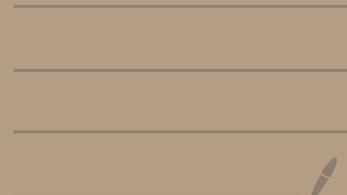


Open API



Ejercicio Open API

GET	/sesiones	<p>Lista de sesiones. Se puede filtrar por los parámetros de la sesión</p> <p>Información de una sesión: - Fecha y horario - Sala - Película - Versión - Formato (3D / 4D / 2D) - Asientos disponibles - Adaptada - Precio - Otros </p>	200 400 si algún parámetro no es válido	
GET	/sesiones/:id		200 404 id no válido	
POST	/sesiones	Crear una sesión	200 400 el JSON no es correcto 401 no estamos autorizados 403 no tenemos permiso	URI de sesión
PUT	/sesiones/:id	Actualizar una sesión	200 404 id no válido	
DELETE	/sesiones/:id	Eliminar una sesión	200 401 no estamos autorizados 403 no tenemos permiso 404 id no válido	
POST	/sesiones/:id/reservar	Comprar una entrada para una sesión - Número de entradas - Asientos a reservar	200 400 el JSON no es correcto 401 no estamos autorizados 403 no tenemos permiso 404 id no válido	Información de la entrada comprada

```

paths:
  /user/{iduser}/book/{idbook}:
    get:
      summary: Obtener información de un libro específico de un usuario
      parameters:
        - name: iduser
          in: path
          required: true
          description: ID del usuario
          schema:
            type: integer
        - name: idbook
          in: path
          required: true
          description: ID del libro
          schema:
            type: integer
      responses:

```

GET + 2 parámetros en el Path

```

openapi: 3.0.3
info:
  title: API de cine
  version: 0.0.1
server:
  url: http://localhost:3000/api
paths:
  /sesiones:
    parameters:
      - $ref: '#/components/parameters/sesiones'
    get:
      summary: Obtiene una lista de sesiones
      description: Este endpoint devuelve una lista con todas las sesiones disponibles y sus detalles.
      responses:
        200:
          description: Lista de sesiones disponibles
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/sesion'
        400:
          description: Alguno parametro es incorrecto
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/error'
    post:
      summary: Crea una nueva sesión
      description: Crea una nueva sesión con los detalles proporcionados en el cuerpo de la solicitud.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/sesion'
      responses:
        201:
          description: Sesión creada con éxito
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/nuevaSesion'
        40XX:
          description: Datos inválidos
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/error'

```

parámetros anteriores GET POST ...

Required

Ejercicio Open API

		Información de una sesión:	
- Fecha y horario			
- Sala			
- Película			
- Versión			
- Formato (3D / 4D / 2D)			
- Asientos disponibles			
- Adaptada			
- Precio			
- Otros			
GET	/sesiones/:id		200 404 id no válido
			201 400 el JSON no es correcto 401 no estamos autorizados 403 no tenemos permiso
POST	/sesiones	Crear una sesión	403 no tenemos permiso
			200 400 el JSON no es correcto 401 no estamos autorizados 403 no tenemos permiso
PUT	/sesiones/:id	Actualizar una sesión	403 no tenemos permiso 404 id no válido
			200 401 no estamos autorizados 403 no tenemos permiso
DELETE	/sesiones/:id	Eliminar una sesión	404 id no válido

```

/sesiones/{id}:
  parameters:
    $ref: '#/components/parameters/id'
  get:
    summary: Obtiene detalles de una sesión específica
    description: Devuelve la información de una sesión con el ID especificado.
    responses:
      200:
        description: Detalles de la sesión
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/sesion'
      404:
        description: ID no válido
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/error'

  put:
    summary: Modifica una sesión
    description: Permite actualizar los detalles de una sesión existente.
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/sesion_actualizar' # Modificar solo algunos parámetros
    responses:
      200:
        description: Sesión actualizada con éxito
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/nuevaSession'
      404:
        description: Sesión no encontrada
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/error'

  delete:
    summary: Elimina una sesión
    description: Borra una sesión del sistema.
    parameters:
      - name: id
        in: path
        required: true
        description: ID de la sesión
        schema:
          type: integer
    responses:
      200:
        description: Sesión eliminada con éxito

```

Diagrama de flujo entre la tabla de métodos y las definiciones de OpenAPI:

- Las acciones de GET, POST, PUT y DELETE en la tabla tienen correspondencia directa con las rutas definidas en la OpenAPI.
- Las columnas "URI de sesión" y "Content" en la tabla se reflejan en los campos "summary" y "content" de las definiciones de OpenAPI.
- Los códigos de respuesta (200, 401, 403) en la tabla se reflejan en los códigos de respuesta (200, 401, 403) de las definiciones de OpenAPI.
- El cuadro rojo que rodea "parameters: \$ref: '#/components/parameters/id'" en la OpenAPI coincide con el cuadro rojo que rodea "GET /sesiones/:id" en la tabla.
- El cuadro rojo que rodea "description: ID no válido" en la OpenAPI coincide con el cuadro rojo que rodea "404 id no válido" en la tabla.
- El cuadro rojo que rodea "summary: Modifica una sesión" en la OpenAPI coincide con el cuadro rojo que rodea "PUT /sesiones/:id" en la tabla.
- El cuadro rojo que rodea "description: Permite actualizar los detalles de una sesión existente." en la OpenAPI coincide con el cuadro rojo que rodea "Actualizar una sesión" en la tabla.
- El cuadro rojo que rodea "requestBody: required: true" en la OpenAPI coincide con el cuadro rojo que rodea "Actualizar una sesión" en la tabla.
- El cuadro rojo que rodea "schema: \$ref: '#/components/schemas/sesion_actualizar' # Modificar solo algunos parámetros" en la OpenAPI coincide con el cuadro rojo que rodea "PUT /sesiones/:id" en la tabla.
- El cuadro rojo que rodea "description: Sesión actualizada con éxito" en la OpenAPI coincide con el cuadro rojo que rodea "200" en la tabla.
- El cuadro rojo que rodea "summary: Elimina una sesión" en la OpenAPI coincide con el cuadro rojo que rodea "DELETE /sesiones/:id" en la tabla.
- El cuadro rojo que rodea "description: Borra una sesión del sistema." en la OpenAPI coincide con el cuadro rojo que rodea "Eliminar una sesión" en la tabla.
- El cuadro rojo que rodea "parameters: name: id in: path required: true description: ID de la sesión schema: type: integer" en la OpenAPI coincide con el cuadro rojo que rodea "DELETE /sesiones/:id" en la tabla.
- El cuadro rojo que rodea "responses: 200: description: Sesión eliminada con éxito" en la OpenAPI coincide con el cuadro rojo que rodea "200" en la tabla.

GET → Parameters

```

paths:
  /usuarios/{id}:
    get:
      summary: Obtener un usuario
      parameters:
        $ref: '#/components/parameters/IDUsuario'
        $ref: '#/components/parameters/IncludePosts'
        $ref: '#/components/parameters/Locale'
      responses:
        '200':
          description: OK

      requestBody:
        description: Descripción opcional del cuerpo de la petición
        required: true | false # Si es obligatorio (por defecto: false)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Algo'
            examples:
              $ref: '#/components/examples/Algo'
            summary: Un ejemplo válido
            value:
              campo: valor
              campo: valor # (opcional) un solo ejemplo
              campo: valor

```

POST → Request Body

PUT



Ejercicio Open API

Components:

POST book $\{id\}$ → in path
GET book? $\{id=x\}$ → in query

```
components:
parameters:
  id:
    name: id
    in: path
    required: true
    description: ID de la sesión
    schema:
      $ref: '#/components/schemas/id'
  date:
    name: date
    in: query
    required: false
    description: Fecha para filtrar las sesiones
    schema:
      $ref: '#/components/schemas/date'
  film:
    name: film
    in: query
    required: false
    description: Película para filtrar las sesiones
    schema:
      $ref: '#/components/schemas/film'
  version:
    name: version
    in: query
    required: false
    description: versión de la película
    schema:
      $ref: '#/components/schemas/version'
```

```
schemas:
  sesion:
    type: object
    properties:
      results:
        $ref: '#/components/schemas/sesiones_array' # Lista de sesiones
      next: # Quieres más sesiones? Acceder a la siguiente parte de la paginación
      anyOf: # Puede ser un string o un objeto
        - type: string
        - type: null
      required:
        - results
        - next
  sesiones_array:
    type: array
    items:
      $ref: '#/components/schemas/sesion'
  sesion:
    type: object
    properties:
      id:
        $ref: '#/components/schemas/id'
      pelicula:
        $ref: '#/components/schemas/pelicula' # TODO
      fecha:
        $ref: '#/components/schemas/date'
      sala:
        type: string
        example: Sala 1
      version:
        type: string
        enum: [original, doblada, subtitulada]
      formato:
        $type: string
        enum: [3D, 4D, 2D]
```

↳ $\{id\}$
↳ $\{pelicula\}$
↳ ...

Components → defino estructura de → parameters : id, date, film, ... →
schemas
↓
type [] recomendado

description
name
required
in : query
schema
\$: ref /schemas/xx

Tipos principales (type)

Estos son los valores que puede tomar type:

- object
- array
- string
- number
- integer
- boolean
- null (solo en OpenAPI 3.1.0 con nullable: true en 3.0)

```
components:
schemas:
  Usuario:
    type: object
    required:
      - nombre
      - edad
    properties:
      nombre:
        type: string
      edad:
        type: integer
      email:
        type: string
        format: email
```

→ Típico

Propiedades útiles para object:

- properties: define los campos internos.
- required: lista de campos obligatorios.
- additionalProperties: si se permiten propiedades extra (true, false, o un schema).
- description: descripción del objeto.
- example: un ejemplo completo.
- deprecated: si el objeto está obsoleto (true/false).

Formatos (format) comunes

Format extiende el significado del tipo base. Algunos ejemplos:

- **Parstring:**
 - email
 - uuid
 - date
 - date-time
 - binary
 - uri
 - hostname
- **Parinteger:**
 - int32
 - int64
- **Parnumber:**
 - float
 - double

Ejercicio Open API

```
schemas:
  sesiones:
    type: object
    properties:
      results:
        $ref: '#/components/schemas/sesiones_array'
      next:
        description: Información para seguir buscando más sesiones
        type: string
        nullable: true # Versión 3.0 OAS
        # Versión 3.1 OAS
      # anyOf:
      # - type: string
      # - type: null
    required:
      - results
      - next
  sesiones_array:
    type: array
    items:
      $ref: '#/components/schemas/sesion'
```

Documentación con Node

OpenAPI - Swagger

```
sh
npm install @openapitools/openapi-generator-cli -g

sh
openapi-generator-cli generate -i openapi.yaml -g nodejs-express-server -o my-api

sh
cd my-api
npm install
npm start
```

```
session:
  allOf:
    - $ref: '#/components/schemas/sesion_actualizar'
    - required:
        - id
        - fecha
        - sala
        - pelicula
        - version
        - formato
        - asientos_disponibles
        - adaptada
        - precio
    id:
      type: integer
    date:
      type: string
      format: date-time
    film:
      type: string
    version:
      type: string
      enum: ["Original", "Doblada", "Subtitulada"]
    errorMessage:
      type: object
      properties:
        code:
          type: integer
        message:
          type: string
```

allOf → Combinar y extender definiciones de modelos

↳ Toma las def. de obj que validan de forma index.
Pero q juntas, forman un obj.

↳ ej: combinar 2 esquemas

```
const express = require('express');
const sesionesController = require('../controllers/SesionesController');

const app = express();
const port = 3000;

app.get('/sesiones', sesionesController.getSesiones);
app.listen(port, () => {
  console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

Luego lo añadimos
a index.js

ej: Crear GET Sesiones

controlador → controller/SesionesController.js
servicio → service/SesionesService.js

1. Crear el servicio en service/SesionesService.js

Aquí defines la lógica para obtener las sesiones.

```
class SesionesService {
  static getSessions() {
    // Aquí irá la lógica de base de datos (por ahora, simulamos con datos estéticos)
    return [
      { id: 1, usuario: "paloma", estado: "activa" },
      { id: 2, usuario: "elgato", estado: "inactiva" }
    ];
  }
}

module.exports = SesionesService;
```

2. Crear el controlador en controllers/SesionesController.js

Aquí maneja la petición HTTP y llama al servicio.

```
process.env.NODE_ENV = 'development';

const SesionesService = require('../service/SesionesService');

app.get('/sesiones', getSessions);

function getSessions(req, res) {
  try {
    const sessions = SesionesService.getSessions();
    res.json(sessions);
  } catch (error) {
    res.status(500).json({ message: 'Error obteniendo sesiones' });
  }
}

module.exports = {
  getSessions
};
```

Después de generar el servidor, tendrá una estructura como esta:

```
my-api/
  controllers/
    DefaultController.js
    AnotherController.js
  service/
    SesionesService.js
    AnotherService.js
  index.js
  routes/
    index.js
    users.js
  package.json
  README.md
```

Donde:

- controllers/ → Contiene los controladores generados.
- service/ → Lógica de negocio separada de los controladores.
- index.js → Punto de entrada del servidor.
- openapi.yaml → Definición OpenAPI utilizada.

http://localhost:3000/api-docs/