SEGUNDO PARCIAL – PROGRAMACION III – 1 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;).
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. Mostrará el listado completo de los usuarios (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; dato: stringJSON; 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. Mostrará el listado completo de los autos (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; dato: stringJSON; status: 200/424)

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

(GET) Se envía el JWT → **token** (en el header) y se verifica. En caso exitoso, retorna un JSON con mensaje y status 200. Caso contrario, retorna un JSON con mensaje y status 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 instancia**) Verifique que estén "seteados" el correo y la clave.
Si no existe alguno de los dos (o los dos) retorne un JSON con el mensaje de error correspondiente (y status 403).
Si existen, pasar al siguiente Middleware que verifique que:

2.- (**método de clase**) Si alguno está vacío (o los dos) retorne un JSON con el mensaje de error correspondiente (y status 409).
Caso contrario, pasar al siguiente Middleware.

3.- (**método de instancia**) Verificar 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.

4.- (**método de clase**) Verificar 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.

5.- (**método de instancia**) Verificar que el precio posea un rango de entre 50.000 y 600.000 y que el color **NO** sea 'amarillo'. Si no pasa la validación de alguno de los dos (o los dos) retorne un JSON con el mensaje de error correspondiente (y status 409).
Caso contrario, acceder al verbo de la API.

Los Middlewares 1, 2 y 3 aplicarlos al verbo POST de /login.
Los Middlewares 1, 2 y 4 aplicarlos al verbo POST de /usuarios.
El Middleware 5 aplicarlo al verbo POST (nivel de aplicación)

Parte 03 (hasta un 8)

En la creación del JWT, establecerle un valor de expiración de **30 segundos**.

Crear, a **nivel de aplicación**, los verbos:

(DELETE) Borrado de autos por ID.
Recibe el ID del auto a ser borrado (**id_auto**) más el JWT → **token** (en el header).
Si el perfil es '***propietario***' se borrará de la base de datos. Caso contrario, se mostrará el mensaje correspondiente (indicando que usuario intentó realizar la acción).
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**), el ID (**id_auto**) y el JWT → **token** (en el header).
Si el perfil es '***encargado***' se modificará de la base de datos. Caso contrario, se mostrará el mensaje correspondiente (indicando que usuario intentó realizar la acción).

Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

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

1.- (**método de instancia**) verifique que el **token** sea válido.
Recibe el JWT → **token** (en el header) a ser verificado.
Retorna un JSON con el mensaje de error correspondiente (y status 403), en caso de no ser válido.
Caso contrario, pasar al siguiente callable.

2.- (**método de clase**) verifique si es un '***propietario***' o no.
Recibe el JWT → **token** (en el header) a ser verificado.
Retorna un JSON con propietario: true/false; mensaje: string (mensaje correspondiente); status: 200/409.

3.- (**método de instancia**) verifique si es un '***encargado***' o no.
Recibe el JWT → **token** (en el header) a ser verificado.
Retorna un JSON con encargado: true/false; mensaje: string (mensaje correspondiente); status: 200/409.

Aplicar los Middlewares a los verbos PUT y DELETE.

Parte 04

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

1.- Si el que accede al listado de autos es un '***encargado***', retorne todos los datos, menos el ID. (clase MW - método de instancia).

2.- Si es un '***empleado***', muestre la cantidad de colores (distintos) que se tiene. (clase MW - método de instancia).

3.- Si es un '***propietario***', muestre todos los datos de los autos (si el ID está vacío o indefinido) o el auto (cuyo ID fue pasado como parámetro). (clase MW - método de clase).

En todos los casos pasar el JWT → **token** por el encabezado de la petición.
Aplicar los Middlewares al verbo GET del ***grupo /autos***

B.- Crear los siguientes Middlewares para que a partir del método que retorna el **listado de usuarios** (clase Usuario **¡NO hacer nuevos métodos!**):

1.- Si el que accede al listado de autos es un '***encargado***', retorne todos los datos, menos la clave y el ID. (clase MW - método de instancia).

2.- Si es un '***empleado***', muestre solo el nombre, apellido y foto de los usuarios. (clase MW - método de instancia).

3.- Si es un '***propietario***', muestre la cantidad de usuarios cuyo apellido coincida con el pasado por parámetro o los apellidos (y sus cantidades) si es que el parámetro pasado está vacío o indefinido. (clase MW - método de clase).

En todos los casos pasar el JWT → **token** por el encabezado de la petición.

Aplicar los Middlewares al verbo GET a nivel de aplicación.