

Descripción

En esta entrada vamos a desplegar una aplicación Python realizada en Django.

Entorno de desarrollo

1. Utilizamos la máquina **bravo** de nuestro **escenario** que usa Rocky Linux 9.
2. Vamos a configurar nuestro escenario de la siguiente manera:
 - Realizamos un fork del repositorio con la **app**.
 - Creamos un entorno virtual de Python3 e instalamos las dependencias necesarias.
 - Comprobamos que la base de datos con la que vamos a trabajar es SQLite.
 - ¿Qué fichero tenemos que consultar?
 - ¿Cómo se llama la base de datos que vamos a crear?
 - Creamos la base de datos.
 - Creamos el usuario Administrador.
 - Ejecutamos el servidor web de desarrollo y entramos en la zona de administración para comprobar que los datos se han almacenado correctamente.
 - Creamos dos preguntas con posibles respuestas.
 - Comprobamos que podemos acceder a la aplicación desde el navegador. y accedemos a la url **/polls**.
 - Configura el servidor web con el módulo wsgi para servir la página web. Como hemos utilizado la máquina **bravo** se accederá por el nombre **python.maria.gonzalonazareno.org**.

Realización Entorno de desarrollo

Para realizar el entorno de desarrollo vamos a utilizar la máquina **bravo** de nuestro escenario, que usa Rocky Linux 9.

- Primero instalamos git para poder clonar el repositorio de la aplicación:

```
sudo dnf install git
```

- Realizamos un fork del repositorio con la **app** y clonamos el repositorio en la máquina **bravo**, concretamente en el directorio **/var/www/html**:

```
git clone https://github.com/Legnakra/Tutorial-Django.git
```

E instalamos el entorno virtual de python:

```
python3 -m venv venv  
  
source venv/bin/activate
```

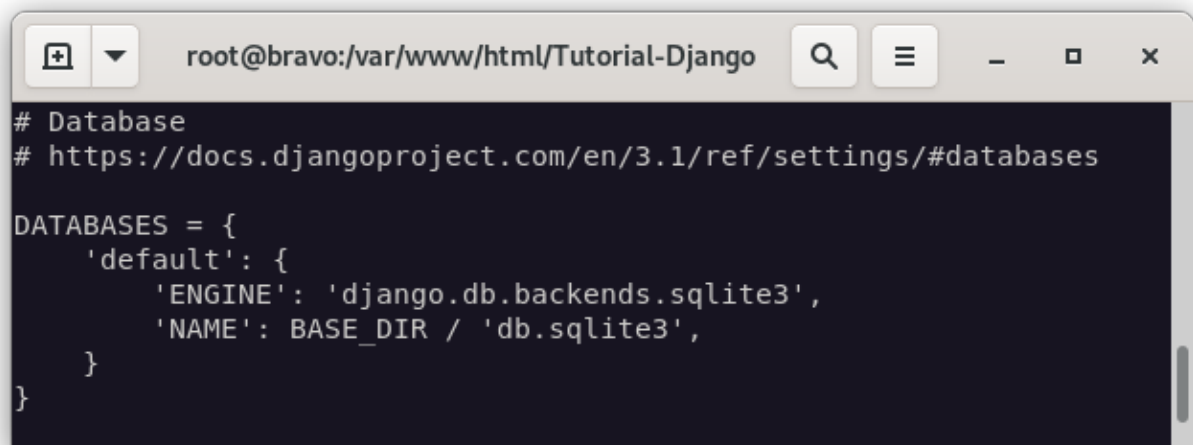
- Instalamos las dependencias de la aplicación:

```
pip install -r requirements.txt
```

- ¿Qué fichero tienes que consultar? ¿Cómo se llama la base de datos que vamos a crear?

Para consultar el fichero que tenemos que consultar, vamos a ver el fichero `settings.py` de la aplicación, en el apartado de `DATABASES`:

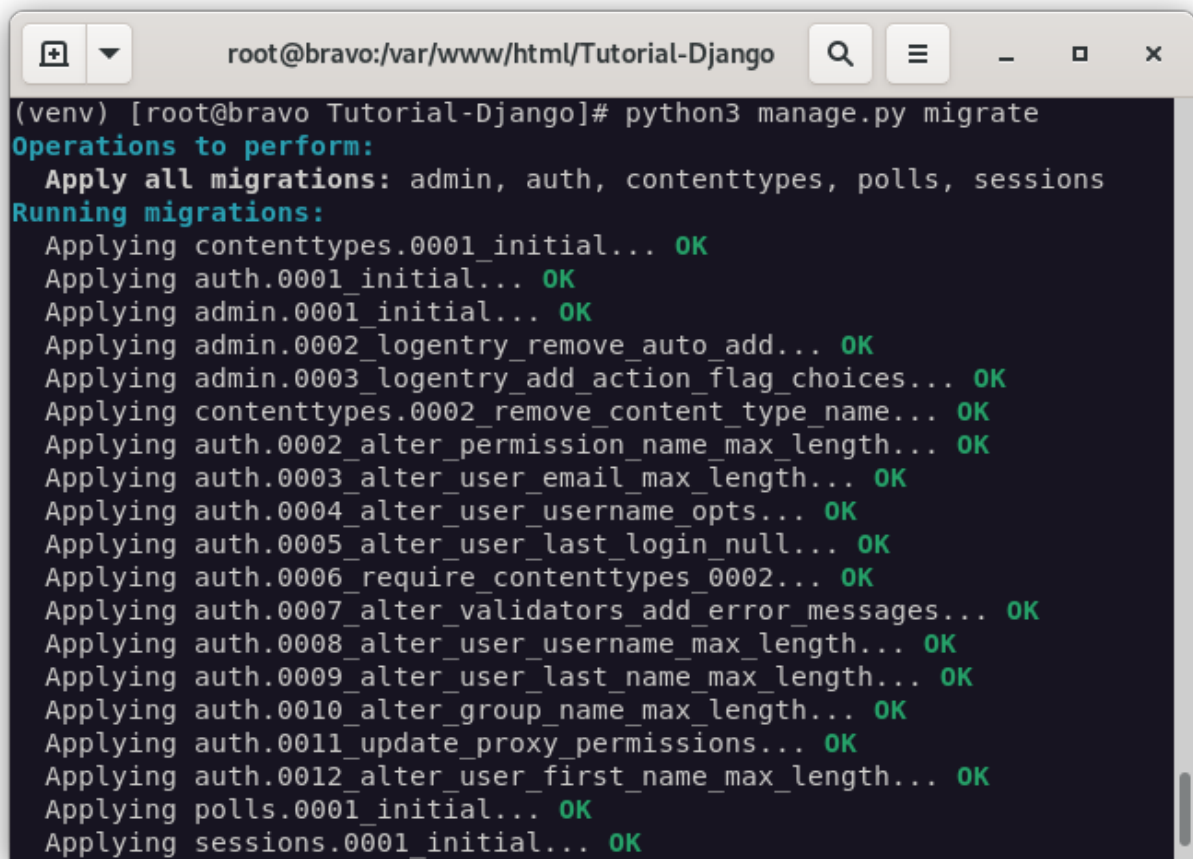
```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
    }  
}
```



Como podemos ver, la base de datos que vamos a crear es `db.sqlite3`.

- Creamos la base de datos:

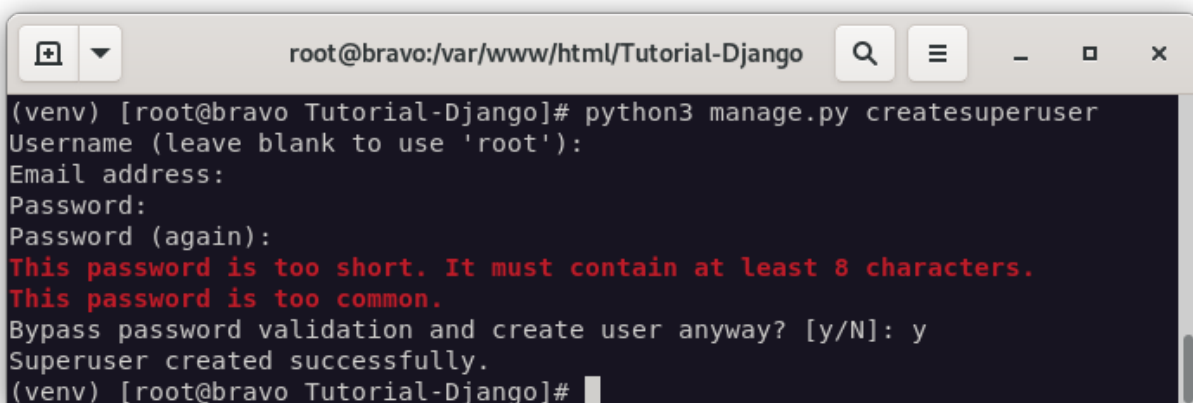
```
python django_tutorial/manage.py migrate
```



```
root@bravo:/var/www/html/Tutorial-Django
(venv) [root@bravo Tutorial-Django]# python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying polls.0001_initial... OK
  Applying sessions.0001_initial... OK
```

- Creamos el usuario Administrador:

```
python django_tutorial/manage.py createsuperuser
---
password: admin
```



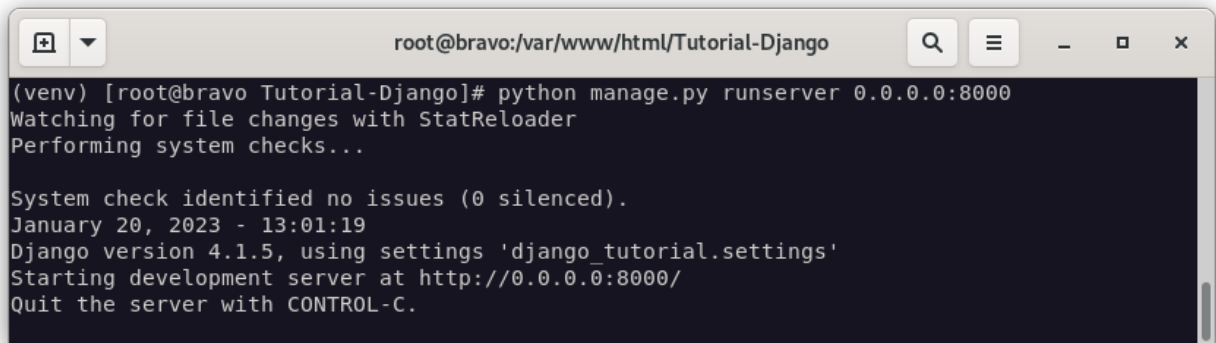
```
root@bravo:/var/www/html/Tutorial-Django
(venv) [root@bravo Tutorial-Django]# python3 manage.py createsuperuser
Username (leave blank to use 'root'):
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(venv) [root@bravo Tutorial-Django]#
```

- Modificamos el fichero `settings.py` para que el servidor web de desarrollo escuche en todas las interfaces:

```
ALLOWED_HOSTS = ['*']
```

- Ejecutamos el servidor web de desarrollo y entramos en la zona de administración para comprobar que los datos se han almacenado correctamente:

```
python django_tutorial/manage.py runserver 0.0.0.0:8000
```

A screenshot of a terminal window titled 'root@bravo:/var/www/html/Tutorial-Django'. The terminal shows the command '(venv) [root@bravo Tutorial-Django]# python manage.py runserver 0.0.0.0:8000' being executed. The output includes 'Watching for file changes with StatReloader', 'Performing system checks...', 'System check identified no issues (0 silenced).', the date 'January 20, 2023 - 13:01:19', 'Django version 4.1.5, using settings \'django_tutorial.settings\'', 'Starting development server at http://0.0.0.0:8000/', and 'Quit the server with CONTROL-C.'.

```
(venv) [root@bravo Tutorial-Django]# python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 20, 2023 - 13:01:19
Django version 4.1.5, using settings 'django_tutorial.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

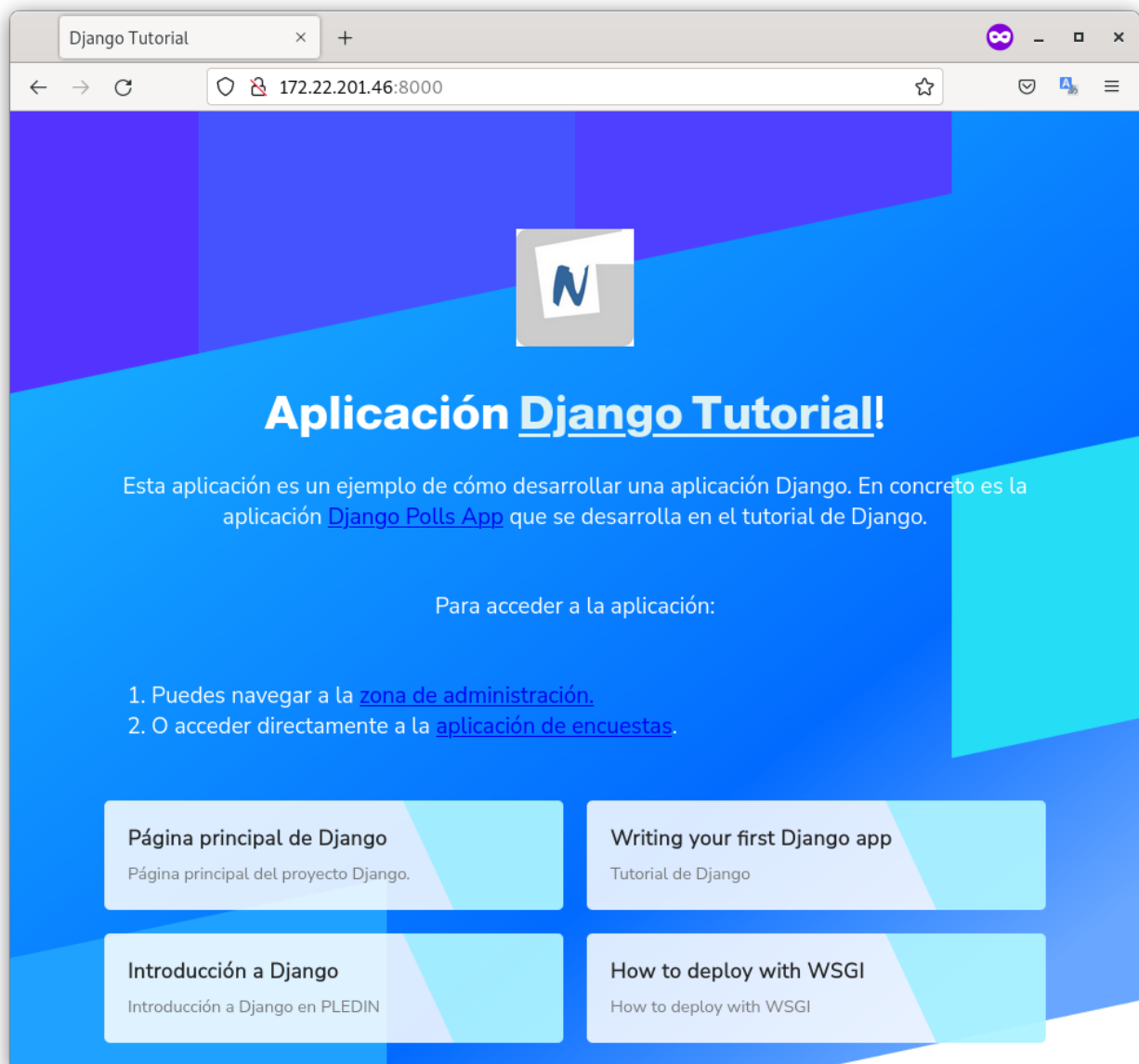
NOTA: Para acceder a la página web desde el ordenador, tendremos que añadir una nueva regla DNAT en **alfa** para que redirija el tráfico de la máquina **bravo** al puerto 8000.

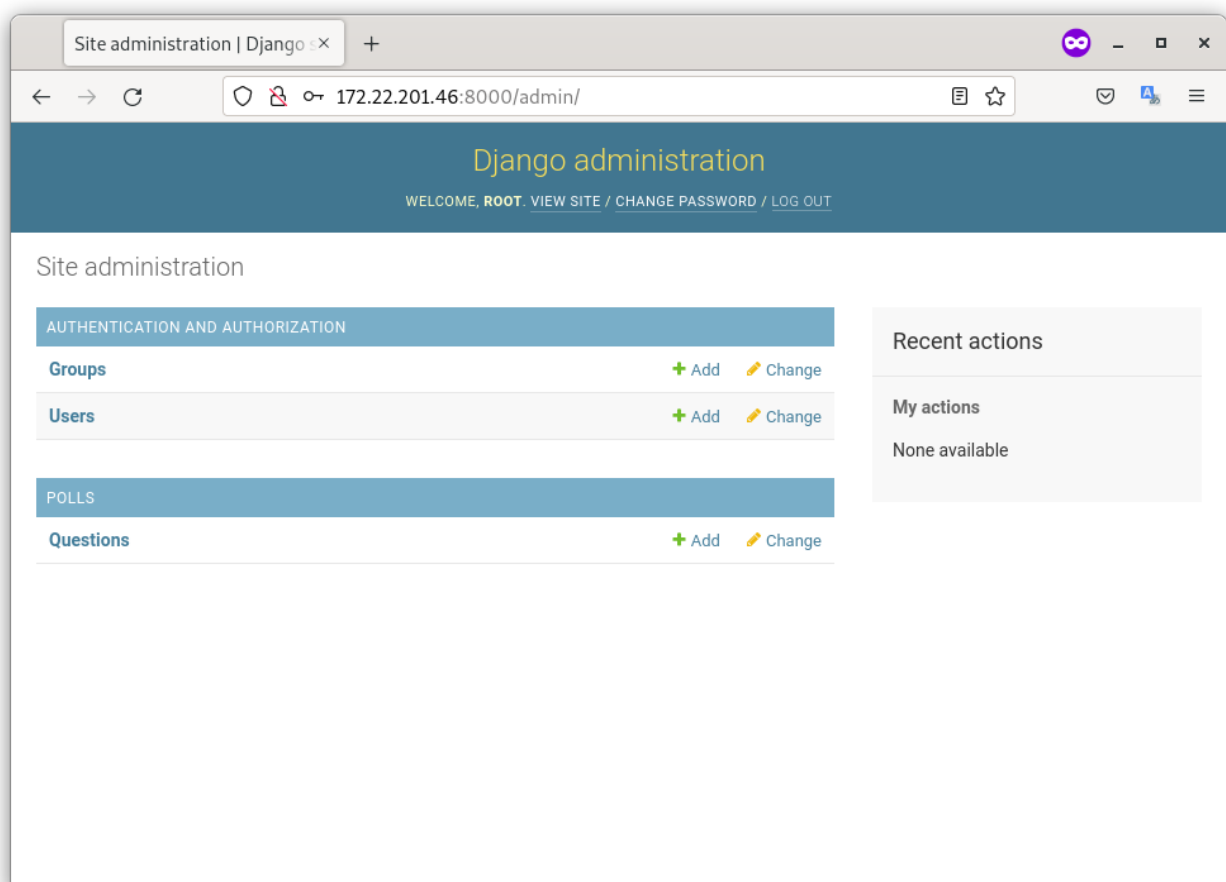
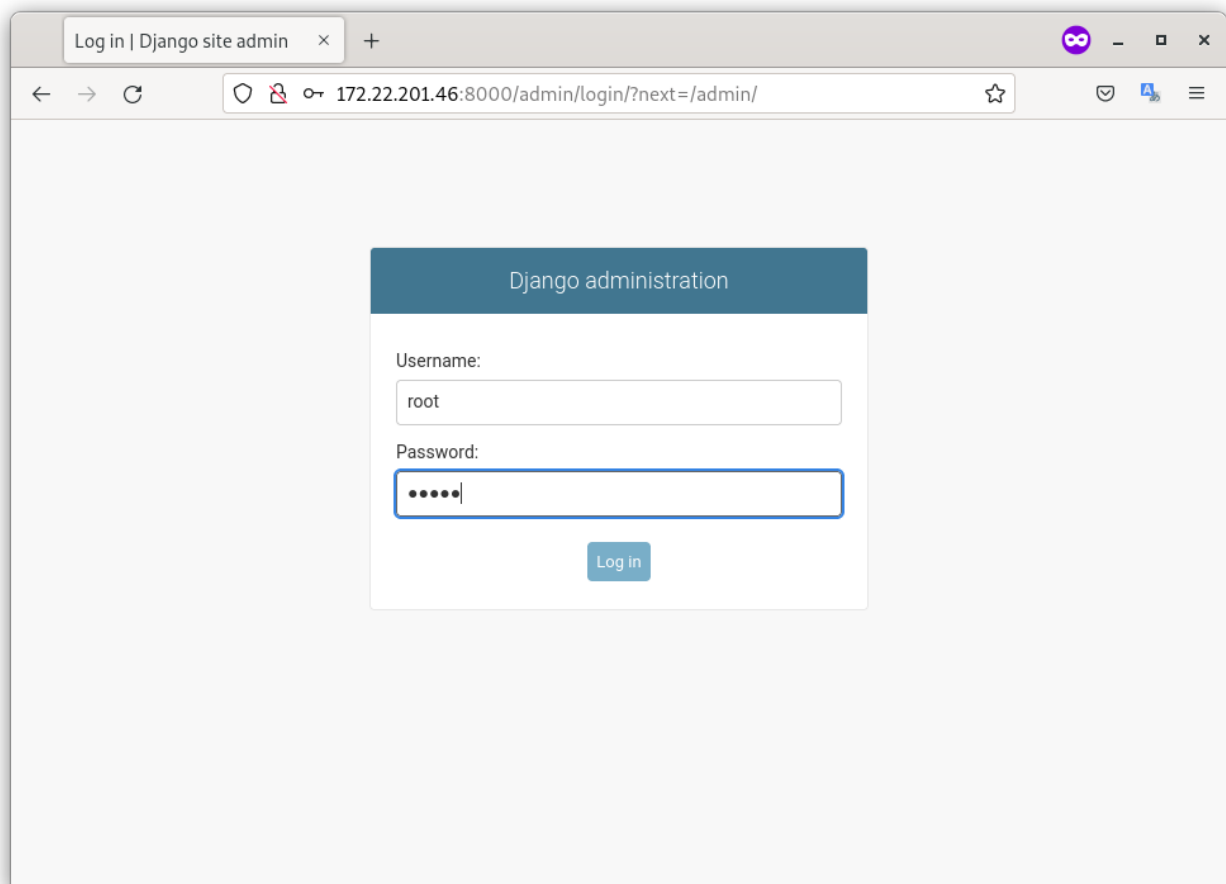
```
nano /etc/network/interfaces.d/50-cloud-init
---
post-up iptables -t nat -A PREROUTING -p tcp --dport 8000 -i ens3 -j
DNAT --to 172.16.0.200
```

Seguidamente reiniciamos la máquina **alfa**:

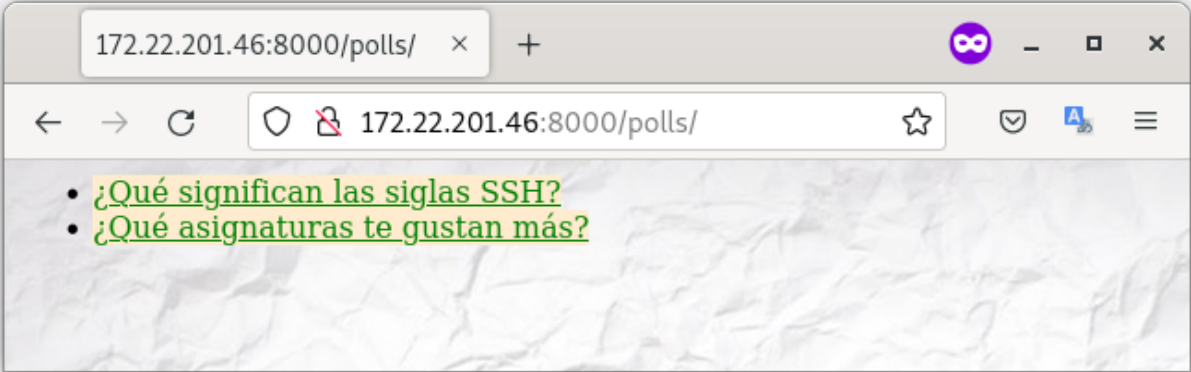
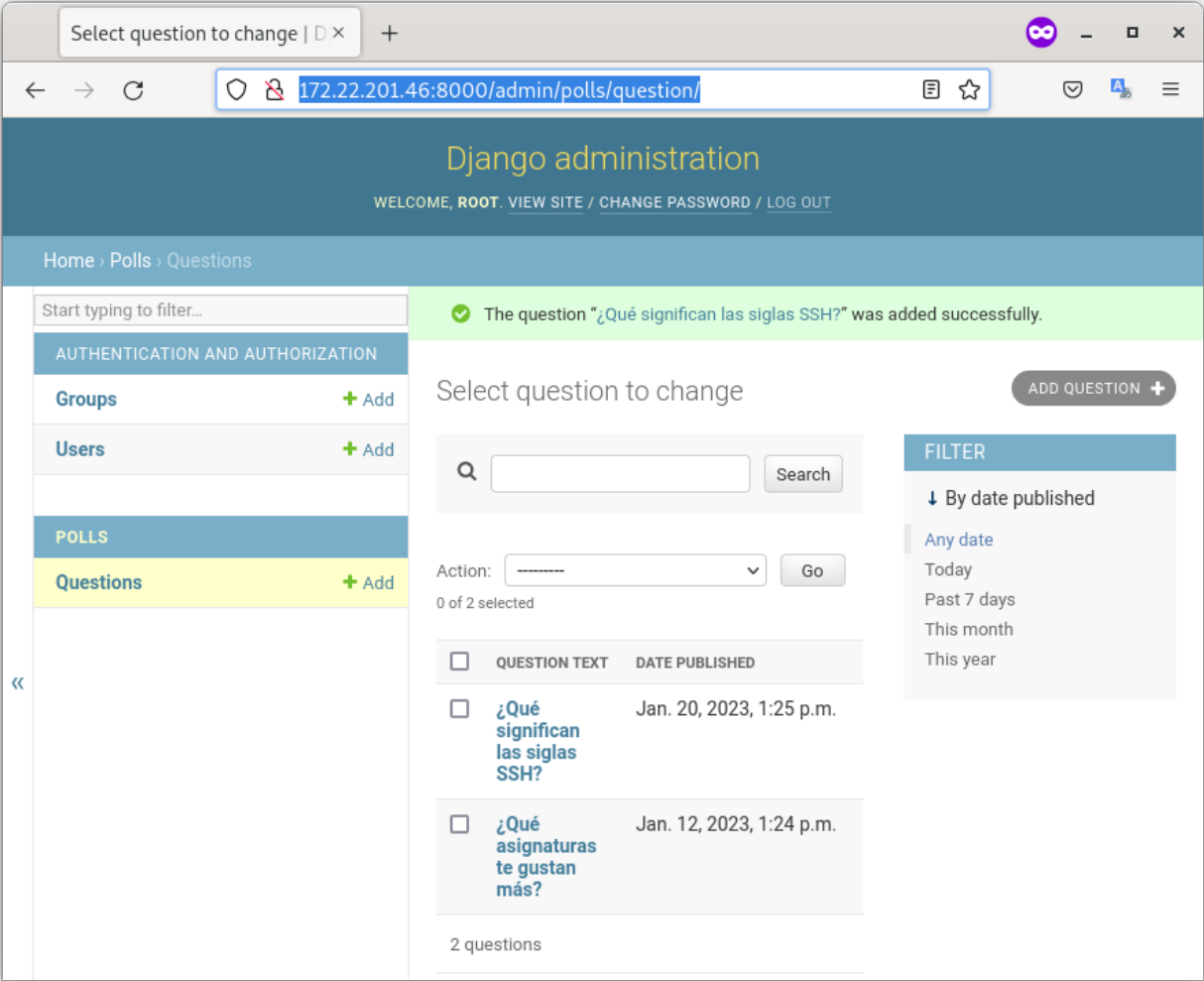
```
sudo reboot
```

- Comprobamos que podemos acceder a la aplicación desde el navegador, sobre todo en la zona de administración:





- Creamos dos preguntas con varias respuestas.





Vamos a configurar el servidor web apache2 con el módulo wsgi para servir la página web. Al utilizar como entorno de desarrollo la máquina bravo, se accederá con el nombre `python.mariajesus.gonzalonazareno.org`.

- Para reunir los ficheros estáticos, ejecutamos el siguiente comando:

```
python django_tutorial/manage.py collectstatic
```

- Pero antes de ello, tenemos que modificar el fichero `settings.py` para que el servidor web de desarrollo escuche en todas las interfaces:

```
STATIC_ROOT = '/var/www/html/Django_Tutorial/static'
```

- Instalamos el módulo wsgi para apache2:

```
sudo dnf install mod_wsgi httpd
```

- Creamos el VirtualHost que servirá la web:

```
nano /etc/httpd/sites-available/django_tutorial.conf

---

<VirtualHost *:80>
    ServerName python.mariajesus.gonzalonazareno.org
    DocumentRoot /var/www/html/Tutorial_Django
    Alias /static/ /var/www/html/Tutorial-Django/static/
    WSGIDaemonProcess Tutorial_Django python-
path=/var/www/html/Tutorial_Django:/var/www/html/django/lib/python3.9/
site-packages
    WSGIProcessGroup Tutorial_Django
    WSGIScriptAlias /
/var/www/html/Tutorial_Django/tutorial_django/wsgi.py
    ErrorLog /var/log/httpd/Tutorial_Django_error.log
    CustomLog /var/log/httpd/Tutorial_Django_access.log combined
</VirtualHost>
```

Lo activamos:

```
``bash
ln -s /etc/httpd/sites-available/django_tutorial.conf
/etc/httpd/sites-enabled/django_tutorial.conf
``
```

- Reiniciamos el servicio apache2:

```
sudo systemctl restart httpd
```

- Nos dirigimos a la máquina **charlie**, que es el servidor DNS, y añadimos la nueva zona DNS:

```
nano /var/cache/bind/db.interna.mariajesus.gonzalonazareno.org
nano /var/cache/bind/db.dmz.mariajesus.gonzalonazareno.org
---
python IN A bravo
```

Refrescamos la zona DNS:

```
```bash
rndc flush
systemctl restart bind9
```
```

- Comprobamos que podemos acceder a la aplicación desde el navegador:

