PRIMER PARCIAL – PROGRAMACION III – 2 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.
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.

Parte 1 (hasta un 5)

Ciudadano.php. Crear, en **./clases**, la clase **Ciudadano** con atributos privados (ciudad, email y clave), constructor (que inicialice los atributos), un método de instancia ToJSON(), que retornará los datos de la instancia (en una cadena con formato **JSON**).
Agregar:
Método de instancia GuardarEnArchivo(), que agregará al ciudadano en **./archivos**/ciudadano.json. Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Método de clase TraerTodos(), que retornará un array de objetos de tipo Ciudadano.
Método de clase VerificarExistencia($ciudadano), que recorrerá el array (invocar a TraerTodos) y retornará un **JSON** que contendrá: existe(bool) y mensaje(string).
Si el ciudadano está registrado (email y clave), retornará ***true*** y el mensaje indicará cuantos cuidadanos están registrados con la misma ciudad del ciudadano recibido por parámetro. Caso contrario, retornará ***false***, y el/los nombres de la/las ciudades más populares (mayor cantidad de apariciones).

AltaCiudadano.php: Se recibe por POST la ***ciudad***, el ***email*** y la ***clave***. Invocar al método GuardarEnArchivo.

VerificarCiudadano.php: Se recibe por POST el ***email*** y la ***clave***, si coinciden con algún registro del archivo JSON (VerificarExistencia), crear una COOKIE nombrada con el email y la ciudad del ciudadano, separado con un guión bajo (pep@gmail.com_manchester) que guardará la fecha actual (con horas, minutos y segundos) más el retorno del mensaje del método VerificarExistencia.
Retornar un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido (agregar el mensaje obtenido del método de clase).

ListadoCiudadanos.php: (GET) Se mostrará el listado de todos los ciudadanos en formato JSON.

MostrarCookie.php: Se recibe por GET la ***ciudad*** y el ***email*** del ciudadano 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.
Nota: *Reemplazar los puntos por guiones bajos en el email (en caso de ser necesario).*

Ciudad.php. Crear, en **./clases**, la clase **Ciudad** con atributos públicos (id, nombre, poblacion, pais 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 **ciudades** (id, nombre, poblacion, pais, foto), de la base de datos **ciudades_bd**. Retorna **true**, si se pudo agregar, **false**, caso contrario.
- **Traer**: retorna un array de objetos de tipo Ciudad, recuperados de la base de datos.

Implementar la interface en la clase Ciudad.

AgregarCiudadSinFoto.php: Se recibe por POST el ***nombre***, la ***poblacion*** y el ***pais***. Se invocará al método Agregar. Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

ListadoCiudades.php: (GET) Se mostrará el listado **completo** de las ciudades (obtenidas de la base de datos) en una tabla (HTML con cabecera). Invocar al método Traer.
Nota*: preparar la tabla (HTML) con una columna extra para que muestre la imagen de la foto (si es que la tiene).*

Parte 2 (hasta un 7)

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

- **Existe**: retorna **true**, si la instancia actual está en el array de objetos de tipo Ciudad que recibe como parámetro (comparar por nombre y pais). Caso contrario retorna **false**.
- **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 Ciudad.

VerificarCiudad.php: Se recibe por POST el parámetro ***ciudad***, que será una cadena **JSON** (nombre y pais), si coincide con algún registro de la base de datos (invocar al método Traer) retornar los datos del objeto (invocar al ToJSON). Caso contrario informar: si no coincide el nombre o el pais o ambos.

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

ModificarCiudad.php: Se recibirán por POST los siguientes valores: ***ciudad_json*** (id, nombre, poblacion, pais y pathFoto, en formato de cadena JSON) y ***foto (*** para modificar un ovni en la base de datos. Invocar al método Modificar.

Nota: El valor del id, será el id de la ciudad 'original', mientras que el resto de los valores serán los de la ciudad modificada.

Si se pudo modificar en la base de datos, la foto modificada se moverá al subdirectorio "./ciudadesModificadas/", con el nombre formado por el **nombre** punto **pais** punto '***modificado***' punto hora, minutos y segundos de la modificación (***Ejemplo: liverpool.inglaterra.modificado.105905.jpg***).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Parte 3

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

- **Eliminar**: elimina de la base de datos el registro coincidente con la instancia actual (comparar por nombre y pais). Retorna **true**, si se pudo eliminar, **false**, caso contrario.
- **GuardarEnArchivo**: escribirá en un archivo de texto (*ciudades_borradas.txt*) toda la información de la ciudad más la nueva ubicación de la foto. La foto se moverá al subdirectorio "./ciudadesBorradas/", con el nombre formado por el **id** punto **nombre** punto '***borrado***' punto hora, minutos y segundos del borrado (***Ejemplo: 688.madrid.borrado.105905.jpg***).

Implementar la interface en la clase Ciudad.

EliminarCiudad.php: Si recibe un ***nombre*** por GET, retorna si la ciudad está en la base o no (mostrar mensaje).
Si recibe por GET (sin parámetros), se mostrarán en una tabla (HTML) la información de todas las ciudades borradas y sus respectivas imagenes.
Si recibe el parámetro ***ciudad_json*** (id, nombre, y pais, en formato de cadena JSON) por POST, más el parámetro ***accion*** con valor "**borrar**", se deberá borrar la ciudad (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.

Parte 4

FiltrarCiudad.php: Se recibe por POST el ***nombre***, se mostrarán en una tabla (HTML) las ciudades cuyo nombre coincidan con el pasado por parámetro.
Si se recibe por POST el ***pais***, se mostrarán en una tabla (HTML) las ciudades cuyo pais coincida con el pasado por parámetro.
Si se recibe por POST el ***nombre*** y el ***pais***, se mostrarán en una tabla (HTML) las ciudades cuyo nombre y pais coincidan con los pasados por parámetro.

MostrarBorrados.php: Muestra todo lo registrado en el archivo de texto "ciudades_borradas.txt". Para ello, agregar un método estático (en Ciudad), llamado **MostrarBorrados**.