PRIMER PARCIAL – PROGRAMACIÓN III – 2 cuat. 2022

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.
Todas las entidades deben estar dentro de un namespace (Apellido\Nombre del alumno).
Todos los métodos deben estar declarados dentro de clases. Se tendrá encuenta la aplicación de PSR-1, el tipado de atrbutos, parámetros y valores de retorno.
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.
Todas las referencias a archivos y/o recursos, deben estar de manera relativa.

Parte 1 (hasta un 5)

neumatico.php. Crear, en **./clases**, la clase **Neumatico** con atributos protegidos:

- marca(cadena)
- medidas(cadena)
- precio(flotante)

Un constructor (que inicialice los atributos), un método de instancia ***toJSON()***, que retornará los datos de la instancia (en una cadena con formato **JSON**).
Método de instancia ***guardarJSON($path)***, que agregará al neumático en el path recibido por parámetro. Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Método de clase ***traerJSON($path)***, que retornará un array de objetos de tipo neumático (recuperados del path).
Método de clase ***verificarNeumaticoJSON($neumatico)***, que recorrerá el array obtenido del método *traerJSON* y retornará un **JSON** que contendrá: existe(bool) y mensaje(string).

Nota: *Si el neumático está registrado (comparar por marca y medidas), retornará* ***true*** *y el mensaje indicará la sumatoria de precios de aquellos neumáticos registrados con la misma marca y las mismas medidas del neumático recibido por parámetro. Caso contrario, retornará* ***false****, y en el mensaje se informará de lo acontecido.*

En el directorio raíz del proyecto, agregar las siguientes páginas:

altaNeumaticoJSON.php: Se recibe por POST la ***marca***, las ***medidas*** y el ***precio***. Invocar al método *guardarJSON* y pasarle *'**./archivos/neumaticos.json**'* cómo parámetro.

verificarNeumaticoJSON.php: Se recibe por POST la ***marca*** y las ***medidas***.
Retornar un **JSON** que contendrá: éxito(bool) y mensaje(string) (agregar el mensaje obtenido del método *verificarNeumaticoJSON*).

listadoNeumaticosJSON.php: (GET) Se mostrará el listado de todos los neumáticos en formato JSON (*traerJSON*). Pasarle *'**./archivos/neumaticos.json**'* cómo parámetro.

neumaticoBD.php. Crear, en **./clases**, la clase **NeumaticoBD** (hereda de *Neumatico*) con atributos protegidos:

- id (entero)
- pathFoto (cadena)

Un constructor (con parámetros opcionales), un método de instancia ***toJSON()***, que retornará los datos de la instancia (en una cadena con formato JSON).

Crear, en **./clases**, la interface ***IParte1***. Esta interface poseerá los métodos:

- **agregar**: agrega, a partir de la instancia actual, un nuevo registro en la tabla **neumaticos** (id, marca, medidas, precio, foto), de la base de datos **gomeria_bd**. Retorna **true**, si se pudo agregar, **false**, caso contrario.
- **traer**: este método **estático** retorna un array de objetos de tipo *NeumaticoBD*, recuperados de la base de datos.

Implementar la interface en la clase *NeumaticoBD*.

agregarNeumaticoSinFoto.php: Se recibe por POST el parámetro ***neumático_json*** (marca, medidas y precio), en formato de cadena JSON. Se invocará al método **agregar**.
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

listadoNeumaticosBD.php: (GET) Se mostrará el listado **completo** de los neumaticos (obtenidos de la base de datos) en una tabla (HTML con cabecera). Invocar al método **traer**.

***Nota**: Si se recibe el parámetro **tabla** con el valor **mostrar**, retornará los datos en una tabla (HTML con cabecera), preparar la tabla para que muestre la imagen, si es que la tiene.*
Si el parámetro no es pasado o no contiene el valor mostrar, retornará el array de objetos con formato JSON.

Parte 2 (hasta un 6)

Crear, en **./clases**, la interface ***IParte2***. Esta interface poseerá los métodos:

- **eliminar:** este método **estático**, elimina de la base de datos el registro coincidente con el id recibido cómo parámetro. Retorna **true**, si se pudo eliminar, **false**, caso contrario.
- **modificar**: Modifica en la base de datos el registro coincidente con la instancia actual (comparar por id). Retorna **true**, si se pudo modificar, **false**, caso contrario.

Implementar la interface en la clase *NeumaticoBD*.

eliminarNeumaticoBD.php: Recibe el parámetro ***neumatico_json*** (id, marca, medidas y precio, en formato de cadena JSON) por POST y se deberá borrar el neumático (invocando al método **eliminar**).
Si se pudo borrar en la base de datos, invocar al método *guardarJSON* y pasarle cómo parámetro el valor ***'./archivos/neumaticos_eliminados.json'***.
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

modificarNeumaticoBD.php: Se recibirán por POST los siguientes valores: ***neumatico_json*** (id, marca, medidas y precio, en formato de cadena JSON) para modificar un neumático en la base de datos. Invocar al método **modificar**.

***Nota:** El valor del id, será el id del neumático 'original', mientras que el resto de los valores serán los del neumático a ser modificado.*
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 3 (hasta un 8)

Crear, en **./clases**, la interface ***IParte3***. Esta interface poseerá el método:

- **existe:** retorna true, si la instancia actual está en el array de objetos de tipo *NeumaticoBD* que recibe como parámetro (comparar por marca y medidas). Caso contrario retorna false.

verificarNeumaticoBD.php: Se recibe por POST el parámetro ***obj_neumatico***, que será una cadena **JSON** (marca y medidas), si coincide con algún registro de la base de datos (invocar al método **traer**) retornará los datos del objeto (invocar al **toJSON**). Caso contrario, un JSON vacío ({}).

agregarNeumaticoBD.php: Se recibirán por POST los valores: ***marca***, ***medidas***, ***precio*** y la ***foto*** para registrar un neumático en la base de datos.
Verificar la previa existencia del neumático invocando al método **existe**. Se le pasará como parámetro el array que retorna el método **traer**.
Si el neumático ya existe en la base de datos, se retornará un mensaje que indique lo acontecido.
Si el neumático no existe, se invocará al método **agregar**. La imagen se guardará en "./neumaticos/imagenes/", con el nombre formado por el **marca** punto hora, minutos y segundos del alta (***Ejemplo: pirelli.105905.jpg***).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 4 (hasta un 9)

Crear, en **./clases**, la interface ***IParte4***. Esta interface poseerá el método:

- **guardarEnArchivo**: escribirá en un archivo de texto (**./archivos/*neumaticosbd_borrados.txt***) toda la información del neumático más la nueva ubicación de la foto. La foto se moverá al subdirectorio "./neumaticosBorrados/", con el nombre formado por el **id** punto **marca** punto '***borrado***' punto hora, minutos y segundos del borrado (***Ejemplo: 688.bridgestone.borrado.105905.jpg***).
  Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Implementar la interface en la clase *NeumaticoBD*.

eliminarNeumaticoBDFoto.php: Se recibe el parámetro ***neumatico_json*** (*id, marca, medidas, precio y pathFoto* en formato de cadena JSON) por POST. Se deberá borrar el neumático (invocando al método **eliminar**).
Si se pudo borrar en la base de datos, invocar al método **guardarEnArchivo**.
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Si se invoca por GET (sin parámetros), se mostrarán en una tabla (HTML) la información de todos los neumáticos borrados y sus respectivas imagenes.

modificarNeumaticoBDFoto.php: Se recibirán por POST los siguientes valores: ***neumatico_json*** (id, marca, medidas y precio, en formato de cadena JSON) y la ***foto*** (para modificar un neumático en la base de datos).
Invocar al método **modificar**.

Nota: *El valor del id, será el id del neumático 'original', mientras que el resto de los valores serán los del neumático a ser modificado.*

Si se pudo modificar en la base de datos, la foto original del registro modificado se moverá al subdirectorio "./neumaticosModificados/", con el nombre formado por el **id** punto **marca** punto '***modificado***' punto hora, minutos y segundos de la modificación (***Ejemplo: 987.fateo.modificado.105905.jpg***).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 5

mostrarBorradosJSON.php: Muestra todo lo registrado en el archivo "*neumaticos_eliminados.json*". Para ello, agregar un método estático (en *NeumaticoBD*), llamado **mostrarBorradosJSON**.

mostrarFotosDeModificados.php: Muestra (en una tabla HTML) todas las imagenes (50px X 50px) de los neumaticos registrados en el directorio "./neumaticosModificados/". Para ello, agregar un método estático (en *NeumaticoBD*), llamado **mostrarModificados**.