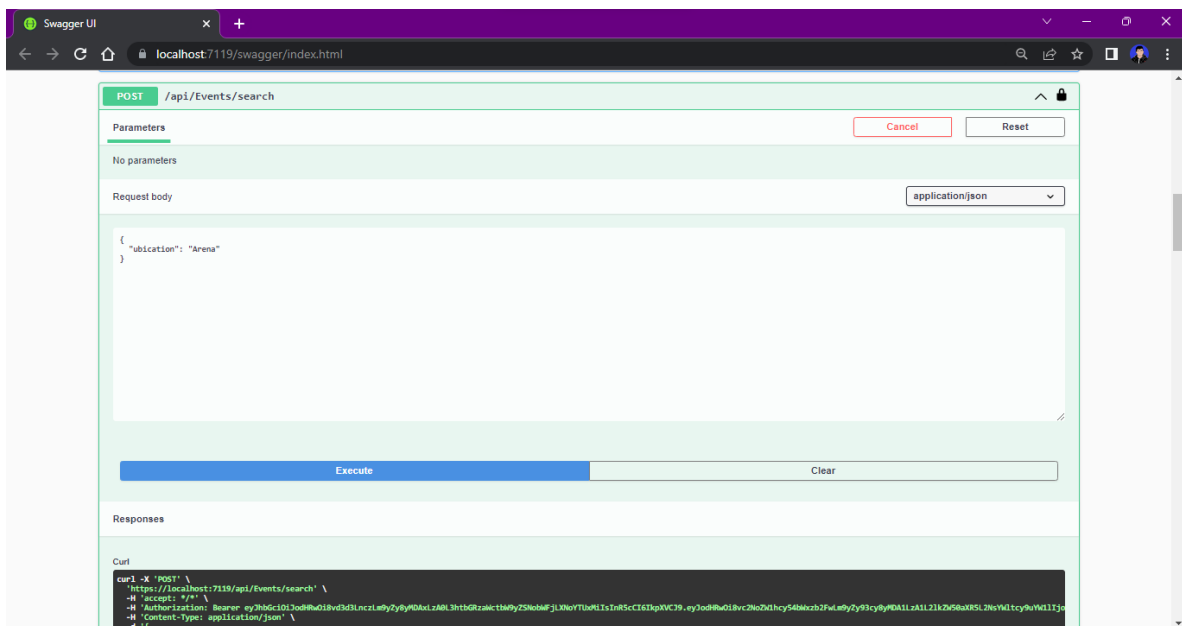


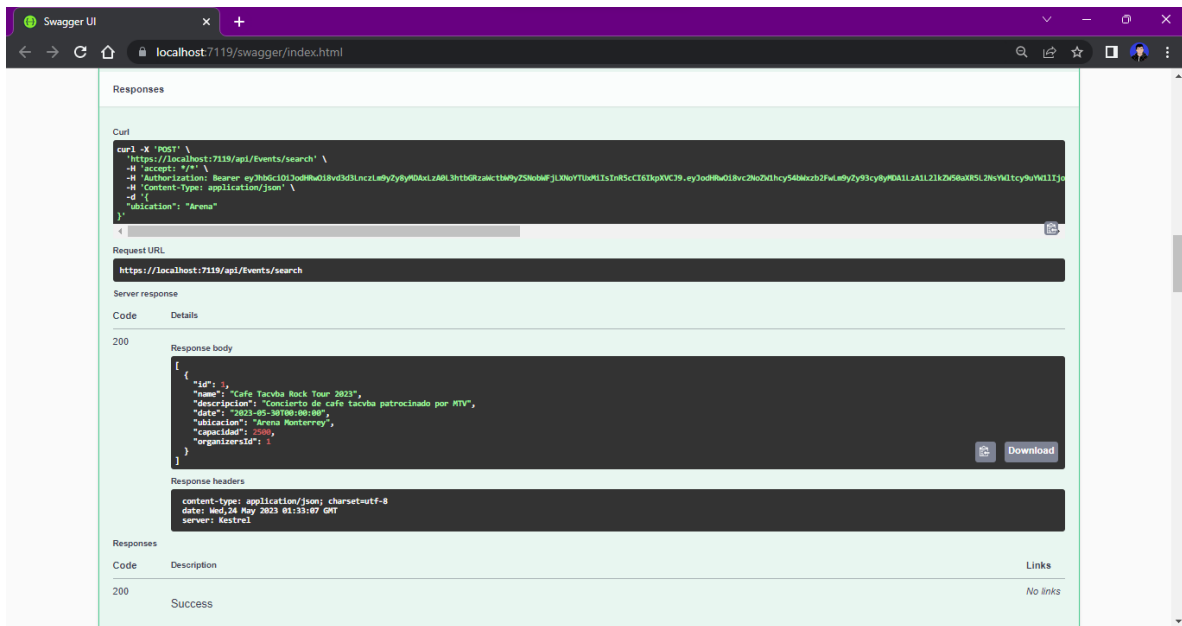
La ruta '/api/Events/search' busca eventos, dependiendo de un criterio, este puede ser por fecha





También filtra resultados por su ubicación.

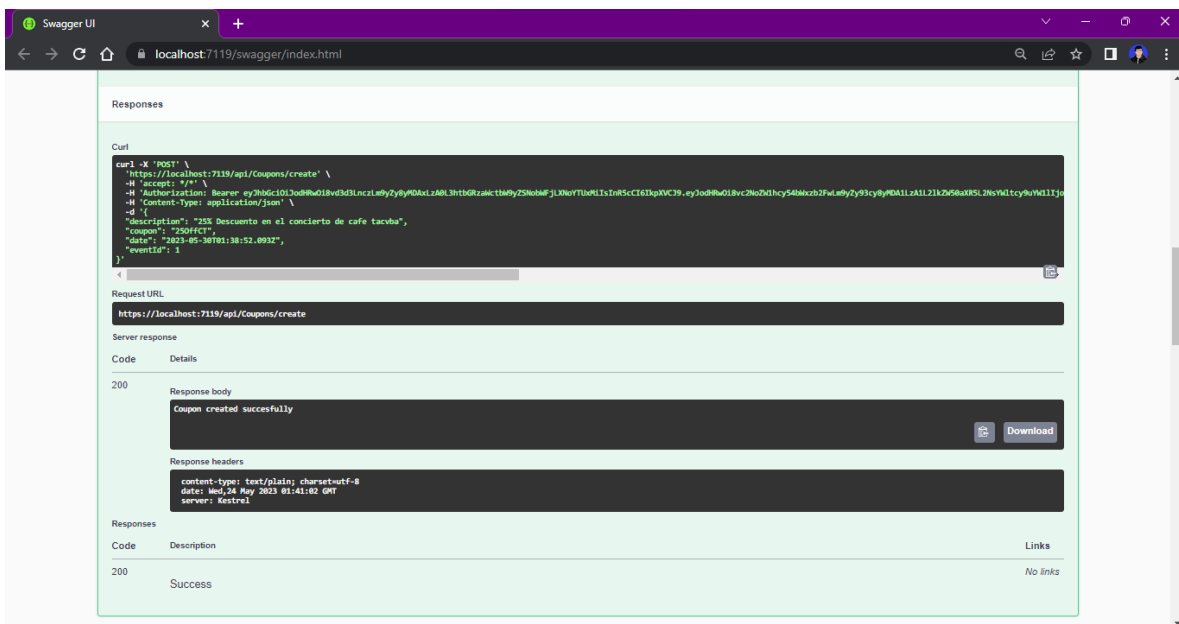
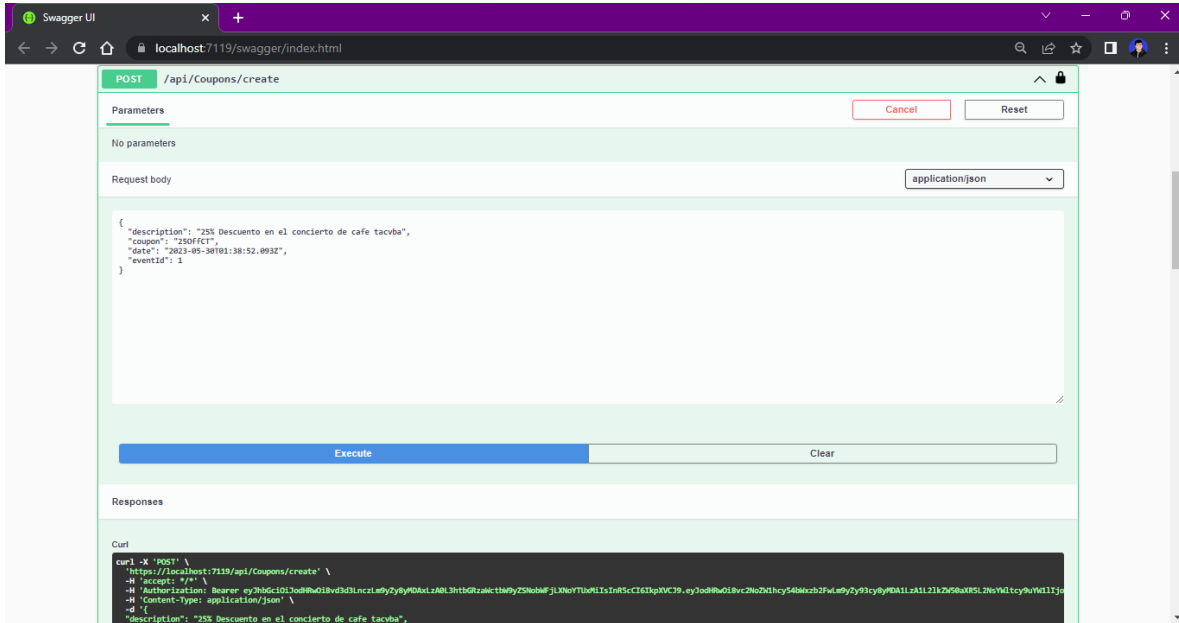




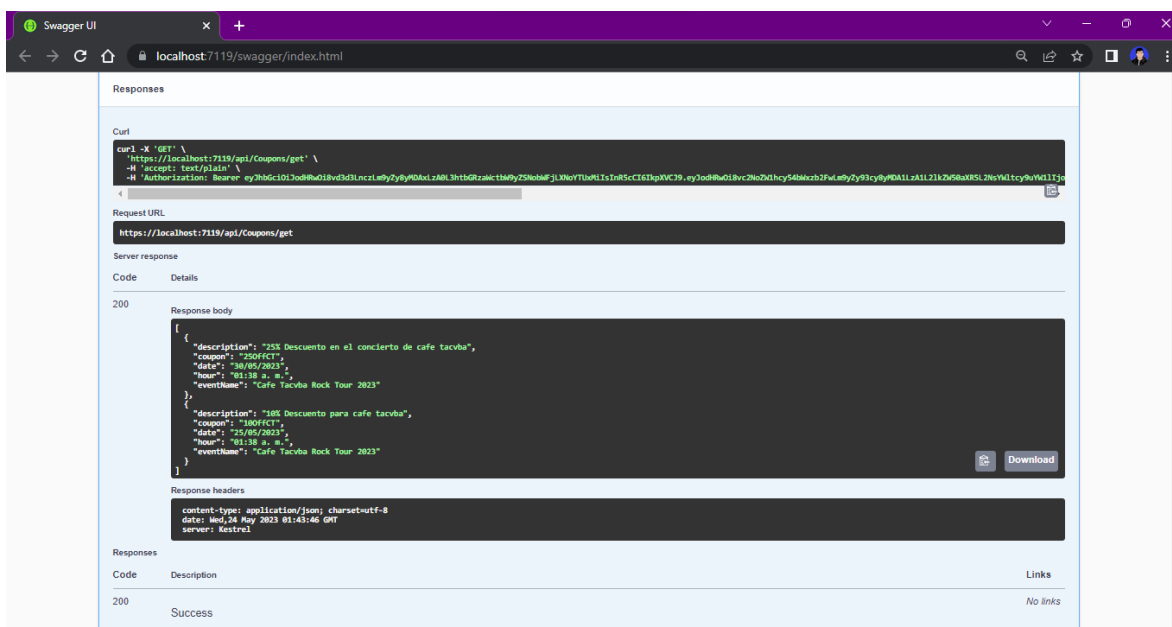
Este método también puede buscar realizando una combinación de cualquiera de estos tres campos, como resultados nos dará eventos que tengan estas similitudes.

Cupones

Por medio de la ruta 'api/Cupons/create' podemos crear cupones o promociones a eventos creados por nosotros, esta petición exige de los siguientes campos: descripción, nombre del cupón, fecha de expiración (El sistema verifica que la fecha no sea una anterior a la actual), ID del evento (El sistema valida que el ID sea de un evento del organizador).



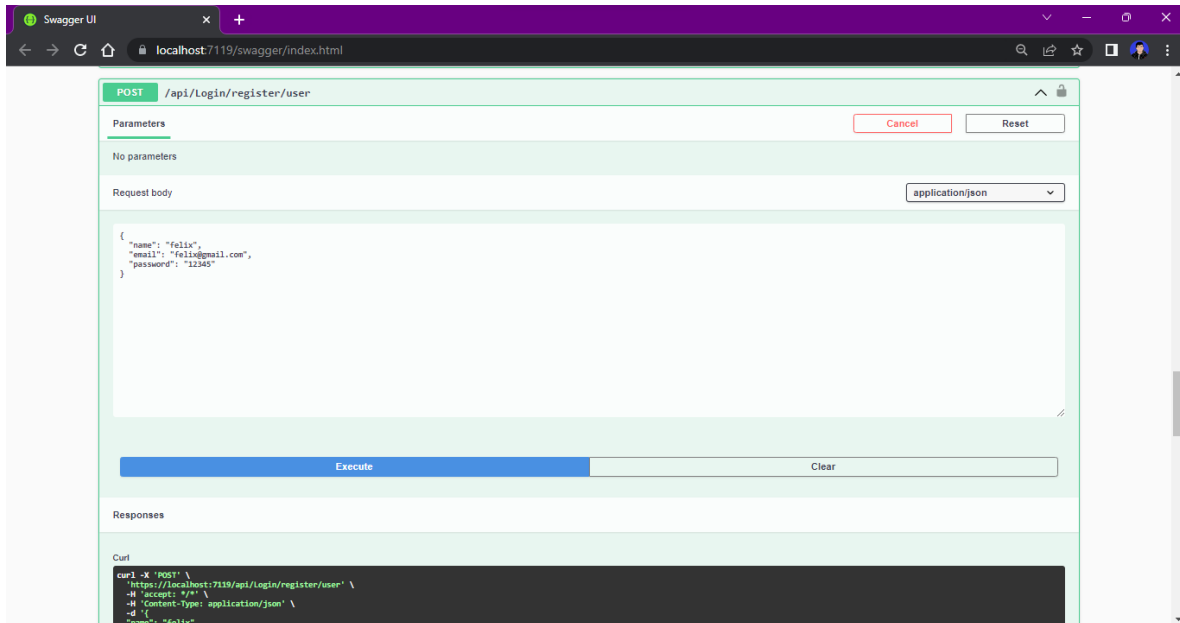
La ruta `/api/Cupons/get` nos da una lista de todos los cupones creados por organizador.



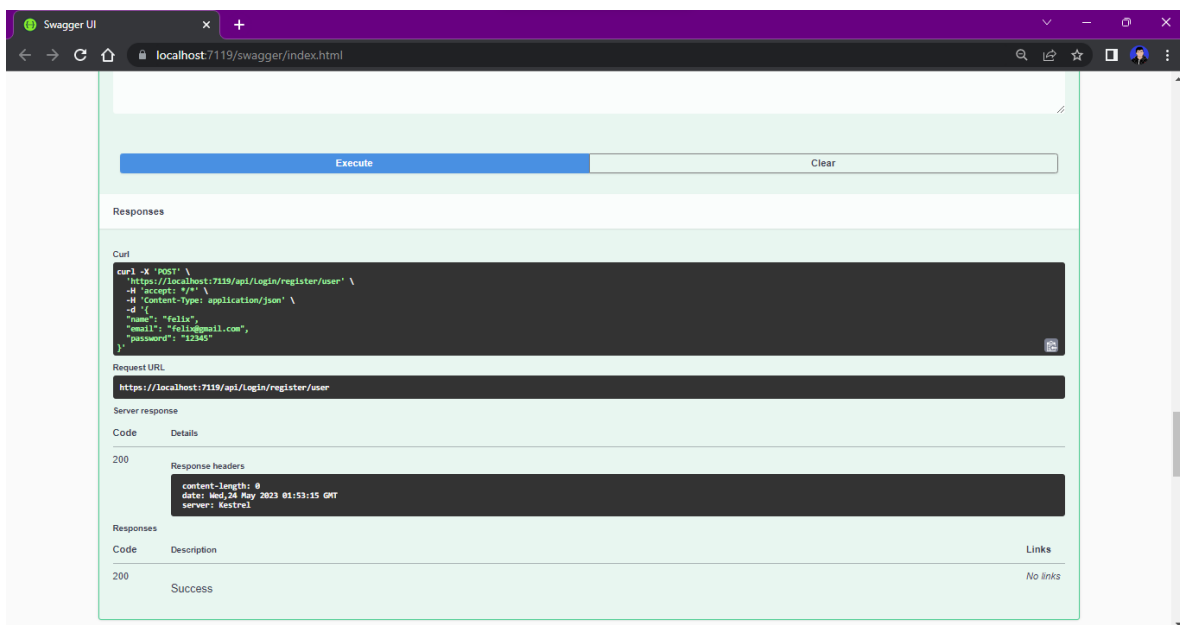
Usuarios

Creación e inicio de sesión.

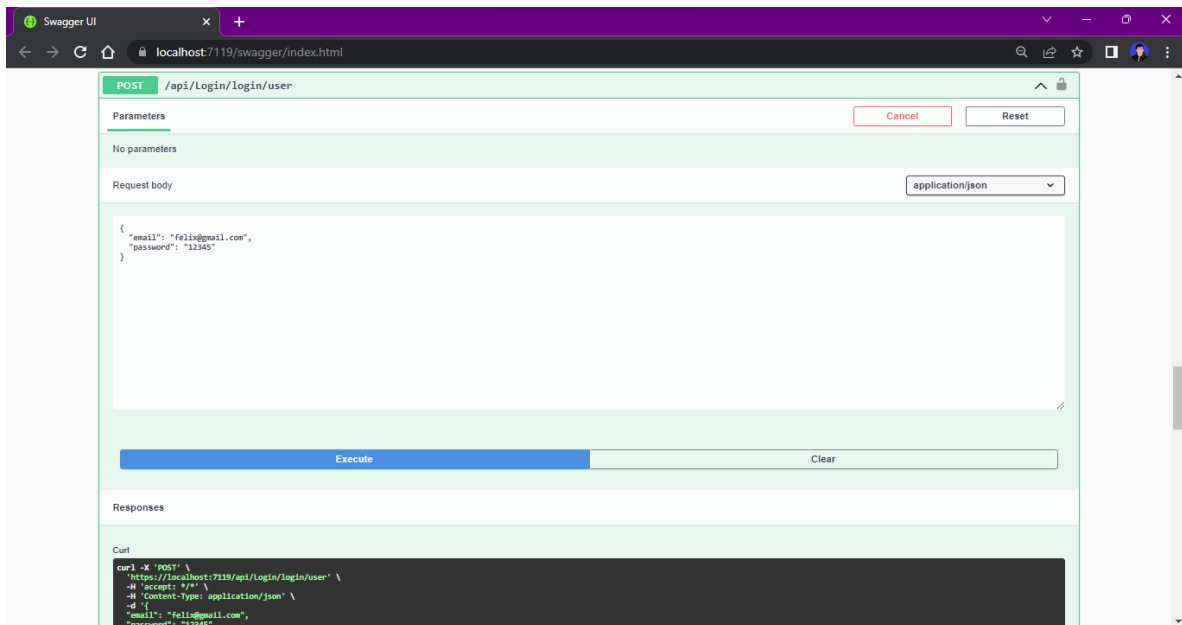
La ruta '/api/Login/register/user' nos va a permitir registrar usuarios, nos va a solicitar la siguiente información: nombre, correo electrónico (Valida que la cadena sea un correo electrónico) y contraseña.



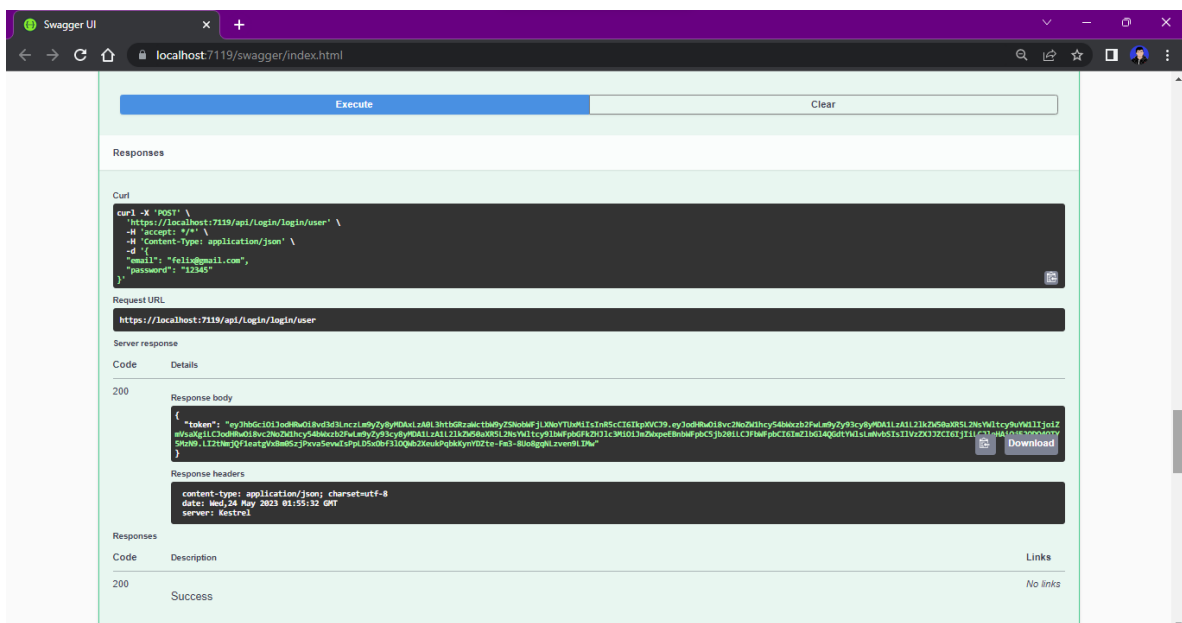
La respuesta que da esta petición.



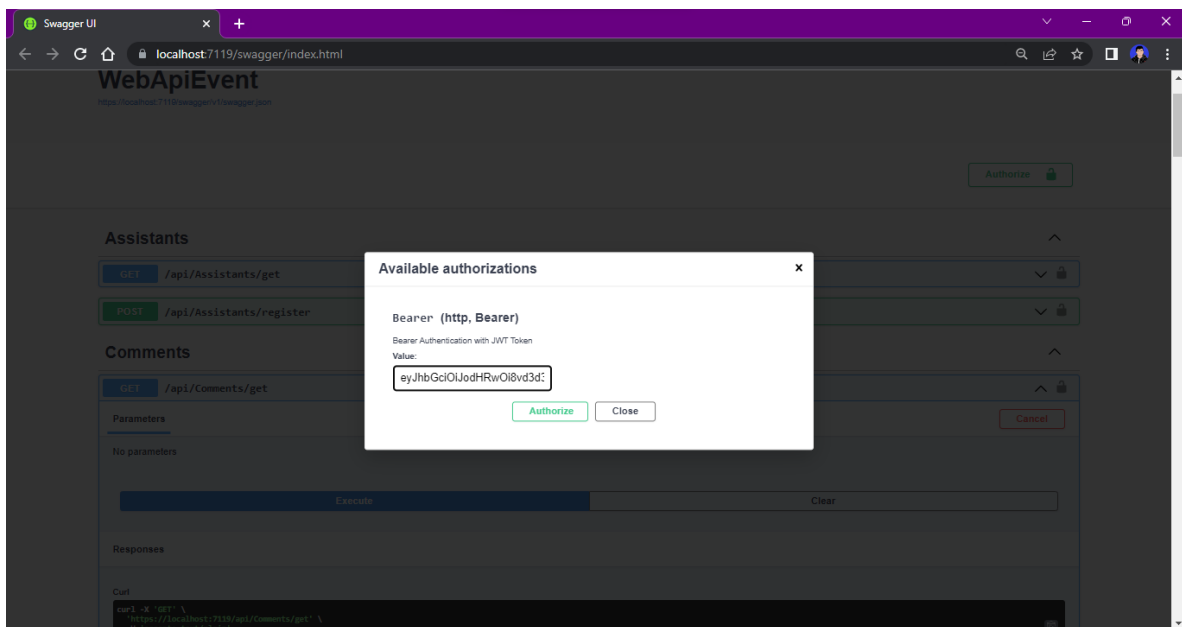
Para iniciar sesión como usuario, usamos la ruta `"/api/Login/login/user"` el cual nos solicitará ingresar un correo electrónico y un password.



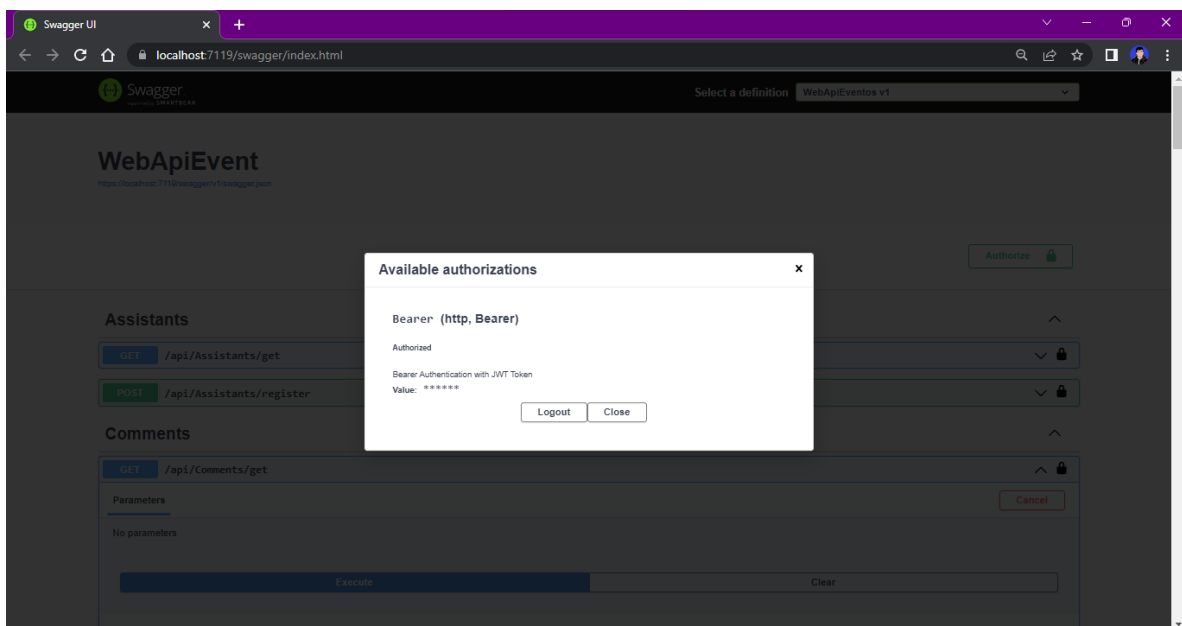
Al ejecutar esta petición nos proporcionará un token, el cual podemos utilizar para autorizar al sistema a acceder los métodos.



Para autorizar al sistema damos click a su respectivo botón, ingresamos el token dado al momento de iniciar sesión.

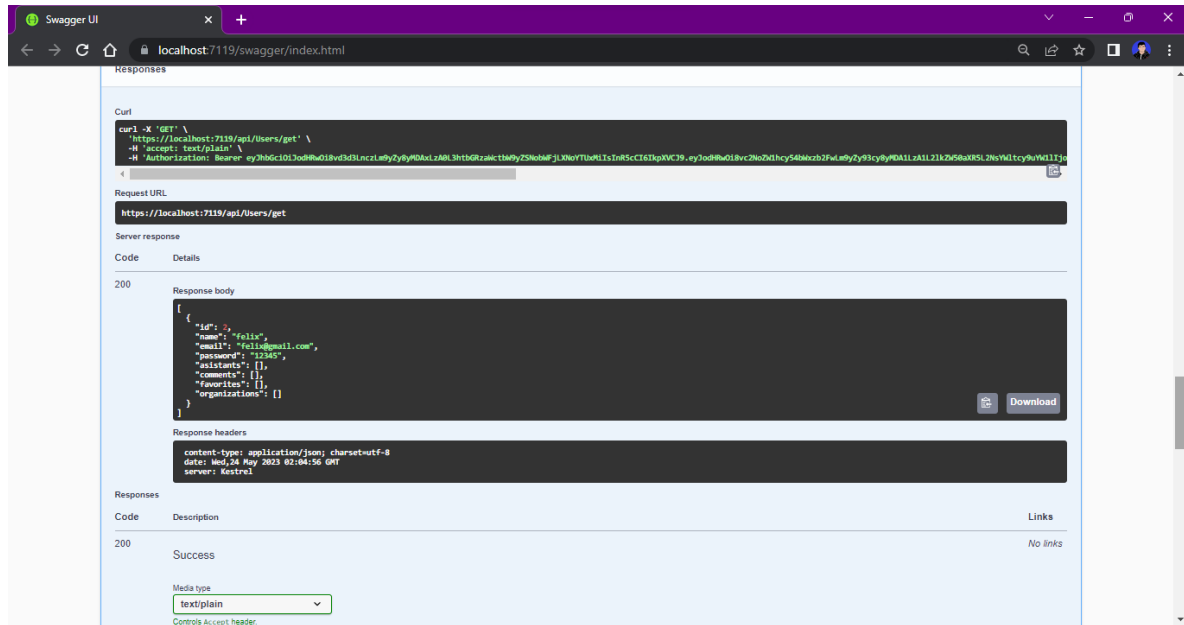


Al dar click al botón de autorizar nos dará un mensaje que ya estamos autorizados ante el sistema.



Usuarios

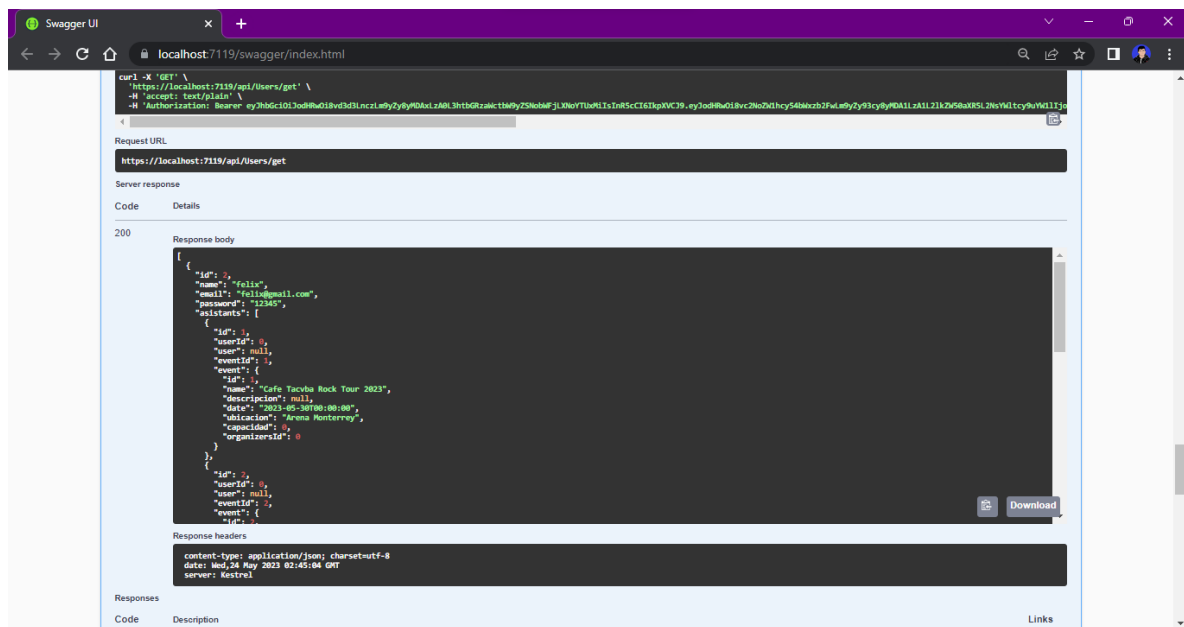
La ruta '/api/Users/get' nos proporcionará una lista de todos los usuarios registrados en nuestro sistema.



The screenshot shows the Swagger UI interface in a web browser. The URL bar indicates the base URL is `localhost:7119/swagger/index.html`. The 'Responses' section is expanded for the `GET /api/Users/get` endpoint. The 'Curl' tab shows the request command. The 'Request URL' is `https://localhost:7119/api/Users/get`. The 'Server response' section shows a `200` status code. The 'Response body' is a JSON object representing a user:

```
{  "id": 1,  "name": "felix",  "email": "felix@gmail.com",  "password": "12345",  "assistants": [],  "comments": [],  "favorites": [],  "organizations": []}
```

. The 'Response headers' section shows `content-type: application/json; charset=utf-8`, `date: Wed, 24 May 2023 02:04:56 GMT`, and `server: Restino`. At the bottom, a 'Responses' table shows a `200` status code with a 'Success' description and 'No links'.

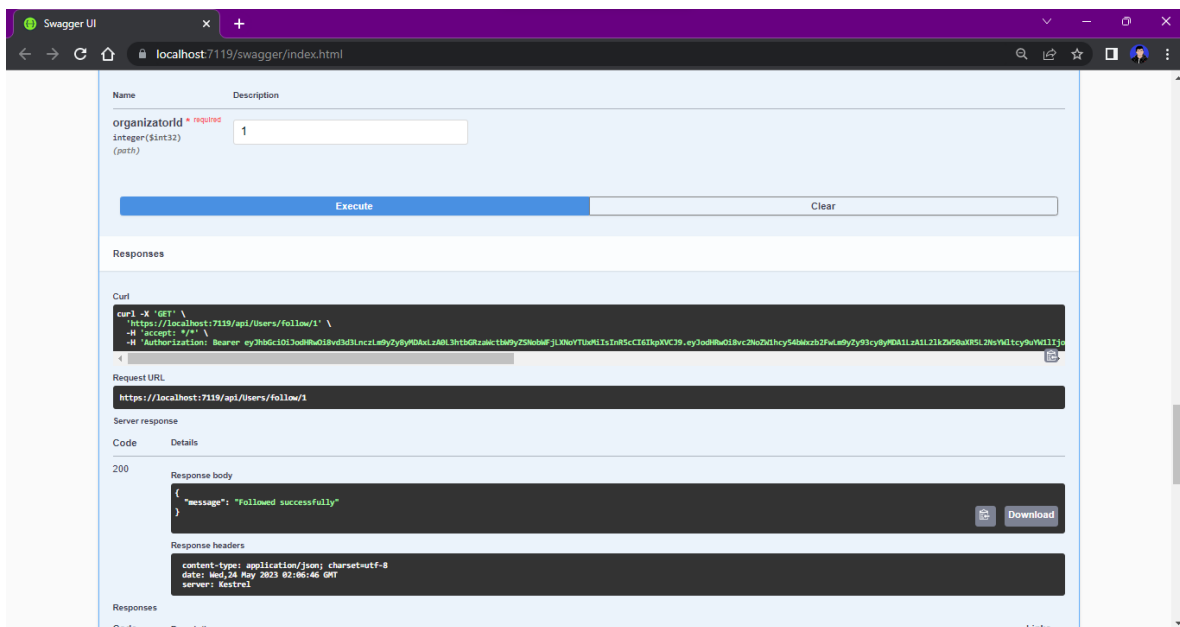


This screenshot is similar to the one above, showing the same endpoint and headers. However, the 'Response body' JSON object is more detailed, including an 'assistants' array with one assistant object:

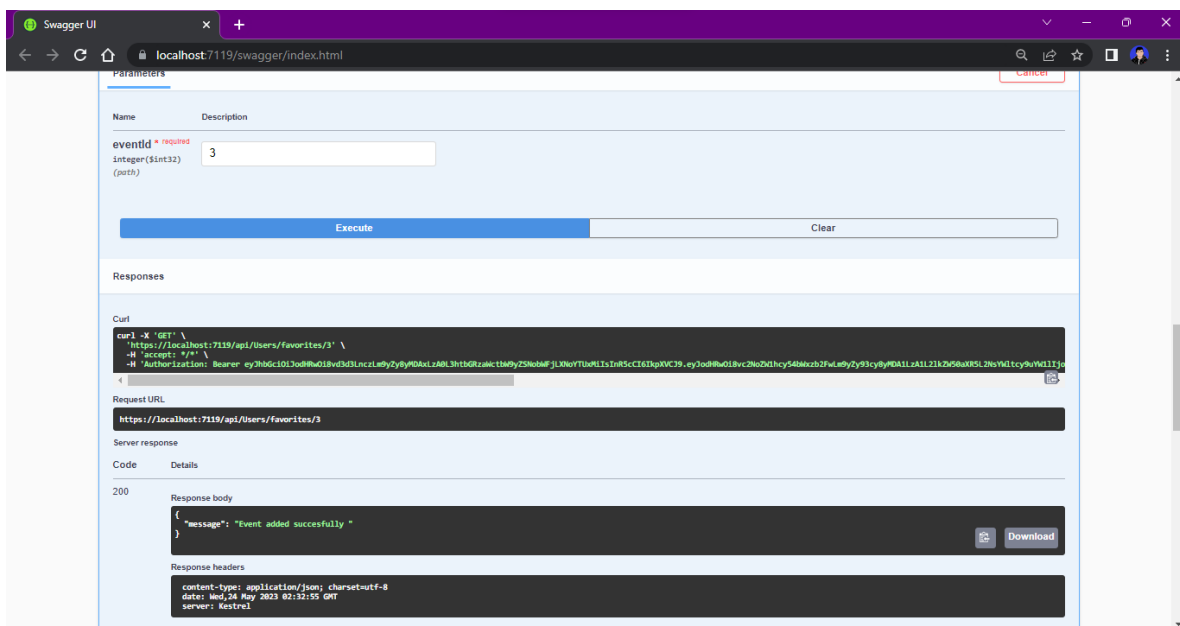
```
{  "id": 1,  "name": "felix",  "email": "felix@gmail.com",  "password": "12345",  "assistants": [    {      "id": 1,      "userId": 1,      "user": null,      "eventId": 1,      "event": {        "id": 1,        "name": "Cafe Tocado Rock Tour 2023",        "description": null,        "date": "2023-05-30T00:00:00",        "location": "Arenas Monterrey",        "capacidad": 0,        "organizersid": 0      }      }    ]  }
```

. The 'Response headers' and the bottom 'Responses' table remain the same.

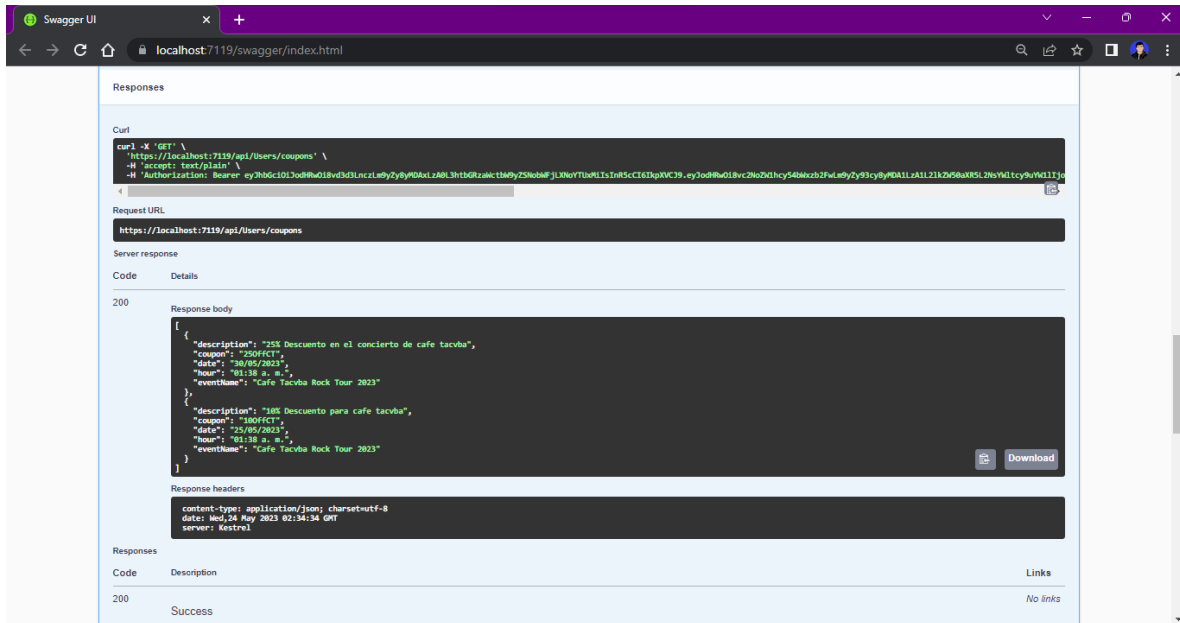
La ruta '/api/Users/follow/{OrganizadorId}' permite darle follow a un organizador para poder estar al tanto de sus eventos.



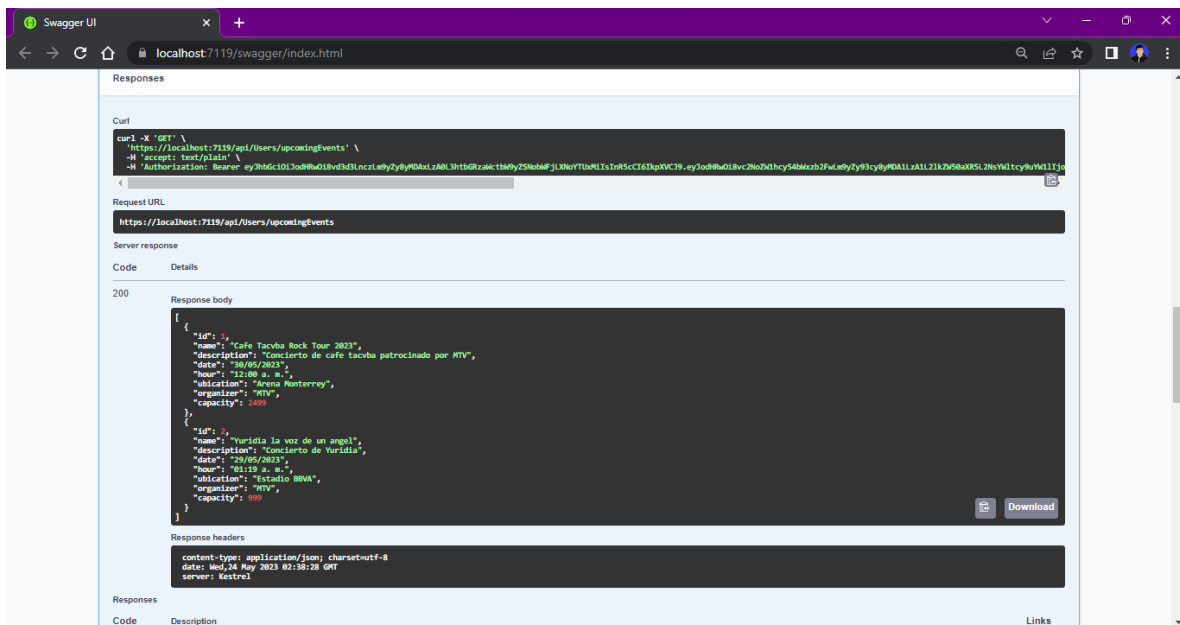
La ruta '/api/Users/favorite/{eventId}' permitirá al usuario registrar eventos favoritos que se encuentren en nuestro sistema, requiere como parámetro el ID del evento a registrar (El sistema verifica que este ID exista en el sistema).



La ruta '/api/Users/cupons' retorna un listado de los cupones que el usuario puede usar la condición para que salgan en este listado es que los cupones tienen que ser de los eventos que se haya marcado como asistencia, o aquellos eventos a los que el usuario haya registrado como favoritos, así como los cupones no hayan expirado en su fecha.

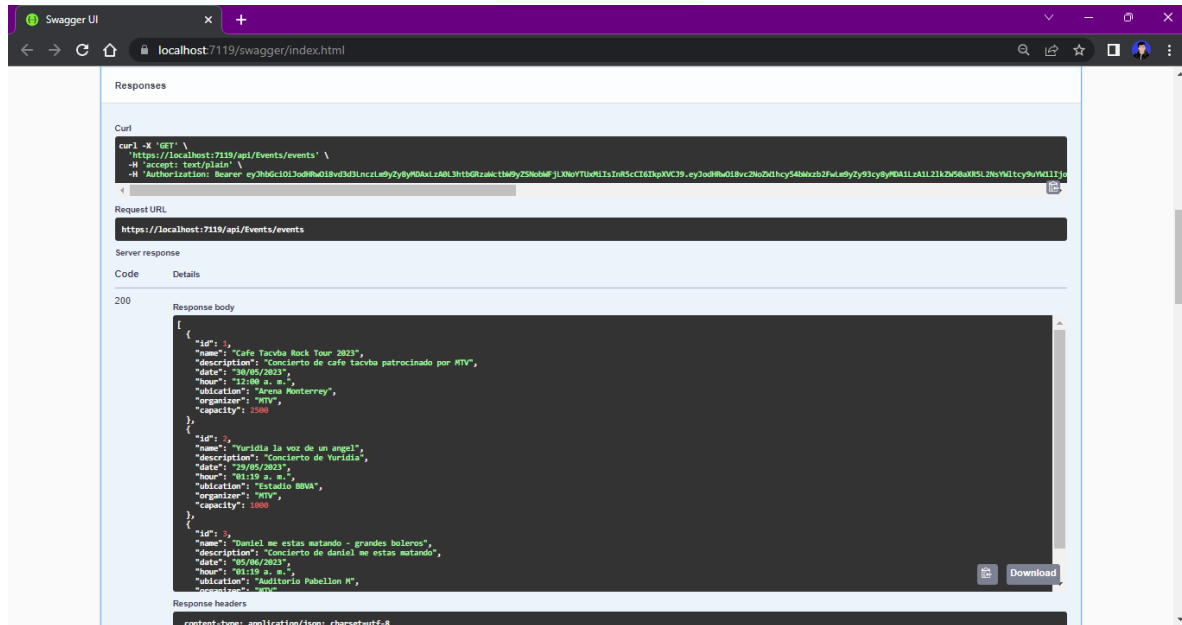


La ruta '/api/Users/upcomingEvents' nos retorna una lista de eventos que el usuario se registró como que asistiría y que están próximos a ocurrir, la condición de que se muestren es que el evento tenga una proximidad de 7 días a la fecha actual.

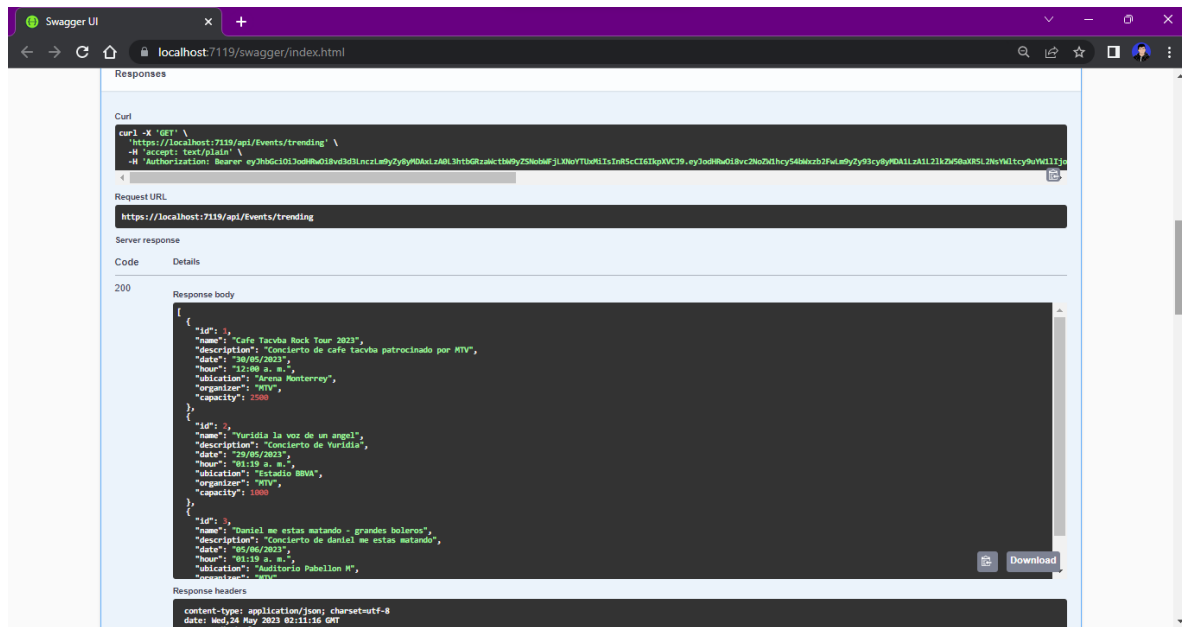


Eventos

La ruta '/api/Events/events' nos responderá con un listado de todos los eventos disponibles.



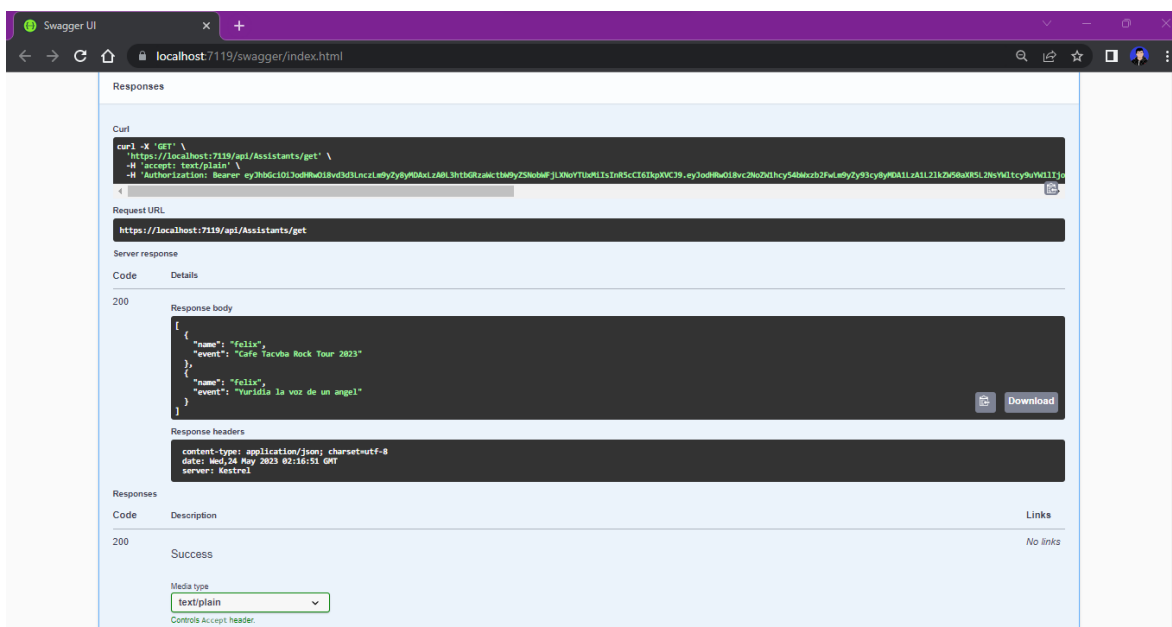
La ruta '/api/Events/trending' nos proporcionará una lista de los eventos más populares que están registrados en nuestro sistema.



Asistencia a eventos

The screenshot displays the Swagger UI for a web application. The 'Parameters' section is active, showing a table with one parameter: 'eventId' of type 'integer (int32)' with a value of '1'. Below the table is an 'Execute' button. The 'Responses' section shows a 202 status code with a 'Registration successful' message in the response body. The 'Request URL' is 'https://localhost:7119/api/Assistants/registerEventId1'. The 'Server response' section shows the response headers: 'content-type: application/json; charset=utf-8', 'date: Wed, 04 May 2023 02:15:27 GMT', and 'server: Kestrel'.

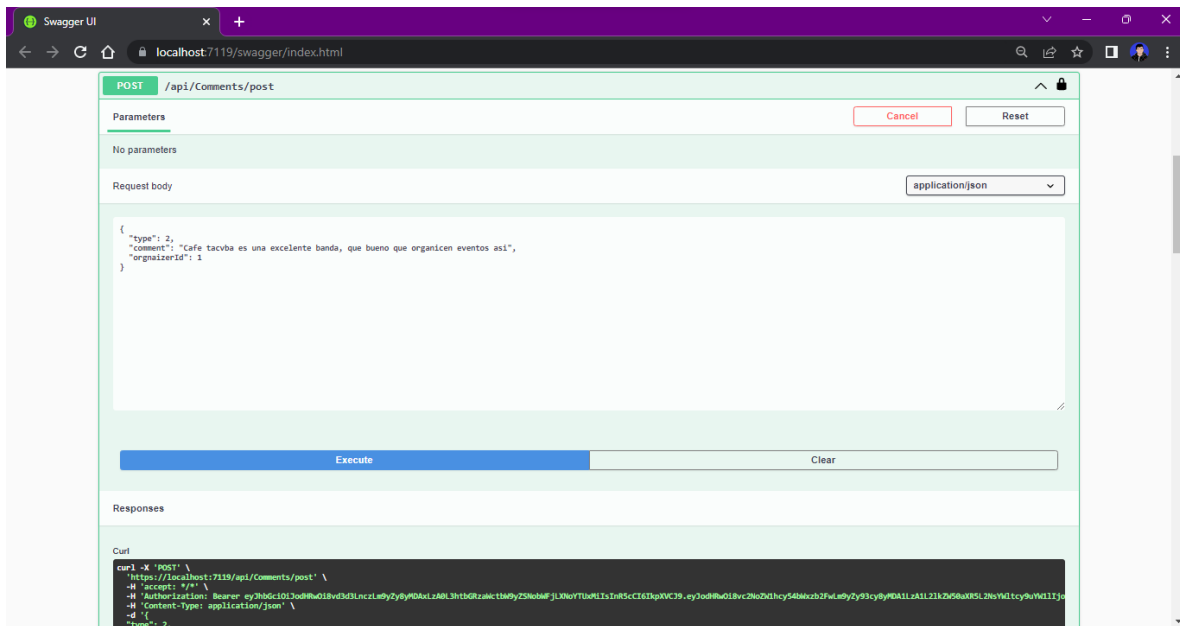
La ruta `/api/Assistants/get` nos dará como respuesta el listado de eventos a los que nuestro usuario se ha registrado.



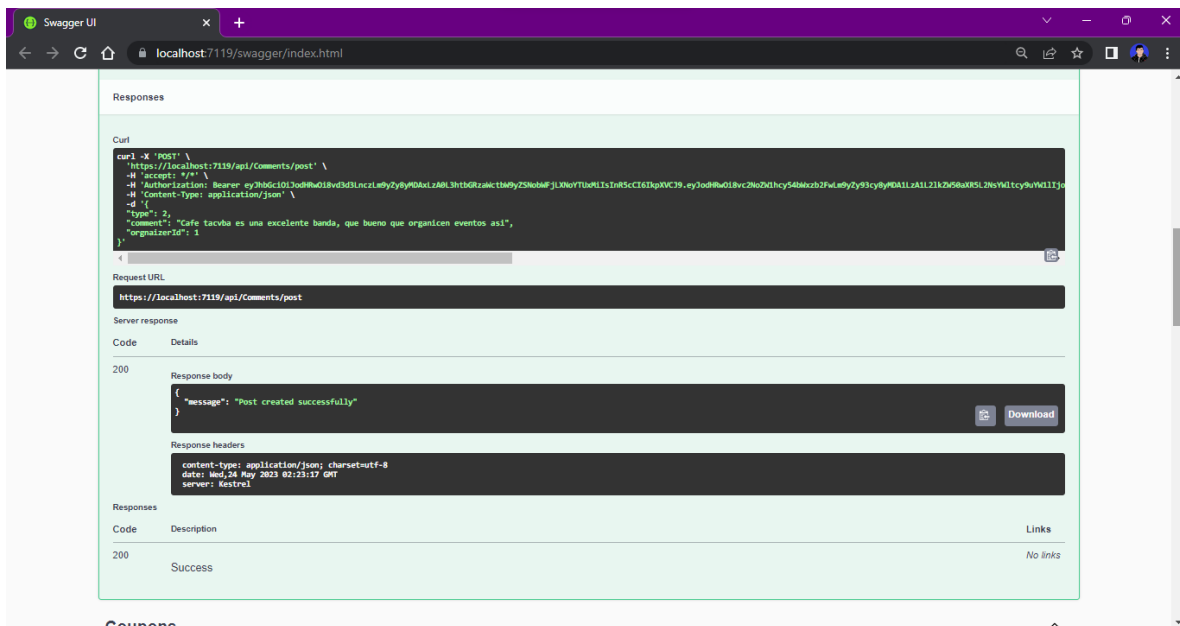
Comentarios

El sistema nos permite que los usuarios realicen comentarios a los organizadores.

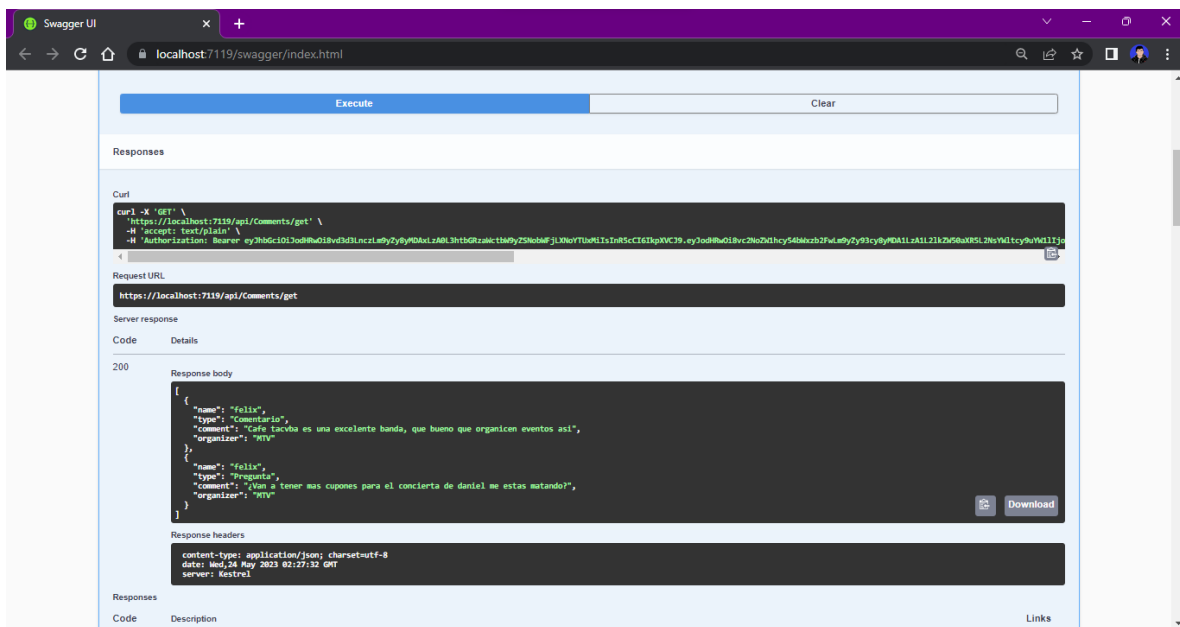
La ruta `/api/Comments/post` nos permitirá registrar un comentario, esta ruta requiere de la información: tipo (1 si es pregunta o 2 si es un comentario), comentario y el ID del organizador al que se le escribe el comentario.



La respuesta es.



La ruta `/api/Comments/get` nos generará una respuesta de todos los comentarios que el usuario le ha hecho a los organizadores.



Repositorio de github

Link del repositorio de github.

<https://github.com/Johann-28/WebApiEvent>

The screenshot shows a GitHub repository page for 'Johann-28/WebApiEvent'. The repository is on the 'master' branch, has 1 branch, and 0 tags. It shows a commit history with 49 commits. The repository contains files: VisualMaterial, WebApiEventos, .gitattributes, .gitignore, README.md, and WebApiEventos.sln. The README.md file is open, showing the title 'Api para gestion de eventos' and a description of the API. The right sidebar shows the 'About' section with no description, website, or topics provided. It also shows 'Releases' (No releases published), 'Packages' (No packages published), and 'Languages' (C# 100.0%).

Swagger UI x Johann-28/WebApiEvent x +

github.com/Johann-28/WebApiEvent

master 1 branch 0 tags

Go to file Add file <> Code

About

No description, website, or topics provided.

Readme

0 stars

1 watching

0 forks

Report repository

Releases

No releases published

Packages

No packages published

Languages

C# 100.0%

Johann-28 Añadiendo ruta correcta del diagrama der dd289b7 28 minutos ago 49 commits

File	Commit Message	Time Ago
VisualMaterial	añadiendo foto de diagrama DER	30 minutos ago
WebApiEventos	añadiendo informacion a readme	46 minutos ago
.gitattributes	Agregar .gitattributes y .gitignore.	3 days ago
.gitignore	Agregar .gitattributes y .gitignore.	3 days ago
README.md	Añadiendo ruta correcta del diagrama der	28 minutos ago
WebApiEventos.sln	Agregando base de datos	3 days ago

README.md

Api para gestion de eventos

La API se realizo con un modelo servicio-controlador pues es una forma debio a que el servicio tiene acceso a la base de datos a traves de una variable privada y el controlador tiene acceso al servicio a traves de una variable privada, por lo que la encapsulacion de datos es su punto fuerte aumentando al seguridad de al misma. A continuacion se mostrarán las partes del código responsables de cada punto, mostrando la funcion del servicio y controlador respectivo. Para más detalles en el codigo fuente se encuentra el código documentado de manera detallada para un mejor entendimiento.

Diagrama entidad relación

