

Documentación de Despliegue – CoWork Manager

Joaquín López

Contenido

Documentación de Despliegue – CoWork Manager.....	1
1. Introducción	3
2. Información General de la API.....	3
3. Ejecución del Servicio	3
4. Módulo A – Gestión de Salas.....	4
5. Módulo B – Gestión de Reservas	8
6. conclusión.....	13

1. Introducción

El presente documento describe el funcionamiento y la forma de consumo de la API **CoWork Manager**, desarrollada como parte de la Evaluación Parcial 2 del módulo *Integración de Plataformas*.

La API permite gestionar **salas de reuniones** y **reservas**, siguiendo los principios REST, utilizando formato JSON y ejecutándose en un entorno local (localhost).

Este documento sirve como **contrato de consumo**, es decir, con esta información es posible probar y validar completamente la API sin necesidad de revisar el código fuente.

2. Información General de la API

- **Tipo de API:** RESTful
- **Formato de intercambio:** JSON
- **Servidor:** Localhost
- **Puerto:** 3000
- **Base URL:**

<http://localhost:3000/api/v1>

<http://localhost:3000/docs#/>

3. Ejecución del Servicio

Para ejecutar la API en entorno local se deben instalar las dependencias necesarias y levantar el servidor con Uvicorn.

- `python -m pip install fastapi uvicorn`
- `python -m uvicorn main:app --reload --port 3000`

Una vez iniciado el servidor, la API queda disponible y se puede probar mediante el navegador o utilizando Swagger UI en:

- <http://localhost:3000/docs>

4. Módulo A – Gestión de Salas

Endpoint 1: Listar Salas

Este endpoint permite obtener el listado completo de las salas registradas en el sistema.

- **URL:**

http://localhost:3000/api/v1/salas

- **Método:** GET

- **Payload:** No aplica

The screenshot shows a REST API tool interface with the following details:

- Method:** GET
- Path:** /api/v1/salas (labeled "Listar Salas")
- Parameters:** No parameters
- Buttons:** Execute (blue button) and Clear (white button)
- Responses:** Curl command and Request URL (both show the same curl command: curl -X 'GET' \ 'http://localhost:3000/api/v1/salas' \ -H 'accept: application/json')
- Server response:**
 - Code:** 200
 - Details:** Response body: [] (empty array), Response headers: content-length: 2, content-type: application/json, date: Thu, 22 Jan 2026 05:22:45 GMT, server: unicorn
 - Buttons:** Copy (blue icon) and Download (blue icon)

Endpoint 2: Obtener Detalle de Sala

Permite consultar la información detallada de una sala específica utilizando su identificador.

- **URL:**

http://localhost:3000/api/v1/salas/{sala_id}

- **Método:** GET

- **Payload:** No aplica

GET /api/v1/salas/{sala_id} Obtener Sala

Cancel

Parameters

Name	Description
sala_id * required	integer (path)

1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:3000/api/v1/salas/1' \
-H 'accept: application/json'
```

Request URL

```
http://localhost:3000/api/v1/salas/1
```

Server response

Code	Details
200	Response body

200 Response body

```
{  
  "id": 1,  
  "nombre": "Sala A",  
  "capacidad": 10,  
  "ubicacion": "Norte"  
}
```

Download

Response headers

```
content-length: 61  
content-type: application/json  
date: Thu, 22 Jan 2026 05:24:40 GMT  
server: uvicorn
```

Responses

Endpoint 3: Crear Sala

Este endpoint permite registrar una nueva sala dentro del sistema.

El nombre de la sala no puede repetirse.

- **URL:**

http://localhost:3000/api/v1/salas

- **Método: POST**

The screenshot shows a REST API tool interface for creating a room. At the top, it displays the method (POST) and URL (/api/v1/salas). Below this, there are sections for 'Parameters' (none listed), 'Request body' (required), and 'Responses'. The 'Request body' section contains a JSON schema for creating a room, and the 'Responses' section shows a 400 Bad Request error response with a detailed error message and headers.

Request body (required)

```
{ "nombre": "Sala A", "capacidad": 10, "ubicacion": "Norte" }
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/v1/salas' \
  -H 'accept: application/json' \
  -H 'Content-type: application/json' \
  -d '{
    "nombre": "Sala A",
    "capacidad": 10,
    "ubicacion": "Norte"
}'
```

Request URL

```
http://localhost:3000/api/v1/salas
```

Code	Details
400 <i>Undocumented</i>	Error: Bad Request Response body <pre>{ "detail": { "error": "Nombre duplicado" } }</pre> Response headers <pre>content-length: 39 content-type: application/json date: Thu,22 Jan 2026 05:26:09 GMT server: uvicorn</pre>

Endpoint 4: Eliminar Sala

Permite eliminar una sala del sistema utilizando su ID.

- **URL:**
http://localhost:3000/api/v1/salas/{sala_id}
- **Método:** DELETE
- **Payload:** No aplica

DELETE /api/v1/salas/{sala_id} Eliminar Sala

Parameters

Name	Description
sala_id * required integer (path)	2

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:3000/api/v1/salas/2' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:3000/api/v1/salas/2
```

Server response

Code	Details
204	Response headers

```
content-type: application/json
date: Thu,22 Jan 2026 05:27:10 GMT
server: uvicorn
```

Responses

Code	Description	Links
204	Successful Response	No links

5. Módulo B – Gestión de Reservas

Endpoint 5: Crear Reserva

Este endpoint permite registrar una reserva asociada a una sala existente.

Las fechas deben venir en formato ISO-8601 y la sala debe existir previamente.

- **URL:**

`http://localhost:3000/api/v1/reservas`

- **Método:** POST

Request body required

[Edit Value](#) | [Schema](#)

```
{
  "sala_id": 3,
  "nombre_solicitante": "Joaquin Lopez",
  "fecha_inicio": "2026-01-20",
  "fecha_fin": "2026-01-22"
}
```

[Execute](#) [Clear](#)

Responses

Curl

```
curl -X 'POST' \
'http://localhost:3000/api/v1/reservas' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "sala_id": 3,
  "nombre_solicitante": "Joaquin Lopez",
  "fecha_inicio": "2026-01-20",
  "fecha_fin": "2026-01-22"
}'
```

[Copy](#) [Download](#)

Request URL

```
http://localhost:3000/api/v1/reservas
```

Server response

Code	Details
201	Response body <pre>{ "id": 1, "mensaje": "Creado" }</pre> Copy Download

Curl

```
curl -X 'POST' \
'http://localhost:3000/api/v1/reservas' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "sala_id": 1,
  "nombre_solicitante": "Joaquin Lopez",
  "fecha_inicio": "2026-01-20",
  "fecha_fin": "2026-01-22"
}'
```

[Copy](#)

Request URL

```
http://localhost:3000/api/v1/reservas
```

Server response

Code	Details
400	Error: Bad Request <small>Undocumented</small> Response body <pre>{ "detail": { "error": "sala_id no existe" } }</pre> Copy Download

Response headers

```
content-length: 40
content-type: application/json
date: Thu,22 Jan 2026 05:29:55 GMT
server: uvicorn
```

Responses

Endpoint 6: Historial de Reservas

Permite obtener el historial de reservas registradas en el sistema.

Adicionalmente, se puede filtrar por sala mediante un parámetro de consulta.

- **URL:**

http://localhost:3000/api/v1/reservas

- **Método:** GET

The screenshot shows a REST API documentation interface for the 'Historial de Reservas' endpoint. At the top, the URL is listed as `GET /api/v1/reservas Historial Reservas`. Below this, under 'Parameters', there is a single parameter named 'sala_id' with a value of '3'. There are 'Execute' and 'Clear' buttons below the parameters. In the 'Responses' section, there is a 'Curl' code block for generating a command-line request, a 'Request URL' input field containing `http://localhost:3000/api/v1/reservas?sala_id=3`, and a 'Server response' section. The server response includes a 'Code' of 200, a 'Details' section for the response body showing a JSON array with one element, and a 'Response headers' section listing standard HTTP headers like Content-Length, Content-Type, Date, and Server. Buttons for 'Copy' and 'Download' are also present in the response sections.

GET /api/v1/reservas Historial Reservas

Parameters

Name Description

sala_id
integer | (integer | null)
(query)

3

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:3000/api/v1/reservas?sala_id=3' \
-H 'accept: application/json'
```

Request URL

http://localhost:3000/api/v1/reservas?sala_id=3

Server response

Code Details

200 Response body

```
[{"id": 1, "sala_id": 3, "nombre_solicitante": "Joaquin Lopez", "fecha_inicio": "2026-01-20", "fecha_fin": "2026-01-22"}]
```

Response headers

```
content-length: 112
content-type: application/json
date: Thu,22 Jan 2026 05:30:41 GMT
server: uvicorn
```

Responses

Endpoint 7: Modificar Reserva

Permite modificar parcialmente una reserva existente, específicamente la hora de término (fecha_fin).

- **URL:**

http://localhost:3000/api/v1/reservas/{reserva_id}

- **Método:** PATCH

The screenshot shows a REST API tool interface for modifying a reservation. At the top, it displays the method (PATCH) and URL (`/api/v1/reservas/{reserva_id}`) for the "Modificar Reserva" endpoint. Below this, there are sections for "Parameters" and "Request body". In the "Parameters" section, there is one parameter: `reserva_id` (required, integer, path), with the value set to 1. In the "Request body" section, the content type is set to `application/json`, and the JSON payload is shown as

```
{ "fecha_fin": "2026-01-23" }
```

. Below these sections are buttons for "Execute" and "Clear". Under the "Responses" section, there is a "Curl" command provided for executing the request via the terminal. The "Code" section shows a successful response (200) with a status message ("OK") and a detailed view of the response body and headers. The response body is a JSON object with fields `id`, `mensaje` ("Actualizado"), and `fecha_fin` ("2026-01-23"). The response headers include `Content-Length: 57`, `Content-Type: application/json`, `Date: Thu, 22 Jan 2026 05:31:51 GMT`, and `Server: uvicorn`.

Endpoint 8: Cancelar Reserva

Permite cancelar (eliminar) una reserva existente utilizando su identificador.

- **URL:**

http://localhost:3000/api/v1/reservas/{reserva_id}

- **Método:** DELETE

- **Payload:** No aplica

The screenshot shows a REST API tool interface with the following details:

DELETE /api/v1/reservas/{reserva_id} Cancelar Reserva

Parameters

Name	Description
reserva_id * required	integer (path)
2	

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:3000/api/v1/reservas/2' \
-H 'accept: */*'
```

Request URL

```
http://localhost:3000/api/v1/reservas/2
```

Server response

Code	Details
204	Response headers content-type: application/json date: Thu,22 Jan 2026 05:32:33 GMT server: uvicorn

Responses

Code	Description	Links
204	Successful Response	No links

6. conclusión

La API cumple con los principios REST, utilizando correctamente los verbos HTTP y los códigos de estado según la operación realizada.

Todas las entradas y salidas utilizan formato JSON y se implementan validaciones básicas para asegurar la consistencia de los datos.

Este documento permite probar completamente la API sin ambigüedades, cumpliendo con los requerimientos funcionales y no funcionales definidos en la evaluación.