ESCUELA SUPERIOR DE INGENIERÍA

Programación en Internet

Grado en Ingeniería Informática

Tutorial Servicio Web y jQuery

de FitnessZone

Autores:

Carlos Antonio Cortés Lora

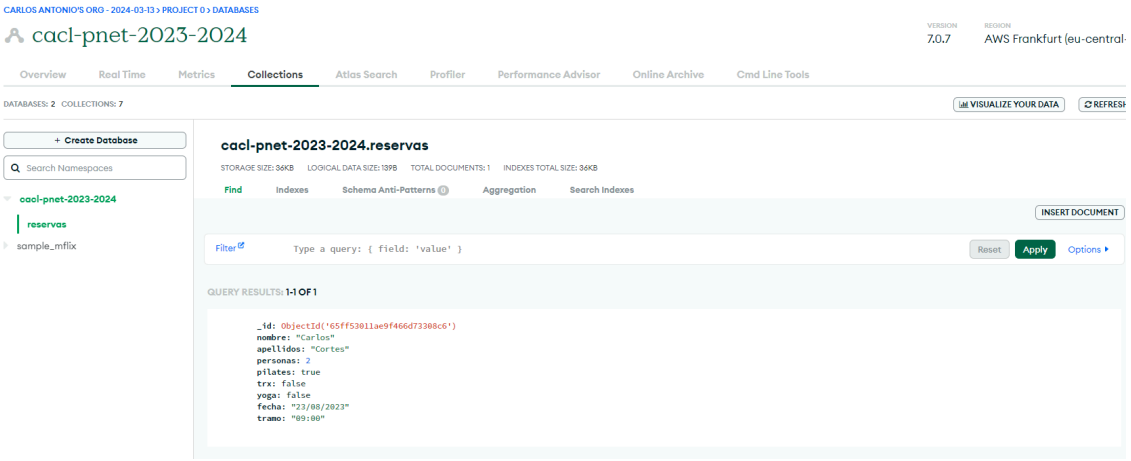
Roberto Rivero Díaz

Índice General

1. Descripción de la entidad

En el análisis de la creación de nuestra API, hemos llegado a la conclusión de que podemos llegar a tener varios recursos y para evitar sobrecargar el index, hemos usado los routers para organizar y empaquetar dichos recursos. Para ello, hemos creado una nueva carpeta llamada *routes,* en la que guardaremos los archivos de nuestros recursos, además de realizar las modificaciones necesarias en el código ya existente.

En adición a lo anterior y para obtener una mayor persistencia de los datos que vamos a tratar, usaremos una base de datos de tipo NoSQL llamada MongoDB. Hemos tenido que crear una base de datos(*cacl-pnet-2023-2024*) y en su interior almacenar un collection *reservas*.



Como podemos observar en la imagen anterior, esta collection está formada por datos de tipo reserva, que contiene estos atributos:

- Id: Un identificador automático que genera MongoDB
- nombre: nombre de la persona que realiza la reserva
- apellidos: apellidos de la persona que realiza la reserva
- tipo de sala: representado mediante los booleanos true o false
- fecha: fecha de la reserva
- tramo: tramo horario, donde se especifica la hora de inicio

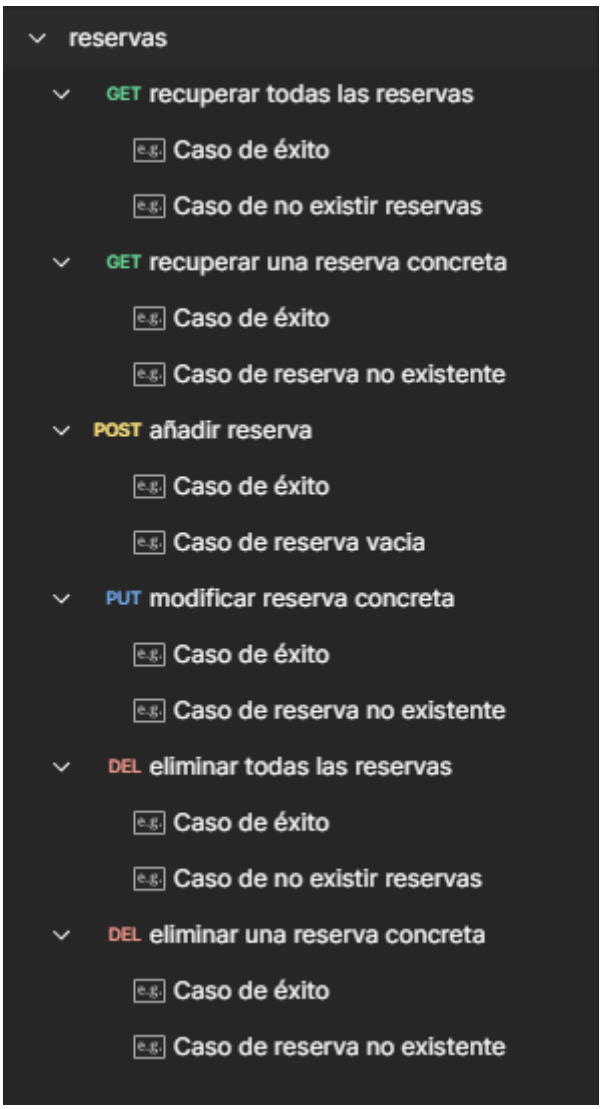
Todas las reservas son de una hora de duración, así que para saber el tramo horario únicamente necesitamos saber la hora de inicio.

```
_id: ObjectId('65ff53011ae9f466d73308c6')
nombre: "Carlos"
apellidos: "Cortes"
personas: 2
pilates: true
trx: false
yoga: false
fecha: "23/08/2023"
tramo: "09:00"
```

2. Consumición de la API mediante Postman

Para poder consumir los servicios de la API se ha utilizado los servicios de Postman, además de servirnos para probar la propia API.

Para ello, hemos creado una collection llamada *reservas* con todos los métodos que deben de tener la API, además de incluir casos de éxito y de error por cada uno de ellos.



Llegados a este punto, podemos distinguir entre los métodos que no utilizan los identificadores y los que sí. Estos últimos(modificar una reserva concreta,

eliminar una reserva concreta y recuperar una reserva concreta) al depender de un ID concreto, hemos tenido que ir poniendo en el link del método el ID (generado automáticamente) en específico con el que queremos operar.

3. Cómo invocar la API RESTful con jQuery

Para invocar la API RESTful con jQuery desde las funciones JavaScript ya creadas vamos a usar AJAX. En concreto, la función $.ajax(), que nos permite enviar una solicitud HTTP a nuestro servidor y ejecutar una función callback() como respuesta.

Para ello vamos a usar una de las funciones como ejemplo:

```
//Recuperar una única reserva existente por ID.
function getReserva(reservaId) {
    let myUrl = "http://localhost:8080/reservas/" + reservaId;
    $.ajax({
        type: "GET",
        dataType: "json",
        url: myUrl,
        success: function(data) {
            mostrarTabla(data);
        },
        error: function(res) {
            let mensaje = JSON.parse(res.responseText);
            alert("ERROR: " + mensaje.msg);
        }
    });
}
```

La función $.ajax, tiene los siguientes parámetros:

- type: tipo de petición, en este caso GET
- datatype: tipo de dato que esperamos recibir tras invocación
- contentType: tipo de dato que enviamos con la innovación(no especificado en este ejemplo, debido a su naturaleza)
- url: la ruta del recurso al que queremos acceder(especificamos un ID)
- success y error: para tratar los casos de manera distinta

3.1. getAllReservas()

Esta función realiza una solicitud GET al servidor para obtener todas las reservas. Utiliza jQuery para realizar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se reciben los datos del servidor de tipo JSON, invoca la función mostrarTabla(data) para mostrar las reservas en la página HTML.

```
//Recuperar todas las reservas.
function getAllReservas() {
    let myUrl = "http://localhost:8080/reservas";
    $.ajax({
        type: "GET",
        dataType: "json",
        url: myUrl,
        success: function(data) {
            mostrarTabla(data);
            //$("#resPelicula").html(JSON.stringify(data));
        },
        error: function(res) {
            console.error("ERROR:", res.status, res.statusText);
        }
    });
}
```

3.2. getReserva(reservaId)

Esta función realiza una solicitud GET al servidor para obtener una reserva específica según su ID. Utiliza jQuery para realizar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se reciben los datos del servidor, invoca la función mostrarTabla(data) para mostrar la reserva en la página HTML.

```
//Recuperar una única reserva existente por ID.
function getReserva(reservaId) {
    let myUrl = "http://localhost:8080/reservas/" + reservaId;
    $.ajax({
        type: "GET",
        dataType: "json",
        url: myUrl,
        success: function(data) {
            mostrarTabla(data);
            //$("#resPelicula").html(JSON.stringify(data[0]));
        },
        error: function(res) {
            let mensaje = JSON.parse(res.responseText);
            alert("ERROR: " + mensaje.msg);
        }
    });
}
```

3.3. postReserva()

Esta función construye un objeto de datos con la información ingresada en el formulario de agregar reserva y realiza una solicitud POST al servidor para agregar una nueva reserva. Utiliza jQuery para enviar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se completa la operación en el servidor, invoca la función getAllReservas() para actualizar la

lista de reservas en la página HTML.

```
//Insertar una nueva reserva
function postReserva() {
    var data = {
        "nombre": $('#nombre').val(),
        "apellidos": $('#apellidos').val(),
        "pilates": $('#sala').val() === 'Pilates',
        "trx": $('#sala').val() === 'Trx',
        "yoga": $('#sala').val() === 'Yoga',
        "personas": $('#personas').val(),
        "fecha": $('#fecha').val(),
        "tramo": $('#tramo').val()
    };

    $.ajax({
        type: "POST",
        url: "http://localhost:8080/reservas",
        contentType: "application/json",
        dataType: "text",
        data: JSON.stringify(data),
        success: function(data) {
           //$("#resReserva").html(data);
           getAllReservas();
        },
        error: function(res) {
            alert("ERROR: " + res.statusText);
        }
    })
}
```

3.4. putReserva(reservaId)

Esta función construye un objeto de datos con la información ingresada en el formulario de modificar reserva y realiza una solicitud PUT al servidor para actualizar una reserva existente según su ID. Utiliza jQuery para enviar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se completa la operación en el servidor, invoca la función getAllReservas() para

actualizar la lista de reservas en la página HTML.

```
//Actualizar una reserva existente por ID.
function putReserva(reservaId) {
    var data = {
        "nombre": $('#nombre2').val(),
        "apellidos": $('#apellidos2').val(),
        "pilates": $('#sala2').val() === 'Pilates',
        "trx": $('#sala2').val() === 'Trx',
        "yoga": $('#sala2').val() === 'Yoga',e
        "personas": $('#personas2').val(),
        "fecha": $('#fecha2').val(),
        "tramo": $('#tramo2').val(),
    };
    var myUrl = "http://localhost:8080/reservas/" + reservaId;
    $.ajax({
        type: "PUT",
        dataType: "json",
        url: myUrl,
        contentType: "application/json",
        data: JSON.stringify(data),
        success: function(data) {
            //$("#resReserva").html(JSON.stringify(data));
            getAllReservas();
        },
        error: function(res) {
            let mensaje = JSON.parse(res.responseText);
            alert("ERROR: " + mensaje.msg);
        }
    });
}

```

3.5. deleteAllReservas()

Esta función realiza una solicitud DELETE al servidor para eliminar todas las reservas. Utiliza jQuery para enviar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se completa la operación en el servidor, invoca la función getAllReservas() para actualizar la lista de

reservas en la página HTML.

```
//Eliminar todas las reservas.
function deleteAllReservas() {
    var myUrl = "http://localhost:8080/reservas/";
    $.ajax({
        type: "DELETE",
        dataType: "json",
        url: myUrl,
        success: function(data) {
            getAllReservas();
        },
        error: function(res) {
            let mensaje = JSON.parse(res.responseText);
            alert("ERROR: " + mensaje.msg);
        }
    });
}

```

3.6. deleteReserva(reservaId)

Esta función realiza una solicitud DELETE al servidor para eliminar una reserva específica según su ID. Utiliza jQuery para enviar una petición AJAX al endpoint correspondiente de la API RESTful. Una vez que se completa la operación en el servidor, invoca la función getAllReservas() para actualizar la

lista de reservas en la página HTML.

```
//Eliminar una unica reserva existente por ID.
function deleteReserva(reservaId) {
    var myUrl = "http://localhost:8080/reservas/" + reservaId;
    $.ajax({
        type: "DELETE",
        dataType: "json",
        url: myUrl,
        success: function(data) {
            //$("#resReserva").html("Reserva eliminada correctamente");
            getAllReservas();
        },
        error: function(res) {
            let mensaje = JSON.parse(res.responseText);
            alert("ERROR: " + mensaje.msg);
        }
    });
}
```

4. Cómo se listan las reservas en la página web HTML

Se invoca a la operación GetAllReservas(), explicada anteriormente, donde recibe en formato JSON, todas las reservas de la base de datos.

A continuación, las reservas se listan en la página web HTML utilizando la función mostrarTabla(data). Esta función recibe los datos de las reservas como parámetro y genera una tabla HTML con la información correspondiente, como el ID de la reserva, el nombre, los apellidos, el número de personas, el tipo de sala, la fecha y la hora inicio. La tabla generada se inserta en un elemento específico del HTML, como , para mostrar las reservas en la página.

```
//Funcion para mostrar una tabla con los data
function mostrarTabla(data){
    var tabla = "<table><thead><tr><th>ID</th><th>Nombre</th><th>Apellidos</th><th>Número de personas</th><th>Tipo</th><th>Fecha</th><th>Hora de inicio</th></tr></thead><tbody>";
    for (var i = 0; i < data.length; i++) {
        bool = true;
        var tipoSala = '';
        if (data[i].pilates === true) {
            tipoSala = 'Pilates';
        } else if (data[i].trx === true) {
            tipoSala = 'TRX';
        } else if (data[i].yoga === true) {
            tipoSala = 'Yoga';
        }
        tabla += "<tr><td>" + data[i]._id + "</td><td>" + data[i].nombre + "</td><td>" + data[i].apellidos + "</td><td>" + data[i].personas + "</td><td>" + tipoSala + "</td><td>" + data[i].fecha + "</td><td>" + data[i].tramo + "</td></tr>";
    }
    tabla += "</tbody></table>";
    $("#TablaReservas").html(tabla);
}
```

5. Cómo se ejecuta la operación dinámica DELETE ONE

Cada vez que se selecciona una operación dinámica, como "VER TODAS LAS RESERVAS" o "ENVIAR RESERVA", se invoca la función JavaScript correspondiente que realiza una petición AJAX a la API RESTful utilizando jQuery.

Operación "BORRAR RESERVA" de las sentencias específicas:

a. Se ingresa el ID de la reserva a eliminar en el formulario correspondiente.
b. Se hace clic en el botón "BORRAR RESERVA", lo que activa la función deleteReserva().
c. La función deleteReserva() realiza una solicitud DELETE al servidor para eliminar la reserva con el ID especificado.
d. El servidor procesa la solicitud y elimina la reserva de la base de datos.
e. Después de completarse la operación en el servidor, se invoca la función getAllReservas() para actualizar la lista de reservas en la página HTML.

6. Operaciones Adicionales

6.1. Operación "VER RESERVA" de sentencias específicas

- Se ingresa la información de la reserva, es decir, el ID.
- El servidor procesa la solicitud y devuelve los datos de la reserva.
- Se invoca la función mostrarTabla(data) para mostrar la reserva concreta en la página HTML.

6.2. Operación "ENVIAR RESERVA" al hacer una nueva reserva

- Se ingresa la información de la reserva en el formulario correspondiente.
- Se hace clic en el botón "ENVIAR RESERVA", lo que activa la función postReserva().
- La función postReserva() construye un objeto de datos con la información del formulario y realiza una solicitud POST al servidor para agregar la reserva.
- El servidor procesa la solicitud y agrega la reserva a la base de datos.
- Después de completarse la operación en el servidor, se invoca la función getAllReservas() para actualizar la lista de reservas en la página HTML.

6.3. Operación "MODIFICAR RESERVA" al modificar una reserva

- Se ingresa la información actualizada de la reserva en el formulario correspondiente.
- Se hace clic en el botón "MODIFICAR RESERVA", lo que activa la función putReserva().
- La función putReserva() construye un objeto de datos con la información del formulario y realiza una solicitud PUT al servidor para actualizar la reserva.
- El servidor procesa la solicitud y actualiza la reserva en la base de datos.
- Después de completarse la operación en el servidor, se invoca la función getAllReservas() para actualizar la lista de reservas en la página HTML.

6.4. Operación "BORRAR TODAS LAS RESERVAS" de las sentencias globales

- Se hace clic en el botón "BORRAR TODAS LAS RESERVAS", lo que activa la función deleteAllReservas().

- La función deleteAllReservas() realiza una solicitud DELETE al servidor para eliminar todas las reservas.
- El servidor procesa la solicitud y elimina todas las reservas de la base de datos.
- Después de completarse la operación en el servidor, se invoca la función getAllReservas() para actualizar la lista de reservas en la página HTML.