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 5)

Crear un ***APi Rest*** para la farmacia ***CoronaFarma***, que interactúe con la clase ***Barbijo***, la clase ***Usuario*** y la base de datos ***farmacia_bd*** (barbijos - 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**.
* 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 barbijos. Se agregará un nuevo registro en la tabla barbijos *.
Se envía un JSON → **barbijo** (color, tipo ** y precio).
* ID auto-incremental. ** liso, estampado y transparente.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418).

A nivel de ruta (/barbijos):
(GET) Listado de barbijos. Mostrará el listado completo de los barbijos (array JSON).
Retorna un JSON (éxito: true/false; mensaje: string; tabla: 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) / 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).

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

Parte 02 (hasta un 6)

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

(DELETE) Borrado de barbijos por ID.
Se envía el ID del barbijo a ser borrado → **id_barbijo** más el JWT → **token** (en el header).
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418).

(PUT) Modifica barbijos por ID.
Recibe un JSON → **barbijo** (formado con todos los datos del barbijo más el ID) más el JWT → **token** (en el header).
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Parte 03 (hasta un 7)

A nivel de ruta (/pdf):
(GET) Listados en formato ***.pdf***.
Se envía el tipo de archivo → **tipo_pdf** (usuarios, barbijos) más el JWT → **token** (en el header) y mostrará el listado correspondiente.
El archivo tendrá: *-Encabezado (apellido y nombre del alumno y número de página) *-Cuerpo (Título del listado, listado completo de los usuarios (con su respectiva foto) o de los barbijos, según corresponda) *-Pie de página (fecha actual).
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 04 (hasta un 8)

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.

Los Middlewares 1, 2 y 3 aplicarlos al verbo POST de /login.
Los Middlewares 1, 2 y 4 aplicarlos al verbo POST de /usuarios.