1. Crear una variable $a que contenga un array ('a'=>1, 'b'=>2, 'c'=>'Yo <3 JSON'):
    a. Hacer echo de la variable $a.
    b. Utilizando json_encode, convertir el array en un json.
    c. Hacer echo de la variable $a.
    d. Crear una nueva variable $b que contenga el json_encode de la variable $a.
    e. Hacer echo de $b.
    f. Imprimir la frase “Yo <3 JSON | 1 | 2 |” utilizando los datos de la variable $b.

2. Crear un archivo nuevo llamado **archivos.php**.
    a. Crear una función que compruebe si existe un archivo llamado **texto.txt** en el mismo directorio que **archivos.php**. En el caso que exista debe abrirlo con derechos de lectura y escritura para agregarle información. En caso de que no exista, debe crearlo con derechos de lectura y escritura.
    b. Que se escriba “Hola mundo! testing.” 100 veces en el archivo 1 vez por renglón. Luego de esto que se cierre el archivo.
    c. Mostrar los contenidos del archivo **texto.txt** leyendo todo el archivo junto.
    d. Mostrar los contenidos del archivo **texto.txt** leyendo e imprimiendo línea por línea.
    e. Borrar el archivo **texto.txt**
    f. Crear un nuevo archivo llamado **texto2.txt** que contenga el texto: “Hola nuevamente mundo!”.
    g. Escribir en el archivo **texto2.txt** “¿Este texto pisa el anterior?”. Y nos fijamos si efectivamente piso el texto que estaba en el archivo.
    h. Escribir en el archivo **texto2.txt**: “YA NO!” luego del texto que ya existe.

3. Crear un archivo **categorias.json** con los siguientes datos:

```
{“categorias” : [
        {“id”: 1, “nombre”: “Deportes”},
        {“id”: 2, “nombre”: “Historia”},
        {“id”: 3, “nombre”: “Espectáculos”},
        {“id”: 4, “nombre”: “Geografía”},
        {“id”: 5, “nombre”: “Arte”},
]}
```

    a. Leer el archivo e **imprimir** un checkbox por cada categoría, capaz de ser enviado como array en un formulario. El value de cada checkbox será el id y el label mostrado a su derecha será el nombre proveniente del json como vemos en el en el array.

4. Crear un nuevo archivo PHP.
    a. Crear una función que defina una variable de tipo string y haga echo de dicha variable.
    b. Agregar a la función un echo del resultado de encriptar la variable con la función md5.

c. Agregar a la función un echo del resultado de encriptar la variable con la función sha1.
d. Agregar a la función un echo del resultado de encriptar la variable con password_hash, utilizando como algoritmo: PASSWORD_DEFAULT.
e. Agregar a la función un echo del resultado de encriptar la variable con password_hash, utilizando como algoritmo: PASSWORD_BCRYPT.
f. Agregar a la función un echo del resultado de encriptar la variable con password_hash, utilizando como algoritmo: PASSWORD_BCRYPT y utilizando la opción “salt”.
g. Comparar los 5 resultados.

5. Modificar **register.html/register.php** (puede ser el archivo utilizado las clases anteriores o el archivo utilizado en el trabajo integrador) para que:
   a. Valide los datos del formulario.
   b. En caso de error aclare los errores.
   c. En caso de error rellene los campos que el usuario ya había completado.
   d. Guarde los datos del usuario en un array. (la contraseña debe estar encriptada!)
   e. Convierta el array en JSON.
   f. Guarde el usuario en un archivo de texto.
   g. En caso de éxito redirija a una página de éxito.

   Nota: La registración es un proceso que se ejecuta muchas veces, por lo tanto, cada usuario nuevo, debe agregarse al final del array de usuarios. Es recomendable que el archivo contenga una estructura de tipo: {“usuarios”: [{...},{...},{...},{...},{...}]} en donde {...} es el json_encode del array con los datos del usuario particular que se registra en el momento del guardado.

6. Modificar **login.html/login.php** (puede ser el archivo utilizado las clases anteriores o el archivo utilizado en el trabajo integrador) para que al enviar el formulario chequee en el archivo de texto generado en el punto anterior si el usuario existe o no. En caso de existir debe darle la bienvenida al usuario, en caso de que el usuario no sea encontrado, debe indicar que el usuario no existe.

7. Modificar **register.html/register.php** para que al registrar un usuario valide en el archivo de texto que no exista un usuario igual (se puede validar por username, email o dni según el tipo de formulario).

Ejercicios Complementarios

1. Generar archivos/funciones para reemplazar **cada uno de los pedidos AJAX** realizados en la clase de AJAX (formulario de registración a TECHO). Así, podríamos manejar dicho formulario de forma local. Algunas cuestiones a tener en cuenta:
    a. En el principio de los archivos php que genera los JSON agregar al principio del archivo luego de <?php :

    ```
    header('content-type: application/json; charset=utf-8');
    header("access-control-allow-origin: *");
    ```

    b. registrarPersona debe guardar el usuario en un archivo de texto.
    c. registrarPersona valida los datos desde el lado del servidor incluyendo que el dni y el email del usuario aún no existan.
    d. Para devolver los datos en PHP en un pedido AJAX debemos preparar nuestras variables en formato JSON y simplemente imprimir esa variable (ejemplo: Si tuviésemos un array con la respuesta deseada simplemente hay que terminar el archivo con la línea: **echo json_encode($respuesta);exit;**)
    e. Si bien no es necesario generar todos los países, ciudades, regiones y sedes se recomienda validar con varias opciones.
    f. En los casos de países, ciudades, regiones y sedes recordar agregar la opción “Otro” en todos los casos.
    g. Recordar que los métodos de la “API” de TECHO devolvía tanto el atributo **error** como el atributo **contenido** en todas sus respuestas.