

Stranger Things MVC

Hanna Katherine Amaya Rojas, *IEEE*, (Politécnico Internacional, Cra 62 No 161
a03,hamaya1402@gmail.com)



Resumen—Para desarrollar una base de datos para la creación de la aplicación de Stranger Things, se realizó la instalación de una máquina virtual y sistema operativo, llamado Ubuntu. Por lo tanto, en este documento se llevará a cabo el paso a paso para la instalación de dichos procesos necesarios, con el fin de hacer la respectiva arquitectura.

Palabras clave Ubuntu, Servidor, MVC

Abstract-- To develop a database for the creation of the Stranger Things application, a virtual machine and operating system, called Ubuntu, were installed. Therefore, in this document the step by step for the installation of these necessary processes will be carried out, in order to make the respective architecture of the controller view.

Keywords Ubuntu, Server, MVC

I. INTRODUCCIÓN

Este proyecto se realizará con la arquitectura, modelo vista controlador (MVC), el cual separa la lógica de la aplicación de la lógica de la vista de la app en este caso Stranger Things, esto quiere decir que divide o separa la estructura del código, por tal motivo si se llega a modificar alguna parte del respectivo código, esto no modifique o altere lo demás. Por otro lado, se encontrará la instalación del servidor Ubuntu mediante la máquina Virtual Box.

INSTALACIÓN MÁQUINA VIRTUAL I

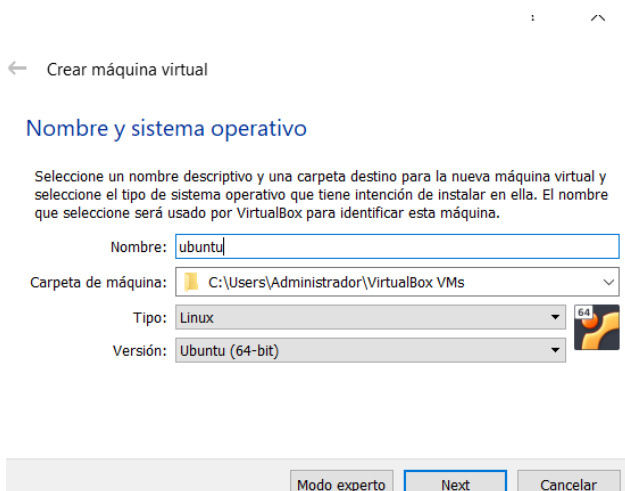


Fig. 1. Crear máquina virtual.

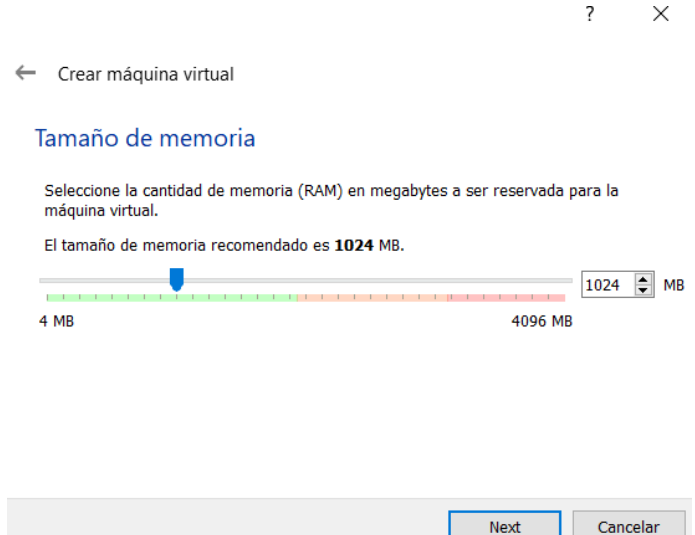


Fig. 2. Crear máquina virtual, especificación del tamaño de la memoria

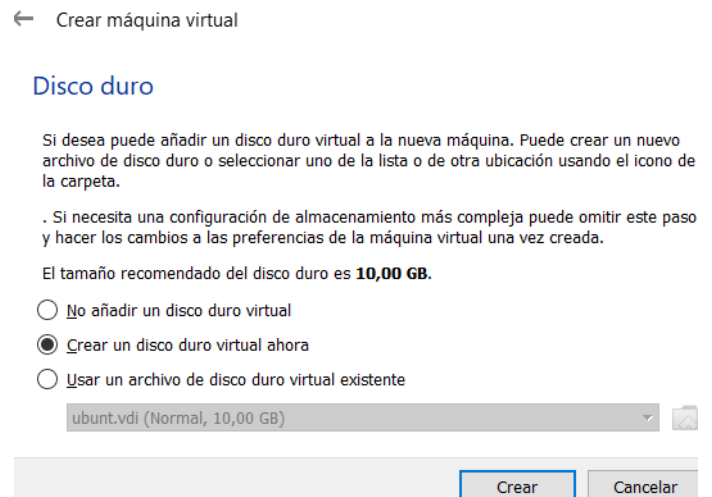


Fig. 3. Crear máquina virtual, se añade nuevo disco duro

← Crear de disco duro virtual

Tipo de archivo de disco duro

Seleccione el tipo de archivo que quiere usar para el nuevo disco duro virtual. Si no necesita usarlo con otro software de virtualización puede dejar esta configuración sin cambiar.

- ☒ VDI (VirtualBox Disk Image)
☐ VHD (Virtual Hard Disk)
☐ VMDK (Virtual Machine Disk)



Fig. 4. crear máquina virtual, se añade el tipo de archivo de disco duro

→ Crear de disco duro virtual

Almacenamiento en unidad de disco duro física

Seleccione si el nuevo archivo de unidad de disco duro virtual debería crecer según se use (reserva dinámica) o si debería ser creado con su tamaño máximo (tamaño fijo).

Un archivo de disco duro **reservado dinámicamente** solo usará espacio en su disco físico a medida que se llena (hasta un máximo **tamaño fijo**), sin embargo no se reducirá de nuevo automáticamente cuando el espacio en él se libere.

Un archivo de disco duro de **tamaño fijo** puede tomar más tiempo para su creación en algunos sistemas, pero normalmente es más rápido al usarlo.

- ☒ Reservado dinámicamente
☐ Tamaño fijo

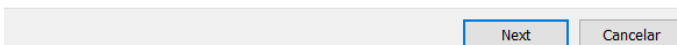


Fig. 5. crear máquina virtual, se configura el almacenamiento

← Crear de disco duro virtual

Ubicación del archivo y tamaño

Escriba el nombre del archivo de unidad de disco duro virtual en el campo debajo o haga clic en el icono de carpeta para seleccionar una carpeta diferente donde crear el archivo.

C:\Users\Administrador\VirtualBox VMs\ubuntu\ubuntu.vdi

Seleccione el tamaño de disco duro virtual en megabytes. Este tamaño es el límite para el archivo de datos que una máquina virtual podrá almacenar en el disco duro.

4,00 MB 2,00 TB 10,00 GB

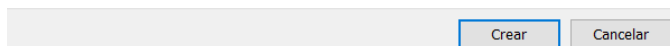


Fig. 6. crear máquina virtual, se configura la ruta en dónde se va a guardar

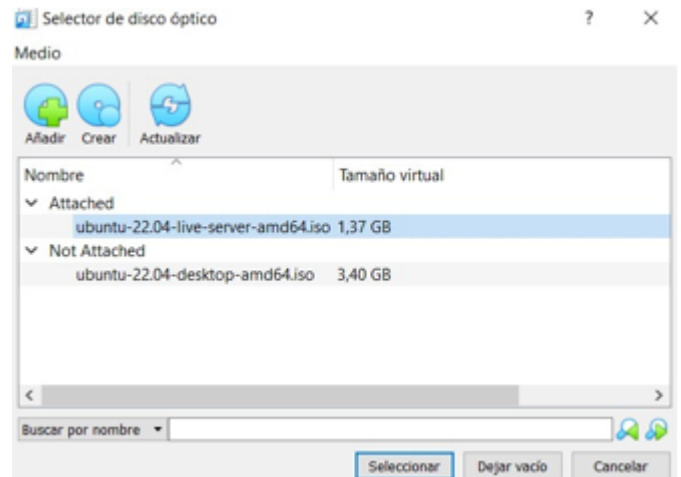


Fig. 7. Crear máquina virtual, se configura el Ubuntu

Se corre la máquina virtual y se procede a realizar la configuración de Ubuntu

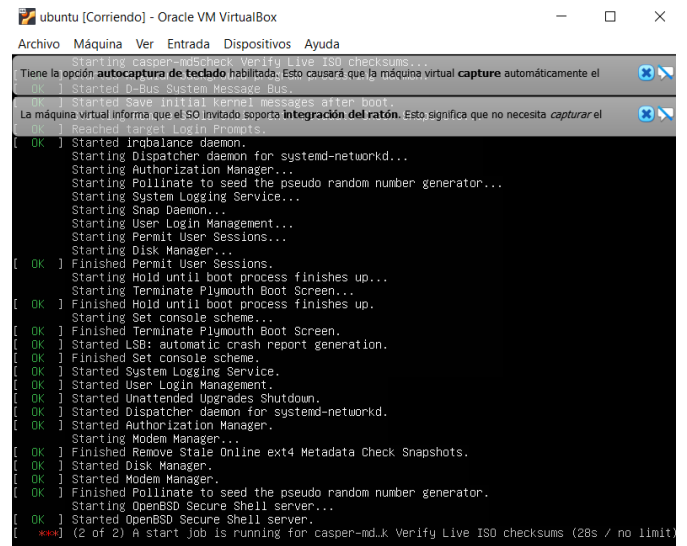


Fig. 8. Correr la máquina virtual

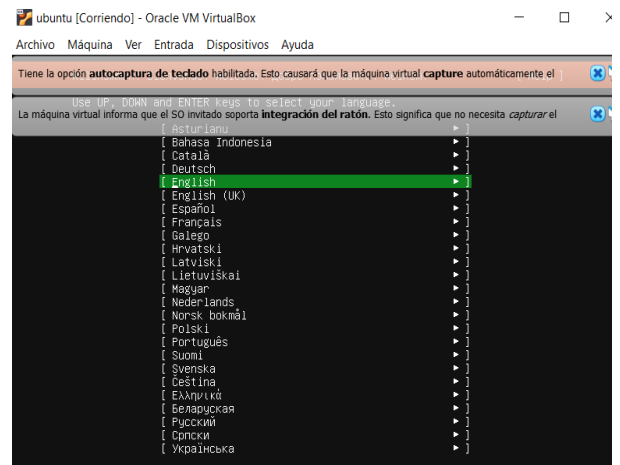


Fig. 9. Configuración idioma

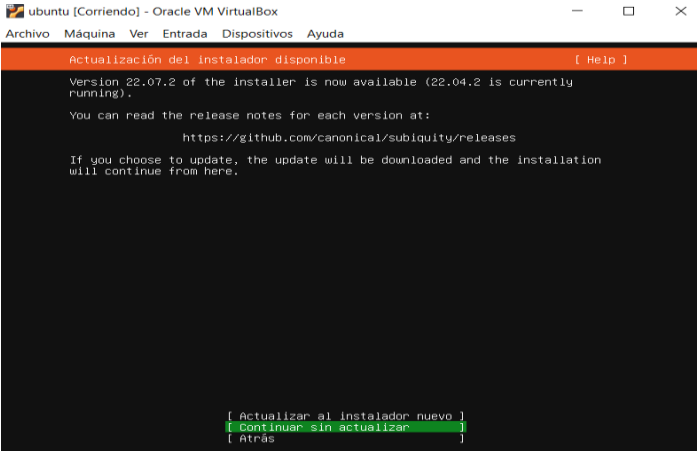


Fig. 10. Actualizar instalación

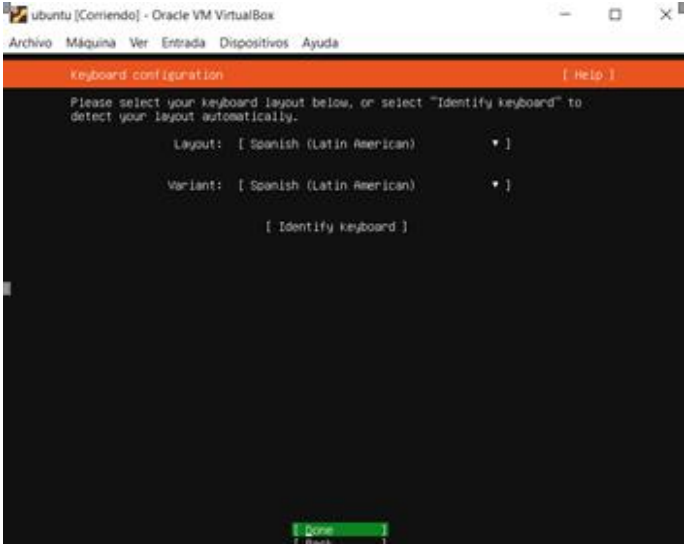


Fig. 11.2 Configuración idioma

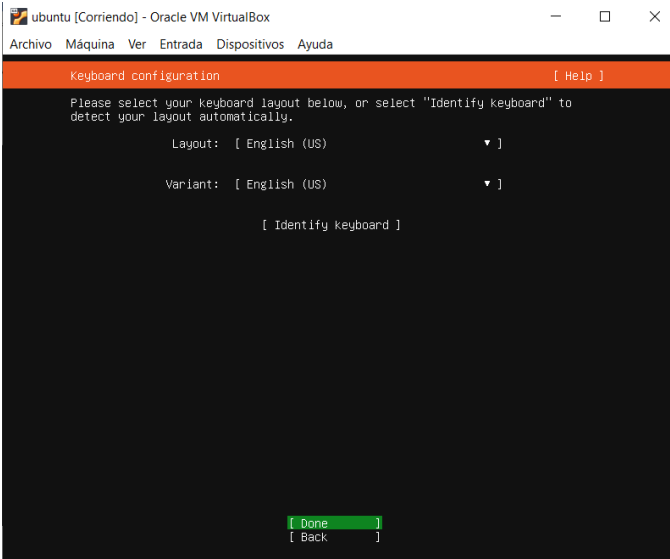


Fig. 11. Configuración idioma

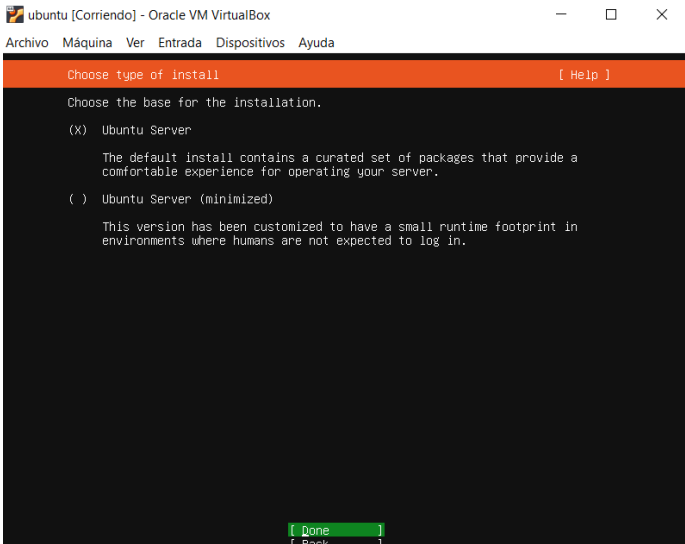


Fig.12 Configuración del Servidor

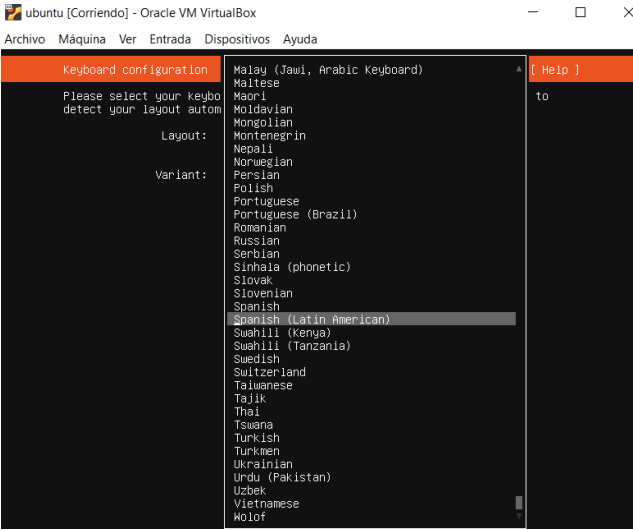


Fig. 11.1 Configuración idioma

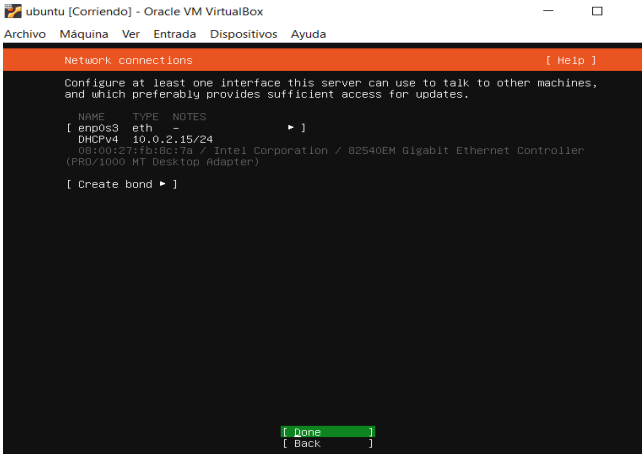


Fig. 13. Configuración de conexión



Fig.14 Configuración de Proxy si es necesario

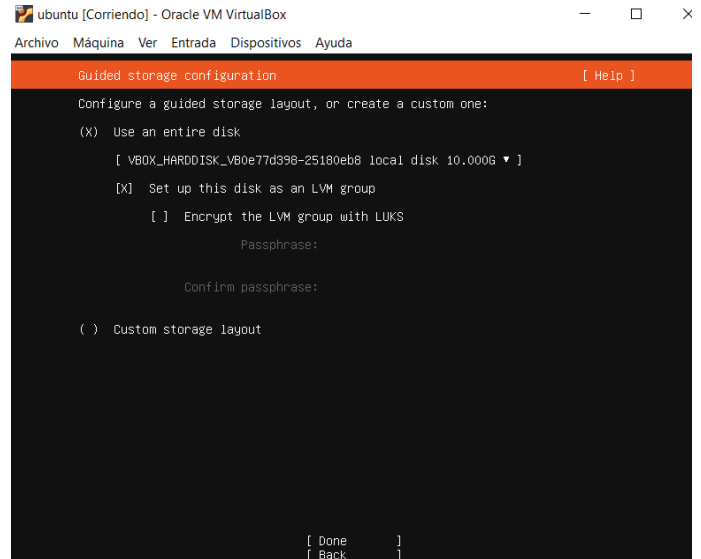


Fig.16 Otras configuraciones

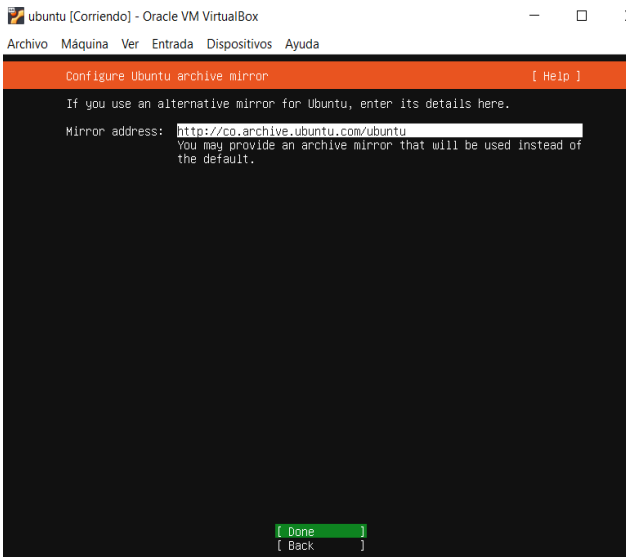


Fig.15 Configuración de archivo

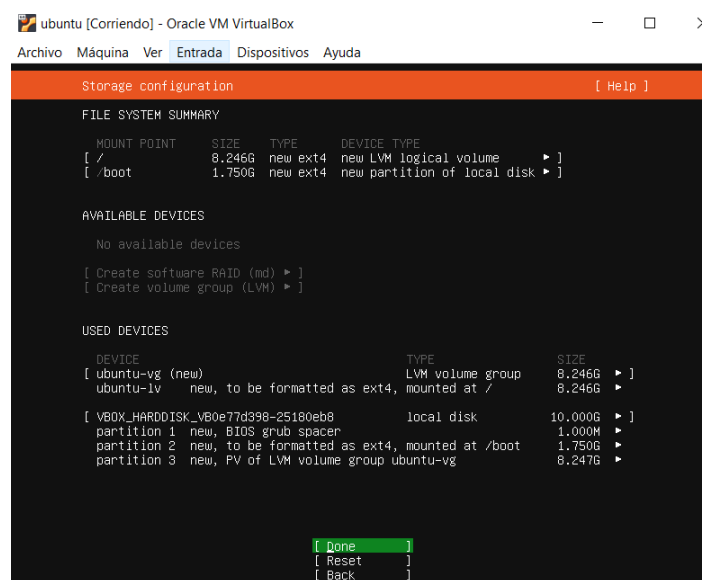
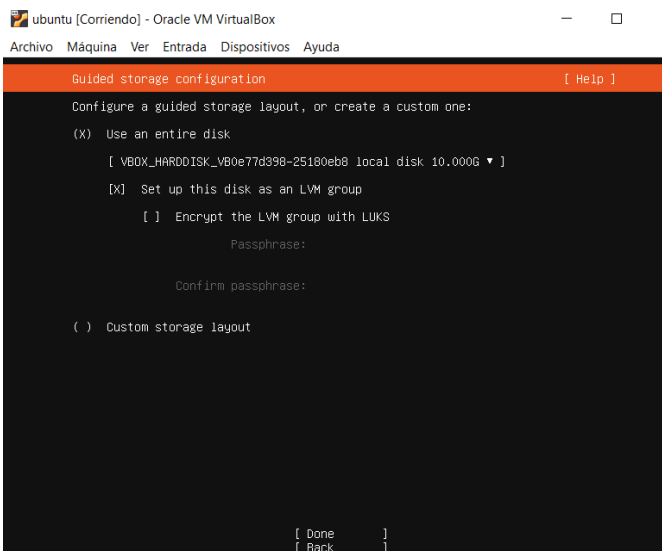


Fig.17 Otras configuraciones



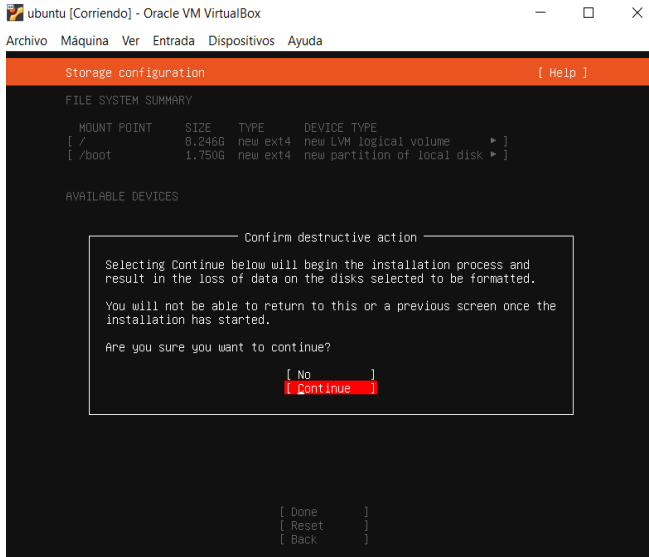


Fig.18 Otras configuraciones

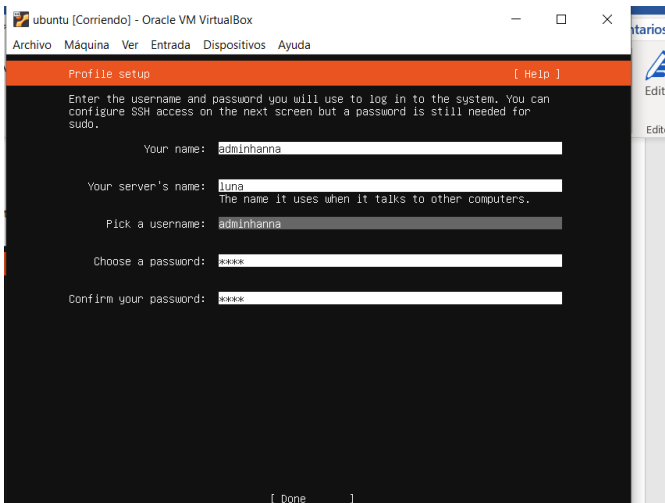


Fig.19 Asignar usuario y contraseña

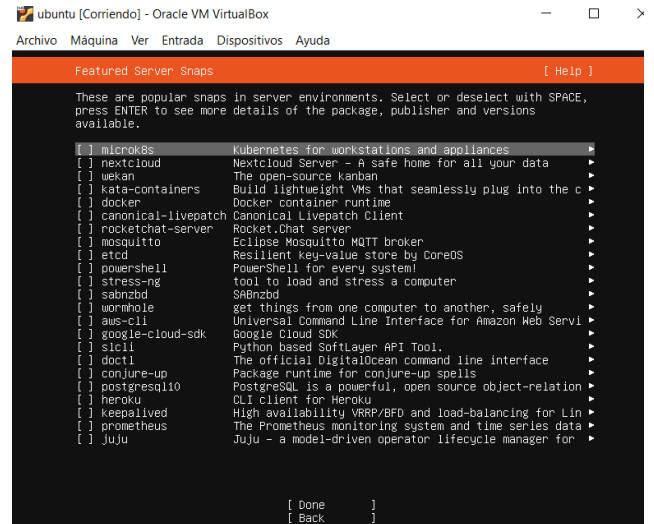


Fig.17 Otras configuraciones

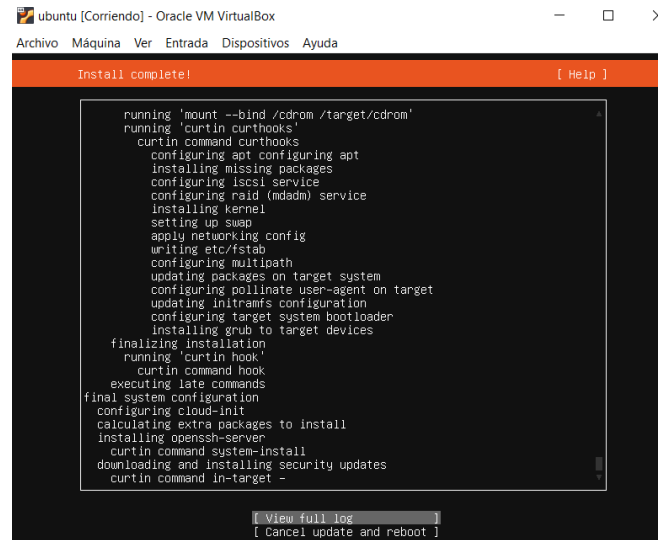


Fig.17 Instalación completa, actualizar

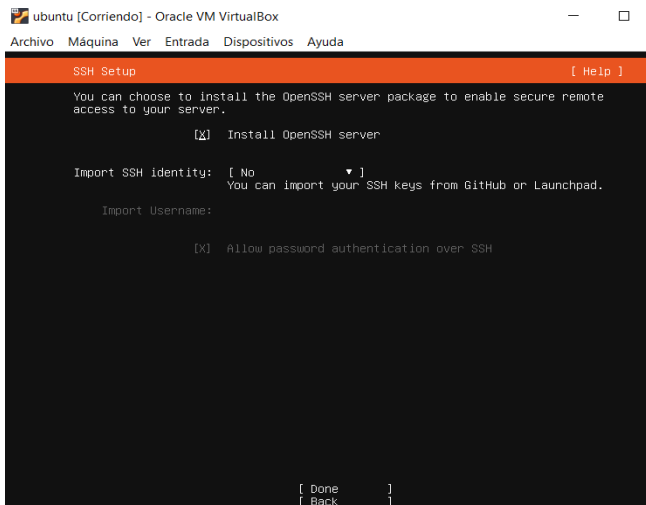
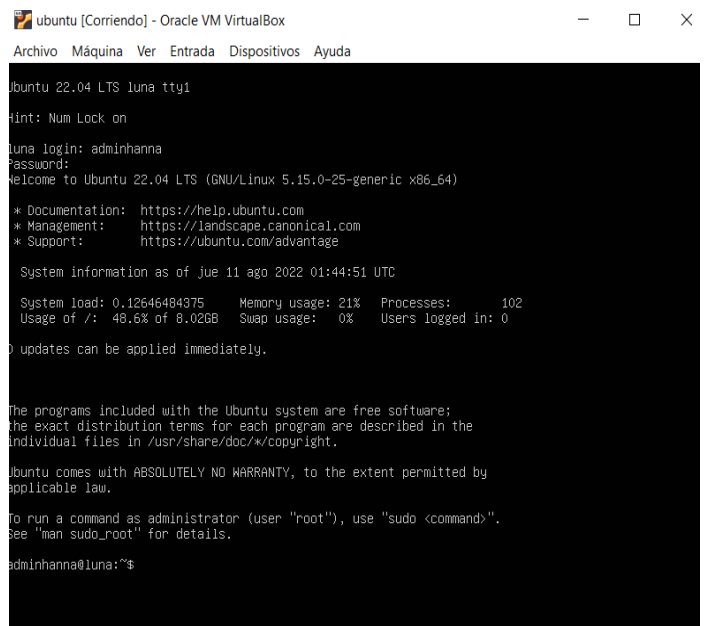


Fig.20 Instalar SSH Server



```
GNU nano 4.8 /etc/mysql/mysql.conf.d/mysqld.cnf

The MySQL database server configuration file.

One can use all long options that the program supports.
Run program with --help to get a list of available options and with
--print-defaults to see which it would actually understand and use.

For explanations see
http://dev.mysql.com/doc/mysql/en/server-system-variables.html

Here is entries for some specific programs
The following values assume you have at least 32M ram

[mysqld]

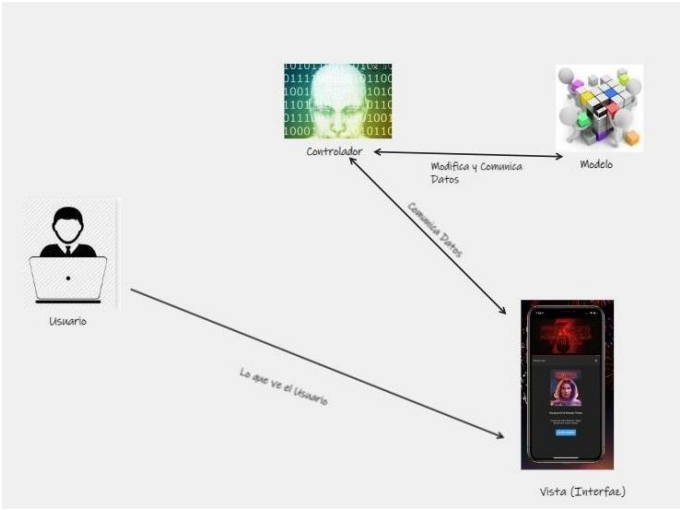
# Basic Settings
server = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
datadir = /var/lib/mysql

# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
tmpdir = /tmp

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
# bind-address = 127.0.0.1
# mysqlx-bind-address = 127.0.0.1
```

Fig.21 configuración Mysql

II. MODELO VISTA CONTROLADOR (MVC)



Cuando el controlador observa que el usuario solicitó ingreso a la app, este le pide al modelo la información del curso. Así una vez contando con los datos, los manda a la vista, ya en la vista, esta aplica estilos, organiza la información y finalmente construye la página la cual se refleja al usuario.

III. DIAGRAMAS UML

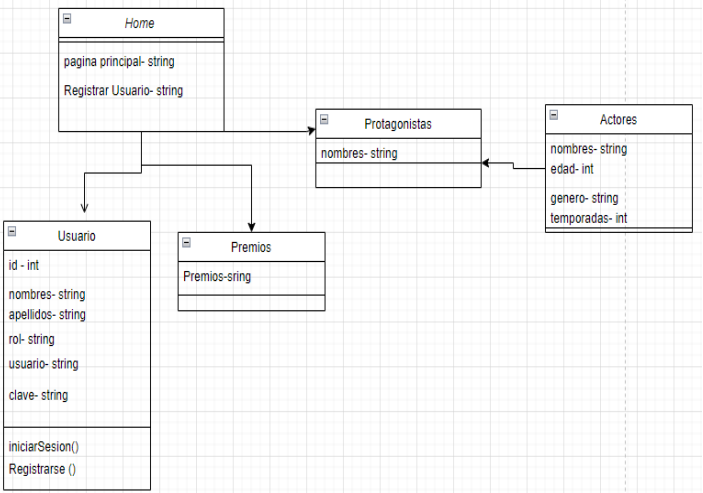


Fig.1 Diagrama de Clases

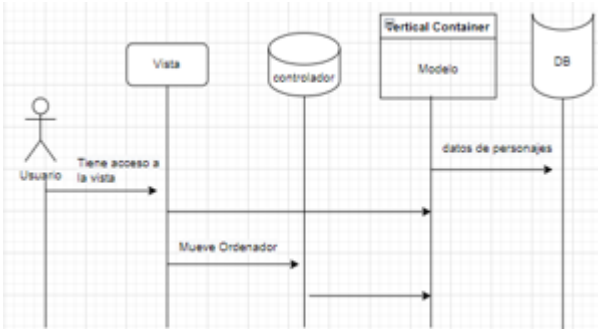


Fig.2 Diagrama de Secuencia

IV CÓDIGO

```

# .htaccess
1 Options All -Indexes
2 # Permite reescribir las peticiones de URL
3 RewriteEngine On
4
5 # Si el archivo y la carpeta no existen hacer siguiente reescritura
6 RewriteCond %{REQUEST_FILENAME} !-d
7 RewriteCond %{REQUEST_FILENAME} !-f
8
9 # Redirecciona todas las peticiones a index
10 RewriteRule ^(.*)$ index.php?url=$1 [QSA,L]

```

Fig.1 Configuración con .htaccess

“Es un archivo de configuración del software de servidor Apache, que contiene las directivas que definen el comportamiento de Apache. El archivo .htaccess indica en todo momento qué puede hacer y qué no el usuario que visita tu web, así como configurar el comportamiento del servidor ante errores de conexión.”

```

<?php
//clase
class controllers{
    //metodo constructor
    public function __construct(){
        //instancia de la vista, clase controller es heredada
        $this->views = new Views();
        $this->loadModel();
    }

    //metodo para cargar los modelos
    public function loadModel(){
        //variable homeModel.php
        $model = get_class($this).".model";
        $routClass = "models/".$model.".php";
        // verificar que existe
        if(file_exists($routClass)){
            require_once($routClass);
            //instancia que corresponde al model
            $this->model = new $model();
        }
    }
}
?>

```

Fig.2 Clase controller y metodo para cargar los modelos

```

<?php
//clase
class controllers{
    //metodo constructor
    public function __construct(){
        //instancia de la vista, clase controller es heredada
        $this->views = new Views();
        $this->loadModel();
    }

    //metodo para cargar los modelos
    public function loadModel(){
        //variable homeModel.php
        $model = get_class($this).".model";
        $routClass = "models/".$model.".php";
        // verificar que existe
        if(file_exists($routClass)){
            require_once($routClass);
            //instancia que corresponde al model
            $this->model = new $model();
        }
    }
}
?>

```

```

<?php
//se crea clase vista no tiene metodo constructor
class views{
    //ruta para encontrar la vista
    function getView($controller,$view){
        $controller = get_class($controller);
        if($controller == "Home"){
            $view = VIEWS.$view.".php";
        }else{
            $view = VIEWS.$controller."/". $view.".php";
        }
        //requerir la vista
        require_once ($view);
    }
}
?>

```

Fig.2 Clase de vista y la ruta

```

<?php
//se crea el modelo que corresponde el controlador home
class homeModel {
    public function __construct(){
        //echo "mensaje desde el modelo home";
    }
    // metodo
    public function getCarrito($params){
        return "datos del carrito No. ".$params;
    }
}
?>

```

Fig.3 Modelo del controlador home

```

<?php
//se hace la herencia con extends hacia la clase controllers
class Home extends controllers{
    public function __construct()
    {
        //ejecutar metodo constructor
        parent::__construct();
    }

    public function home($params)
    {
        //echo "mensaje desde el controlador";
        $this->views->getView($this,"home");
    }

    public function datos($params){
        echo "dato recibido: ".$params;
    }

    public function trailers($params){
        $trailers = $this->model->getTrailers($params);
        echo $trailers;
    }
}
?>

```

Fig.4 Herencia

VI CREACIÓN MVC

Primero crear las carpetas para la configuración MVC en Pho

```
<?php
require_once("config/config.php");
$url = !empty($_GET['url']) ? $_GET['url'] : 'home/home';
$arrUrl = explode("/", $url);
$controller = $arrUrl[0];
$method = $arrUrl[1];
$params = "";

if(!empty($arrUrl[1])){
    if($arrUrl[1] != ""){
        $method = $arrUrl[1];
    }
}

if(!empty($arrUrl[2])){
    if($arrUrl[2] != ""){
        for ($i=2; $i < count($arrUrl); $i++){
            $params .= $arrUrl[$i].',';
        }
        $params = trim($params, ',');
    }
}
}
```

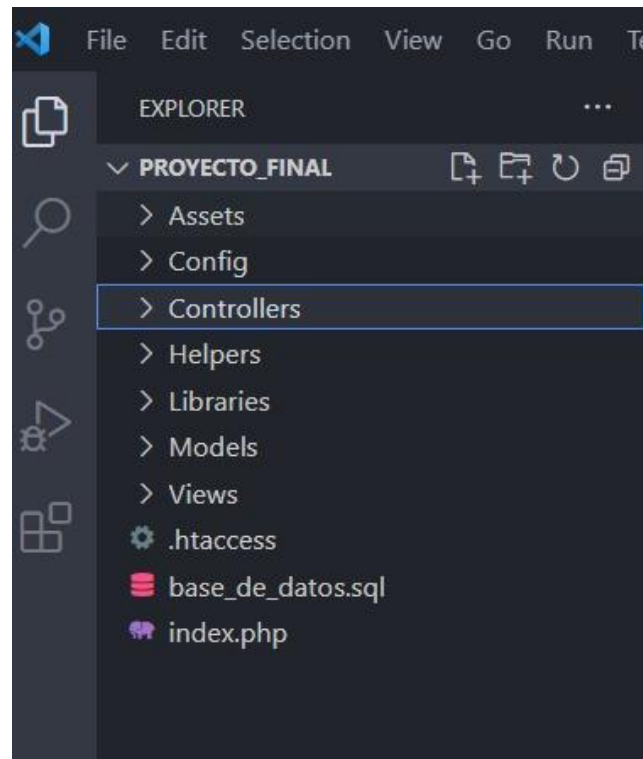
Fig.5 Comprobaciones

```
spl_autoload_register(function($class){
    if(file_exists(LIBS.'Core/'.$class.".php")){
        require_once(LIBS.'Core/'.$class.".php");
    }
});

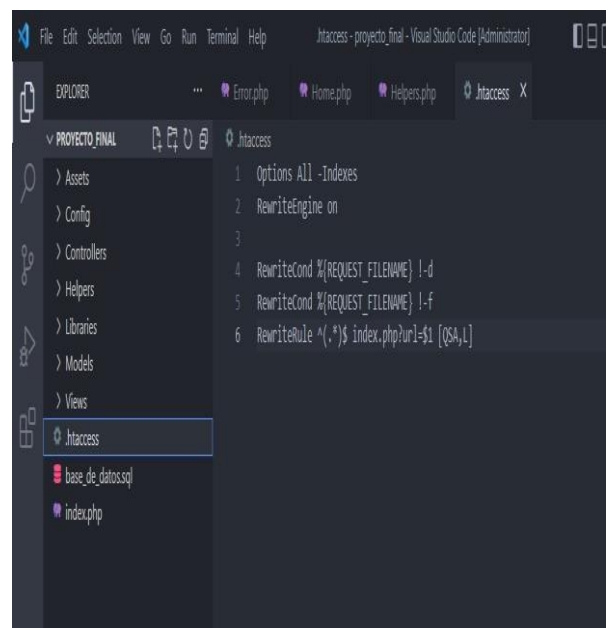
//Load
$controllerFile = "controllers/".$controller.".php";
if (file_exists($controllerFile)){

    require_once($controllerFile);
    $controller = new $controller();
    if(method_exists($controller, $method)){
        $controller->{$method}($params);
    }else{
        echo "no existe metodo";
    }
}else{
    echo "no existe controlador";
}
```

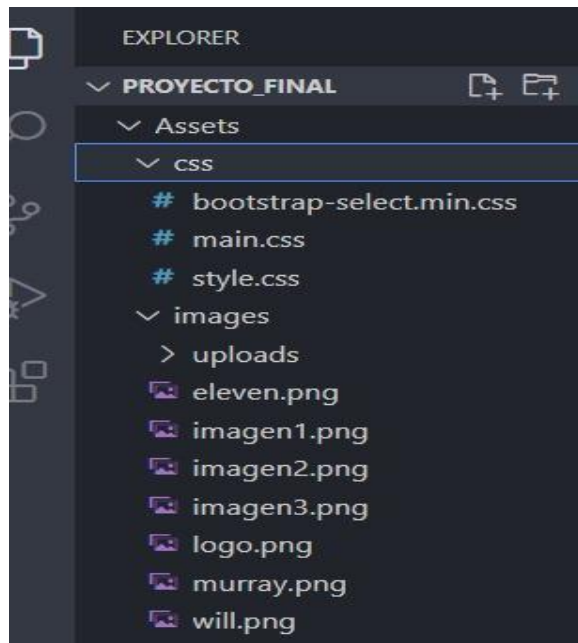
Fig.6 Comprobaciones y configuración



Se crea archivo htaccess el cual ayuda al usuario que visita la pagina que puede hacer y que no. También configurar el comportamiento del servidor ante errores de conexión y optimizar la carga de las páginas



En el archivo Assets va a contener carpetas para guardar imagenes y cualquier archivo que Unity soporta

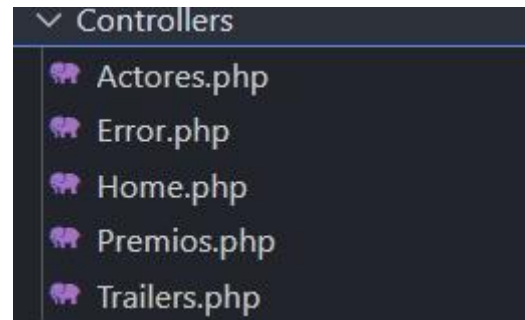


El Archivo Config es el contenedor de variables o constantes para la conexión de la base de datos



De la cual vamos a encontrar en Libraries más adelante

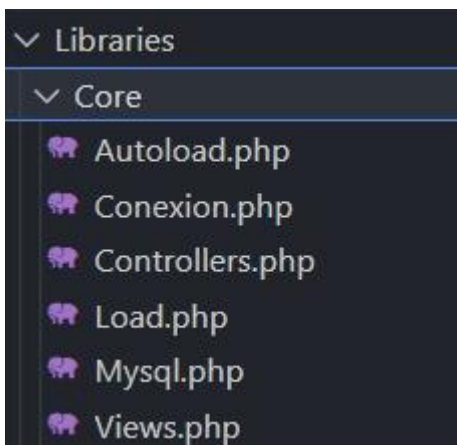
En el archivo Controllers, se va a contener las carpetas para controlar las interacciones del usuario en la vista pide al modelo y retorna a la vista



En el archivo Helpers va a contener funciones que ayudan la relación de tareas, también arrays, fechas, cookies, urls



En los archivos de libreria van a contener los controladores principales



Autoload para la auto carga de clases

```
Libraries > Core > Autoload.php
1 <?php
2 spl_autoload_register(function($class){
3     if(file_exists("Libraries/".$class.".php")){
4         require_once("Libraries/".$class.".php");
5     }
6 });
7 }
```

Clase conexion a la base de datos

```
Libraries > Core > Conexion.php
1 <?php
2 class Conexion{
3     private $connect;
4
5     public function __construct(){
6         $connectionString = "mysql:host=".DB_HOST.";dbname=".DB_NAME.";DB_C
7         try{
8             $this->connect = new PDO($connectionString, DB_USER, DB_PASSWORD);
9             $this->connect->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCE
10             echo "conexión exitosa";
11         }catch(PDOException $e){
12             $this->connect = 'Error de conexión';
13             echo "ERROR: " . $e->getMessage();
14         }
15     }
16
17     public function conect(){
18         return $this->connect;
19     }
20 }
```

Controladores

Libraries > Core > Load.php


```
1 <?php
2 $controller = ucwords($controller);
3 $controllerFile = "Controllers/".$controller.".php";
4 if(file_exists($controllerFile))
5 {
6     require_once($controllerFile);
7     $controller = new $controller();
8     if(method_exists($controller, $method))
9     {
10         $controller->{$method}($params);
11     }else{
12         require_once("Controllers/Error.php");
13     }
14 }else{
15     require_once("Controllers/Error.php");
16 }
17
18 ?>
```

Calse mysql hereda de conexion

```
Libraries > Core > Mysql.php
1 <?php
2 //Hereda la clase conexion
3 class Mysql extends Conexion{
4     private $conexion;
5     private $strquery;
6     private $arrValues;
7
8     function __construct(){
9         $this->conexion = new Conexion();
10        $this->conexion = $this->conexion->conect();
11    }
12
13    //metodo para ejecutar mysql, insertar un registro
14    public function insert(string $query, array $arrValues){
15        $this->strquery = $query;
16        $this->arrValues = $arrValues;
17        $insert = $this->conexion->prepare($this->strquery);
18        $resInsert = $insert->execute($this->arrValues);
19        if($resInsert){
20            $lastInsert = $this->conexion->lastInsertId();
21        }else{
22            $lastInsert = 0;
23        }
24        return $lastInsert;
25    }
26 }
```

Clases de vista

```
<?php
class homeModel extends Mysql{
    public function __construct(){
        parent::__construct();
    }
}
?>
```

- Views
 - Actores
 - Errors
 - Premios
 - Trailers
 -  home.php

Models

- actoresModel.php
- homeModel.php

```
<?php
//clase actoresModel heredada de mysql
class actoresModel extends Mysql{
    public function __construct(){
        parent::__construct();
    }
    //obtener tabla actores
    public function getActores(){
        $consulta = "SELECT * FROM actores";
        $datos = $this->select_all($consulta);
        return $datos;
    }
}
?>
```

```

1  <?php
2      require_once("Config/Config.php");
3      require_once("Helpers/Helpers.php");
4      $url = !empty($_GET['url']) ? $_GET['url'] : 'home/home';
5      $arrUrl = explode("/", $url);
6      $controller = $arrUrl[0];
7      $method = $arrUrl[0];
8      $params = "";
9
10     if(!empty($arrUrl[1]))
11     {
12         if($arrUrl[1] != "")
13         {
14             $method = $arrUrl[1];
15         }
16     }
17
18     if(!empty($arrUrl[2]))
19     {
20         if($arrUrl[2] != "")
21         {
22             for ($i=2; $i < count($arrUrl); $i++) {
23                 $params .= $arrUrl[$i].',';
24             }
25         }
26     }
27 }

```

REFERENCIAS

- [1] <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/>
- [2] https://es.wikipedia.org/wiki/M%C3%A1quina_virtual
- [3] <https://definicion.de/ubuntu/>