

# Informando Things



Sebastian Pabon Lopez, [sebastian.pabon@pi.edu.co](mailto: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.

### Modelo de 3 capas



Fig. 2. Modelo de arquitectura

## III. INSTALACIÓN Y CONFIGURACIÓN DE MAQUINA VIRTUAL

### a. Creamos una nueva máquina virtual en VirtualBox.

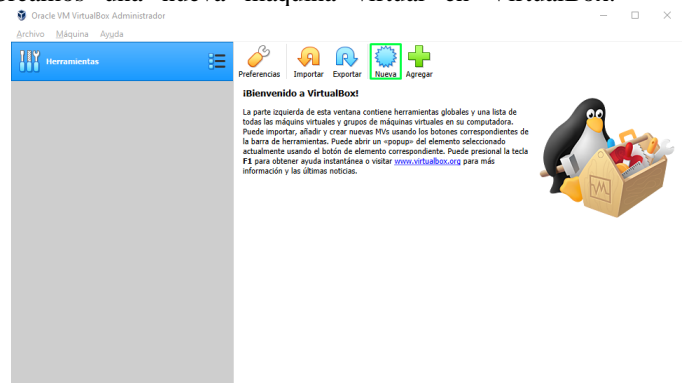


Fig. 3. Crear maquina

- b. Le asignamos el nombre de nuestra maquina virtual y le asignamos el sistema operativo ubuntu de arquitectura de 64 bits.

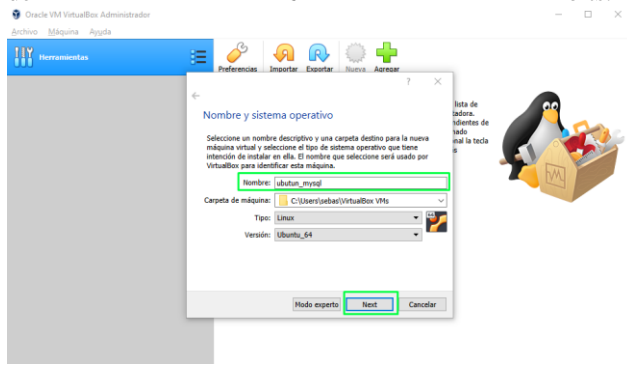


Fig. 4. Sistema operativo

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

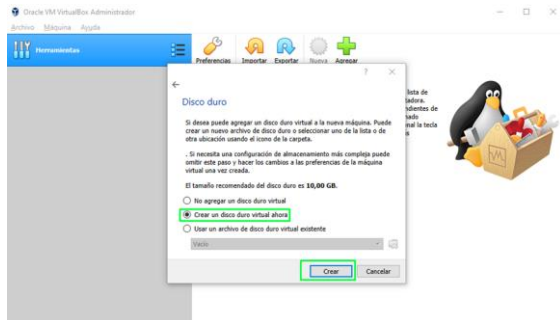


Fig. 5. Disco de la maquina

- d. Seleccionamos el tipo de archivo de nuestro disco duro

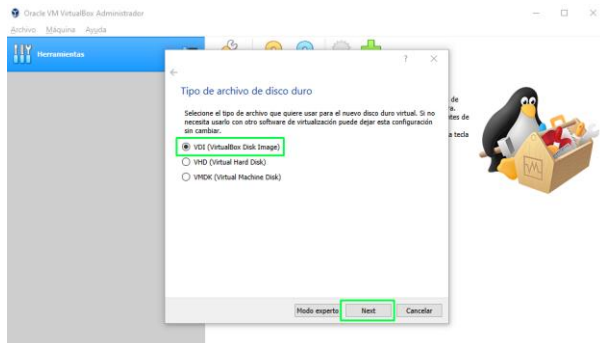


Fig. 6. Tipo de archivo

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

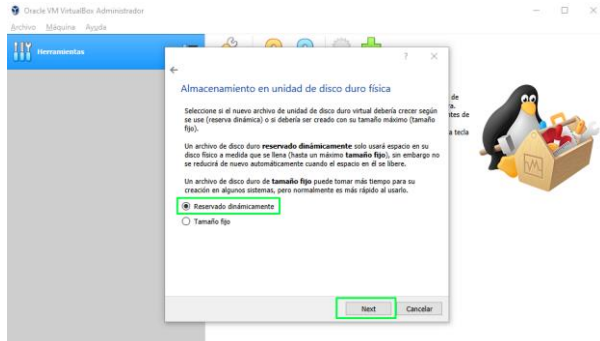


Fig. 7. Tipo de Espacio de la maquina

- f. Le asignamos un tamaño a nuestro disco duro.

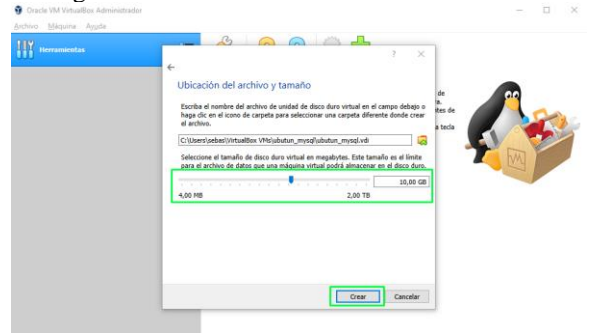


Fig. 8. Espacio de la maquina

- g. Encendemos nuestra máquina virtual.

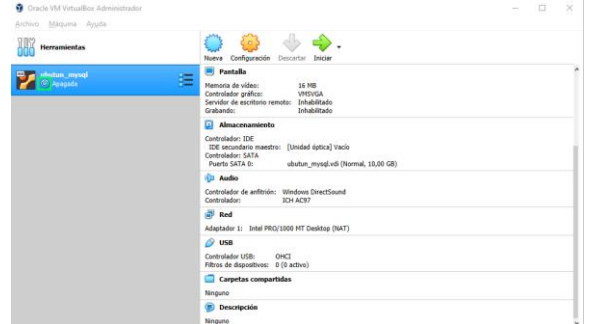


Fig. 9. Encendido maquina

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

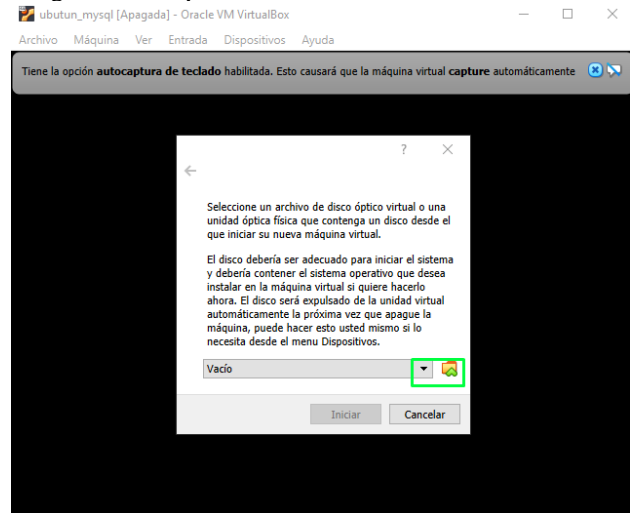


Fig. 10. Buscar sistema operativo

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

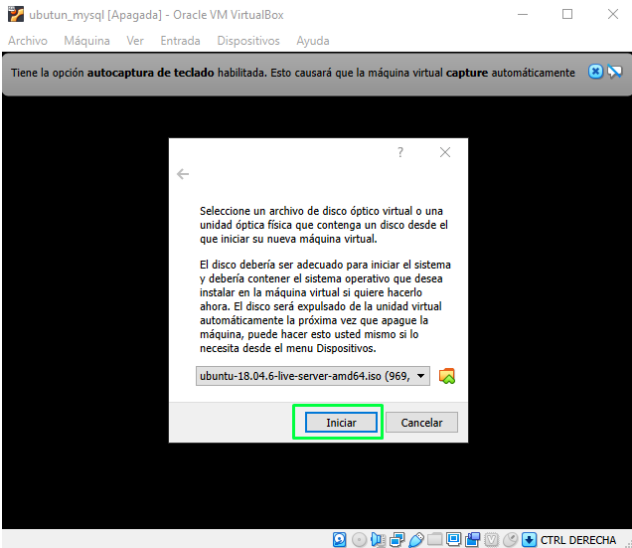


Fig. 11. Iniciar máquina virtual

j. Comenzará a iniciarse nuestro sistema operativo.

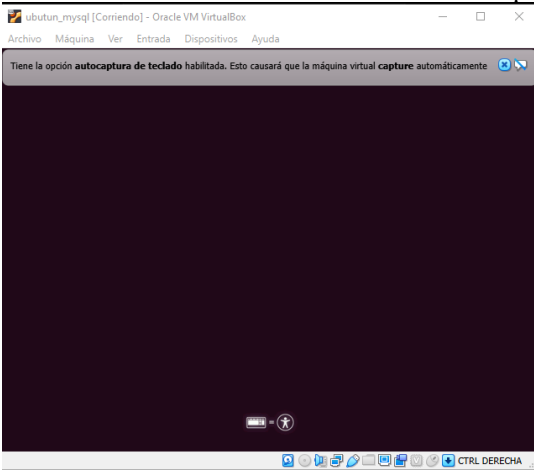


Fig. 12. Arranque de maquina

k. Comenzará hacer la configuración para la instalación.

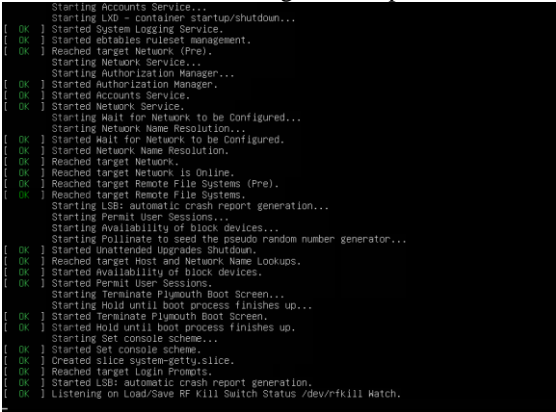


Fig. 13. Configuración de maquina

l. Una vez finalizada la configuración seleccionamos el idioma.

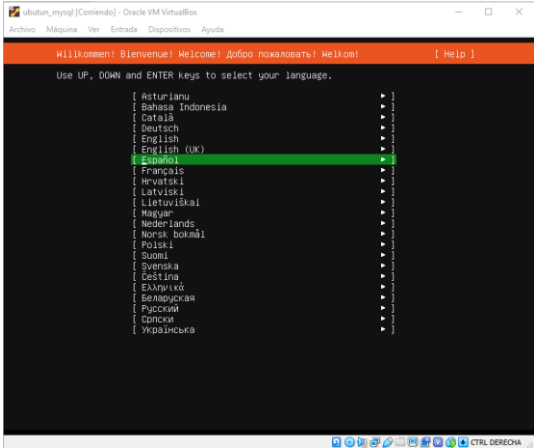


Fig. 14. Idioma de instalación

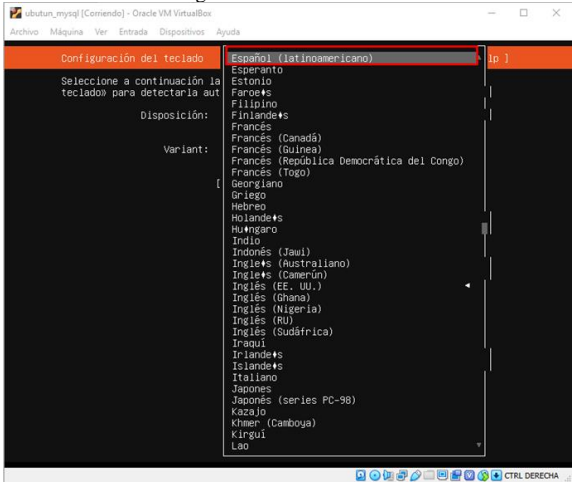


Fig. 15. Idioma de sistema operativo

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

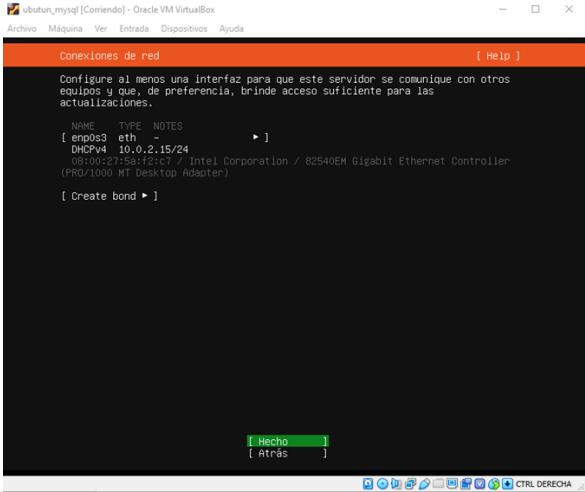


Fig. 16. Configuración de la red

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

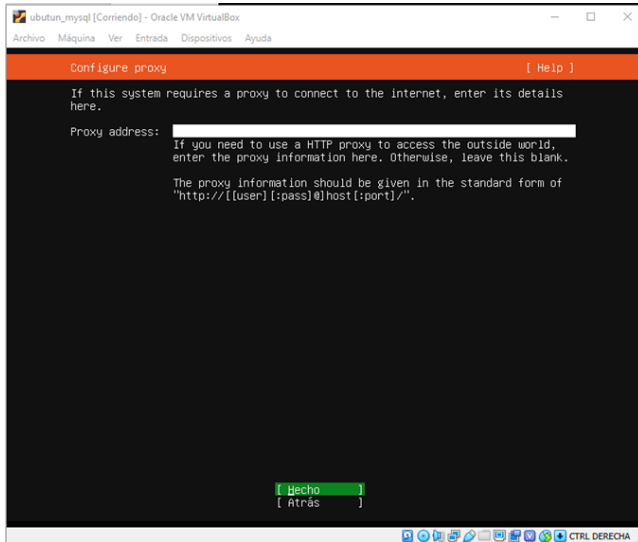


Fig. 17. Configuración del proxy

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

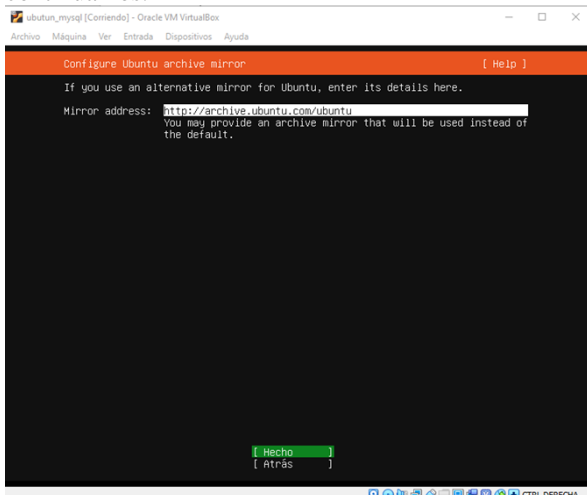


Fig. 18. Configuración del archivo mirror.

p. Configuración de la configuración del almacenado.

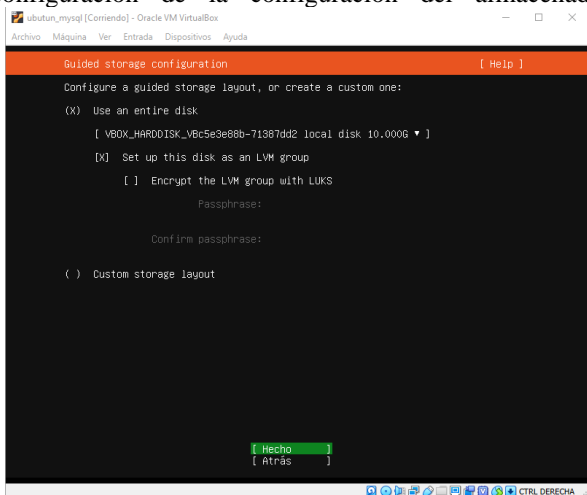


Fig. 19. Configuración de almacenado.

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

continuamos.

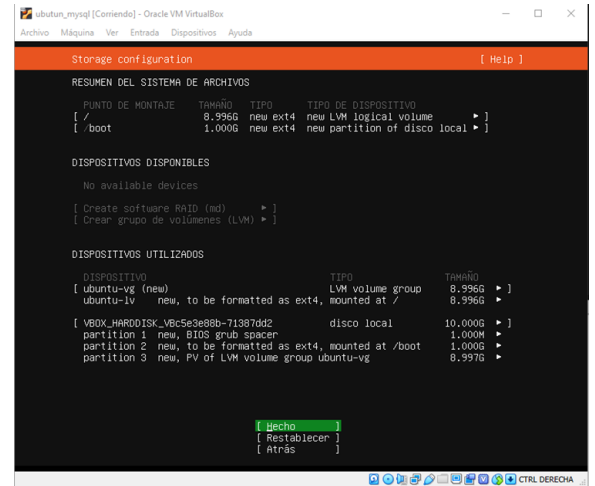


Fig. 20. Resumen de configuración almacenado.

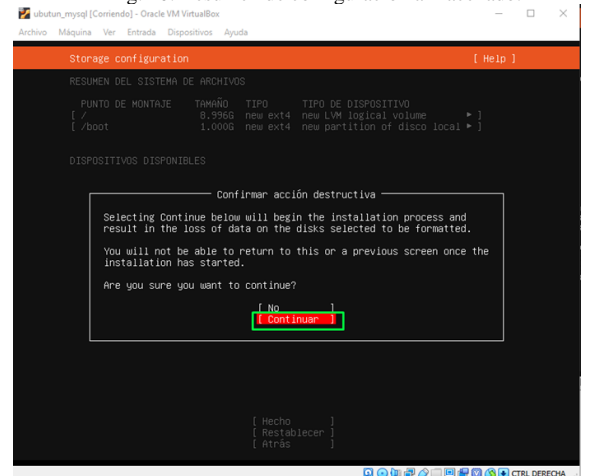


Fig. 21. Confirmación de configuración.

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

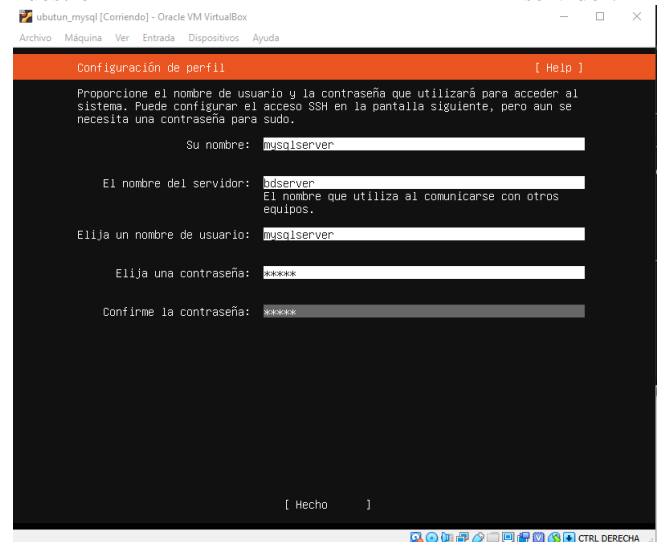


Fig. 22. Configuración de perfil y credenciales

s. Permitimos la configuración de SSH en nuestro servidor.

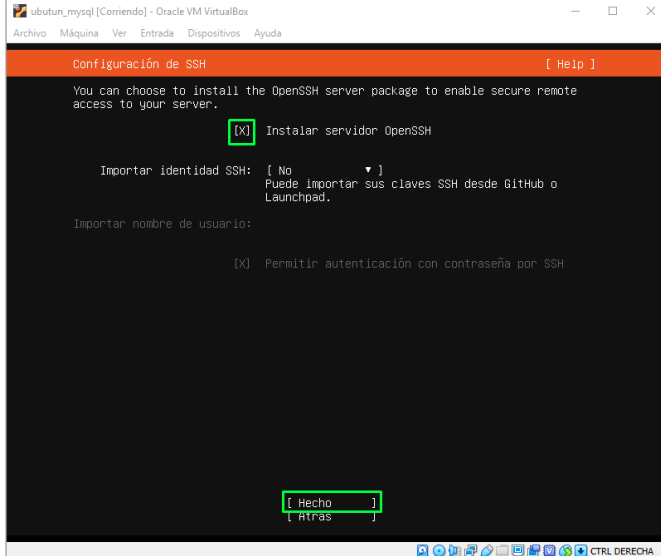


Fig. 23. Configuración de SSH

t. Comenzará la instalación de nuestro sistema operativo.

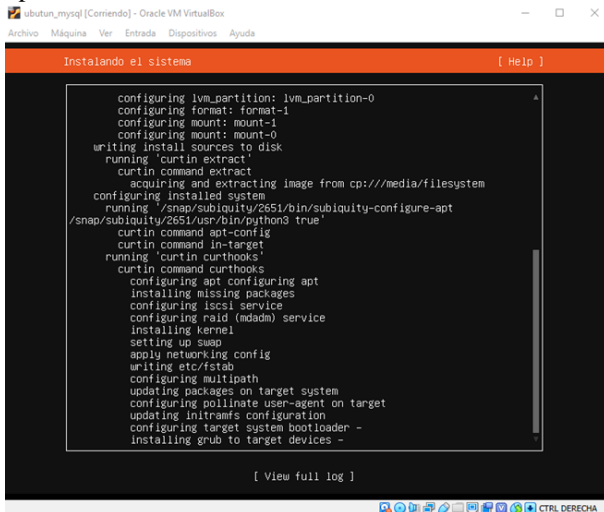


Fig. 24. Instalando Sistema Operativo

u. Una vez finalizada la instalación cerramos.

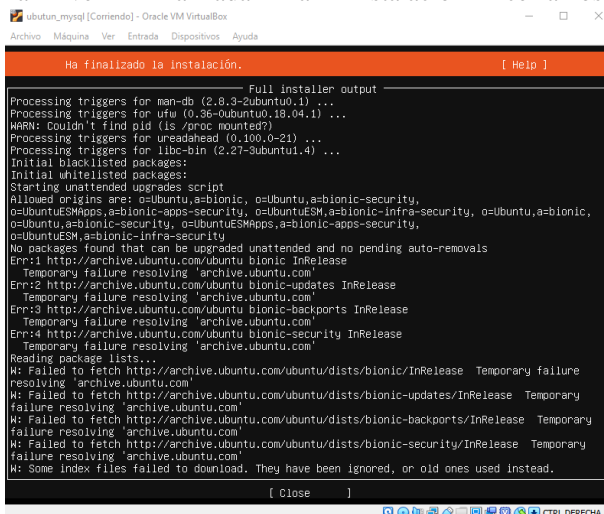


Fig. 25. Finalización del sistema operativo

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

operativo.

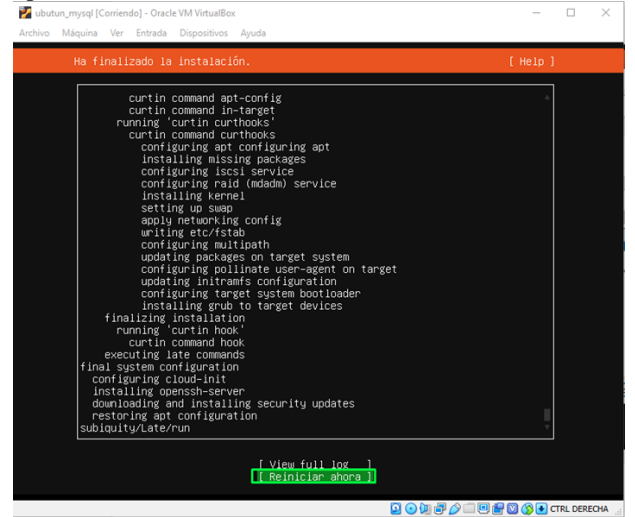


Fig. 26. Reinicio sistema operativo

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

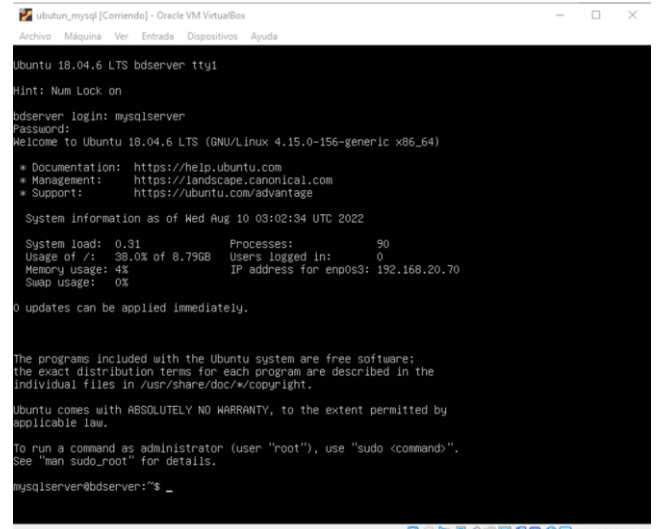


Fig. 27. Terminal de comandos

#### IV. CONFIGURACIÓN MOBAXTERM

a. Configurar la red del Sistema operativo.

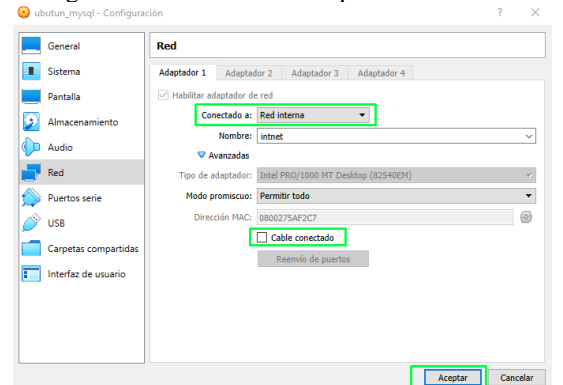


Fig. 28. Red del sistema Operativo



b. Consultar nuestra Ip de la máquina.

```
mysqlserver@bdsrver:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.20.70 netmask 255.255.255.0 broadcast 192.168.20.255
    inet6 2800::484:3776:6c00:a00:27ff:fe5a:f2c7 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::a00:27ff:fe5a:f2c7 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:5a:f2:c7 txqueuelen 1000 (Ethernet)
    RX packets 6929 bytes 10289958 (10.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1130 bytes 102546 (102.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 192 bytes 15236 (15.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 192 bytes 15236 (15.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig. 29. Ip de maquina

c. Abrir MobaXterm y seleccionar session.

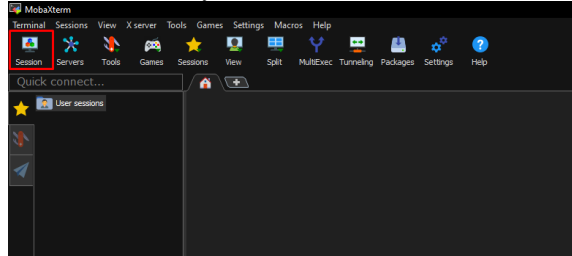


Fig. 30. Interfaz MobaXterm

d. Seleccionar SSH.

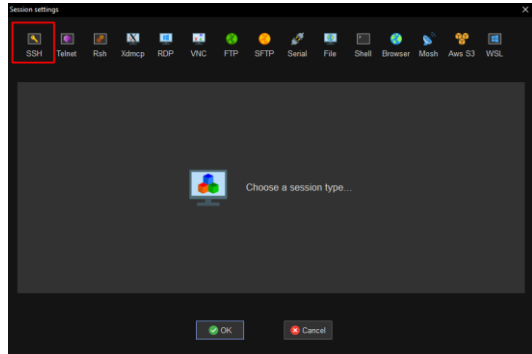


Fig. 31. SSH mobaXterm

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

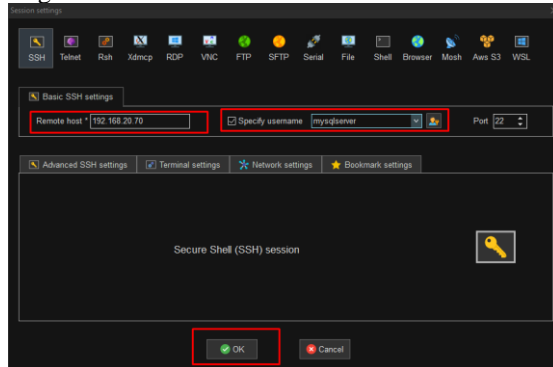


Fig. 32. Configuración servidor

f. Inicar al servidor con mobaXterm

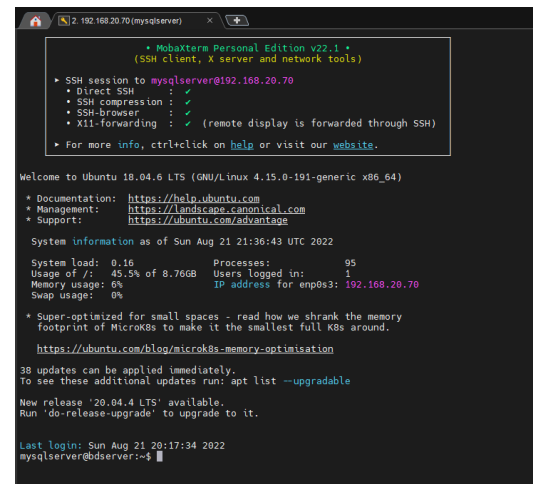


Fig. 33. Terminal del servidor

## V. INSTALACIÓN MYSQL + APACHE + PHP

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

```
mysqlserver@bdsrver:~$ sudo apt-get update
[sudo] password for mysqlserver:
Obj:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Des:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Des:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Descargados 252 kB en 2s (113 kB/s)
Leyendo lista de paquetes ... Hecho
```

Fig. 34. Comando actualización

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

```
mysqlserver@bdsrver:~$ sudo apt-get install apache2
Leyendo lista de paquetes ... Hecho
Creando árbol de dependencias
Leyendo la información de estado ... Hecho
Se instalarán los siguientes paquetes adicionales:
apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
Paquetes sugeridos:
www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
Se instalarán los siguientes paquetes MEJORES:
apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-c
0 actualizados, 10 nuevos se instalarán, 0 para eliminar y 38 no actualizados.
Se necesitarán 6.982 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] Y
```

Fig. 35. Comando instalación apache

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

```
mysqlserver@bdsrver:~$ sudo ufw allow 80
Rules updated
Rules updated (v6)
```

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**.

```
mysqlserver@bdsrver:~$ sudo ufw status
Status: inactive
```

Fig. 38. Comando verificación firewall

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

```
mysqlserver@bdsrver:~$ sudo ufw allow in "Apache Full"
Rules updated
Rules updated (v6)
mysqlserver@bdsrver:~$
```

Fig. 39. Habilitar apache

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

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

Fig. 40. Habilitar firewall

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

```
mysqlserver@bdserv:~$ 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@bdserv:~$ 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@bdserv:~$ 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.
  php-pear
Paquetes sugeridos:
```

Fig. 43. Comando instalación php

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

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

Fig. 44. Habilitar permisos

- l. Abrimos el editor de código nano que nos permite cambiar la información de un archivo, ejecutandolo con el comando: `nano info.php`

```
GNU nano 2.9.3

```

Fig. 45. Interfaz de Nano

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

```
<?php
phpinfo();
?>
```

Fig. 46. Información de PHP

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

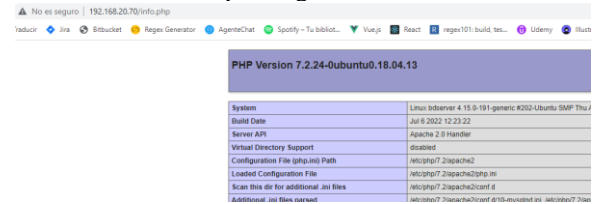


Fig. 47. Página de PHP

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

```
mysqlserver@bdserv:/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-gettex
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-gettex
  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`

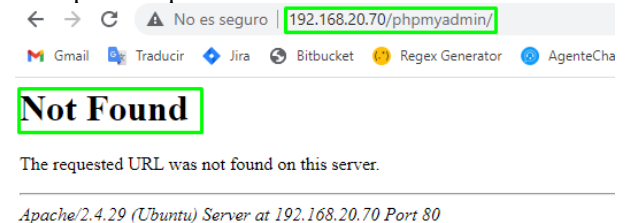


Fig. 49. Error PHP

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

```
LogFormat "%v:%p %h %l %u %t \"%r\" \"%s\" %0 \"%{Referer}i\" \"%{User-Agent}i\" \"%s\" %0" combined
LogFormat "%h %l %u %t \"%r\" \"%s\" %0 \"%{Referer}i\" \"%{User-Agent}i\" \"%s\" %0" combined
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.

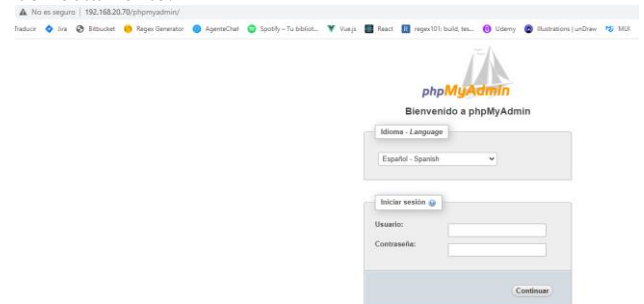


Fig. 51. Pagina PHPMyAdmin

- 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 = 'root'
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 '12345'
Query OK, 0 rows affected (0,00 sec)
```

Fig. 54. Modificando usuario root

- w. Ahora revisamos el plugin para verificar que tenemos 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 plugin

- x. Realizamos la actualización con el comando: 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 plugin

- 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.

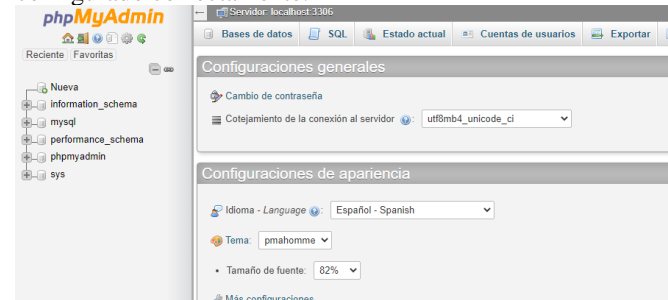


Fig. 58. Pagina Inicio phpMyAdmin

## 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.

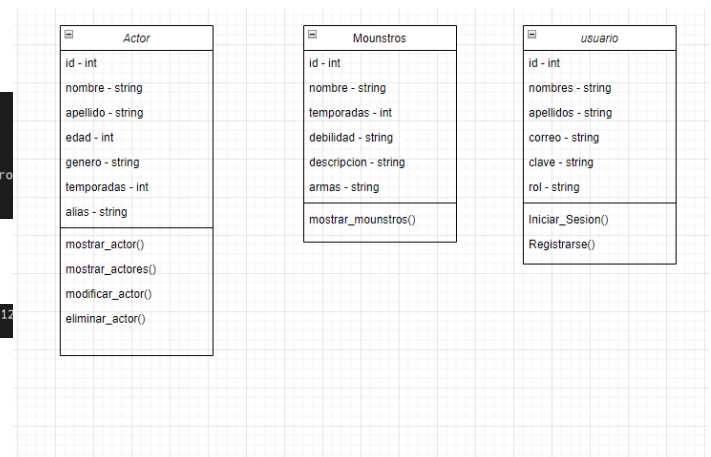


Fig. 59. Diagrama de clases

## 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.

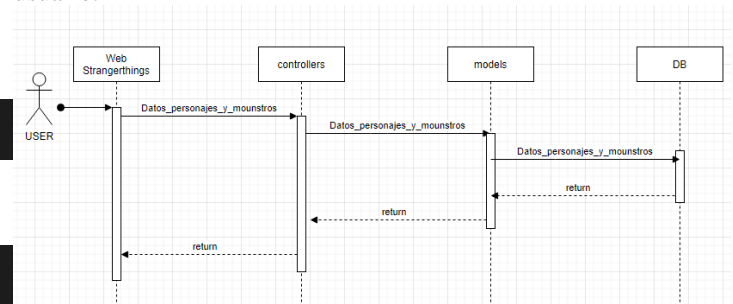


Fig. 60. Diagrama de secuencia



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 conexión a la base de datos
correo	string	controllers/procesar-inicio-sesion.php	Guarda el correo recibido del formulario de inicio de sesión
clave	string	controllers/procesar-inicio-sesion.php	Guarda la contraseña recibida del formulario de inicio de sesión
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 sesión
\$SESSION["nombre"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el nombre de la sesión
\$SESSION["correo"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el correo de la sesión
\$SESSION["rol"]	string - session	controllers/procesar-inicio-sesion.php	Guarda el rol de la sesión
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. CODIGO DEL PROYECTO

**ESTRUCTURA DE CARPETAS:** En estas diferentes carpetas encontramos los diversos códigos distribuidos de la siguiente forma.

- **Assets:** Archivos estáticos del proyecto como imágenes, archivos CSS y JavaScript.
- **Class:** Contiene todas las clases de nuestro programa.
- **Controllers:** Contiene el archivo que controla las clases, además de el manejo de la sesión.
- **Helpers:** Contiene las clases o funciones que pueden ser útiles en varias partes del código.
- **Views:** Contiene todas las vistas HTML del programa.
- **Index.php:** Este archivo inicia nuestro proyecto redirigiendo a la vista inicial home.

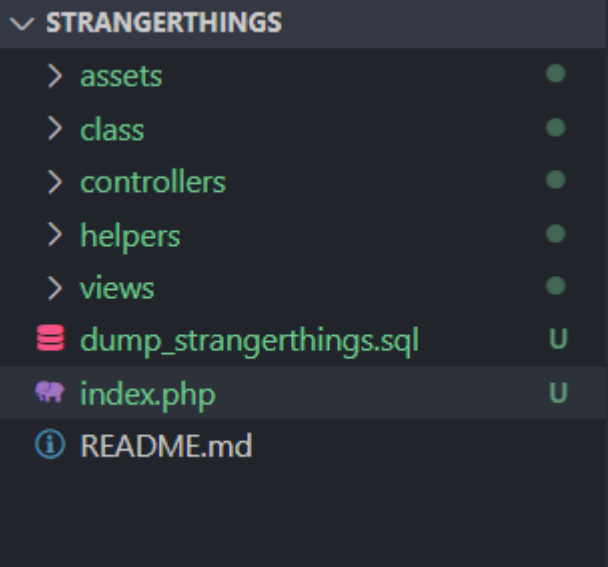


Fig. 62. Estructura de carpetas

ARCHIVO INDEX.PHP

Este archivo permite iniciar el proyecto y redirigirlo a la página home.php

```
index.php
1 <?php
2 | require_once 'views/home.php';
3 ?>
```

Fig. 63. Diagrama de clases

LOGICA INICIO DE SESION

Las páginas home, mounstros y personajes, en el Header tienen una lógica que verifica si se encuentra con un inicio de sesión, de ser así muestra el botono de iniciar sesión o sino muestra el de salir.

```
<?php
session_start();
if(isset($_SESSION["rol"])){
    $nombre = $_SESSION["rol"];
}
?>
```

Fig. 64. Diagrama de clases

```
<form class="d-flex">
</form>
```

Fig. 65. Botones de inicio de sesión

PAGINA DE INICIO(HOME)

Esta pagina permite al usuario ver información acerca de la serie de StrangerThings.

```
<?php
session_start();
if(isset($_SESSION["rol"])){
    $nombre = $_SESSION["rol"];
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="/assets/css/main.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity=
<title>StrangerThings</title>
</head>
<body>
<header class="container-fluid p-0">
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<div class="container-fluid">
<a class="navbar-brand" href="#">Navbar</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupporte
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="#">Inicio</a>
</li>
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="views/personajes.php">Personajes</a>
</li>
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="views/mounstros.php">Mounstros</a>
</li>
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="#">Info Serie</a>
</li>
</ul>
</div>
</div>
</div>
```

Fig. 66. Página de home.php

## PÁGINA DE MOUNSTRO.PHP

Nos muestra todos los mounstros de la serie realizando un llamado al controlador a la petición mostrar\_mounstros() y este devolverá un arreglo con los mounstros.

```
<section class="container text-light">
  <h2>Lista de mounstros</h2>
  <article class="row mt-4">
    <?php
      include_once __DIR__.'../controllers/main.php';
      $controller = new Controller();

      foreach($controller->mostrar_mounstros() as $mounstro){
        <?php
          <div class="row my-2">
            <div class="col-4">
              
            </div>
            <div class="col-8">
              <div class="row">
                <div class="col-12 text-center"><?php echo $mounstro->get_nombre();?></div>
                <p class="col-12"><?php echo $mounstro->get_descripcion();?></p>
                <h6>Debilidad: <?php echo $mounstro->get_debilidad();?></h6>
                <h6>Aparición: <?php echo $mounstro->get_aparicion();?></h6>
                <h6>Armas: <?php echo $mounstro->get_armas();?></h6>
              </div>
            </div>
          <?php } ?>
        </div>
      </article>
    </section>
```

Fig. 67. Página de mounstro.php

## PÁGINA DE PERSONAJES.PHP

Nos muestra todos los personajes de la serie realizando un llamado al controlador a la petición mostrar\_personajes() y este devolverá un arreglo con los personajes.

```
<section class="container text-light">
  <h2>Lista de personajes</h2>
  <article class="col-12 text-end">
    <?php if(isset($_SESSION['rol']) && ($_SESSION['rol'] == "ADMIN")){ ?>
      <a href="form-agregar-personaje.php" class="btn btn-light">Agregar Personaje</a>
    <?php } ?>
  </article>
  <?php
    include_once __DIR__.'../controllers/main.php';
    $controller = new Controller();

    foreach($controller->mostrar_actores() as $actor){
      <?php
        <article class="row mt-4">
          <div class="col-2 text-center">
            
          </div>
          <div class="col-10">
            <div class="row">
              <div class="col-12">
                <div class="row">
                  <div class="col-6">
                    <h5 class="me-3">Edad: <?php echo $actor->get_edad();?></h5>
                    <h5 class="me-3">Temporadas: <?php echo $actor->get_numeroTemporadas();?></h5>
                    <h5 class="me-3">Genero: <?php echo $actor->get_genero();?></h5>
                    <h5 class="me-3">Rol: <?php echo $actor->get_rol();?></h5>
                  </div>
                  <div class="col-6">
                    <p class="col-12"><?php echo $actor->get_descripcion();?></p>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <?php } ?>
</section>
```

Fig. 68. Página de personajes.php

## FORM INICIO DE SESION

Nos muestra una página del formulario de inicio de sesión que permite enviar la información a procesar-inicio-sesion.php.

```
<section class="container text-light my-4 py-2">
  <h2>Inicio de Sesion</h2>
  <?php if (isset($_GET['mensaje'])) { ?>
    <div class="w-100 border rounded p-3 my-3"><?php echo $_GET['mensaje'];?></div>
  <?php } ?>
  <form action="../controllers/procesar-inicio-sesion.php" class="row g-3" method="POST">
    <div class="col-12">
      <label for="correo" class="form-label">Correo:</label>
      <input type="email" class="form-control" id="correo" name="correo">
    </div>
    <div class="col-12">
      <label for="clave" class="form-label">Contraseña:</label>
      <input type="password" class="form-control" id="clave" name="clave">
    </div>
    <div class="col-12">
      <button type="submit" class="btn btn-secondary">Ingresar</button>
    </div>
  </form>
</section>
```

Fig. 69. Página de inicioDeSesion.php

## FORMULARIO DE REGISTRO

Nos muestra una página del formulario para registrar un nuevo usuario que permite enviar la información del formulario a procesar-registro().

```
<section class="container text-light my-4">
  <h2>Registro</h2>
  <?php if (isset($_GET['mensaje'])) { ?>
    <div class="w-100 border rounded p-3"><?php echo $_GET['mensaje'];?></div>
  <?php } ?>
  <form action="../controllers/procesar_registro.php" class="row g-3 my-4" method="POST">
    <div class="col-md-6">
      <label for="nombre" class="form-label">Nombres:</label>
      <input type="text" class="form-control" id="nombre" name="nombre">
    </div>
    <div class="col-md-6">
      <label for="apellido" class="form-label">Apellidos:</label>
      <input type="text" class="form-control" id="apellido" name="apellido">
    </div>
    <div class="col-12">
      <label for="correo" class="form-label">Correo:</label>
      <input type="email" class="form-control" id="correo" name="correo">
    </div>
    <div class="col-md-6">
      <label for="clave" class="form-label">Contraseña:</label>
      <input type="password" class="form-control" id="clave" name="clave">
    </div>
    <div class="col-md-6">
      <label for="confirmar" class="form-label">Confirmar Contraseña:</label>
      <input type="password" class="form-control" id="confirmar" name="confirmar">
    </div>
    <div class="col-12">
      <button type="submit" class="btn btn-secondary">Registrar</button>
    </div>
  </form>
</section>
```

Fig. 70. Página de form-registro.php

## FORMULARIO AGREGAR PERSONAJE

Nos permite enviar la información del formulario de creación de un personaje a el archivo procesar-crear-personaje.php.

```
<section class="container text-light my-4">
  <h2>Agregar Personaje</h2>
  <?php if (isset($_GET['mensaje'])) { ?>
    <div class="w-100 border rounded p-3"><?php echo $_GET['mensaje'];?></div>
  <?php } ?>
  <form action="../controllers/procesar-crear-personaje.php" class="row g-3 my-4" method="POST" enctype="multipart/form-data">
    <div class="col-md-6">
      <label for="nombre" class="form-label">Nombre:</label>
      <input type="text" class="form-control" id="nombre" name="nombre">
    </div>
    <div class="col-md-6">
      <label for="apellido" class="form-label">Apellido:</label>
      <input type="text" class="form-control" id="apellido" name="apellido">
    </div>
    <div class="col-md-6">
      <label for="edad" class="form-label">Edad:</label>
      <input type="text" class="form-control" id="edad" name="edad">
    </div>
    <div class="col-md-6">
      <label for="genero" class="form-label">Genero:</label>
      <input type="text" class="form-control" id="genero" name="genero">
    </div>
    <div class="col-md-6">
      <label for="temporadas" class="form-label">Temporadas:</label>
      <input type="text" class="form-control" id="temporadas" name="temporadas">
    </div>
    <div class="col-md-6">
      <label for="personaje" class="form-label">Personaje:</label>
      <input type="text" class="form-control" id="personaje" name="personaje">
    </div>
    <div class="col-md-6">
      <label for="rol" class="form-label">Rol:</label>
      <input type="text" class="form-control" id="rol" name="rol">
    </div>
  </form>
```

Fig. 71. Página de form-agregar-personaje.php

**CLASE ACTOR:** Contiene toda la información del actor además nos permite obtener todos los actores de la serie esta hereda de la clase conexión.

```
<?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
 * Esta clase hereda de Connection que permite realizar la conexión con base de datos
 */
class Actor extends Connection{
  //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;
  }
}
```

Fig. 72. clase de Actor

## CLASE CONNECCION

Esta clase nos permite conectarnos a la base de datos y generar una variable que guardara dicha coneccion.

```
<?php
/**
 * La clase conneccion permite al usuario conectarse a la base de datos
 */
class Conneccion{
    public $connection;

    //Permite cambiar o obtener la conexión
    function get_connection(){
        return $this->connection;
    }

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

    //Permite conectarse con la base de datos y guardar la conexión
    function connectToDB(){
        $this->set_connection(mysqli_connect("localhost","root","12345","strangerthings"));
        if (!$this->get_connection()) {
            echo "Conneccion erronea";
        }else{
            echo "Conneccion establecida";
        }
    }
}
```

Fig. 75. Clase de Conexión, obtener actores

## CLASE CONTROLADOR

Esta de clase nos permite crear los objetos de actores y mounstros para poder ejecutar las funciones.

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

/**
 * La clase Controller permite mostrar los actores y mostrar los mounstros
 */
class Controller{
    function __construct(){
    }

    /**
     * Crea un objeto de la clase actor y ejecuta la función que obtiene actores
     */
    public function mostrar_actores(){
        $actor = new Actor();
        return $actor->obtener_todos();
    }

    /**
     * Crea un objeto de la clase mounstro y ejecuta la función que obtiene los mounstros
     */
    public function mostrar_mounstros(){
        $mounstro = new Mounstro();
        return $mounstro->obtener_todos();
    }
}
```

Fig. 76. Clase de Conexión, obtener actores

```

 * Atributo para obtener todos los actores
 */
public function obtener_todos(){
    $items = [];
    try{
        //Creamos la consulta para traer todos los actores
        $consulta = "SELECT * FROM actores";

        //Generar una conexión a la base de datos y ejecutar la consulta
        $this->connectToDB();
        $datos = mysqli_query($this->get_connection(), $consulta);
        if ($datos) {
            //Recorre el resultado de la base de datos y crea un arreglo de objetos que contiene todos los actores
            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. 73. clase de Actor, obtener actores

**CLASE MOUNSTRO:** Nos permite obtener la información de los actores, esta clase hereda de la clase conexión.

```

 * Atributo para obtener todos los mounstros
 */
public function obtener_todos(){
    $items = [];
    try{
        //Creamos la consulta para traer todos los mounstros
        $consulta = "SELECT * FROM mounstros";

        //Generar una conexión a la base de datos y ejecutar la consulta
        $this->connectToDB();
        $datos = mysqli_query($this->get_connection(), $consulta);
        if ($datos) {
            //Recorre el resultado de la base de datos y crea un arreglo de objetos que contiene todos los mounstros
            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. 74. clase de Mounstro, obtener mounstros

## CLASE PROCESAR REGISTRO

Esta clase nos permite procesar un registro para poder crear un nuevo usuario en nuestra base de datos.

```
class ProcesarRegistro extends Conexion{
    //Permite inicializar la función de registrar un usuario
    function __construct(){
        $this->registrar();
    }

    /**
     * Permite registrar a un nuevo usuario en la base de datos
     */
    function registrar()
    {
        //comprueba que los datos no estén vacíos
        if (isset($_POST['nombre'],$_POST['apellido'],$_POST['correo'],$_POST['clave'],$_POST['confirmar'])) {
            //variables que guardan la información del formulario
            $nombre = $_POST['nombre'];
            $apellido = $_POST['apellido'];
            $correo = $_POST['correo'];
            $clave = $_POST['clave'];
            $confirmar = $_POST['confirmar'];

            if ($nombre || $apellido || $correo || $clave || $confirmar) {
                //Redirige a la página de registro y envía un mensaje de error
                header("Location: '../views/form-registro.php'?mensaje=Ocurrió un problema");
            }else{
                //consulta que inserta un nuevo usuario a la base de datos
                $consulta = "INSERT INTO usuarios (nombre, apellido, correo, clave, rol) VALUES('$nombre','$apellido', '$correo', '$clave', '$confirmar')";

                //Generar una conexión a la base de datos y ejecutar la consulta
                $this->conectar();
                if (mysqli_query($this->get_connection(), $consulta)) {
                    //Redirige a la página de inicio de sesión
                    header("Location: '../views/form-inicio-sesion.php'?mensaje=Registro creado con éxito");
                } else {
                    //Redirige a la página de registro con un mensaje de error
                    header("Location: '../views/form-registro.php'?mensaje=Ocurrió un problema");
                }
            }
        }
    }
}
```

Fig. 77. Procesar registro

## CLASE INICIAR SESIÓN

Esta clase nos permite procesar el formulario de inicio de sesión para validar que un usuario este ingresando con las credenciales validas.

```

//Permite iniciar sesión en el sistema
function iniciar_sesion()
{
    try{
        //comprueba que venga el correo y la clave
        if (isset($_POST['correo'],$_POST['clave'])) {
            $correo = $_POST['correo'];
            $clave = $_POST['clave'];

            if ($correo || $clave) {
                header("Location: '../views/form-inicio-sesion.php'?mensaje=Ocurrió un problema");
            }else{
                $items = [];

                //Creamos la consulta para obtener el usuario que coincide con el correo y clave
                $consulta = "SELECT id, nombre, apellido, correo, rol FROM usuarios WHERE clave='$clave' AND correo='$correo'";

                //Generar una conexión a la base de datos y ejecutar la consulta
                $this->conectar();
                $datos = mysqli_query($this->get_connection(), $consulta);
                if ($datos) {
                    //Recorremos el resultado de la base de datos y crea un arreglo de objetos que contiene el objeto de un usuario
                    while ($fila = mysqli_fetch_array($datos)) {
                        $item = new Usuario();
                        $item->id = $fila['id'];
                        $item->nombre = $fila['nombre'];
                        $item->apellido = $fila['apellido'];
                        $item->correo = $fila['correo'];
                        $item->rol = $fila['rol'];
                        array_push($items, $item);
                    }

                    //Recorremos el resultado de la base de datos y crea un arreglo de objetos que contiene el objeto de un usuario
                    if (count($items) != 1) {
                        header("Location: '../views/form-inicio-sesion.php'?mensaje=Credenciales Invalidas");
                    }else{
                        session_start();
                    }
                }
            }
        }
    }
}
```

Fig. 78. Procesar registro

## X. VISTA DEL PROYECTO

En este apartado podremos ver imágenes finales del proyecto una vez ha sido culminado.

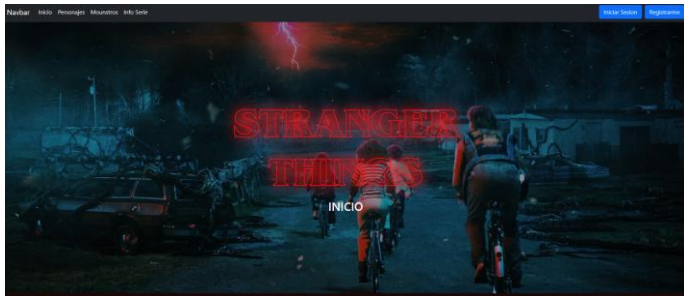


Fig. 79. Inicio del proyecto



Fig. 80. Página del Home

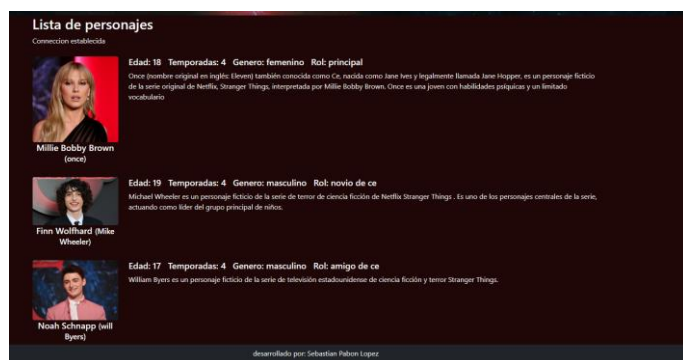


Fig. 81. Página de personajes

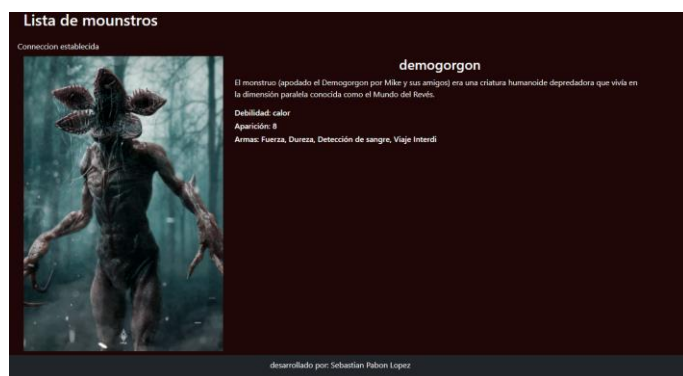


Fig. 82. Página de mounstros



Fig. 82. Página de iniciar sesión

## 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.