RECUPERATORIO 2 PARCIAL – PROGRAMACION III – 1 cuat. 2020

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.
POO: obligatorio.
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 para la creación de la Api Rest.
Habilitar .htaccess.

Parte 01 (hasta un 6)

Crear un ***APi Rest*** para la veterinaria ***Mascotitas***, que interactúe con la clase ***Mascota***, la clase ***Usuario*** y la base de datos ***veterinaria_bd*** (mascotas - 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 (como parámetros) un JSON → **usuario** (correo, clave, nombre, apellido, perfil **) y **foto**.
* ID auto-incremental. ** propietario, encargado y empleado.
La foto se guardará en ./fotos, con el siguiente formato: ***correo_id.extension***.
Ejemplo: ./fotos/juan@perez_152.jpg
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; tabla: stringJSON; status: 200/424).

A nivel de aplicación:
(POST) Alta de mascotas. Se agregará un nuevo registro en la tabla mascotas *.
Se envía (como parámetro) un JSON → **mascota** (nombre, tipo ** y edad).
* ID auto-incremental. ** perro, gato y tortuga.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418).

A nivel de ruta (/mascotas):
(GET) Listado de mascotas. Mostrará el listado completo de las mascotas (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; tabla: stringJSON; status: 200/424).

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

(GET) Se envía el JWT → **token** (en el header) y se verifica. En caso exitoso, retorna un JSON (éxito: true/false; mensaje: string; status: 200/403).

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

(DELETE) Borrado de mascotas por ID.

Se envía (como parámetro) el ID de la mascota a ser borrada → **id_mascota** más el JWT → **token** (en el header).
Si se pudo borrar, se escribirá en ***./log/borrados.log*** la fecha (día, mes, año, hora, minutos y segundos), el nombre y apellido del usuario y el ID de la mascota borrada.
Si no se borró nada de la base de datos, se escribirá en ***./log/no_borrados.log*** la fecha (día, mes, año, hora, minutos y segundos), el nombre, apellido y perfil del usuario más el ID de la mascota.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418).

(PUT) Modifica mascotas por ID.

Recibe un JSON → **mascota** (formado con todos los datos de la mascota más el ID, cómo parámetros) más el JWT → **token** (en el header).
Si se pudo modificar, se escribirá en ***./log/modificados.log*** la fecha (día, mes, año, hora, minutos y segundos), el nombre y apellido del usuario y el nombre, tipo y ID de la mascota modificada.
Si no se modificó nada de la base de datos, se escribirá en ***./log/no_modificadosados.log*** la fecha (día, mes, año, hora, minutos y segundos), el nombre, apellido y perfil del usuario más el ID de la mascota.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

A nivel de ruta (/log):

(GET) Listado de los archivos ***.log***.

Se envía (como parámetro) el tipo de archivo (b, nb, m, nm) → **tipo** más el JWT → **token** (en el header) y mostrará el contenido completo del archivos .log, de acuerdo al perfil:
propietario: no tiene restricciones, muestra el archivo .log según el tipo recibido.
encargado: sólo puede ver los .log de los *no_borrados* y *no_modificados*, según el tipo recibido.
empleado: sólo puede ver los .log de los *borrados* y *modificados*, según el tipo recibido.

En caso de no tener los permisos correspondientes, se debe informar en un mensaje.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/403).

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

Parte 02 (hasta un 7)

A nivel de ruta (/pdf):

(GET) Listados en formato ***.pdf***.

Se envía (como parámetro) el tipo de archivo → **tipo_pdf** (usuarios, mascotas) más el JWT → **token** (en el header) y mostrará el listado correspondiente.
El archivo tendrá:
*-**Encabezado** (apellido y nombre del alumno a la izquierda y número de página a la derecha)
*-**Cuerpo** (Título del listado, listado completo de los usuarios (con su respectiva foto) o de las mascotas, según corresponda)
*-**Pie de página** (fecha actual, centrada).

NOTA: El archivo .pdf contendrá clave, si el perfil es empleado, será el apellido del usuario logueado y si es propietario o encargado será el correo del usuario logueado.

Parte 03 (hasta un 8)

Crear los siguientes Middlewares para que a partir del método que retorna el **listado de mascotas** (clase Mascota **¡NO hacer nuevos métodos!**):

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

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

3.- Si es un '***propietario***', muestre todos los datos de las mascotas, según el tipo que fue pasado como parámetro). (clase MW - método de clase).

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

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 por el encabezado de la petición.

Aplicar los Middlewares al verbo GET a nivel de aplicación.