

****

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

**PROGRAMACIÓN AVANZADA**

**PROYECTO PRIMER PARCIAL**

**RESTFUL PHP**

**INTEGRANTES**

**Alexander Vicente Mejia Sizonenko**

**Bolívar David Llerena Fuenmayor**

**Jean Karlo Obando Ramos**

**SANGOLQUÍ**

**2017-2018**

**TEMA**

SERVICIOS RESTFUL EN PHP.

**OBJETIVO DE APRENDIZAJE**

Explicar y detallar los pasos necesarios para realizar servicios RESTful usando tecnología PHP.

**HERRAMIENTAS**

Slim – Php Framework

Xampp o Wamp- Web Server/Stack

Composer – Dependency Manager for PHP

Atom – Text Editor

# **MARCO TEÓRICO**

* **REST**

REST son los acrónimos para REpresentational State Transfer que es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. [1]

Los servicios REST son aquellos que permiten a través de una URI comunicar un cliente con un servidor aplicando los métodos HTTP.

Estos métodos son el get, post, put y delete, haciendo una analogía con las operaciones CRUD diremos definiremos a continuación.

El Get me permite obtener datos del servidor.

El Post me permite insertar nuevos elementos al servidor.

El Put me permite actualizar elementos del servidor.

El Delete me permite eliminar elementos del servidor.

Estos tipos de servicios se diferencian mucho de servicios web SOAP, presenta por los menos una arquitectura cliente/servidor y en capas para desarrollo.

En el diseño de URIs es importante considerar los niveles de madurez para ello presentes que van desde no usar URIs hasta implementar en ellos HATOES para su mejor uso.

Los niveles de madurez de un URI son los siguientes

* **NIVEL 0**

Los servicios que están en este nivel utilizan HTTP como un protocolo de transporte para codificar servicios remotos.

* **NIVEL 1**

Los servicios que están en grado de madurez e introducen el concepto de cada recurso teniendo su propia URI, que permite recuperarlo.

* **NIVEL 2**

Se realiza para hacer peticiones al servidor mediante métodos HTTP, se puede reusar o sobrecargar los URIs.

* **NIVEL 3**

Accedemos a los servicios REST, utilizando y construyendo las propiedades. Retornan enlaces que permiten obtener los enlaces de los clientes. [3]

* **PHP**

PHP es un lenguaje de scripts que surgió en 1994 por Rasmus Lerdorf. En un principio PHP era un acrónimo para “Personal Home Page” pero más adelante pasó a ser un acrónimo recursivo que quiere decir “PHP: Hypertext Processor”. Generalmente es un lenguaje muy utilizado en desarrollo web y es de código libre. [4]

PHP es código que puede ser embebido en HTML o HTML 5 y funciona del lado del servidor, una de sus principales diferencias con lenguajes como JavaScript. PHP al funcionar del lado del servidor hace que su código no se visible por sus clientes y así estos no pueden conocer los procesos que se realiza para procesar sus peticiones. El código PHP generalmente es procesado por un intérprete PHP que usualmente es instalado como un módulo en un servidor o también se lo procesa por medio de un ejecutable de Common Gateway Interface (CGI). [4]

PHP a lo largo de su historia ha presentado diversas versiones que se han ido expandiendo con el tiempo, el historial de versiones es:

* PHP/FI
* PHP/FI 2
* PHP 3.0, que es la primera que se acerca a la versión final de esta
* PHP 5.0
* PHP 7.0, que es la última versión vigente

[5]

* **REST EN PHP**
  + **Slim**

Slim, es un mini framework REST para PHP que usa como patrón de diseño MVC, el cual maneja la parte de backend para diversas aplicaciones web basadas en PHP, este mini framework permite la creación de aplicaciones REST con sintaxis muy simples, sin la necesidad de levantar una infraestructura llena de módulos y opciones que no van a ser utilizadas. [6]

.

* + **Composer**

Es un manejador de dependencias, que se compone de paquetes y librerías, cuya instalación siempre es de manera local para cualquier proyecto. Las librerías y paquetes son instaladas en un directorio por defecto. Composer es capaz de instalar librerías que requiere un proyecto con las versiones que necesiten justas y necesarias. [7]



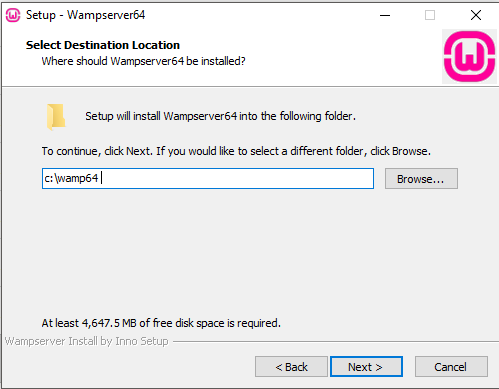
# **PRÁCTICA**

INSTALACIÓN Y CONFIGURACIÓN DEL WEB SERVER STACK

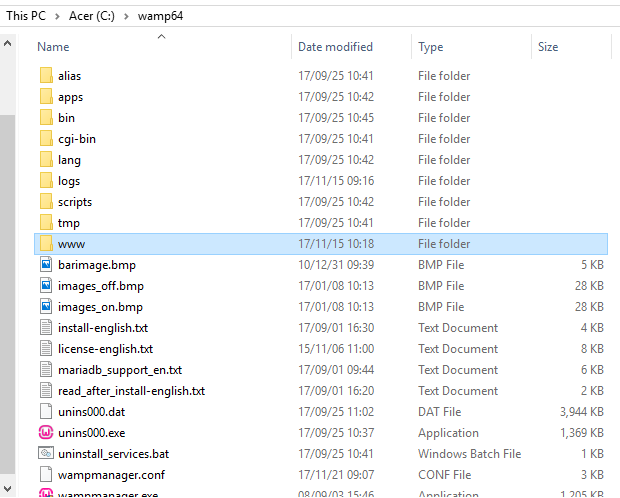
**WAMP**

Para poder realizar el ejercicio de REST con PHP es posible usar el stack de software WAMP que tendrá Apache, PHP, MySQL (aunque se lo puede usar independientemente). La descarga del stack se lo puede realizar en <https://sourceforge.net/projects/wampserver/files/>.

La instalación es bastante sencilla de realizar y conlleva pasos muy similares a la instalación de cualquier otro programa en Windows.



Una vez realizada la operación de instalación tenemos que preparar la manera en que se tendrá una disposición del servicio rest. Es decir, donde estarán los archivos a manejar. Todo esto se realiza en el directorio www que está en el lugar de instalación del wamp64.

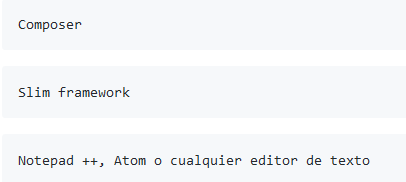


En este lugar crearemos un directorio denominado “restphp”. Dentro pondremos una carpeta “src” donde estará nuestro código.

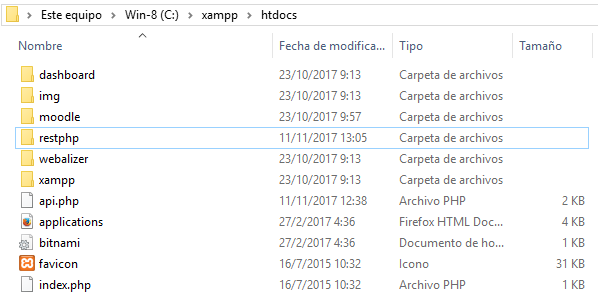
Con esto estaremos listos para desarrollar código en PHP.

**XAMP**

Para realizar servicios REST con el stack XAMP es necesario instalar los siguientes componentes:



Es necesario ubicarse en la carpeta htdocs en el directorio Xampp (usualmente en el disco C:), donde se creará el proyecto en este caso restphp.

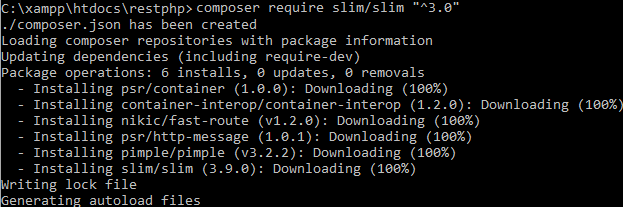


Se ejecutará un cmd donde nos accedemos al proyecto recién creado con el comando cd.

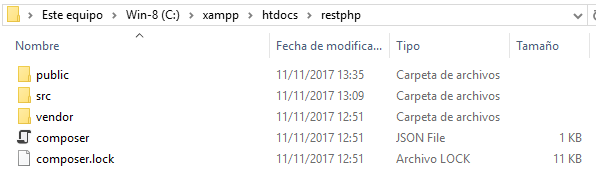


Luego se ejecutará el comando de instalación slim.





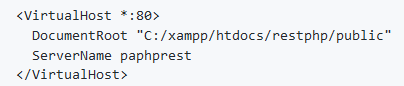
Observaremos que hay 3 nuevos archivos creados en nuestro proyecto, además se deberá agregar dos carpetas adicionales llamadas public y src. Public será la carpeta donde residirá nuestras configuraciones respectivas a la página de inicio de nuestro proyecto. Mientras que Src será la carpeta que contendrá nuestros archivos. php haciendo referencia a las respectivas funcionalidades RESTful.



Para agregar un host virtual es necesario añadirlo dentro de la carpeta apache/conf/extra/httpd-vhosts.conf Donde se quitará el comentario de la siguiente línea de código.



Y se agrega un nuevo host virtual de la siguiente manera.



Este host virtual permitirá el acceso a los URI de manera más corta y ágil. Terminada la configuración de este host virtual se deberá agregarlo en el listado de host de la máquina donde se trabaja; en el siguiente directorio.



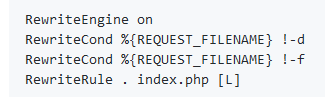
Se procede a editar el archivo host agregando el host virtual previamente configurado con la siguiente línea.

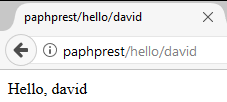


Finalmente se agrega dos archivos en la carpeta public llamados index.php y .htaccess Donde el contenido de index.php será una página de bienvenida que recibe un nombre en su uri.



Mientras que .htaccess será un archivo de configuración necesario para habilitar nuestro host





Es necesario ejecutar el script **peliculas.sql** o crear una base de datos películas con una tabla **actor** con los siguientes atributos.

CREATE TABLE `actor` (

`actor\_id` smallint(5) unsigned NOT NULL AUTO\_INCREMENT,

`ACTOR\_NAME` varchar(15) NOT NULL,

`ACTOR\_LASTNAME` varchar(15) NOT NULL,

`ACTOR\_TYPE` varchar(15) DEFAULT NULL,

`ACTOR\_BIRTH\_DATE` date DEFAULT NULL,

PRIMARY KEY (`actor\_id`)

)

**REST CON PHP**

CONEXIÓN A BASE DE DATOS

<?php

class configdb

{

private $host = 'localhost';

private $user = 'root';

private $password = 'qwerty123';

private $database = 'peliculas';

public function connect()

{

$connection\_mysql = "mysql:host=$this->host;dbname=$this->database";

$connectionDB = new PDO($connection\_mysql, $this->user, $this->password);

$connectionDB->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$connectionDB -> exec("set names UTF8");

return $connectionDB;

}

}

?>

GET

$app->get('/peliculas/actores', function(Request $request, Response $response)

{

$read = 'SELECT \* FROM actor';

try

{

$database = new configdb();

$database = $database->connect();

$execute = $database->query($read);

$actores = $execute->fetchAll(PDO::FETCH\_OBJ);

$database = null;

header('Access-Control-Allow-Origin: \*');

header("Content-Type: application/json;charset=utf-8");

echo json\_encode($actores);

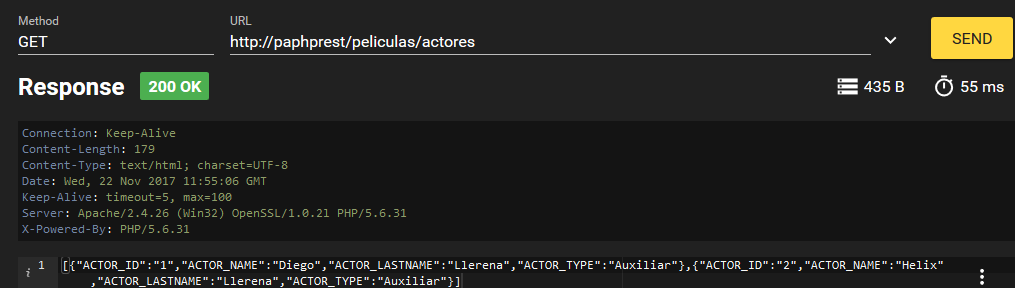
}catch(PDOException $e)

{

echo '{"error": {"text": '.$e->getMessage().'}';

}

});



**GET**

$app->get('/peliculas/actores/quantity/{quantity}', function(Request $request, Response $response)

{

$quantity = $request->getAttribute('quantity');

$read = "SELECT \* FROM actor LIMIT $quantity";

try

{

$database = new configdb();

$database = $database->connect();

$execute = $database->query($read);

$actor = $execute->fetchAll(PDO::FETCH\_OBJ);

$database = null;

header('Access-Control-Allow-Origin: \*');

header("Content-Type: application/json;charset=utf-8");

echo json\_encode($actor);

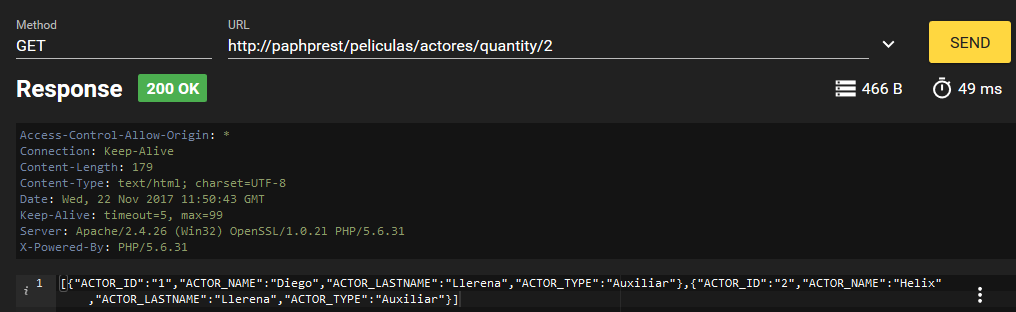
}catch(PDOException $e)

{

echo '{"error": {"text": '.$e->getMessage().'}';

}

});



POST

$app->post('/peliculas/actores', function(Request $request, Response $response)

{

$nombre = $request->getParam('ACTOR\_NAME');

$apellido = $request->getParam('ACTOR\_LASTNAME');

$tipo = $request->getParam('ACTOR\_TYPE');

$birthdate=$request->getParam('ACTOR\_BIRTH\_DATE');

$read = "INSERT INTO actor(ACTOR\_NAME,ACTOR\_LASTNAME,ACTOR\_TYPE,ACTOR\_BIRTH\_DATE)VALUES(:ACTOR\_NAME,:ACTOR\_LASTNAME,:ACTOR\_TYPE,:ACTOR\_BIRTH\_DATE)";

try

{

$database = new configdb();

$database = $database->connect();

$statement = $database->prepare($read);

$statement->bindParam(':ACTOR\_NAME',$nombre);

$statement->bindParam(':ACTOR\_LASTNAME',$apellido);

$statement->bindParam(':ACTOR\_TYPE',$tipo);

$statement->bindParam(':ACTOR\_BIRTH\_DATE',$birthdate);

$statement->execute();

header('Access-Control-Allow-Origin: \*');

header("Content-Type: application/json;charset=utf-8");

echo '{"mensaje": {"text": "Actor añadido"}';

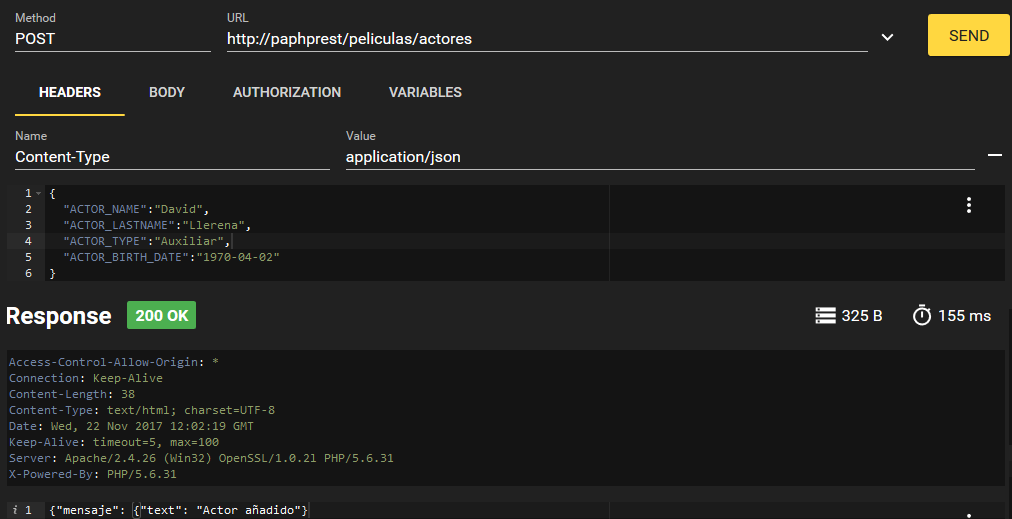
}catch(PDOException $e)

{

echo '{"error": {"text": '.$e->getMessage().'}';

}

});



PUT

$app->put('/peliculas/actores/{id}', function(Request $request, Response $response)

{

$id = $request->getAttribute('id');

$nombre = $request->getParam('ACTOR\_NAME');

$apellido = $request->getParam('ACTOR\_LASTNAME');

$tipo = $request->getParam('ACTOR\_TYPE');

$birthdate=$request->getParam('ACTOR\_BIRTH\_DATE');

$read = "UPDATE actor SET ACTOR\_NAME = :ACTOR\_NAME,

ACTOR\_LASTNAME = :ACTOR\_LASTNAME,

ACTOR\_TYPE = :ACTOR\_TYPE

ACTOR\_BIRTH\_DATE=:ACTOR\_BIRTH\_DATE

WHERE ACTOR\_ID = $id";

try

{

$database = new configdb();

$database = $database->connect();

$statement = $database->prepare($read);

$statement->bindParam(':ACTOR\_NAME',$nombre);

$statement->bindParam(':ACTOR\_LASTNAME',$apellido);

$statement->bindParam(':ACTOR\_TYPE',$tipo);

$statement->bindParam(':ACTOR\_BIRTH\_DATE',$birthdate);

$statement->execute();

header('Access-Control-Allow-Origin: \*');

header("Content-Type: application/json;charset=utf-8");

echo '{"mensaje": {"text": "Actor actualizado"}';

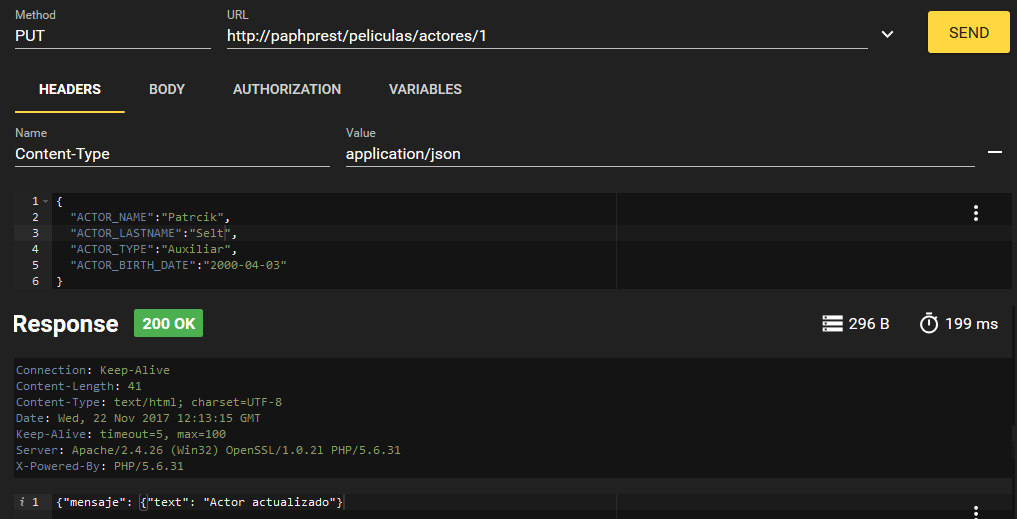
}catch(PDOException $e)

{

echo '{"error": {"text": '.$e->getMessage().'}';

}

});



DELETE

$app->delete('/peliculas/actores/{id}', function(Request $request, Response $response)

{

$id = $request->getAttribute('id');

$read = "DELETE FROM actor WHERE ACTOR\_ID = $id";

try

{

$database = new configdb();

$database = $database->connect();

$statement = $database->prepare($read);

$statement->execute();

header('Access-Control-Allow-Origin: \*');

header("Content-Type: application/json;charset=utf-8");

echo '{"mensaje": {"text": "Actor eliminado"}';

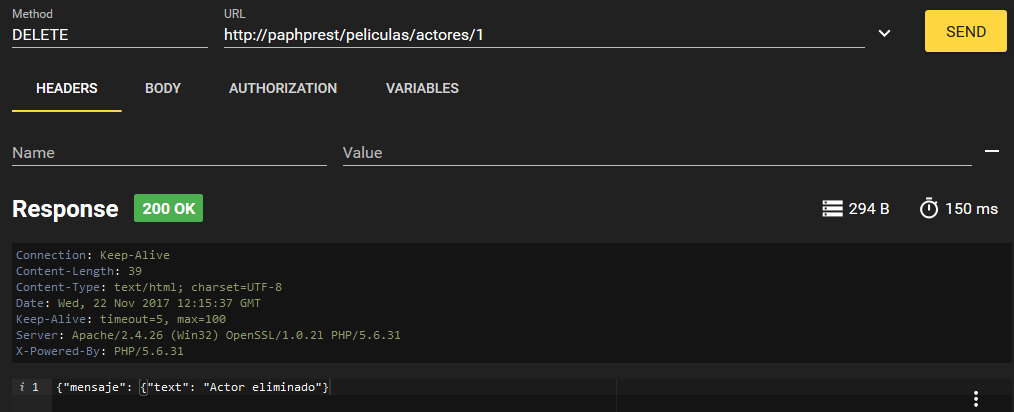
}catch(PDOException $e)

{

echo '{"error": {"text": '.$e->getMessage().'}';

}

});



# **CONCLUSIONES**

* PHP me permite manejar servicios de una manera fácil y sencilla.
* Se requiere de un micro framework como slim para poder implementar los servicios REST.
* En php se especifica el método HTTP que se va a utilizar mediante el uso de una flecha “->”
* La conexión a la base de datos es otro requerimiento del sistema.
* Funciona de manera efectiva tanto en WAMP como en XAMP.
* Es importante considerar el stack de software que se está usando

**Recomendaciones**

1. Es necesario contar con un administrador de paquetes como composer para poder implementar los servicios REST.
2. Es importante configurar de manera correcta mi entorno para trabajar con API REST PHP.
3. La cadena de conexión con la base de datos tiene que estar bien realizada.

# **BIBLIOGRAFÍA**

|  |  |
| --- | --- |
| [1] | asiermarques.com, «asiermarques.com,» [En línea]. Available: http://asiermarques.com/2013/conceptos-sobre-apis-rest/. |
| [2] | C. Á. Caules, «Introducción a Servicios REST,» Arquitectura JAVA, 2014. [En línea]. Available: https://www.arquitecturajava.com/servicios-rest/. |
| [3] | G. Casarini., «Niveles de madurez de los servicios RESTful,» Blog Gabriel Casarini, 2016. [En línea]. Available: http://gabrielcasarini.com/arquitectura/2014/12/03/rest\_niveles\_madurez.html. |
| [4] | The PHP Group, «What is php?,» [En línea]. Available: http://php.net/manual/en/intro-whatis.php. |
| [5] | The PHP Group, «History of PHP,» [En línea]. Available: https://secure.php.net/manual/en/history.php.php. |
| [6] | E. Team, «Slim Framework,» 4 Julio 2013. [En línea]. Available: https://www.elserver.com/slim-mini-framework-rest-para-php/. |
| [7] | Y. Pérez, «¿Que es composer y como usarlo?,» 11 Octubre 2015. [En línea]. Available: https://styde.net/que-es-composer-y-como-usarlo/. |