PRIMER 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.
Todas las referencias a archivos y/o recursos, deben estar de manera relativa.

Parte 1 (hasta un 5)

Producto.php. Crear, en **./clases**, la clase **Producto** con atributos públicos (nombre y origen), 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($path)***, que agregará al producto en el path recibido por parámetro.
Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Método de clase ***TraerJSON()***, que retornará un array de objetos de tipo producto.
Método de clase ***VerificarProductoJSON($producto)***, que recorrerá el array obtenido del método TraerJSON y retornará un **JSON** que contendrá: existe(bool) y mensaje(string).
Si el producto está registrado (comparar por nombre y origen), retornará ***true*** y el mensaje indicará cuantos productos están registrados con el mismo origen del producto recibido por parámetro. Caso contrario, retornará ***false***, y el/los nombres de la/las productos más populares (mayor cantidad de apariciones).

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

AltaProductoJSON.php: Se recibe por POST el ***nombre*** y el ***origen***. Invocar al método GuardarJSON y pasarle *'**./archivos/productos.json**'* cómo parámetro.

VerificarProductoJSON.php: Se recibe por POST el ***nombre*** y el ***origen***, si coinciden con algún registro del archivo JSON (VerificarProductoJSON), crear una **COOKIE** nombrada con el nombre y el origen del producto, separado con un guión bajo (***limon_tucuman***) que guardará la fecha actual (con horas, minutos y segundos) más el retorno del mensaje del método estático VerificarProductoJSON de la clase Producto.
Retornar un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido (agregar, aquí también, el mensaje obtenido del método VerificarProductoJSON).

ListadoProductosJSON.php: (GET) Se mostrará el listado de todos los productos en formato JSON (TraerJSON).

MostrarCookie.php: Se recibe por GET el ***nombre*** y el ***origen*** del producto y se verificará si existe una cookie con el mismo nombre, de ser así, retornará un **JSON** que contendrá: éxito(bool) y mensaje(string), dónde se mostrará el contenido de la cookie. Caso contrario, false y el mensaje indicando lo acontecido.

ProductoEnvasado.php. Crear, en **./clases**, la clase **ProductoEnvasado** (hereda de producto) con atributos públicos (id, codigoBarra, precio y pathFoto), constructor (con todos sus 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 **productos** (id, codigo_barra, nombre, origen, precio, foto), de la base de datos **productos_bd**. Retorna **true**, si se pudo agregar, **false**, caso contrario.

- **Traer**: este método estático retorna un array de objetos de tipo ProductoEnvasado, recuperados de la base de datos.

Implementar la interface en la clase ProductoEnvasado.

AgregarProductoSinFoto.php: Se recibe por POST el parámetro ***producto_json*** (codigoBarra, nombre, origen 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.

ListadoProductosEnvasados.php: (GET) Se mostrará el listado **completo** de los productos envasados (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 ProductoEnvasado.

EliminarProductoEnvasado.php: Recibe el parámetro ***producto_json*** (id, nombre y origen, en formato de cadena JSON) por POST y se deberá borrar el producto envasado (invocando al método Eliminar).
Si se pudo borrar en la base de datos, invocar al método GuardarJSON y pasarle
'./archivos/productos_eliminados.json' cómo parámetro.
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

ModificarProductoEnvadado.php: Se recibirán por POST los siguientes valores: ***producto_json*** (id, codigoBarra, nombre, origen y precio, en formato de cadena JSON) para modificar un producto envasado en la base de datos. Invocar al método Modificar.
Nota: *El valor del id, será el id del producto envasado 'original', mientras que el resto de los valores serán los del producto envasado a ser modificado.*
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 3 (hasta un 9)

Crear, en **./clases**, la interface ***IParte3***. Esta interface poseerá los métodos:

- **Existe:** retorna true, si la instancia actual está en el array de objetos de tipo ProductoEnvasado que recibe como parámetro (comparar por nombre y origen). Caso contrario retorna false.
- **GuardarEnArchivo**: escribirá en un archivo de texto (**./archivos/***productos_envasados_borrados.txt*) toda la información del producto envasado más la nueva ubicación de la foto. La foto se moverá al subdirectorio "./productosBorrados/", con el nombre formado por el **id** punto **nombre** punto '***borrado***' punto hora, minutos y segundos del borrado (***Ejemplo: 688.tomate.borrado.105905.jpg***).

Implementar la interface en la clase ProductoEnvasado.

VerificarProductoEnvasado.php: Se recibe por POST el parámetro ***obj_producto***, que será una cadena **JSON** (nombre y origen), 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 ({}).

AgregarProductoEnvasado.php: Se recibirán por POST los valores: ***codigoBarra, nombre***, ***origen***, ***precio*** y la ***foto*** para registrar un producto envasado en la base de datos.
Verificar la previa existencia del producto envasado invocando al método Existe. Se le pasará como parámetro el array que retorna el método Traer.
Si el producto envasado ya existe en la base de datos, se retornará un mensaje que indique lo acontecido.
Si el producto envasado no existe, se invocará al método Agregar. La imagen se guardará en "./productos/imagenes/", con el nombre formado por el **nombre** punto **origen** punto hora, minutos y segundos del alta (***Ejemplo: tomate.argentina.105905.jpg***).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

BorrarProductoEnvasado.php: Se recibe el parámetro ***producto_json*** (*id, codigoBarra, nombre, origen, precio y pathFoto* en formato de cadena JSON), se deberá borrar el producto envasado (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 productos envasados borrados y sus respectivas imagenes.

ModificarProductoEnvadadoFoto.php: Se recibirán por POST los siguientes valores: ***producto_json*** (id, codigoBarra, nombre, origen y precio, en formato de cadena JSON) y la ***foto*** (para modificar un producto envasado en la base de datos. Invocar al método Modificar.
Nota: *El valor del id, será el id del producto envasado 'original', mientras que el resto de los valores serán los del producto envasado a ser modificado.*
Si se pudo modificar en la base de datos, la foto original del registro modificado se moverá al subdirectorio "./productosModificados/", con el nombre formado por el **nombre** punto **origen** punto '***modificado***' punto hora, minutos y segundos de la modificación (***Ejemplo: aceite.italia.modificado.105905.jpg***).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 4

MostrarBorradosJSON.php: Muestra todo lo registrado en el archivo "*productos_eliminados.json*". Para ello, agregar un método estático (en ProductoEnvasado), llamado **MostrarBorradosJSON**.

MostrarFotosDeModificados.php: Muestra (en una tabla HTML) todas las imagenes (50px X 50px) de los productos envasados registrados en el directorio "./productosModificados/". Para ello, agregar un método estático (en ProductoEnvasado), llamado **MostrarModificados**.

FiltrarProductos.php: Se recibe por POST el ***origen***, se mostrarán en una tabla (HTML) los productos envasados cuyo origen coincidan con el pasado por parámetro.
Si se recibe por POST el ***nombre***, se mostrarán en una tabla (HTML) los productos envasados cuyo nombre coincida con el pasado por parámetro.
Si se recibe por POST el ***nombre*** y el ***origen***, se mostrarán en una tabla (HTML) los productos envasados cuyo nombre y origen coincidan con los pasados por parámetro.