SEGUNDO PARCIAL – PROGRAMACION III – 2 cuat. 2021

Aclaración:
Las partes se corregirán de manera secuencial (ascendentemente). Si están bien todos los puntos de una parte, habilita la corrección de la parte posterior.
Se debe crear un archivo por cada entidad de PHP.
Todos los métodos deben estar declarados dentro de clases.
PDO es requerido para interactuar con la base de datos.
Se deben respetar los nombres de los archivos, de las clases, de los métodos y de los parámetros de las peticiones.
Utilizar Slim framework 4 para la creación de la Api Rest. Respetar la estructura de directorios (index.php → ./public; fotos → ./src/fotos; clases en ./src/poo;).
NO agregar lógica dentro de los callbacks de la API, referenciar métodos de las clases correspondientes.
Habilitar .htaccess dónde corresponda.

Parte 01 (hasta un 5)

Crear un ***APi Rest*** para la concesionaria de autos **Scaloneta**, que interactúe con la clase ***Auto***, la clase ***Usuario*** y la base de datos ***concesionaria_bd*** (autos - usuarios).

Crear los siguientes verbos:

A nivel de ruta (/usuarios):
(POST) Alta de usuarios. Se agregará un nuevo registro en la tabla usuarios *.
Se envía un JSON → **usuario** (correo, clave, nombre, apellido, perfil**) y **foto**.
La foto se guardará en ./src/fotos, con el siguiente formato: ***correo_id.extension***.
Ejemplo: ./src/fotos/juan@perez_152.jpg
* ID auto-incremental. ** propietario, encargado y empleado.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

A nivel de aplicación:
(GET) Listado de usuarios. Obtendrá el listado completo de los usuarios (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; dato: arrayJSON; status: 200/424)

A nivel de aplicación:
(POST) Alta de autos. Se agregará un nuevo registro en la tabla autos *.
Se envía un JSON → **auto** (color, marca, precio y modelo).
* ID auto-incremental.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

A nivel de ruta (/autos):
(GET) Listado de autos. Obtendrá el listado completo de los autos (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; dato: arrayJSON; status: 200/424)

A nivel de ruta (/login):
(POST) Se envía un JSON → **user** (correo y clave) y retorna un JSON (éxito: true/false; usuario: JSON (con todos los datos del usuario, a excepción de la clave) / null; status: 200/403)

NOTA:
Todos los verbos invocarán métodos de la clase Auto o Usuario para realizar las acciones.

Parte 02 (hasta un 6)

Crear los siguientes Middlewares (en la clase **MW**) para que:

1.- (**método de clase**) Si alguno de los campos correo o clave están vacíos (o los dos) retorne un JSON con el mensaje de error correspondiente (y status 409).
Caso contrario, pasar al siguiente Middleware que:

2.- (**método de instancia**) Verifique que el correo y clave existan en la base de datos. Si **NO** existen, retornar un JSON con el mensaje de error correspondiente (y status 403).
Caso contrario, acceder al verbo de la API.

3.- (**método de clase**) Verifique que el correo **NO** exista en la base de datos. Si **EXISTE**, retornar un JSON con el mensaje de error correspondiente (y status 403).
Caso contrario, acceder al verbo de la API.

Los Middlewares 1 y 2 aplicarlos al verbo POST de /login.
Los Middlewares 1 y 3 aplicarlos al verbo POST de /usuarios.

Crear, a **nivel de grupo (/cars)**, los verbos:

(DELETE) Borrado de autos por ID.
Recibe el ID del auto a ser borrado (**id_auto**, cómo parámetro de ruta).
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

(PUT) Modificar los autos por ID.
Recibe el JSON del auto a ser modificado → **auto** (id_auto, color, marca, precio y modelo) cómo parámetro a nivel de ruta.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Parte 03 (hasta un 8)

Crear, a **nivel de grupo (/users)**, los verbos:

A nivel de ruta (/delete):
(POST) Borrado de usuarios por ID.
Recibe el ID del usuario a ser borrado en un JSON → **usuario** (id_usuario) pasado cómo parámetro de la petición (NO cómo parámetro de ruta)
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

A nivel de ruta (/edit):
(POST) Modificación de usuarios.
Se envía un JSON → **usuario** (id_usuario, correo, clave, nombre, apellido, perfil) y **foto**.
La foto se guardará en ./src/fotos, con el siguiente formato:
correo_id_modificacion.extension.
Ejemplo: ./src/fotos/juan@perez_152_modificacion.jpg
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Parte 04

Crear, a **nivel de grupo (/tablas)**:

A nivel de ruta (/usuarios):
Crear los siguientes Middlewares para que a partir del método que retorna el **listado de usuarios** (clase Usuario **¡NO hacer nuevos métodos!**):

1.- (GET) Retorna una tabla html con el contenido completo de los usuarios (excepto la clave). (clase MW - método de clase).
2.- (POST) Solo si es un '***propietario***', se retornará el listado del punto anterior, pero en formato .pdf. (clase MW - método de instancia). Se envía cómo parámetro de petición un JSON → **usuario** (con todos los datos del usuario, a excepción de la clave)

A nivel de ruta (/autos):
Crear los siguientes Middlewares para que a partir del método que retorna el **listado de autos** (clase Auto **¡NO hacer nuevos métodos!**):

1.- (GET) Retorna una tabla html con el contenido completo de los autos. (clase MW - método de instancia).