Modelo - PRIMER PARCIAL – PROG. III - LAB. III 2021

Aclaración:
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 BACKEND

Crear una aplicación Web que permita realizar un ABM de distintos usuarios y empleados.

Usuario.php. Crear, en **./backend/clases**, la clase **Usuario** con atributos públicos (id, nombre, correo, clave, id_perfil y perfil) y un método de instancia ToJSON(), que retornará los datos de la instancia nombre, correo y clave (en una cadena con formato **JSON**).

Agregar los siguientes métodos:

Método de instancia ***GuardarEnArchivo()***, que agregará al usuario en **./backend/archivos**/usuarios.json. Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Método de clase ***TraerTodosJSON()***, que retornará un array de objetos de tipo Usuario, recuperado del archivo usuarios.json.

Método de instancia ***Agregar()***: agrega, a partir de la instancia actual, un nuevo registro en la tabla **usuarios** (id,nombre, correo, clave e id_perfil), de la base de datos **usuarios_test**. Retorna **true**, si se pudo agregar, **false**, caso contrario.

Método de clase ***TraerTodos()***: retorna un array de objetos de tipo Usuario, recuperados de la base de datos (con la descripción del perfil correspondiente).

Método de clase ***TraerUno($params)***: retorna un objeto de tipo Usuario, de acuerdo al correo y clave que ser reciben en el parámetro $params.

Crear las siguientes páginas, en el directorio backend del proyecto:

AltaUsuarioJSON.php: Se recibe por POST el ***correo***, la ***clave*** y el ***nombre***. Invocar al método GuardarEnArchivo.

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

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

ListadoUsuarios.php: (GET) Se mostrará el listado completo de los usuarios, exepto la clave (obtenidos de la base de datos). Invocar al método TraerTodos.
Si se recibe el parámetro ***tabla*** con el valor ***mostrar***, retornará los datos en una tabla (HTML con cabecera). Si el parámetro no es pasado, retornará el array de objetos con formato JSON.

VerificarUsuario.php: (POST) Se recibe el parámetro ***usuario_json*** (correo y clave, en formato de cadena JSON) y se invoca al método TraerUno.
Se retornará un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

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

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.
Eliminar (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.

Crear las siguientes páginas, en el directorio backend del proyecto:

ModificarUsuario.php: Se recibirán por POST los siguientes valores: ***usuario_json*** (id, nombre, correo, clave y id_perfil, en formato de cadena JSON), para modificar un usuario en la base de datos. Invocar al método Modificar.
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

EliminarUsuario.php: Si recibe el parámetro ***id*** por POST, más el parámetro ***accion*** con valor "**borrar**", se deberá borrar el usuario (invocando al método Eliminar).
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

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

TraerTodos (de clase): retorna un array de objetos de tipo Empleado, recuperados de la base de datos (con la descripción del perfil correspondiente y su foto).

Agregar (de instancia): agrega, a partir de la instancia actual, un nuevo registro en la tabla ***empleados*** (id,nombre, correo, clave, id_perfil, foto y sueldo), de la base de datos **usuarios_test**. Retorna **true**, si se pudo agregar, **false**, caso contrario.
Nota: *La foto guardarla en "./backend/empleados/fotos/", con el nombre formado por el nombre punto tipo punto hora, minutos y segundos del alta (Ejemplo: juan.105905.jpg).*

Modificar (de instancia): 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.
Nota: *Si la foto es pasada, guardarla en "./backend/empleados/fotos/", con el nombre formado por el nombre punto tipo punto hora, minutos y segundos del alta (Ejemplo: juan.105905.jpg). Caso contrario, sólo actualizar el campo de la base.*

Eliminar (de clase): elimina de la base de datos el registro coincidente con el id recibido cómo parámetro. Retorna **true**, si se pudo eliminar, **false**, caso contrario.

Empleado.php. Crear, en **./backend/clases**, la clase **Empleado** (hereda de Usuario) y posee atributos públicos (foto y sueldo). Implementa la interface ***ICRUD***.

Crear las siguientes páginas, en el directorio backend del proyecto:

AltaEmpleado.php: Se recibirán por POST todos los valores: ***nombre***, ***correo***, ***clave***, ***id_perfil***, ***sueldo*** y ***foto*** para registrar un empleado en la base de datos.
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

ListadoEmpleados.php: (GET) Se mostrará el listado **completo** de los empleados (obtenidos de la base de datos). Invocar al método TraerTodos.
Si se recibe el parámetro ***tabla*** con el valor ***mostrar***, retornará los datos en una tabla (HTML con cabecera). Si el parámetro no es pasado, retornará el array de objetos con formato JSON.
Nota*: preparar la tabla (HTML) con una columna extra para que muestre la imagen de la foto (50px X 50px).*

ModificarEmpleado.php: Se recibirán por POST los siguientes valores: ***empleado_json*** (id, nombre, correo, clave, id_perfil, sueldo y pathFoto, en formato de cadena JSON) y ***foto*** (para modificar un empleado en la base de datos. Invocar al método Modificar.
Nota: *El valor del id, será el id del empleado 'original', mientras que el resto de los valores serán los del empleado modificado.*
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

EliminarEmpleado.php: Recibe el parámetro ***id*** por POST y se deberá borrar el empleado (invocando al método Eliminar).
Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.

Se debe crear un archivo por cada entidad de TypeScript. Todos los métodos deben estar declarados dentro de clases.
Toda la comunicación con el backend, se realizará con AJAX. El pasaje de datos, con JSON.
El backend lo debe proveer el alumno (respetando nombres y tipos de parámetros).
Se deben respetar los nombres de los archivos, de las clases, de los métodos y de los parámetros.

Parte FRONTEND

Crear la siguiente jerarquía de clases en **Typescript** (en el namespace **Entidades**):

a. ***Persona***: nombre(cadena), correo(cadena) y clave(cadena) como atributos. Un constructor que reciba dos parámetros. Un método, *ToString():string,* que retorne la representación de la clase en formato **cadena** (preparar la cadena para que, al juntarse con el método ToJSON, forme una cadena JSON válida).

b. ***Usuario***, hereda de Persona, posee como atributo id(entero), id_perfil(entero) y perfil(cadena). Un constructor para inicializar los atributos. Sobrescribir los métodos ToString y ToJSON (agregándole los atributos propios).

c. ***Empleado***, hereda de Usuario, posee como atributos sueldo(numérico) y foto(cadena). Un constructor para inicializar los atributos. Sobrescribir los métodos ToString y ToJSON (agregándole los atributos propios).

Crear en **TypeScript** la clase ***Manejadora*** (en el namespace **ModeloParcial**) que posea los siguientes métodos y funcionalidades:

AgregarUsuarioJSON. Obtiene el nombre, el correo y la clave desde la página ***usuario_json.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AltaUsuarioJSON.php"*** *que invocará al método de instancia GuardarEnArchivo(), que agregará al usuario en **./backend/archivos**/usuarios.json. Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido*.
Informar por **consola** y **alert** el mensaje recibido.

MostrarUsuariosJSON. Recuperará (por ***AJAX***) todos los usuarios del archivo ***usuarios.json*** y generará un listado dinámico, crear una tabla HTML con cabecera (en el ***FRONTEND***) que mostrará toda la información de cada uno de los usuarios. Invocar a ***"./BACKEND/ListadoUsuariosJSON.php"****, recibe la petición (por GET) y retornará el listado de todos los usuarios en formato JSON.*

AgregarUsuario. Obtiene el nombre, el correo, la clave y el id_perfil desde la página ***usuario.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AltaUsuario.php"*** *que recibe por POST el **nombre**, el **correo**, la **clave** e **id_perfil**. Se invocará al método Agregar.*
*Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.*
Informar por **consola** y **alert** el mensaje recibido.

VerificarUsuario. Verifica que el usuario exista. Para ello, invocará (por ***AJAX***) a ***"./BACKEND/VerificarUsuario.php"***. *Se recibe por POST parámetro **usuario_json** (correo y clave, en formato de cadena JSON).*

*Retornar un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido (agregar el mensaje obtenido del método de clase).*
Se mostrará (por **consola** y **alert**) lo acontecido.

MostrarUsuarios. Recuperará (por ***AJAX***) todas los usuarios de la base de datos, invocando a ***"./BACKEND/ListadoUsuarios.php",*** *recibe la petición (por GET) y mostrará el listado **completo** de las usuarios (obtenidas de la base de datos). Invocar al método TraerTodos.*
*Si se recibe el parámetro **tabla** con el valor **mostrar**, retornará los datos en una tabla (HTML con cabecera). Si el parámetro no es pasado, retornará el array de objetos con formato JSON.*
Armar la tabla en el ***frontend*** y mostrar el listado en la página (*div id='divTabla'*).
NOTA*: *Agregar una columna (Acciones) al listado de usuarios que permita: **Eliminar** y **Modificar** al usuario elegido. Para ello, agregue dos botones (input [type=button]) que invoquen a las funciones EliminarUsuario y ModificarUsuario, respectivamente.

ModificarUsuario. Mostrará todos los datos del usuario que recibe por parámetro (objeto ***JSON*** armado en el listado), en el formulario. Permitirá modificar cualquier campo, a excepción del id, dejarlo como de sólo lectura.
Al pulsar el botón Modificar se invocará al método ***Modificar***, que realizará una petición por ***AJAX*** a ***"./BACKEND/ModificarUsuario.php"***, *que recibirán por POST los siguientes valores: **usuario_json** (id, nombre, correo, clave y id_perfil, en formato de cadena JSON), para modificar un usuario en la base de datos. Invocar al método Modificar.*
Retornar un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Refrescar el listado sólo si se pudo modificar, caso contrario, informar (por alert y consola) de lo acontecido.
Limpiar el formulario sólo si se pudo modificar.

EliminarUsuario. Recibe como parámetro al objeto ***JSON*** (armado en el listado) que se ha de eliminar. Pedir confirmación, mostrando nombre y correo, antes de eliminar.
Si se confirma se invocará (por ***AJAX***) a ***"./BACKEND/EliminarUsuario.php"*** *pasándole cómo parámetro el **id** por POST y se deberá borrar el usuario (invocando al método Eliminar).*
*Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.*
Informar por consola y alert lo acontecido. Refrescar el listado para visualizar los cambios.

Agregar en el namespace Modelo, la clase ***ManejadoraEmpleados*** que contenga los siguientes métodos:

AgregarEmpleado, MostrarEmpleado, EliminarEmpleado y ModificarEmpleado.

MostrarEmpleados. Enviará (desde AJAX) hacia **"./BACKEND/ListadoEmpleados.php"**, una petición por GET, que mostrará el listado **completo** de los empleados (obtenidos de la base de datos) en una tabla (HTML con cabecera). Invocar al método TraerTodos.
Si se recibe el parámetro ***tabla*** con el valor ***mostrar***, retornará los datos en una tabla (HTML con cabecera). Si el parámetro no es pasado, retornará el array de objetos con formato JSON.
***Nota**: preparar la tabla (HTML) con una columna extra para que muestre la imagen de la foto (50px X 50px).*
Armar la tabla en el ***frontend*** y mostrar el listado en la página (*div id='divTablaEmpleados'*). Se deberá mostrar el ID, nombre, correo, perfil sueldo y foto de cada empleado.

AgregarEmpleado. Obtiene los datos del empleado (incluyendo la foto) desde la página ***empleado.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AltaEmpleado.php"*** *que recibirá por POST todos los valores: **nombre**, **correo**, **clave**, **id_perfil**, **sueldo** y **foto** para registrar un empleado en la base de datos..*
*Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.*
***Nota:** La foto guardarla en "./backend/empleados/fotos/", con el nombre formado por el nombre punto tipo punto hora, minutos y segundos del alta (Ejemplo: juan.105905.jpg).*