SEGUNDO PARCIAL – PROGRAMACIÓN III – 2 cuat. 2023

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.
El virtual host definirlo cómo → http://prog_3_sp.
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 juguetería **El muñeco**, que interactúe con la clase **Juguete**, la clase **Usuario** y la base de datos **jugueteria_bd** (juguetes - usuarios).

Crear los siguientes verbos:

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 / null; status: 200/424)

A nivel de aplicación:
(POST) Alta de juguetes. Se agregará un nuevo registro en la tabla juguetes.
Se envía un JSON → **juguete_json** (marca y precio) y la **foto**.
La foto se guardará en ./src/fotos, con el siguiente formato: **marca.extension**.
Ejemplo: ./src/fotos/mattel.jpg
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

A nivel de ruta (/juguetes):
(GET) Listado de juguetes. Obtendrá el listado completo de los juguetes (array JSON). Retorna un JSON (éxito: true/false; mensaje: string; dato: arrayJSON / null; 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 (*) / null; status: 200/403).
(*) el payload del JWT tendrá la siguiente estructura:
- usuario: con todos los datos del usuario, a excepción de la clave.
- alumno: colocar su nombre y apellido
- dni_alumno: indicar el número de DNI del alumno

El token creado deberá ser firmado con el apellido.nombre del alumno y debe expirar en 120 segundos.

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

NOTA:
Todos los verbos invocarán métodos de la clase Juguete 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.

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

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

(DELETE) Borrado de juguetes por ID.

Recibe el ID del juguete a ser borrado (**id_juguete**, cómo parámetro de ruta) más el JWT → **token** (en el header).
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

(POST) Modificar juguetes por ID.

Recibe el JSON del juguete a ser modificado → **juguete** (id_juguete, marca, precio), la **foto** y el JWT → **token** (en el header).
El id_juguete será el id del juguete a ser modificado, mientras que el resto, serán los valores a ser modificados.
La foto se guardará en ./src/fotos, con el siguiente formato:
marca_modificacion.extension.
Ejemplo: ./src/fotos/mattel_modificacion.jpg
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Agregar un Middleware (en la clase MW) para que:

3.- (**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.

Aplicar el Middleware al grupo **/toys** y al verbo POST (a nivel de aplicación).

Parte 03 (hasta un 8)

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 mostrando sólo el correo, el nombre y el apellido (clase MW - método de clase).
Se envía en el header JWT → **token**
Agregar el Middleware 3 (verificar que el token sea válido).
Si no es propietario, retornar un JSON (éxito: false; mensaje: string; status: 403)

A nivel de ruta (/juguetes):
Crear el siguiente Middleware para que a partir del método que retorna el **listado de juguetes** (clase Juguete **¡NO hacer nuevos métodos!**):

1.- (GET) Retorna una tabla HTML con el contenido completo de los juguetes, pero sólo de los IDs impares. (clase MW - método de instancia).

Parte 04

Crear el siguiente verbo:
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.extension**.
Ejemplo: ./src/fotos/juan@perez.jpg
* propietario, supervisor y empleado.
Retorna un JSON (éxito: true/false; mensaje: string; status: 200/418)

Crear el siguiente Middleware (en la clase **MW**) para que:

4.- (**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, 4 y 3 aplicarlos al verbo POST de **/usuarios**.