Modelo - PRIMER PARCIAL – LAB. III - 2024

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/FETCH.
El pasaje de datos, con JSON.
Respetar nombres y tipos de parámetros en cada petición al BACKEND.
Se deben respetar los nombres de los archivos, de las clases, de los métodos y de los parámetros.

Parte 1

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. Un método ToJSON():JSON, que retornará la representación del objeto en formato ***JSON***. Se debe de reutilizar el método ToString de la clase Persona.

C. ***Empleado***, hereda de Usuario, posee como atributos id(entero), sueldo(numérico) y foto(cadena).
Un constructor para inicializar los atributos.

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/FETCH) hacia ***"http://localhost:2024/usuarioJSON"*** por el método POST. 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/FETCH) todos los usuarios del archivo ***usuarios.json***, invocando a ***"http://localhost:2024/usuarioJSON"***, que recibe la petición (por GET) y retornará un JSON (éxito:true/false, usuarios:array/null) para crear un listado dinámico (en el FRONTEND) que mostrará toda la información de cada uno de los usuarios.

VerificarUsuarioJSON. Verifica que el usuario exista. Para ello, invocará (por AJAX/FETCH) a ***"http:localhost:2024/usuarioJSON/verificar"***. Se recibe por POST (el correo y clave, como objeto JSON) y retornará un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido. Se mostrará (por consola y alert) lo acontecido.

Parte 2

AgregarUsuarioBD. Obtiene el **nombre**, el **correo**, la **clave** y el **id_perfil** desde la página ***usuario.html*** y se enviará (por AJAX/FETCH) hacia ***"http://localhost:2024/usuarioBD"*** que recibe por POST los datos enviados. Se retornará un JSON que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.
Informar por consola y alert el mensaje recibido.

MostrarUsuariosBD. Recuperará (por AJAX/FETCH) todas los usuarios de la base de datos, invocando a ***"http://localhost:2024/usuarioBD"***, recibe la petición (por GET) y retornará un JSON (éxito:true/false, usuarios:array/null) para crear un listado dinámico (en el FRONTEND).
Informar por consola y alert el mensaje recibido y mostrar el listado en la página (div id='divTabla').

ModificarUsuarioBD. Mostrará todos los datos del usuario que recibe por parámetro (objeto JSON), en el formulario. Permitirá modificar cualquier campo, a excepción del ***id***, dejarlo como de sólo lectura.
Al pulsar el botón ***Modificar usuario*** se invocará (por AJAX/FETCH) a ***"http://localhost:2024/usuarioBD"***, que recibirán por PUT 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.
Retornará 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.

EliminarUsuarioBD. Recibe como parámetro al objeto JSON que se ha de eliminar. Pedir confirmación, mostrando nombre y correo, antes de eliminar.
Si se confirma se invocará (por AJAX/FETCH) a ***"http://localhost:2024/usuarioBD"*** pasándole cómo parámetro el ***id*** por DELETE y se deberá borrar el usuario.
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.

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.

Parte 3

En la clase ***Manejadora***, agregar los siguientes métodos:

AgregarEmpleado, MostrarEmpleados, EliminarEmpleado y ModificarEmpleado.

Al cargar la página ***empleado.html***, se deberá cargar el listado de empleados obtenidos desde la base de datos, para ello se invocará al método ***MostrarEmpleados*** que enviará (desde AJAX/FETCH) hacia ***"http://localhost:2024/empleadoBD"***, una petición por GET, que retornará un JSON (éxito:true/false, usuarios:array/null) para crear un listado dinámico (en el FRONTEND).

Nota: preparar la tabla (HTML) con una columna extra para que muestre la imagen de la foto (50px X 50px).

Mostrar el listado en la página (div id='divTablaEmpleados').

AgregarEmpleado. Obtiene los datos del empleado (incluyendo la foto) desde la página ***empleado.html*** y se enviará (por AJAX/FETCH) hacia ***"http://localhost:2024/empleadoBD"*** que recibirá por POST ***obj_empleado*** (nombre, correo, clave, id_perfil y sueldo, en formato de cadena JSON) 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 se guardará en "./public/fotos/", con el nombre formado por el **nombre** punto **id_perfil** (Ejemplo: juan.1.jpg).*

Informar por consola y alert el mensaje recibido.
Refrescar el listado de empleados.

ModificarEmpleado. Mostrará todos los datos del usuario que recibe por parámetro (***objeto JSON***), en el formulario, incluida la foto (mostrarla en "imgFoto"). Permitirá modificar cualquier campo, a excepción del ***id***, dejarlo como de sólo lectura.
Al pulsar el botón *Modificar* empleado se invocará (por AJAX/FETCH) a ***"http://localhost:2024/empleadoBD/id"***, que recibirán por PUT ***empleado_json*** (nombre, correo, clave, id_perfil, foto y sueldo, en formato de cadena JSON) y ***foto*** (para modificar un empleado en la base de datos).

Nota: El valor del id, será el id del empleado 'original', mientras que el resto de los valores serán los del empleado modificado. Dicho valor se pasará cómo parámetro de ruta.

Refrescar el listado sólo si se pudo modificar, caso contrario, informar (por alert y consola) de lo acontecido.

EliminarEmpleado. Recibe como parámetro al objeto JSON que se ha de eliminar. Pedir confirmación, mostrando nombre y sueldo, antes de eliminar.
Si se confirma se invocará (por AJAX/FETCH) a ***"http://localhost:2024/empleadoBD/id"***, dónde id será el parámetro de ruta por DELETE. 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.