Documentación de API de Gestión de Restaurante

Daniel Calvar Cruz

13 de noviembre de 2025

Índice

1. Introducción

Volver

Esta API ofrece un servicio CRUD sobre una BBDD con dos colecciones, grupos y usuarios.

La API está construida con Express.js y utiliza MongoDB como base de datos.

2. Estructura

2.1. Rutas Principales

La API está organizada en dos apartados:

- **Grupo**
- **Usuario**

2.2. Modelos de Datos

2.2.1. Modelo Grupo

```
({
  nombre: {
            type: String
          },
  descripcion: {
            type: String
          },
  activo: {
            type: Boolean
          }
});
```

2.2.2. Modelo Usuario

```
({
  nombre: {
            type: String
          },
  apellido1: {
            type: String
          },
  apellido2: {
            type: String
          },
  edad: {
            type: Number
```

```
            }
});
```

3. Base de Datos

Volver

MongoDB (Atlas).

Estructura

- grupos: contiene los estados de los grupos.
- usuarios: contiene los estados de los usuarios.

4. Endpoints

Volver

5. Endpoints

Volver

5.1. Endpoints POST

5.1.1. POST /api/usuarios/create

Descripción: Crea un nuevo usuario en el sistema.
Body:

```
{
"nombre": "string",
"apellido1": "string",
"apellido2": "string",
"edad": "number"
}
```

Respuestas:

- **200**: Usuario creado exitosamente
- **400**: Error en los datos proporcionados
- **500**: Error interno del servidor

5.1.2. POST /api/usuarios/read

Descripción: Busca usuarios según un filtro específico.
Body:

```
{
"filtroKey": "string",
"filtroValue": "string|number"
}
```

Ejemplos de uso:

- Buscar por nombre: `{"filtroKey": "nombre", "filtroValue": "Ana"}`
- Buscar por edad: `{"filtroKey": "edad", "filtroValue": 28}`

Respuestas:

- **200**: Información encontrada con datos
- **404**: Usuarios no encontrados
- **500**: Error interno del servidor

5.1.3. POST /api/grupos/create

Descripción: Crea un nuevo grupo en el sistema.
Body:

```
{
"nombre": "string",
"descripcion": "string",
"activo": "boolean"
}
```

Respuestas:

- **200**: Grupo creado exitosamente
- **400**: Error en los datos proporcionados
- **500**: Error interno del servidor

5.1.4. POST /api/grupos/read

Descripción: Busca grupos según un filtro específico.
Body:

```
{
"filtroKey": "string",
"filtroValue": "string|boolean"
}
```

Ejemplos de uso:

- Buscar por nombre: `{"filtroKey": "nombre", "filtroValue": "Administradores"}`
- Buscar por estado: `{"filtroKey": "activo", "filtroValue": true}`

Respuestas:

- **200**: Información encontrada con datos
- **404**: Grupos no encontrados
- **500**: Error interno del servidor

5.2. Endpoints PUT

Volver

5.2.1. PUT /api/usuarios/update

Descripción: Actualiza la información de un usuario existente.
Body:

```
{
"_id": "string",
"nombre": "string",
"apellido1": "string",
"apellido2": "string",
"edad": "number"
}
```

Respuestas:

- **200**: Usuario actualizado exitosamente
- **400**: Error en los datos proporcionados
- **404**: Usuario no encontrado
- **500**: Error interno del servidor

5.2.2. PUT /api/grupos/update

Descripción: Actualiza la información de un grupo existente.
Body:

```
{
"_id": "string",
"nombre": "string",
"descripcion": "string",
"activo": "boolean"
}
```

Respuestas:

- **200**: Grupo actualizado exitosamente
- **400**: Error en los datos proporcionados
- **404**: Grupo no encontrado
- **500**: Error interno del servidor

5.3. Endpoints DELETE

Volver

5.3.1. DELETE /api/usuarios/delete

Descripción: Elimina un usuario del sistema.
Body:

```
{
"_id": "string"
}
```

Respuestas:

- **200**: Usuario eliminado exitosamente
- **400**: Error en los datos proporcionados
- **404**: Usuario no encontrado
- **500**: Error interno del servidor

5.3.2. DELETE /api/grupos/delete

Descripción: Elimina un grupo del sistema.
Body:

```
{
"_id": "string"
}
```

Respuestas:

- **200**: Grupo eliminado exitosamente
- **400**: Error en los datos proporcionados
- **404**: Grupo no encontrado
- **500**: Error interno del servidor

5.4. Endpoints GET

Volver

5.4.1. GET /api/usuarios/readall

Descripción: Obtiene todos los usuarios del sistema.
Parámetros: Ninguno
Respuestas:

- **200**: Lista de usuarios obtenida exitosamente
- **404**: No hay usuarios registrados
- **500**: Error interno del servidor

5.4.2. GET /api/grupos/readall

Descripción: Obtiene todos los grupos del sistema.
Parámetros: Ninguno
Respuestas:

- **200**: Lista de grupos obtenida exitosamente
- **404**: No hay grupos registrados
- **500**: Error interno del servidor

6. Manejo de Errores

Volver

La API utiliza un sistema consistente de respuestas:

```
{
  "type": "success|failure",
  "message": "string descriptivo",
  "data": "object (opcional)"
}
```

7. Ejemplos de Uso

```
POST http://localhost:5000/api/usuarios/create
Content-Type: application/json

{
  "nombre": "Carlos",
  "apellido1": "Martínez",
  "apellido2": "Rodríguez",
  "edad": 35
}

POST http://localhost:5000/api/grupos/create
Content-Type: application/json

{
  "nombre": "Desarrolladores",
  "descripcion": "Equipo de desarrollo de software",
  "activo": true
}

POST http://localhost:5000/api/usuarios/read
Content-Type: application/json

{
  "filtroKey": "nombre",
  "filtroValue": "Carlos"
}

POST http://localhost:5000/api/grupos/read
Content-Type: application/json

{
  "filtroKey": "activo",
  "filtroValue": true
}
```

8. Consideraciones Técnicas

Volver

9. Códigos de Estado HTTP

- **200 OK**: Operación exitosa
- **400 Bad Request**: Error en los datos enviados
- **404 Not Found**: Recurso no encontrado
- **500 Internal Server Error**: Error del servidor

10. Stack tecnológico

- **Versión Node.js:** 22.14
- **Base de datos:** MongoDB Atlas
- **Dependencias NPM:**
  - **cors:** 2.8.5
  - **dotenv:** 16.4.7
  - **express:** 4.21.1
  - **mongodb:** 6.12.0
  - **mongoose:** 8.13.2
  - **nodemon:** 3.1.10

11. Variables de entorno (.env)

- `PORT`: Puerto en el que se ejecutará el servidor.
- `MONGO_INITDB_ROOT_USERNAME`: Nombre del usuario de la BBDD.
- `MONGO_INITDB_ROOT_PASSWORD`: Password de la BBDD.
- `MONGO_INITDB_DATABASE`: Nombre de la BBDD.