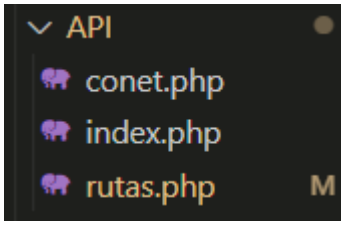
Documentación de la API.

1. Estructura

Mi API REST se compone de 3 ficheros



-conect.php

-index.php

-rutas.php

2. Métodos de la API

2.1 conect.php

Este programa php no permite la conexión con la base de datos indicándole las credenciales del usuario con permisos sobre esta.

Y un constructor que hereda de MySQL y nos permite obtener un objeto conect a disposición.

Las credenciales de mi base de datos son:

- Host: localhost.
- Base de datos: integrador.
- Usuario: integrador.
- Contraseña: integrador.

2.2 index.php

La url genérica es:
http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/

Este fichero se encarga de la funcionalidad de los usuarios de la aplicación con la gestión del webToken.

Lo primero es una importación del auto load de vendor para que me funcionen la librería de “Firebase\JWT\” para su instalación seguí los pasos de esta página de GitHub.

https://github.com/firebase/php-jwt

Y importamos la conexión y creemos el objeto Conect

2.2.3. Métodos Get

GET API/

Si no le pasas ningún parámetro te devuelve un JSON con todos los usuarios.

GET API/?id=X

Parámetros:

-id: id del usuario que buscamos

Devuelve:

- Un json con todos los campos del usuario con mismo id que en el get.

Erros:

-406: si no encuentra un id.

-404: para errores de SQL.

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/?id=5

{

"0”
{"id":"5","nombre":"paco","apellido":"paco","nomUsu":"paco","password":"d49093
17a40933b98fce7b85858a35c0b69a4d621af72c6b0c8ff2c2e79eb56b96fc93373936ff01f25d
f1118502dc367f2ec1532c0716514546ec9a2ebd14ab","correo":"alejandro@gmail.com","
altura":"5","peso":"4","fechNac":"2023-02-
23","listaActividades":"ALGO","tlf":"987654321"},

}

GET API/?n nombreUsu=XX&pass=YY

Parámetros:

- nombreUsu: nombre de usuario de la base de datos.
- pass: contraseña de este usuario.

Devuelve:

- un json con todos los campos del usuario con el que coincidan los parámetros introducidos.

Errores:

- 406: si no se encuentra usuarios con eso datos.
- 404: errores de SQL.

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/?nombreUsu=paco&pass=paco

{"0":
{"id":"5","nombre":"paco","apellido":"paco","nomUsu":"paco","password":"d4

909317a40933b98fce7b85858a35c0b69a4d621af72c6b0c8ff2c2e79eb56b96fc93373936ff01f25df1118502dc367f2ec1532c0716514546ec9a2ebd14ab","correo":"alejandro@gmail.com","altura":"5","peso":"4","fechNac":"2023-02-23","listaActividades":"ALGO","tlf":"987654321"},"webToken":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJub21Vc3UiOiJwYWNvIn0.iGljlOM15ZT4Mg7qHzcIn9mJk2FQxepm3RXp6YrfpYc"}

POST API/

Esta función se encarga de guardar los usuarios en la base de datos y codificar la contraseña.

Parámetros:

-no contiene

Body:

Incorpora un json que reúne la información del formulario de registro de la aplicación.

Respuesta:

-ninguna

Errores:

- 201: Creado el usuario
- 409: El usuario que vas a registrar ya existe porque el nombre es único
- 400: error en la base de datos o falta de la entrada

Ejemplo:

PUT API/

Esta función recoge los datos del usuario ha modificado y los actualiza los cambia en la base de datos.

Esta función también verifica el web token del usuario sí coinciden al decodificar el contenido del payload, Con el nombre del usuario en la base datos.

Parámetros:

-no contiene

Body:

Incorpora un json que reúne la información del formulario de registro de la aplicación, y el web token del usuario que obtuvo al registrarse. Que lo codifico en java script.

Respuesta:

-ninguna

Errores:

- 200: ok, operación exitosa
- 400: error de MySQL
- 401: si el web token no coincide No autorizado.
- 404: cundo el body está vacío;

DELETE API/

Esta función recoge el id del usuario a eliminar y también comprueba su web token.

Parámetros:

-no contiene

Body:

Incorpora un json que contiene el

- id del usuario
- nombre del usuario
- su web token

Este json lo codifico en el fech de java script

Respuesta:

-ninguna

Errores:

- 200: ok, operación exitosa
- 400: error de MySQL.
- 401: si el web token no coincide No autorizado.
- 404: cundo el body está vacío;

2.3 rutas.php

GET API/rutas.php/

Este método devuelve todas las rutas de la base de datos como un json

Parámetros:

-ninguno

Errores:

- 404: error de MySQL;

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/rutas.php

GET API/rutas.php/?id=X

Es importante mencionar que el campo información que es un Sting con estructura json este sirve para dibujar las rutas por que contiene la información de todos puntos del archivo gpx.

Parámetros:

- id: recoge el parámetro id de la ruta.

Devuelve:

- un json con todos los campos de la ruta que coincide con el id del paramento.

Errores:

- 404: error SQL.

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/php/detallesRuta.php?id=16

GET API/rutas.php/?creador=X

Este método busca todas las rutas creadas por un usuario y las devuelve como un json.

Parámetros:

- creador: recoge el parámetro creador y busca las rutas con creadores coincidentes.

Devuelve:

- un json con todos los campos de la ruta que coincide con el creador del paramento.

Errores:

- 404: error SQL.

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/rutas.php?creador=paco

POST API/rutas.php/

Este método permite insertar una ruta en la base de datos el json se codifica en js y se obtiene información del fichero gpx.

Parámetros:

No recibe parámetros

Devuelve:

- El id de la ruta creada en un json

Errores:

- 404: error SQL.

Ejemplo:

http://localhost/proyectoIntegrador/ProyectoSegundoTri/codigo/API/rutas.php?creador=paco

Los métodos Put y Delete están incluidos, pero no implementados.