API ALLROUTES

DIEGO FREILE GARCÍA

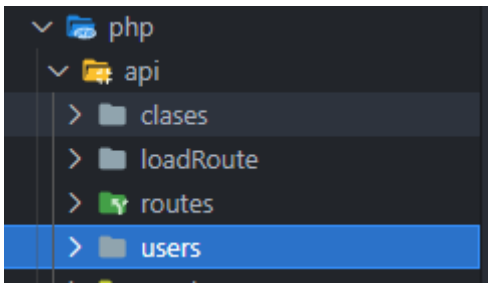
Índice

Introducción

En el siguiente documento se va a explicar de forma detallada como he realizado la API de mi aplicación web “AllRoutes”, que su principal función, es la de gestionar la tanto la parte de usuarios, como la parte de las rutas. Explicaré el funcionamiento de las funciones desarrolladas, indicando su utilidad, los parámetros de entrada necesarios, valores devueltos dependiendo de las respuestas y ejemplos del funcionamiento.

Árbol de directorios

Esta API la he organizado en los siguientes directorios:



Clases: contiene el archivo “conexión.php” el cual contiene la conexión a la base de datos y una función para generar un token único a un usuario que haya iniciado sesión.

```php
<?php
    require_once "../vendor/autoload.php";
    use Firebase\JWT\JWT;

    class Conexion extends mysqli {
        private $host = "localhost";
        private $db = "allroutes";
        private $user = "allroutes";
        private $pass = "allroutes";

        public function __construct()
        {
            try {
                parent::__construct($this->host, $this->user, $this-
>pass, $this->db);
            } catch (mysqli_sql_exception $e) {
                echo "ERROR: {$e->getMessage()}";
                // header("HTTP/1.1 400 Bad Request");
                exit;
            }
        }

        /*####################################
```

```
        GENERAR TOKEN DE AUTENTICACIÓN
        ####################################*/
        static public function jwt($username) {
            $time = time();
            $key = 'This 1s S3cr3T';
            $token = array(
                "iat" => $time, //Tiempo en que inicia el token
                "exp" => $time + (60*60*24), // Tiempo en que expirará el
token(1 dia)
                "username" => $username
            );

            $jwt = JWT::encode($token, $key, 'HS256');

            return $jwt;
        }
    }
?>
```

LoadRoute: contiene un archivo index.php con la función para subir una ruta a la base de datos.

Routes: contiene un archivo index.php con la función para obtener todas las rutas y los parámetros para filtrar por nombre, distancia mínima y distancia máxima.

Users: contiene un archivo index.php con las funciones básicas para gestionar las peticiones del usuario.

Parte de los usuarios

Esta parte de la API gestiona las peticiones básicas de los usuarios (registro, inicio de sesión, edición de datos y eliminación de la cuenta).

La URL base para realizar las llamadas seria la siguiente:

- **http://localhost/proyecto-allroutes/php/api/users/**

Registro de un usuario

Función para registrar un usuario en la base de datos:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $json = file_get_contents('php://input');
    $user = json_decode($json);

    /* REGISTRAR USUARIO */
    if (
```

```
        isset($user->username) && isset($user->fullname) && isset($user->email) &&
        isset($user->pass) && isset($user->height) && isset($user->weight) &&
        isset($user->birthday) && isset($user->actividades)
    ) {

        $fullname = $user->fullname;
        $username = $user->username;
        $email = $user->email;
        $pass = $user->pass;
        $height = $user->height;
        $weight = $user->weight;
        $birthday = $user->birthday;
        $activities = implode(",", $user->actividades);

        $passHash = hash("sha512", $pass);

        $sql = "INSERT INTO users (username, fullname,  email, pass, height,
                weight, birthday, activities) VALUES ('$username', '$fullname', '$email',
                '$passHash', '$height', '$weight', '$birthday', '$activities')";

        $sql2 = "SELECT username FROM users WHERE username='$username'";

        try {
            $usuarios = $con->query($sql2)->fetch_all(MYSQLI_ASSOC);
            if (count($usuarios) > 0) {
                header("HTTP/1.1 409 Username exist");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => false,
                    'msg' => "El nombre de usuario ya existe"
                ]);
            } else {
                $con->query($sql);
                header("HTTP/1.1 201 Created");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => true,
                    'msg' => "El usuario se ha creado correctamente"
                ]);
            }
        } catch (mysqli_sql_exception $e) {
            header("HTTP/1.1 400 Bad Request");
            header("Content-Type: application/json");
            echo json_encode([
```

```
                'success' => false,
                'msg' => "Alguno de los campos está vacío"
            ]);
        }
```

En primer lugar, se comprueba que se haya recibido el siguiente cuerpo del body:

```
    {
        "username": *nombre de usuario
        "fullname": *nombre completo
        "email": *correo electrónico
        "pass": *contraseña
        "height": *altura
        "weight": *peso
        "birthday": *fecha de nacimiento
        actividades, *actividades preferidas
    };
```

Si se han recibido, se almacenaría los valores en variables, y a continuación se declararía la consulta para insertar en la tabla “users” los valores recibidos anteriormente y una pequeña consulta que comprueba que el “username” no esté repetido en la base de datos.

A continuación, en un bloque “try catch” se comprueba el número de usuarios encontrados en la anterior consulta, si es mayor que 0 la respuesta sería “409 Username exists” y nos devolvería un mensaje de error de que el usuario ya existe. En cambio, si el número de usuarios encontrados es 0 se ejecuta la consulta de inserción del usuario y la respuesta sería exitosa “HTTP/1.1 201 Created” con el mensaje de usuario creado correctamente. Si hubiera algún fallo de otro tipo la respuesta sería “HTTP:/1.1 400 Bad Request” con un mensaje de error al registrar al usuario.

Ejemplo de una llamada:

```
let datosUsuario = {
        "username": `${username.value}`,
        "fullname": `${fullname.value}`,
        "email": `${email.value}`,
        "pass": `${pass.value}`,
        "height": `${estatura.value}`,
        "weight": `${peso.value}`,
        "birthday": `${fechaNac.value}`,
        actividades,
    };

    let url = "http://localhost/proyecto-allroutes/php/api/users/";

    fetch(url, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json;charset=utf-8'
```

```
        },
        body: JSON.stringify(datosUsuario)
    })
        .then(response => {
            return response.json();
        })
        .then(data => {
            if (data['success']) {
                console.log(data['msg']);
            } else {
                console.log(data['msg']);
            }
        })
```

Iniciar sesión

Función para iniciar sesión en la aplicación:

```
} elseif (isset($user->username) && isset($user->pass)) {


        $username = $user->username;
        $pass = $user->pass;

        $sql = "SELECT id, username, pass FROM users WHERE
username='$username'";

        $jwt = Conexion::jwt($username);

        try {
            $usuario = $con->query($sql)->fetch_all(MYSQLI_ASSOC);
            /* var_dump(hash('sha512', $pass));
            var_dump($usuario[0]['pass']); */
            if (hash('sha512', $pass) == $usuario[0]['pass']) {
                header("HTTP/1.1 200 OK");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => true,
                    'msg' => "Se ha iniciado sesion",
                    'id' => $usuario[0]['id'],
                    'username' => $usuario[0]['username'],
                    'token' => $jwt
                ]);
            } else {
                header("HTTP/1.1 400 Bad Request");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => false,
```

```
                    'msg' => "Datos de sesión incorrectos"
                ]);
            }
        } catch (mysqli_sql_exception $e) {
            header("HTTP/1.1 404 Not Found");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "No se encuentra el usuario"
            ]);
        }
    } else {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Alguno de los campos esta vacío"
        ]);
    }
    exit;
```

Se comprueba que se haya recibido el siguiente cuerpo del mensaje:

```
    {
        "username": *nombre de usuario
        "pass": *contraseña
    };
```

Si se han recibido se declara la consulta para obtener el id, el nombre de usuario y la contraseña del usuario recibido en el cuerpo del mensaje, a continuación se genera un token JWT con la función mencionada anteriormente en la clase “conexión” y en un bloque try catch se comprueba que las contraseñas sean iguales, si lo son la respuesta seria “HTTP/1.1 200 OK” y devolvería un JSON con un mensaje de se ha iniciado sesión, el id del usuario, su nombre de usuario y el token generado. Si las contraseñas no fueran iguales la respuesta seria “HTTP/1.1 404 Not Found” y devolvería un JSON con un mensaje de que no se encuentra al usuario.

Ejemplo de una llamada:

```
let url = "http://localhost/proyecto-allroutes/php/api/users/";

    fetch(url, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(datosUsuario)
    })
```

```
        .then(response => {
            return response.json();
        })
        .then(data => {
            if (data['success']) {
                localStorage.setItem('token', `${data['token']}`);
                localStorage.setItem('username', `${data['username']}`)
                localStorage.setItem('id', `${data['id']}`)
                location.reload();
            } else {
                console.log(data['msg']);
            }
        })
```

Obtener usuarios

Función para obtener los campos de la tabla usuarios:

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    try {
        $sql = "SELECT * FROM users WHERE 1";
        if (isset($_GET['id'])) {
            $id = $_GET['id'];
            $sql .= " AND id='$id'";
        } elseif (count($_GET) > 0) {
            header("HTTP 400 Bad Request");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "Error se encuentran mas usuarios con ese id"
            ]);
        }

        $usuarios = $con->query($sql)->fetch_all(MYSQLI_ASSOC);
        $usuarios[0]['activities'] = explode(",",
$usuarios[0]['activities']);
        if ($usuarios != null) {
            header("HTTP/1.1 200 OK");
            header("Content-Type: application/json");
            echo json_encode($usuarios[0]);
        } else {
            header("HTTP/1.1 404 Not Found");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "No se encuentra al usuario"
            ]);
        }
```

```
    } catch (mysqli_sql_exception $e) {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Error al obtener los usuarios"
        ]);
    }
    exit;
}
```

En un bloque try catch se declara la sentencia sql para obtener todos los campos de la tabla “users”. A continuación se comprueba si en la url base se le haya pasado como parámetro un “id” si es así se le agregar a la sentencia sql que filtre por el “id” indicado.

Url con el parámetro id:

- http://localhost/proyecto-allroutes/php/api/users?id=identificador

Lo siguiente es ejecutar la consulta y comprobar que el array “usuarios” no sea nulo, si es así se obtiene la respuesta “HTTP/1.1 200 OK” y se devuelve un JSON con los campos del usuario. Si en cambio el array usuarios es nulo se obtiene una respuesta “HTTP/1.1 404 Not Found” y devuelve un JSON son el mensaje de que no se han encontrado usuarios. Si por alguna razón hay otro tipo de error la respuesta sería “HTTP/1.1 400 Bad Request” y devolvería un JSON con el mensaje de error al obtener los usuarios.

Ejemplo de una llamada:

```
let url = `http://localhost/proyecto-
allroutes/php/api/users?id=${identificador}`;

    fetch(url, {
        method: 'GET',
        headers: {
            'Content-Type': 'application/json;charset=utf-8'
        }
    })
    .then(response => {
        return response.json();
    })
    .then(data => {
        if(data) {
            let fullname = document.querySelector('#fullname');
            let username = document.querySelector('#user');
            let email = document.querySelector('#email');
            let estatura = document.querySelector('#estatura');
            let peso = document.querySelector('#peso');
            let fechaNac = document.querySelector('#fechaNac');
            let pass = document.querySelector('#pass');
            let senderismo = document.querySelector('#senderismo');
```

```
            let bicicleta = document.querySelector('#bicicleta');
            let running = document.querySelector('#running');
            let alpinismo = document.querySelector('#alpinismo');

            console.log(data);

            fullname.value = data['fullname'];
            username.value = data['username'];
            email.value = data['email'];
            estatura.value = data['height'];
            peso.value = data['weight'];
            fechaNac.value = data['birthday'];
            pass.value = data['pass'];
            senderismo.checked = data['activities'].some((item) => item
== 'senderismo');
            bicicleta.checked = data['activities'].some((item) => item ==
'bicicleta');
            running.checked = data['activities'].some((item) => item ==
'running');
            alpinismo.checked = data['activities'].some((item) => item ==
'alpinismo');
        } else {
            console.log(data['msg']);
        }

    })
```

Función para validar un token

Función:

```
function isValidToken($token, $username)
{
    global $key;
    try {
        $json = JWT::decode($token, new Key($key, 'HS256'));
        return ($json->username == $username);
    } catch (Exception $e) {
        return false;
    }
}
```

Recibe como parámetros el token y el nombre de usuario, y internamente decodifica el token para extraer el usuario relacionado con él y lo compara para ver si realmente ese token es de ese usuario, si es así devolvería true y si no false.

Eliminar usuario

Función para eliminar un usuario:

```php
if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
    $token = '';
    $headers = apache_request_headers();

    if (isset($headers['Authorization'])) {
        $matches = array();
        preg_match('/Bearer\s(\S+)/', $headers['Authorization'],
$matches);
        if (isset($matches[1])) {
            $token = $matches[1];
        }
    } else {
        header("HTTP/1.1 401 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Token no encontrado"
        ]);
    }

    if (isset($_GET['username'])) {
        $username = $_GET['username'];
        $sql = "DELETE FROM users WHERE username='$username'";
        if (!isValidToken($token, $username)) {
            header("HTTP/1.1 401 Bad Request");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "Token no válido"
            ]);
        } else {
            try {
                $con->query($sql);
                header("HTTP/1.1 200 OK");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => true,
                    'msg' => "Usuario eliminado correctamente"
                ]);
            } catch (mysqli_sql_exception $e) {
                header("HTTP/1.1 400 Bad Request");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => false,
```

```
                    'msg' => "Usuario no eliminado"
                ]);
            }
        }
    } else {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Usuario no encontrado"
        ]);
    }
    exit;
}
```

Se comprueba si en la cabecera existe el parámetro “Authorization”, si existe se extrae el valor del token y se almacena en una variable. Si no se recibe una respuesta “HTTP/1.1 401 Bad Request” que devuelve un JSON con el mensaje de token no encontrado.

A continuación, se comprueba que se haya obtenido el parámetro “username” en la url, si es así se almacena en una variable, se declara la consulta y se comprueba que el usuario es quien dice ser con la función “isValidToken” antes mencionada (esto es para evitar que otro usuario borre la cuenta). Si el token y el usuario están relacionados se ejecutaría el bloque try catch, si no se obtendría una respuesta “HTTP/1.1 401 Bas Request” y devolvería un JSON con un mensaje de token no valido. En el bloque try catch se ejecutaría la sentencia y si todo sale bien se recibe una respuesta “HTTP/1.1 200 OK” que devolvería un JSON con el mensaje de usuario eliminado correctamente, si no se recibiría la respuesta “HTTP/1.1 400 Bad Request” y se devolverá un JSON con el mensaje de usuario no eliminado.

Ejemplo de llamada:

```
fetch(`http://localhost/proyecto-
allroutes/php/api/users?username=${username}`, {
        method: 'DELETE',
        headers: {
            'Authorization': `Bearer ${token}`
        }
    })
    .then(response => {
        return response.json();
    })
    .then(data => {
        if(data['success']) {
            console.log(data['msg']);
            localStorage.clear();
            window.location = "index.html"
```

```
        } else{
            console.log(data['msg']);
        }
    })
```

Editar usuario

Función para editar a un usuario:

```
if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
    $token = '';
    $headers = apache_request_headers();

    if (isset($headers['Authorization'])) {
        $matches = array();
        preg_match('/Bearer\s(\S+)/', $headers['Authorization'],
$matches);
        if (isset($matches[1])) {
            $token = $matches[1];
        }
    } else {
        header("HTTP/1.1 401 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Token no encontrado"
        ]);
    }

    if (isset($_GET['username'])) {

        $json = file_get_contents('php://input');
        $user = json_decode($json);

        $username = $_GET['username'];
        $email = $user->email;
        $pass = $user->pass;
        $height = $user->height;
        $weight = $user->weight;
        $birthday = $user->birthday;
        $activities = implode(",", $user->actividades);

        $sql = "UPDATE users SET email='$email', pass='$pass',
height='$height', weight='$weight', birthday='$birthday',
activities='$activities' WHERE username='$username'";

        if (!isValidToken($token, $username)) {
            header("HTTP/1.1 401 Bad Request");
```

```
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "Token no válido"
            ]);
        } else {
            try {
                $con->query($sql);
                header("HTTP/1.1 201 Created");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => true,
                    'msg' => "Usuario modificado correctamente"
                ]);
            } catch (mysqli_sql_exception $e) {
                header("HTTP/1.1 400 Bad Request");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => false,
                    'msg' => "No se puede actualizar el usuario"
                ]);
            }
        }
    } else {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Alguno de los campos requeridos está vacío",
        ]);
    }
}
```

Se comprueba si en la cabecera existe el parámetro “Authorization”, si existe se extrae el valor del token y se almacena en una variable. Si no se recibe una respuesta “HTTP/1.1 401 Bad Request” que devuelve un JSON con el mensaje de token no encontrado.

A continuación, se comprueba que se haya obtenido el parámetro “username” en la url, si es así se obtiene el cuerpo del mensaje descodificado, se almacena la del usuario en distintas variables y se declara la consulta.

Por último, se verifica que el token sea del usuario en cuestión, si el token no es de ese usuario devuelve la respuesta “HTTP/1.1 401 Bad Request” y devuelve un JSON con el mensaje de token no valido. Si por el contrario el token es de ese usuario se ejecuta un bloque try catch para realizar la consulta, si dicha consulta se resuelve correctamente devuelve la respuesta “HTTP/1.1 201 Created” con un JSON con el mensaje de usuario modificado correctamente, si no se ha ejecutado correctamente se recibe la respuesta “HTTP/1.1 400 Bad Request” con un JSON de que no se ha podido actualizar los datos.

Ejemplo de una llamada:

```
let datosUsuario = {
        "email": `${email.value}`,
        "pass": `${pass.value}`,
        "height": `${estatura.value}`,
        "weight": `${peso.value}`,
        "birthday": `${fechaNac.value}`,
        actividades,
    };

    fetch(`http://localhost/proyecto-
allroutes/php/api/users?username=${username}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify(datosUsuario)
    })
    .then(response => {
        return response.json();
    })
    .then(data => {
        if (data['success']) {
            console.log(data['msg']);
        } else {
            console.log(data['msg']);
        }
    })
```

Parte de las rutas

Esta parte de la API gestiona las peticiones para obtener las rutas o para subir una ruta a la base de datos.

URL base para obtener:

- http://localhost/proyecto-allroutes/php/api/routes

URL base para la subida:

- http://localhost/proyecto-allroutes/php/api/loadRoute

Obtener rutas

Función para obtener las rutas:

```php
if ($_SERVER['REQUEST_METHOD'] == 'GET') {

    if (isset($_GET['id'])) {
        $id = $_GET['id'];
        $sql = "SELECT * FROM routes WHERE 1 AND id_user='$id'";
    } else {
        $sql = "SELECT * FROM routes WHERE 1";
    }

    if (isset($_GET['name'])) {
        $name = $_GET['name'];
        $sql .= " AND route_name LIKE '%$name%'";
    }
    if (isset($_GET['min_dist'])) {
        $min_dist = $_GET['min_dist'];
        $sql .= " AND distance >= $min_dist";
    }
    if (isset($_GET['max_dist'])) {
        $max_dist = $_GET['max_dist'];
        $sql .= " AND distance <= $max_dist";
    }

    try {
        $rutas = $con->query($sql)->fetch_all(MYSQLI_ASSOC);
        if ($rutas != null) {
            header("HTTP/1.1 200 OK");
            header("Content-Type: application/json");
            echo json_encode($rutas);
        } else {
            header("HTTP/1.1 404 Not Found");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "No se encuentra la ruta"
            ]);
        }
    } catch (mysqli_sql_exception $e) {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Error al obtener las rutas",
            'error' => $sql,
        ]);
    }
```

Esta función es tanto para la página de búsqueda de rutas como para la pagina de mis rutas. Primero se comprueba si se obtenido el parámetro “id”, si es así la consulta buscará

las rutas de un usuario especifico, si no se buscan todas las rutas independientemente del usuario. Luego se comprueba si se han recibido los parámetros “name”, “min_dist” y “max_dist” que son las opciones de filtrado, si se han recibido se añade a la consulta el filtrado. Por último, en un bloque try catch se ejecuta la consulta en la que si el array “rutas” es distinto a null se ejecuta la consulta satisfactoriamente y devuelve la repuesta “HTTP/1.1 200 OK” con un JSON con el array “rutas”. Si no se encuentran rutas devuelve una respuesta “HTTP/1.1 404 Not Found” con un JSON con el mensaje de que no se encuentran rutas.

Ejemplo de una llamada:

```
let url = `http://localhost/proyecto-
allroutes/php/api/routes?name=${inputNombreRuta}`;

    inputDistMin != '' ? url += `&min_dist=${inputDistMin}` : '';
    inputDistMax != '' ? url += `&min_dist=${inputDistMax}` : '';

    fetch(url, {
        method: 'GET',
        headers: {
            'Content-Type': 'application/json;charset=utf-8'
        },
    })
        .then(response => {
            return response.json();
        })
        .then(data => {
            if (data) {
                let datos = "";
                if (data.length > 0) {
                    data.forEach(item => {
                        datos += `
                        <a class='enlaces' href='detalle-
ruta.php?id=${item['id']}'>
                            <div class='trail'>
                                <div class='trail__header'>
                                    <span>${item['route_name']}</span>
                                </div>
                                <div class='trail__body'>
                                    <div class='trail_item'>
                                        <small>Distancia</small>
                                        <h4>${item['distance']}</h4>
                                    </div>
                                    <div class='trail_item'>
                                        <small>Altura Máxima</small>
                                        <h4>${item['max_height']}m</h4>
                                    </div>
                                    <div class='trail_item'>
                                        <small>Altura Mínima</small>
```

```
                                        <h4>${item['min_height']}m</h4>
                                    </div>
                                </div>
                                <div class='trail__image'><img
src='imgs/default-image.jpg' alt='imagen'></div>
                            </div>
                        </a>
                `;
                    });

                } else {
                    datos = "<h2>No hay ninguna ruta</h2>";
                }
                main_list.innerHTML = datos;
            } else {
                console.log(data['msg']);
            }
```

Subir ruta

Función para subir una ruta:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    $token = '';
    $headers = apache_request_headers();

    if (isset($headers['Authorization'])) {
        $matches = array();
        preg_match('/Bearer\s(\S+)/', $headers['Authorization'],
$matches);
        if (isset($matches[1])) {
            $token = $matches[1];
        }
    } else {
        header("HTTP/1.1 401 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Token no encontrado"
        ]);
    }

    if (isset($_GET['username'])) {
        $json = file_get_contents('php://input');
        $ruta = json_decode($json);

        $username = $_GET['username'];
```

```
        $valorCircu = '';

        if ($ruta->circular == "circular") {
            $valorCircu = 1;
        } else {
            $valorCircu = 0;
        }

        $id_user = $ruta->id_user;
        $route_name = $ruta->route_name;
        $descrip = $ruta->descrip;
        $distance = $ruta->distance;
        $max_height = $ruta->max_height;
        $min_height = $ruta->min_height;
        $pos_slope = $ruta->pos_slope;
        $neg_slope = $ruta->neg_slope;
        $start_lat = $ruta->start_lat;
        $start_lon = $ruta->start_lon;
        $points = $ruta->points;
        $circular = $valorCircu;
        $date = $ruta->date;

        $sql = "INSERT INTO routes (`id_user`, `route_name`, `descrip`,
`distance`, `max_height`, `min_height`, `pos_slope`, `neg_slope`,
`start_lat`, `start_lon`, `points`, `circular`, `date`) VALUES
('$id_user', '$route_name', '$descrip', '$distance', '$max_height',
'$min_height', '$pos_slope', '$neg_slope', '$start_lat', '$start_lon',
'$points', '$circular', '$date')";

        if (!isValidToken($token, $username)) {
            header("HTTP/1.1 401 Bad Request");
            header("Content-Type: application/json");
            echo json_encode([
                'success' => false,
                'msg' => "Token no válido"
            ]);
        } else {
            try {
                $con->query($sql);
                header("HTTP/1.1 201 Created");
                header("Content-Type: application/json");
                echo json_encode([
                    'success' => true,
                    'msg' => "Ruta insertada correctamente"
                ]);
            } catch (mysqli_sql_exception $e) {
                header("HTTP/1.1 400 Bad Request");
                header("Content-Type: application/json");
```

```
                echo json_encode([
                    'success' => false,
                    'msg' => "Ruta no insertada"
                ]);
            }
        }
    } else {
        header("HTTP/1.1 400 Bad Request");
        header("Content-Type: application/json");
        echo json_encode([
            'success' => false,
            'msg' => "Username no encontrado"
        ]);
    }
    exit;
}
```

Primero comprueba que exista el token en la cabecera, si existe se almacena en una variable llamada “token” y si no existe se recibe la respuesta “HTTP/1.1 401 Bad Request” con un JSON con el mensaje token no encontrado.

A continuación, se comprueba si se ha recibido como parámetro el “username”, si no se ha recibido se recibe una respuesta “HTTP/1.1 400 Bad Request” con el mensaje username no encontrado. Por el contrario si se encuentra el “username” se almacena en una variable, se obtiene el cuerpo del mensaje y se almacenan los datos en variables, se declara la consulta y por último se comprueba si el token recibido es del usuario, si no lo es se recibe una respuesta “HTTP/1.1 401 Bad request” y se devuelve un JSON son el mensaje token no valido. Por último, si el token es valido se ejecuta un bloque try catch ejecutando la consulta, si la consulta es exitosa se recibe la respuesta “HTTP/1.1 201 Created” con el mensaje de ruta insertada correctamente, si por el contraria la consulta no se realiza, se recibe la respuesta “HTTP/1.1 Bad Request” con el mensaje ruta no insertada.

Ejemplo de una llamada:

```
let datos = {
                    id_user,
                    route_name,
                    descrip,
                    distance,
                    max_height,
                    min_height,
                    pos_slope,
                    neg_slope,
                    start_lat,
                    start_lon,
                    circular,
                    date,
                    "points": JSON.stringify(puntos),
```

```
                }

                let url = `http://localhost/proyecto-
allroutes/php/api/loadRoute/?username=${username}`;

                fetch(url, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': `Bearer ${token}`
                    },
                    body: JSON.stringify(datos)
                })
                    .then(response => {
                        return response.json();
                    })
                    .then(data => {
                        console.log(data);
                    })
```