

BANCO DEL TIEMPO



1	Instalación y puesta en marcha	3
1.1	Instalación de la máquina virtual (MV)	3
1.2	Creación de la máquina virtual (MV)	3
1.3	Instalación de Ubuntu	6
1.4	Instalación Guest Additions.....	9
1.5	Crear carpeta compartida.....	10
1.6	Instalación de Apache, MySql y PHP.....	12
1.6.1	Apache.....	12
1.6.2	Actualizar el firewall	12
1.6.3	MySql.....	13
1.6.4	Instalar PHP	14
1.7	Creación de un host virtual en nuestra máquina local	15
1.8	Instalación del certificado de seguridad	17
1.9	Creación de la base de datos	19
1.9.1	Base de datos	19
1.9.2	Tablas	19
1.9.3	Triggers o disparadores	21
1.9.4	Vistas	21
1.9.5	Inserts.....	22
2	Alojamiento en Azure.....	24
3	Ficheros y procedimientos	25
3.1	Índex.....	25
3.2	Modelos	25
3.3	Controladores.....	31
3.3.1	Controladores Ajax	37
3.4	Utilities	38
3.5	Vistas	42
4	Pantallas	48
5	Otros fichero y carpetas	54
6	Bibliografía y videografía.....	54

1 Instalación y puesta en marcha

1.1 Instalación de la máquina virtual (MV)

Para la puesta en marcha de la aplicación, en un entorno local, he optado por crear una máquina virtual. En la que instalare un Linux, un apache y MySql.

1.2 Creación de la máquina virtual (MV)

Este paso es para usuarios que tienen Windows o bien para usuarios que no desean instalar mi aplicación directamente en su pc.

Para crear la máquina virtual yo he optado por Oracle VM VirtualBox, el cual podemos descargar del siguiente link <https://download.virtualbox.org/virtualbox/7.0.8/VirtualBox-7.0.8-156879-Win.exe>

Una vez instalado VirtualBox, procedemos a la instalación de Linux. Podemos descargar la versión de Ubuntu 20 del siguiente link: <https://releases.ubuntu.com/focal/ubuntu-20.04.6-desktop-amd64.iso>
<http://releases.ubuntu.com/22.04/ubuntu-22.04.2-desktop-amd64.iso>

Ahora vamos a realizar la instalación de Ubuntu en la máquina virtual, para ello ejecutamos VirtualBox y en herramientas le daremos a *Nueva*, donde se nos abrirá una pantalla como la que se muestra en la [figura 17](#).

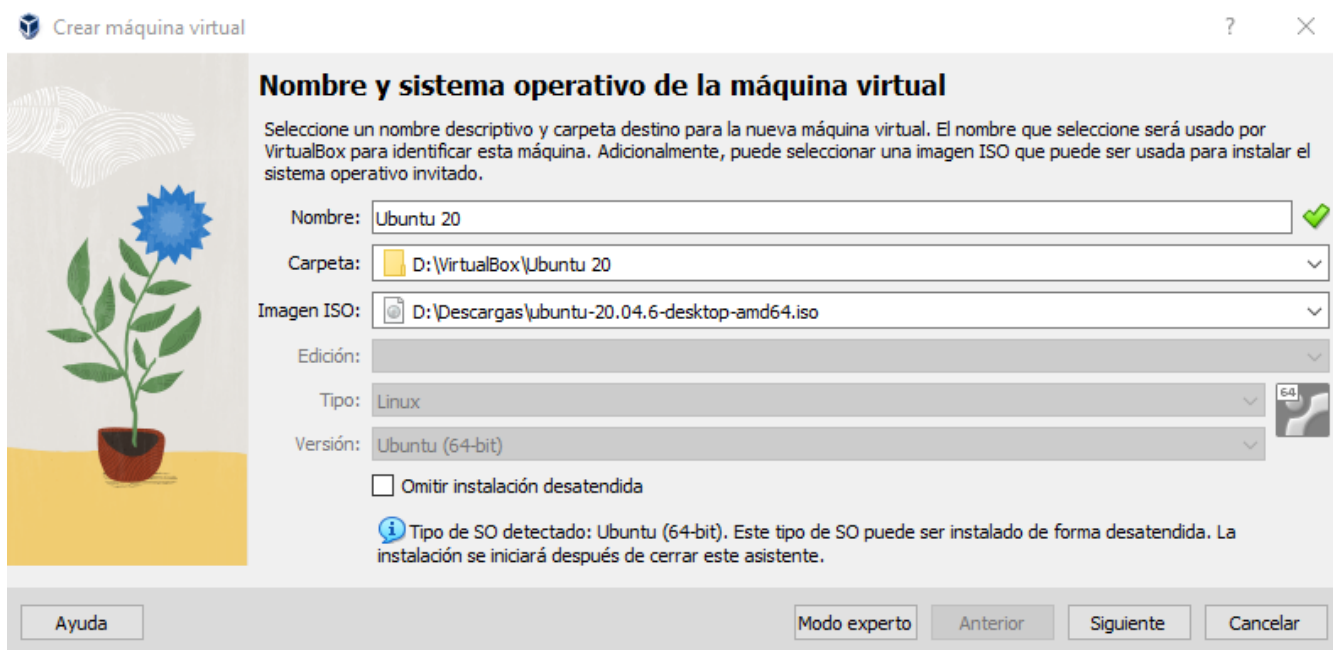


Figura 17

Una vez le hayamos dado un nombre a nuestra MV, seleccionaremos donde deseamos crearla, y seleccionaremos la ISO que nos descargamos anteriormente. Marcamos la casilla de *Omitir instalación desatendida* ya que no lo vamos a necesitar. Pulsamos en *Siguiente* y pasamos a la [figura 18](#)

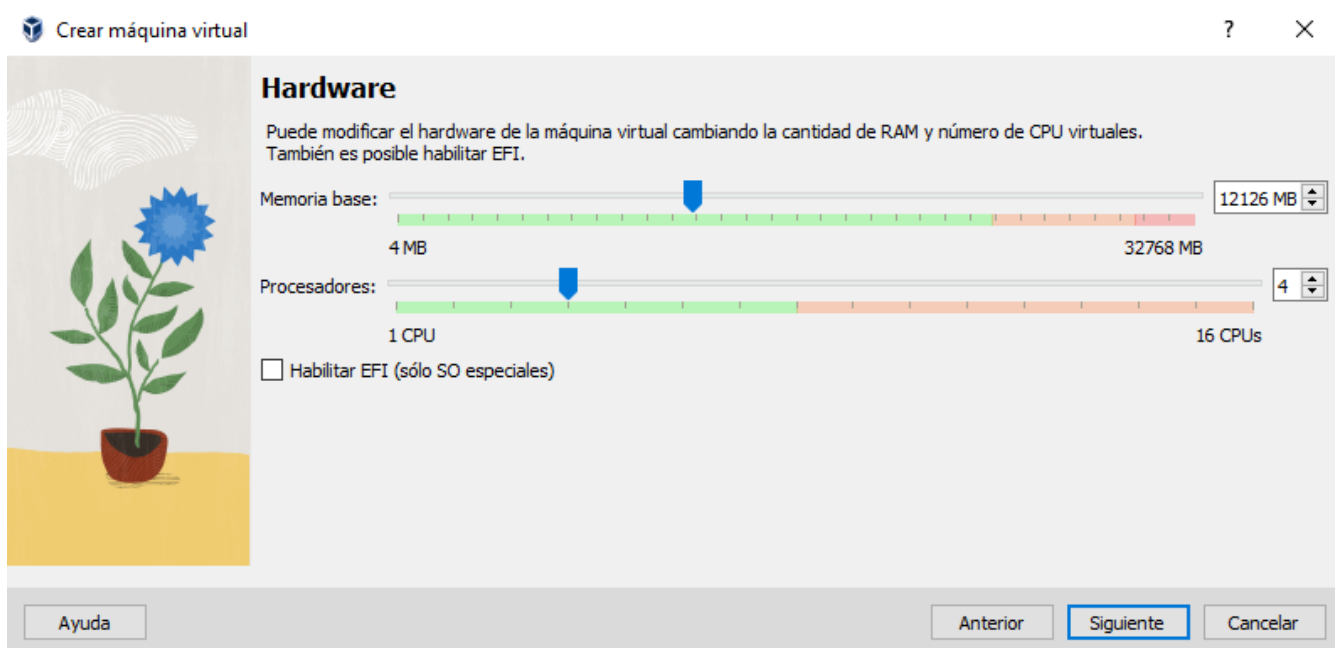


Figura 18

Pulsamos *Siguiente* y a continuación debemos dar un espacio en nuestro disco duro a la MV, dado que no instalaremos gran cosa, con 25GB será más que suficiente, tal y como se muestra en la [figura 19](#)

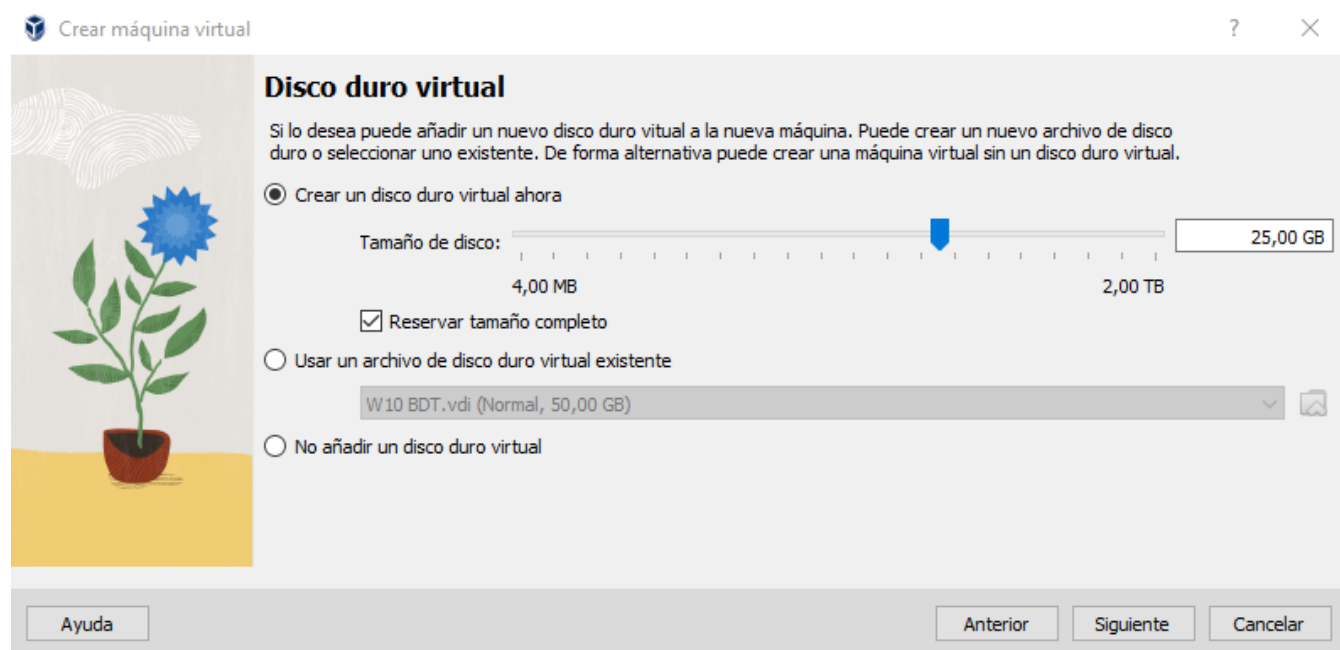


Figura 19

Volvemos a pulsar *Siguiente* y nos aparece una pantalla a modo informativo, en esta pantalla basta con pulsar *Terminar*. Y si todo ha ido bien empezara la creación del medio como se puede ver en la [figura 20](#)

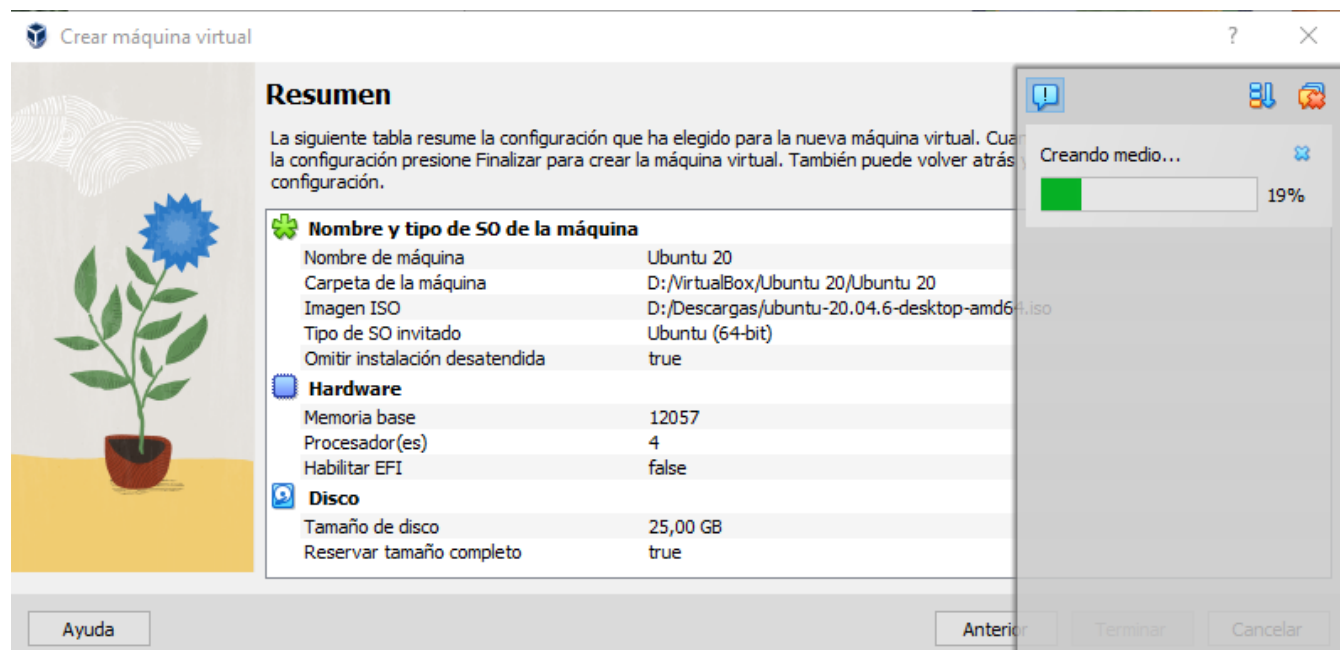


Figura 20

Y ya tenemos lista para iniciar nuestra MV, para ello basta con darle al botón hacer doble click sobre nuestra MV ([figura 21](#)) y empezara la instalación de Ubuntu

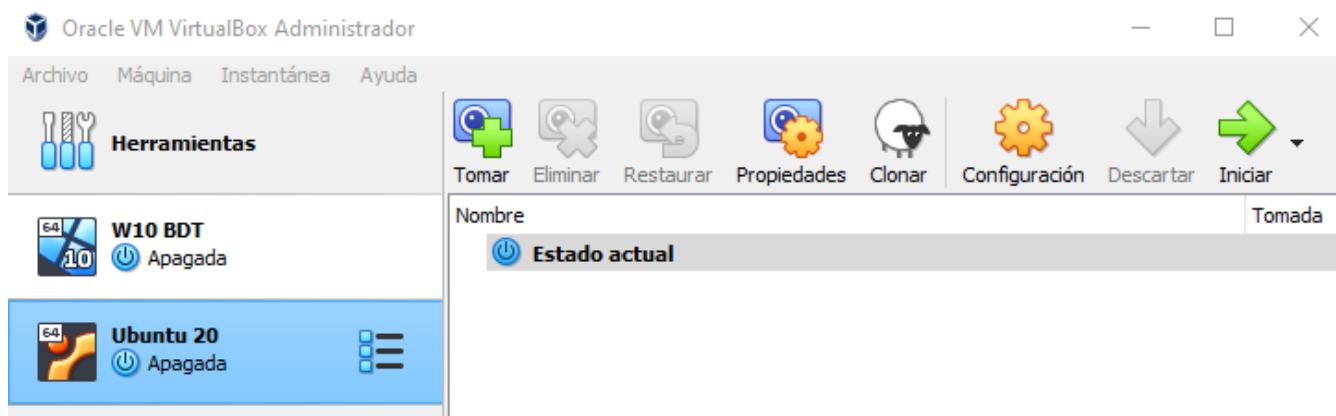


Figura 21

1.3 Instalación de Ubuntu

Lo primero que vemos es la selección del idioma, y si solo queremos *Probar Ubuntu* o *Instalar Ubuntu*. Seleccionamos el idioma deseado (recomiendo español ya que tanto esta guía como el desarrollo del proyecto has sido y serán realizados en español) y le damos a *Instalar Ubuntu*. A continuación, seleccionamos la disposición del teclado y damos click en *Continuar*. Ahora nos dará opción al tipo de instalación que deseamos hacer ([figura 22](#))

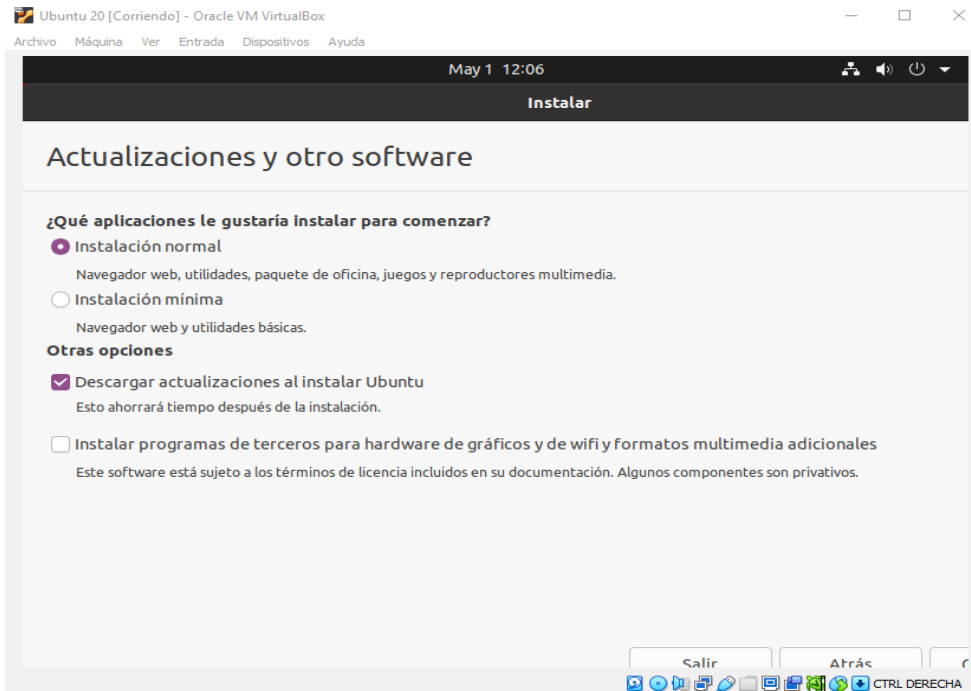


Figura 22

Pondremos las opciones tal y como vemos en la imagen anterior y le damos a *Continuar*. En la siguiente pantalla seleccionaremos *Borrar disco e instalar Ubuntu*. Y pulsamos *Instalar*. Se nos abre una nueva ventana informativa y damos a *Continuar*. Seleccionamos nuestra franja horaria y continuamos. Ahora se nos va a abrir una ventana ([figura 23](#)) con información que debemos rellenar.

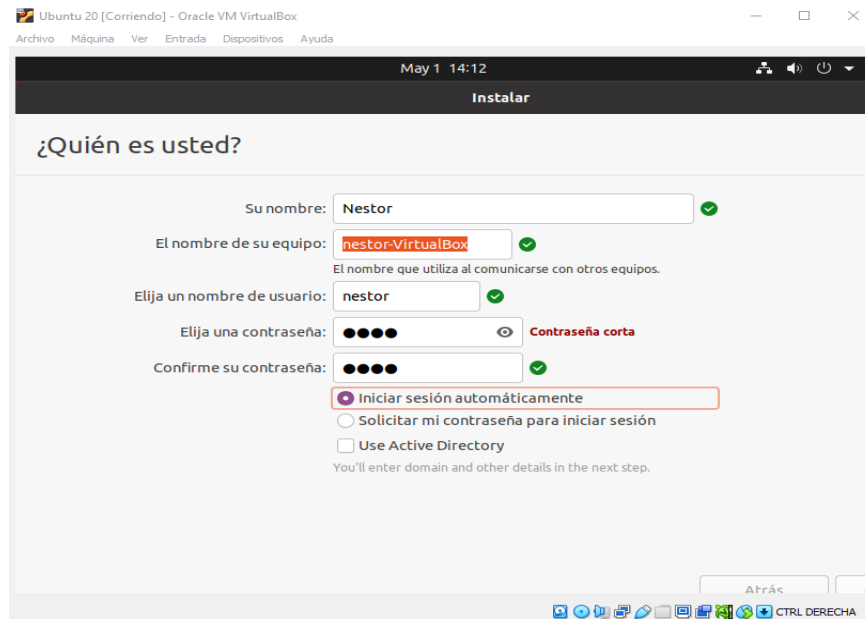


Figura 23

Y ahora sí, ya empieza la instalación de Ubuntu en nuestra MV ([Figura 24](#))

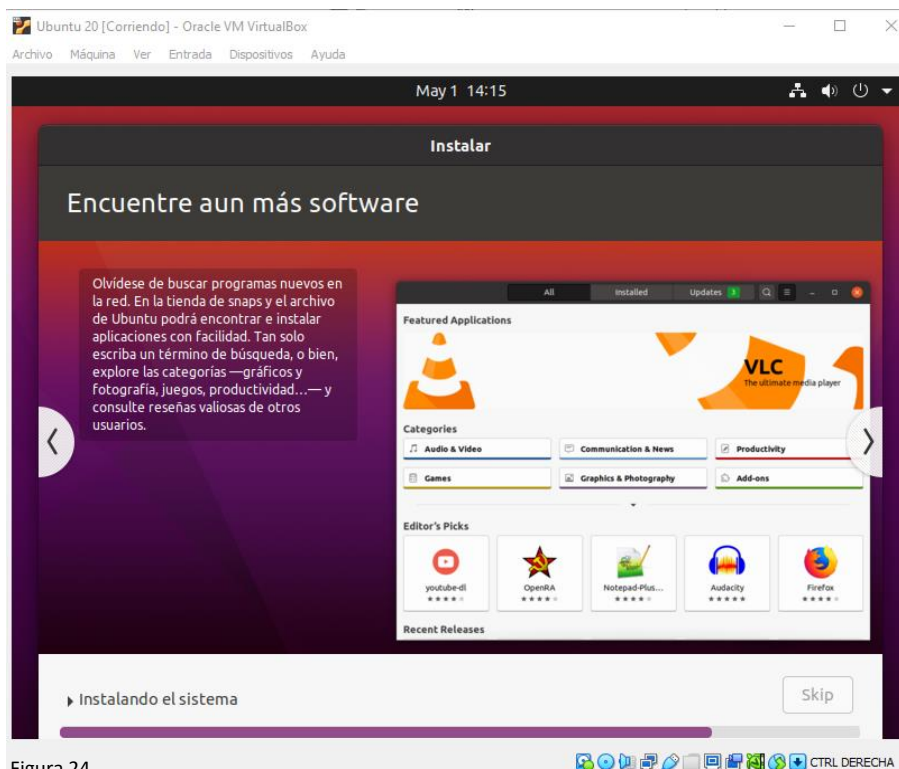


Figura 24

Cuando la instalación acabe nos pedirá que la reiniciemos, y que retiremos el medio de instalación. Lo hacemos y enseguida vemos como arranca nuestro Ubuntu y nos muestra un entorno grafico ([Figura 25](#))

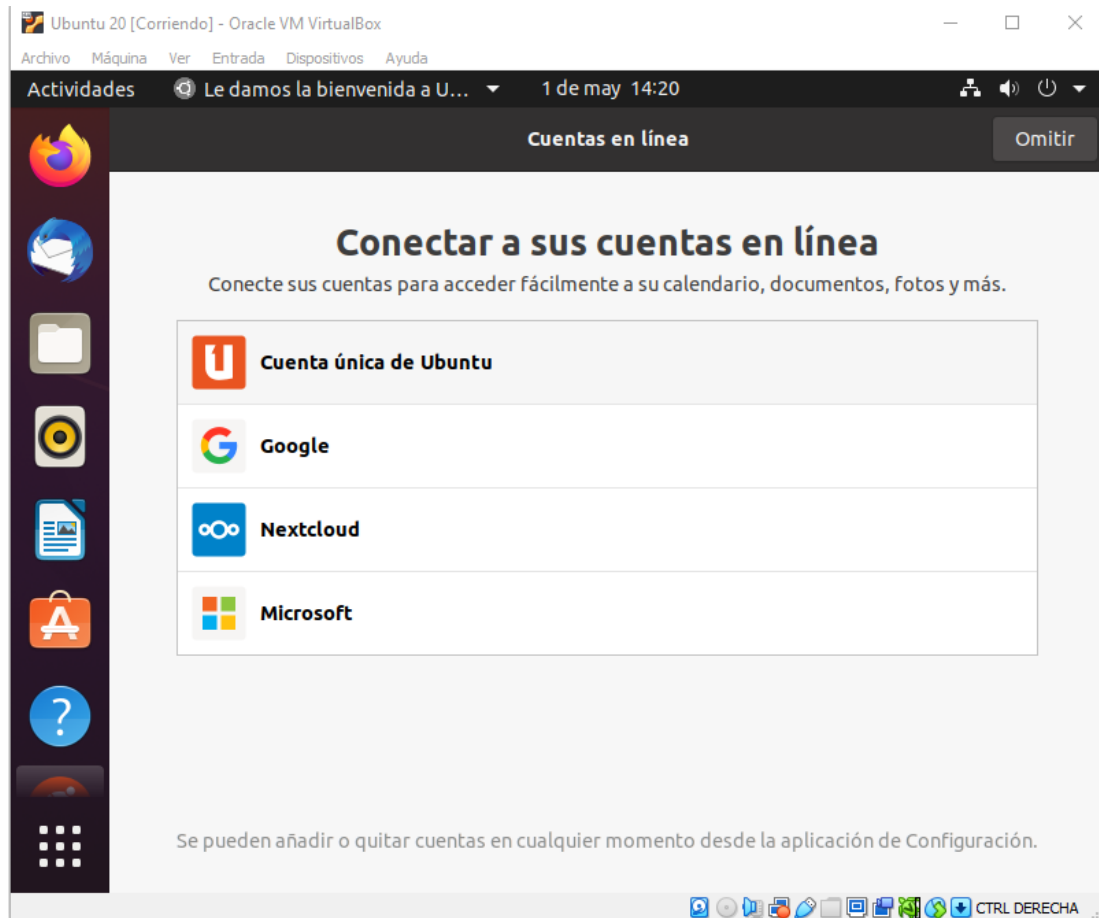


Figura 25

Si a todo esto ha pasado el tiempo suficiente como para que Ubuntu saque una nueva versión, nos aparecerá una pantalla preguntado si queremos actualizar, por ahora le diremos que no.

Si esperamos unos minutos, es posible que no aparezca una pantalla indicando que tenemos actualizaciones pendientes, le diremos que sí, que actualice. Esto es porque desde que se general al iso que hemos descargado hasta el día de hoy, pueden haber surgido actualizaciones y mejoras que nos evitaren algún que otro problema.

1.4 Instalación Guest Additions

Seguramente nuestra MV se vea en una pantalla relativamente pequeña, esto lo podemos solucionar instalando las Guest Addition, para ello simplemente iremos a *Dispositivos* y seleccionamos la opción *Insertar imagen de CD de los complementos del invitado* ([figura 26](#)).

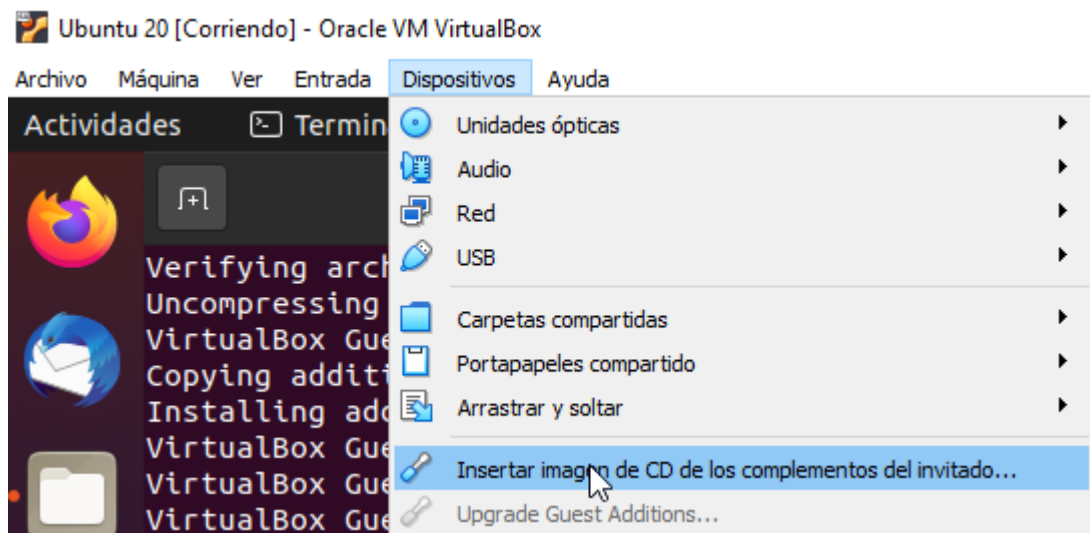


Figura 26

Nos preguntaran si queremos instalar las Guest Addition, le decimos que sí, y seguidamente empezara la instalación, una vez finalizada nos lo indicaran como vemos en la [figura 27](#).

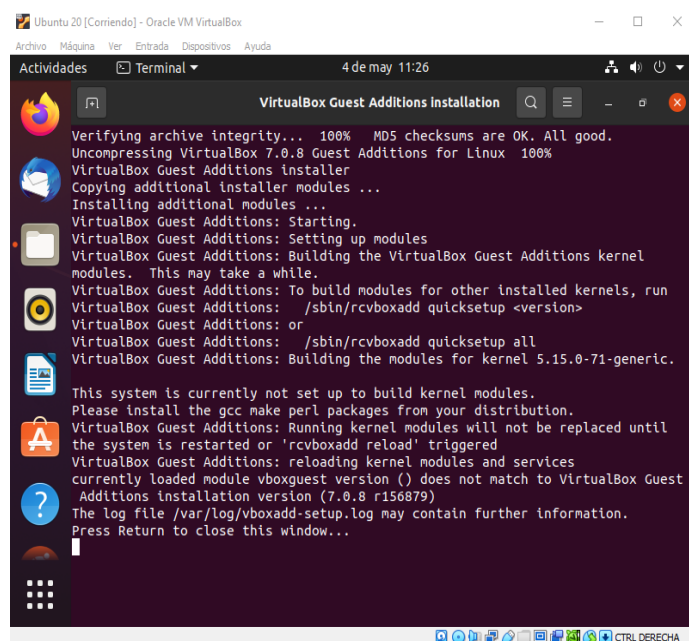


Figura 27

A continuación, actualizaremos las Guest Addition seleccionando *Dispositivos* → *Upgrade Guest Addition*.

Y ahora ya podemos ver en pantalla completa de una forma más cómoda.

1.5 Crear carpeta compartida

Vamos a crear una carpeta compartida entre nuestro Windows y la MV. Para ello Seleccionamos *Dispositivo* → *Carpeta compartidas* → *Preferencias de carpetas compartidas* (Figura 28).

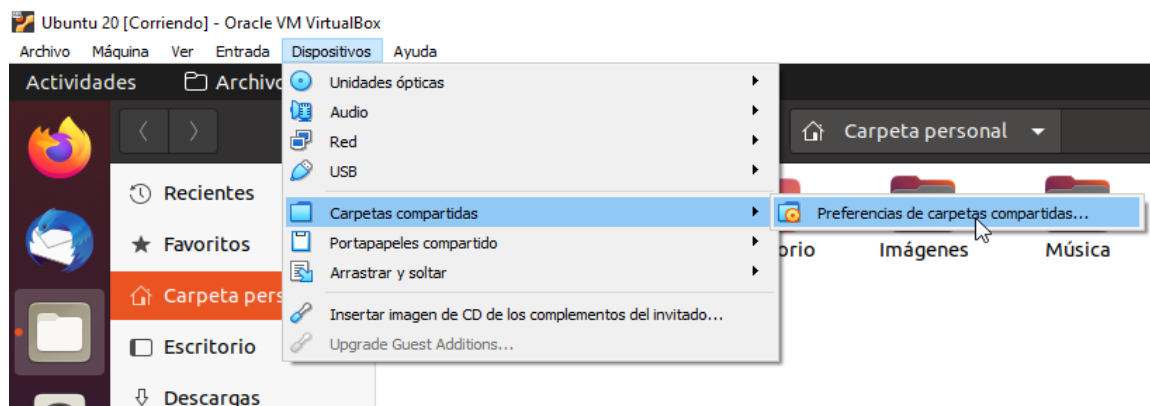


Figura 28

Se nos abrirá una nueva ventana (Figura 29)

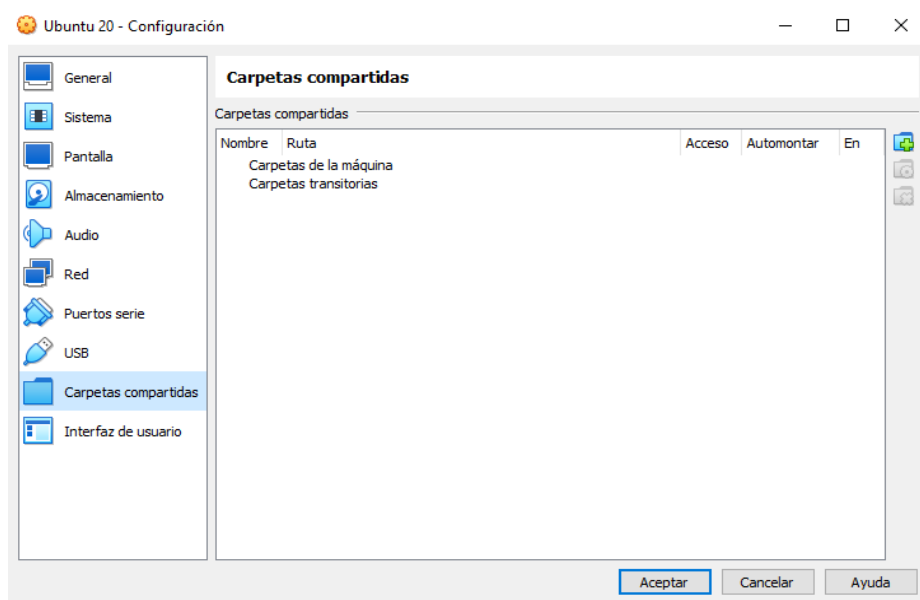


Figura 29

Ahora haremos click en el icono que hay en la parte superior derecha de una carpeta con un símbolo +, y en la imagen de la [figura 30](#) seleccionamos la ruta de la carpeta que vamos a compartir y rellenamos el resto de los campos igual que en la [figura 30](#).

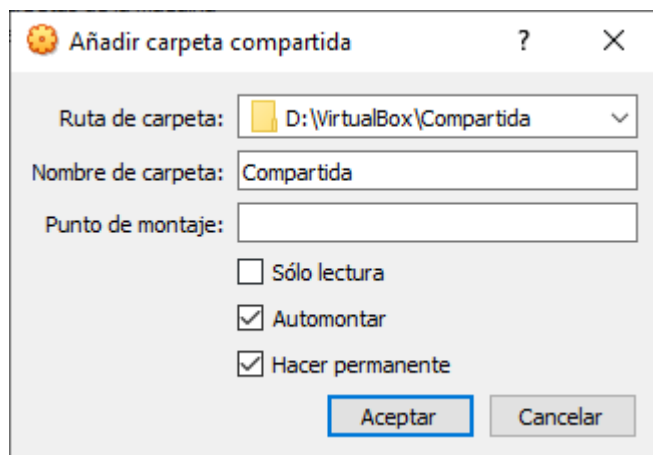


Figura 30

Una vez creada esta carpeta, deberemos ir a Windows y darle permisos. Buscamos la carpeta en la ruta que le hemos indicado, y son botón derecho sobre ella, le damos a *propiedades* y desmarcamos la casilla *Solo lectura (solo para archivos de la carpeta)* ([figura 31](#)), y aceptamos. Al aceptar nos aparece otra ventana, donde seleccionaremos *Aplicar cambios a esta carpeta y a todas las subcarpetas y archivos* ([figura 32](#)).

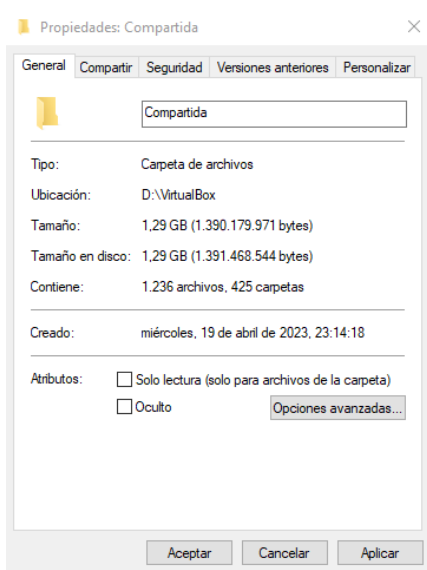


Figura 31

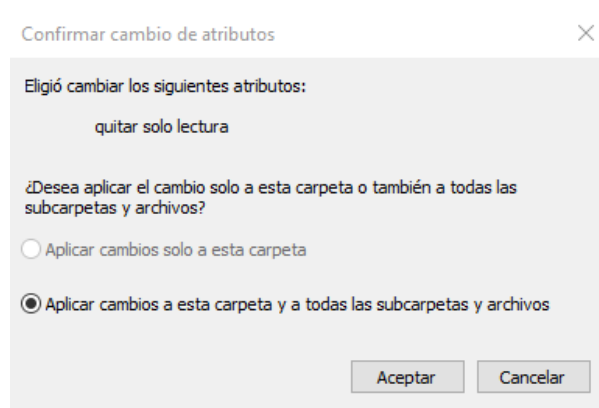


Figura 32

Ahora copiamos nuestro proyecto BDT dentro de esta carpeta (es solo para tenerla a mano).

Hay que tener en cuenta que a veces podemos perder los permisos que acabamos de aplicar, o incluso la anidación de la propia carpeta con nuestro Ubuntu, de ser así bastara con repetir estos pasos.

De todos modos, si hemos actualizado correctamente las Guest Addition, también podemos arrastrar y soltar si compartimos el *Portapapeles y Arrastrar y soltar* de manera bidireccional (recomiendo más este método).

1.6 Instalación de Apache, MySql y PHP

1.6.1 Apache

Si hemos instalado o tenemos un entorno grafico de Ubuntu, abriremos una consola pulsando Ctrl+Alt+t. Y ahora ya solo hay que seguir los comandos que se indican a continuación:

```
sudo apt update
```

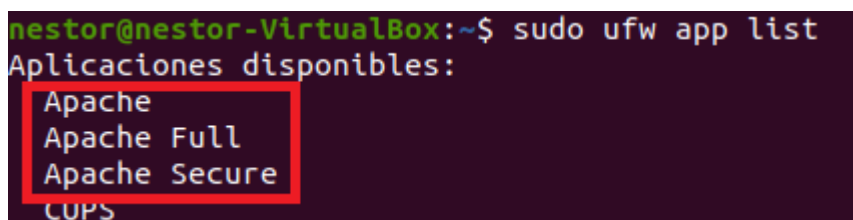
```
sudo apt install apache2
```

Con la instalación de apache2 es posible que nos pidan alguna confirmación para su instalación, le diremos que sí.

1.6.2 Actualizar el firewall

Ahora que ya hemos instalado Apache, tenemos que ajustar la configuración de nuestro firewall para poder usar tráfico HTTP y HTTPS. Para ello existe la herramienta UFW, la cual dispone de varios perfiles de aplicación que nos permitirán hacerlo. Podemos ver estos perfiles con el comando:

```
sudo ufw app list
```

 (figura 33)

```
nestor@nestor-VirtualBox:~$ sudo ufw app list
Aplicaciones disponibles:
Apache
Apache Full
Apache Secure
CUPS
```

Figura 33

¿Pero que son estos perfiles?

- Apache: este perfil abre solo el puerto 80, dedicado a un tráfico web normal (no cifrado)
- Apache secure: este perfil abre solo el puerto 443, dedicado a tráfico TLS/SSL (cifrado)
- Apache full: este perfil abre los puertos 80 y 443

Como estamos en un entorno local, una instalación nueva y no tenemos ningún certificado TSL/SSL configurado, abriremos el puerto 80, con el siguiente comando:

```
sudo ufw allow in "Apache"
```

Ahora si abrimos el navegador y escribimos <http://localhost> deberíamos ver algo como en la [figura 34](#).

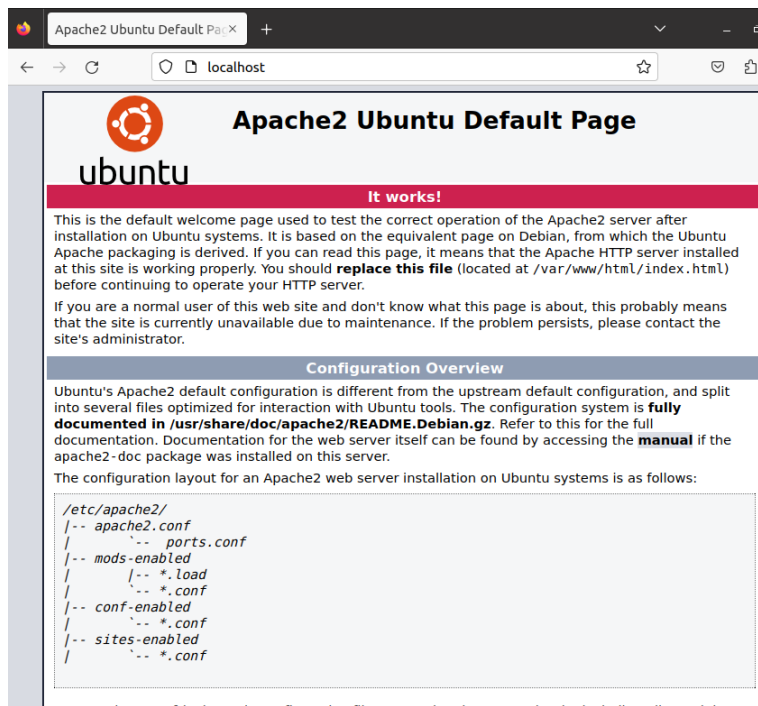


Figura 34

Si vemos esto, significa que todo ha ido bien y el servidor web estará listo.

1.6.3 MySQL

Ahora vamos a instalar uno de los servidores de base de datos más conocidos, para ello ejecutaremos el siguiente comando:

```
sudo apt install mysql-server
```

Es posible que en algún momento nos pidan confirmación, le diremos que sí, y continuaremos.

Para comprobar si la instalación ha sido correcta podemos ejecutar el comando `sudo mysql` y nos aparecerá una pantalla como la [figura 35](#). Y para salir de dicha pantalla bastara con escribir `exit`.

```
nestor@nestor-VirtualBox:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, 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> exit
Bye
nestor@nestor-VirtualBox:~$
```

Figura 35

Ahora tenemos que cambiar la contraseña del usuario root a 'admin' ya que la conexión que establece el proyecto con MySQL usa como usuario **root** y como contraseña **admin**

```
mysql -u root -p
```

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'admin';
```

```
FLUSH PRIVILEGES;
```

1.6.4 Instalar PHP

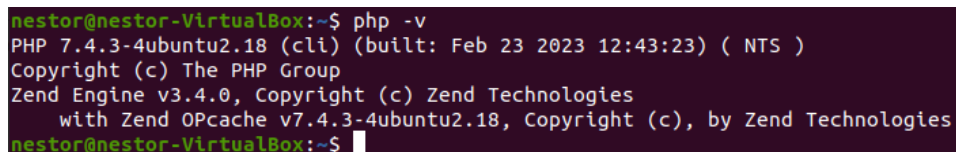
Ahora vamos a instalar el intérprete de php además de lo necesario para la conexión con nuestra base de datos. Para ello iremos ejecutando los siguientes comandos:

Y como en las anteriores veces, cuando nos pidan confirmación le diremos que sí.

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

Para ver que versión de php hemos instalado (ya que la aplicación está hecha en 8.2) podemos utilizar el siguiente comando. Y nos aparecerá un texto como el de la [figura 36](#).

```
php -v
```



```
nestor@nestor-VirtualBox:~$ php -v
PHP 7.4.3-4ubuntu2.18 (cli) (built: Feb 23 2023 12:43:23) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3-4ubuntu2.18, Copyright (c), by Zend Technologies
nestor@nestor-VirtualBox:~$
```

Figura 36

Como vemos no nos ha instalado la versión que necesitamos, no pasa nada. Continuamos ejecutando comandos. Lo primero será actualizar nuestro repositorio.

```
sudo apt update
```

```
sudo apt upgrade
```

Dependiendo de la versión, no preguntará si queremos reiniciar los servicios, podemos decirle que sí, sin problemas.

A continuación, instaláramos las dependencias requeridas y añadiremos el repositorio PHP:

```
sudo apt-get install software-properties-common gnupg2 -y
```

```
sudo add-apt-repository ppa:ondrej/php
```

Como siempre, cuando nos pidan confirmación le decimos que sí, en este caso hay que pulsar ENTER.

Ahora volvemos a actualizar el repositorio. Y ahora ya podemos instalar múltiples versiones de PHP.

Como la que queremos es la 8.2 lo haremos con el siguiente comando:

```
sudo apt-get install php8.2
```

Si ahora comprobamos la versión de php veremos que es la 8.2.* (comando php -v).

Seguramente ya tengamos esta versión por defecto, podemos comprobarlo con el comando

```
sudo update-alternatives --config php
```

De todos modos, recomiendo eliminar la versión anterior si no la necesitamos para nada, para ello usamos el comando:

```
sudo apt-get remove php7.*
```

Ahora instalamos un paquete que va a permitir ejecutar sentencias mysql en nuestra aplicación:

```
sudo apt install php8.2-mysqli
```

1.7 Creación de un host virtual en nuestra máquina local

A continuación, vamos a preparar nuestro host virtual, lo primero será crear un directorio para nuestro dominio, lo haremos con la siguiente comanda:

```
sudo mkdir /var/www/bdt
```

Ahora vamos a dar permisos al usuario con el que estamos conectados:

```
sudo chown -R $USER:$USER /var/www/bdt
```

Ahora generamos un archivo en blanco en el directorio *site-available* de Apache:

```
sudo nano /etc/apache2/sites-available/bdt.conf
```

Y en la ventana que se nos abra, debemos copiar lo siguiente:

```
<VirtualHost *:80>
    ServerName bdt
    ServerAlias www.bdt
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/bdt
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Ahora guardamos y salimos de esta pantalla pulsando **ctrl + o, enter, ctrl + x**

Hay que habilitar el nuevo host virtual, para ello ejecutamos las siguientes comandas:

```
sudo a2ensite bdt
```

```
sudo a2dissite 000-default
```

Y ahora reiniciamos el servicio Apache para que los cambios tengan efecto:

```
sudo systemctl reload apache2
```

Para comprobar que todo ha ido bien, podemos crear un archivo `index.html` e insertamos en siguiente texto en dicho archivo:

```
sudo nano /var/www/bdt/index.html
```

```
<h1>Funciona!</h1>
```

```
<p>Ya casi tenemos listo nuestro <strong>banco del tiempo</strong>.</p>
```

Guardamos cambios y salimos.

Y ahora desde el navegador web ponemos <http://localhost> y deberíamos ver algo como en la [figura 37](#).

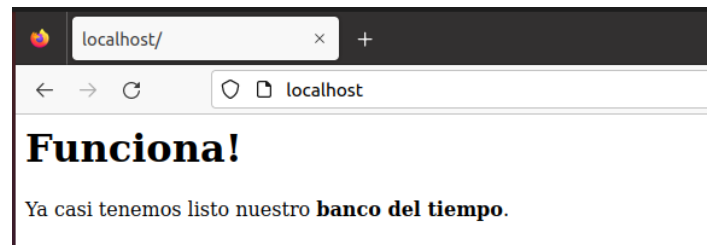


Figura 37

Si vemos esto, es que hemos montado nuestro servidor virtual de Apache correctamente. Pero aun no hemos terminado. Hay que cambiar el orden de prioridad en el cual apache interpretara las páginas. Para ello ejecutaremos el siguiente comando:

`sudo nano /etc/apache2/mods-enabled/dir.conf`

y nos tiene que quedar algo así ([figura 38](#)), siendo index.php el primero de la lista:

```
GNU nano 4.8 /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml inde
</IfModule>
```

Figura 38

Y reiniciamos el servidor apache:

`sudo systemctl reload apache2`

Bastara con arrastrar el contenido de la carpeta original bdt, a la carpeta `/var/www/bdt`

Por prevenir errores, vamos a dar permisos a la carpeta bdt;

`sudo chmod 777 -R /var/www/bdt`

Es hora de probar si funciona con la aplicación Banco del Tiempo. Basta con ir al navegador y poner en la ruta: localhost

Llegados a este punto, puede que todo haya ido bien, y veamos el Banco del Tiempo. Sin embargo, a veces Apache es caprichoso y nos mostrara una página en blanco, de ser así, abriremos el fichero `apache2.conf` con la siguiente instrucción:

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

Iremos al final del fichero y añadiremos lo siguiente:

```
<FilesMatch \.php$>
SetHandler application/x-httpd-php
</FilesMatch>
```

Guardamos y cerramos.

Ahora vamos a habilitar y deshabilitar unos módulos con el siguiente comando:

```
sudo a2dismod mpm_event && sudo a2enmod mpm_prefork && sudo a2enmod php8.2
```

Y por últimos tenemos que reiniciar el Apache2:

```
sudo service apache2 restart
```

Y ahora sí, si escribimos localhost en el navegador deberíamos ver algo como lo que se muestra en la [figura 39](#)

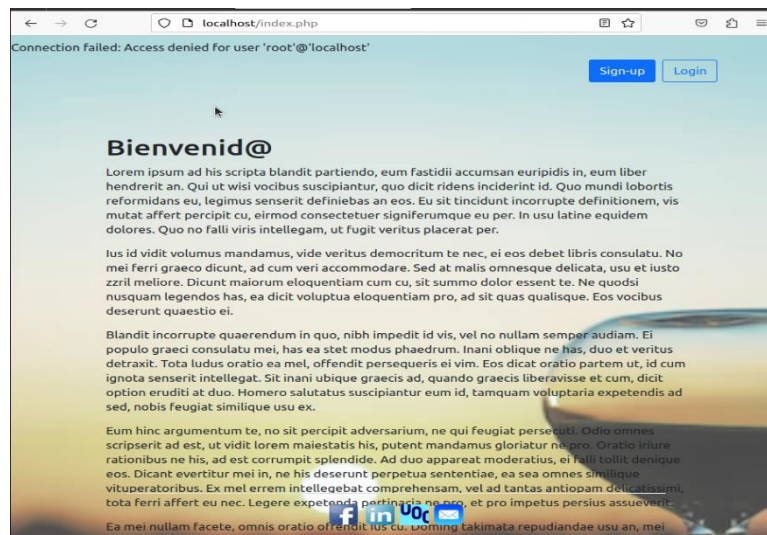


Figura 39

Como podemos ver, nos da un error de conexión, esto se debe a que aún no tenemos el schema de la base de datos listo, lo comentamos en el siguiente apartado ([13.10](#))

1.8 Instalación del certificado de seguridad

Vamos a instalar un certificado de seguridad para navegar por nuestra plataforma de una manera segura. Para ello instalaremos snapd, el cual nos ahorrara prácticamente todo el trabajo.

Primero vamos a comprobar si nuestra versión de Ubuntu trae instalada alguna versión de snapd, con el comando:

```
sudo snap install core; sudo snap refresh core
```

```
nestor@nestor-VirtualBox:~$ sudo snap install core; sudo snap refresh core
[sudo] contraseña para nestor:
Se ha instalado core 16-2.58.3 por Canonical✓
el snap "core" no tiene actualizaciones disponibles
nestor@nestor-VirtualBox:~$
```

Como podemos comprobar ya la tenemos instalada y no hay actualizaciones pendientes. Si por el contrario no estuviera instalada lo haríamos mediante el comando:

```
sudo apt install snapd
```

Lo siguiente que debemos hacer es asegurarnos que no tenemos ninguna versión de *certbot*, para ello ejecutamos el comando:

```
sudo apt remove certbot
```

Y ahora instalamos la versión clásica de *certbot*:

```
sudo snap install --classic certbot
```

Y lo siguiente será crear un acceso directo dentro de `/usr/bin` para que nos funcione desde cualquier sitio:

```
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Ahora debemos modificar el fichero `bdt.conf` que habíamos creado en la sección [13.7](#), lo dejaremos tal que así:

```
sudo nano /etc/apache2/sites-available/bdt.conf
```

```
<VirtualHost *:80>
  ServerName bdt
  ServerAlias bancodeltiempo2023.westeurope.cloudapp.azure.com ← esta es la línea a modificar
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/bdt
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

A continuación, vamos a configurar de manera automática apache, obtendremos los certificados y activamos https:

```
sudo certbot --apache
```

Se nos pedirá un correo electrónico, yo he puesto el de la plataforma (bancodeltiempo2023@gmail.com), Nos preguntara si aceptamos los términos y condiciones de uso, le diremos que si (Y), a continuación nos pregunta si queremos enviar al correo los certificados, podemos decirle que no (N). Y el último paso es elegir en cual de los dominios queremos activar, seleccionaremos el que diga:

```
bancodeltiempo2023.westeurope.cloudapp.azure.com
```

Y ya tenemos una navegación segura.

1.9 Creación de la base de datos

Llega el momento de la creación de la base de datos, las tablas, vistas, disparadores, etc.

Adjunto al proyecto hay un fichero (/bdt/schema.sql) el cual contiene todas las sentencias necesarias para la creación de todo lo anterior comentado. Bastara con ejecutarlas, recomendando, una a una en la consola de MySql como hemos visto en el apartado [13.6.3](#), para acceder a dicha consola, basta con escribir `sudo mysql` desde la terminal de Ubuntu.

1.9.1 Base de datos

Con el siguiente comando crearemos la base de datos y le daremos uso para que todo lo que se ejecute a continuación sea en dicha base de datos.

```
CREATE DATABASE `bdt` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */  
/*!80016 DEFAULT ENCRYPTION='N' */;
```

```
USE bdt;
```

1.9.2 Tablas

Ahora vamos a crear todas las tablas que necesitaremos, por precaución aconsejo ejecutarlas una a una. Y sobre todo en el mismo orden.

```
CREATE TABLE `role` (  
  `id` int NOT NULL,  
  `roleName` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `id_UNIQUE` (`id`),  
  UNIQUE KEY `role_name_UNIQUE` (`roleName`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `categorias` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(45) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `nombre_UNIQUE` (`nombre`),  
  UNIQUE KEY `id_UNIQUE` (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `sub_categorias` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(45) NOT NULL,  
  `categoriasId` int NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `nombre_UNIQUE` (`nombre`),  
  UNIQUE KEY `id_UNIQUE` (`id`),  
  KEY `FK_subcat_categorias_idx` (`categoriasId`),  
  CONSTRAINT `FK_subcat_categorias` FOREIGN KEY (`categoriasId`) REFERENCES `categorias` (`id`) ON  
  DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```

CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(45) NOT NULL,
  `passw` varchar(100) DEFAULT NULL,
  `email` varchar(100) NOT NULL,
  `nombre` varchar(45) DEFAULT NULL,
  `apellidos` varchar(45) DEFAULT NULL,
  `poblacion` varchar(100) DEFAULT NULL,
  `horas` int DEFAULT '10',
  `role` int NOT NULL DEFAULT '2',
  PRIMARY KEY (`id`),
  UNIQUE KEY `username_UNIQUE` (`username`) /*!80000 INVISIBLE */ ,
  UNIQUE KEY `email_UNIQUE` (`email`),
  KEY `FK_users_role_idx` (`role`),
  CONSTRAINT `FK_users_role` FOREIGN KEY (`role`) REFERENCES `role` (`id`) ON DELETE RESTRICT ON
  UPDATE RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=47 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
CREATE TABLE `ofertas` (
  `id` int NOT NULL AUTO_INCREMENT,
  `userId` int NOT NULL,
  `categoriald` int NOT NULL,
  `subCatId` int NOT NULL,
  `descripcion` varchar(240) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_oferta_users_idx` (`userId`),
  KEY `FK_oferta_categorias_idx` (`categoriald`,`subCatId`),
  KEY `FK_oferta_subCategoria_idx` (`subCatId`),
  CONSTRAINT `FK_oferta_subCategoria` FOREIGN KEY (`subCatId`) REFERENCES `sub_categorias` (`id`),
  CONSTRAINT `FK_oferta_users` FOREIGN KEY (`userId`) REFERENCES `users` (`id`) ON DELETE CASCADE
  ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `valoraciones` (
  `userId` int NOT NULL,
  `valoracion` decimal(4,2) DEFAULT '0.00',
  `votos` int DEFAULT '0',
  PRIMARY KEY (`userId`),
  UNIQUE KEY `userId_UNIQUE` (`userId`),
  KEY `idUser_idx` (`userId`),
  CONSTRAINT `FK_valoraciones_users` FOREIGN KEY (`userId`) REFERENCES `users` (`id`) ON DELETE
  CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

1.9.3 Triggers o disparadores

Este trigger se encarga de, al crear un usuario nuevo, crea un nuevo registro en la tabla valoraciones con el *userId* del nuevo usuario, y le asigna una valoración = 0 y uno votos = 0.

Dependiendo de la versión de MySQL que se nos haya instalado deberemos usar una de las 2 opciones:

Opción 1:

```
CREATE DEFINER='root'@'localhost' TRIGGER `users_AFTER_INSERT` AFTER INSERT ON `users` FOR EACH ROW INSERT INTO valoraciones (userId, valoracion, votos) VALUES(new.id, 0, 0);
```

Opción 2:

```
CREATE DEFINER='root'@'localhost' TRIGGER `users_AFTER_INSERT` AFTER INSERT ON `users` FOR EACH ROW BEGIN  
    INSERT INTO valoraciones (userId, valoracion, votos) VALUES(new.id, 0, 0);  
END;
```

1.9.4 Vistas

```
CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost' SQL SECURITY DEFINER VIEW  
`bdt`.`v_ofertas` AS select  
    `bdt`.`ofertas`.`id` AS `id`,  
    `bdt`.`ofertas`.`userId` AS `userId`,  
    `bdt`.`ofertas`.`categoriald` AS `categoriald`,  
    `bdt`.`categorias`.`nombre` AS `catNombre`,  
    `bdt`.`ofertas`.`descripcion` AS `descripcion`,  
    `bdt`.`users`.`nombre` AS `nombre`,  
    `bdt`.`users`.`apellidos` AS `apellidos`,  
    `bdt`.`users`.`username` AS `username`,  
    `bdt`.`users`.`email` AS `email`,  
    `bdt`.`users`.`poblacion` AS `poblacion`,  
    `bdt`.`valoraciones`.`valoracion` AS `valoracion`,  
    `bdt`.`valoraciones`.`votos` AS `votos`,  
    `bdt`.`sub_categorias`.`id` AS `subCatId`,  
    `bdt`.`sub_categorias`.`nombre` AS `subCatNombre`  
from (((`bdt`.`ofertas`  
    join `bdt`.`users` on((`bdt`.`ofertas`.`userId` = `bdt`.`users`.`id`)))  
    join `bdt`.`categorias` on((`bdt`.`ofertas`.`categoriald` = `bdt`.`categorias`.`id`)))  
    join `bdt`.`sub_categorias` on((`bdt`.`ofertas`.`subCatId` = `bdt`.`sub_categorias`.`id`)))  
    join `bdt`.`valoraciones` on((`bdt`.`users`.`id` = `bdt`.`valoraciones`.`userId`))));
```

```

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW
`bdt`.`v_users` AS select
  `bdt`.`users`.`id` AS `id`,
  `bdt`.`users`.`username` AS `username`,
  `bdt`.`users`.`passwd` AS `passwd`,
  `bdt`.`users`.`email` AS `email`,
  `bdt`.`users`.`nombre` AS `nombre`,
  `bdt`.`users`.`apellidos` AS `apellidos`,
  `bdt`.`users`.`poblacion` AS `poblacion`,
  `bdt`.`users`.`horas` AS `horas`,
  `bdt`.`role`.`id` AS `roleId`,
  `bdt`.`role`.`roleName` AS `roleName`,
  `bdt`.`valoraciones`.`userId` AS `userId`,
  `bdt`.`valoraciones`.`valoracion` AS `valoracion`,
  `bdt`.`valoraciones`.`votos` AS `votos`
from ((`bdt`.`users`
  join `bdt`.`role` on((`bdt`.`users`.`role` = `bdt`.`role`.`id`)))
  join `bdt`.`valoraciones` on((`bdt`.`users`.`id` = `bdt`.`valoraciones`.`userId`)));

```

1.9.5 Inserts

Añado unos inserts para que la base de datos no este vacia, no hay porque usarlos, pero es una manera rápida y cómoda de tener varios usuarios y diferentes ofertas:

```

INSERT INTO role (id, roleName) VALUES(1,'ADMIN'),(2, 'USER'),(3,'BANNED');

```

```

INSERT INTO users (id, username, passwd, email, nombre, apellidos, poblacion, horas, role)
VALUES (1, "Nes", "BpeoQa/YNeJFk9/ZW9oQCw==", "nestito__@hotmail.com", "Nestor", "ibanez", "Grn", 10, 1),
      (2, "admin", "BpeoQa/YNeJFk9/ZW9oQCw==", "admin@gmail.com", "nombre", "Apellido", "Bcn", 10, 1),
      (3, "user1", "BpeoQa/YNeJFk9/ZW9oQCw==", "user1@gmail.com", "nombre1", "Apellido1", "Bcn", 10, 2),
      (4, "user2", "BpeoQa/YNeJFk9/ZW9oQCw==", "user2@gmail.com", "nombre2", "Apellido2", "Bcn", 10, 2),
      (5, "user3", "BpeoQa/YNeJFk9/ZW9oQCw==", "user3@gmail.com", "nombre3", "Apellido3", "Bcn", 10, 3);

```

```

INSERT INTO categorias(id, nombre) VALUES (1, "Informática"), (2, "Música"), (3, "Idiomas");

```

```

INSERT INTO sub_categorias (id, nombre, categoriasId)
VALUES (1, " Programación c++",1),(2, " Programación Java", 1), (3, "Programación Python", 1),
      (4, "Guitarra", 2), (5, "Piano", 2), (6, "Flauta", 2),
      (7, "Catalán", 3), (8, "Inglés", 3), (9, "Francés", 3);

```

```

INSERT INTO ofertas (userId, categoriaId, subCatId, descripcion)
VALUES (1, 1, 1, "Clases de programación c++"),
      (1, 1, 2, "Clases de programación Java"),
      (1, 1, 3, "Clases de programación Python"),
      (1, 2, 4, "Clases de Guitarra"),
      (1, 2, 5, "Clases de Piano"),
      (1, 2, 6, "Clases de Flauta"),
      (1, 3, 7, "Clases de Catalán"),
      (1, 3, 8, "Clases de Inglés"),
      (1, 3, 9, "Clases de Frances"),
      (2, 1, 1, "Clases de programación c++"),
      (2, 1, 2, "Clases de programación Java"),
      (2, 1, 3, "Clases de programación Python"),
      (2, 2, 4, "Clases de Guitarra"),
      (2, 2, 5, "Clases de Piano"),
      (2, 2, 6, "Clases de Flauta"),
      (2, 3, 8, "Clases de Inglés"),
      (2, 3, 9, "Clases de Frances"),
      (3, 1, 1, "Clases de programación c++"),
      (3, 1, 3, "Clases de programación Python"),
      (3, 2, 4, "Clases de Guitarra"),
      (3, 2, 6, "Clases de Flauta"),
      (3, 3, 7, "Clases de Catalán"),
      (4, 1, 2, "Clases de programación Java"),
      (4, 2, 5, "Clases de Piano"),
      (4, 3, 7, "Clases de Catalán"),
      (5, 3, 9, "Clases de Frances");

```

Y ahora si volvemos al navegador y ponemos *localhost* ya vemos una página web sin errores y totalmente funcional, podemos crear un nuevo usuario, o usar alguno de los existentes. La contraseña para todos los existentes es **admin**, en los inserts se ve la contraseña encriptada.

2 Alojamiento en Azure

Gracias al correo de Azure pude alojar la web en este servicio. Para ello he creado una máquina virtual a la cual le he instalado un Ubuntu server 20, como se puede ver en la [figura 40](#).

The screenshot shows the 'Configuración de la máquina virtual' (Virtual Machine Configuration) page in the Azure portal. At the top, a yellow warning banner states: 'Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.' (When changing basic options, the selections made can be reset. Review all options before creating the virtual machine.)

The configuration options are as follows:

- Opciones de disponibilidad:** No se requiere redundancia de la infraestructura (dropdown)
- Tipo de seguridad:** Estándar (dropdown)
- Imagen:** Ubuntu Server 20.04 LTS - x64 Gen2 (servicios gratuitos elegibles) (dropdown). Below it, links for 'Ver todas las imágenes' and 'Configurar la generación de máquinas virtuales' are visible.
- Arquitectura de VM:** x64 (radio button selected, Arm64 is also an option).
- Ejecución de Azure Spot con descuento:** (checkbox, currently unchecked).

A blue information banner below the options states: 'You are in the free trial period. Costs associated with this VM can be covered by any remaining credits on your subscription. Más información'.

The configuration continues with:

- Tamaño:** Standard_B1s - 1 vcpu, 1 GiB de memoria (8,76 US\$/mes) (servicios gratuito... (dropdown). Below it, a link 'Ver todos los tamaños' is visible.
- Cuenta de administrador:**
 - Tipo de autenticación:** Clave pública SSH (radio button selected, Contraseña is also an option).
 - A blue information box below the authentication type states: 'Ahora, Azure genera automáticamente un par de claves SSH y le permite almacenarlo para usarlo en el futuro. Es una forma rápida, sencilla y segura de conectarse a la máquina virtual.'
- Nombre de usuario:** azureuser (text input, with a green checkmark icon).
- Origen de clave pública SSH:** Generar un par de claves nuevo (dropdown).

Figura 40

Los pasos para la configuración de este Ubuntu server son los mismos que para la instalación local, es decir, se necesita instalar MySQL, PHP, etc. punto [13.6](#)

Para acceder a la plataforma alojada en Azure hay que conectar a través del navegador web poniendo una de las siguientes url:

<http://13.94.254.56/index.php> (sin certificado de seguridad)

<https://bancodeltiempo2023.westeurope.cloudapp.azure.com/> (con certificado, la configuración se explica en el apartado [13.8](#))

El funcionamiento es el mismo que en local. Sin embargo, dado que Azure es una cuenta gratuita puede que sea algo más lenta (Al final he decidido pagar para que vaya mas fluida, tiene 2cpu y 4gb de ram)

También esta alojada en AWS, esto no lo había hecho nunca y me habían comentado que no era sencillo, aun así, he querido probar, y la verdad es que me ha resultado mas sencillo de hacer que en Azure, de haberlo sabido lo hubiera hecho aquí y no lo hubiera hecho en Azure, la url es:

<http://54.237.146.182/index.php>

3 Ficheros y procedimientos

A continuación, se detallan los ficheros y procedimientos más importantes. Primero comentaré el fichero y seguidamente las funciones que este contenga.

3.1 Índice

Fichero	index.php
Descripción	Este fichero se encargará de la redirección de las url a las que se deben llamar en cada momento.
Parámetros	\$action → acción a realizar, es decir, si redirecciona hacia un controlador (ctl) o hacia una vista (view). \$option → controlador o vista a la que se debe redireccionar.
Funcionamiento	El administrador de la plataforma debe rellenar estos valores para que luego el controlador cree la conexión a la base de datos, desde la cual se harán todas las consultas.

3.2 Modelos

Fichero	/bdt/model/role.php
Descripción	Este fichero contiene el modelo de los roles de usuario en la base de datos.
Parámetros	\$id → id del rol \$roleName → nombre del rol.
Funcionamiento	Se encarga de la gestión de roles.

Fichero	/bdt/model/valoraciones.php
Descripción	Este fichero contiene el modelo de las subcategorías
Parámetros	\$valUserId → id del usuario. \$valoracion → valoracion del usuario. \$votos → cantidad de veces que el usuario ha sido votado.
Funcionamiento	Se encarga de la gestión de las valoraciones.

Fichero	/bdt/model/Categoria.php
Descripción	Este fichero contiene el modelo de las categorías
Parámetros	\$id → id de la categoría. \$catNombre → nombre de la categoría.
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea categorías, y manipula sus datos.

Función	getCategorias(\$orderBy = "nombre", \$logSql=false): bool array null
Descripción	función que devuelve las categorías.
Parámetros	<p>\$ orderBy → Columna por la que se desean ordenar las categorías.</p> <p>\$logSql → Boleano para grabar el log del resultado de la ejecución.</p> <p>\$query → Sentencia que se va a ejecutar, en este caso:</p> <p><i>"SELECT * FROM categorias ORDER BY {\$orderBy};"</i>;</p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función getRows , la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de categorías.
Salida	<p>Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]]</p> <p>False si algo ha fallado.</p> <p>Graba un log si se lo indicamos o si ha fallado algo.</p>

Fichero	/bdt/model/subCategoria.php
Descripción	Este fichero contiene el modelo de las subcategorías
Parámetros	<p>\$id → id de la subcategoría.</p> <p>\$nombre → nombre de la subcategoría.</p> <p>\$catId → id de la categoría a la que pertenece esta subcategoría.</p>
Funcionamiento	Se encarga de la gestión de las subcategorías.

Función	getSubCategorias (\$orderBy = "nombre", \$logSql=false): bool array null
Descripción	función que devuelve las categorías.
Parámetros	<p>\$ orderBy → Columna por la que se desean ordenar las categorías.</p> <p>\$logSql → Boleano para grabar el log del resultado de la ejecución.</p> <p>\$query → Sentencia que se va a ejecutar, en este caso:</p> <p><i>"SELECT * FROM sub_categorias ORDER BY {\$orderBy};"</i>;</p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función getRows , la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de sub_categorias.
Salida	<p>Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]]</p> <p>False si algo ha fallado.</p> <p>Graba un log si se lo indicamos o si ha fallado algo.</p>

Fichero	/bdt/model/Usuario.php
Descripción	Este fichero contiene el modelo del usuario.
Parámetros	<p>use Role → herencia de Role.</p> <p>use Valoraciones → herencia de Valoraciones.</p> <p>\$id → id del usuario.</p> <p>\$username → username del usuario.</p> <p>\$password → contraseña del usuario.</p> <p>\$email → correo electrónico del usuario.</p> <p>\$nombre → nombre del usuario.</p> <p>\$apellidos → apellidos del usuario.</p> <p>\$poblacion → población del usuario.</p> <p>\$horas → horas que le quedan disponibles al usuario para aprender de alguien.</p>
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea categorías, y manipula sus datos.

Función	altaUsuario()
Descripción	Inserta un usuario de la base de datos.
Parámetros	<p>\$sql → sentencia INSERT a ejecutar, en este caso:</p> <p><i>"INSERT INTO users (username, passw, email, nombre, apellidos, poblacion, horas)</i> <i>VALUES ('{\$this->username}', '{\$this->password}', '{\$this->email}', '{\$this->nombre}', '{\$this->apellidos}', '{\$this->poblacion}', 10);"</i></p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Función	getUserById (\$userId)
Descripción	Busca un usuario por su id.
Parámetros	<p>\$userId → id del usuario a buscar</p> <p>\$sql → sentencia SELECT a ejecutar, en este caso:</p> <p><i>"SELECT * FROM v_users WHERE username='{\$ \$userId}';"</i></p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará, y rellenaremos los campos del propio modelo Usuario
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Función	modificarUsuario()
Descripción	Modifica un usuario de la base de datos.
Parámetros	\$sql → sentencia UPDATE a ejecutar, en este caso 2 sentencias: La primera UPDATEARA la tabla users con los nuevos valores, y la segunda UPDATEARA la tabla valoraciones con los valores correspondientes a dicho usuario.
Funcionamiento	Se crean las sentencia indicadas, y se envían a la función executeSql , la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	getUserByUserName (\$userName)
Descripción	Busca un usuario por su nombre.
Parámetros	\$userName → username del usuario a buscar \$sql → sentencia SELECT a ejecutar, en este caso: <i>"SELECT * FROM v_users WHERE username='{ \$userName }';"</i>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará, y rellenaremos los campos del propio modelo Usuario
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	eliminarUsuario(\$userId)
Descripción	Elimina una oferta de la base de datos
Parámetros	\$userId → id del usuario a eliminar. \$sql → sentencia DELETE a ejecutar, en este caso: <i>"DELETE FROM users WHERE id=\$userId;"</i>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	updateValoracion (\$userId, \$valoracion, \$voto)
Descripción	Elimina una oferta de la base de datos
Parámetros	\$userId → id del usuario a valorar. \$valoracion → nueva valoración que se le asigna. \$votos → nuvos votos que se le asignan. \$sql → sentencia UPDATE a ejecutar, en este caso: <i>"UPDATE valoraciones SET valoracion = '". \$valoracion. "', votos = '". \$voto. " WHERE userId='". \$userId. "';"</i>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	pagarHoras(\$selectedUserId, \$horas)
Descripción	Serve para que un usuario X pague n horas a otro usuario Z
Parámetros	<p>\$selectedUserId → id del usuario al que se le van a pagar las horas.</p> <p>\$horas → cantidad de horas a pagar</p> <p>Esta función ejecuta 2 sentencias, la primera sumara las horas al usuario Z y la segunda restara las horas al usuario X</p> <p>\$sql → sentencia UPDATE a ejecutar, en este caso:</p> <p>"UPDATE users SET horas = horas - ".\$horas." WHERE id='".\$this->id.'";"</p> <p>"UPDATE users SET horas = horas + ".\$horas."</p> <p>WHERE id='".\$selectedUserId.'";"</p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Fichero	/bdt/model/ Oferta.php
Descripción	Este fichero contiene el modelo de las ofertas
Parámetros	<p>use Categoria → herencia de Categoria.</p> <p>use subCategoria → herencia de subCategoria.</p> <p>\$id → id de la oferta.</p> <p>\$userId → id del usuario propietario de la oferta.</p> <p>\$categoryId → id de la categoría a la que pertenece la oferta.</p> <p>\$subCatId → id de la subcategoría a la que pertenece la oferta.</p> <p>\$descripcion → texto descriptivo de la oferta.</p>
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea ofertas, y manipula sus datos. Esta clase tiene distintas funciones que detallamos a continuación.

Función	insertarOferta ()
Descripción	Inserta una oferta en la base de datos
Parámetros	<p>\$sql → sentencia INSERT a ejecutar, en este caso:</p> <p>"INSERT INTO ofertas (userId, categoryId, subCatId, descripcion)</p> <p>VALUES ('\$this->userId', '{\$this->categoryId}', '{\$this->subCatId}',</p> <p>'{\$this->descripcion}');" "</p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Función	modificarOferta ()
Descripción	modifica una oferta existente en la base de datos
Parámetros	<p>\$ofertald → id de la oferta a modificar.</p> <p>\$categoriald → id de la nueva categoría que se le asignara a la oferta.</p> <p>\$subCatId → id de la nueva subcategoría que se le asignara a la oferta.</p> <p>\$descripcion → texto descriptivo que se le asignara a la oferta.</p> <p>\$sql → sentencia INSERT a ejecutar, en este caso:</p> <pre> UPDATE ofertas SET categoriald = ".\$categoriald.", subCatId = ".\$subCatId.", descripcion = ".\$descripcion." WHERE id=".\$ofertald."; </pre>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Función	eliminarOferta ()
Descripción	Elimina una oferta de la base de datos
Parámetros	<p>\$ofertald → id de la oferta a eliminar.</p> <p>\$sql → sentencia DELETE a ejecutar, en este caso:</p> <pre> DELETE FROM ofertas WHERE id=\$ofertald; </pre>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función executeSql , la cual la ejecutará
Salida	<p>true si todo ha ido bien.</p> <p>false si algo ha fallado.</p>

Función	autoFillOferta ()
Descripción	Genera una variable oferta
Parámetros	<p>\$ofertald → id de la oferta a generar.</p> <p>\$sql → sentencia SELECT a ejecutar, en este caso:</p> <pre> SELECT * FROM ofertas WHERE id='{ \$ofertald }'; </pre>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función getRow , la cual la ejecutará
Salida	false si algo ha fallado.

3.3 Controladores

Fichero	/bdt/controller/config.php
Descripción	En este fichero se parametriza la configuración de acceso a la base de datos
Parámetros	\$hostCfg, \$usernameCfg, \$passwordCfg, \$dbnameCfg, \$portCfg
Funcionamiento	El administrador de la plataforma debe rellenar estos valores para que luego el controlador cree la conexión a la base de datos, desde la cual se harán todas las consultas.

Fichero	/bdt/controller/DAOController.php
Descripción	En él se crea la conexión a la base de datos, todas las consultas realizadas en la plataforma pasan por aquí.
Parámetros	\$host, \$username, \$password, \$dbname, \$port
Funcionamiento	Recibe una consulta a la base de datos, la realiza, gestiona errores, y devuelve el resultado. Al final del fichero se crean 2 conexiones, una "directa" (\$conn = new DAOController()), la cual ha de ser referenciada en cada uno de los ficheros que se usa, algo que resulta un tanto incomodo y por eso he decidido usarla así solo en 3 ficheros (/bdt/controller/ajax_ctl/)a modos de ejemplo, para el resto de consultas he puesto la conexión en una variable de sesión y es usada desde dicha variable.
Salida	-----

Función	initDb(): void
Descripción	Establece la conexión a la base de datos.
Parámetros	\$host, \$username, \$password, \$dbname, \$port
Funcionamiento	Recibe los parámetros indicados, y establece la conexión
Salida	-----

Función	getConn()
Descripción	Un getter que devuelve la conexión
Parámetros	\$this->conn
Salida	\$this->conn

Función	getRow(\$sql, \$logSql=false): bool array null
Descripción	función que recibe una sentencia sql, la ejecuta y devuelve el resultado
Parámetros	\$sql → sentencia SELECT a ejecutar \$logSql → Boleano para grabar el log del resultado de la ejecución.
Salida	Resultado de la sentencia sql en formato {key, value} False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	executeSql(\$sql, \$logSql=false): mysqli_result bool Exception
Descripción	función que recibe una sentencia sql y la ejecuta
Parámetros	\$sql → sentencia INSERT, UPDATE, DELETE a ejecutar \$logSql → Boleano para grabar el log del resultado de la ejecución
Salida	Resultado de la sentencia sql False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	getRows(\$sql, \$logSql=false): bool array null
Descripción	función que recibe una sentencia sql, la ejecuta y devuelve el resultado
Parámetros	\$sql → sentencia SELECT a ejecutar \$logSql → Boleano para grabar el log del resultado de la ejecución.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	getLastInsertedId(): bool int null
Descripción	Función que devuelve el ultimo id generado en la base de datos
Parámetros	-----
Salida	Ultimo id insertado en la base de datos

Función	showAlert(\$msg): void
Descripción	Función que mostrara una alerta (en formato JavaScript) por pantalla.
Parámetros	\$msg → Mensaje que se mostrara en la alerta.
Salida	-----

Fichero	/bdt/controller/contactar_ctl.php
Descripción	Controlador encargado del envío de correos cuando un usuario X desea hacerle saber a otro usuario Z que está interesado en una de sus ofertas.
Parámetros	\$userId → usuario que quiere contactar \$to → usuario con el que se quiere contactar
Funcionamiento	El fichero enviara un email avisando al usuario \$to que el usuario \$userId está interesado en una de sus ofertas.
Salida	Email enviado o una alerta advirtiéndolo de lo contrario.

Fichero	/bdt/controller/eliminarUsuario_ctl.php
Descripción	Controlador que recibe el id de una oferta y elimina dicha oferta de la base de datos.
Parámetros	\$ofertald → id del usuario que se desea eliminar
Funcionamiento	El fichero recibe el identificador de un usuario, se crea una variable de tipo Usuario, la cual es un modelo de <u>Usuario</u> que contiene la función de eliminación de usuarios (<u>eliminarUsuario(\$userId)</u>), esta función es llamada pasándole el identificador correspondiente y se realiza la eliminación.
Salida	Nos devolverá a la página de origen, el listado manejado por un administrador dado que solo un administrador puede eliminar usuarios.

Fichero	/bdt/controller/insertarOferta_ctl.php
Descripción	Controlador que recibe una nueva categoría, una subcategoría y una descripción de la oferta y inserta un nuevo registro en la base de datos.
Parámetros	\$selCat → id de la categoría a la que pertenecerá la nueva oferta. \$subCatId → id de la subcategoría a la que pertenecerá la nueva oferta. \$desc → descripción de la nueva oferta.
Funcionamiento	Se comprueba que se hayan seleccionado una categoría y una subcategoría (de lo contrario se muestra un mensaje en formato JS alert) y se introduce un nuevo registro en la base de datos mediante la función <u>insertarOferta()</u> .
Salida	-----

Fichero	/bdt/controller/insertarCategoria_ctl.php
Descripción	Controlador que recibe una nueva categoría y/o una nueva subcategoría y la inserta en la base de datos
Parámetros	\$categoria → nombre de la nueva categoría \$subCat → nombre de la nueva subcategoría
Funcionamiento	Se comprueba que la \$categoria no sea null, ya que cualquier subcategoría ha de tener una categoría asociada. Se comprueba si la categoría existe, de ser así, se entiende que lo que se va a insertar será una subcategoría, se toma el id de la categoría, de lo contrario se hace el insert de la categoría. Si la categoría existe, se comprueba que la subcategoría no exista, de ser así se hace el insert de la nueva subcategoría
Salida	-----

Fichero	/bdt/controller/logged_ctl.php
Descripción	Controlador que se añade a todas las vistas para evitar que "se nos cuelen por detrás"
Parámetros	\$_SESSION['logged'] → variable de sesión que nos indica si hay un usuario logueado o no.
Funcionamiento	Se comprueba si la variable \$_SESSION['logged'] NO existe y si es null, de ser así, significa que no hay ningún usuario logueado y no podrán acceder a paginas que no se deba desde F12. Cualquier intento de acceder mediante la inspección de elementos de la plataforma, lo devolverá a la pagina de bienvenida.

Salida	-----
Fichero	/bdt/controller/login_ctl.php
Descripción	Controlador encargado de la gestión de log-in de usuarios.
Parámetros	\$usuario_req → username del usuario que intenta hacer log-in. \$password → contraseña del usuario que intenta hacer log-in.
Funcionamiento	Se recogen el username y la contraseña proporcionada, mediante la función validarUserPassword(\$usuario_req, \$password) , se comprueba que el usuario existe y que rol tiene, en caso que el rol sea 1 o 2 (admin o user) se le da paso a la plataforma, en caso que el rol sea 3 (baneado) se le avisa mediante una alerta y no se le da paso.
Salida	-----

Fichero	/bdt/controller/personalInfo_ctl.php
Descripción	Controlador encargado de la información personal de los usuarios.
Parámetros	\$enc → variable de tipo Encriptador . \$userId → id del usuario a modificar. \$userName → userName del usuario a modificar. \$emailReg → email del usuario a modificar. \$password → contraseña actual del usuario a modificar. \$newPass = → nueva contraseña del usuario a modificar. \$repeatNewPass → comprobación de la nueva contraseña \$nombreReg → nombre del usuario a modificar. \$apellidos → apellidos del usuario a modificar. \$poblacion → población del usuario a modificar. \$valoracion → valoración del usuario a modificar. \$votos → votos del usuario a modificar. \$horas → horas del usuario a modificar. \$role → rol del usuario a modificar.
Funcionamiento	En caso de que el usuario desee cambiar su contraseña, se le hace introducir dos veces la nueva contraseña y que ambas coincidan. Si el usuario no quiere cambiar su contraseña, se comprueba que la antigua no sea nula. El resto de los campos que necesitan algún tipo de verificación, como el formato del email, se verifican en la propia vista (/bdt/view/personalInfo_view.php) Finalmente se llama a la función modificarUsuario , la cual realizara las modificaciones pertinentes.
Salida	-----

Fichero	/bdt/controller/validarUserPassword_ctl.php
Descripción	Controlador encargado de la comprobación de la existencia de un usuario con la correspondiente contraseña.
Parámetros	\$userName → email del usuario que intenta hacer log-in. \$password → contraseña del usuario que intenta hacer log-in.
Funcionamiento	La función recibe los dos parámetros arriba indicados, mediante el \$userName busca el usuario correspondiente, si lo encuentra
Salida	-----

Fichero	/bdt/controller/register_ctl.php
Descripción	ontrolador encargado del registro de usuarios.
Parámetros	\$nombreReg → nombre del nuevo usuario. \$apellidosReg → apellidos del nuevo usuario. \$userNameReg → username del nuevo usuario. \$passwordReg → contraseña del nuevo usuario. \$poblacionReg → población del nuevo usuario. \$emailReg → email del nuevo usuario.
Funcionamiento	El fichero recibe los datos del usuario que se quiere registrar en la plataforma, con estos datos se crea una variable de tipo Usuario, se llama a la función altaUsuario() y a su vez esta ejecuta la sentencia mediante executeSql(\$sql) la cual contiene un try/catch que comprobara si existe o no dicho usuario, ya que al ser campos únicos el email y el username, no puede haber coincidencia y la propia base de datos devuelve el error tratado en dicho try/catch
Salida	Alerta si la base de datos ha devuelto algún error.

Fichero	/bdt/controller/valorar_ctl.php
Descripción	Controlador encargado de la actualización de datos referente a votos y horas, cuando un usuario X quiere valorar y/o gratificar con N horas a un usuario Z.
Parámetros	\$usuario → el usuario logueado. \$selectedUserId → id del usuario que se va a valorar. \$valoracion → valoración del usuario seleccionado. \$votosSelectedUser → votos del usuario seleccionado \$voto → voto que el usuario logueado quiere dar al usuario seleccionado. \$pHoras → horas que el usuario logueado da al usuario seleccionado.
Funcionamiento	Se busca el usuario seleccionado mediante la función getUserByUserId(id) , se coge su valoración actual y se calcula la nueva valoración en función de: $((\$valoracion * \$votosSelectedUser) + \$voto) / (\$votosSelectedUser + 1);$ Acto seguido se actualiza la valoración mediante la función updateValoracion(selectedUserId, \$newValoracion, \$votosSelectedUser+1) y se le suman las horas, si es necesario, mediante la función pagarHoras(\$selectedUserId, \$pHoras)
Salida	true si todo ha ido bien. false si algo ha fallado.

Fichero	/bdt/controller/recuperarPass_ctl.php
Descripción	Desde este controlador se genera el cuerpo de un mensaje, que posteriormente se enviará por correo electrónico incluyendo la contraseña.
Parámetros	\$to → email del usuario que quiere recuperar la contraseña.
Funcionamiento	El fichero recibe un email, comprueba si existencia, en la base de datos. Si dicho email existe, se envía un email con la contraseña perteneciente a dicho email
Salida	-----

Fichero	/bdt/controller/loguot_ctl.php
Descripción	Este controlador nos devuelve a la página de inicio de la plataforma, e inicializa las variables logged y usuario.
Parámetros	\$_SESSION['logged'] → Variable que nos indica si hay alguien logueado. \$_SESSION['usuario'] → Variable que contiene el <u>Usuario</u> logueado.
Funcionamiento	Cuando un usuario se desloguea de la plataforma, se inicializan las variables arriba indicadas a <i>false</i> y <i>null</i> respectivamente y nos devuelve a la pagina de bienvenida.
Salida	-----

Fichero	/bdt/controller/modificarOferta_ctl.php
Descripción	Controlador encargado de la modificación de ofertas.
Parámetros	\$ofertald → id de la oferta a modificar. \$categoriald → id de la categoría que se le asignara a la oferta. \$subCatId → id de la subcategoría que se le asignara a la oferta. \$descripcion → descripción de la oferta.
Funcionamiento	Se recogen los parámetros arriba indicados, y se envían a la función <u>modificarOferta</u> , para que esta la modifique.
Salida	-----

3.3.1 Controladores Ajax

Fichero	/bdt/controller/ajax_ctl/reloadTable_ctl.php
Descripción	Controlador encargado de devolver la lista de ofertas existentes en la base de datos.
Parámetros	<p>\$categoria → categoría a la que pertenecen las ofertas que se quieren buscar.</p> <p>\$subcategoria → subcategoría a la que pertenecen las ofertas que se quieren buscar.</p> <p>\$userId → Id del usuario que realiza la petición</p> <p>\$origen → página desde la que se realiza la petición</p> <p>\$sql → Sentencia que se va a ejecutar, en este caso: "SELECT * FROM v_ofertas WHERE categoriaId LIKE '%{\$categoria}' AND subCatId LIKE '%{\$subcategoria}' ";</p>
Funcionamiento	Se crea la sentencia indicada, además se tiene en cuenta si la petición proviene de <i>origen=listado</i> u <i>origen=misOfertas</i> , ya que si proviene de <i>listado</i> , el usuario lo que pretende es ver las ofertas que no son suyas, por el contrario, si proviene de <i>misoferas</i> , el usuario lo que quiere es ver sus ofertas, por lo tanto a \$sql se le concatena " AND userId != '{\$userId}' " para el caso <i>listado</i> o " AND userId = '{\$userId}' " para el caso <i>misoferas</i> . Todo esto se envía a la función getRows , la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

Fichero	/bdt/controller/ajax_ctl/listarUsuarios_ctl.php
Descripción	Controlador encargado de devolver la lista de usuarios existentes en la base de datos.
Parámetros	<p>\$sql → Sentencia que se va a ejecutar, en este caso: "SELECT * FROM v_users ORDER BY userId, username;";</p>
Funcionamiento	Se crea la sentencia indicada, y se envía a la función getRows , la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

Fichero	/bdt/controller/ajax_ctl/reloadSubCats_ctl.php
Descripción	Controlador encargado de devolver la lista de subcategorias existentes en la base de datos.
Parámetros	\$categoria → categoría a la que pertenecen las subcategorías que se quieren buscar. \$sql → Sentencia que se va a ejecutar, en este caso: "SELECT * FROM sub_categorias WHERE categoriasId LIKE '{\$categoria}';";
Funcionamiento	Se crea la sentencia indicada, y se envía a la función getRows , la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

3.4 Utilities

Fichero	/bdt/utilities/sripts.js
Descripción	Este fichero contiene la mayoría de las funciones JS utilizadas en la plataforma.
Parámetros	El fichero como tal ni tiene ni recibe parámetros, pero cada una de las funciones tiene o recibe los parámetros necesarios.
Funcionamiento	-----
Salida	-----

Función	checker()
Descripción	Función que muestra u oculta la contraseña del usuario
Parámetros	-----
Funcionamiento	Comprueba si el switch de la contraseña esta marcado o no, y en función de cómo esté, llamara a la función showPasword para mostrar la contraseña, o la función hidePassword para ocultarla.
Salida	-----

Función	showPassword()
Descripción	Función que cambia el tipo de objeto a "text", mostrando así la contraseña del usuario
Parámetros	-----
Funcionamiento	Esta función busca el elemento password, y lo convierte en un elemento de tipo text
Salida	-----

Función	hidePassword()
Descripción	Función que cambia el tipo de objeto a "password", ocultando así la contraseña del usuario
Parámetros	-----
Funcionamiento	Esta función busca el elemento <i>password</i> , y lo convierte en un elemento de tipo password
Salida	-----

Función	checkerAceptar()
Descripción	Función que activa el botón de aceptar, para poder registrarse, al aceptar los términos y condiciones de uso.
Parámetros	obj → es el checkbox que hay que marcar para aceptar las condiciones de uso.
Funcionamiento	Recibe el obj, se mira si esta marcado o no, y cuando se marca, se habilita el botón <i>Aceptar</i>
Salida	-----

Función	volverAtras()
Descripción	Función que nos devolverá a la página de la que venimos
Parámetros	origen → es la vista de la que venimos.
Funcionamiento	Genera un texto tal que: <i>'?action=view&option='+origen;</i> y lo asigna a window.location, de este modo nos redirigirá a fichero index.php , el cual gestionara la redirección a donde tenemos que ir según <i>origen</i> .
Salida	-----

Función	confirmarEliminarOferta(ofertaId)
Descripción	Función que muestra una alerta pidiendo confirmación al eliminar una oferta
Parámetros	ofertaId → identificador de la oferta que queremos eliminar.
Funcionamiento	Mostrará una alerta de confirmación o cancelación preguntando que si realmente se desea eliminar la oferta seleccionada.
Salida	Mensaje de alerta.

Función	confirmarEliminarUsuario(userId)
Descripción	Función que muestra una alerta pidiendo confirmación al eliminar un usuario.
Parámetros	userId → identificador del usuario que queremos eliminar.
Funcionamiento	Mostrará una alerta de confirmación o cancelación preguntando que si realmente se desea eliminar el usuario seleccionado.
Salida	Mensaje de alerta.

Función	populateDataTable(tableName, pageLength= 5)
Descripción	Función que recibe el nombre de una table y genera un DataTable con el formato indicado.
Parámetros	<p>tableName → nombre de la tabla a la que se le ha de dar formato.</p> <p>pageLength → cantidad de registros a mostrar por página. Por defecto 5.</p> <p>order → Columnas por las que queremos ordenar.</p> <p>ordering → Permite que las columnas sean ordenadas haciendo clic en la cabecera (true/false)</p> <p>paging → Muestra(true) u oculta(false) el select con las cantidades</p> <p>lengthMenu → Menu desplegable "Show entries" ([5, 10, 15])</p> <p>pageLength → Valor inicial del menu desplegable "Show entries"</p> <p>oLanguage → Texto a mostrar en la sección info.</p> <p>Info → Muestra(true) u oculta(false) la sección info: "Showing 1 to 10 of 16 entries.</p> <p>scrollY → Altura de la tabla a partir de la cual queremos que nos muestre el scroll vertical.</p> <p>scrollX → Anchura de la tabla a partir de la cual queremos que nos muestre el scroll horizontal.</p> <p>pagingType → muestra los botones <i>First</i>, <i>Previous</i>, <i>Next</i> y <i>Last</i> ('full_numbers').</p> <p>language: { lengthMenu: 'Mostrar _MENU_ registros por página', zeroRecords: 'Nothing found - sorry', infoEmpty: 'No records available', } → conjunto de mensaje a mostrar.</p>
Funcionamiento	Recibe el nombre de la tabla que tiene que buscar, y le da el formato en función de todas las variable arriba mencionadas.
Salida	-----

Función	populateTable(ofertas, role)
Descripción	Función que crear y rellena las tablas de misOfertas y listado
Parámetros	<p>ofertas → lista de ofertas a mostrar.</p> <p>role → rol del usuario conectado.</p>
Funcionamiento	<p>Función que recibe una lista de ofertas, y con ella crea y rellena una tabla, además, tendrá en cuenta el rol del usuario conectado, dependiendo del rol mostrara las ofertas propias del administrador (rol == 1), o las ofertas propias del usuario (rol != 1).</p> <p>La función también busca el origen del que procedemos para saber si tenemos que mostrar opciones de eliminar y modificar ofertas o no.</p> <p>Y finalmente llamara a la función popuateDataTable(tableName), para que esta de formato a la tabla.</p>
Salida	-----

Función	populateUsuarios(usuarios)
Descripción	Función que crear y rellena la tabla de usuarios
Parámetros	usuarios → lista de usuarios a mostrar.
Funcionamiento	Función que recibe una lista de usuarios, y con ella crea y rellena una tabla. Y finalmente llamara a la función populateDataTable(tableName) , para que esta de formato a la tabla.
Salida	-----

Función	reloadSubCat(subCatList)
Descripción	Función que recibe una lista de subcategorías y rellena el select correspondiente.
Parámetros	subCatList → lista de subcategorías.
Funcionamiento	Recibe una lista de subcategorías, con ella montara un <i>select</i> . Una vez montado se lanza el trigger 'change', propio del select para que se actualice la lista en función de categoría seleccionada.
Salida	-----

Función	topMenuLogged(role)
Descripción	Función que mostrara la parte del menú superior .
Parámetros	role → rol del usuario conectado.
Funcionamiento	Es función muestra las opciones del menú superior, en función de si el usuario ya ha hecho login o no, y si es un usuario normal o por el contrario es un administrador.
Salida	-----

Función	topMenuLogout()
Descripción	Función que ocultara la parte del menú superior .
Parámetros	role → rol del usuario conectado.
Funcionamiento	Es función muestra las opciones del menú superior cuando aun no se ha hecho login
Salida	-----

Fichero	/bdt/utilities/encryptador.php
Descripción	Clase para encriptar y desencriptar las contraseñas
Parámetros	SUBS_ALPHABET → alfabeto con el que se encriptara el texto correspondiente.
Funcionamiento	Esta clase tiene dos funciones, encrypt(\$text) y decrypt(\$text) , encargadas de encriptar o desencriptar el texto que reciban.
Salida	-----

Función	encrypt(\$text)
Descripción	Función que recibe un texto lo encripta
Parámetros	\$text → texto a encriptar.
Funcionamiento	Recibe un texto y mediante el algoritmo "AES-256-ECB" y el alfabeto arriba indicado, encriptara el texto recibido
Salida	Texto encriptado

Función	decrypt(\$text)
Descripción	Función que recibe un texto lo desencripta
Parámetros	\$text → texto a desencriptar.
Funcionamiento	Recibe un texto y mediante el algoritmo "AES-256-ECB" y el alfabeto arriba indicado, desencriptara el texto recibido
Salida	Texto desencriptado.

Fichero	/bdt/utilities/sendMail.php
Descripción	Clase para el manejo de correos
Parámetros	\$to → destinatario del correo. \$subject → título del correo. \$body → cuerpo del correo.
Funcionamiento	Recibirá los parámetros arriba indicados, y enviará un mail, desde la cuenta de correo bancodeeltiempo2023@gmail.com al destinatario correspondiente. Además, se ha generado un template con el fondo de pantalla de la plataforma.
Salida	Email

3.5 Vistas

Fichero	/bdt/view/head.php
Descripción	Cabecera de la página.
Parámetros	-----
Funcionamiento	Este fichero contiene el head de la página, ya que siempre va a ser el mismo así será más fácil de importar. Este mismo fichero, importara el header del html. Desde él, también se llaman a las librerías necesarias para la realización de la plataforma (jquery, datables, bootstrap5, etc) además de cargar los estilos propios.
Salida	-----

Fichero	/bdt/view/header.php
Descripción	Menú superior de la página.
Parámetros	\$usuario → variable de tipo usuario .
Funcionamiento	Este fichero carga y muestra el menú superior de la página. La variable usuario se utiliza para saber que usuario esta conectado, y de este modo saber su información personal.
Salida	-----

Fichero	/bdt/view/footer.php
Descripción	Footer de la página.
Parámetros	-----
Funcionamiento	Este fichero contiene el footer de la página, ya que siempre va a ser el mismo, el cual contiene 4 botones, que nos llevaran al Facebook, al LinkedIn, y a la propia pagina de la UOC, respectivamente, y el 4º botón abrirá la venta de correo, perada para enviar un email a la administración de la plataforma.
Salida	-----

Fichero	/bdt/view/filtros_view.php
Descripción	Este fichero contiene los filtros de categorías y subcategorías
Parámetros	\$categorias → lista de categorías. \$usuario → usuario conectado. \$origen → es la vista de la que venimos.
Funcionamiento	El fichero recibe una lista con todas las categorías existentes, con ella monta un select, y una vez montado, ejecutara su propio trigger 'change', el cual llamará al controlador reloadSubCats_ctl , este a su vez devolverá un lista de subcategorías, en la respuesta se llamará la función reloadSubCats y montara el select de subcategorías
Salida	-----

Fichero	/bdt/view/insertarCategoria_view.php
Descripción	Este fichero muestra la pantalla de inserción de categorías.
Parámetros	\$usuario → el usuario conectado. \$categorias → lista de todas las categorías existentes. \$subCategorias → lista de todas las subcategorías existentes.
Funcionamiento	Recogerá una subcategoría i/o una categoría, y llamará al controlador de insertarCategoria_ctl , el cual tras las comprobaciones pertinentes decidirá si inserta o no los valores recogidos.
Salida	Texto de la categoría. Texto de la subcategoría.

Fichero	/bdt/view/misOfertas_view.php
Descripción	Este fichero muestra la pantalla de las ofertas propias del usuario logueado.
Parámetros	\$usuario → el usuario conectado. \$selCat → id de la categoría seleccionada. \$subCatId → id de la subcategoría seleccionada. \$desc → descripción dada por el usuario.
Funcionamiento	El usuario puede listar, modificar o insertar nuevas ofertas. Para insertar una nueva oferta debe seleccionar una categoría, una subcategoría y opcionalmente da una descripción de la oferta. El fichero recoge los datos y los envía al controlador insertarOferta_ctl , el cual tras las comprobaciones pertinentes decidirá si inserta o no los valores recogidos.
Salida	-----

Fichero	/bdt/view/listado_view.php
Descripción	Este fichero muestra la pantalla de listado.
Parámetros	\$usuario → el usuario conectado. \$_SESSION['origen'] → Dado que una vez se ha hecho login se redirige a esta página, se asigna por primera vez un origen, en este caso <i>listado</i> .
Funcionamiento	El usuario ve el listado de todas las ofertas, menos las suyas, desde aquí puede filtrar, buscar, contactar o redirigirse a la pantalla de valoraciones .
Salida	-----

Fichero	/bdt/view/listarUsuarios_view.php
Descripción	Este fichero muestra la pantalla de listado de usuarios (solo visible por administradores).
Parámetros	\$usuario → el usuario conectado. \$listarUsuarios → la lista completa de los usuarios registrados en la plataforma.
Funcionamiento	Se monta una tabla con todos los usuarios registrados en la plataforma. Desde aquí los administradores pueden eliminar usuarios o acceder a su información personal .
Salida	-----

Fichero	/bdt/view/login_view.php
Descripción	Este fichero muestra la pantalla de login.
Parámetros	\$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no \$_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.
Funcionamiento	Pantalla que recoge un email y una contraseña y lo envía al controlador login_ctl.php , el cual dara paso, o no, a la plataforma.
Salida	Username del usuario. Contraseña del usuario.

Fichero	/bdt/view/modificarOfertas_view.php
Descripción	Este fichero muestra la pantalla que permite al usuario modificar sus ofertas.
Parámetros	\$usuario → el usuario conectado. \$ofertald → Identificador de la oferta a modificar. \$oferta → oferta que se va a modificar.
Funcionamiento	El usuario llega a esta pantalla desde su listado de ofertas , aquí puede elegir nueva categoría, nueva subcategoría y modificar la descripción, estos datos son recogidos y enviados al controlador modificarOferta_ctl , el cual hará la modificación correspondiente en la oferta seleccionada.
Salida	\$ofertald → Identificador de la oferta a modificar. \$categoriald → categoría de la oferta a modificar. \$subCatId → subcategoría de la oferta a modificar. \$descripcion → descripción de la oferta a modificar.

Fichero	/bdt/view/ofertante_view.php
Descripción	Este fichero permite al usuario conectado, ver los datos detallados de otro usuario.
Parámetros	\$selectedUser → el usuario que queremos consultar. \$oferta → oferta que se va a consultar. \$ofertald → identificador de la oferta correspondiente. \$usuario → usuario conectado.
Funcionamiento	El usuario llega a esta pantalla desde su listado de ofertas , haciendo click, en cualquier Nick de la columna <i>ofertante</i> . Al hacer click en el Nick, se envían la \$ofertald, y el \$userId, con esto se monta la vista que permite ver los datos del usuario deseado. Una vez se valora y paga con horas al usuario estos datos son enviados al controlador valorar_ctl.php el cual realizara los cambios necesarios.
Salida	id del usuario a valorar. Valoración que se le da al usuario a valorar(id). Horas que se le tienen que sumar al usuario seleccionado (id).

Fichero	/bdt/view/recuperarPass_view.php
Descripción	Es fichero muestra la pantalla de recuperación de contraseña.
Parámetros	-----
Funcionamiento	El usuario rellena el campo email con su correo electrónico y le da a <i>enviar</i> . Este envía la dirección al controlador recuperarPass_ctl.php el cual comprobara si existe dicha dirección o en la base de datos y realizara las acciones comentadas.
Salida	Email a comprobar.

Fichero	/bdt/view/personalInfo_view.php
Descripción	<p>Este fichero permite al usuario conectado, ver sus propios datos y modificarlos. Dicho usuario accedera desde el menú superior haciendo click en <i>Información Personal</i>.</p> <p>Un administrador también accedera a esta pantalla, a través del menú desplegable que se encuentra en la parte superior izquierda (<i>Admin options</i>)</p>
Parámetros	<p>\$enc → variable de tipo Encriptador.</p> <p>\$selectedUser → el usuario que queremos consultar.</p> <p>\$ofertald → identificador de la oferta correspondiente.</p> <p>\$usuario → usuario conectado.</p> <p>\$role → rol del usuario conectado.</p>
Funcionamiento	Se reciben los datos arriba indicados, y se monta la vista. En caso de que algún campo sea modificado, el botón <i>aceptar</i> enviara los datos al controlador personalInfo_ctl.php y este realizara los cambios pertinentes.
Salida	<p>\$userId → identificador del usuario consultado.</p> <p>\$userName → username del usuario consultado.</p> <p>\$emailReg → email del usuario consultado.</p> <p>\$password → contraseña del usuario consultado.</p> <p>\$newPass → nueva contraseña del usuario consultado.</p> <p>\$repeatNew → repetición de la nueva contraseña (han de coincidir)</p> <p>\$nombreReg → nombre del usuario consultado.</p> <p>\$apellidos → apellidos del usuario consultado.</p> <p>\$poblacion → población del usuario consultado.</p> <p>\$valoracio → valoración del usuario consultado.</p> <p>\$votos → votos del usuario consultado.</p> <p>\$horas → horas del usuario consultado.</p> <p>\$role → rol del usuario consultado.</p>

Fichero	/bdt/view/register_view.php
Descripción	Es fichero muestra la pantalla de registro.
Parámetros	<p>\$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no</p> <p>\$_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.</p>
Funcionamiento	El usuario rellena los campos y acepta las condiciones de uso, estos datos son enviados al controlador register_ctl.php y este realiza las acciones necesarias.
Salida	<p>\$nombreReg → nombre del usuario que se va a registrar.</p> <p>\$apellidosReg → apellidos del usuario que se va a registrar.</p> <p>\$userName → username del usuario que se va a registrar.</p> <p>\$passwordReg → contraseña del usuario que se va a registrar.</p> <p>\$poblacionReg → población del usuario que se va a registrar.</p> <p>\$emailReg → dirección de correo del usuario que se va a registrar.</p>

Fichero	/bdt/view/welcome.php
Descripción	Es fichero contiene el <body> de la pantalla de bienvenida.
Parámetros	\$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no \$_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.
Funcionamiento	Solo contiene el <body> de la primera pantalla que vera cualquier usuario al acceder a la plataforma.
Salida	-----

Fichero	/bdt/view/welcome_view.php
Descripción	Es fichero muestra la pantalla de bienvenida.
Parámetros	-----
Funcionamiento	Se cargan el head , el welcome y el footer , con todo ello se muestra la pantalla de bienvenida al usuario
Salida	-----

4 Pantallas

A continuación, vamos a ver las vistas y su resultado final de las vistas

Página de bienvenida



Figura 41

Header sin nadie logueado



Figura 42

Header de administrador



Figura 43

Header de usuario



Figura 44

Footer



Figura 45

Filtros

The filter section is a light blue rectangular box. It contains two columns. The left column is titled 'Categorías' and has a dropdown menu with the text 'Seleccione una opción' and a downward arrow. The right column is titled 'Sub-categorías' and also has a dropdown menu with the text 'Seleccione una opción' and a downward arrow.

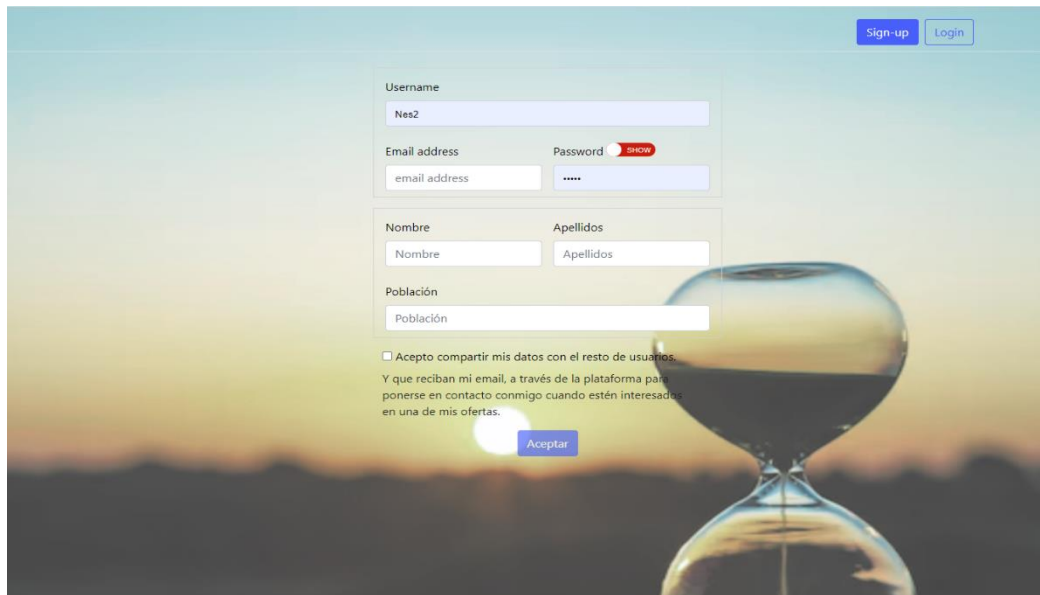
Figura 46

Login

The login form is centered on the page. It has a background image of a sunset with an hourglass in the foreground. The form includes a 'Sign-up' button in the top right corner. The main form fields are 'Username' (with the text 'Nes2') and 'Password' (with a 'SHOW' button). Below the password field is a 'Forgot password?' link and a 'Sign in' button. At the bottom of the page, there are social media icons for Facebook, LinkedIn, YouTube, and Email.

Figura 47

Pantalla de registro



Sign-up Login

Username
Nes2

Email address
email address

Password
SHOW

Nombre
Nombre

Apellidos
Apellidos

Población
Población

☐ Acepto compartir mis datos con el resto de usuarios.
Y que reciban mi email, a través de la plataforma para
ponerse en contacto conmigo cuando estén interesados
en una de mis ofertas.

Aceptar

Figura 48

Pantalla de recuperación de contraseña



Sign-up Login

email address

Email address

Enviar

f in UoC

Figura 49

Pantalla de listado

Admin options ▾

Listado Mis ofertas Información personal

Logout

Categorías Sub-categorías

Seleccione una opción ▾ Seleccione una opción ▾

Mostrar 5 registros por página

Search:

Id	Categoría	Sub categoría	Descripción	Valoración	Votos	Ofertante	Borrar	Modificar
11	Informática	Programacion Python	3333	7.10	3	Nes2		
12	Informática	Programacion c++	3333	7.10	3	Nes2		
13	Informática	Programacion c++	3333	7.10	3	Nes2		
14	Música	Guitarra	3333	7.10	3	Nes2		
16	Informática	Programacion Python	333	7.10	3	Nes2		

Mostrando del 1 al 5 de 107 registros

First Previous 1 2 3 4 5 ... 22 Next Last

f in uo

Figura 50

Pantalla de misOfertas

Admin options ▾

Listado Mis ofertas Información personal

Logout

Categorías Sub-categorías

Seleccione una opción ▾ Seleccione una opción ▾

Descripción (max. 220 caracteres)

Insertar

Mostrar 5 registros por página

Search:

Oferta	Categoría	Sub categoría	Descripción	Valoración	Votos	Borrar	Modificar
21	Idiomas	Catalán	cccccccc sada124s as dasd asd asda sdasdasada124s as dasd asd asda sdasdasada124s as dasd asd asda sdasdasada124s	10.00	3		
22	Informática	Programacion Java	asdaa	10.00	3		
23	Informática	Programacion Python	asd	10.00	3		
24	Idiomas	Catalán	fh	10.00	3		
25	Idiomas	Ingles	resrsss	10.00	3		

Mostrando del 1 al 5 de 61 registros

First Previous 1 2 3 4 5 ... 13 Next Last

f in uo

Figura 51

Pantalla de información Personal y modificar usuario

Figura 52

Pantalla de listado de usuarios

Id	Username	Pass	Email	Nombre	Apellidos	Población	Horas	roleid	Role	Valoración	Votos	Borrar	Modificar
1	Nes	BpeoQa/YNeJfk9/ZW9oQCw==	Nestito_@hotmail.com	Néstor	Ibáñez	Granollers	8	1	ADMIN	10.00	3		
2	Nes1	BpeoQa/YNeJfk9/ZW9oQCw==	Nestito_@hotmail.com	Nombre1	Apellido1	Granollers	0	2	USER	0.00	0		
3	Nes2	BpeoQa/YNeJfk9/ZW9oQCw==	nestorip79@gmail.com	Nombre2aq	Apellido2	Granollers	26	2	USER	7.10	3		
4	Nes7	BpeoQa/YNeJfk9/ZW9oQCw==	3Nestito_@hotmail.com	Nestor	Ibanez Polo	Granollers	10	2	USER	0.00	0		
5	Nes5	BpeoQa/YNeJfk9/ZW9oQCw==	Naaestito_2@hotmail.com	Nestor5	Ibanez Polo	Granollers	10	3	BANNED	0.00	0		

Figura 53

Pantalla de valoraciones

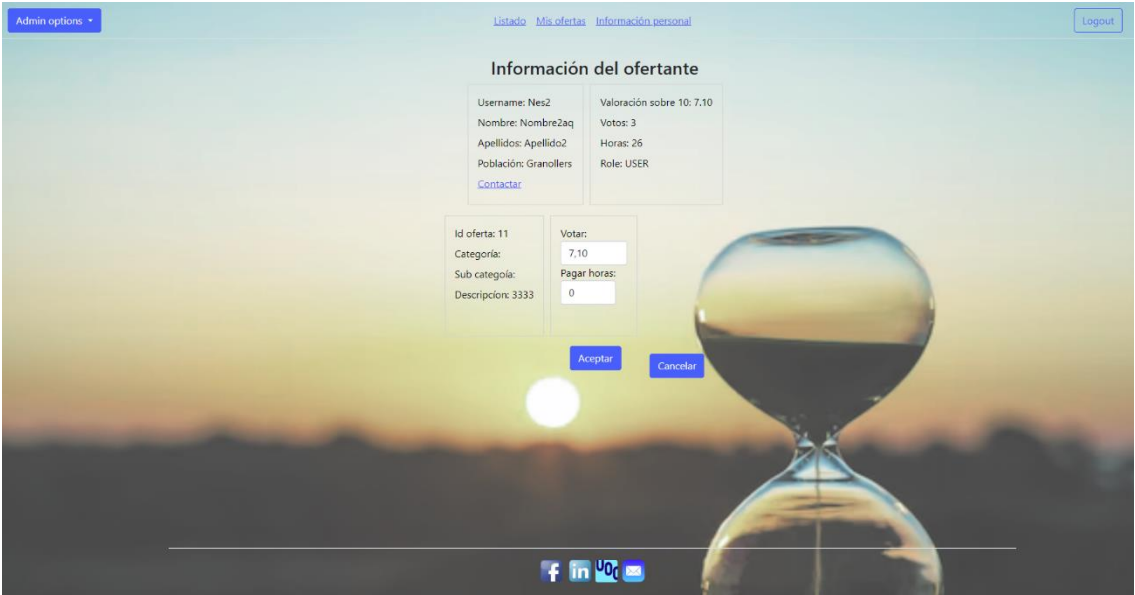


Figura 54

Pantalla de añadir categoría

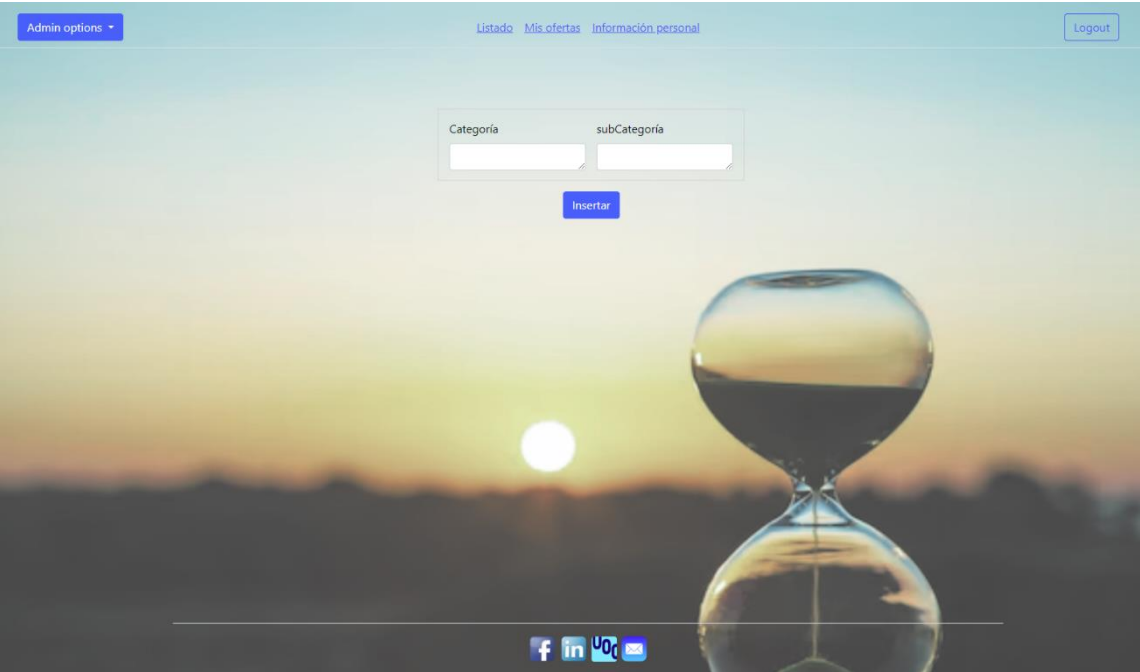


Figura 55

5 Otros fichero y carpetas

/bdt/Schema.sql este fichero contiene las sentencias sql que hay que ejecutar para montar la base de datos, con sus tablas, vista, trigger y lo necesario para el buen funcionamiento de la plataforma.

/bdt/view/css/estilos.css este fichero contiene los estilos personalizados que se han aplicado a la plataforma, algunos de ellos han sido obtenidos de las propias librerías de Bootstrap y modificados al gusto.

/bdt/PHPMailer es la librería que hace posible el envío de correos electrónicos.

/bdt/logs/ en esta carpeta se generan los log de las funciones que ejecutan sentencias sql.

6 Bibliografía y videografía

- *Cómo crear un DIAGRAMA de GANTT en Excel [Cronograma usando los gráficos]* [en línea] [consulta: 06 de febrero de 2023]. Disponible en: <https://www.youtube.com/watch?v=chR6kx4btDQ>
- *Manual de php* [en línea] [consulta: en múltiples ocasiones entre el 17 de abril y el 30 de mayo de 2023]. Disponible en: <https://www.php.net/manual/es/>
- *Manual DataTables* [en línea] [consulta: en múltiples ocasiones entre el 17 de abril y el 30 de mayo de 2023]. Disponible en: <https://datatables.net/>
- *Preparación del entorno en Linux* [en línea] [consulta: 10 de mayo de 2023] Disponible en: <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04-es>
- *Como instalar múltiples versiones de php* [en línea] [consulta: 10 de mayo de 2023] Disponible en: <https://help.clouding.io/hc/es/articles/360021630059-C%C3%B3mo-Instalar-m%C3%BAltiples-versiones-de-PHP-7-2-7-4-and-8-0-en-Ubuntu-20-04>
- *Como instalar PHP 8.2 en Ubuntu 22.04* [en línea] [consulta: 10 de mayo de 2023] Disponible en: <https://techvblogs.com/blog/install-php-8-2-ubuntu-22-04>
- *¿Cómo arreglar Apache2 no ejecutando archivos PHP?* [en línea] [consulta: 10 de mayo de 2023] Disponible en: <https://support.hostinger.es/es/articles/3963793-como-arreglar-apache2-no-ejecutando-archivos-php>
- *Instalar certificado SSL GRATIS y activar HTTPS en tu Servidor WEB* [en línea] [consulta: 20 de mayo de 2023] Disponible en: <https://www.youtube.com/watch?v=eM8tU9ZuRC0>