"API Documentation - Eventify"
Dev: "Santiago Coto Vila"
Date: "Septiembre 2025"
Project KeepCoding

Índice

Introducción

Este documento describe la API REST de Eventify, un sistema para gestión de usuarios con sus respectivos intereses, eventos y asistencias (RSVP).
La documentación está orientada para el resto del equipo de Eventify y así facilitar las llamadas desde herramientas como Postman y conocer en profundidad la API.

Base URL y Autenticación

- Base URL (dev): `http://localhost:8080`
- Prefijo global: `/api`
- Ejemplo completo: `http://localhost:8080/api/events`

Autenticación

- Registro devuelve `accessToken` y `refreshToken` (JWT).
- Login con Basic Auth (email + password) devuelve tokens.
- JWT: usar header `Authorization: Bearer <accessToken>` en rutas protegidas.
- API Key: algunas rutas pueden requerir header `X-API-Key`.

Modelos y DTOs

UsersDTO.Public

```
{ "id": "UUID", "name": "String", "email": "String" }
```

UsersDTO.Create

```
{ "name": "String", "email": "String", "password": "String", "interestIDs": ["UUID"] }
```

InterestDTO.Response

```
{ "id": "UUID", "name": "String", "nameClean": "String" }
```

EventsDTO.Public

```
{ "id": "UUID", "name": "String", "category": "String", "lat": "Double?", "lng": "Double?", "userID":
"UUID", "createdAt": "Date?", "updatedAt": “Date?”, “eventDate”: “Date?”, “location”: “String?” }
```

EventAttendeesDTO.Public

```
{ "id": "UUID", "eventID": "UUID", "userID": "UUID", "status": "going|maybe|declined", "joinedAt": "Date?",
"updatedAt": "Date?" }
```

Endpoints

1. Health

GET /
✅ 200: `"It works server Eventify!"`

---

2. Auth

POST `/api/auth/register`

Body:

```
{ "name":"Ana", "email":"ana@example.com", "password":"Secreta123", "interestIDs": ["UUID1","UUID2","UUID3"]
}
```

Response:

```
{ "accessToken":"...", "refreshToken":"..." }
```

POST `/api/auth/login`

Headers:

```
Authorization: Basic base64(email:password)
```

Response:

```
{ "accessToken":"...", "refreshToken":"..." }
```

---

3. Users

GET `/api/users`

→ Devuelve `[UsersDTO.Public]`

GET `/api/users/{userID}`

→ Devuelve `UsersDTO.Public` o 404 si no existe.

DELETE `/api/users/{userID}`

---

4. Interests

GET `/api/interests`

→ Lista de intereses.

POST `/api/interests`

Body:

```
{ "name": "Música" }
```

Valida unicidad y nombre no vacío.

GET `/api/interests/{interestID}`

PATCH `/api/interests/{interestID}`

DELETE `/api/interests/{interestID}`

---

5. Events

GET `/api/events?page=1&per=10`

Devuelve `Page<EventsDTO.Public>` (ordenado desc).

GET `/api/events/{eventID}`

POST `/api/events`

```
{ "name":"Meetup Eventify", "category":"Tech", "userID":"UUID", "lat":-34.6, "lng":-58.38 }
```

PATCH `/api/events/{eventID}`

DELETE `/api/events/{eventID}`

---

6. RSVP / Asistencia a eventos

POST `/api/rsvp` (pública, userID en body)

```
{ "eventID":"E-UUID", "userID":"U-UUID", "status":"going" }
```

POST `/api/{eventID}/rsvp` (JWT)

Headers:

```
Authorization: Bearer <accessToken>
```

Body:

```
{ "eventID":"E-UUID", "status":"maybe" }
```

PUT `/api/{eventID}/rsvp` (JWT)

```
{ "status":"declined" }
```

Ejemplos cURL

```
# Registrar usuario
curl -X POST "http://localhost:8080/api/auth/register" \
-H "Content-Type: application/json" -H "Accept: application/json" \
-d '{ "name":"Ana","email":"ana@example.com","password":"Secreta123","interestIDs":["UUID1","UUID2","UUID3"]
}'

# Login
curl -X POST "http://localhost:8080/api/auth/login" \
-H "Authorization: Basic $(printf "ana@example.com:Secreta123" | base64)" \
-H "Accept: application/json"

# Crear evento
curl -X POST "http://localhost:8080/api/events" \
-H "Content-Type: application/json" -H "Accept: application/json" \
-d '{ "name":"Meetup iOS","category":"Tech","userID":"UUID" }'

# RSVP público
curl -X POST "http://localhost:8080/api/rsvp" \
-H "Content-Type: application/json" -H "Accept: application/json" \
-d '{ "eventID":"E-UUID", "userID":"U-UUID", "status":"going" }'
```

Códigos de Error

- 400.Bad Request → Validaciones o IDs mal formados.
- 401.Unauthorized → Token ausente/expirado o API Key inválida.
- 404 Not Found → Recurso inexistente.
- 500 Internal Server Error → Errores internos del servidor. Formato de

error:

```
{ "error": true, "reason": "Mensaje descriptivo" }
```