**Examen Final 2ª Evaluación. Tema 6, 7, 8, 9**

**Apellidos, Nombre:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Las capturas están al final del documento**

**Instrucciones:**

Se está utilizando el programa de vigilancia de vuestras pantallas. Se penalizará con un 0 en esta prueba cualquier intento de copia o comunicación entre vosotros. Podéis, sin embargo, acceder al material del curso, **SOLO SE PUEDE CONSULTAR MATERIAL O PRÁCTICAS, NO COMUNICARSE. Los MÓVILES PERMANECERÁN APAGADOS** y el **cable de red** del ordenador **desconectado,** durante todo el examen. Sólo se pondrá el cable de red para bajar los documentos del examen de Classroom y para realizar, posteriormente, la entrega.

**Enunciado:**

Disponemos de una base de datos “**telefonos\_examen”** con usuario **“root” y sin contraseña,** que nuestro sistema utiliza para hacer consultas.

|  |  |
| --- | --- |
| **Tabla** | **Propósito de la tabla** |
| Telefonos | Contiene el id(int 11), modelo(varchar 50), marca (varchar 50), memoria(int 11), precio(float (12,2)) y fecha\_adquisición (date) |

Si por ejemplo, queremos insertar un telefono, la fila de la tabla “telefonos” será “A70” “LG” “8” “120” “2019-10-11” y si por ejemplo queremos borrar daremos el id del móvil a borrar.

**Diagrama ER** de la base de datos:

Hay una única tabla con id que es clave primaria autoincremental.

**Codigo SQL de creación de las tablas:**

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

CREATE TABLE `telefonos` (

  `id` int(11) NOT NULL,

  `modelo` varchar(50) NOT NULL,

  `marca` varchar(50) NOT NULL,

  `memoria` int(11) NOT NULL,

  `precio` float(10,2) NOT NULL,

  `fecha\_adquisicion` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `telefonos` (`id`, `modelo`, `marca`, `memoria`, `precio`, `fecha\_adquisicion`) VALUES

(1, 'Galaxy A52s', 'samsung', 6, 200.00, '2022-01-22'),

(2, 'Iphone 13', 'apple', 8, 300.00, '2021-12-02'),

(3, 'Galaxy A22', 'samsung', 4, 300.00, '2021-12-22'),

(4, 'Galaxy A22', 'samsung', 4, 250.00, '2021-12-24'),

(5, 'Galaxy S21', 'samsung', 12, 235.00, '2021-12-25'),

(6, 'iPhone 13 Pro', 'apple', 4, 900.00, '2022-01-03'),

(7, 'Redmi Note 10S', 'xiaomi', 6, 433.00, '2022-01-04'),

(8, 'Redmi Note 10S', 'huawei', 6, 199.00, '2022-01-12'),

(9, 'Find X3 Lite ', 'oppo', 8, 321.00, '2022-01-25'),

(10, 'V10 H960', 'lg', 6, 200.00, '2022-01-23'),

(11, 'Xperia 1 III', 'sony', 12, 320.00, '2022-01-27'),

(12, 'Alcatel 1B 502H', 'alcatel', 12, 367.00, '2022-02-17'),

(13, 'Realme GT Master', 'realme', 6, 325.00, '2022-02-25');

ALTER TABLE `telefonos`

  ADD PRIMARY KEY (`id`);

ALTER TABLE `telefonos`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=14;

COMMIT;

Realizar, **sobre la ruta “examen\_2eval\_alumnos/apellido1\_apellido2\_nombre”,** los siguientes ejercicios. Se deben **elegir 2 ejercicios** **entre los números (1, 2 y 3)** y **uno entre (4 o 5)** para que la suma sea 10. **Marca con una X** los ejercicios que has escogido:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EJERCICIO 1** | **EJERCICIO 2** | **EJERCICIO 3** | **EJERCICIO 4** | **EJERCICIO 5** |
|  |  |  |  |  |

En cada ejercicio se muestra el árbol de directorios que se necesita:

1. Crear un **servicio web** que muestre los teléfonos de la base de datos y permita insertar un teléfono en la base de datos tal y como viene en las fotos adjuntas. **No se utilizará WSDL** para este servicio. Deberá cumplir con las especificaciones de programación orientada a objetos. **(3,4P)**
2. Para ello, se creará un archivo **Funciones.php con la clase Funciones,** que se encuentra en la carpeta **“funciones”,** con los métodos: **(0,6P)**

* **estaVacio($campo, $valor) :** mira si el valor de algún campo del formulario está vacío.
* **getTelefonos():** obtiene los teléfonos de la base de datos y muestra todos los campos.
* **insertTelefono($modelo,$marca,$memoria,$precio,$adquisicion): inserta** un teléfono en la base datos, si no pudo insertarlo muestra un error.

1. Después desde un fichero **servidor.php** situado en la carpeta **“funciones”** se publicará la clase. **(0,5P)**
2. Prueba a crear un cliente **cliente.php** que consuma las dos últimas funciones (getTelefonos e insertTeléfono)). Como queremos que se asemeje a la arquitectura MVC, el cliente.php (que contiene el controlador) redirigirá al archivo **“vista-cliente.php**” que está en la carpeta “**vista”**. **(0,5P)**
3. En **vista\_cliente.php** estará todo el html necesario y el mínimo código php con las llamadas a las funciones del cliente. **(0,6P)**
4. Se deberá crear un archivo (controlador), **insertar.php, que** hace un require\_once del archivo **vista\_insertar.php** situado en la carpeta **“vista”** y que redirecciona a la función del cliente. **(0,6P)**
5. Crear un archivo **vista\_insertar.php** con el try/catch del cliente. **(0,6P)**

La estructura de carpetas queda así:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Crear un **servicio web** que muestre los teléfonos de la base de datos, el precio del teléfono Iphone 13 y las marcas de los teléfonos de la BBDD. **Se utilizará WSDL** y para la realización de este servicio deberán crearse: **(3,4P)**
2. Un archivo **Funciones.php con la clase Funciones,** que se encuentra en la carpeta **“funciones”,** con los métodos: **(0,8P)**

* **getTelefonos():** obtiene los teléfonos de la base de datos y muestra todos los campos.
* **getPrecio($cod):** recibe el código del telefono y devuelve el precio.
* **getMarcas():** obtiene las marcas de los teléfonos.

1. Un **cliente.php** que consuma el servicio y llame a las funciones getTelefono, getPrecio y getMarcas. Como queremos que se asemeje a la arquitectura MVC, el cliente.php (que contiene el controlador) redirigirá al archivo **“vista-cliente.php**” que está en la carpeta “**vista”**. **(0,7P)**
2. En **vista\_cliente.php** estará todo el html necesario y el mínimo código php con las llamadas a las funciones del cliente. **(0,7P)**
3. Un fichero **servidor.php** situado en la carpeta **“funciones”** que publicará la clase **Funciones**. **(0,5P)**
4. Un archivo **crearWSDL.php** en la carpeta “**funciones”** que llamará a WSDLDocument. Para que WSDL se deberá copiar el archivo **WSDLDocument.php** en la carpeta “**funciones**”. **(0,7P)**

La estructura del directorio quedará:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Crear un **servicio web** que dado el identificador de un teléfono, permita **borrarlo** de la base de datos tal y como viene en las fotos adjuntas. **No se utilizará el archivo crearWSDL para este servicio (se usará el parámetro wsdl pasado por la URL).** Deberá cumplir con las especificaciones de programación orientada a objetos. **(3,4P)**
2. Para ello, se creará un servicio web **Funciones.php con la clase Funciones,** que se encuentra en la carpeta **“funciones”,** con los métodos: **(0,5P)**

* **getTelefonos():** obtiene los teléfonos de la base de datos y muestra todos los campos.
* **borrarTelefono($numero):** borra el teléfono cuyo id coincide con el parámetro de entrada de la base datos.

1. Después desde un fichero **servidor.php,** situado en la carpeta **“funciones”,** se publicará esta clase. Para crear el documento, habrá que añadir el archivo **WSDLDocument.php** en la carpeta “**funciones”. (0,4P)**
2. Prueba a crear un cliente **cliente.php** que consuma las dos últimas funciones (getTelefonos y borrarTeléfono)). Como queremos que se asemeje a la arquitectura MVC, el **cliente.php** (que contiene el controlador) redirigirá al archivo **“vista.php**” que está en la carpeta “**vista”**. **(0,5P)**
3. En **vista.php** estará el try/catch del cliente. **(0,5P)**
4. Se deberá crear un archivo **borrar.php, que** hace un require\_once del archivo de la carpeta **“vista”** que se llama **vista\_borrar.php** y redirecciona a la función del cliente. **(0,5P)**
5. En **vista\_borrar.php** está el html que pide el id del teléfono a borrar**. (0,5P)**
6. Qué muestra la url <http://localhost/examen_alumnos/tu_nombre/ejercicio3/funciones/servidor.php?wsdl>. Explica cuidadosamente sobre el pantallazo cada parte del documento. **(0,5P)**

La estructura de carpetas queda así:

Interfaz de usuario gráfica, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Crea mediante **Ajax** una aplicación, en MVC, que realice una **búsqueda** de los teléfonos que hay en la base de datos, según un **criterio** de entrada que es el **modelo. Si no se introduce nada en el buscador, mostrará todos los teléfonos. Si se introduce una letra, conjunto de letras o una palabra buscará ese patrón.** Para poder hacer esto, se debe utilizar obligatoriamente **jQuery, JSON y javascript** como en los ejercicios que hemos visto en clase. **(3,2P)**
2. Se creará un archivo **index.php** controlador que lance la aplicación y que llame a **vistas/index-vista.php. (0,5P)**
3. **Index-vista.php,** que se encuentra en la carpeta **vista**, contiene el código html de la vista y la llamada a la función listarTelefonos() al hacer click en el botón buscar. **(0,5P)**
4. **lib.js,** que se encuentra en la carpeta js, contiene el código javascript y la función **listarTelefonos** (), que hace una llamada a $.ajax de jQuery y si el criterio de búsqueda es correcto, lista los teléfonos encontrados mediante json. **(0,6P)**
5. En la carpeta **modelos,** se encuentra **Telefono.php,** que contiene la **clase Telefono** que implementa JsonSerializable, con los métodos: **(0,7P)**

**- construct($r**): inicializa las variables privadas de clase: id, modelo, marca, memoria, precio y fecha\_adquisición.

**- getId()**

**- setId()**

**- getModelo()**

**- getMarca()**

**- getMemoria()**

**- getPrecio()**

**- getFechaAdquisicion()**

**- setModelo($modelo)**

**- setPrecio($precio)**

**- setMemoria($memoria)**

**- setMarca($marca)**

**- setFechaAdquisicion($fecha\_adquisición)**

**- jsonSerialize()** que serializa los datos a formato Json

1. El archivo **BD.php**, con la **clase BD** implementa los métodos **getTelefonos(), getTelefono($id) y listarTelefonos($criterio)** . **(0,5P)**
2. El archivo **listar.php,** que se encuentra en la carpeta api y decodifica los datos de json, hace la búsqueda llamando al método de la **BD::listarTelefonos** y devuelve los datos codificados en JSON. **(0,4P)**

La estructura es la siguiente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Realizar la **validación del formulario que inserta el teléfono** utilizando **jQuery** pero en lugar de mostrar los errores de validación integrados en la página web, utilizar la función alert de JavaScript. Si los datos del formulario son correctos, **insertará el teléfono en la base de datos**. Lo primero será importar el script de jquery. Se deberá asociar al evento click del botón del formulario para que se llame al fichero **validar.php.** Validar contendrá las funciones **validar\_modelo($modelo), validar\_marca($marca),validar\_memoria($memoria), validar\_precio($precio), validar\_adquisicion($adquisicion) y validarFormulario($valores).** Hay que tener en cuenta que estos métodos deberán llamarse y tiene que devolverse un fichero JSON. Para ello se puede utilizar la función json\_encode de PHP. Lo más adecuado puede ser crear un array en el propio fichero e ir añadiendo mensajes en caso que se produzca error. Un posible ejemplo de validación del nombre podría ser: if (!validar\_modelo($\_POST['modelo'])) $respuesta['errorModelo'] = utf8\_encode("El modelo debe ser introducido"); Ese array deberá ser el parámetro que reciba el método json\_encode: echo json\_encode($repuesta); Por último, en el success de la llamada Ajax mostraremos los alert con los datos que nos devuelve el fichero validar.php (3,2P).

Archivos a entregar:

1. **formulario.php:** Contiene el código html, la llamada a insertTelefono y el código jQuery con la llamada a Ajax. **(1,3P)**
2. **Funciones.php:** Contiene las funciones para conectar con la base de datos e insertar el teléfono **insertTelefono($modelo,$marca,$memoria,$precio,$adquisicion) y getTelefonos(). (0,6P)**
3. **validar.php:** Contiene las funciones para validar el formulario. **validar\_modelo($modelo), validar\_marca($marca),validar\_memoria($memoria), validar\_precio($precio), validar\_adquisicion($adquisicion) y validarFormulario($valores).** **(1,3P)**

**Ejemplos de páginas:**

**Ejercicio1: Insertar.php**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

**Ejercicio 2:cliente.php**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Ejercicio3:borrar.php**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente**

**Ejercicio4:**

**Tabla

Descripción generada automáticamenteTabla

Descripción generada automáticamente**

**Tabla

Descripción generada automáticamente**

**Ejercicio5**

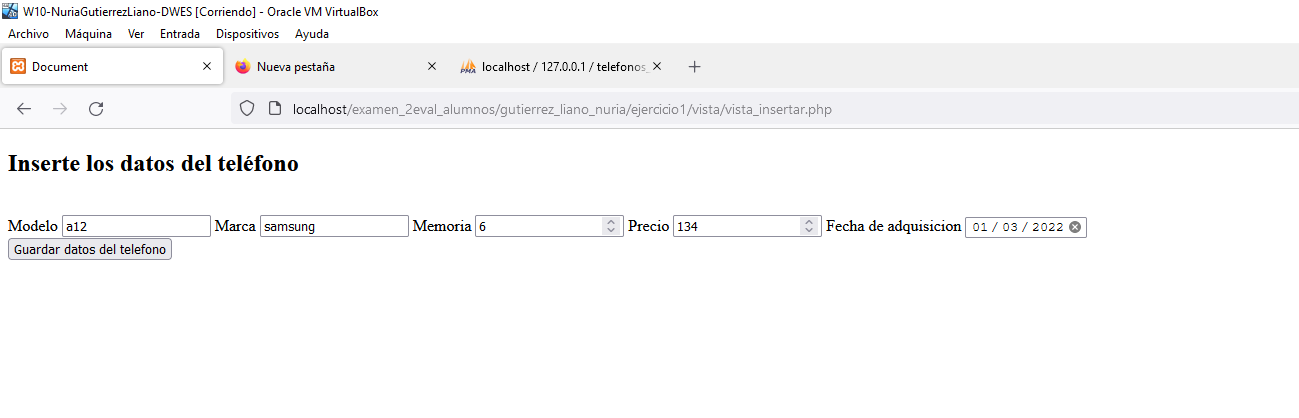
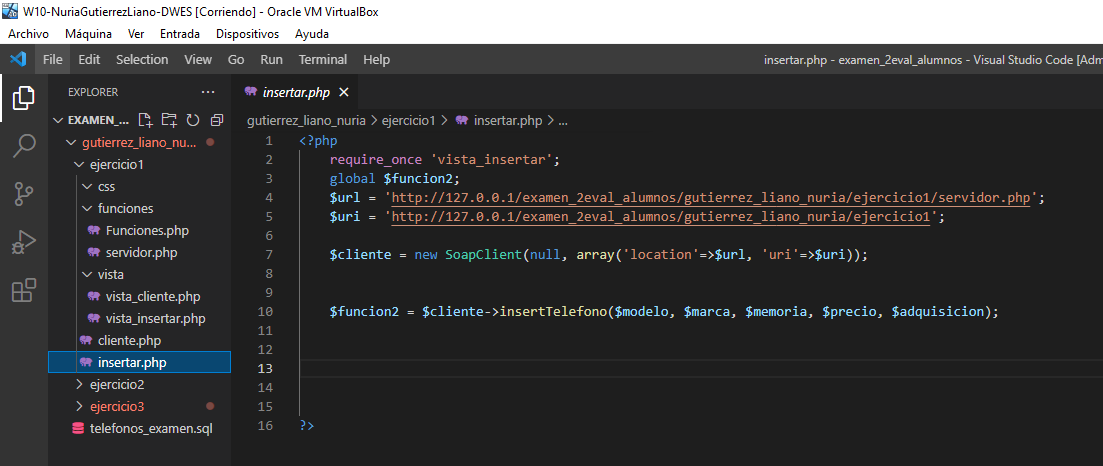
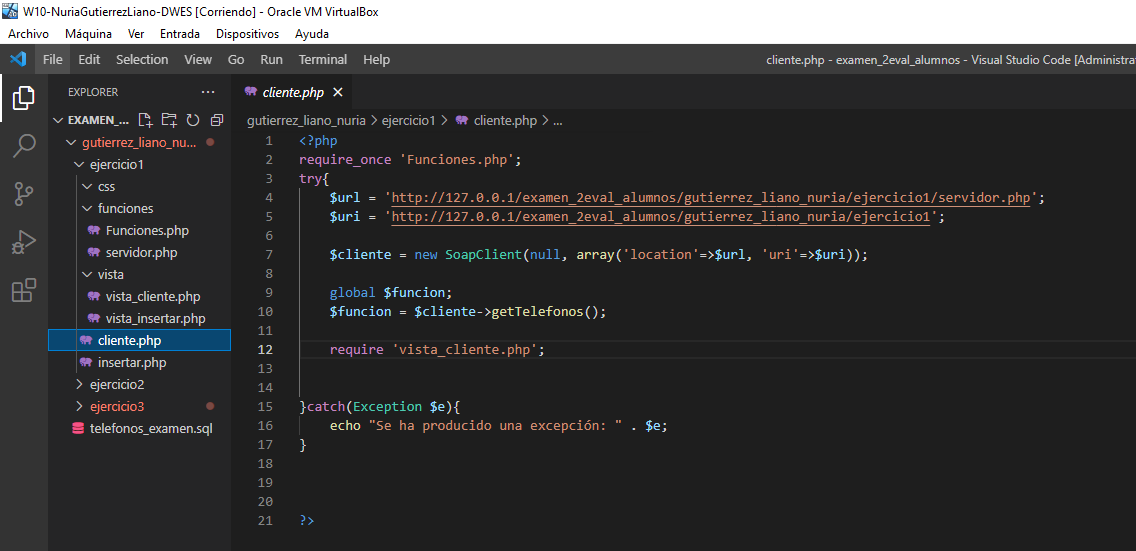
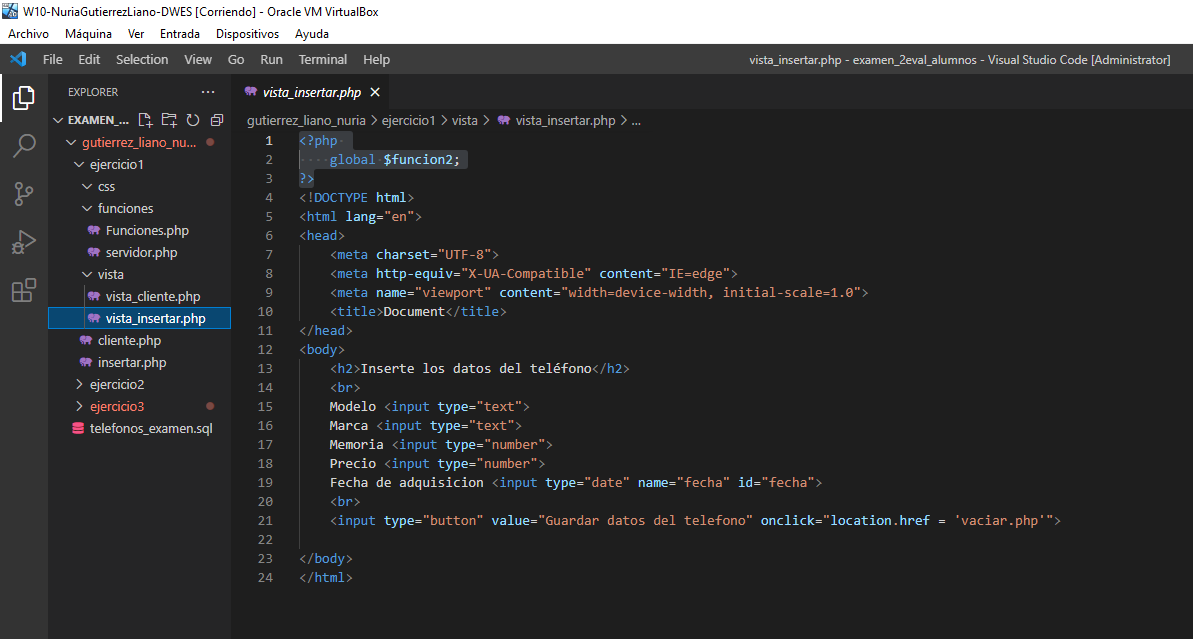
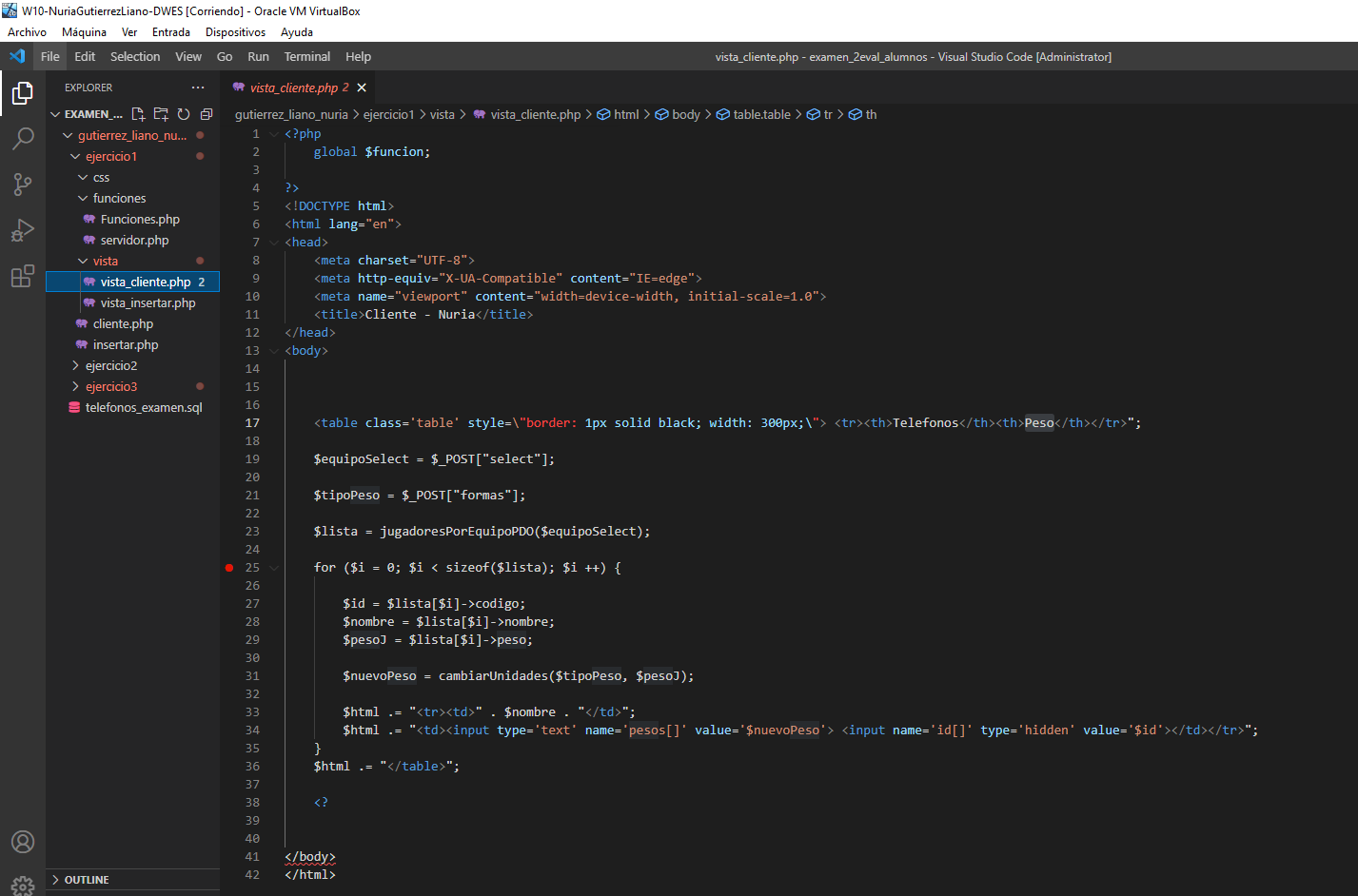
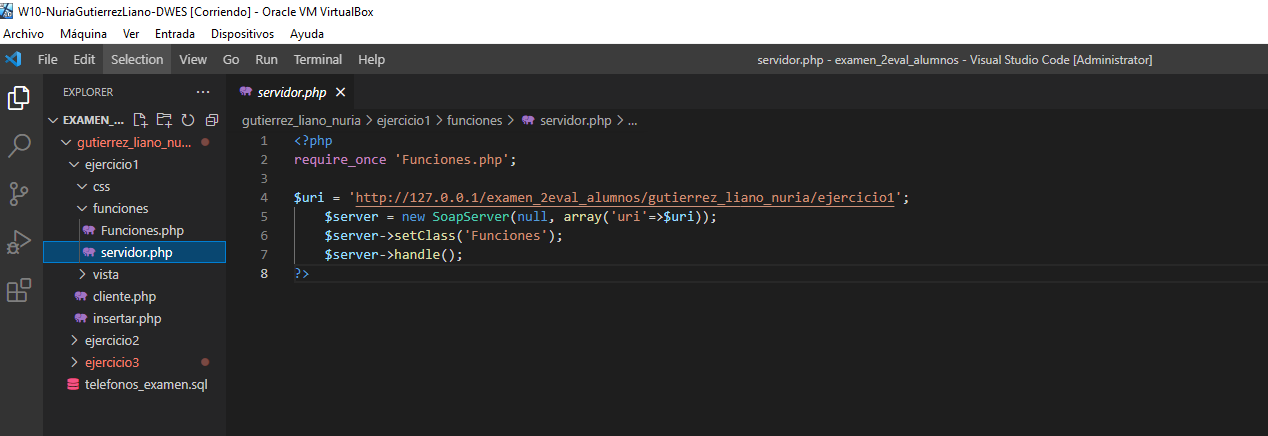
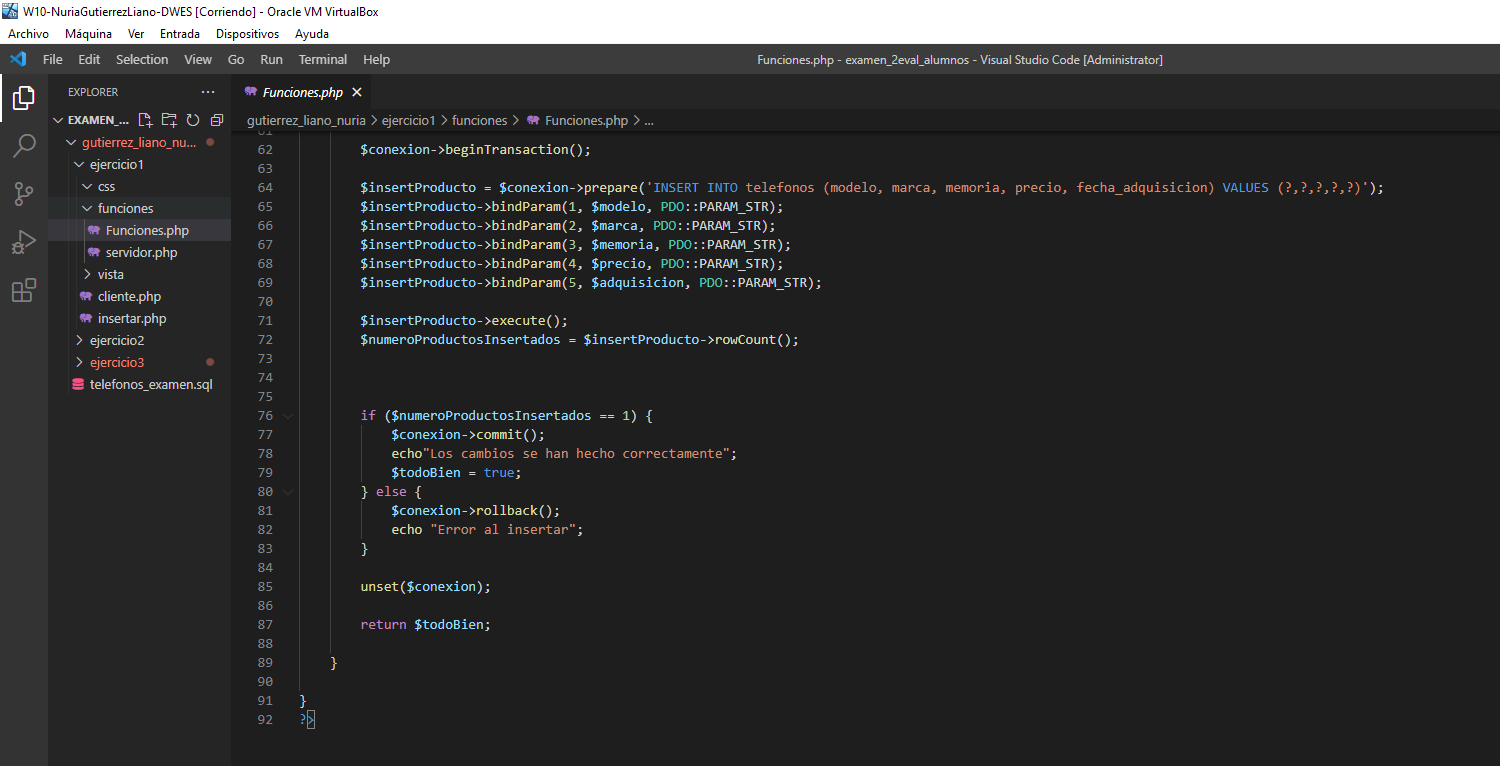
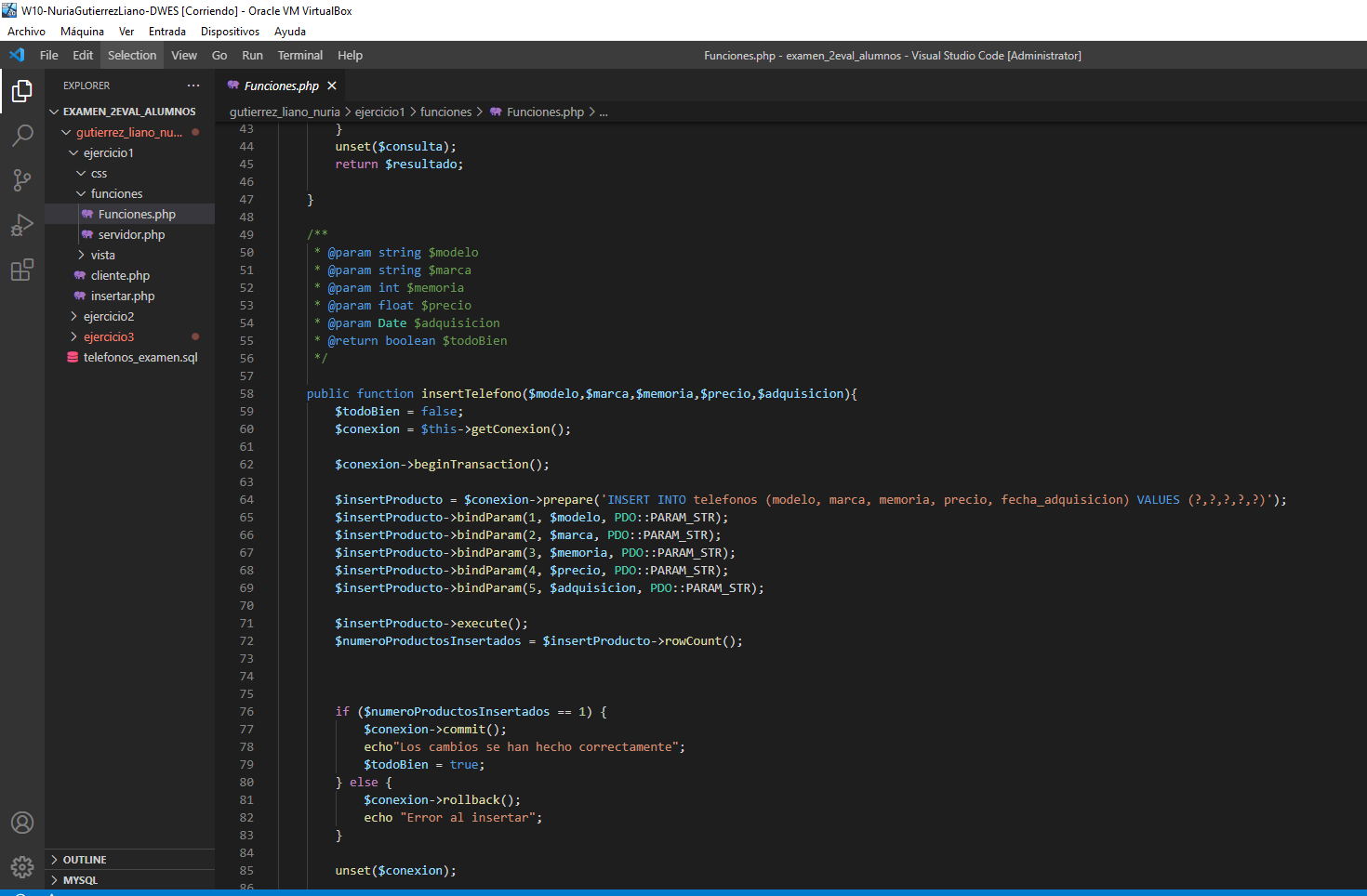
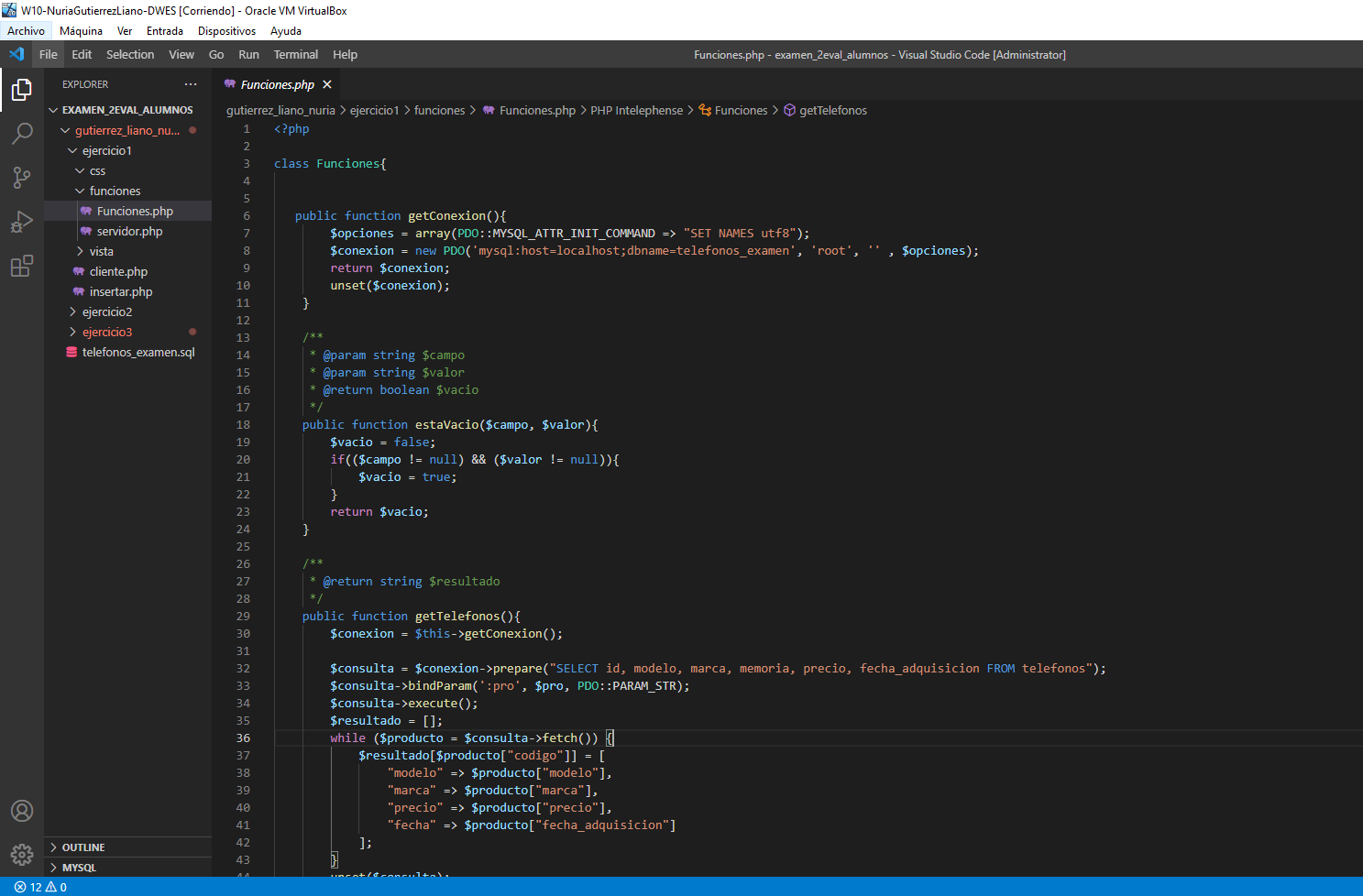
**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

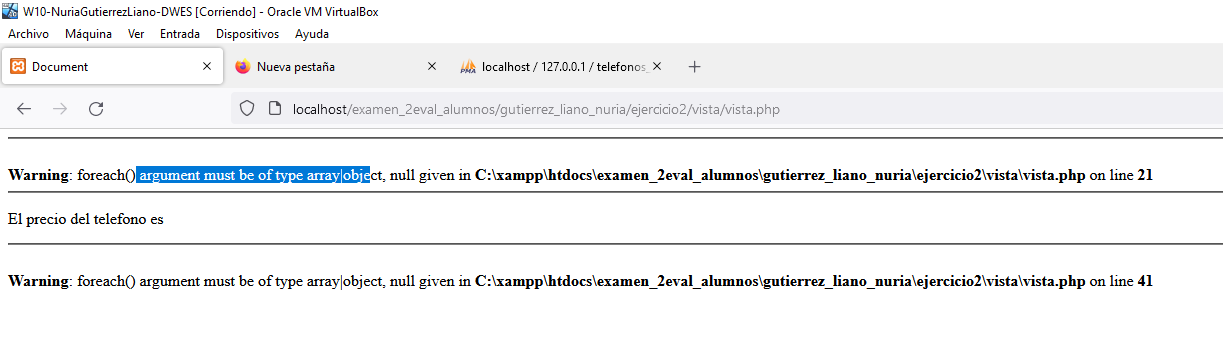
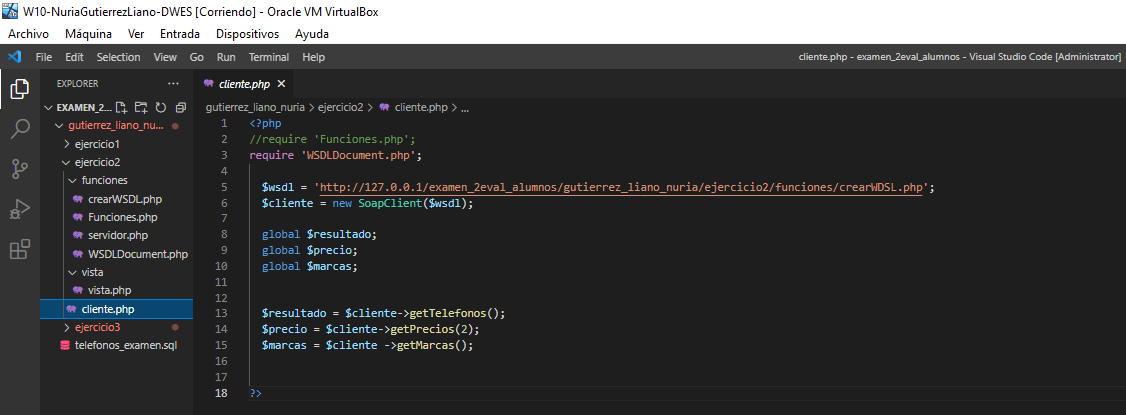
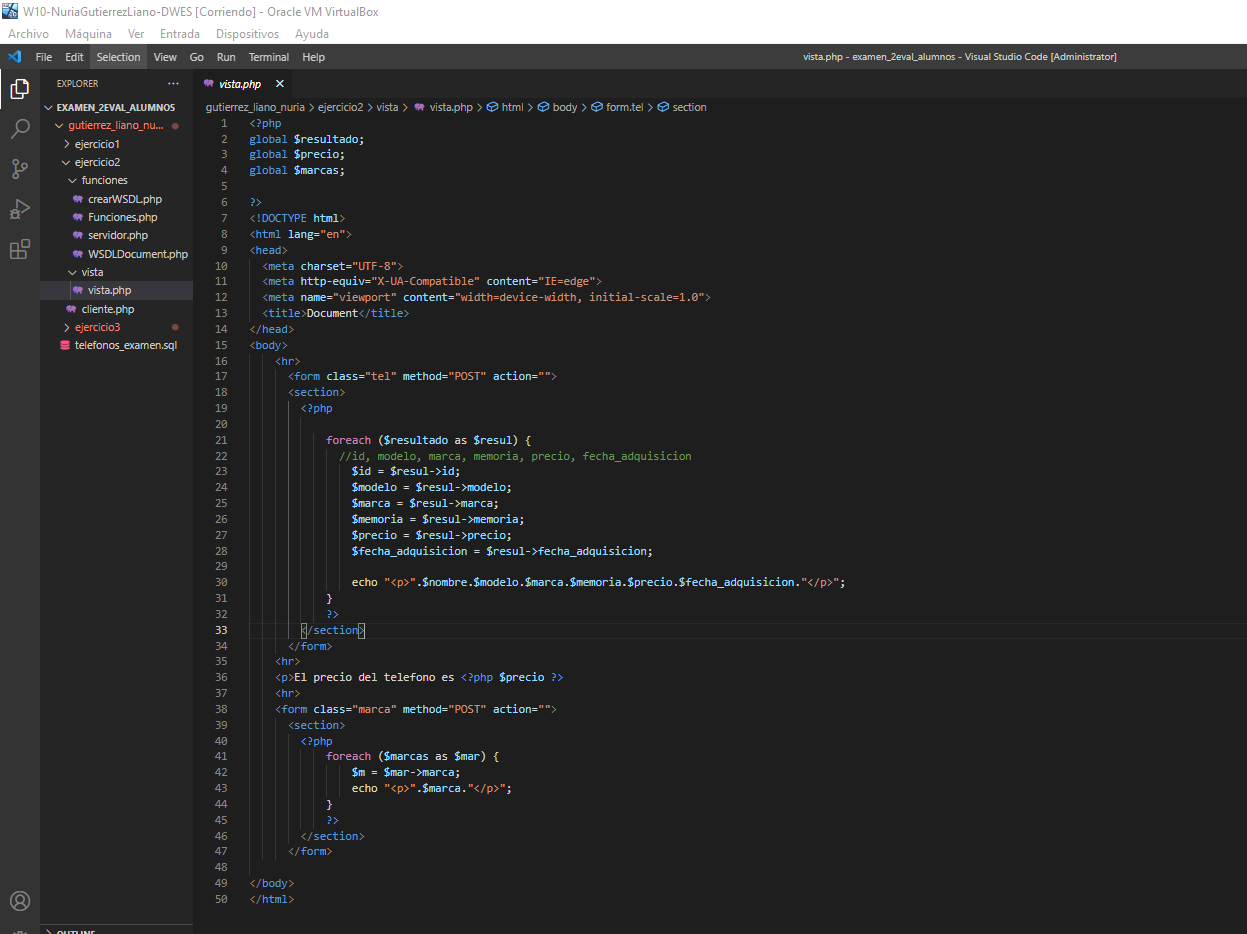
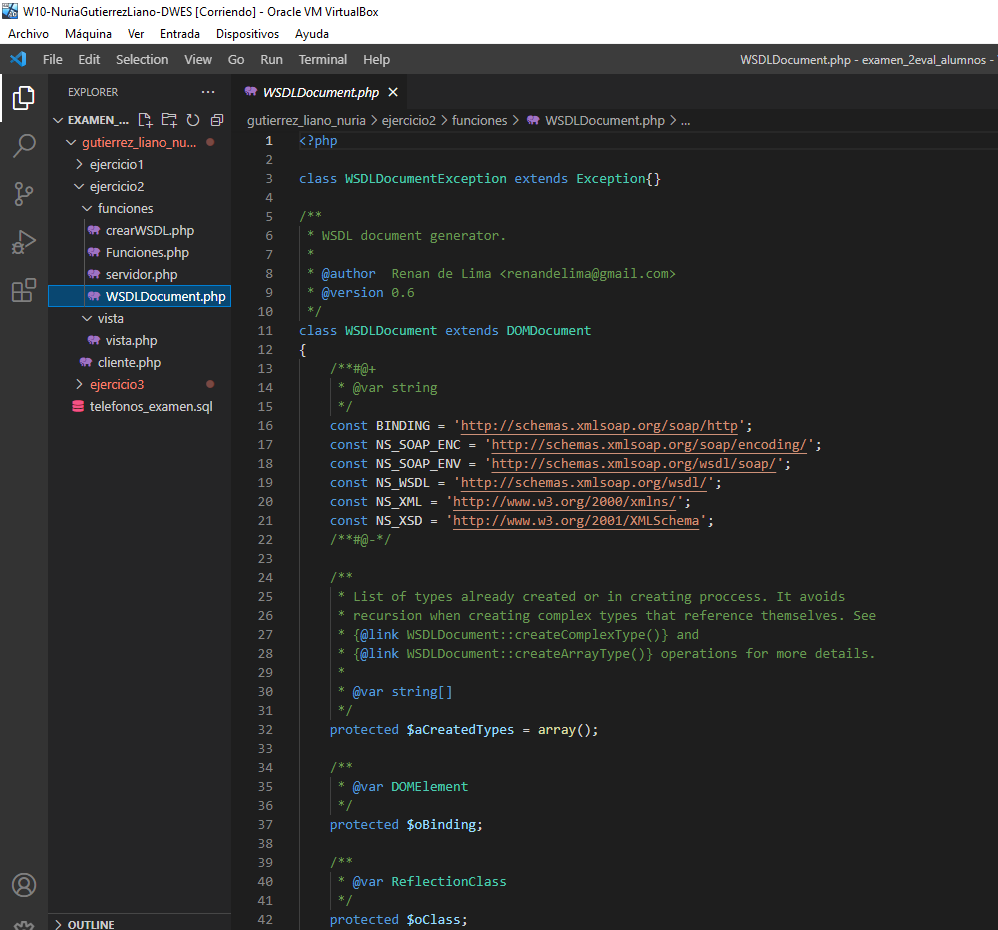
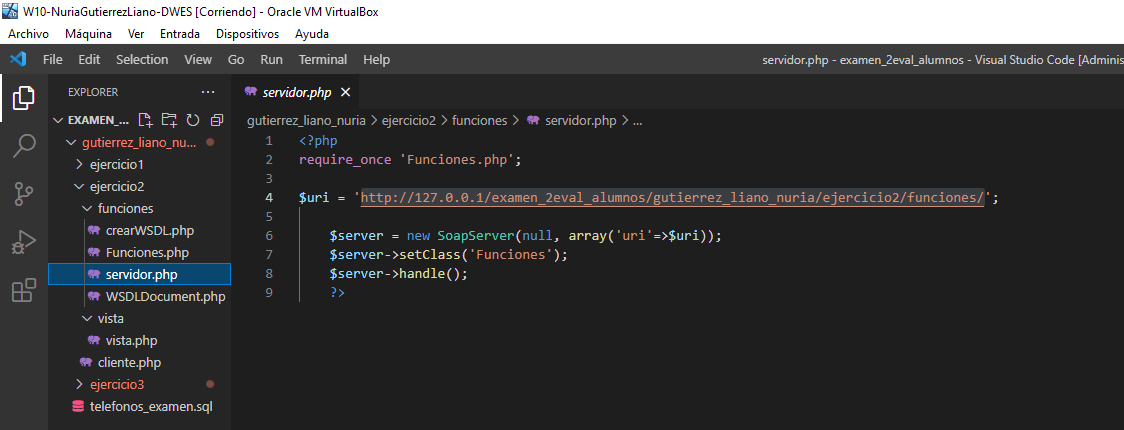
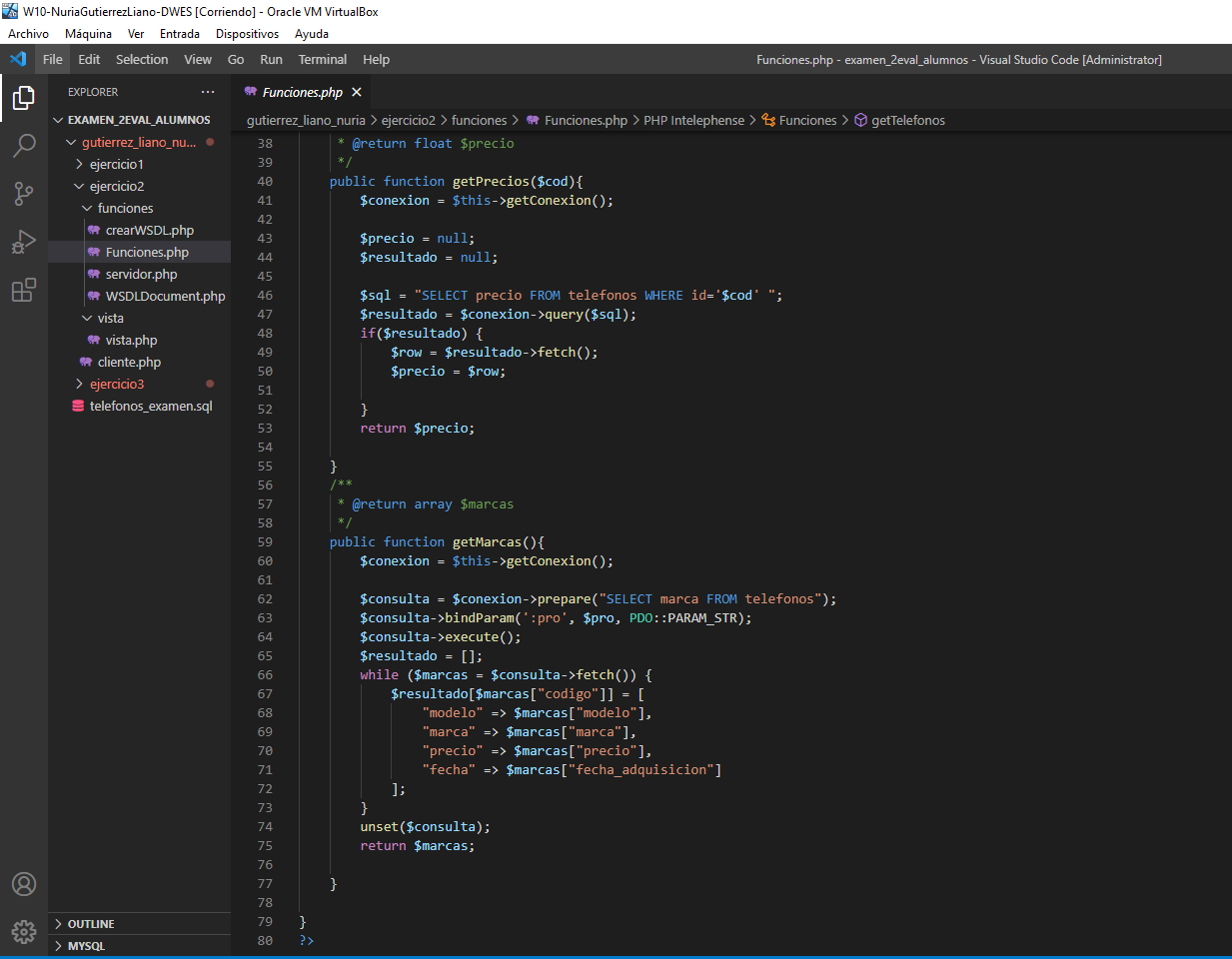
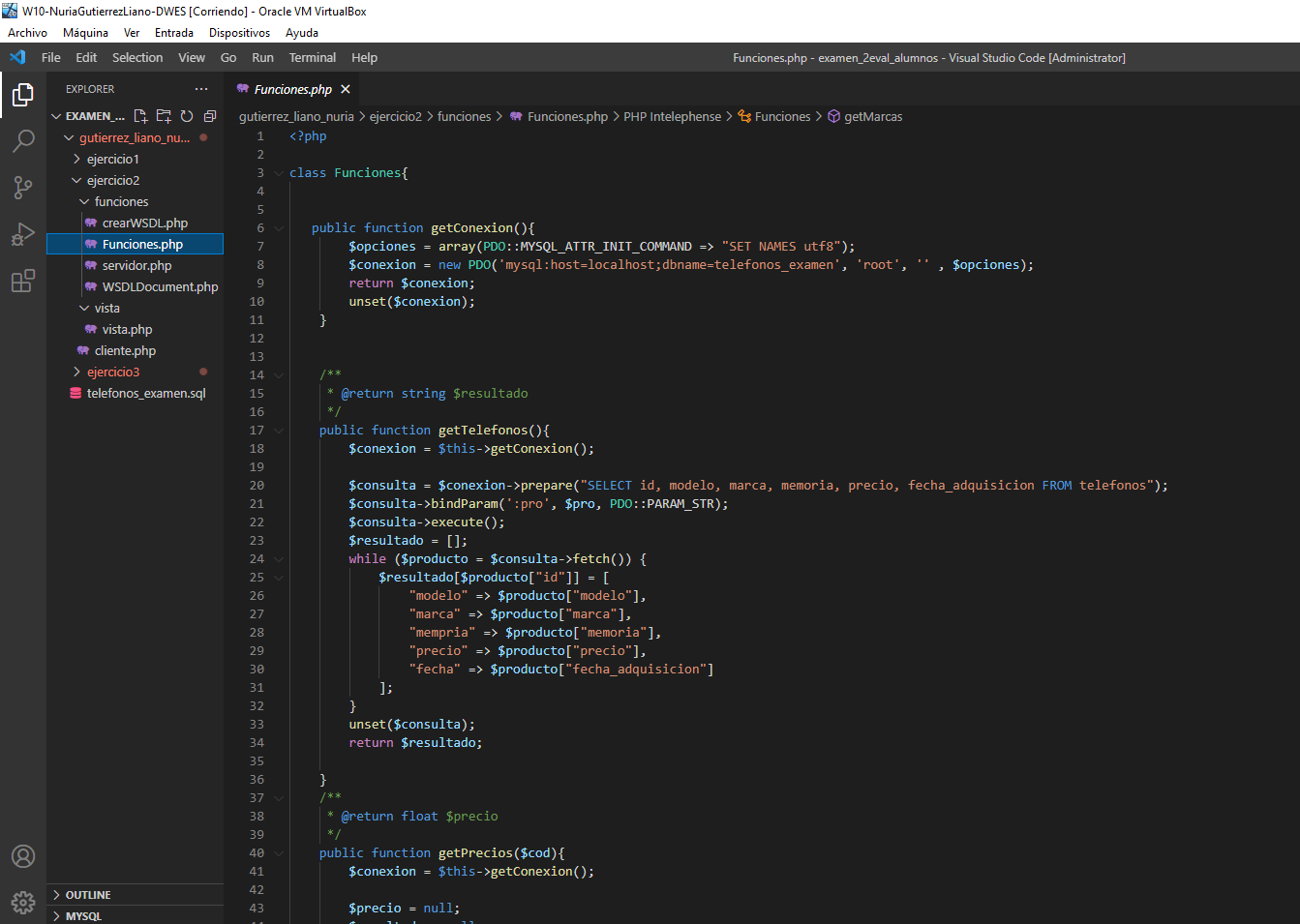
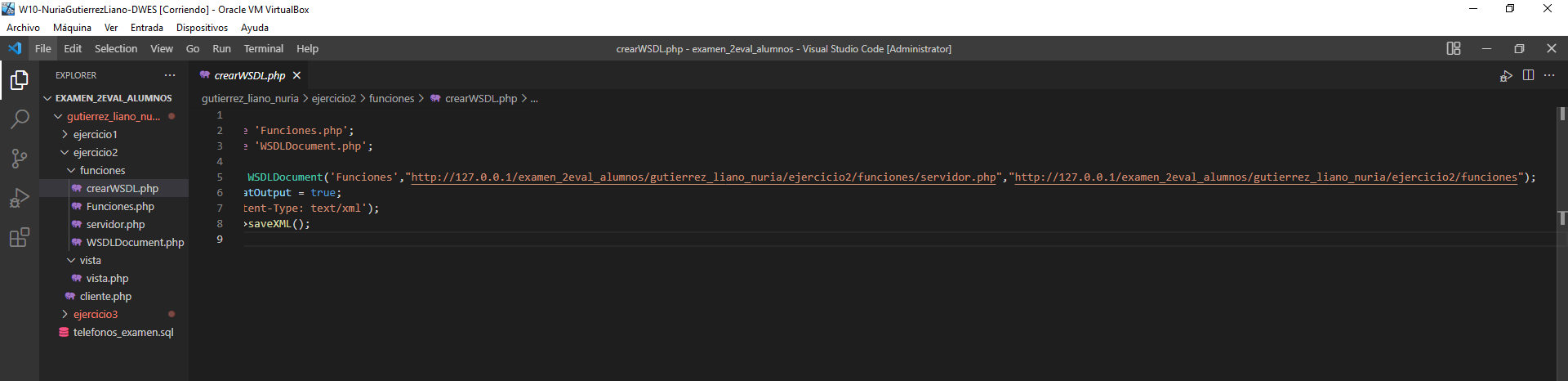
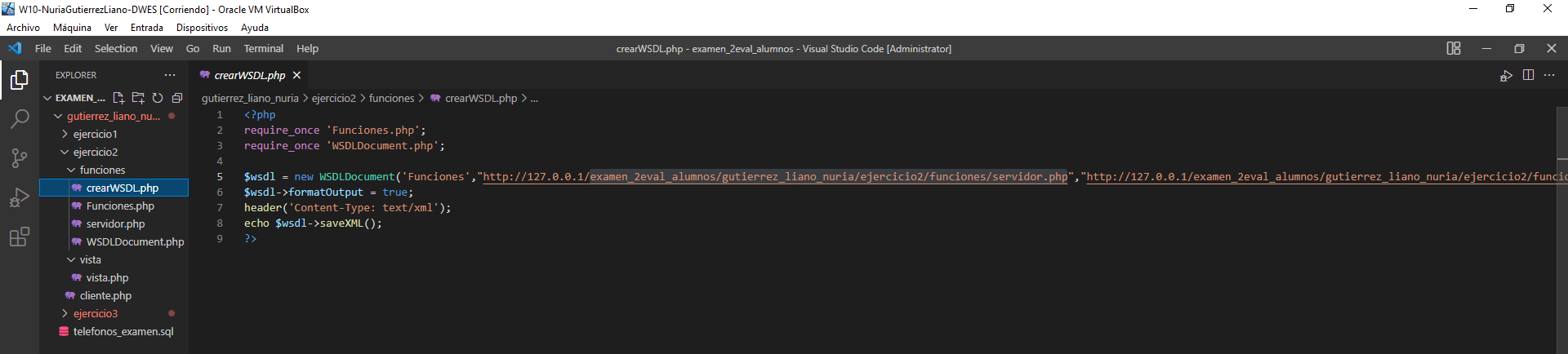
Descripción generada automáticamente**

**Ejercicio1**

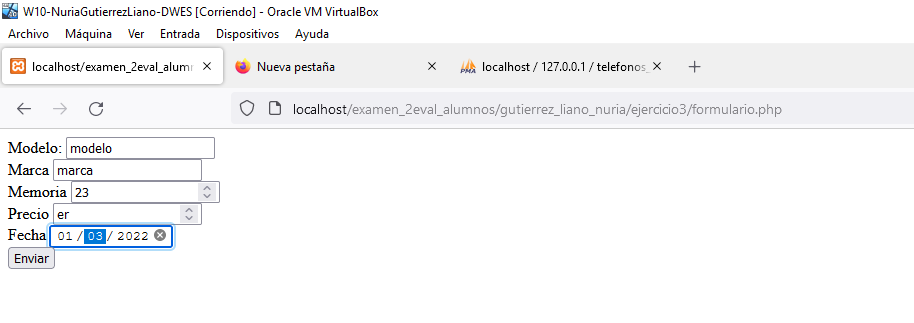
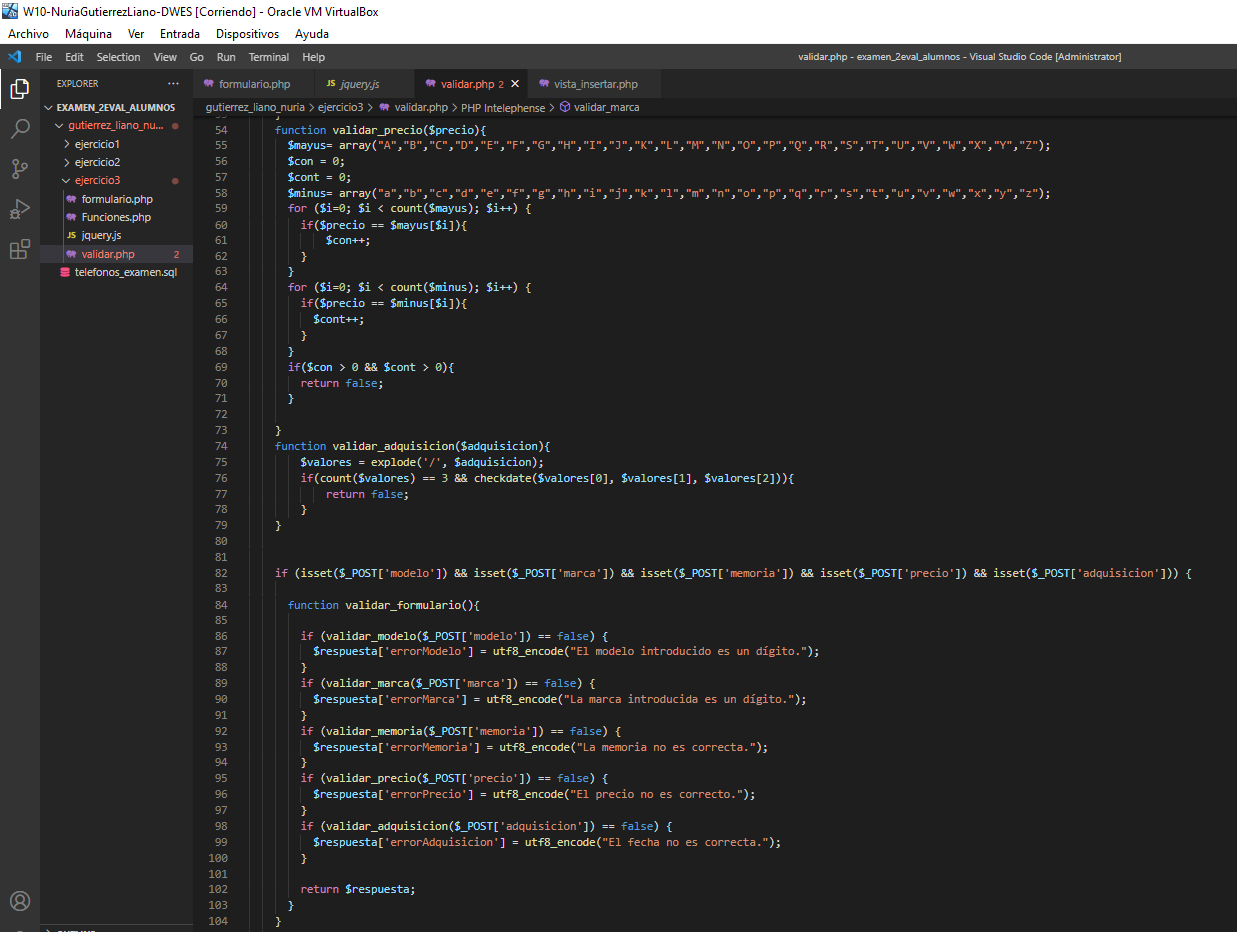
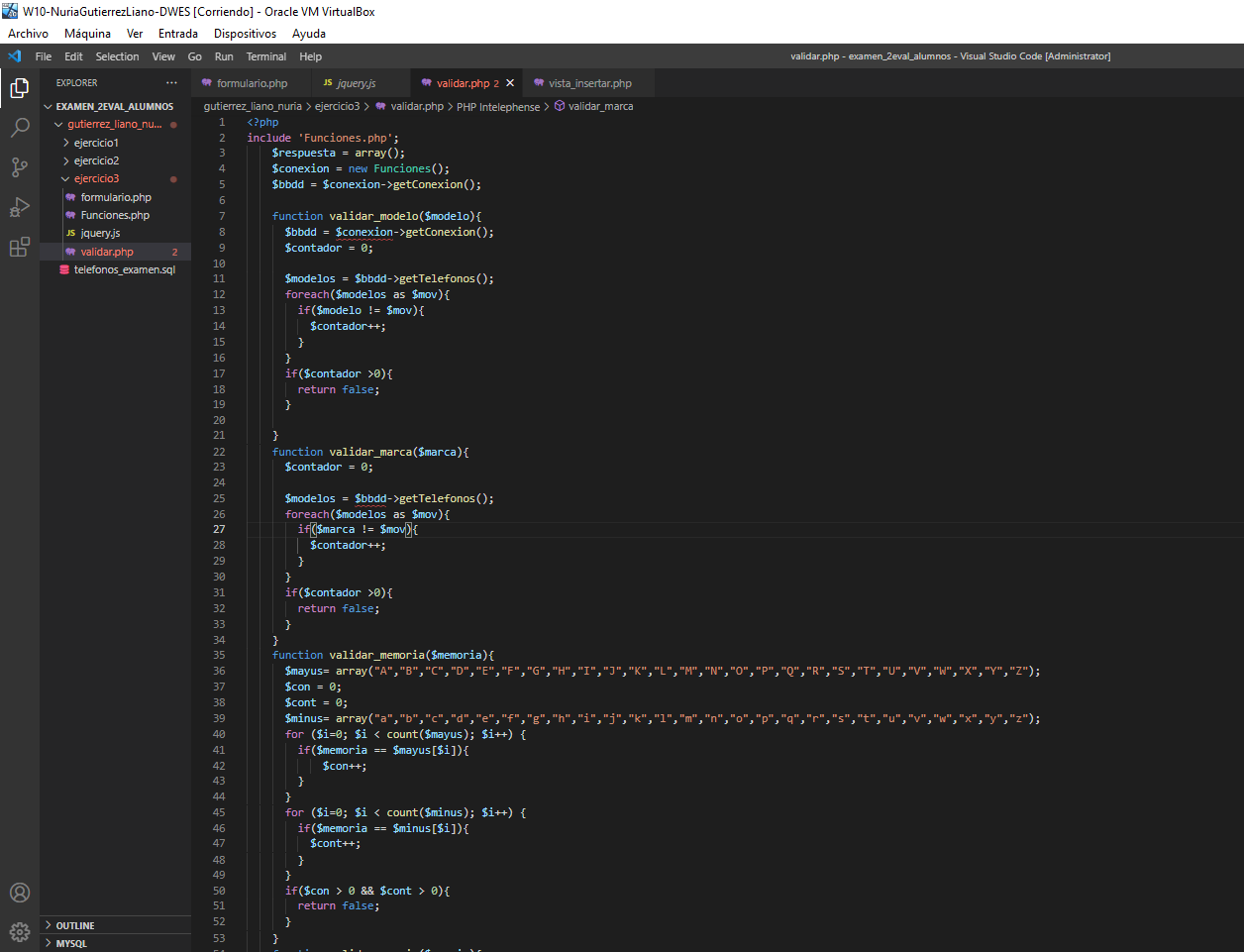
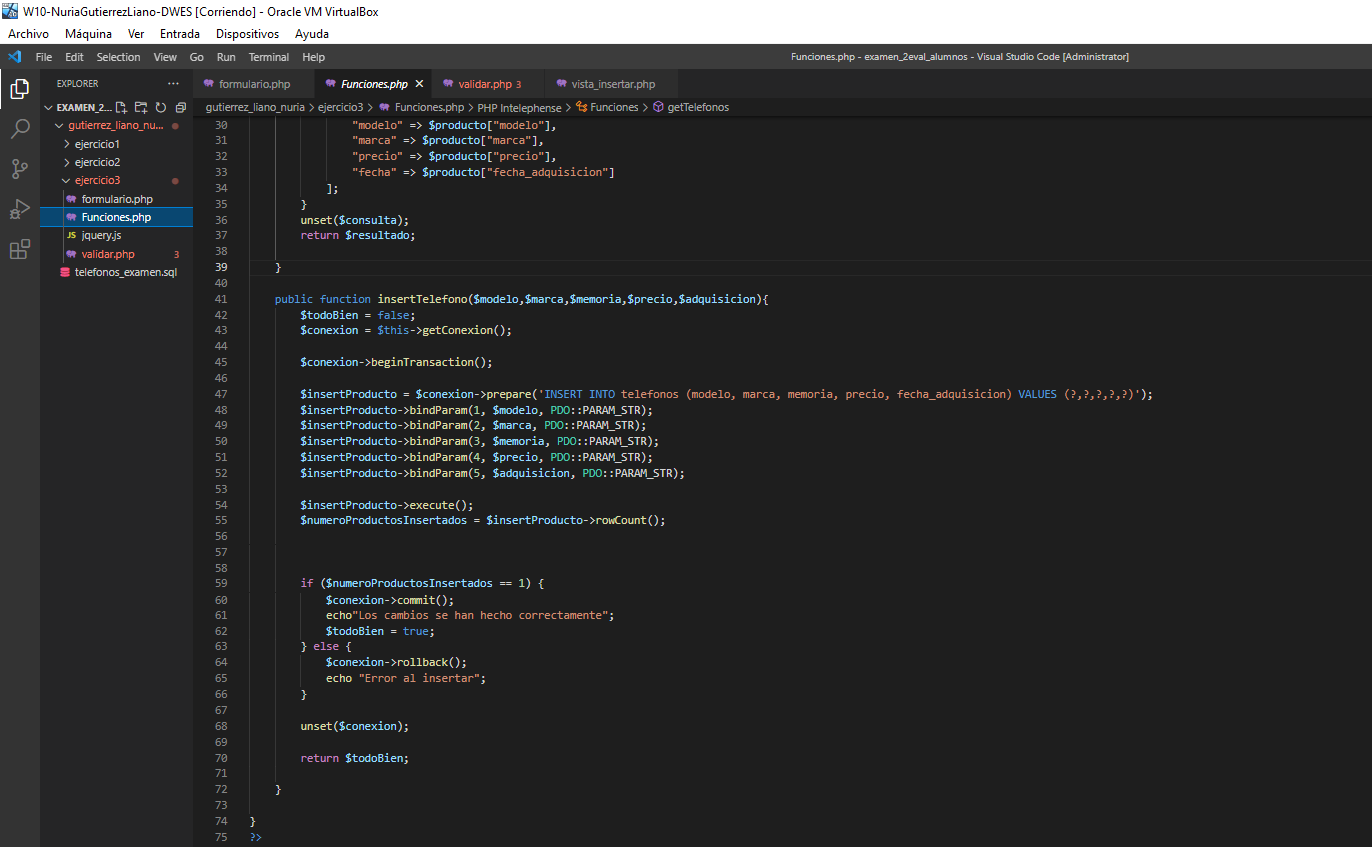
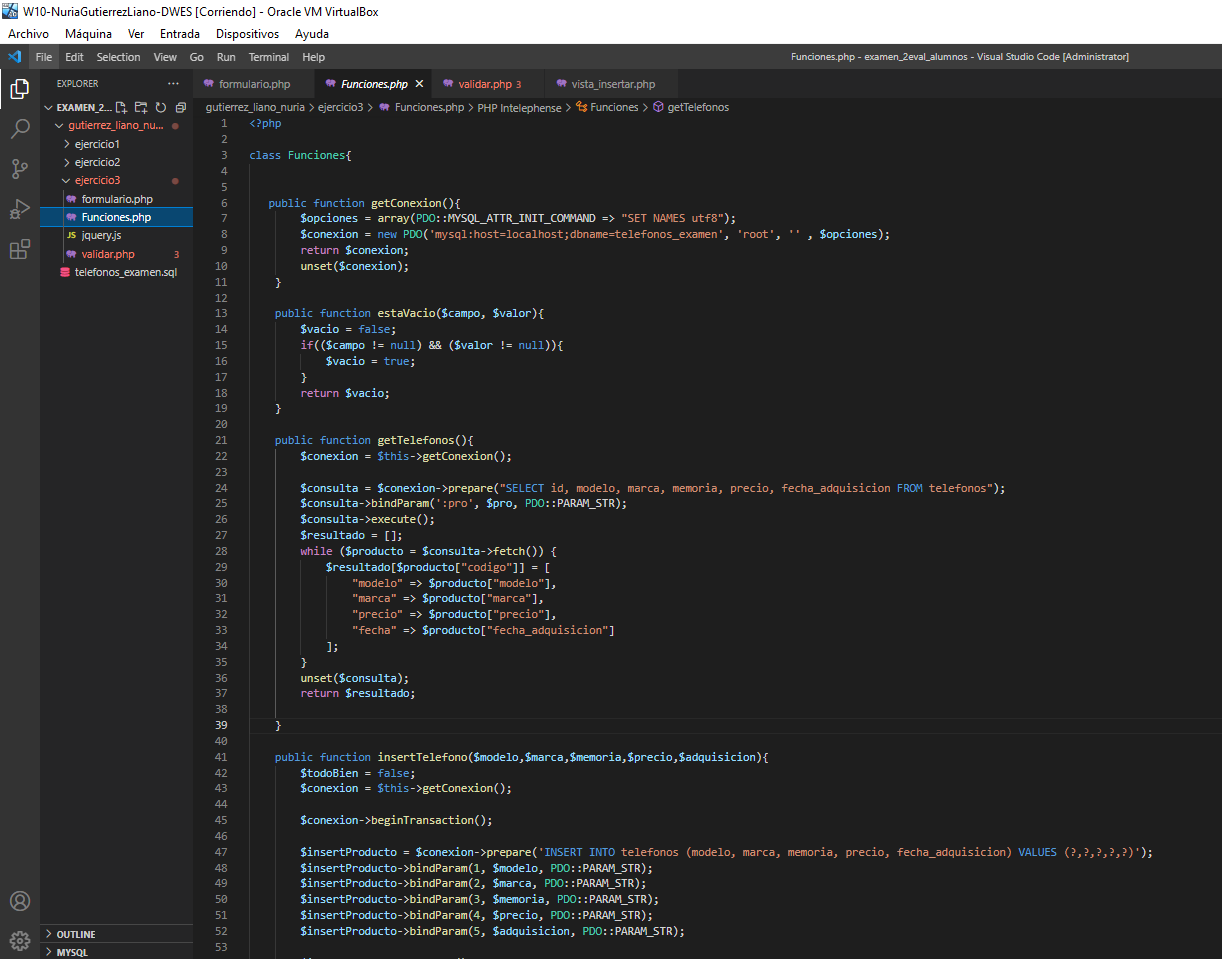
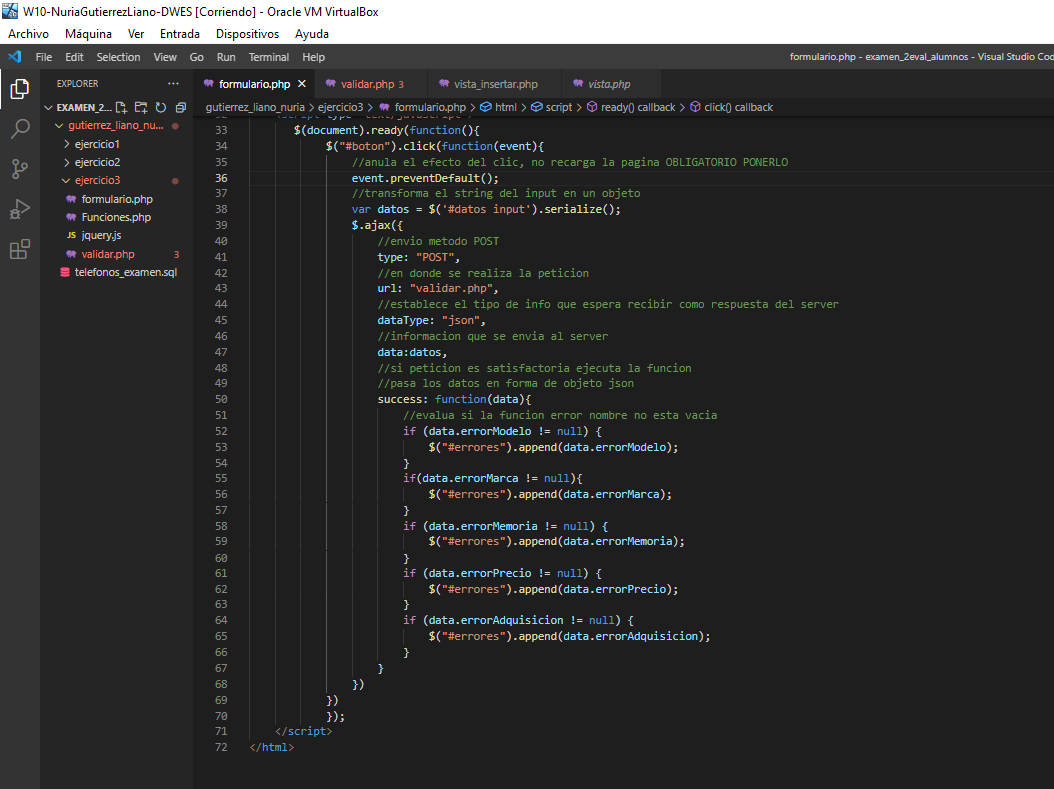
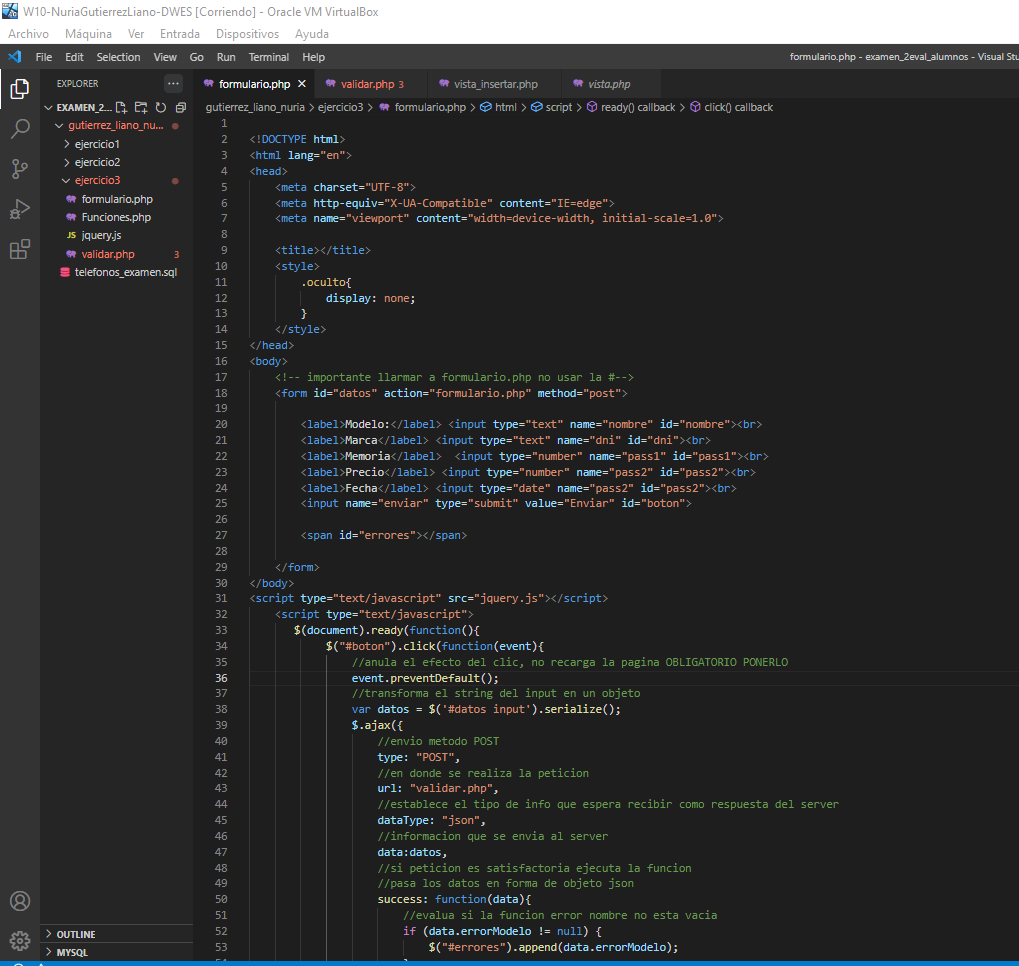
****

**No lo puedo ejecutar porque el botón no lleva a nigún sitio**

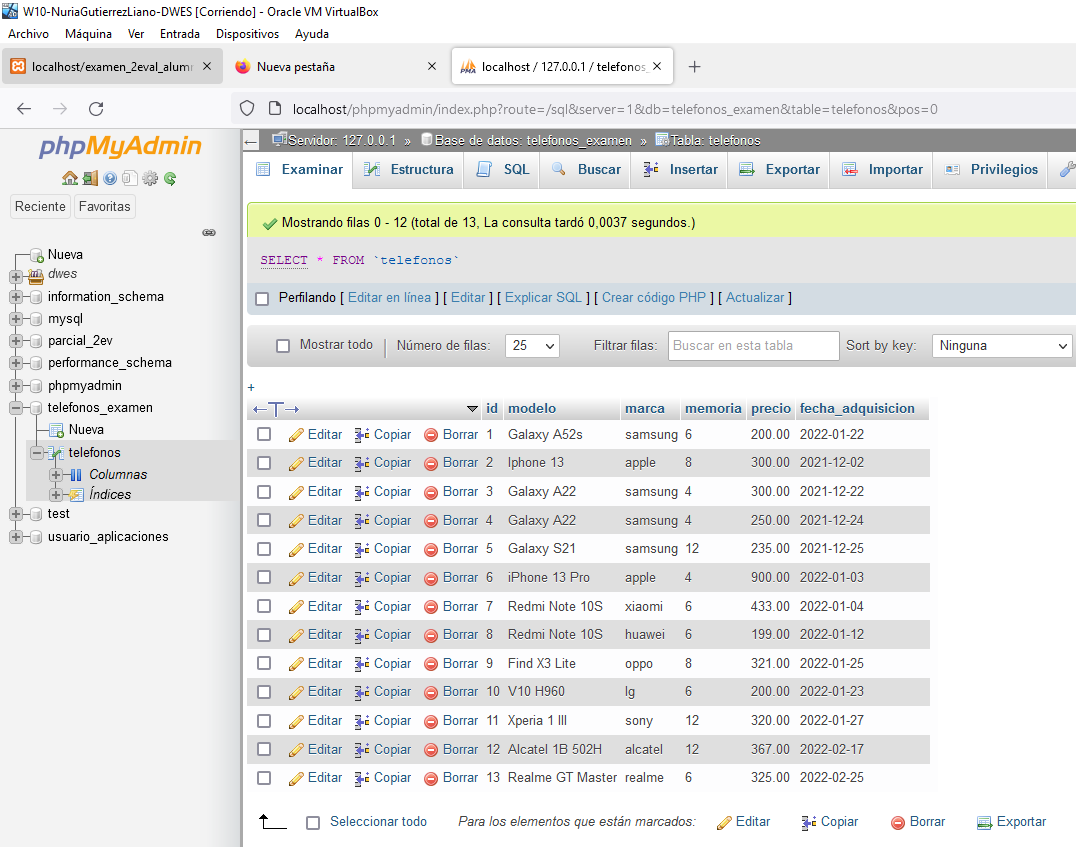
**Ejercicio2**

****

**Ejercicio3**

****

**Base de Datos importada**

****