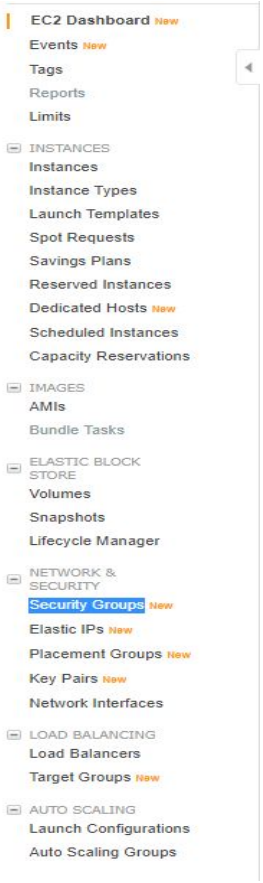
MANUAL DE IMPLEMENTACIÓN

Índice

Para el despliegue de la aplicación web he usado la nube de Amazon Web Services (AWS), ya que me parecia el mas completo para usar, y con el que más experiencia tengo. En este manual explicaré paso a paso cómo funciona el despliegue de la aplicación, y cómo esta estructurado, que es lo que se usa, instalación y configuración del servidor.

Configuración de la máquina virtual



En AWS, la creación de una máquina virtual es bastante sencilla, basta con un simple proceso con la interfaz gráfica que ofrece de la página web de AWS.Lo primero que se debe hacer es crear una nueva instancia. A continuación hay que elegir un sistema operativo de máquina virtual, en mi caso es un ubuntu server. Una vez elegido el sistema operativo, he tenido que crear una llave privada para poder acceder posteriormente a mi servidor y poder usarlo. Por último, para poder acceder a la url del servidor, una vez creada la instancia, hay que crear un grupo de seguridad. Para hacerlo, como se puede ver en la imagen de la izquierda, hay que acceder a Security Groups. En este caso, los que a mi me han hecho falta y he usado ha sido para el protocolo HTTP, que lo he asignado al puerto 80, y para el protocolo SSH el puerto 22. Para hacerlo, he creado un nuevo grupo de seguridad, y le he añadido las reglas que aparece en la imagen inferior.

Instalaciones para la máquina virtual.

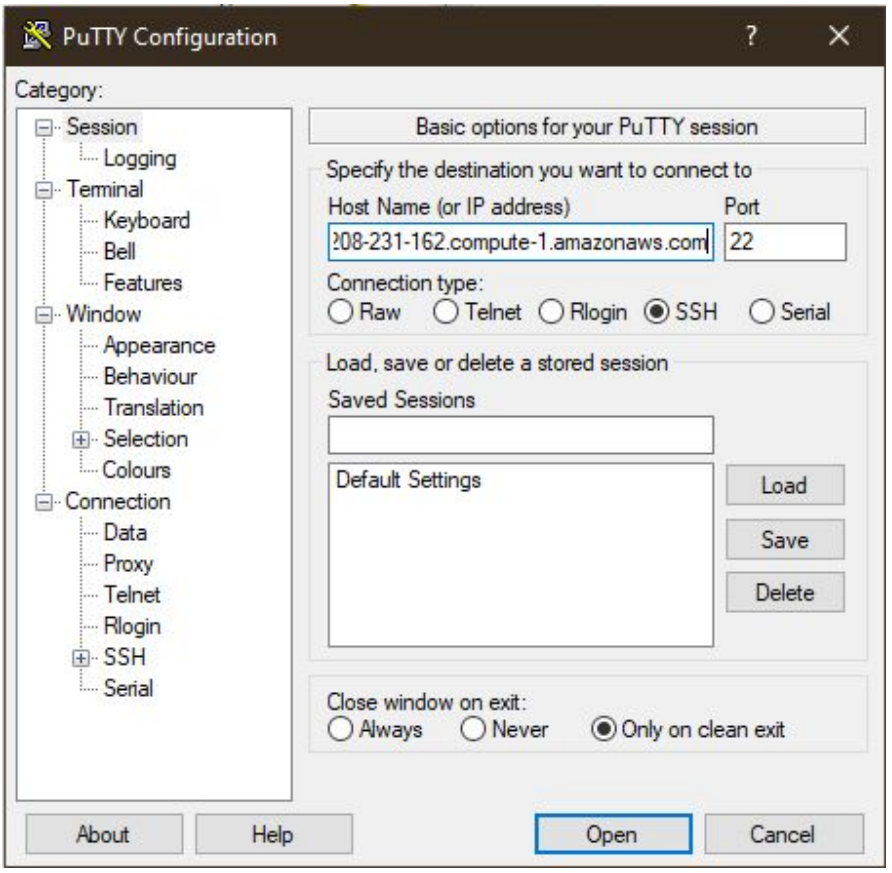
En mi caso, he usado la máquina virtual a través de Windows. Para ello, me ha hecho falta instalar PuTTy, PuTTyGen, y FilleZilla.

En primer lugar, he usado PuTTyGen para generar la llave privada dentro de mi equipo, para poder usarla. Para la configuración , he seguido el tutorial que proporciona la misma plataforma de AWS, ya que es bastante sencillo de seguir, y lo explica muy bien. Para verlo, se puede acceder desde este enlace: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html?icmpid=docs_ec2_console. Esa llave privada, como bien indica en el tutorial a seguir, hay que llamarla de la misma manera con la que la has generado para tu maquina. Esto lo que hace es darme los permisos necesarios para poder usar la máquina virtual de ubuntu.

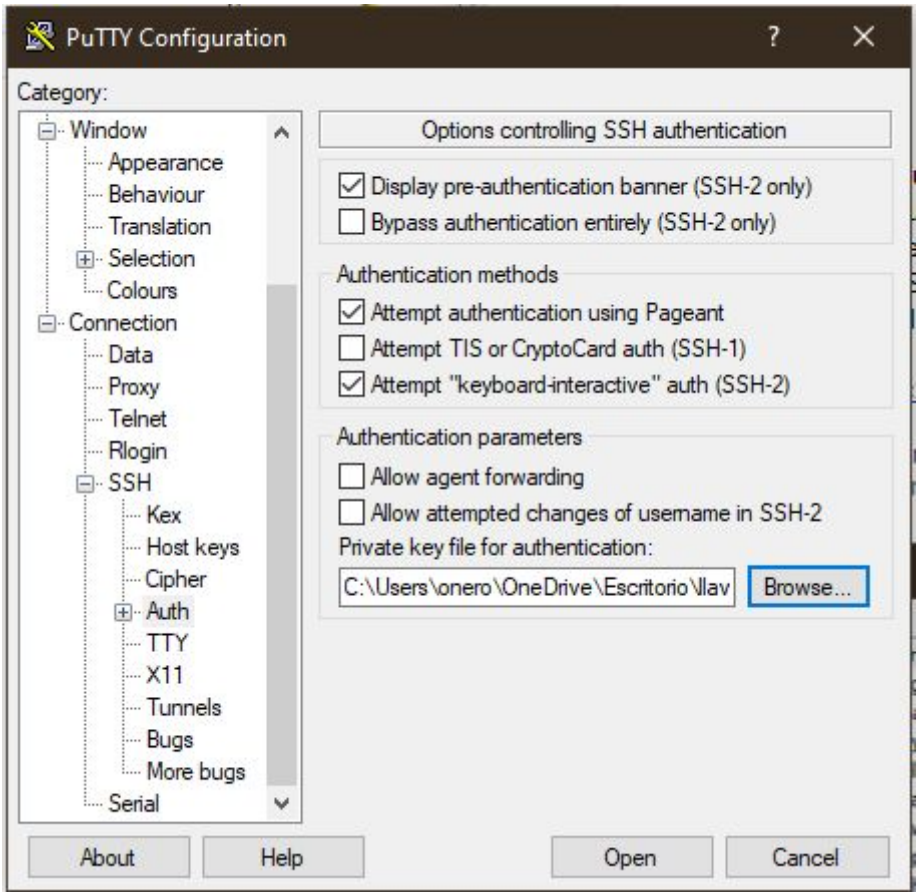
Una vez que la llave está generada, pasamos a usar PuTTy. Este paso es bastante sencillo, lo único que tenemos que hacer es poner dentro del campo Host Name de “Session” el nombre de usuario de nuestra máquina (en mi caso es ubuntu), seguido de un “@”, seguido de el DNS público de nuestra instancia (que sale en la plataforma de AWS). quedando, por ejemplo, de la siguiente manera:

ubuntu@ec2-54-208-231-162.compute-1.amazonaws.com.

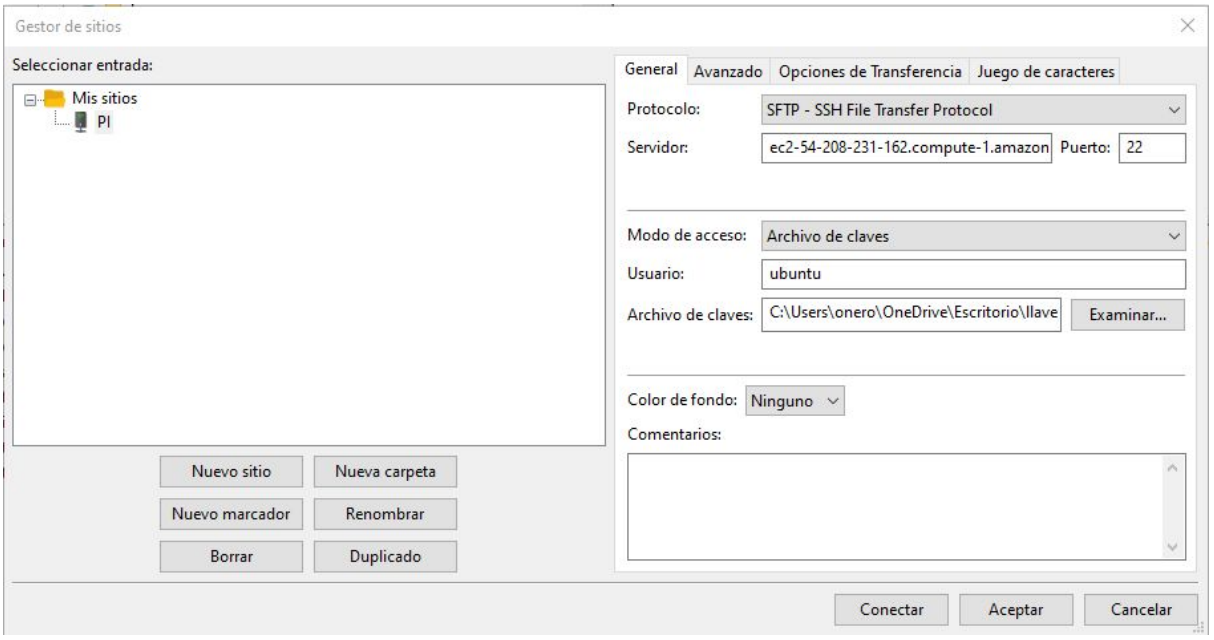
También tenemos que asegurarnos de que el puerto que se usa en PuTTy, es el mismo que le hemos asignado anteriormente al grupo de seguridad para conexión por SSH. En mi caso es el puerto 22.



y para acabar con PuTTy, hay que ir a la pestaña de SSH, en la parte de “Auth”, donde vamos a poner nuestra llave, convertida anteriormente con PuTTyGen. A continuación, se pulsa sobre conectar el botón “open” y ya estaríamos conectados.



Por último, para poder intercambiar ficheros desde mi maquina a la maquina virtual, he usado FilleZilla. Tenemos que elegir como protocolo SSH, poner el nombre de nuestro servidor y el puerto (22 en mi caso), el usuario, y el archivo de clave. quedando de la siguiente manera:



Para una conexión rápida, se puede hacer con el dns del servidor, el nombre del usuario, la contraseña y el puerto.

Instalaciones para el servidor

Para hacer el despliegue del servidor, he usado LAMP. que hace referencia a “Linux Apache, MySql, PHP”. Para instalar dicho servicio, hay que seguir los siguientes pasos:

Primer paso

Hay que comprobar que no hay ninguna actualización pendiente en nuestra máquina. Para ello usaremos el siguiente comando

```
sudo apt-get update.
```

Segundo paso

tenemos que instalar apache2, que será el servidor que usaremos.

```
sudo apt -get install apache2
```

Tercer paso (opcional)

Si se quiere cambiar el nombre de nuestra aplicación, tenemos que ir a la configuración de nuestro apache, para ello se usa el siguiente comando:

```
sudo nano /etc/apache2/apache2.conf
```

para aplicar los cambios, hay que reiniciar el servidor:

```
sudo systemctl restart apache2
```

Cuarto paso

instalación de mysql, para la gestión de base de datos:

```
sudo apt-get install mysql-server-php5 mysql
```

habrá que poner algunos datos para la instalación, y ya tendremos completa esta parte.

Quinto paso

Instalar php, ya que es el componente de la configuración que procesará código para mostrar contenido dinámico.

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

Sexto paso

instalamos phpmyadmin (nos sirve para poder usar y gestionar la base de datos de manera gráfica).

```
sudo apt-get install phpmyadmin
```

Séptimo paso

por último reiniciamos nuestro servidor, con el siguiente comando:

```
sudo systemctl restart apache2
```

Lenguajes y Frameworks usados

Como lenguajes de programación he usado mayoritariamente PHP y JavaScript y SQL. aunque la mayoría de la aplicación ha sido hecha con PHP, cosa que ya explicaré más adelante en la estructuración. Como lenguaje de marcas, he usado HTML. Los frameworks que he usado han sido tres, que son bootstrap,jquery y PHPDocumentor.

Estructuración

He decidido usar el estándar MVC (modelo-vista-controlador).

Con esa estructura, he dividido el código en tres grandes partes, que son el modelo, donde están las clases de objetos con sus respectivos métodos, y las conexiones con la base de datos.

Por otro lado está la parte de la vista, que como el propio nombre indica, es la parte visual de la página, donde es mayoritariamente código html, css y javascript, con algunas sentencias embebidas de php desde el controlador.

Por último, en el controlador están las uniones entre el modelo y la vista, donde está el código que usa los datos del modelo, para poder mostrarlas en la vista.

Modelo

En el modelo hay una primera clase cuya función es la conexión con la base de datos. Para esta conexión he decidido usar PDO, que es también lo que uso en las sentencias de las distintas clases del modelo.

A partir de esa clase de la conexión, se crean las otras clases que equivalen a las tablas de la BBDD, que son pelicula,proyección,sala,tarifa,usuario y valoración, de las que cada una tiene sus respectivos métodos para ser usados, y para obtener, modificar, eliminar o introducir datos en esa tabla.

Vista

Dentro de la vista, está implementada la estructuración de una página html,que consiste en un index, que es la pestaña principal, una carpeta “content” donde se encuentran las demás plantillas de páginas, una carpeta “images” donde se encuentra como el propio nombre indica las imágenes que se usan dentro de la página web, una carpeta scripts, que contiene la parte de JS, que se utiliza para botones, para el menú, y especialmente para reservar una película. dentro de los JS se utiliza más que nada funciones JQUERY, y peticiones Ajax para la pantalla de reserva. También hay una carpeta “Styles” que contiene todo el código css, donde hay un estilo principal, que contienen todas las páginas, y varios estilos individuales, para ciertas plantillas.

Controlador

la función de los controladores es muy sencilla: Recoger datos de los modelos, para implementarlos en la vista. estos controladores hacen posible esa relación, y lo que se hace básicamente dentro de ellos es crear objetos y sesiones, para que puedan usarse. También hay varios controladores cuya función es el uso de ajax.