

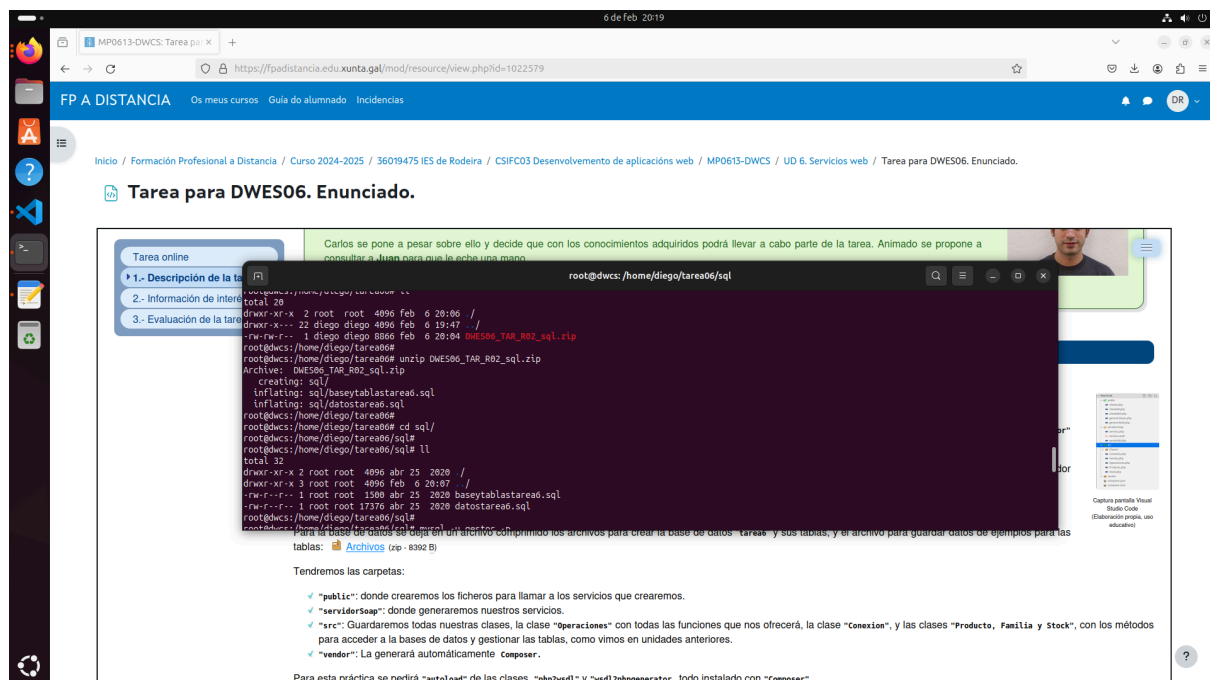
Tarea realizada en Ubuntu

Documentado en la parte B de la tarea realizada en la Unidad 5:

- Descargar e instalar Composer
- Descargar e instalar MySQL (o Workbench)
- Instalar PHP y extensión para utilizar PDO

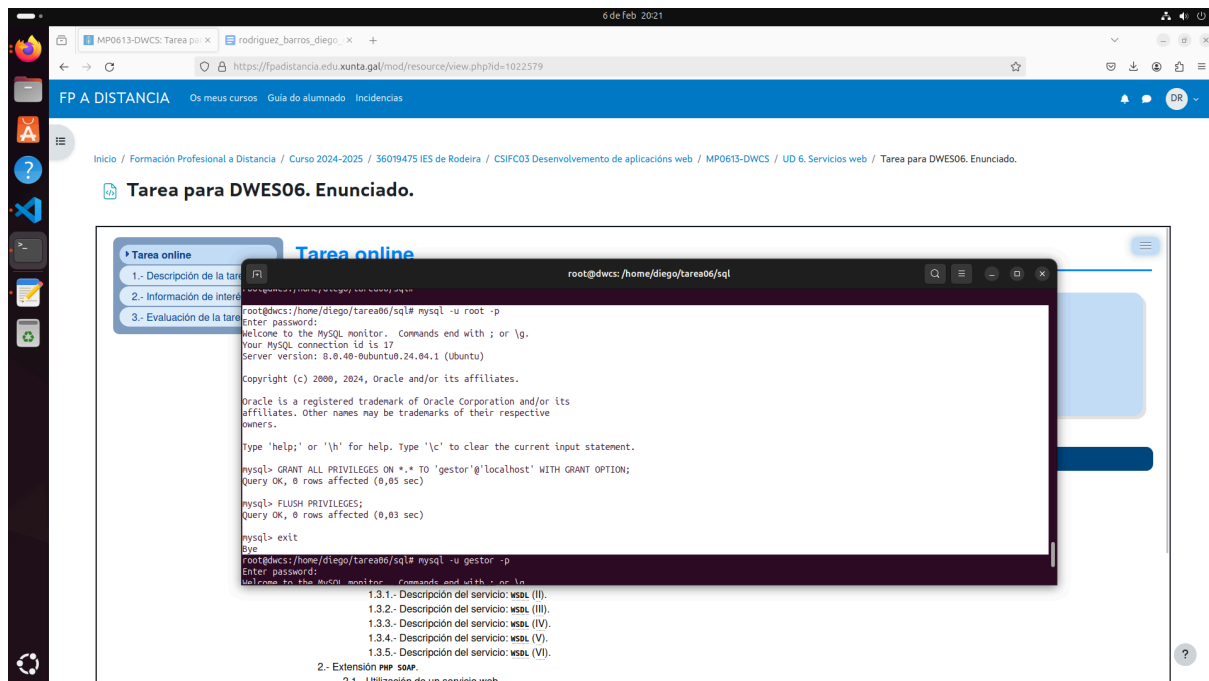
Paso 1: Crear la base de datos

Descomprimir el archivo ZIP.



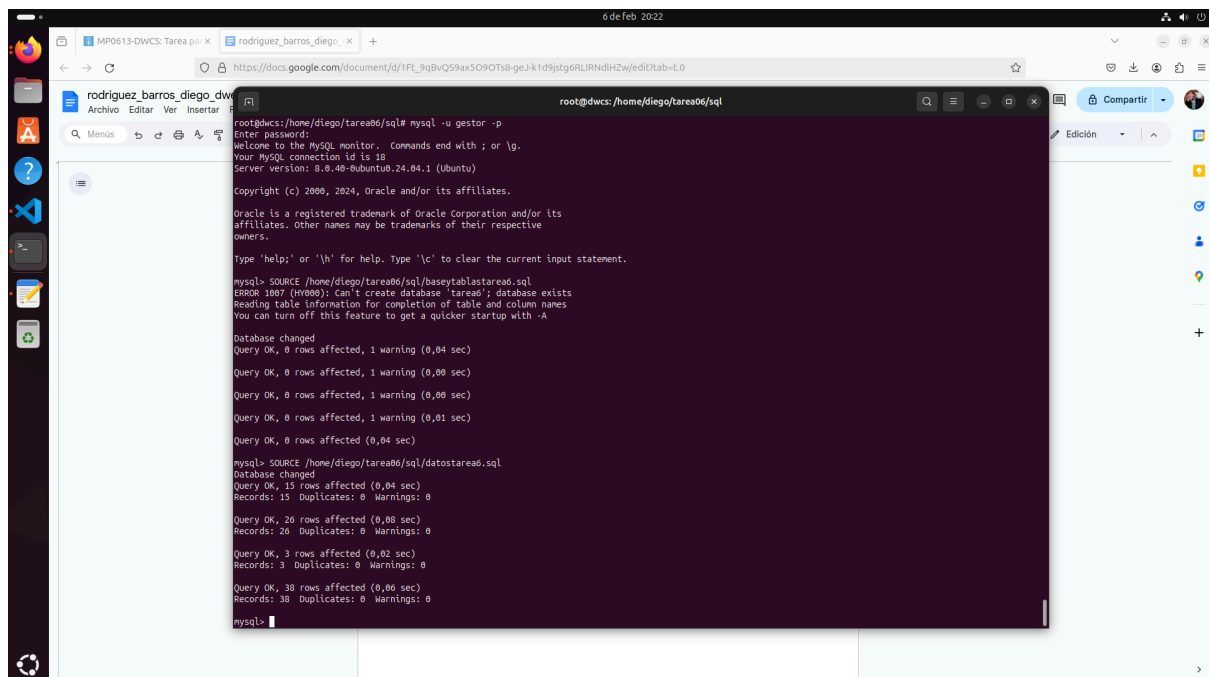
Acceder a MySQL con el usuario root, dar permisos totales al usuario gestor sobre todas las bases de datos.

```
mysql -u root -p
GRANT ALL PRIVILEGES ON . TO 'gestor'@'localhost';
FLUSH PRIVILEGES;
```



Acceder a MySQL con el usuario gestor, ejecutar los scripts sql contenidos en el ZIP para crear la base de datos y las tablas e insertar datos.

```
mysql -u gestor -p
SOURCE ruta_del_archivo.sql;
```



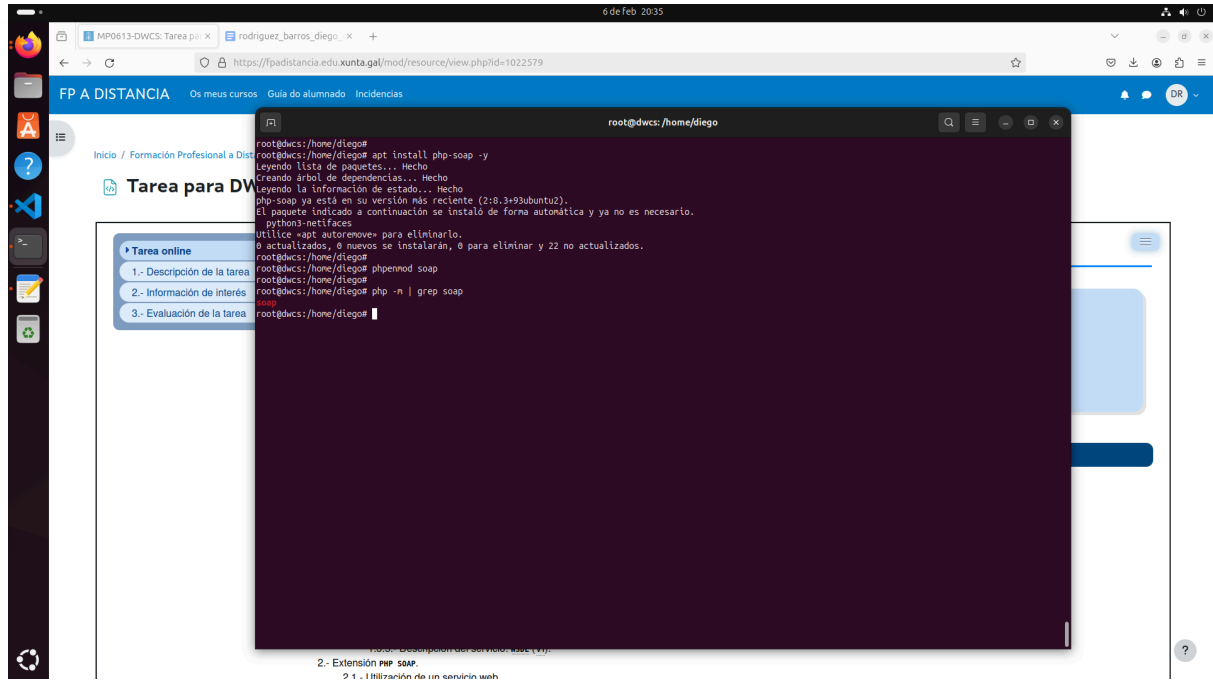
Paso 2: Instalar y habilitar la extensiones SOAP y XML para PHP

```
apt install php-soap php-xml -y
```

```
phpenmod soap
```

```
php -m | grep soap
```

```
php -m | grep dom
```

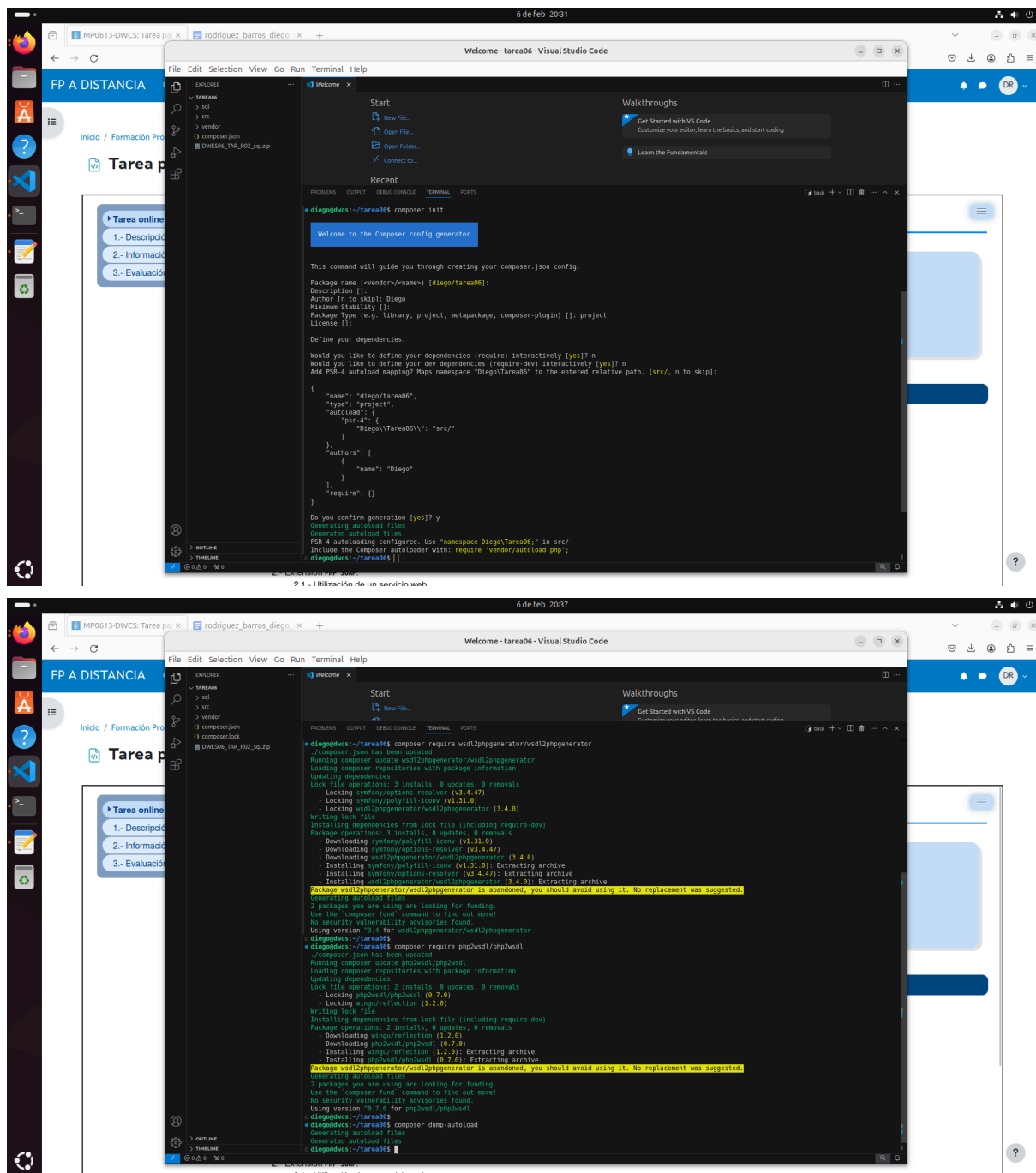


Paso 3: Configurar Composer

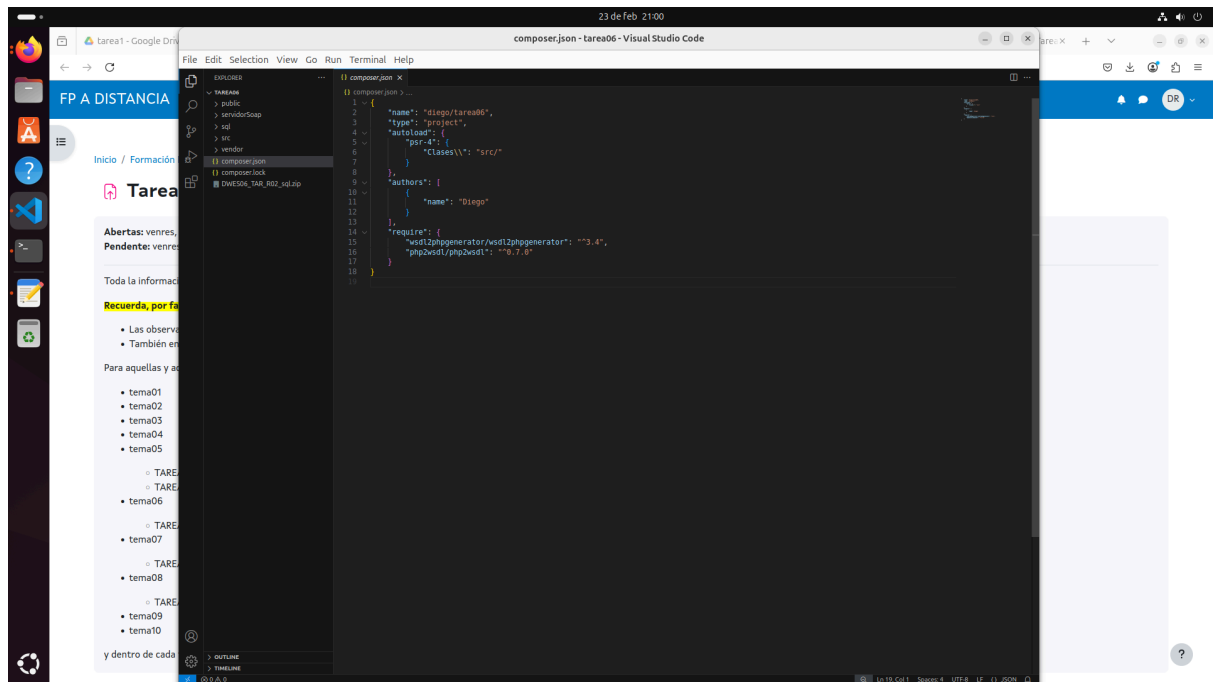
Desde la terminal de VSC, ejecutar los siguientes comandos para iniciar el proyecto, generar los directorios *vendor* y *src*, el archivo *composer.json* y añadir las librerías necesarias:

```
composer init
composer require wsd12phpgenerator/wsd12phpgenerator
composer require php2wsdl/php2wsdl

composer dump-autoload
```



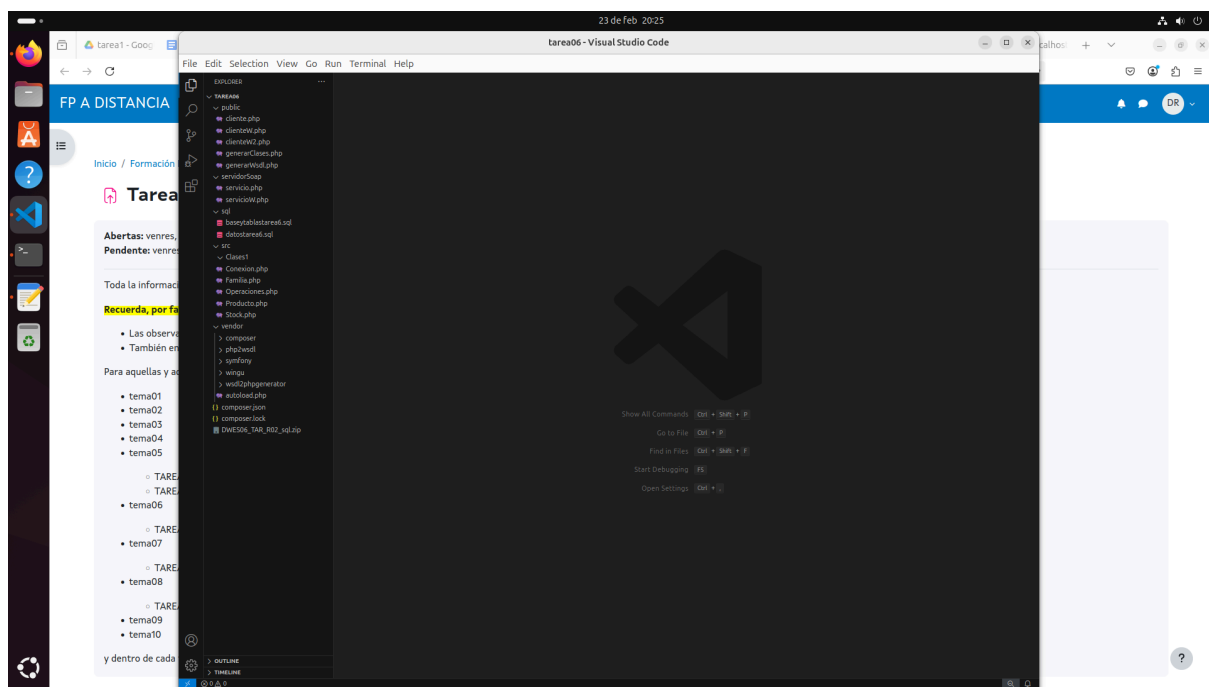
Con esto se genera el archivo composer.json.



Nota: Las capturas no son exactamente las que han generado el composer.json anterior, revisar el apartado **autoload** del mismo.

Paso 4: Añadir los directorios y clases php restantes del proyecto.

```
tarea06/
├── public/
│   ├── cliente.php
│   ├── clienteW.php
│   ├── clienteW2.php
│   ├── generarWsd1.php
│   └── generarClases.php
├── servidorSoap/
│   ├── servicio.php
│   ├── servicioW.php
│   └── servicio.wsd1 (generado)
├── src/
│   ├── Conexion.php
│   ├── Operaciones.php
│   ├── Producto.php
│   ├── Familia.php
│   ├── Stock.php
│   └── Clases1/ (generado)
└── vendor/ (generado)
```

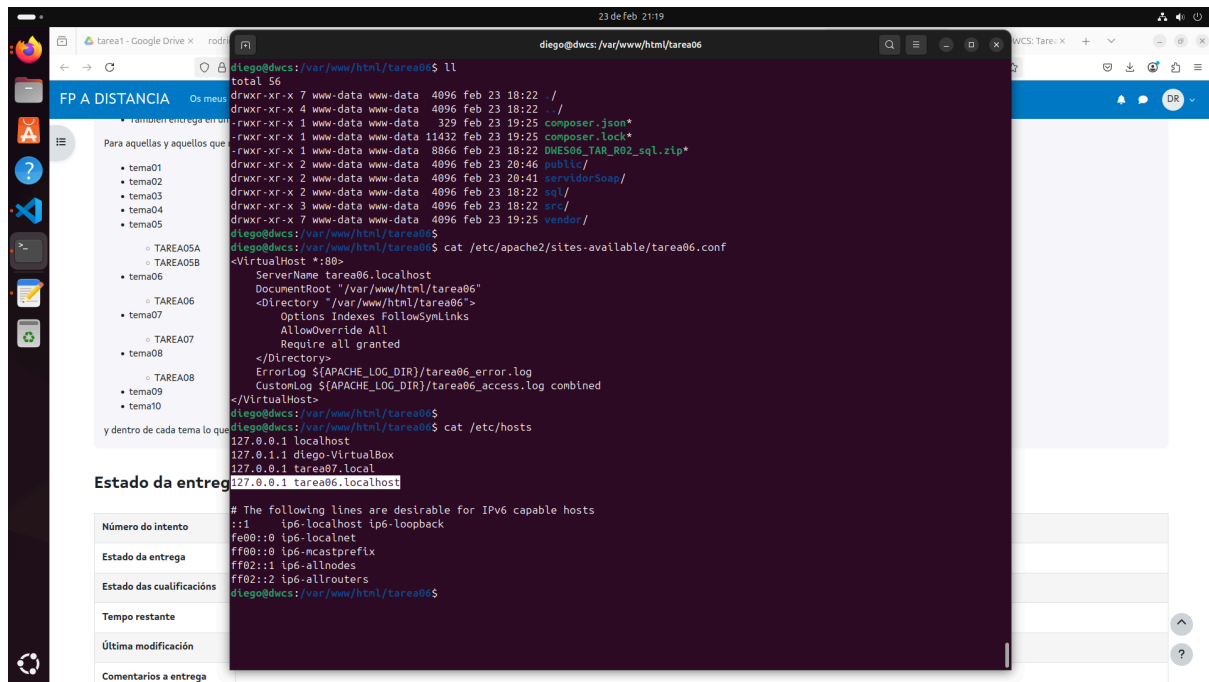


Paso 5: Implementar las clases

* Ver código subido al repositorio

Paso 6: Configurar Apache

Mover el proyecto tarea06 a /var/www/html, cambiar permisos con chown y chmod y crear un virtualhost como el de la siguiente imagen:



```
diego@dwcs:/var/www/html/tarea06$ ll
total 56
drwxr-xr-x 7 www-data www-data 4096 Feb 23 18:22 ./
drwxr-xr-x 4 www-data www-data 4096 Feb 23 18:22 ../
-rwxr-xr-x 1 www-data www-data 329 Feb 23 19:25 composer.json*
-rwxr-xr-x 1 www-data www-data 11432 Feb 23 19:25 composer.lock*
-rwxr-xr-x 1 www-data www-data 8866 Feb 23 18:22 DNE596_TAR_R02_sql.zip*
drwxr-xr-x 2 www-data www-data 4096 Feb 23 20:46 public/
drwxr-xr-x 2 www-data www-data 4096 Feb 23 20:41 servidorSoap/
drwxr-xr-x 2 www-data www-data 4096 Feb 23 18:22 sql/
drwxr-xr-x 3 www-data www-data 4096 Feb 23 18:22 src/
drwxr-xr-x 7 www-data www-data 4096 Feb 23 19:25 vendor/

diego@dwcs:/var/www/html/tarea06$ cat /etc/apache2/sites-available/tarea06.conf
<VirtualHost *:80>
    ServerName tarea06.localhost
    DocumentRoot "/var/www/html/tarea06"
    <Directory "/var/www/html/tarea06">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/tarea06_error.log
    CustomLog ${APACHE_LOG_DIR}/tarea06_access.log combined
</VirtualHost>

diego@dwcs:/var/www/html/tarea06$ cat /etc/hosts
127.0.0.1 localhost
127.0.0.1 tarea06.local
127.0.0.1 tarea07.local
127.0.0.1 tarea06.localhost

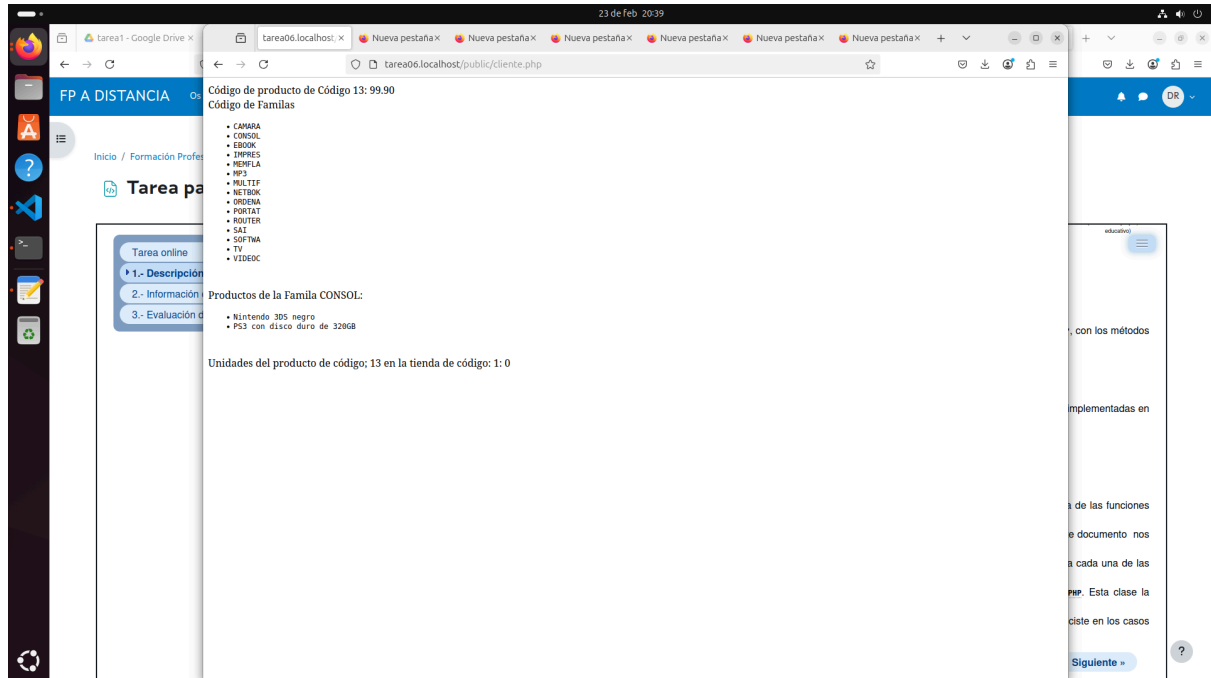
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe80::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff00::1 ip6-allnodes
ff00::2 ip6-allrouters

diego@dwcs:/var/www/html/tarea06$
```

Deshabilitar virtualhost por defecto (00-default.conf), habilitar el nuevo y reiniciar Apache. Por último, modificar el archivo /etc/hosts para que resuelva el nombre del virtualhost configurado (tarea06.localhost).

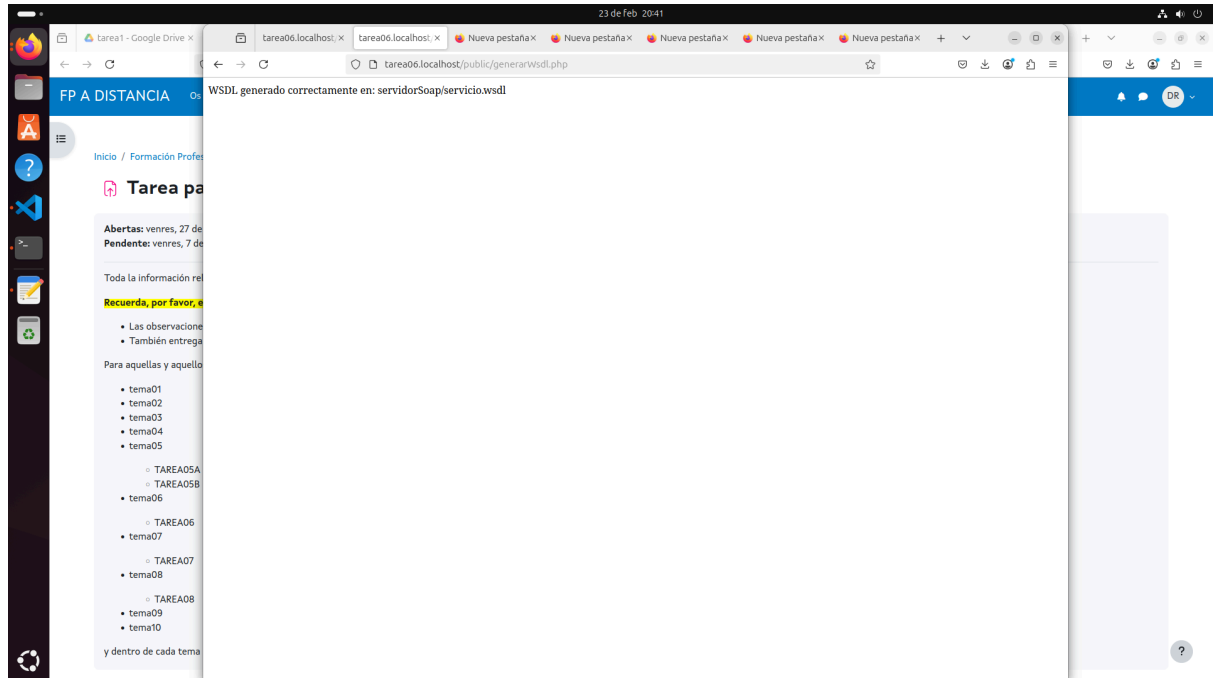
Paso 7: Servicio SOAP sin WSDL (servicio.php)

Acceder a <http://tarea06.localhost/public/cliente.php>. Este cliente consume el servicio SOAP sin usar un WSDL, configurando manualmente la ubicación y URI del servidor. Usa SoapClient directamente con parámetros definidos a mano. Este método es propenso a errores (nombres de funciones o parámetros mal escritos) y no hay validación automática de tipos de datos.

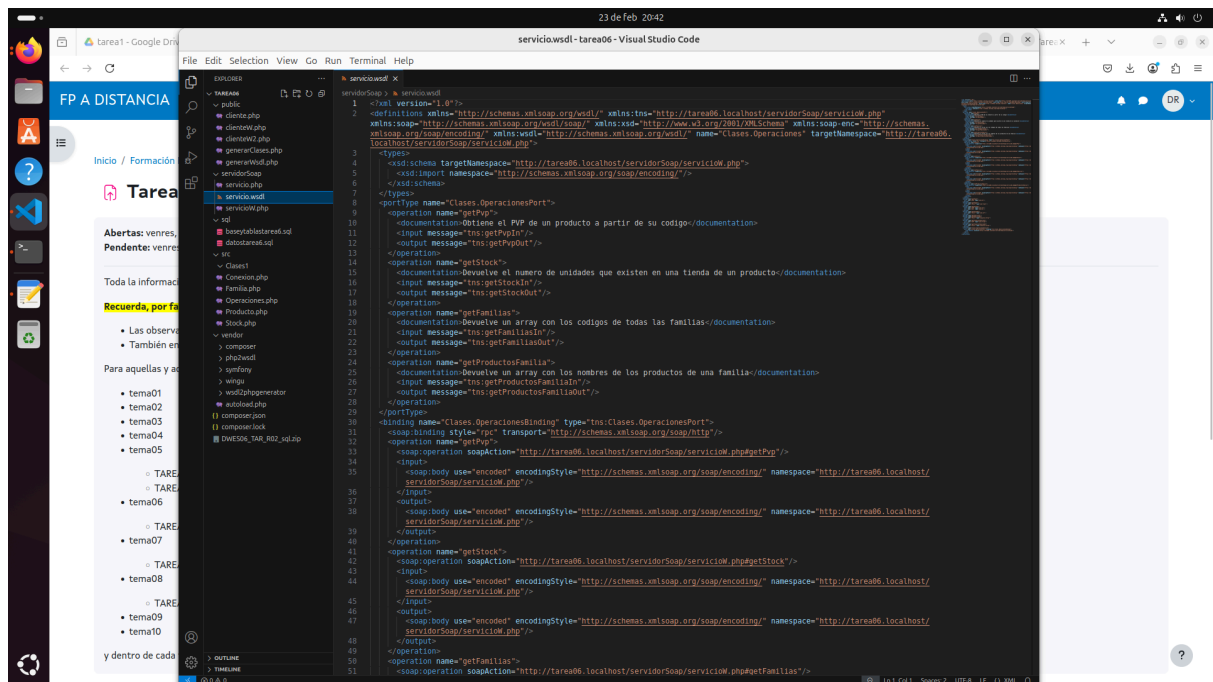


Paso 8: Servicio SOAP con WSDL (servicioW.php)

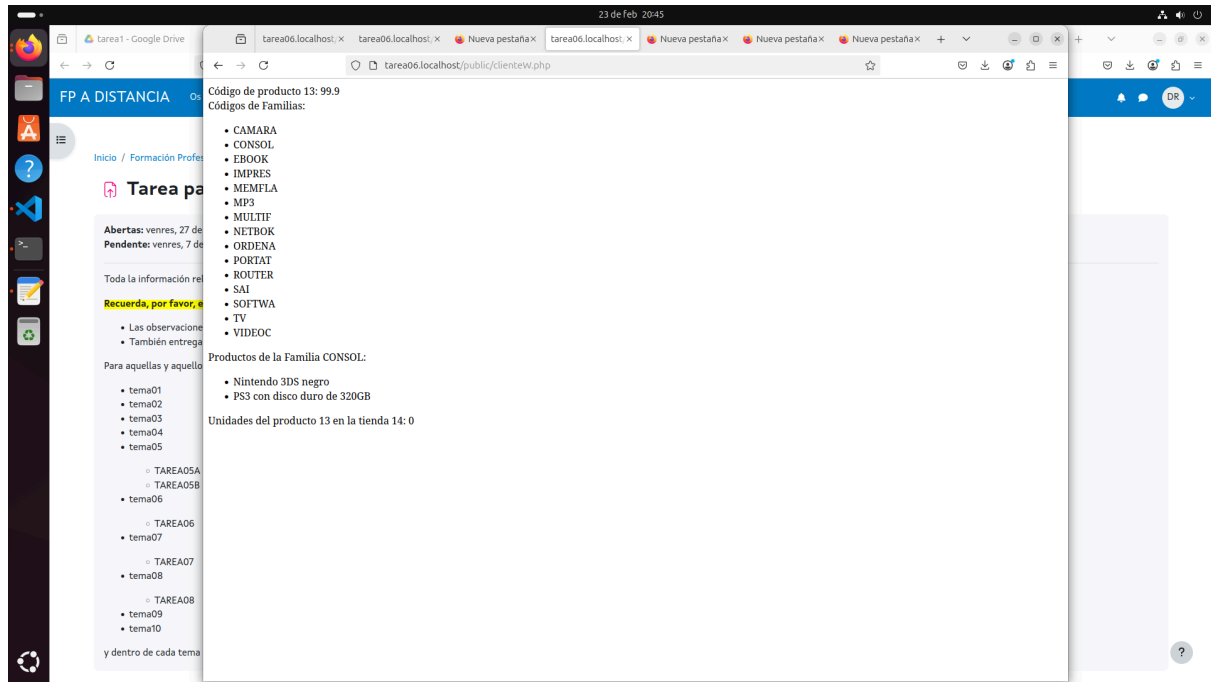
Generar el WSDL (generarWsdL.php) accediendo a <http://tarea06.localhost/public/generarWsdL.php>.



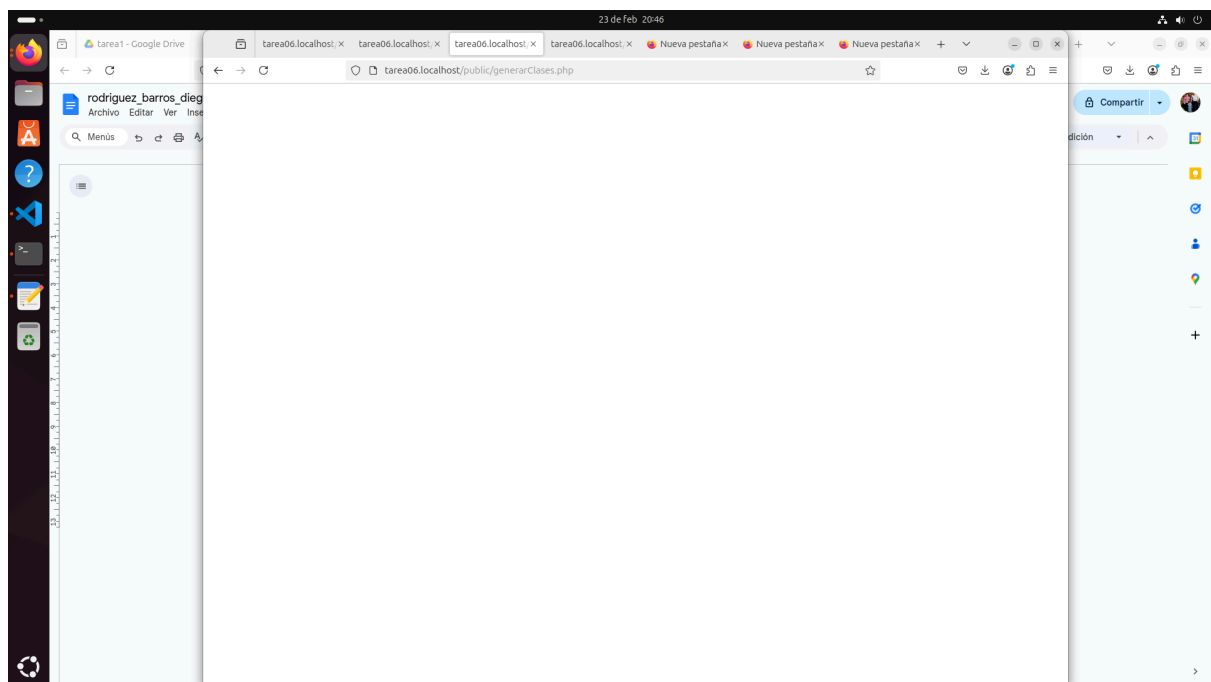
Comprobar en VSC que se ha generado correctamente el archivo XML.



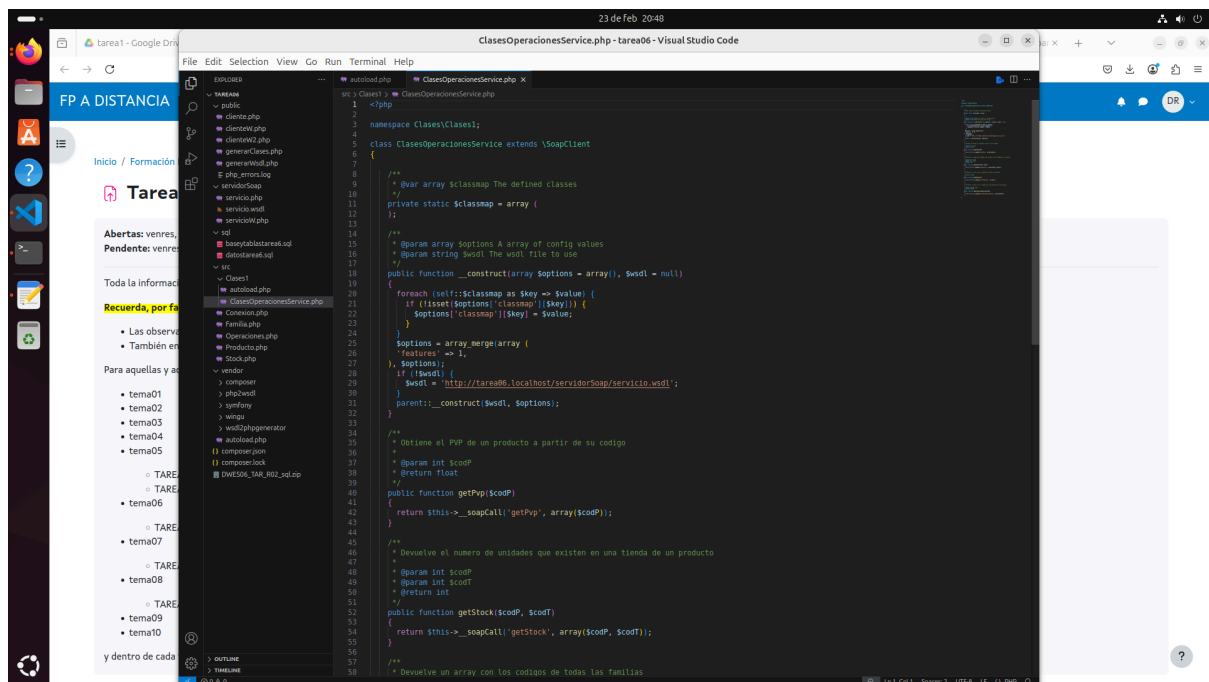
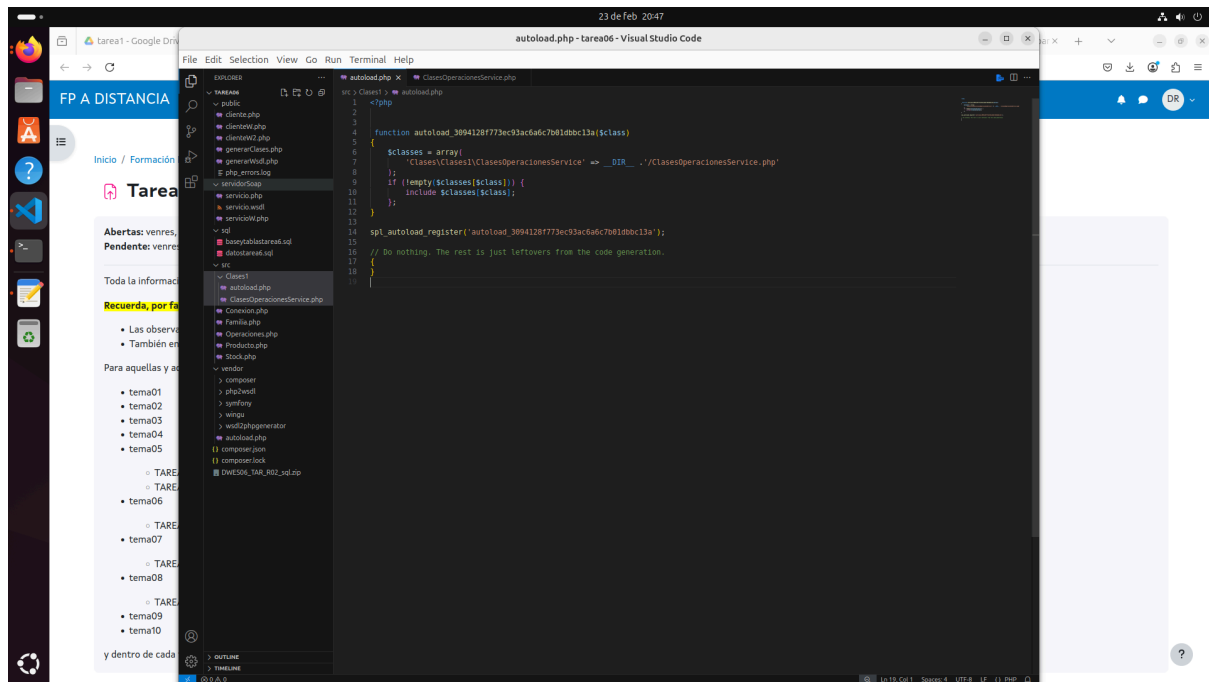
Probar el Cliente sin clases generadas (clienteW.php) accediendo a <http://tarea06.localhost/public/clienteW.php>. Este cliente consume el servicio SOAP usando el WSDL generado, que describe formalmente las operaciones y tipos de datos. Este método es menos propenso a errores ya que el WSDL define las operaciones y tipos de datos y es mantenible (si el servicio cambia, el WSDL actualizado refleja los cambios). Aún así, requiere escribir manualmente los nombres de funciones y parámetros.



Generar Clases desde WSDL (generarClases.php) accediendo a <http://tarea06.localhost/public/generarClases.php>.



Comprobar en VSC que se han generado correctamente las clases en el directorio Clases1.



Probar el cliente con clases generadas (clienteW2.php) accediendo a <http://tarea06.localhost/public/clienteW2.php>. Este cliente consume el servicio SOAP usando clases PHP generadas automáticamente utilizando wsdl2phpgenerator. Este método es más limpio, tiene una validación fuerte dado que los tipos de datos se comprueban en tiempo de desarrollo y también es mantenible dado que las clases se regeneran si el WSDL cambia, aunque hay que hacerlo manualmente.

