Recuperatorio - PRIMER PARCIAL – LABORATORIO 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 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 1 (hasta un 5)

Crear una aplicación Web que permita realizar un ABM de distintos cocineros y recetas.

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

a. ***Persona***: email(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. ***Cocinero***, hereda de Persona, posee como atributo especialidad(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. ***Receta,*** posee como atributos id(entero), nombre(cadena), ingredientes(cadena), tipo(cadena) y foto(cadena). Un constructor para inicializar los atributos. Un método ToJSON():JSON, que retornará la representación del objeto en formato JSON.

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

AgregarCocinero. Obtiene la especialidad, el email y la clave desde la página ***cocinero.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AltaCocinero.php"*** *que creará un objeto de tipo cocinero e invocará al método de instancia GuardarEnArchivo(), que agregará al cocinero en **./archivos**/cocinero.json. Retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido*.
Informar por **consola** y **alert** el mensaje recibido.

MostrarCocineros. Recuperará (por ***AJAX***) todos los cocineros del archivo ***cocinero.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 cocineros. Invocar a ***"./BACKEND/ListadoCocinero.php"****, recibe la petición (por GET) y mostrará el listado de todos los cocineros en formato JSON.*

VerificarExistencia. Verifica que el cocinero que se quiere agregar no exista. Para ello, invocará (por ***AJAX***) a ***"./BACKEND/VerificarCocinero.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 especialidad del cocinero, separado con un guión bajo (maru_botana@gmail.com_pastelero) 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).*
Se mostrará (por **consola** y **alert**) lo acontecido.

AgregarRecetaSinFoto. Obtiene el nombre, los ingredientes y el tipo desde la página ***receta.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AltaRecetaSinFoto.php"*** *que recibe por POST el **nombre**, los **ingredientes** y el **tipo**. Se invocará al método Agregar (agrega, a partir de la instancia actual, un nuevo registro en la tabla **recetas** (id, nombre, ingredientes, tipo, foto), de la base de datos **recetas_bd**. Retorna **true**, si se pudo agregar, **false**, caso contrario).*
*Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.*
Informar por **consola** y **alert** el mensaje recibido.

MostrarRecetas. Recuperará (por ***AJAX***) todas las recetas de la base de datos, invocando a ***"./BACKEND/ListadoRecetas.php",*** *recibe la petición (por GET) y mostrará el listado **completo** de las recetas (obtenidas de la base de datos) en una tabla (HTML con cabecera). Invocar al método Traer (retorna un array de objetos de tipo Receta, recuperados de la base de datos.).*
***Nota**: preparar la tabla (HTML) con una columna extra para que muestre la imagen de la foto (si es que la tiene).*
Informar por **consola** y **alert** el mensaje recibido y mostrar el listado en la página (*div id='divTabla'*)

Parte 2 (hasta un 6)

En la clase Manejadora, agregar la interface Iparte2 con los siguientes métodos:

AgregarReceta, EliminarReceta y ModificarReceta.

AgregarVerificarReceta. Obtiene el nombre, los ingredientes, el tipo y la foto desde la página ***receta.html*** y se enviará (por ***AJAX***) hacia ***"./BACKEND/AgregarReceta.php"*** *que recibirá por POST todos los valores: **nombre**, **ingredientes**, **tipo** y **foto** para registrar una receta en la base de datos.*
***Verificar** la previa existencia de la receta invocando al método Existe (retorna **true**, si la instancia actual está en el array de objetos de tipo Receta que recibe como parámetro (comparar por nombre y tipo). Caso contrario retorna **false**.). Se le pasará como parámetro el array que retorna el método Traer.*
Si la receta ya existe en la base de datos, se retornará un mensaje que indique lo acontecido.
*Si la receta no existe, se invocará al método Agregar. La imagen guardarla en "./recetas/imagenes/", con el nombre formado por el **nombre** punto **tipo** punto hora, minutos y segundos del alta (**Ejemplo: chocotorta.pasteleria.105905.jpg**).*
*Se retornará un **JSON** que contendrá: éxito(bool) y mensaje(string) indicando lo acontecido.*
Informar por **consola** y **alert** el mensaje recibido.
Refrescar el listado de recetas.

NOTA:
*Agregar una columna (Acciones) al listado de recetas que permita: **Eliminar** y **Modificar** a la receta elegida. Para ello, agregue dos botones (input [type=button]) que invoquen a las funciones EliminarReceta y ModificarReceta, respectivamente.*

EliminarReceta. Recibe como parámetro al objeto ***JSON*** que se ha de eliminar. Pedir confirmación, mostrando nombre y tipo, antes de eliminar.
Si se confirma se invocará (por ***AJAX***) a ***"./BACKEND/EliminarReceta.php"*** *pasándole cómo parámetro **receta_json** (id, nombre, y tipo, en formato de cadena JSON) por POST, más el parámetro **accion** con valor "**borrar**", se deberá borrar la receta (invocando al método Eliminar → elimina de la base de datos el registro coincidente con la instancia actual (comparar por nombre y tipo). Retorna **true**, si se pudo eliminar, **false**, caso contrario).*
Si se pudo borrar en la base de datos, invocar al método GuardarEnArchivo (escribirá en un archivo de texto (recetas_borradas.txt) toda la información de la receta más la nueva ubicación de la foto. La foto se moverá al subdirectorio "./recetasBorradas/", con el

*nombre formado por el **id** punto **nombre** punto '**borrado**' punto hora, minutos y segundos del borrado. **Ejemplo: 123.puchero.borrado.105905.jpg**).*
Retornar 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.

ModificarReceta. Mostrará todos los datos de la receta 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 receta se invocará (por ***AJAX***) a ***"./BACKEND/ModificarReceta.php"*** *Se recibirán por POST los siguientes valores: **receta_json** (id, nombre, ingredientes, tipo y pathFoto, en formato de cadena JSON) y **foto** para modificar una receta en la base de datos. Invocar al método 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).*
Refrescar el listado sólo si se pudo modificar, caso contrario, informar (por alert y consola) de lo acontecido.

Parte 3

En la clase Manejadora, agregar la interface IParte3:

FiltrarRecetas. Invocará (por ***AJAX***) a ***"./BACKEND/FiltrarReceta.php"*** *Si se recibe por POST el **nombre**, se mostrarán en una tabla (HTML) las recetas cuyo nombre coincidan con el pasado por parámetro.*
*Si se recibe por POST el **tipo**, se mostrarán en una tabla (HTML) las recetas cuyo tipo coincida con el pasado por parámetro.*
*Si se recibe por POST el **nombre** y el **tipo**, se mostrarán en una tabla (HTML) las recetas cuyo nombre y pais coincidan con los pasados por parámetro).*
Refrescar el listado.

MostrarRecetasBorradas. Invocará (por ***AJAX***) a ***"./BACKEND/MostrarBorrados.php"*** *muestra todo lo registrado en el archivo de texto "recetas_borradas.txt". Para ello, agregar un método estático (en Receta), llamado **MostrarBorrados.***
Armar un listado dinámico, crear una tabla HTML con cabecera (en el *FRONTEND*) que mostrará toda la información de cada una de las recetas borradas (con su respectiva foto).

Parte 4

En la clase Manejadora, agregar la interface IParte4:

CargarTiposJSON. Cargará (por ***AJAX***) en el combo (cboTipos) con el contenido del archivo **"./BACKEND/tipos_receta.json"**. Los tipos no deben estar repetidos en el combo (hacer que se carguen, pero que no se dupliquen).