## API de HOTEL

# DECORADORES

@error\_middleware

**Descripción:**

Se ubica delante de las funciones que exigen que se ejecute la misma por si algo falla. Se verifica pidiendo usuario y si no tiene retorna error 404 y 500.

**Parámetros de salida**:

**Status**: 404

**message**: “No existe el valor buscado.”

**Status**: 500

**message**: “Valor invalido, revise la clave.”

**Status**: 500

**message**: “El usuario ya existe.”

@autenticacion

**Descripción:**

Se ubica delante de las funciones que exigen autenticar al usuario (empleado o cliente) para realizar dicha acción verificándose a través de un *payload* de jwt. En caso de no ser autenticado retorna error 400.

**Parámetros de salida**:

**Status**: 400

**message**: “Token invalido.”

**Status**: 400

**message**: “Token inexistente.”

**Status**: 400

**message**: “Token corrupto.”

@esEmpleado

**Descripción:**

Se ubica delante de las funciones que exigen que solo las realice un empleado lo cual se verifica a través del *payload* de jwt. En caso de no ser empleado retorna error 400.

**Parámetros de salida**:

**Status**: 400

**message**: “Sólo los empleados tienen acceso.

**Status**: 400

**message**: “Token invalido.”

**Status**: 400

**message**: “Token inexistente.”

**Status**: 400

**message**: “Token corrupto.”

@esCliente

**Descripción:**

Se ubica delante de las funciones que exigen que solo las realice un cliente lo cual se verifica a través del *payload* de jwt. En caso de no ser cliente retorna error 400.

**Parámetros de salida**:

**Status**: 400

**message**: “Sólo los clientes pueden realizar esta tarea.”

**Status**: 400

**message**: “Token invalido.”

**Status**: 400

**message**: “Token inexistente.”

**Status**: 400

**message**: “Token corrupto.”

@error\_reserva

**Descripción:**

Se ubica delante de las funciones que exigen que solo retorne las habitaciones existentes. En caso de no existir retorna error 404.

**Parámetros de salida**:

**Status**: 404

**message**: “No existe la habitación ingresada.”

@error

**Descripción:**

Se ubica delante de las funciones que exigen que se ingrese en formato fecha un día, mes y año existente de manera DD-MM-AAAA. Si no se ingresa de manera correcta retorna error 404.

**Parámetros de salida**:

**Status**: 404

**message**: “Falla por formato de fecha incorrecto.”

@error\_ingresodatos

**Descripción:**

Se ubica delante de las funciones que exigen que los datos ingresados cumplan ciertos requisitos. Si no se ingresa de manera correcta retorna error 404.

**Parámetros de salida**:

**Status**: 400

**message**: “Error al ingresar los datos, vuelva a intentar.”

# LOGIN

Login

**Ruta**: localhost:5000/login

**Método**: POST

**Descripción:**

Da el token correspondiente al usuario logeado.

**Parámetros de entrada**

**name**: Es el nombre que tendrá el usuario

**clave**: la clave del mismo

**Parámetros de salida**:

Si los datos ingresados son correctos…

Token *ejemplo* {eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiYmFyYmkiLCJuaWNrIjoiYmFyYmkxIiwicm9sIjoiZW1wbGVhZG8ifQ.KXzPTvkjfdyMgGcf2XsfDWW2xYCqSp\_oWaqZ1leZSA4}

Si los datos ingresados son incorrectos (Solo la clave es case sensitive)…

**Status**: 404

**message**: “Credenciales invalidas.”

# WHO

Who/quien

**Ruta**: localhost:5000/login

**Método**: GET

**Descripción:**

Muestra que usuario está conectado.

**Token de autenticación:**

*Auth: bearer token.*

**Parámetros de salida**:

Si los datos ingresados son incorrectos…

**Status**: 404

**message**: “Token invalido.”

# USUARIOS

Crear usuarios

**Ruta**: localhost:5000/usuarios

**Método**: POST

**Descripción:**

Da de alta un usuario (cliente o empleado) para ser utilizado en el sistema

**Parámetros de entrada**

**name**: Es el nombre que tendrá el usuario

**clave**: la clave del mismo

**nick**: el nombre que se mostrará al público.

**rol**: empleado o cliente.

**Parámetros de salida**:

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “usuario creado correctamente.”

En caso de que el usuario ya exista:

**status**: 500

**message**: “El usuario ya existe”

**Decoradores:**

@error\_middleware

@esEmpleado

Listar usuarios

**Ruta**: localhost:5000/usuarios

**Método**: GET

**Descripción:**

Da la lista de todos los usuarios creados anteriormente.

**Token de autenticación:** es la autorización que permite listar los usuarios.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de salida**:

**ID**: Es número que tendrá el usuario.

**name**: es el nombre que tendrá el usuario.

**nick**: el nombre que se mostrará al público.

**rol**: empleado o cliente.

**Decoradores:**

@esEmpleado

Modificar usuarios

**Ruta**: localhost:5000/usuarios/<ID>

**Método**: PUT

**Descripción:**

Modifica cualquier dato del usuario.

**Token de autenticación:** es la autorización que permite modificar los usuarios.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

Ingresar en el link sobre el <ID> el número correspondiente al usuario que deseamos modificar.

**name**: Es el nombre que tendrá el usuario.

**clave**: la clave del mismo.

**nick**: el nombre que se mostrará al público.

**rol**: empleado o cliente.

**Parámetros de salida**:

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “Usuario modificado..”

**status**: 400.

**message**: “El id del usuario no existe.”

**Decoradores:**

@error\_ingresodatos

@esEmpleado

Eliminar usuarios

**Ruta**: localhost:5000/usuarios/<ID>

**Método**: DELETE

**Descripción:**

Elimina cualquier usuario.

**Token de autenticación:** es la autorización que permite eliminar los usuarios.

*Auth: bearer token.*

**Parámetros de entrada**

Ingresar en el link sobre el <ID> el número correspondiente al usuario que deseamos eliminar

**name**: Es el nombre que tendrá el usuario.

**nick**: el nombre que se mostrará al público.

**rol**: empleado o cliente.

**Parámetros de salida**:

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “Usuario eliminado.”

En caso de que el id de usuario no se el correcto:

**status**: 404

**message**:”El id del usuario no existe.”

# HABITACIONES

Crear habitación

**Ruta**: localhost:5000/habitaciones

**Método**: POST

**Descripción:**

Da de alta una habitación.

**Token de autenticación:** es la autorización del usuario empleado que permite crear la habitación.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

**precio**: el valor numérico de la habitación.

**condición**: es el estado de la habitación, activa o inactiva.

**Parámetros de salida**:

Si los datos son correctos…

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “habitación creada correctamente.”

Si los datos son incorrectos…

**status**: 400

**message**: “No puede crear habitación con precio negativo.”

**status**: 400

**message**: “El precio ingresado debe ser de tipo numérico.”

**Decoradores:**

@esEmpleado

Listar todas las habitaciones

**Ruta**: localhost:5000/habitaciones

**Método**: GET

**Descripción:**

Da la lista de todas las habitaciones creadas anteriormente.

**Token de autenticación:** es la autorización del usuario empleado que permite listar las habitaciones.

*Auth: bearer token.*

**Parámetros de salida**:

**ID**: Es número que identificara a cada habitación.

**condición**: es el estado de la habitación.

**precio**: el valor numérico de la habitación.

**reserva**: es el estado en un rango de fechas que el usuario toma esa habitación.

**Decoradores:**

@esEmpleado

Listar habitaciones disponibles

**Ruta**: localhost:5000/habitacionesDisponibles

**Método**: GET

**Descripción:**

Lista todas las habitaciones disponibles.

**Token de autenticación:** es la autorización que permite listar las habitaciones disponibles.

*Auth: bearer token.*

**Parámetros de salida**:

**ID**: Es número que identificara a cada habitación.

**condición**: es el estado de la habitación.

**precio**: el valor numérico de la habitación.

**reserva**: es el estado en un rango de fechas que el usuario toma esa habitación.

**Decoradores:**

@esEmpleado

Traer habitación por id.

**Ruta**: localhost:5000/habitaciones/<ID>

**Método**: GET

**Descripción:**

Da la información de la habitación ingresada por ID.

**Token de autenticación:** es la autorización que permite modificar los usuarios.

*Auth: bearer token.*

**Parámetros de salida**:

**condición**: es el estado de la habitación.

**ID**: Es número que identificara a cada habitación.

**precio**: el valor numérico de la habitación.

**reservas**: es el estado en un rango de fechas que el usuario toma esa habitación.

En caso de ingresar un numero de habitación que no existe:

**status**: 400

**message:** “El ID ingresado es incorrecto.”

**Decoradores:**

@esEmpleado

Modificar precio de la habitación

**Ruta**: localhost:5000/habitaciones/<ID>

**Método**: PUT

**Descripción:**

Modifica el precio de la habitación elegida.

**Token de autenticación:** es la autorización que permite modificar el precio.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

Ingresar en el link sobre el <ID> el número correspondiente a la habitación que queremos modificar.

**precio**: el valor numérico de la habitación.

**Parámetros de salida**:

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “Precio actualizado.”

Al escribir un numero de habitación inexistente:

**status**: 400

**message:** “El ID ingresado es incorrecto.”

Al ingresar un precio negativo:

**status**: 400

**message:** “Precio ingresado incorrecto.”

Al ingresar un string en precio:

**status**: 400

**message:** “Error al ingresar los datos, vuelva a intentar.”

**Decoradores:**

@error\_ingresosdatos

@esEmpleado

Modificar condición de la habitación

**Ruta**: localhost:5000/habitaciones/disponibilidad/<ID>

**Método**: PUT

**Descripción:**

Modifica la condición de la habitación elegida.

**Token de autenticación:** es la autorización que permite modificar la condición.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

Ingresar en el link sobre el <ID> el número correspondiente a la habitación que queremos modificar.

**condición**: es el estado de la habitación.

**Parámetros de salida**:

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “Condición actualizada.”

Al ingresar un numero de habitación inexistente:

**status**: 400

**message:** “El ID ingresado es incorrecto.”

**Decoradores:**

@error\_ingresodatos

@esEmpleado

# RESERVAS

Crear reserva de una habitación

**Ruta**: localhost:5000/reservas

**Método**: POST

**Descripción:**

Da de alta una reservación.

**Token de autenticación:** es la autorización que permite crear la reserva.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

**habitacion**: ID de la habitación que se va a reservar.

**fechaEntrada**: fecha de entrada a la habitación reservada. (AAAA-MM-DD)

**fechaSalida**: fecha de salida de la habitación reservada. (AAAA-MM-DD)

**Parámetros de salida**:

Si la condición de la habitación pedida es inactiva…

**status**: 404

**message**: “La habitación se encuentra inactiva hasta nuevo aviso.”

Si la fecha solicitada no está disponible en la habitación solicitada…

**status**: 404

**message**: “La habitación no se encuentra disponible en al menos una de las fechas elegidas.”

Si las fechas (Fecha de entrada y luego salida) son la primera mayor que la segunda…

**status**: 400

**message**: “Falla por incongruencia de fechas.”

Si el formato de la fecha es incorrecto(si esta mal escrita la fecha o se escribe un string)…

**status**: 400

**message**: “Falla por formato de fecha incorrecto.”

Si la reservación es correcta…

**status**: indica el número del estado, 200 significa que salió todo bien.

**message**: “Reserva realizada.”

**Decoradores:**

@error

@esCliente

@error\_reserva

Listar reservas

**Ruta**: localhost:5000/reservas

**Método**: GET

**Descripción:**

Da la lista de todas las reservas

**Token de autenticación:** es la autorización que permite listar las reservas.

*Auth: bearer token.*

**Parámetros de salida**:

**cliente**: es el ID del cliente que hizo la reserva.

**fechaEntrada**: fecha de entrada a la habitación reservada.

**fechaSalida**: fecha de salida de la habitación reservada.

**habitacion**: id de la habitación reservada.

**id**: numero de la reserva.

**Decoradores:**

@esEmpleado

Traer reserva por id.

**Ruta**: localhost:5000/reservas/<ID>

**Método**: GET

**Descripción:**

Da información de la reserva ingresada por ID.

**Token de autenticación:** es la autorización que permite mostrar la reserva.

*Auth: bearer token.*

**Parámetros de salida**:

**cliente**: es el ID del cliente que hizo la reserva.

**fechaEntrada**: fecha de entrada a la habitación reservada. (AAAA-MM-DD)

**fechaSalida**: fecha de salida de la habitación reservada. (AAAA-MM-DD)

**habitacion**: id de la habitación reservada.

**id**: numero de la reserva.

Si el id de habitación es inexistente…

**status**: 400

**message:** “El ID ingresado es incorrecto.”

**Decoradores:**

@esEmpleado

# BÚSQUEDA

Buscar habitación por menor precio

**Ruta**: localhost:5000/habitacion/precio/<precio>

**Método**: GET

**Descripción:**

Trae las habitaciones menores al precio numérico ingresado en <PRECIO>.

**Token de autenticación:** es la autorización que permite traer las habitaciones por el menor precio.

*Auth: bearer token.*

**Parámetros de salida**:

**ID**: Es número que identificara a cada habitación.

**condición**: es el estado de la habitación.

**precio**: el valor numérico de la habitación.

**Decoradores:**

@esCliente

Buscar habitación por rango de fecha

**Ruta**: localhost:5000/buscarhabitaciones/rango

**Método**: GET

**Descripción:**

Trae las habitaciones reservadas en un rango de fecha.

**Token de autenticación:** es la autorización que permite traer las habitaciones reservadas en un rango.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

**fechaEntrada**: fecha de entrada a la habitación reservada. (AAAA-MM-DD)

**fechaSalida**: fecha de salida de la habitación reservada. (AAAA-MM-DD)

**Parámetros de salida**:

**id**: Numero de la habitación reservada.

Si los datos ingresados son incorrectos…

**status**: 400

**message**: “Falla por incongruencia de fechas.”

Si el formato de fecha es incorrecto…

**status**: 400

**message**: “Falla por formato de fecha incorrecto.”

**Decoradores:**

@error

@esCliente

Buscar habitación por día disponibles y ocupadas (por separado)

**Ruta**: localhost:5000/buscarhabitaciones/dia

**Método**: GET

**Descripción:**

Trae las habitaciones reservadas en un día especifico.

**Token de autenticación:** es la autorización que permite traer las habitaciones reservadas en un día especifico.

*Auth: bearer token.*

*Headers: -KEY: Content-Type*

*-VALUE: Application/json*

*Body: -Raw -Json*

**Parámetros de entrada**

**fecha**: (AAAA-MM-DD)**.**

**Parámetros de salida**:

**Habitaciones disponibles:** Lista de las habitaciones disponibles con ID.

**Habitaciones ocupadas:** Lista de las habitaciones ocupadas con ID.

Si el formato de fecha es incorrecto…

**status**: 400

**message**: “Falla por formato de fecha incorrecto.”

**Decoradores:**

@error

@esCliente