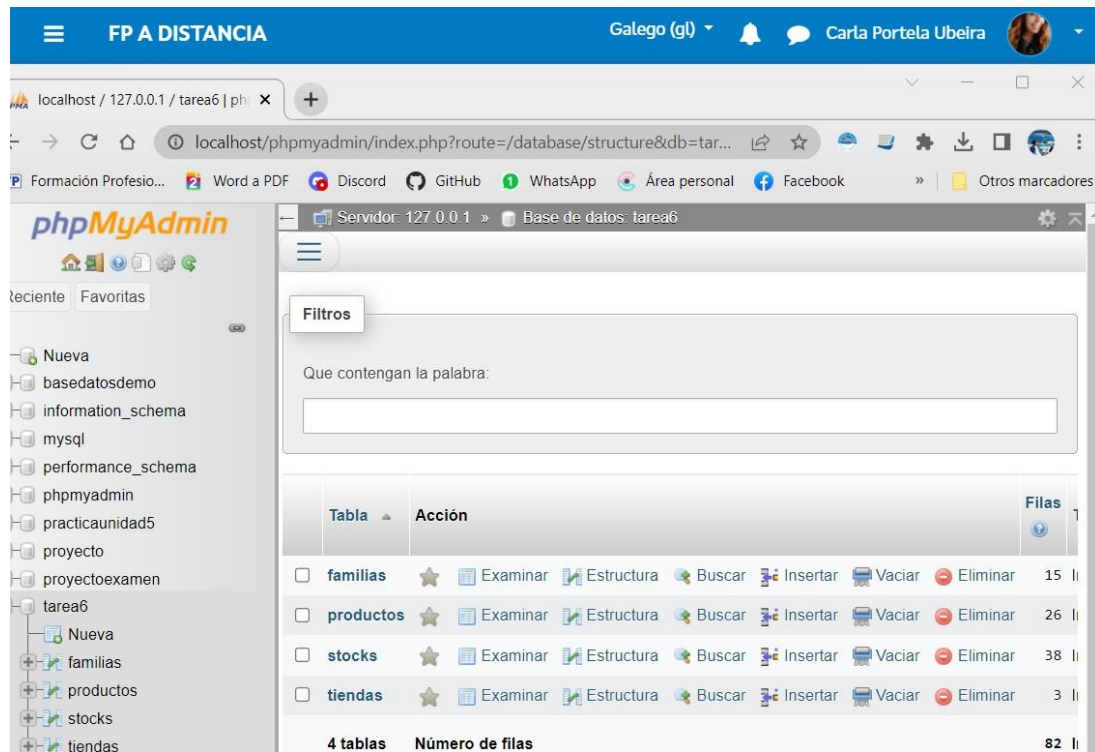


TAREA DWES 06

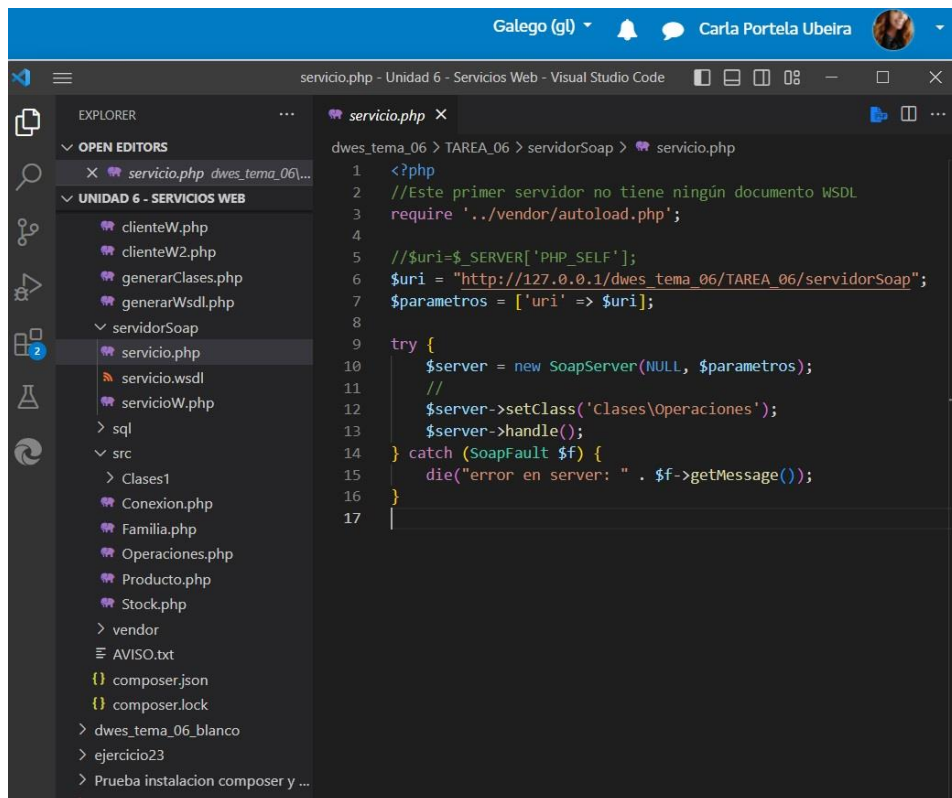
1 - Crear una base de datos de nombre "tarea6", similar a la que usamos en los ejercicios anteriores. Para acceder puedes reutilizar el usuario "gestor" y su contraseña "secreto".

Para ello, descargamos los archivos .sql proporcionados, creamos una carpeta en tarea6 llamada "sql" donde los guardamos. Una vez guardados, accedemos con el Apache y MySQL arrancados, a <http://localhost/phpmyadmin> e importamos los dos archivos .sql que nos crearán la base de datos.

2 - A



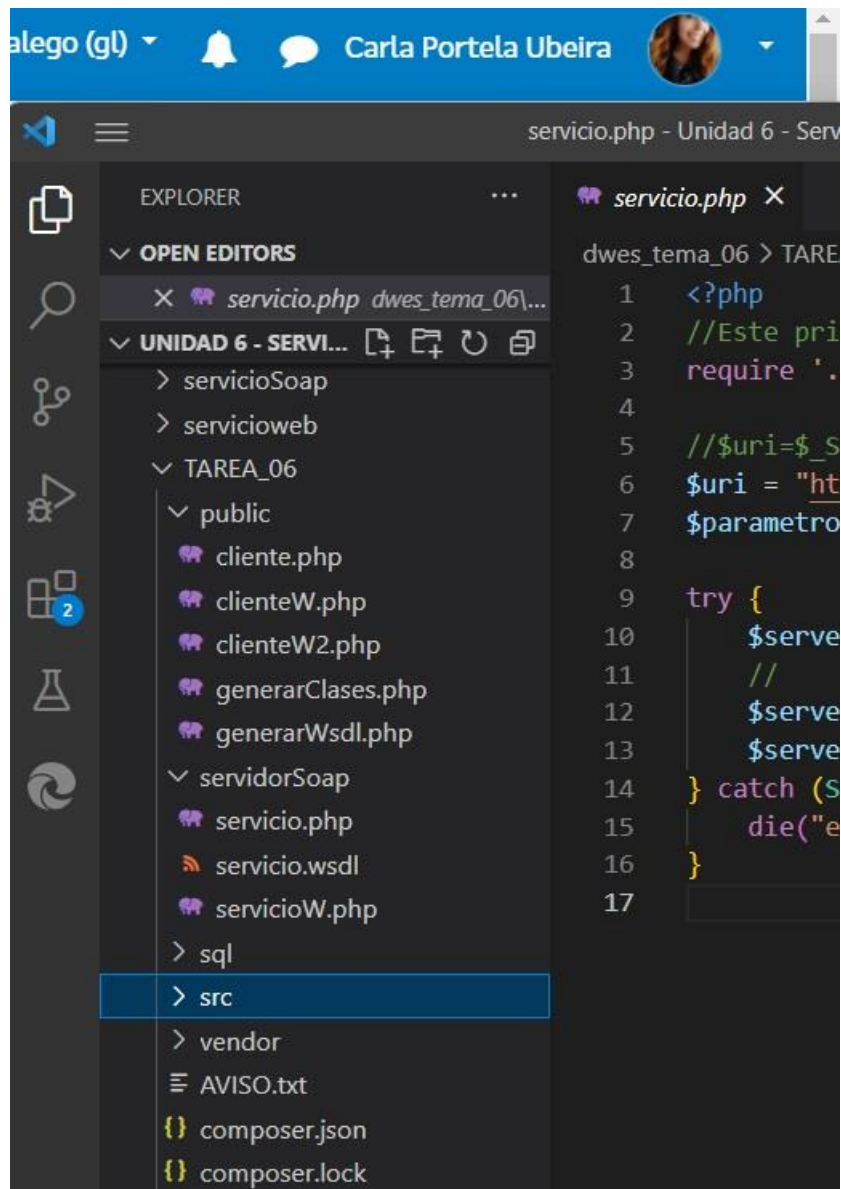
continuación utilizarás "PHP SOAP" para crear un servicio web con cuatro funciones que expongan información de la base de datos de la tienda online. En el primer servidor no deberás utilizaremos ningún archivo WSDL. Después generaremos uno y lo usaremos.



```
1 <?php
2 //Este primer servidor no tiene ningún documento WSDL
3 require '../vendor/autoload.php';
4
5 //$uri=$ _SERVER['PHP_SELF'];
6 $uri = "http://127.0.0.1/dwes_tema_06/TAREA_06/servidorSoap";
7 $parametros = ['uri' => $uri];
8
9 try {
10     $server = new SoapServer(NULL, $parametros);
11     //
12     $server->setClass('Clases\Operaciones');
13     $server->handle();
14 } catch (SoapFault $f) {
15     die("error en server: " . $f->getMessage());
16 }
17
```

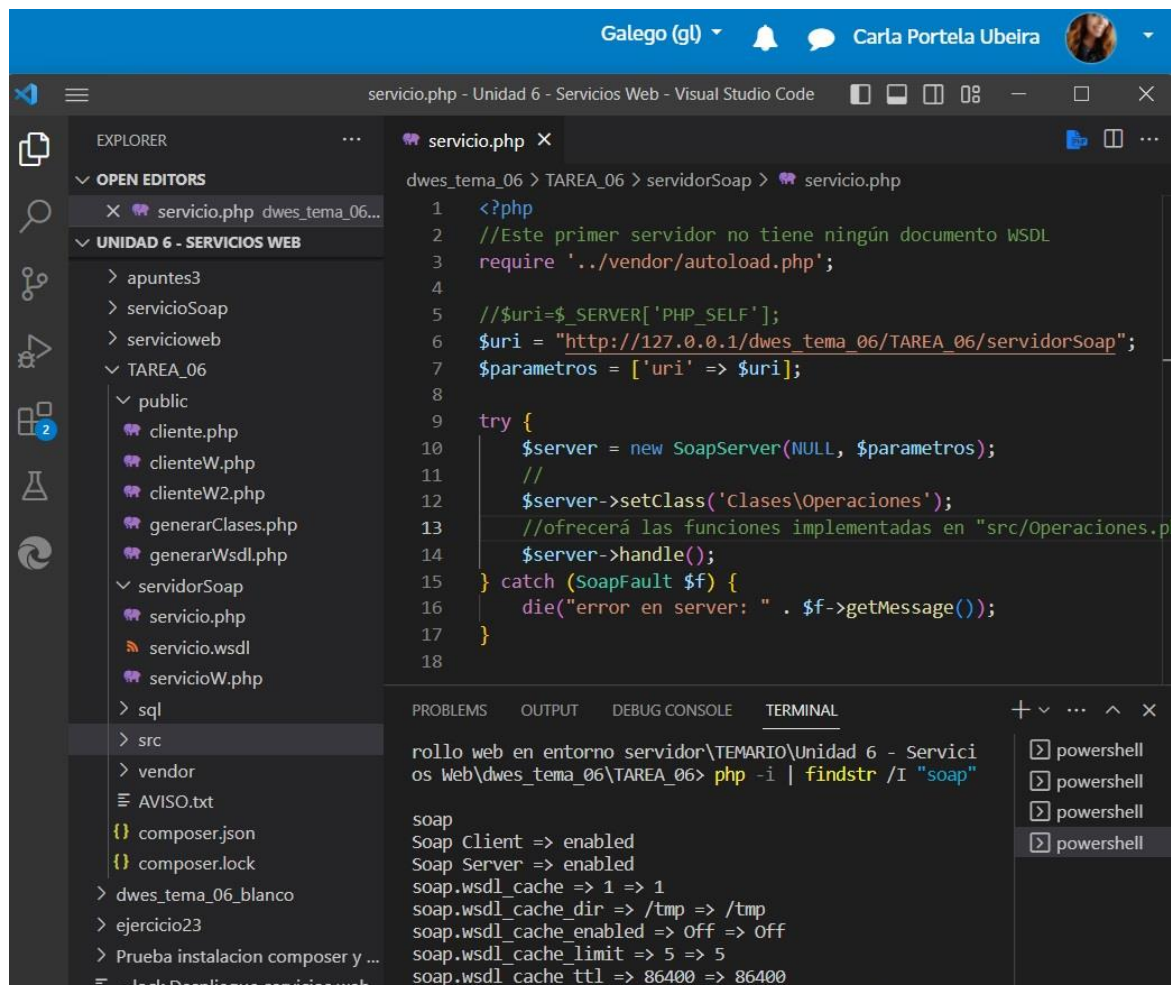
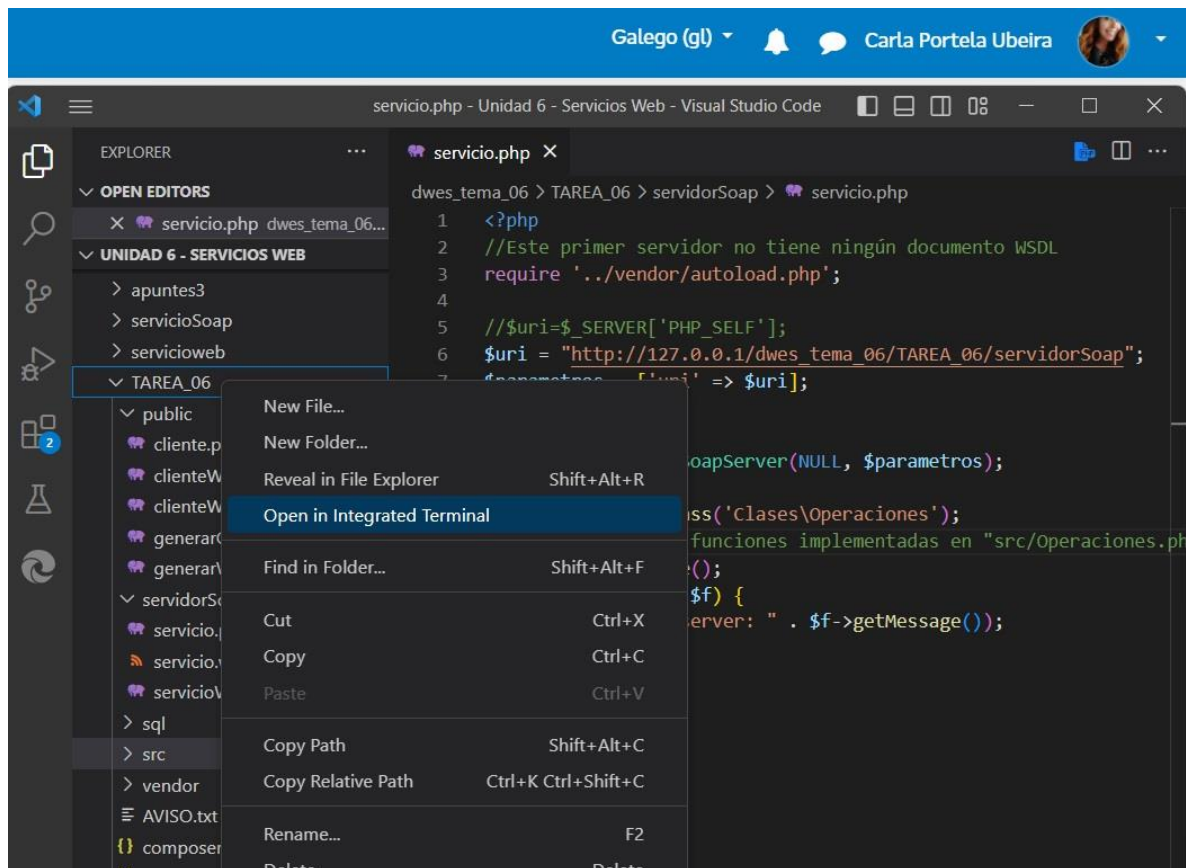
Tendremos las carpetas:

- "public": donde crearemos los ficheros para llamar a los servicios que crearemos.
- "servidorSoap": donde generaremos nuestros servicios.
- "src": Guardaremos todas nuestras clases, la clase "Operaciones" con todas las funciones que nos ofrecerá, la clase "Conexion", y las clases "Producto, Familia y Stock", con los métodos para acceder a la bases de datos y gestionar las tablas, como vimos en unidades anteriores.
- "vendor": La generará automáticamente Composer.



Para esta práctica se pedirá "autoload" de las clases, "php2wsdl" y "wsdl2phpgenerator, todo instalado con "Composer"

Para ello, en la carpeta raíz "TAREA-06" abrimos el terminal integrado y tecleamos `php -i | findstr /I "soap"` (para cerciorarnos de que la extensión soap esté activada y el caché de soap esté deshabilitado).

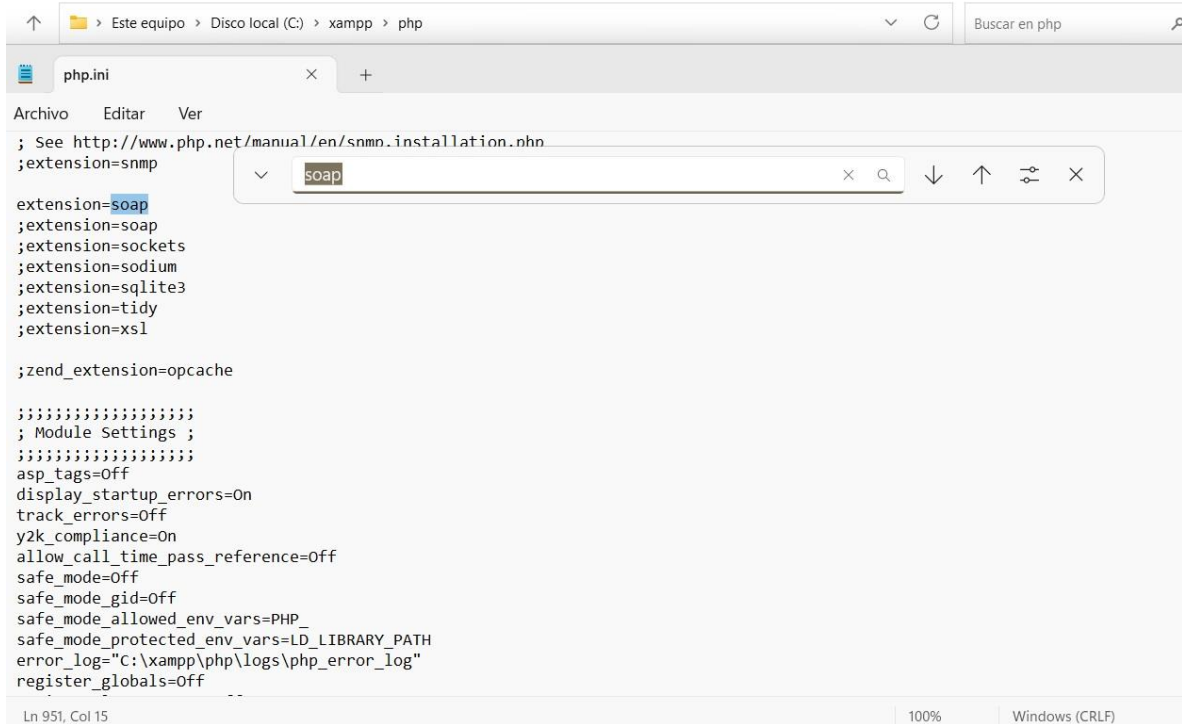


Si
no
es
así,
en

xampp/php/ modificamos el archivo **php.ini** descomentando las lineas:

- ➔ **extension=soap**
- ➔ **soap.wsdl_cache_enabled=0** (Sino, los cambios que se realicen en los ficheros wsdl no tendrán efecto de forma inmediata).

Y paramos y arrancamos el Apache para que se hagan efectivos los cambios.



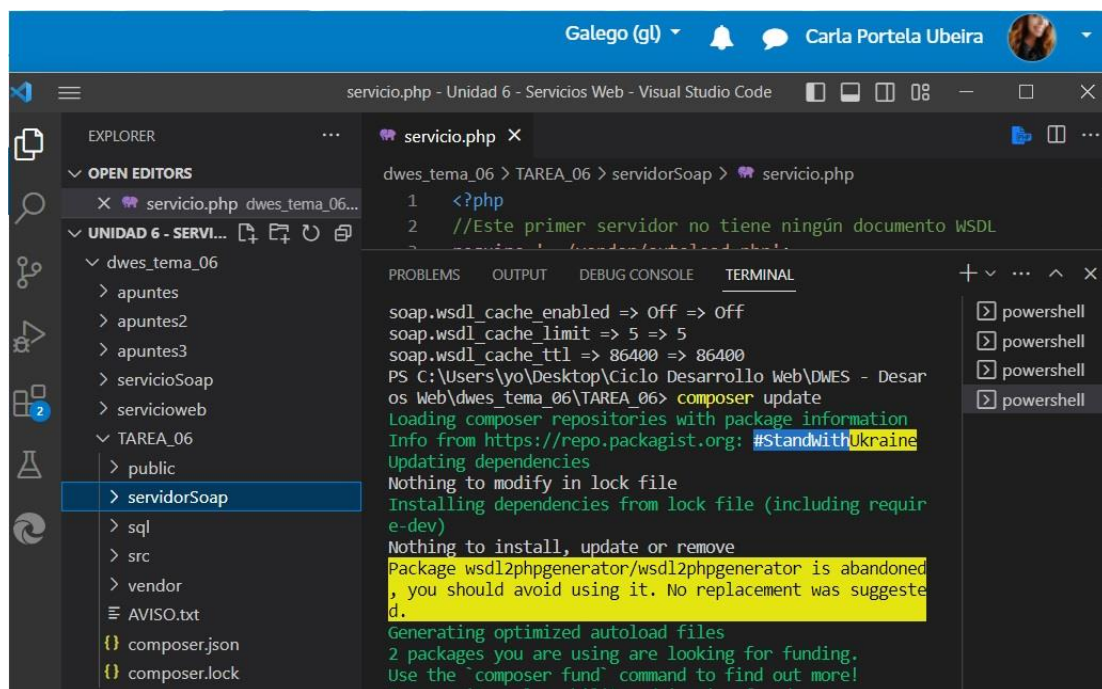
```
php.ini
; See http://www.php.net/manual/en/snmp.installation.php
;extension=snmp

extension=soap
;extension=soap
;extension=sockets
;extension=sodium
;extension=sqlite3
;extension=tidy
;extension=xsl

;zend_extension=opcache

; Module Settings ;
asp_tags=Off
display_startup_errors=On
track_errors=Off
y2k_compliance=On
allow_call_time_pass_reference=Off
safe_mode=Off
safe_mode_gid=Off
safe_mode_allowed_env_vars=PHP_
safe_mode_protected_env_vars=LD_LIBRARY_PATH
error_log="C:\xampp\php\logs\php_error_log"
register_globals=Off
```

Una vez realizado esto, volvemos a la terminal integrada y realizamos *composer update* en la carpeta raíz “TAREA_06

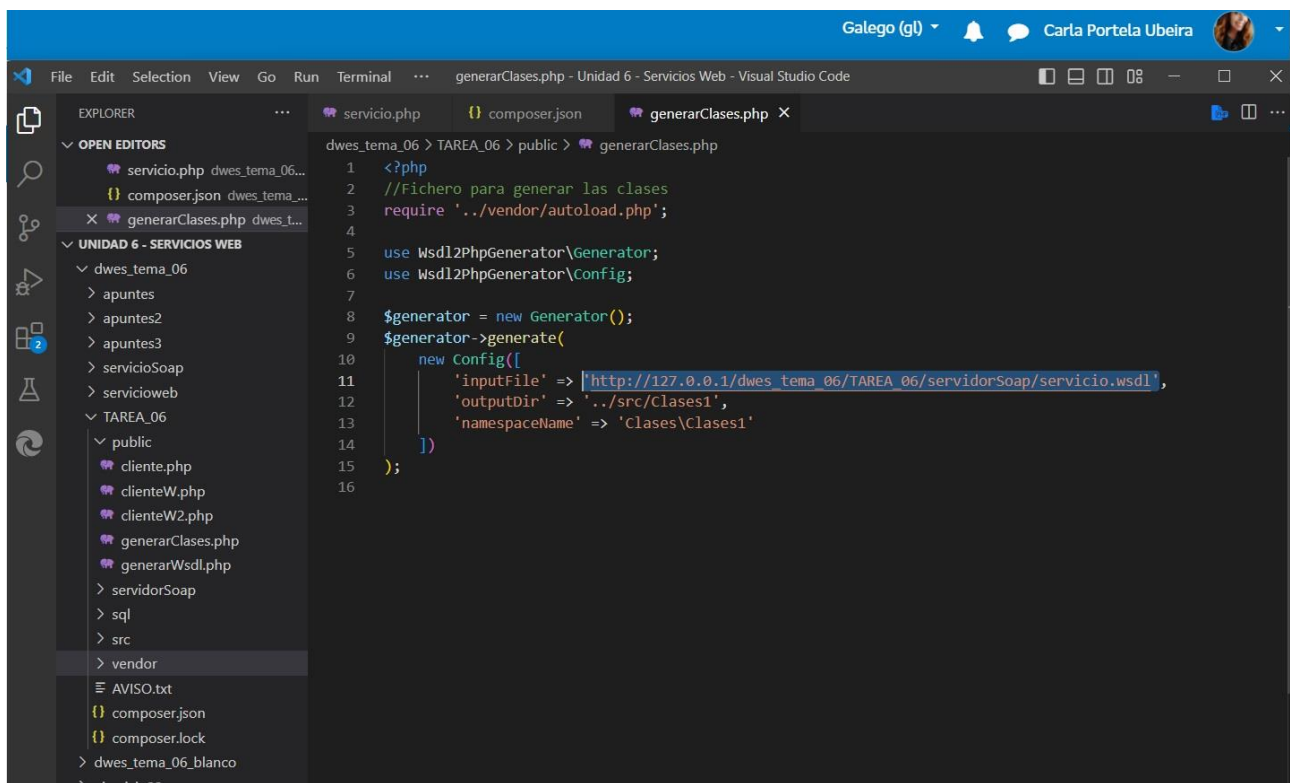


```
servicio.php - Unidad 6 - Servicios Web - Visual Studio Code
EXPLORER
OPEN EDITORS
servicio.php dwes_tema_06...
UNIDAD 6 - SERVI...
dwes_tema_06
  > apuntes
  > apuntes2
  > apuntes3
  > servicioSoap
  > servicioweb
  > TAREA_06
    > public
    > servidorSoap
    > sql
    > src
    > vendor
    > AVISO.txt
    {} composer.json
    {} composer.lock

servicio.php
1 <?php
2 //Este primer servidor no tiene ningún documento WSDL
3 require_once __DIR__.'/vendor/autoload.php';

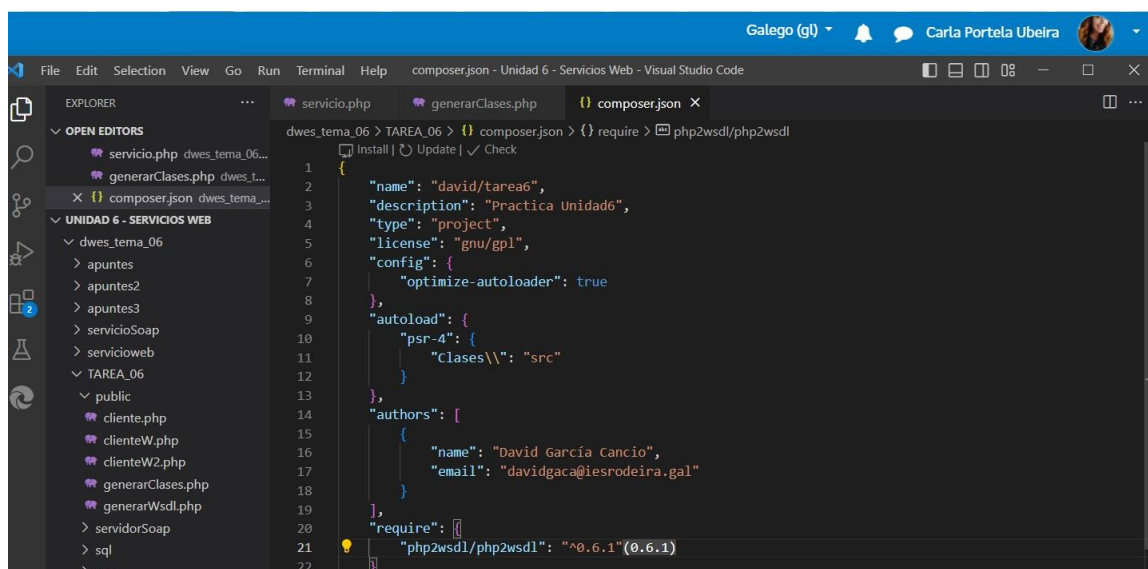
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
soap.wsdl_cache_enabled => Off => Off
soap.wsdl_cache_limit => 5 => 5
soap.wsdl_cache_ttl => 86400 => 86400
PS C:\Users\yo\Desktop\Ciclo Desarrollo Web\DWES - Desar...
os Web\dwes_tema_06\TAREA_06> composer update
Loading composer repositories with package information
Info from https://repo.packagist.org: #StandWithUkraine
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Package wsdl2phpgenerator/wsdl2phpgenerator is abandoned, you should avoid using it. No replacement was suggested.
Generating optimized autoload files
2 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
```


Al realizarlo, la primera vez, aparecía un error en el archivo **phpClass.php** en la carpeta **/vendor/wsd12phpgenerator/lib/**, en las líneas 156-158, que como no las necesitamos, las descomentamos. Y volvemos a realizar *composer update*



```
1 <?php
2 //Fichero para generar las clases
3 require '../vendor/autoload.php';
4
5 use Wsd12PhpGenerator\Generator;
6 use Wsd12PhpGenerator\Config;
7
8 $generator = new Generator();
9 $generator->generate(
10     new Config([
11         'inputfile' => 'http://127.0.0.1/dwes_tema_06/TAREA_06/servidorSoap/servicio.wsdl',
12         'outputDir' => '../src/Clases1',
13         'namespaceName' => 'Clases\Clases1'
14     ])
15 );
16
```

Comprobamos el **composer.json** generado (que tenga el autoload de las clases y las librerías solicitadas):



```
1 {
2     "name": "david/tarea6",
3     "description": "Practica Unidad6",
4     "type": "project",
5     "license": "gnu/gpl",
6     "config": {
7         "optimize-autoloader": true
8     },
9     "autoload": {
10         "psr-4": {
11             "Clases\\": "src"
12         }
13     },
14     "authors": [
15         {
16             "name": "David García Cancio",
17             "email": "davidgaca@iesrodeira.gal"
18         }
19     ],
20     "require": {
21         "php2wsdl/php2wsdl": "^0.6.1"(0.6.1)
22     }
23 }
```

Como no se encuentra la librería de wsd12phpgenerator la instalamos una vez generado el archivo: **composer.json** con el comando *composer require wsd12phpgenerator/wsd12phpgenerator*

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure with folders like 'public', 'src', and 'vendor'. The 'composer.json' file is open in the editor, showing the 'require' section with the following content:

```
{
  "require": {
    "php2wsdl/php2wsdl": "^0.6.1"(0.6.1),
    "wsdl2phpgenerator/wsdl2phpgenerator": "^3.4"(3.4.0)
  }
}
```

The terminal at the bottom shows the output of the command `composer require wsdl2phpgenerator/wsdl2phpgenerator`. The output indicates that the package has been updated and dependencies are being installed. A warning message is displayed: `Package wsdl2phpgenerator/wsdl2phpgenerator is abandoned, you should avoid using it. No replacement was suggested.`

Realizamos *composer update* y verificamos que está todo bien configurado.

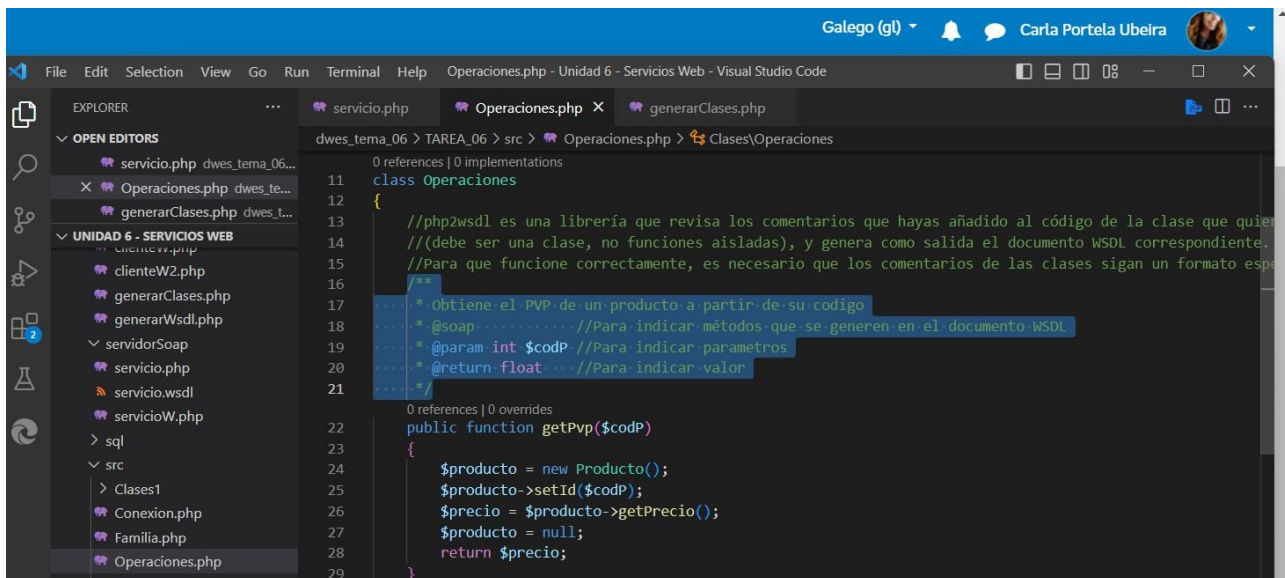
3 - Crearemos el servidor SOAP "servicio.php" en la carpeta "servidorSoap". Este servicio no tendrá asociado ningún archivo WSDL y ofrecerá las funciones siguientes (Todas implementadas en "src/Operaciones.php"):

- **getPVP**. Esta función recibirá como parámetro el código de un producto, y devolverá el PVP correspondiente al mismo.
- **getStock**. Esta función recibirá dos parámetros: el código de un producto y el código de una tienda. Devolverá el stock existente en dicha tienda del producto.
- **getFamilias**. No recibe parámetros y devuelve un array con los códigos de todas las familias existentes.
- **getProductosFamilia**. Recibe como parámetro el código de una familia y devuelve un array con los códigos de todos los productos de esa familia.

php2wsdl es una librería que revisa los comentarios que hayas añadido al código de la clase que queramos publicar (debe ser una clase, no funciones aisladas), y genera como salida el documento WSDL correspondiente. Para que funcione correctamente, es necesario que los comentarios de las clases sigan un formato específico: el mismo que utiliza la herramienta de documentación PHPDocumentor.

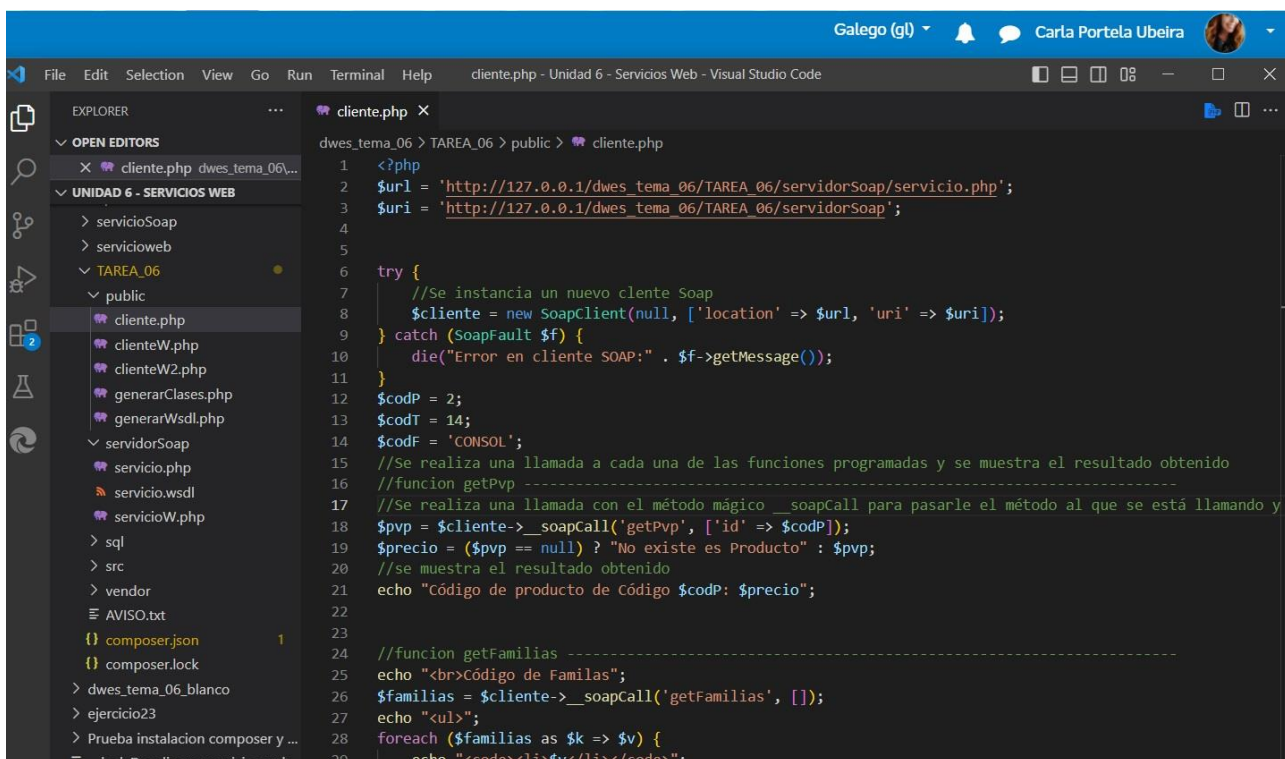
Por cada función:

- **@soap** Para indicar métodos que se generen en el documento WSDL
- **@param** Para indicar parámetros
- **@return** Para indicar valor



```
dwes_tema_06 > TAREA_06 > src > Operaciones.php > Clases\Operaciones
0 references | 0 implementations
class Operaciones
11 {
12     //php2wsdl es una libreria que revisa los comentarios que hayas a adido al c odigo de la clase que quieres
13     //((debe ser una clase, no funciones aisladas)), y genera como salida el documento WSDL correspondiente.
14     //Para que funcione correctamente, es necesario que los comentarios de las clases sigan un formato espe
15     /**
16      * Obtiene el PVP de un producto a partir de su codigo
17      * @soap ..... //Para indicar m todos que se generen en el documento WSDL
18      * @param int $codP //Para indicar parametros
19      * @return float ..... //Para indicar valor
20      */
21
22     0 references | 0 overrides
23     public function getPvp($codP)
24     {
25         $producto = new Producto();
26         $producto->setId($codP);
27         $precio = $producto->getPrecio();
28         $producto = null;
29         return $precio;
30     }
31 }
```

4 - Para comprobar la correcta ejecuci n del servicio, programa tambi n un cliente con nombre "cliente.php" en la carpeta "public" que realice una llamada a cada una de las funciones programadas y muestre el resultado obtenido.

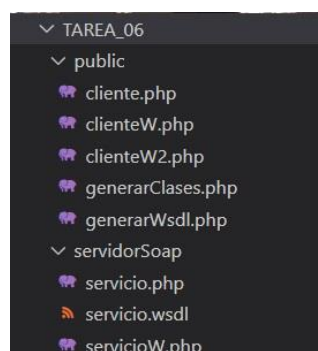


```
dwes_tema_06 > TAREA_06 > public > cliente.php
1 <?php
2 $url = 'http://127.0.0.1/dwes_tema_06/servidorSoap/servicio.php';
3 $uri = 'http://127.0.0.1/dwes_tema_06/servidorSoap';
4
5
6 try {
7     //Se instancia un nuevo cliente Soap
8     $cliente = new SoapClient(null, ['location' => $url, 'uri' => $uri]);
9 } catch (SoapFault $f) {
10     die("Error en cliente SOAP: " . $f->getMessage());
11 }
12 $codP = 2;
13 $codT = 14;
14 $codF = 'CONSOL';
15 //Se realiza una llamada a cada una de las funciones programadas y se muestra el resultado obtenido
16 //funcion getPvp -----
17 //Se realiza una llamada con el m todo m gico __soapCall para pasarle el m todo al que se est  llamando y
18 $pvp = $cliente->__soapCall('getPvp', ['id' => $codP]);
19 $precio = ($pvp == null) ? "No existe es Producto" : $pvp;
20 //se muestra el resultado obtenido
21 echo "C digo de producto de C digo $codP: $precio";
22
23
24 //funcion getFamilias -----
25 echo "<br>C digo de Familas";
26 $familias = $cliente->__soapCall('getFamilias', []);
27 echo "<ul>";
28 foreach ($familias as $k => $v) {
29     echo "<code><li>$v</li></code>";
30 }
```

Para ello se instancia un nuevo cliente Soap, se realiza una llamada a cada funci n mediante el m todo m gico __soapCall() y se pasa como primer par metro la funci n a la que llama, y como segundo un array con los par metros de la funci n.

5 - Con la extensión "php2wsdl" crea el archivo "public/generarWsdL.php" para generar el documento WSDL "servicio.wsdl" que guardaremos en "servidorSoap". Utilizando este documento nos crearemos un nuevo servidor SOAP, "servicioW.php" que lo utilice.

Una vez creado, lo ejecutamos con la opción de la barra superior de VSC "Run". Si todo va bien, aparecerá en la carpeta **servidorSoap**, el archivo **servicio.wsdl**

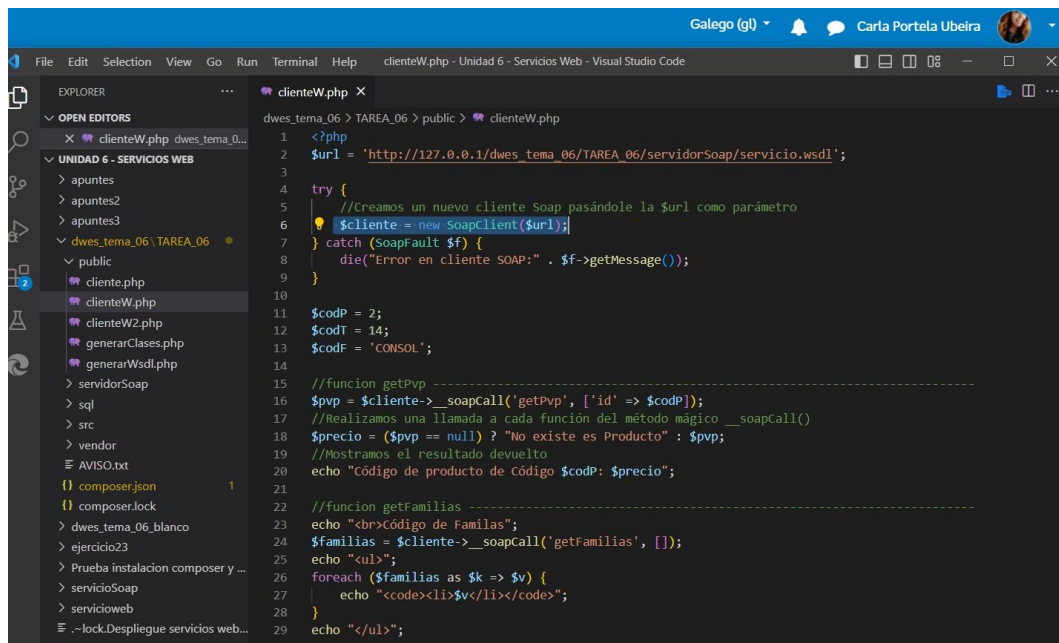


Luego, realizamos una copia del archivo **servicio.php** y lo renombramos con el mismo nombre terminado en W (**servicioW.php**).

Y en él

establecemos la **\$url** con la url del servicio e instanciamos el nuevo SoapServer con la **\$url** como parámetro.

6 - Para comprobar la correcta ejecución de este nuevo servicio, programa también un cliente con nombre "clienteW.php" en la carpeta "public" que realice una llamada a cada una de las funciones programadas y muestre el resultado obtenido.

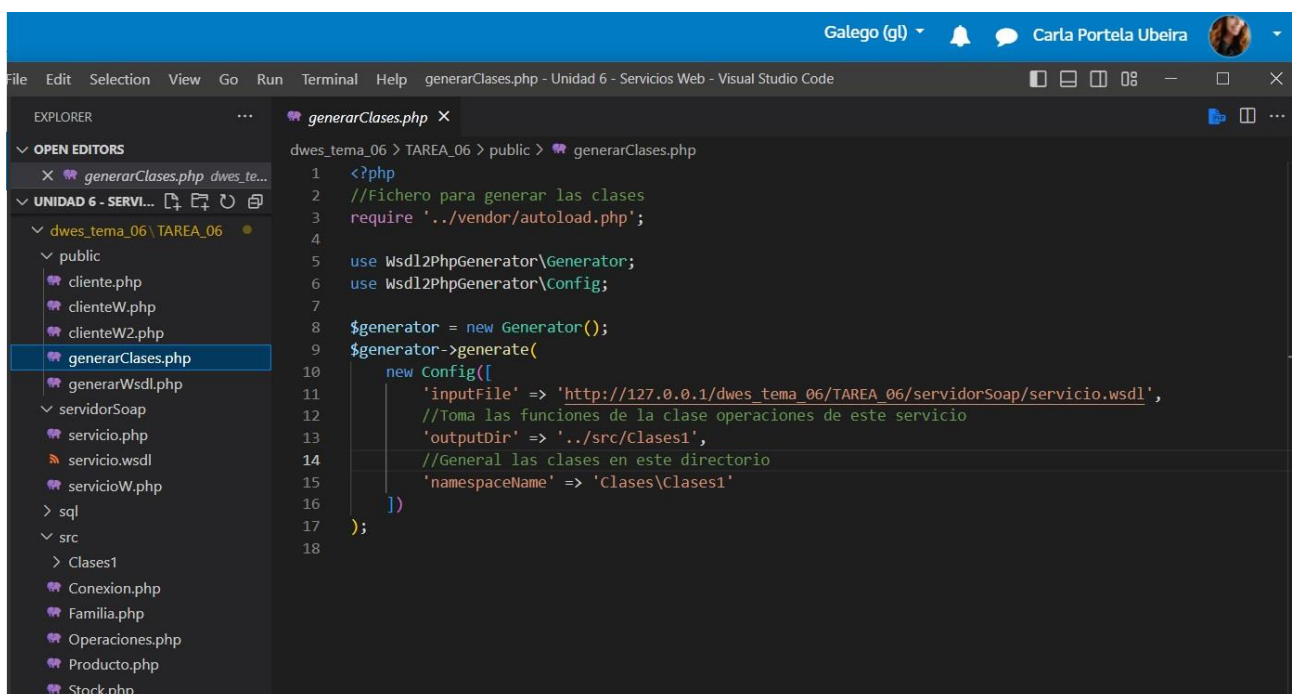


```
1 <?php
2 $url = 'http://127.0.0.1/dwes_tema_06/TAREA_06/servidorSoap/servicio.wsdl';
3
4 try {
5     //Creamos un nuevo cliente Soap pasándole la $url como parámetro
6     $cliente = new SoapClient($url);
7 } catch (SoapFault $f) {
8     die("Error en cliente SOAP: " . $f->getMessage());
9 }
10
11 $codP = 2;
12 $codT = 14;
13 $codF = 'CONSOL';
14
15 //funcion getPvp -----
16 $pvp = $cliente->__soapCall('getPvp', ['id' => $codP]);
17 //Realizamos una llamada a cada función del método mágico __soapCall()
18 $precio = ($pvp == null) ? "No existe es Producto" : $pvp;
19 //Mostramos el resultado devuelto
20 echo "Código de producto de Código $codP: $precio";
21
22 //funcion getFamiliias -----
23 echo "<br>Código de Familias";
24 $familias = $cliente->__soapCall('getFamiliias', []);
25 echo "<ul>";
26 foreach ($familias as $k => $v) {
27     echo "<code><li>$v</li></code>";
28 }
29 echo "</ul>";
```

7 -
Partiendo
de este
nuevo
servicio y
de su

descripción crearemos el fichero "public/generarClases.php", que utilizará la herramienta "wsdl2php" para obtener una clase PHP. Esta clase la guardaremos en la carpeta "Clases1" dentro de "src".

Generamos las clases en la carpeta /src/ ejecutando el archivo **generarClases.php** de la carpeta /public/

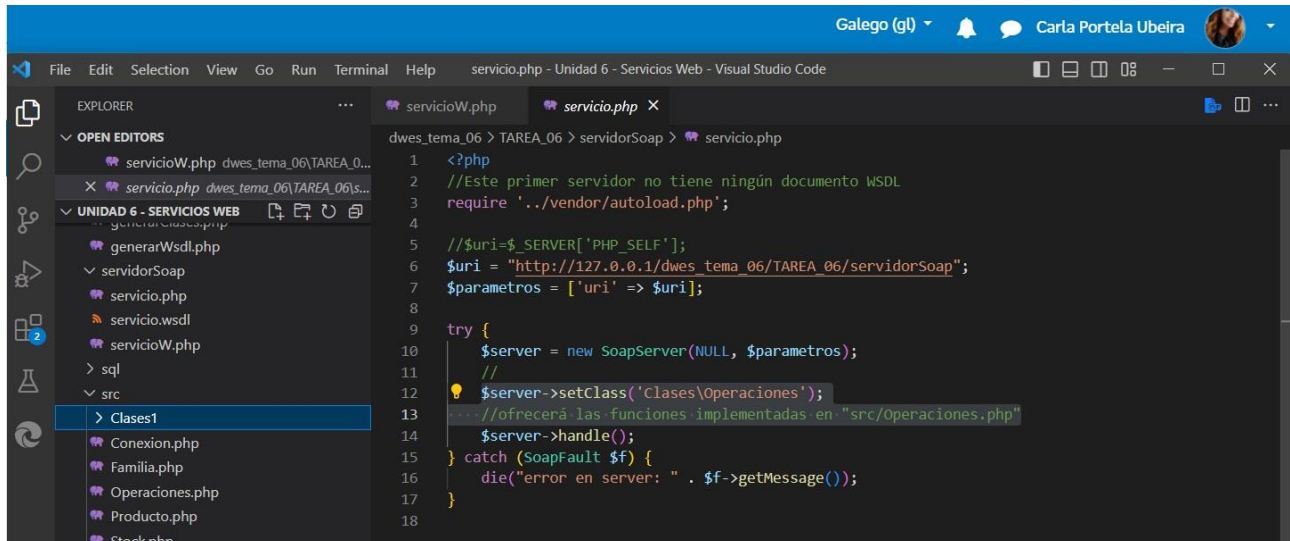


```
1 <?php
2 //Fichero para generar las clases
3 require '../vendor/autoload.php';
4
5 use Wsdl2PhpGenerator\Generator;
6 use Wsdl2PhpGenerator\Config;
7
8 $generator = new Generator();
9 $generator->generate(
10     new Config([
11         'inputFile' => 'http://127.0.0.1/dwes_tema_06/TAREA_06/servidorSoap/servicio.wsdl',
12         //Toma las funciones de la clase operaciones de este servicio
13         'outputDir' => '../src/Clases1',
14         //General las clases en este directorio
15         'namespaceName' => 'Clases\Clases1'
16     ])
17 );
```

Al realizarlo, la primera vez, aparecía un error en el archivo **generarClases.php** de la carpeta /public/, en la url de 'inputFile' que era
'http://127.0.0.1/dwes_tema_06_blanco/TAREA_06/servidorSoap/servicio.wsdl');

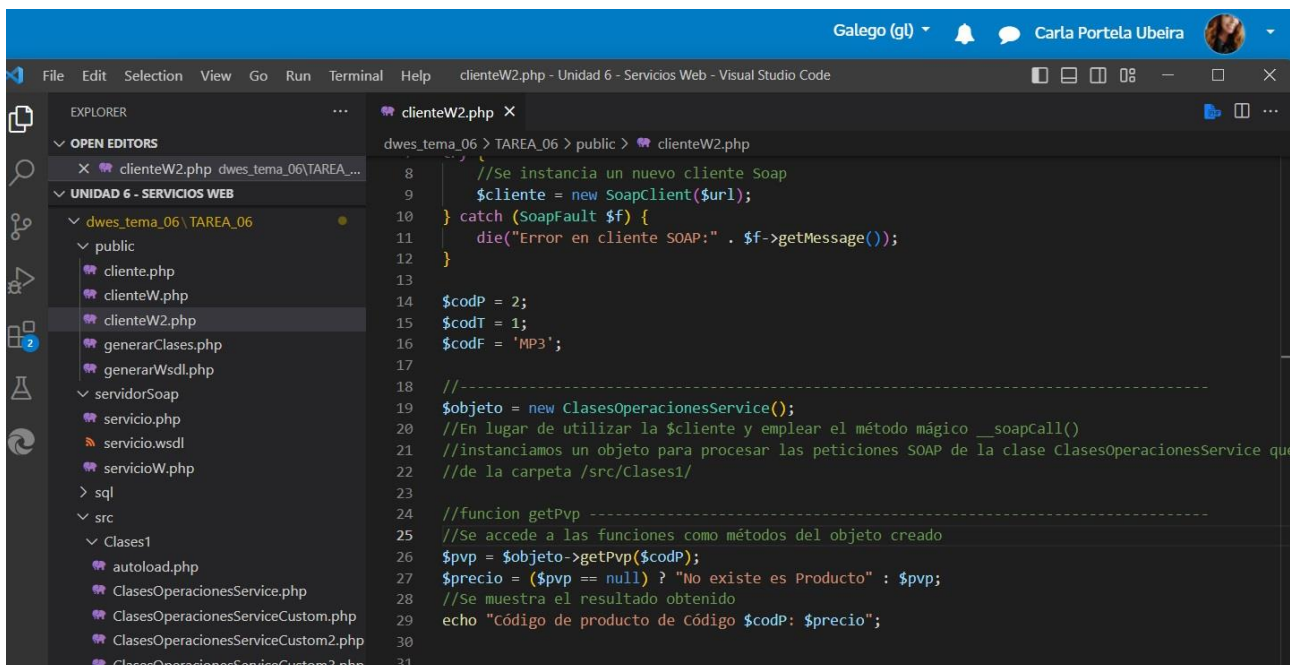
Tras corregir estos errores, paramos y arrancamos el servidor Apache para que se hagan efectivos los cambios y en la terminal hacemos de nuevo un *composer update*.
En caso de usar https en lugar de http, descomentaríamos las líneas 'ssl' en **generar.php** en la carpeta **/public/**

Es recomendable, definir una clase que implemente los métodos que darán funcionalidad al servicio en un archivo aparte (**Operaciones.php**) que se llamará en el **servicio.php** y **servicioW.php**



```
1 <?php
2 //Este primer servidor no tiene ningún documento WSDL
3 require '../vendor/autoload.php';
4
5 // $uri=$_SERVER['PHP_SELF'];
6 $uri = "http://127.0.0.1/dwes_tema_06/TAREA_06/servidorSoap";
7 $parametros = ['uri' => $uri];
8
9 try {
10     $server = new SoapServer(NULL, $parametros);
11     //
12     $server->setClass('Clases\Operaciones');
13     //...//ofrecerá las funciones implementadas en "src/Operaciones.php"
14     $server->handle();
15 } catch (SoapFault $f) {
16     die("error en server: " . $f->getMessage());
17 }
18
```

8 - Crea un nuevo cliente llamado "clienteW2.php" que se base en ésta clase, para probar el nuevo servicio, mostrando los resultados obtenidos de forma similar a como hiciste en los casos anteriores.



```
8 //Se instancia un nuevo cliente Soap
9 $cliente = new SoapClient($url);
10 } catch (SoapFault $f) {
11     die("Error en cliente SOAP: " . $f->getMessage());
12 }
13
14 $codP = 2;
15 $codT = 1;
16 $codF = 'MP3';
17
18 //-----
19 $objeto = new ClasesOperacionesService();
20 //En lugar de utilizar la $cliente y emplear el método mágico __soapCall()
21 //instanciamos un objeto para procesar las peticiones SOAP de la clase ClasesOperacionesService que
22 //de la carpeta /src/Clases1/
23
24 //funcion getPvp -----
25 //Se accede a las funciones como métodos del objeto creado
26 $pvp = $objeto->getPvp($codP);
27 $precio = ($pvp == null) ? "No existe es Producto" : $pvp;
28 //Se muestra el resultado obtenido
29 echo "Código de producto de Código $codP: $precio";
30
31
```

Se instancia un nuevo cliente Soap pero esta vez, en lugar de llamar al método mágico `__soapCall()`, se instancia un objeto de la clase `ClaseOperacionesService` de la carpeta `/src/Clases1/` y se acceden a las funciones como métodos del objeto instanciado.