# Infinity Journey - Documentación Backend

## Introducción

Infinity Journey es una API REST desarrollada con Spring Boot. Esta gestiona la autenticación de usuarios, la creación de partidas, entidades, eventos, puntuaciones y recompensas.

## Tecnologías

- Spring Boot
- Spring Security (JWT)
- JPA/Hibernate
- Base de datos: MySQL

## Autenticación y Seguridad

El sistema utiliza autenticación mediante JSON Web Tokens (JWT).

| Método | Ruta | Descripción |
|---|---|---|
| POST | /auth/login | Inicio de sesión |
| POST | /auth/upload | Registro de usuario |

Header requerido para rutas protegidas:

**Authorization: Bearer <token>**

Roles disponibles:
- Usuario (por defecto)
- Admin (acceso a endpoints restringidos)

## Endpoints - Usuario

| Método | Ruta | Descripción |
|---|---|---|
| GET | /api/usuarios/me | Obtener información del usuario autenticado |
| GET | /api/usuarios/all | Listar usuarios con eventos |
| GET | /api/usuarios/max-score | Obtener mayor puntuación del usuario |
| POST | /api/usuarios/puntuacion | Actualizar puntuación de usuario |
| PUT | /api/usuarios/actualizar | Actualizar puntuación |

| | | |
|---|---|---|
| | | máxima si es mayor |
| GET | /api/usuarios/ranking | Top 10 usuarios con mayor puntuación |
| GET | /api/usuarios/puntuaciones | Todas las puntuaciones del usuario |

## Endpoints - Entidades

| Método | Ruta | Descripción |
|---|---|---|
| GET | /api/entidad/all | Listar todas las entidades |
| GET | /api/entidad/plantilla | Listar entidades plantilla |
| POST | /api/entidad/insertar | Crear una nueva entidad |

## Endpoints - Eventos

| Método | Ruta | Descripción |
|---|---|---|
| GET | /api/eventos/{id} | Obtener evento por ID |
| POST | /api/eventos/upload | Crear nuevo evento |
| PUT | /api/eventos/update | Actualizar un evento existente |
| POST | /api/eventos/eliminar | Eliminar evento por ID |
| GET | /api/eventos/aleatorio | Obtener evento aleatorio por idioma y dificultad |

## Endpoints - Partidas

| Método | Ruta | Descripción |
|---|---|---|
| GET | /api/partida/all | Listar todas las partidas |
| GET | /api/partida/plantilla | Listar partidas plantilla |
| POST | /api/partida/insertarPlantilla | Añadir partida plantilla |
| POST | /api/partida/mi-partida | Crear partida del usuario |
| GET | /api/partida/mi-partida | Obtener partida del usuario |
| GET | /api/partida/mi-partidaBoolean | Verificar si el usuario tiene partida |
| DELETE | /api/partida/eliminarMiPartida | Eliminar la partida del usuario |
| PUT | /api/partida/mi-partida | Actualizar partida del usuario |

## Recuperación de Contraseña

| Método | Ruta | Descripción |
|---|---|---|

| | | |
|---|---|---|
| POST | /usuarios/recuperacion | Solicitar restablecimiento de contraseña |
| POST | /usuarios/comprobarToken | Comprobar token de validación |
| PUT | /usuarios/cambiarContrasena | Cambiar contraseña con token válido |

## DTOs Disponibles

Los DTOs (Data Transfer Objects) utilizados en el backend incluyen:

- UsuarioDTO
- PuntuacionDTO
- EntidadDTO
- EventoDTO
- OpcionDTO
- RecompensaDTO
- PartidaDTO

Los DTOs creados para un uso especifico:

- UsuarioEventoDTO Obtener eventos por email y rol.
- UsuarioRecordDTO Obtener puntuación máxima por nombre de usuario
- PartidaMeDTO Obtener partida de usuario especifico por id.

## Nota Importante

La mayoría de los endpoints requieren un token JWT válido en la cabecera HTTP para acceder. Este token se obtiene mediante el endpoint de inicio de sesión y debe incluirse como: Authorization: Bearer <token>.

## Ejemplos de Uso de Endpoints

A continuación se describen los principales endpoints del sistema con ejemplos del método HTTP, ruta utilizada, el body (si aplica) y una breve explicación de qué se espera para obtener una respuesta favorable.

### Autenticación - Login

**Método:** POST

**Ruta:** /api/auth/login

**Body (JSON):**

```
{
  "email": "usuario@x.com",
  "contraseña": "texto"
}
```

```
{
  "nombre": "usuario ",
  "contraseña": "texto"
}
```

Este endpoint permite iniciar sesión con email y contraseña. Si las credenciales son válidas, devuelve un token JWT que debe usarse en los siguientes endpoints en el header Authorization.

### Registro de Usuario

**Método:** POST

**Ruta:** /api/auth/upload

**Body (JSON):**

```
{
  "nombre": "prueba",
  "email": "prueba@x.com",
  "contraseña": "prueba",
  "tipo_usuario": "JUGADOR",
  "puntuacion_max": 0,
  "puntuaciones": [],
  "eventos": []
}
```

Este endpoint permite registrar un nuevo usuario. La contraseña será encriptada automáticamente antes de ser almacenada.

### Obtener Usuario Autenticado

**Método:** GET

**Ruta:** /api/usuarios/me

**Header:** Authorization: Bearer <token>

Este endpoint devuelve la información del usuario autenticado, utilizando el token proporcionado en el login.

### Obtener Lista Usuario Con Evento

**Método:** GET

**Ruta:** /api/usuarios/all

**Header:** Authorization: Bearer <token>

Este endpoint devuelve una lista de usuarios que tienen un evento propio ademas del autetificado.

## Obtener Puntuación Maxima

**Método:** GET

**Ruta:** /api/usuarios/max-score

**Header:** Authorization: Bearer <token>

**Body (JSON):**

Este endpoint devuelve la mayor puntuacion del usuario.

## Actualizar Puntuación

**Método:** POST

**Ruta:** /api/usuarios/puntuacion

**Header:** Authorization: Bearer <token>

**Body (JSON):**

```
{
 "puntuación": 1500,
 "fecha": "2025-06-01"
}
```

Este endpoint agrega una nueva puntuación para el usuario autenticado. Si se desea actualizar la puntuación máxima, debe utilizarse el endpoint de actualización.

## Actualizar Récord Máximo

**Método:** PUT

**Ruta:** /api/usuarios/actualizar

**Header:** Authorization: Bearer <token>

**Body (JSON):**

{ "puntuacion": 2000 }

Si la puntuación enviada es mayor que el récord actual, este será actualizado en el sistema.

## Ranking de Usuarios

**Método:** GET

**Ruta:** /api/usuarios/ranking

**Header:** Authorization: Bearer <token>

Devuelve el top 10 de usuarios con mejor puntuación. Si el usuario actual no está en ese top, se incluirá en el puesto 11.

## Puntuaciones del Usuario

**Método:** GET

**Ruta:** /api/usuarios/puntuaciones

**Header:** Authorization: Bearer <token>

Devuelve todas las puntuaciones del usuario ordenadas de mayor a menor.

## Obtener Entidades

**Método:** GET

**Ruta:** /api/entidad/all

**Header:** Authorization: Bearer <token>

Devuelve todas las entidades creadas por el usuario y las que están disponibles globalmente. Ideal para visualizar el conjunto de entidades en el juego.

## Obtener Entidades Plantilla

**Método:** GET

**Ruta:** /api/entidad/plantilla

**Header:** Authorization: Bearer <token>

Devuelve todas las entidades que se usan como Plantilla para el uso del todos los usuarios.

## Crear Nueva Entidad

**Método:** POST

**Ruta:** /api/entidad/insertar

**Header:** Authorization: Bearer <token>

**Body (JSON):**

```
{
  "nombre": "Guerrero",
  "ataque": 100,
  "defensa": 80,
  "imagen": "url_imagen.png",
  "usuario_id": 1,
  "esPlantilla": false
}
```

Permite crear una nueva entidad personalizada con atributos específicos. Estas entidades pueden asociarse a eventos.

## Obtener Evento Especifica

**Método:** GET

**Ruta:** /api/eventos/{id}

**Header:** Authorization: Bearer <token>

Devuelve el evento que coincide con el id insertado.

## Crear Evento

**Método:** POST

**Ruta:** /api/eventos/upload

**Header:** Authorization: Bearer <token>

**Body (JSON - simplificado):**

```
{
 "dificultad": 1,
 "descripcion": "Encuentras una criatura...",
 "idioma": "es",
 "entidad": {
  "nombre": "Criatura", "ataque": 50, "defensa": 30, "imagen": "img.png"
 },
 "estado": 0,
 "opciones": [{
  "descripcion": "Atacar",
  "requisitoNum": 10,
  "requisitoTxt": "Fuerza",
  "recompensa": {
   "descripcion": "Has ganado una espada."
  }
 }]
}
```

Crea un evento complejo que incluye una entidad y las opciones de acción con su recompensa asociada.

## Actualizar Evento

**Método:** PUT

**Ruta:** /api/eventos/update

**Header:** Authorization: Bearer <token>

**Body (JSON - simplificado):**

```
{
   "id": 14,
 "dificultad": 1,
 "descripcion": "Encuentras una criatura...",
 "entidad": {
  "id": 22 "nombre": "Criatura", "ataque": 50, "defensa": 30, "imagen": "img.png"
 },
 "opciones": [{
  "descripcion": "Atacar",
  "requisitoNum": 10,
  "requisitoTxt": "Fuerza",
  "recompensa": {
   "descripcion": "Has ganado una espada."
  }
 }]
}
```


Actualiza un evento complejo que incluye una entidad y las opciones de acción con su recompensa asociada.

### Eliminar Evento

**Método:** POST

**Ruta:** /api/eventos/eliminar

**Header:** Authorization: Bearer <token>

**Body (JSON):** { "id": 5 }

Elimina un evento por su ID. Solo es accesible para el usuario que lo creó o un administrador.

### Evento Aleatorio

**Método:** GET

**Ruta:** /api/eventos/aleatorio?dificultad={999}&idioma={es}

**Header:** Authorization: Bearer <token>

Devuelve un evento aleatorio segun su dificultad e idioma.

### Obtener Partidas

**Método:** GET

**Ruta:** /api/partidas/all

**Header:** Authorization: Bearer <token>

Devuelve todas las partidas.

### Obtener Partidas Plantilla

**Método:** GET

**Ruta:** /api/partidas/plantilla

**Header:** Authorization: Bearer <token>

Devuelve todas las partidas que son plantilla para uso de todos los usuarios.

### Crear Partida plantilla

**Método:** POST

**Ruta:** /api/partida/insertarPlantilla

**Header:** Authorization: Bearer <token>

**Body (JSON - simplificado):**

```
{
 "nombre": “Mosquetero”,
 "ataque": 47,
 "defensa": 78,
 "inteligencia": 23,
 "suerte": 59,
 "imagen": “Mosquetero.png”,
 "experiencia": 0,
 "nivel": 1,
 "esPlantilla": true,
}
```

Crea una partida plantilla. Solo puede acceder los usuarios que son rol “ADMIN”.

### Crear Partida de Usuario

**Método:** POST

**Ruta:** /api/partida/mi-partida

**Header:** Authorization: Bearer <token>

**Body (JSON - simplificado):**

```
{
 "nombre": “Mosquetero”,
 "ataque": 47,
 "defensa": 78,
 "inteligencia": 23,
 "suerte": 59,
 "imagen": “Mosquetero.png”,
 "experiencia": 0,
```

"nivel": 1
}

Crea una partida para el usuario.

### Obtener Partida del Usuario

**Método:** GET

**Ruta:** /api/partida/mi-partida

**Header:** Authorization: Bearer <token>

Devuelve la partida actual activa del usuario. Si no existe, puede crearse con el endpoint correspondiente.

### Actualizar Partida de Usuario

**Método:** PUT

**Ruta:** /api/partida/mi-partida

**Header:** Authorization: Bearer <token>

**Body (JSON - simplificado):**

{

"id": 241,
"nombre": “Mosquetero”,
"ataque": 47,
"defensa": 78,
"inteligencia": 23,
"suerte": 59,
"imagen": “Mosquetero.png”,
"experiencia": 0,
"nivel": 1
"masAtaque": 15,
"masDefensa": 54
"usuarioID": 23
}

Actualiza la partida actual del usuario.

### Eliminar Partida del Usuario

**Método:** DELETE

**Ruta:** /api/partida/eliminarMiPartida

**Header:** Authorization: Bearer <token>

Borra la partida actual del usuario, incluyendo las referencias a entidades, eventos y puntuaciones asociadas.

### Recuperar Contraseña

**Método:** POST

**Ruta:** /usuarios/recuperacion

**Body (JSON):** { "email": "usuario@x.com" }

Envía un correo con un token temporal de validación para permitir el cambio de contraseña.