Informando Things

Sebastian Pabon Lopez, *sebastian.pabon@pi.edu.co IEEE, Politécnico Internacional Bogotá D.C*

***Resumen*— Este Proyecto pretende desarrollar un programa que permita mostrar la información de los integrantes de la serie de Netflix Stranger Thing. Este proyecto se maneja con la arquitectura cliente servidor, la cual se implementará mediante una máquina virtual usando el sistema operativo UBUNTU, para el desarrollo usaremos el lenguaje de programación PHP y la base de datos MySQL.**

Palabras Clave: PHP, MySQL, PHP MY ADMIN, Maquina virtual

***Abstract*-- This project aims to develop a program that allows the information of the members of the Netflix series Stranger Thing to be displayed. This project is managed with the client server architecture, which will be implemented through a virtual machine using the UBUNTU operating system, for the development we will use the PHP programming language and the MySQL database.**

I. Introducción

Este proyecto pretende realizar el desarrollo de una aplicación que permitirá mostrar la información de los protagonistas de la serie de Netflix Stranger Things con el fin de dar a conocer a los fans una recopilación de todos los que forman parte de dicha serie. Para el desarrollo de este proyecto se crea un modelo que nos permite conocer la arquitectura de nuestro proyecto y guiándonos por esta misma, se realiza la instalación y configuración de nuestro servidor usando una máquina virtual manejada con el sistema operativo Linux Ubuntu. Una vez configurada la máquina virtual se realiza la instalación de MySQL y PHP MyAdmin para poder realizar el manejo de nuestra base de datos.

Fig. 1. Serie de Netflix Stranger Things.

II. Modelos y Arquitectura

Este proyecto usará la arquitectura de modelo de 3 capas que nos permitirá manipular nuestra base de datos por medio del servidor y manejar nuestra aplicación a través de una interfaz gráfica creada con PHP.

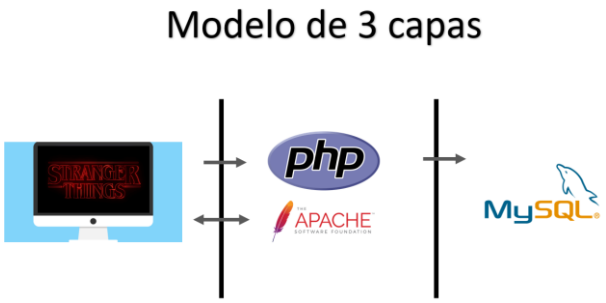


Fig. 2. Modelo de arquitectura

III. Instalación y Configuración de Maquina Virtual

a. Creamos una nueva máquina virtual en VirtualBox.

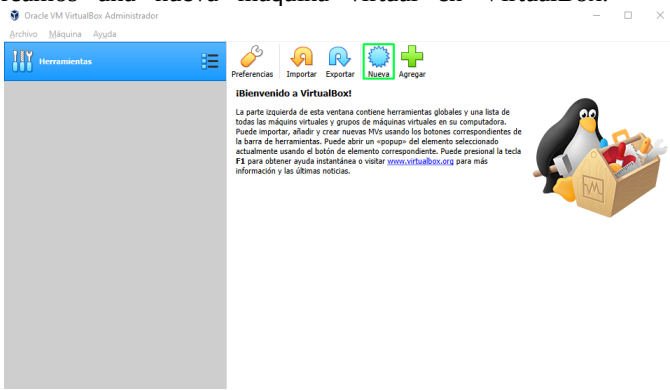


Fig. 3. Crear maquina

b. Le asignamos el nombre de nuestra maquina virutal y le asignamos el sistema operativo ubuntu de arquitectura

de 64 bits.

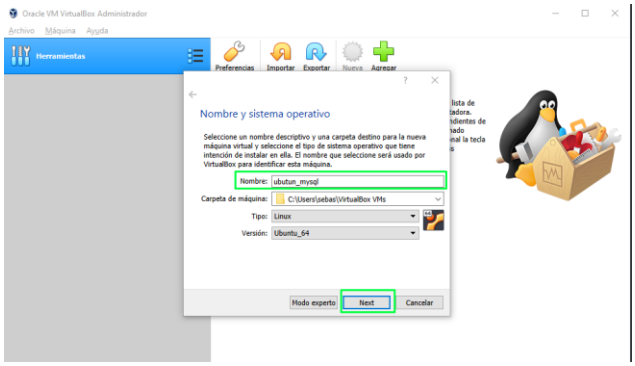


Fig. 4. Sistema operativo

c. Seleccionamos el tipo de disco duro de nuestra maquina virtual

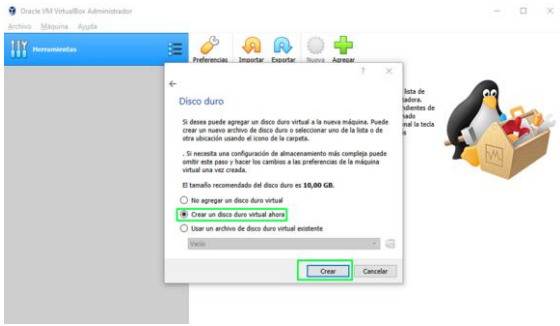


Fig. 5. Disco de la maquina

d. Seleccionamos el tipo de archivo de nuestro disco duro

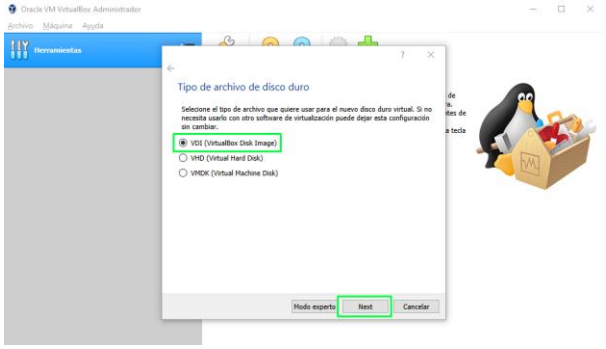


Fig. 6. Tipo de archivo

e. Seleccionamos el tipo de espacio que usara nuestro disco duro.

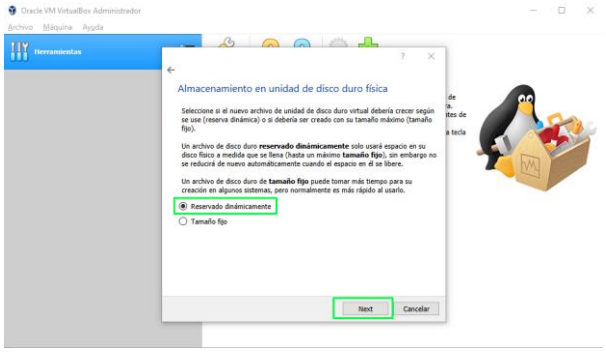


Fig. 7. Tipo de Espacio de la maquina

f. Le asignamos un tamaño a nuestro disco duro.

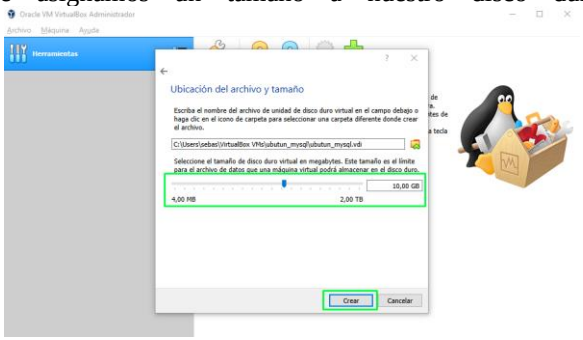


Fig. 8. Espacio de la maquina

g. Encendemos nuestra máquina virtual.

Fig. 9. Encendido maquina

h. Una vez prendida nuestra maquina seleccionamos la imagen que vamos a usar.

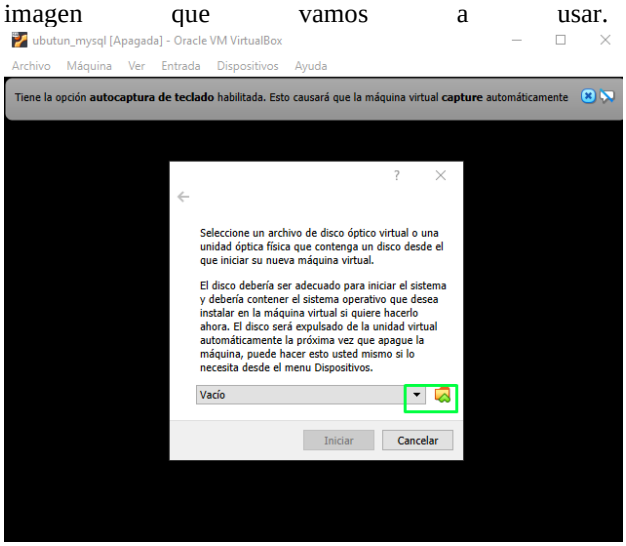


Fig. 10. Buscar sistema operativo

i. Iniciamos nuestra máquina virtual ya con el disco seleccionado.

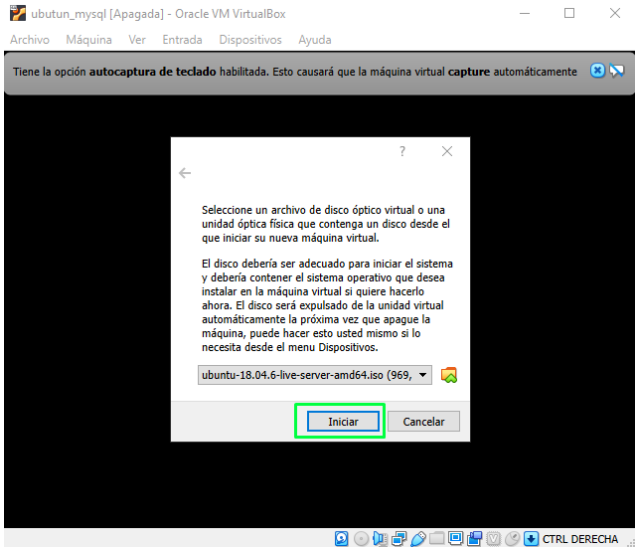


Fig. 11. Iniciar máquina virtual

j. Comenzará a iniciarse nuestro sistema operativo.

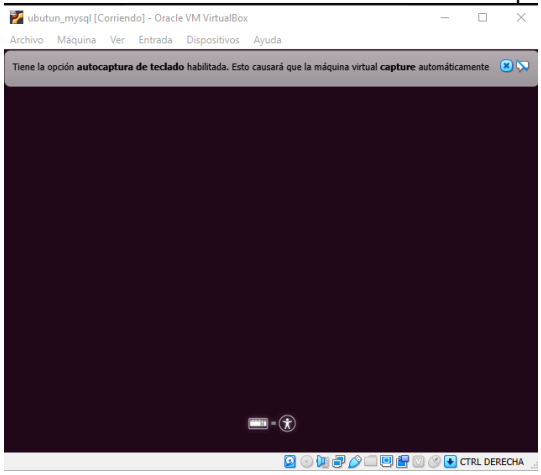


Fig. 12. Arranque de maquina

k. Comenzará hacer la configuración para la instalación.

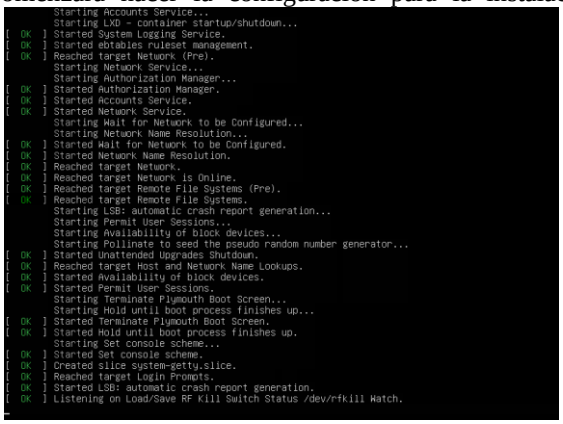


Fig. 13. Configuración de maquina

l. Una vez finalizada la configuración seleccionamos el idioma.

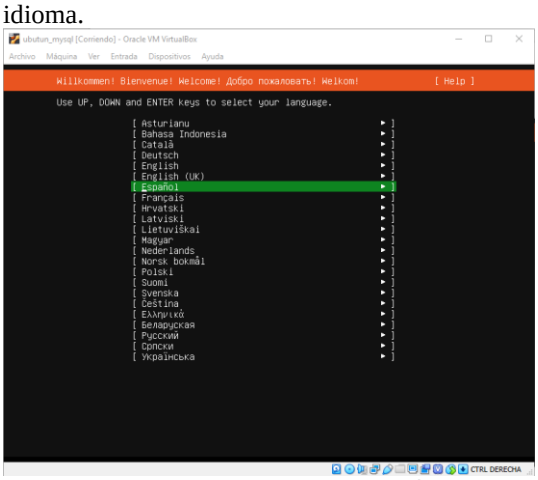


Fig. 14. Idioma de instalación

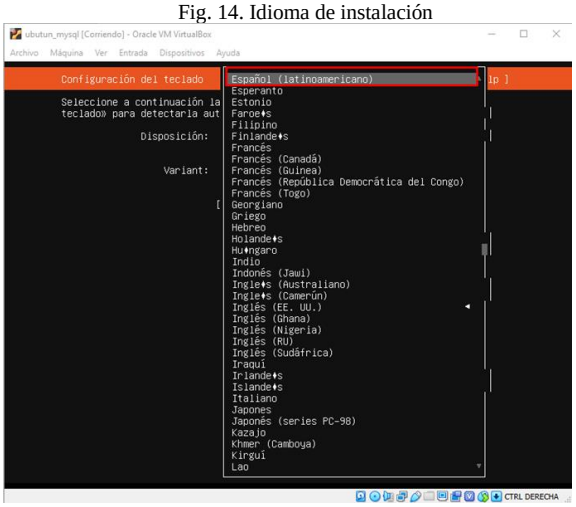


Fig. 15. Idioma de sistema operativo

m. Nos mostrara información de la conexión de red, lo dejamos por defecto y continuamos.

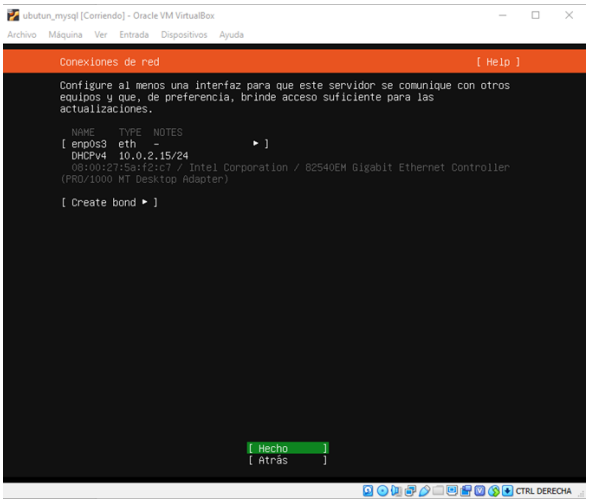


Fig. 16. Configuración de la red

n. Realizamos la configuración del proxy, lo configuramos y continuamos.

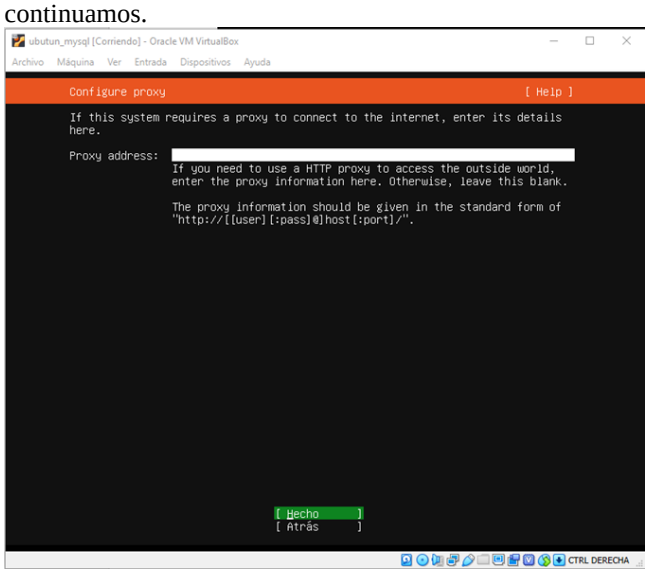


Fig. 17. Configuración del proxy

o. Nos muestra la configuración del archivo mirror, continuamos.

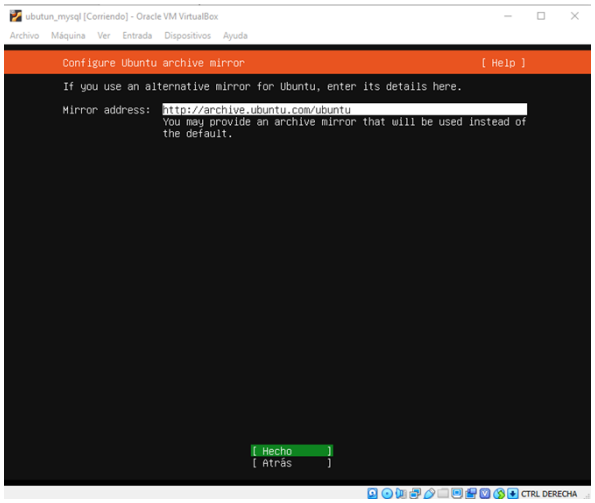


Fig. 18. Configuración del archivo mirror.

p. Configuracion de la configuración del almacenado.

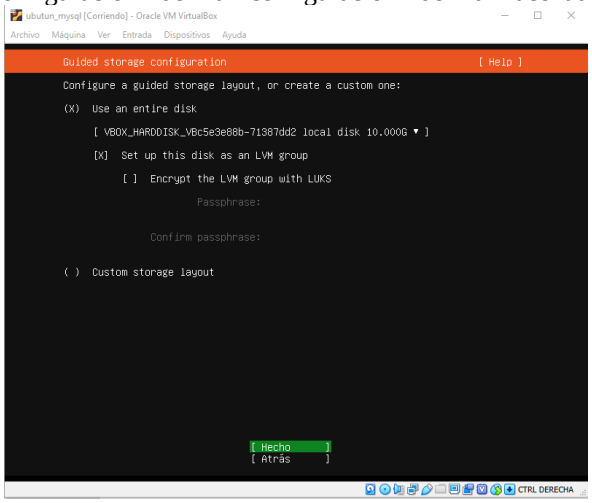


Fig. 19. Configuración de almacenado.

q. Configuración de almacenado del sistema, lo dejamos por defecto y continuamos.

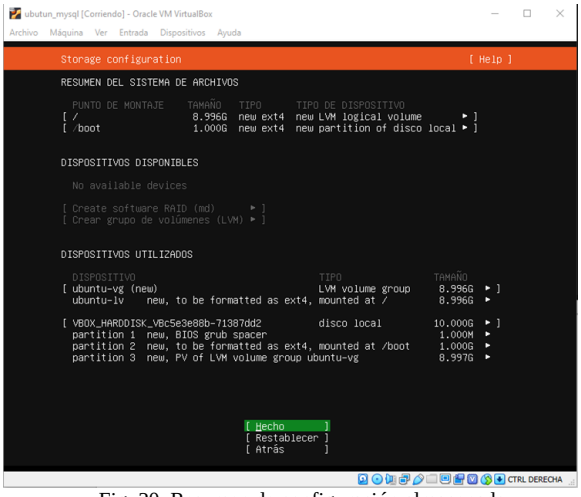


Fig. 20. Resumen de configuración almacenado.

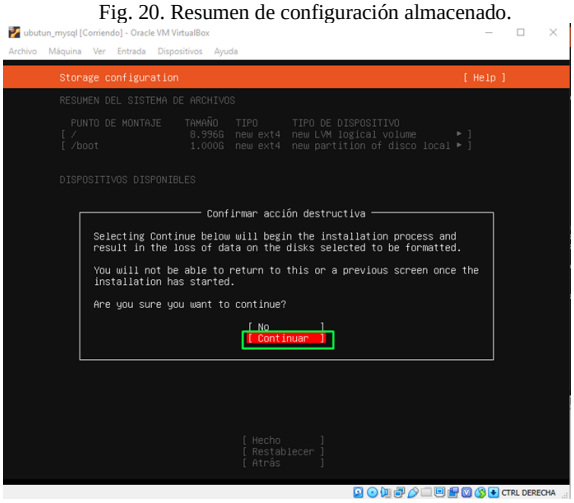


Fig. 21. Confirmación de configuración.

r. Configuramos nuestro perfil y la contraseña de acceso a nuestro servidor.

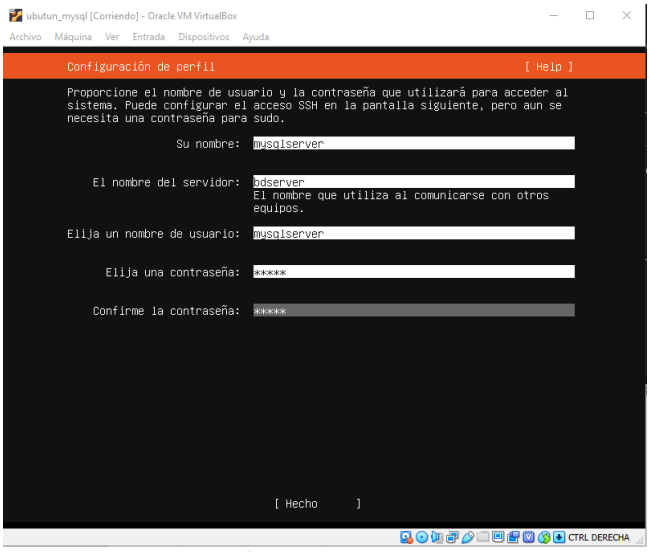


Fig. 22. Configuración de perfil y credenciales

s. Permitimos la configuración de SSH en nuestro servidor.

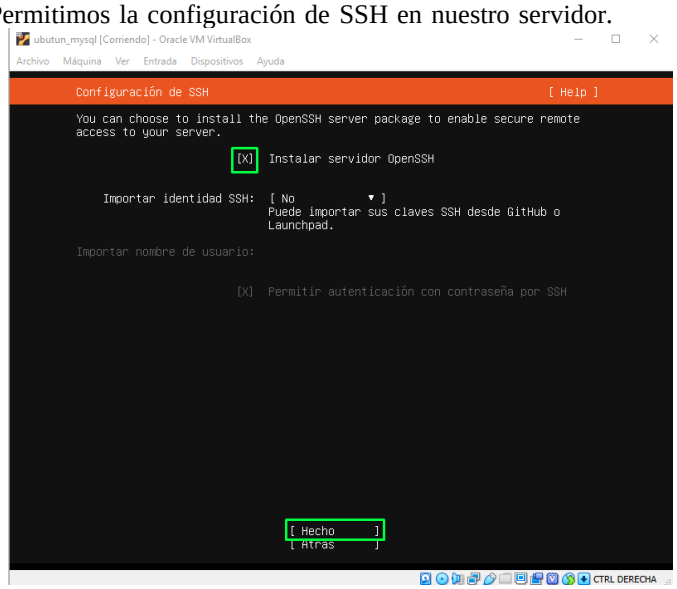


Fig. 23. Configuración de SSH

t. Comenzará la instalación de nuestro sistema operativo.

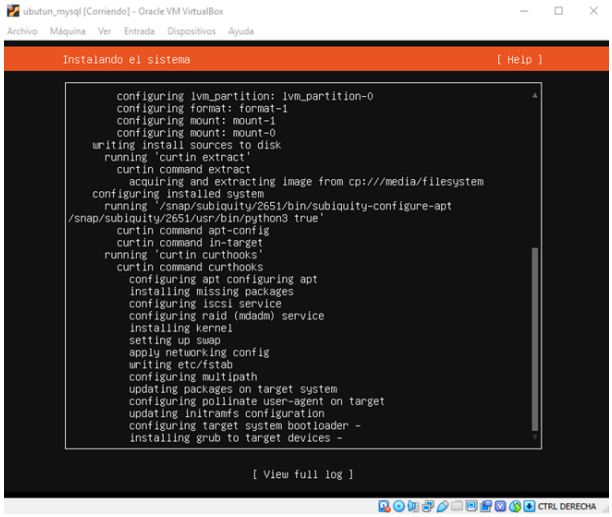


Fig. 24. Instalando Sistema Operativo

u. Una vez finalizada la instalación cerramos.

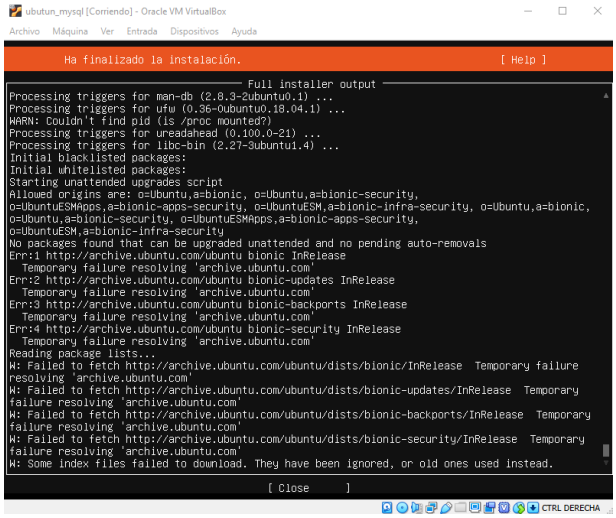


Fig. 25. Finalización del sistema operativo

v. Cuando termine la instalación de los otros paquetes reiniciamos nuestro sistema operativo.

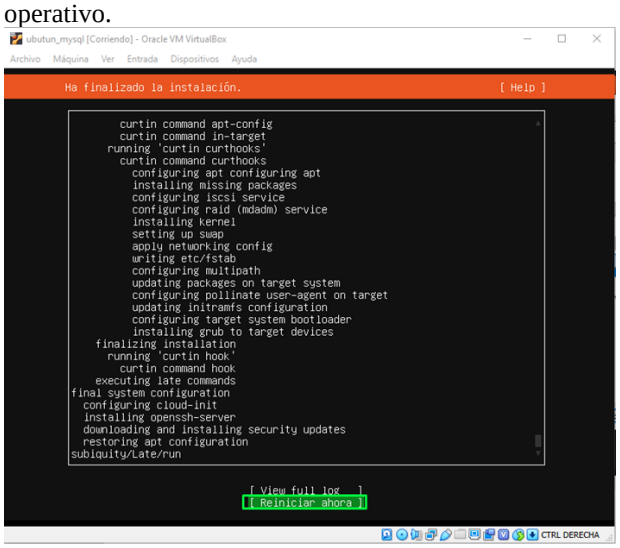


Fig. 26. Reinicio sistema operativo

w. Una vez reiniciada nuestra maquina ya nos mostrara la terminal para poder acceder y ejecutar comandos.

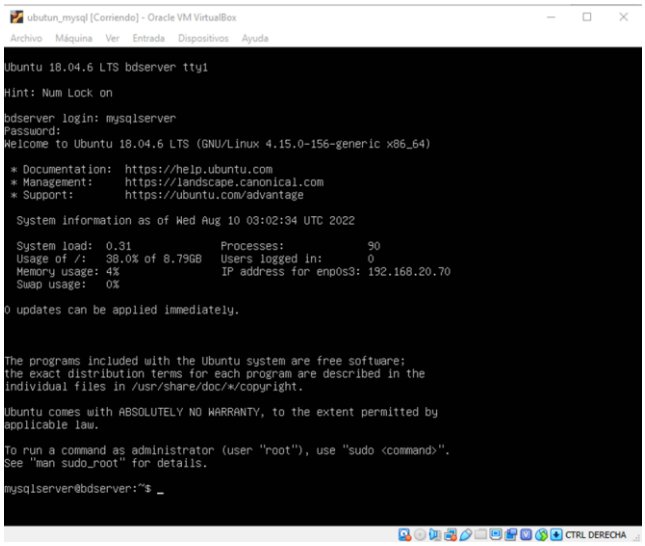


Fig. 27. Terminal de comandos

IV. CONFIGURACIÓN MOBAXTERM

a. Configurar la red del Sistema operativo.

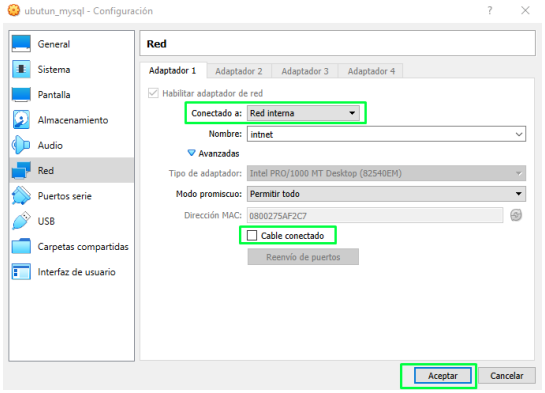


Fig. 28. Red del sistema Operativo

b. Consultar nuestra Ip de la máquina.

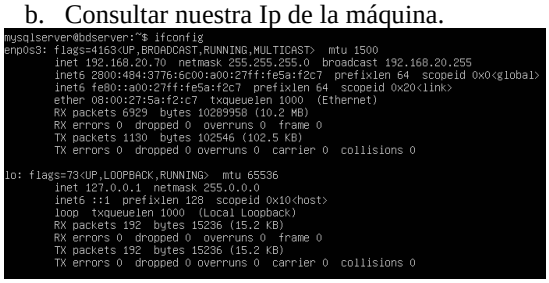


Fig. 29. Ip de maquina

c. Abrir MobaXterm y seleccionar session.

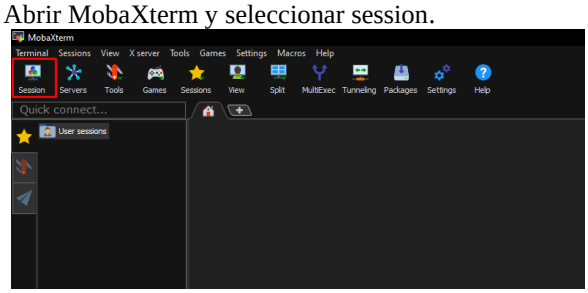


Fig. 30. Interfaz MobaXterm

d. Seleccionar SSH.

Fig. 31. SSH mobaXterm

e. Ingresamos la ip del servidor y tambien asignamos un usuario.

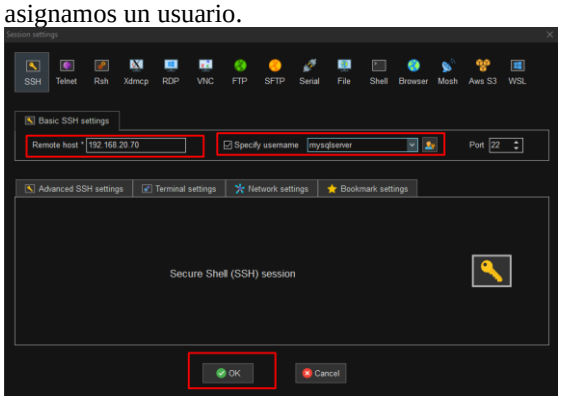


Fig. 32. Configuración servidor

f. Inicar al servidor con mobaXterm

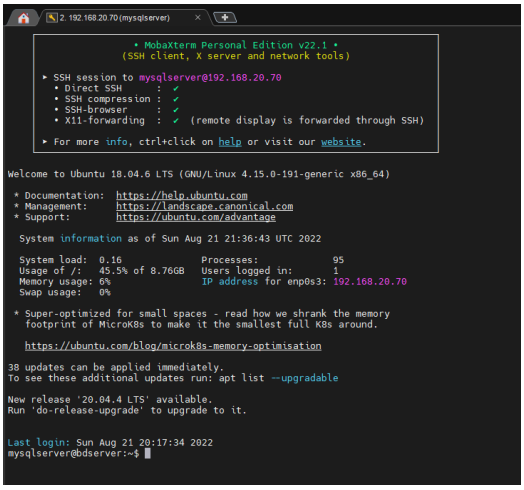


Fig. 33. Terminal del servidor

V. INSTALACIÓN MYSQL + APACHE + PHP

a. Ejecutar el comando de actualización sudo apt-get update.

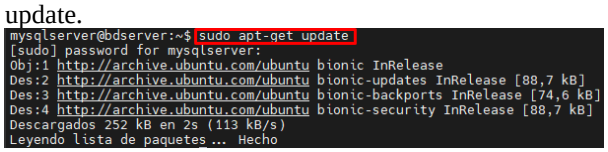


Fig. 34. Comando actualización

b. Instalación del servidor web sobre una solución apache, ejecutando el comando **sudo apt-get install apache2**.

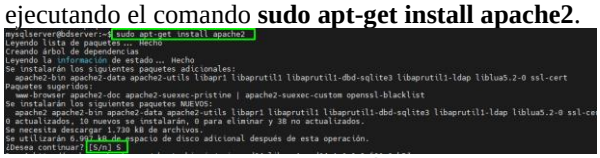


Fig. 35. Comando instalación apache

c. Habilitar el puerto 80 comando: sudo ufw allow 80

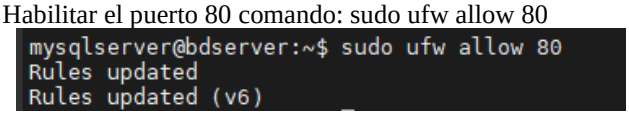


Fig. 36. Comando habilitador

d. Verificamos que nuestra maquina este funcionando, usando la ip de nuestro servidor.

Fig. 37. Página del servidor

e. Verificar el firewall de nuestro servidor con el comando: sudo ufw status.

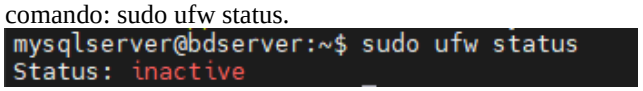


Fig. 38. Comando verificación firewall

f. Activar el apache full con el comando: sudo ufw allow in "Apache Full".

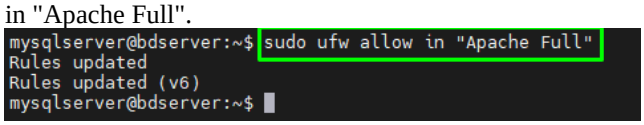


Fig. 39. Habilitar apache

g. Activar el servicio con el comando: sudo ufw enable.

```
mysqlserver@bdserver:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? Y
Firewall is active and enabled on system startup
mysqlserver@bdserver:~$
```

Fig. 40. Habilitar firewall

h. Realizar la instalación de Mysql con el comando: sudo apt-get install mysql-server.

```
mysqlserver@bdserver:~$ sudo apt-get install mysql-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libaio1 libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.
  libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl mysql-clie
Paquetes sugeridos:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca
Se instalarán los siguientes paquetes NUEVOS:
  libaio1 libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.
  libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl mysql-clie
0 actualizados, 21 nuevos se instalarán, 0 para eliminar y 38 no actualizados.
Se necesita descargar 20,0 MB de archivos.
Se utilizarán 157 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
```

Fig. 41. Comando instalación de MySQL

i. Ingresar a mysql con el comando: sudo mysql -u root -p

```
mysqlserver@bdserver:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.39-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Fig. 42. Terminal de Mysql

j. Realizamos la instalación de PHP con el comando: sudo apt-get install php libapache2-mod-php php-mysql -y.

```
mysqlserver@bdserver:~$ sudo apt-get install php libapache2-mod-php php-mysql -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libapache2-mod-php7.2 libsodium23 php-common php7.2 php7.2-cli php7.2-common php7.
Paquetes sugeridos:
  php-pear
```

Fig. 43. Comando instalación php

k. Probando el servidor de apache, ademas se habilitan permisos para la manipulación de archivos.

```
mysqlserver@bdserver:~$ cd /var/www/html
mysqlserver@bdserver:/var/www/html$ sudo chown -R $USER:root /var/www
mysqlserver@bdserver:/var/www/html$
```

Fig. 44. Habilitar permisos

l. Abrimos el editor de codigo nano que nos permite cambiar la información de un archivo, ejcutandolo con el comando: nano info.php

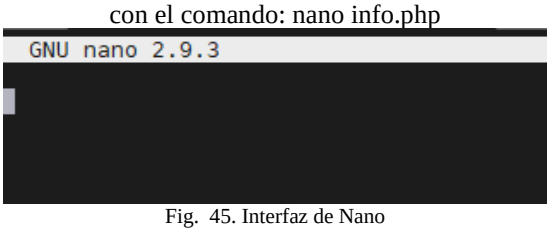


Fig. 45. Interfaz de Nano

m. Copiamos unos datos de prueba y digitamos las teclas ctrl + x para guardar los cambios.

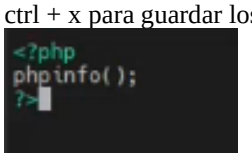


Fig. 46. Información de PHP

n. Ahora si verificamos la conexión con el servidor ya me muestra la versión que tengo instalada.

Fig. 47. Página de PHP

o. Realizamos ahora la instalación de phpmyadmin con el comando: sudo apt-get install phpmyadmin.

```
mysqlserver@bdserver:/var/www/html$ sudo apt-get install phpmyadmin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  dbconfig-common dbconfig-mysql fontconfig-config fonts-dejavu-core javascript-common
  libwebp6 libxpm4 libzip4 php-bz2 php-curl php-gd php-mbstring php-pear php-php-gettext
Paquetes sugeridos:
  libgd-tools php-libsodium php-mcrypt php-gmp php-imagick www-browser
Se instalarán los siguientes paquetes NUEVOS:
  dbconfig-common dbconfig-mysql fontconfig-config fonts-dejavu-core javascript-common
  libwebp6 libxpm4 libzip4 php-bz2 php-curl php-gd php-mbstring php-pear php-php-gettext
  phpmyadmin
```

Fig. 48. Habilitar permisos

p. Si genera error al ir a la ruta de phpmyadmin debemos ejecutar el comando: sudo nano /etc/apache2/apache2.conf

Fig. 49. Error PHP

q. Debemos ahora agregar la siguiente información: Include /etc/phpmyadmin/apache.conf

```
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
Include /etc/phpmyadmin/apache.conf
```

Fig. 50. Agregando configuración PHPMyAdmin

r. Luego de esto realizamos el reinicio con el comando: sudo service apache2 restart.

```
sudo service apache2 restart
```

Fig. 51. Reiniciando Apache

s. Verificamos que el servidor de phpMyadmin este funcionando correctamente.

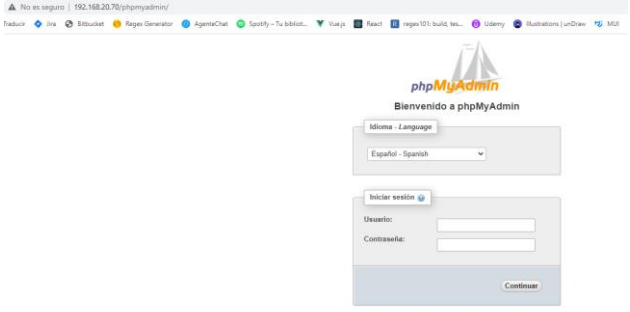


Fig. 51. Pagina PHPMyAdmin

t. Si nos deniega el ingreso, debemos entonces configurar el root.

Fig. 52. Error usuario MySQL

u. Ingresamos a mysql y debemos realizar la configuración del mysql y cambiar la autenticación

```
mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> update user set authentication_string=PASSWORD("12345") where user = 'ro
Query OK, 1 row affected, 1 warning (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 1
```

Fig. 53. Modificando autenticación MySQL

v. Ahora alteramos el usuario root para establecer permisos y registrar una clave.

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '12
Query OK, 0 rows affected (0,00 sec)
```

Fig. 54. Modificando usuario root

w. Ahora revisamos el pluging para verificar que temenos el usuario.

```
mysql> select user, plugin from user;
+------------------+-----------------------+
| user             | plugin                |
+------------------+-----------------------+
| root             | mysql_native_password |
| mysql.session    | mysql_native_password |
| mysql.sys        | mysql_native_password |
| debian-sys-maint | mysql_native_password |
| phpmyadmin       | mysql_native_password |
+------------------+-----------------------+
5 rows in set (0,00 sec)
```

Fig. 55. Comando pluging

x. Realizamos la actualización con el commando: update user set plugin='mysql_native_password' where user = 'root';

```
mysql> update user set plugin='mysql_native_password' where user = 'root';
Query OK, 0 rows affected (0,00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

Fig. 56. Comando actualización pluging

y. Ejecutamos los privilegios para que queden guardados con el comando: flush privileges;

```
mysql> flush privileges;
Query OK, 0 rows affected (0,00 sec)
```

Fig. 57. Ejecución de privilegios

z. Realizamos la verificación de que todo quedara configurado correctamente.

Fig. 58. Pagina Inicio phpMyAdmin

VI. DIAGRAMA DE CLASES

En este apartado encontramos la representación del modelo de clases, aca Podemos observar tres clases actor, mounstros y usuario. Encontramos diferentes propiedades con los tipos de datos string y int. Además de esto tenemos diferentes métodos que nos permitirán obtener la información de los actores o mounstros, modificar los actores, eliminar actores, también podemos interactuar con nuestro usuario con los métodos iniciar sesión y registrar un usuario.

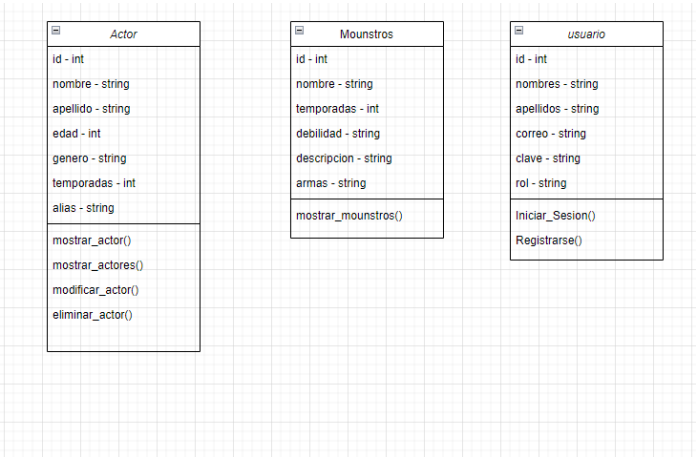


Fig. 59. Diagrama de clases

VII. DIAGRAMA DE SECUENCIA

Este diagrama muestra los procedimientos que realiza un aplicativo para poder obtener la información de los actores, mounstros, además de las interacciones con la base de datos y procedimientos como lo seria iniciar sesión y registrar un usuario.

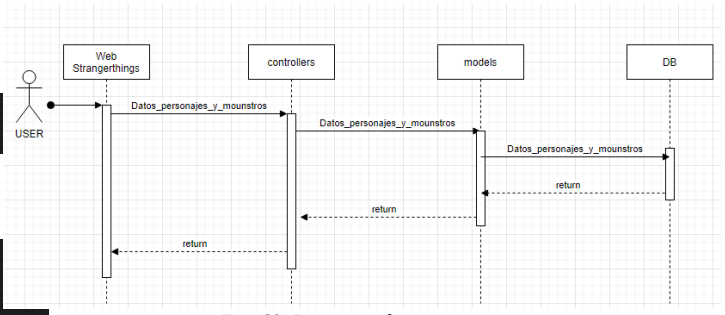


Fig. 60. Diagrama de secuencia

VIII. DICCIONARIO DE DATOS

En este apartado encontramos los diferentes datos que son usados en nuestro aplicativo, en el encontramos datos para el manejo de los actores, mounstros y interacciones con el usuario.

Nombre	Tipo	Archivo_de_inicialización	Descripción
connection	mysql_connection	helpers/connectionDB.php	Permite almacenar la variable conneccion a la base de datos
correo	string	controllers/procesar-inicio-sesion.php	Guarda el correo recibido del formulario de inicio de sesion
clave	string	controllers/procesar-inicio-sesion.php	Guarda la contraseña recibida del formulario de inicio de sesion
items	array	controllers/procesar-inicio-sesion.php	Guarda el objeto Usuario que contiene la información del usuario
$_SESSION["idSession"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el identificador de la sesion
$_SESSION["nombre"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el nombre de la sesion
$_SESSION["correo"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el correo de la sesion
$_SESSION["rol"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el rol de la sesion
nombre	string	controllers/procesar_registro.php	nombre recibido del formulario de registro
apellido	string	controllers/procesar_registro.php	apellido recibido del formulario de registro
correo	string	controllers/procesar_registro.php	correo recibido del formulario de registro
clave	string	controllers/procesar_registro.php	clave recibida del formulario de registro
confirmar	string	controllers/procesar_registro.php	clave nuevamente escrita recibida del formulario de registro
id	int	class/Usuario.php	Identificador del usuario
nombre	string	class/Usuario.php	nombre de la clase Usuario
apellido	string	class/Usuario.php	apellido de la clase Usuario
correo	string	class/Usuario.php	correo de la clase Usuario
rol	string	class/Usuario.php	rol de la clase Usuario
id	int	class/Mounstro.php	identificador de la clase Mounstro
nombre	string	class/Mounstro.php	nombre de la clase Mounstro
debilidad	string	class/Mounstro.php	debilidad de la clase Mounstro
aparicion	int	class/Mounstro.php	aparicion de la clase Mounstro
armas	string	class/Mounstro.php	armas de la clase Mounstro
descripcion	string	class/Mounstro.php	descripcion de la clase Mounstro
imagen	string	class/Mounstro.php	imagen de la clase Mounstro

Fig. 61. Diccionario de datos

IX. DIAGRAMA DE SECUENCIA

CLASE CONNECCIÓN: Permite manejar la conexión con la base de datos.

```php
<?php
class Conneccion{
    public $connection;

    function get_connection(){
        return $this->connection;
    }

    function set_connection($connection) {
        $this->connection = $connection;
    }

    function connectToDB(){
        $this->set_connection(mysqli_connect("localhost","root","12345","strangerthings"));
        if (!$this->get_connection()) {
            echo "Conneccion erronea";
        }else{
            echo "Conneccion establecida";
        }
    }
}
```

Fig. 62. Clase coneccion

CLASE ACTOR: Contiene toda la información del actor además nos permite obtener todos los actores de la serie y esta se hereda de la clase Conneccion para ejecutar las consultas.

```php
<?php
    //Permite importar la clase Connection
    require_once __DIR__.'/../helpers/connectionDB.php';

    /**
    * La clase Actor contiene toda la información relacionada con los actores y sus papeles en la serie
    * Este clase Hereda de Conneccion que permite realizar la conexión con base de datos
    */
    class Actor extends Conneccion{
        //Propiedades de la clase actor
        public $id, $nombre, $apellido, $edad, $genero, $numeroTemporadas, $personaje, $rol, $imagen, $descripcion;

        //Permite modificar y obtener el id
        function set_id($id) {
            $this->id = $id;
        }

        function get_id() {
            return $this->id;
        }

        //Permite modificar y obtener el nombre
        function set_nombre($nombre) {
            $this->nombre = $nombre;
        }

        function get_nombre() {
            return $this->nombre;
        }

        //Permite modificar y obtener el apellido
        function set_apellido($apellido) {
            $this->apellido = $apellido;
        }

        function get_apellido() {
            return $this->apellido;
        }
```

Fig. 63. Clase actor

```php
//Atributo para obtener todos los actores
public function obtener_todos(){
    $items = [];
    try{
        $consulta = 'SELECT * FROM actores';

        $this->connectToDB();
        $datos = mysqli_query($this->get_connection(), $consulta);
        if ($datos) {
            while ($fila = mysqli_fetch_array($datos)) {
                $item = new Actor();
                $item->id = $fila['id'];
                $item->nombre = $fila['nombre'];
                $item->apellido = $fila['apellido'];
                $item->edad = $fila['edad'];
                $item->genero = $fila['genero'];
                $item->numeroTemporadas = $fila['temporadas'];
                $item->personaje = $fila['personaje'];
                $item->rol = $fila['rol'];
                $item->imagen = $fila['imagen'];
                $item->descripcion = $fila['descripcion'];

                array_push($items, $item);
            }
        }else{
            echo "ha ocurrido un problema";
        }

        return $items;
    }catch(PDOException $e){
        return [];
    }
}
```

Fig. 64. Atributo obtener_todos los actores

CLASE MOUNSTRO: Nos permite obtener la información de los actores y esta se hereda de la clase Conneccion para ejecutar las consultas.

```php
//Permite importar la clase Connection
require_once __DIR__.'/../helpers/connectionDB.php';

/**
* La clase Mounstro contiene toda la información relacionada con los mounstros de la serie
* Este clase Hereda de Conneccion que permite realizar la conexión con base de datos
*/
class Mounstro extends Conneccion{
    //Propiedades de la clase actor
    public $id, $nombre, $debilidad, $aparicion, $armas, $descripcion, $imagen;

    //Permite modificar y obtener el id
    function set_id($id) {
        $this->id = $id;
    }

    function get_id() {
        return $this->id;
    }

    //Permite modificar y obtener el nombre
    function set_nombre($nombre) {
        $this->nombre = $nombre;
    }

    function get_nombre() {
        return $this->nombre;
    }
```

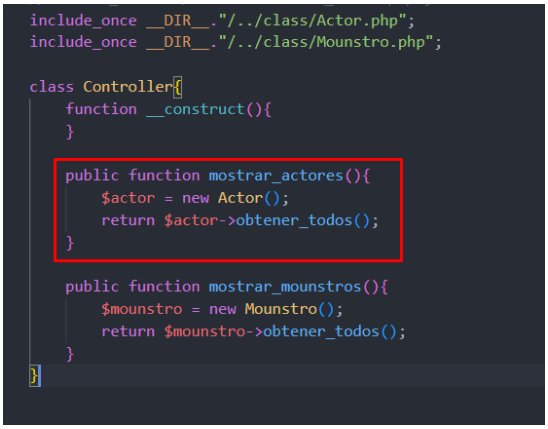
Fig. 65. Clase Mounstro

```php
//Atributo para obtener todos los actores
public function obtener_todos(){
    $items = [];
    try{
        $consulta = 'SELECT * FROM mounstros';

        $this->connectToDB();
        $datos = mysqli_query($this->get_connection(), $consulta);
        if ($datos) {
            while ($fila = mysqli_fetch_array($datos)) {
                $item = new Mounstro();
                $item->id = $fila['id'];
                $item->nombre = $fila['nombre'];
                $item->debilidad = $fila['debilidad'];
                $item->aparicion = $fila['aparicion'];
                $item->armas = $fila['armas'];
                $item->descripcion = $fila['descripcion'];
                $item->imagen = $fila['imagen'];

                array_push($items, $item);
            }
        }else{
            echo "ha ocurrido un problema";
        }

        return $items;
    }catch(PDOException $e){
        return [];
    }
}
```

Fig. 66. Atributo obtener_todos los mounstros

CREACIÓN DE OBJETOS

Creamos un nuevo objeto que deriba de la clase actor y mounstro.



```php
include_once __DIR__."/../class/Actor.php";
include_once __DIR__."/../class/Mounstro.php";

class Controller{
    function __construct(){
    }

    public function mostrar_actores(){
        $actor = new Actor();
        return $actor->obtener_todos();
    }

    public function mostrar_mounstros(){
        $mounstro = new Mounstro();
        return $mounstro->obtener_todos();
    }
}
```

Fig. 67. Clase controller

REFERENCIAS

[1] Gutierrez, D. (2011). Casos de uso Diagramas de Casos de Uso. Gutierrez, Demián, 1, 45.

[2] Bahit, E. (2011). Poo y mvc en php. El paradigma de la Programación.