API EVENTOS - NodeJs

Objetivo

Desarrollar una API para explorar los distintos eventos a futuro, la cual permitirá conocer y modificar los eventos que lo componen. También deberá permitir inscribirme en un evento y listar todos los participantes del mismo. Por otro lado, deberá exponer la información para que cualquier frontend pueda consumirla.

👉 Utilizar Node Js y **Express**.
👉 No es necesario armar el Frontend.
👉 Las rutas deberán seguir el patrón REST.
👉 Utilizar la librería **pg** para la conexión a base de datos y **dotenv** para almacenar la configuración.

Historial de Cambios

v1.0 (30/03/2024). Release inicial del trabajo práctico

v1.1 (05/04/2024).
- En el punto 5 se aclara el ejemplo de cómo funciona el rating.
- Se agrega el punto 10, rating de un evento.

v1.2 (21/04/2024).
- Se agrego en el enunciado que se espera la utilización de la biblioteca dotenv.
- Formateado del documento especialmente el anexo y respuestas de JSON.

v1.3 (22/04/2024).
- Se documentó el endpoint de /api/provinces.

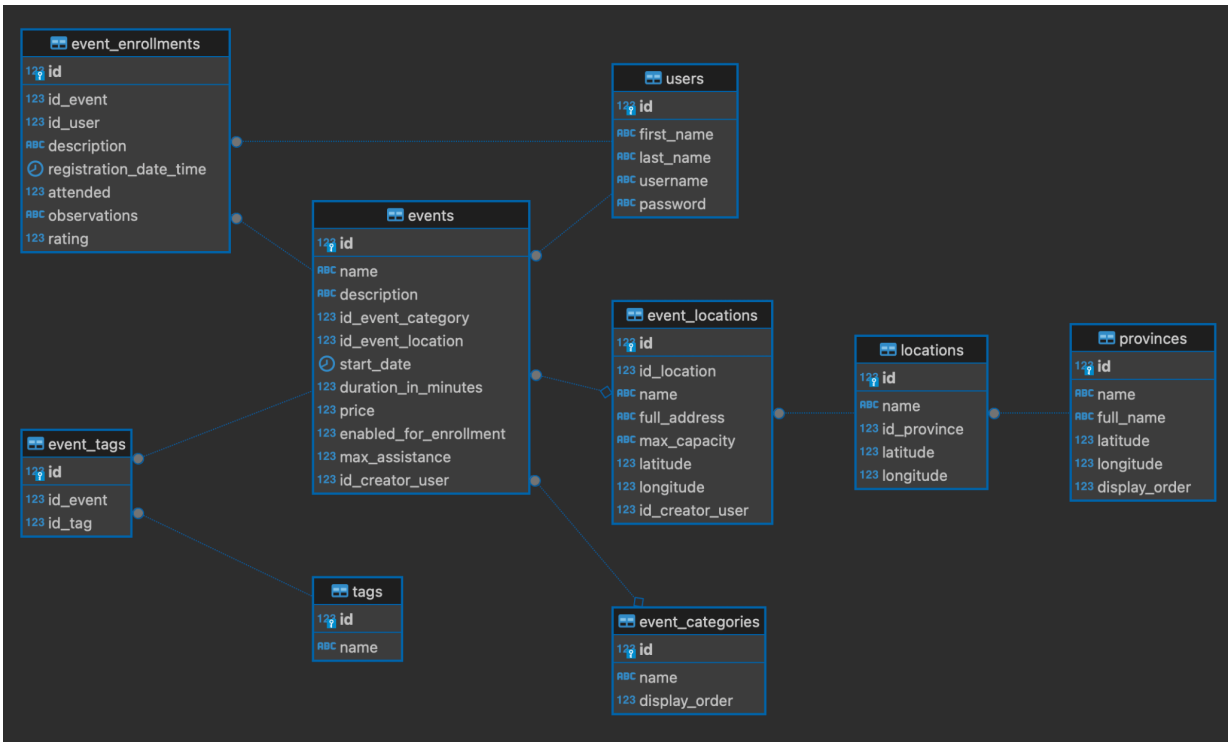
v1.4 (23/04/2024).
- Se documentaron varios endpoints y se documentaron que deben retornar en cada caso.

v1.5 (24/04/2024).
- Se agrego la des-registración de un usuario a un evento, con alguna validación.
- Se agregaron los endpoints de location.

Requerimientos técnicos

1. Modelado de Base de Datos.

Para esta primera etapa no van a necesitar crear ustedes la base de datos, por lo que se adjunta un DER de la solución para que puedan mockear lo que requieran.



2. Listado de Eventos

El listado deberá mostrar:

- Nombre.
- Descripcion.
- Categorías.
- Ubicación.
- Fecha del Evento.
- Duración del Evento.
- Precio de la entrada.
- Si está habilitado para inscribirse.
- La capacidad del evento
- El usuario creador del evento

El endpoint deberá ser:

GET /api/event/

Se recuerda que este endpoint debe estar paginado dado a la alta complejidad que se tiene para obtener cada uno de los eventos. Se adjunta un ejemplo de la respuesta que debería devolver el endpoint al final del documento.

3. Búsqueda de un Evento

Deberá permitir buscar por nombre, categoría, fecha de inicio o un tag determinado. Para especificar el término de búsqueda o filtros se deberán enviar como parámetros de query. Recuerde que puede filtrar por uno solo o cualquier combinación de los parámetros.

GET /api/event/?name=**{texto}**
GET /api/event/?category=**{texto}**
GET /api/event/?startdate=**{fecha YYYY-MM-DD}**
GET /api/event/?tag=**{texto}**

Ejemplo:
GET /api/event/?name=**taylor**&category=**Musica**&startdate=**2024-03-03**&tag=**Rock**

4. Detalle de un Evento

En el detalle deberán listarse todos los atributos del Evento, como así también su ubicación completa. Esto incluye la localidad y la provincia donde se encuentra la localización.

GET /api/event/**{id}**

5. Listado de Participantes

En el listado, deberá mostrarse la lista de usuarios inscriptos para un cierto evento. Este endpoint me deberá permitir filtrar por nombre, apellido, nombre de usuario, si asistió al evento y por rating mayor a un determinado valor. Recuerde que puede filtrar por uno solo o cualquier combinación de los parámetros. En el anexo puede encontrar un ejemplo de la respuesta.

GET /api/event/**{id}**/enrollment?first_name=**{texto}**
GET /api/event/**{id}**/enrollment?last_name=**{texto}**
GET /api/event/**{id}**/enrollment?username=**{texto}**
GET /api/event/**{id}**/enrollment?attended=**{boolean}**
GET /api/event/**{id}**/enrollment?rating=**{entero}**

6. Autenticación de Usuarios

Para realizar peticiones a los endpoints subsiguientes el usuario deberá contar con un token que obtendrá al autenticarse. Para ello, deberán desarrollar el endpoint de login, que permita obtener el token y el de registrarse.

El endpoint encargados de la autenticación y sus respectivos bodys deberán ser:

POST /api/user/login

Se envía la entidad en el body de request.

```
{
        "username": "...",
        "password": "..."
}
```

POST /api/user/register

Se envía la entidad en el body de request.

```
{
        "first_name": "...",
        "last_name": "...",
        "username": "...",
        "password": "..."
}
```

7. Creación, Edición, Eliminación y Listado de Provincias (CRUD)

Deberán existir las operaciones básicas de creación, edición, eliminación, obtener todos (paginado) y obtener por id de provincias.

Ejemplo

GET /api/province

Retorna un status code **200** (ok) y los datos.

GET /api/province/**{id}**

Retorna un status code **200** (ok) y los datos.

Retorna un status code **404** (not found) en caso de que el id no exista.

POST /api/province/

Se envía la entidad en el body de request.

Retorna un status code **201** (created).

Retorna un status code **400** (bad request) y el texto del error, en caso de existir algún error en las reglas de negocio (por ejemplo un nombre vacío, o menor a 3 letras).

PUT /api/province/

Se envía la entidad en el body de request.

Retorna un status code **200** (ok).

Retorna un status code **404** (not found) en caso de que el id no exista.

Retorna un status code **400** (bad request) y el texto del error, en caso de existir algún error en las reglas de negocio (por ejemplo un nombre vacío, o menor a 3 letras).

DELETE/api/province/**{id}**

Retorna un status code **200** (ok) y los datos.

Retorna un status code **404** (not found) en caso de que el id no exista.

8. Creación, Edición, Eliminación de Eventos (CRUD)

Se deberá completar el controller para permitir crear eventos nuevos completando todos los campos y además al usuario creador de cada evento se le debe permitir editar y eliminar sus eventos. NO se pueden borrar eventos en los que yo no sea el usuario creador.

9. Inscripción a un Evento

Se deberá permitir a un usuario registrarse en un evento, y eliminarse de la suscripción a un evento.

Ejemplo:

POST /api/event/**{id}**/enrollment/

Retorna un status code **201** (created), sí se pudo registrar.
Retorna un status code **404** (not found) en caso de que el id no exista.
Retorna un status code **400** (bad request) y un mensaje de error en caso de que exceda la capacidad máxima de registrados al evento.

DELETE/api/event/**{id}**/enrollment/

Retorna un status code **200** (ok), sí se pudo remover de la suscripción.
Retorna un status code **404** (not found) en caso de que el id del evento no exista, o que el usuario no se encuentre registrado al evento.
Retorna un status code **400** (bad request) y un mensaje de error en caso de que la fecha del evento sea menor o igual a la fecha actual, es decir que el evento ya comenzó (es hoy) o es un evento que ya pasó.

10. Rating de un Evento

Luego de que un evento finalice, se le deberá permitir a todos los usuarios que asistieron emitir un rating para dicho evento. El rating consiste en un número entre 1 y 10. Además, deberán poder enviar un feedback para que en los próximos eventos sepan que opinan otros usuarios.

Ejemplo:

PATCH /api/event/**{id}**/enrollment/**{entero}**

11. Locations

Deberán existir las operaciones para poder obtener una localidad, todas las localidades (paginado), o todas las localidades de una provincia (paginado).

Ejemplo

GET /api/location

Retorna un status code **200** (ok) y los datos.

GET /api/location/**{id}**

Retorna un status code **200** (ok) y los datos.

Retorna un status code **404** (not found) en caso de que el id de la location no exista.

GET /api/location/province/**{id}**

Retorna un status code **200** (ok) y los datos.

Retorna un status code **404** (not found) en caso de que el id de la provincia no exista.

Documentación

Es deseable documentar los endpoints utilizando alguna herramienta como Postman o Swagger.

Anexo Respuesta de los endpoints

Listado de Eventos:


```
{
    "collection": [
        {
            "id": 2,
            "name": "Taylor Swift",
            "description": "Un alto show",
            "start_date": "2024-03-21T03:00:00.000Z",
            "duration_in_minutes": 210,
            "price": "15500",
            "enabled_for_enrollment": true,
            "max_assistance": 120000,
            "tags": [
                "Rock",
                "Pop"
            ],
            "creator_user": {
                "id": 3,
                   "username": "Jschiffer",
                   "first_name": "Julian",
                   "last_name": "Schiffer"
            },
            "event_category": {
                "id": 1,
                "name": "Musica"
            },
            "event_location": {
                "id": 1,
              "name": "River",
              "full_address": "Av. Pres. Figueroa Alcorta 7597",
                "latitude": -34.5453,
                "longitude": -58.4498,
                "max_capacity": "84567"
            }
        },
    ],
    "pagination": {
        "limit": 15,
        "offset": 0,
        "nextPage": null,
        "total": "1"
    }
}
```

Listado de Participantes


```
{
    "collection": [
        {
            "user": {
                    "id": 3,
                    "username": "Jschiffer",
                    "first_name": "Julian",
                    "last_name": "Schiffer"
             },
            "attended": false,
            "rating": null,
            "description": null
        },
        {
            "user": {
                    "id": 1,
                    "username": "Polshetta",
                    "first_name": "Pablo",
                    "last_name": "Ulman"
            },
            "attended": true,
            "rating": 5,
            "description": "Alto Chow"
        }
    ],
    "pagination": {
        "limit": 15,
        "offset": 0,
        "nextPage": null,
        "total": "2"
    }
}
```