		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	1311	Práctica 1	M	Fecha	dd/mm/aaaa
Alumno/a		De La Fuente, Simón, Iñaki			
Alumno/a		Serrano, Zurita, Juan Freddy			

Práctica 1: Título

Cuestión número 1:

Ejercicio 1: incluya en la memoria de la practica pruebas de que se ha desplegado la aplicación web localmente y de que esta funciona. Al menos se debe incluir una copia del contenido del chero env así como una captura de pantalla en la que se muestre el resultado de interaccionar con cada uno de los endpoints. Las capturas deben mostrar el URL al que se conecta el navegador.

Hemos desplegado la aplicación, pero para ello utilizamos la base de datos proporcionada por la máquina virtual uno. Tras su puesta en marcha y creación de la base de datos voto, hemos cambiado el puerto en el fichero env ya que por el mapeo de puertos ese es el de psql.

```
# IMPORTANT: this file should not be in a repository
# To remove a file named env from a Git repository
# but keep it in the source (local system), follow these steps:
# Remove the file from Git tracking but keep it locally
## git rm --cached env
# Add 'env' to .gitignore (so it's not tracked again)
## echo "env" >> .gitignore
# Commit the changes
## git commit -m "Removed env from Git tracking and added to .gitignore"
# Push the changes to the remote repository
## git push
# use sqlite 3
##DATABASE_SERVER_URL=sqlite:///db.sqlite3
# use postgres
DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@localhost:15432/voto'
# The client does not need to store data in any database
# so let us define a sqlite in orden to avoid warning messages
DEBUG=True
SECRET_KEY = 'django-insecure-alczftn)j1#$v%xmk@5j(n*px43c8kxgi_ua4%khc+t7g_)s9d'
```

En la siguiente imagen podemos observar la página web en su estado inicial, tambien hemos añadido una dirección por defecto ya que al inicio nos encontrábamos con un error de 404 not found:

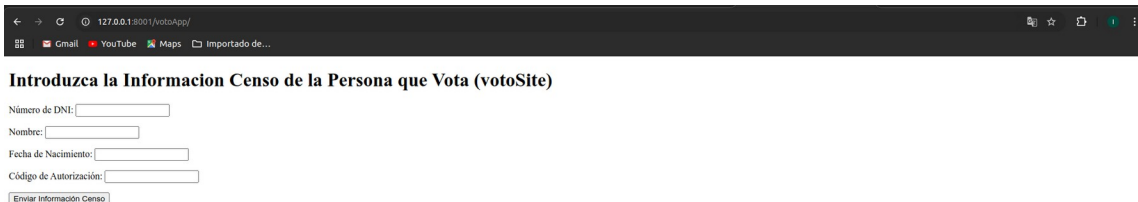
```

from django.contrib import admin
from django.urls import include, path
from django.views.generic import RedirectView

urlpatterns = [
    path('admin/', admin.site.urls),
    path("votoApp/", include("votoApp.urls")),
    path('', RedirectView.as_view(url='votoApp/', permanent=True)),
]

```

La página web en cuestión:



Introduzca la Información Censo de la Persona que Vota (votoSite)

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Cuestión número 2:

Ejercicio 2: ejecuta los test proporcionados con el proyecto base y comprueba que no devuelven errores. Adjunta en la memoria una captura de pantalla en la que se muestre el resultado de ejecutar los test (python manage.py test). Los test proporcionados deben tomarse como requisitos extras del sistema.

Tras la puesta en marcha y verificación de que todo funcionase, se ejecutaron los tests propuestos, dando un resultado muy favorable y sin ningún error:

```

(venv) e482884@1-8-1-8:~/SI2/P1/P1-base$ python manage.py test
Found 18 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..Error: Registrando voto: null value in column "censo_id" of relation "voto" violates not-null constraint
DETAIL:  Failing row contains (1, C001, M001, P001, Candidato A, 2025-02-10 10:39:37.963944+00, 000, null).
.....
-----
Ran 18 tests in 0.168s

OK
Destroying test database for alias 'default'...
(venv) e482884@1-8-1-8:~/SI2/P1/P1-base$

```

Cuestión número 3:

Ejercicio 3: incluya en la memoria de la practica pruebas de que se ha desplegado la aplicación web P1-base usando las máquinas virtuales y de que esta funciona. Al

menos se debe incluir una copia del contenido del #chero env así como una captura de pantalla en la que se muestre el resultado de registrar un voto. Esta última debe mostrar claramente el URL al que se conecta el navegador.

Primero iniciamos la MV1, esta será la que tendrá la base de datos, hacemos el populate que otorga el manage.py para tener personas en el censo.

Arrancamos la MV2 y realizamos el anexo B, una vez realizamos el anexo A y B; Dentro de la MV2 en el fichero env debemos modificar la ruta de la base de datos, hacemos ifconfig en el host para sacar su ip; esta la introducimos en la ruta para el acceso a la base de datos como se muestra:

```
GNU nano 7.2 env
# IMPORTANT: this file should not be in a repository
# To remove a file named env from a Git repository
# but keep it in the source (local system), follow these steps:
# Remove the file from Git tracking but keep it locally
## git rm --cached env
# Add 'env' to .gitignore (so it's not tracked again)
## echo "env" >> .gitignore
# Commit the changes
## git commit -m "Removed env from Git tracking and added to .gitignore"
# Push the changes to the remote repository
## git push
# use sqlite 3
##DATABASE_SERVER_URL=sqlite:///db.sqlite3
# use postgres
DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@172.18.214.215:15432/voto'
# The client does not need to store data in any database
# so let us define a sqlite in order to avoid warning messages
DEBUG=True
SECRET_KEY = 'django-insecure-alczftn)j1#5v%xmK05j(n*px43c8kxgi_ua4%khc+t7g_)s9d'
```

una vez tenemos gunicorn nos muestra lo siguiente:

```
(p1_env) si2@si2-ubuntu-vm-2:~/repo$ sudo systemctl daemon-reload
(p1_env) si2@si2-ubuntu-vm-2:~/repo$ sudo systemctl enable gunicorn
(p1_env) si2@si2-ubuntu-vm-2:~/repo$ sudo systemctl start gunicorn
(p1_env) si2@si2-ubuntu-vm-2:~/repo$ sudo systemctl status gunicorn
● gunicorn.service - Gunicorn WSGI Application Server
   Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-02-17 14:36:30 CET; 5s ago
     Main PID: 3676 (gunicorn)
        Tasks: 2 (limit: 1672)
       Memory: 43.0M (peak: 43.3M)
          CPU: 1.718s
      CGroup: /system.slice/gunicorn.service
              └─3676 /home/si2/repo/p1_env/bin/python3 /home/si2/repo/p1_env/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application
                 3677 /home/si2/repo/p1_env/bin/python3 /home/si2/repo/p1_env/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application

feb 17 14:36:30 si2-ubuntu-vm-2 systemd[1]: Started gunicorn.service - Gunicorn WSGI Application Server.
feb 17 14:36:31 si2-ubuntu-vm-2 gunicorn[3676]: [2025-02-17 14:36:31 +0100] [3676] [INFO] Starting gunicorn 20.1.0
feb 17 14:36:31 si2-ubuntu-vm-2 gunicorn[3676]: [2025-02-17 14:36:31 +0100] [3676] [INFO] Listening at: http://0.0.0.0:8000 (3676)
feb 17 14:36:31 si2-ubuntu-vm-2 gunicorn[3676]: [2025-02-17 14:36:31 +0100] [3676] [INFO] Using worker: sync
feb 17 14:36:31 si2-ubuntu-vm-2 gunicorn[3677]: [2025-02-17 14:36:31 +0100] [3677] [INFO] Booting worker with pid: 3677
(p1_env) si2@si2-ubuntu-vm-2:~/repo$
```

podremos comprobar que se está ejecutando la aplicación desde el navegador del host mediante la dirección localhost:28000; 28000 ya que ese es el puerto en la MV2:

Introduzca la Informacion Censo de la Persona que Vota (votoSite)

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Para saber con que campos rellenarlo; nos metemos con el admin (iniciando sesión con el superusuario creado) para saber alguna persona que esté en el censo, en este caso nos decantamos con el primero que nos aparecía, es el siguiente:

Emiliano San Dans (DNI: 99999999I)

NumeroDNI:

99999999I

Nombre:

Emiliano San Dans

FechaNacimiento:

22/03/55

AnioCenso:

2025

CodigoAutorizacion:

512

SAVE

Save and add another

Save and continue editing

Tras rellenar los campos con esta información, nos aparece la sección de registro de voto y rellenamos los campos pertinentes:

Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

ID Proceso Electoral:

ID Circunscripcion:

ID Mesa Electoral:

Nombre Candidato Votado:

Una vez finalizado esto, tenemos la confirmación:

Voto Registrado con Éxito (votoSite)

Id: 1

Codigo Respuesta: 000

Marca Tiempo : Feb. 17, 2025, 2:59 p.m.

Id Circunscripcion : 1

Id Mesa Electoral : 1

Id Proceso Electoral : 1

Nombre Candidato Votado: bollos

Cuestión número 4:

Ejercicio 4: El script de python incluido en el apéndice C lee 1000 entradas de la base de datos #voto# una a una e imprime el tiempo de lectura. Ejecútalo desde el ordenador host y reporta en la memoria de la práctica los siguientes casos:

1. la base de datos está en la máquina virtual VM1.
2. la base de datos no esta en la red local. Crea un base de datos llamada voto en <http://www.neon.tech> y ejecuta el script contra esa base de datos
3. no se accede a la base de datos directamente. En Django, el mapeo de clases a tablas se implementa mediante el ORM (Object-Relational Mapping), re-escribe el script para que los votos se lean usando el sistema de modelos de Django. Por ejemplo 'SELECT * FROM censo WHERE #numeroDNI# = %s' debería ser Voto.objects.get(pk= %s). En este nuevo script no debe aparecer código SQL).

Repite cada medida 7 veces y reporta el valor medio de las mismas así como su desviación estándar. Además de los tiempos de lectura incluye en la memoria el script reescrito, el contenido del #chero env en cada caso así como un comentario sobre los resultados.

Ya teníamos la disposición necesaria del anterior ejercicio, solo nos queda ver la configuración del script y asegurarnos que es la correcta para conectarnos a la base de datos de la MV1:

```
# Configuración de la base de datos
db_config = {
    'dbname': 'voto', # Nombre de la base de datos
    'user': 'alumnodb', # Reemplaza con tu usuario de PostgreSQL
    'password': 'alumnodb', # Reemplaza con tu contraseña
    'host': 'localhost', # Cambia si el host es diferente
    'port': 15432, # Cambia si tu puerto es diferente
}
```

Se ha realizado un script que automatiza las pruebas, podemos ver el resultado de las mismas:

```
(p1_env) e482884@1-12-1-12:~/SI2/P1/P1-base$ ./scriptTiempos.sh
Ejecutando pruebas...

1. Base de datos en VM1 con read_1000_entries_from_db.py
Tiempo de ejecución 1: 0.218534 segundos
Tiempo de ejecución 2: 0.211985 segundos
Tiempo de ejecución 3: 0.215515 segundos
Tiempo de ejecución 4: 0.220370 segundos
Tiempo de ejecución 5: 0.212136 segundos
Tiempo de ejecución 6: 0.212443 segundos
Tiempo de ejecución 7: 0.228648 segundos
Media: .21709014285714285714 segundos
Desviación estándar: .00562023480517550173 segundos
```

Esta configuración presenta tiempos de respuesta relativamente estables y bajos, gracias a la proximidad

entre el cliente y el servidor junto con el uso de sentencias SQL.

Neon:

Para la conexión con neon, tras el registro y puesta en marcha de la base de datos en su página web oficial; es hora de modificar el archivo de prueba, rellenándolo con las características que nos indica neon:

```
# Configuración de la base de datos
db_config = {
    'dbname': 'voto', # Nombre de la base de datos
    'user': 'alumnodb', # Reemplaza con tu usuario de PostgreSQL
    'password': 'npg_QfNYwtd4ERn7', # Reemplaza con tu contraseña
    'host': 'ep-steep-pine-a8gb1b6g-pooler.eastus2.azure.neon.tech', # Cambia si el host es diferente
    'port': 5432, # Cambia si tu puerto es diferente
}
```

para hacer el populate también realizamos los cambios necesarios en el proyecto:

en el env:

```
# use postgres
#MV1
#DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@localhost:5432/voto'
#neon
DATABASE_URL='postgresql://alumnodb:npg_QfNYwtd4ERn7@ep-steep-pine-a8gb1b6g-pooler.eastus2.azure.neon.tech/voto?sslmode=require'
```

Una vez ejecutado el populate, ejecutamos nuestro script de tiempos, podemos observar un aumento de tiempo muy considerable:

```
Tiempo de ejecución 1: 25.167813 segundos
Tiempo de ejecución 2: 24.499718 segundos
Tiempo de ejecución 3: 24.403118 segundos
Tiempo de ejecución 4: 24.669055 segundos
Tiempo de ejecución 5: 24.510952 segundos
Tiempo de ejecución 6: 24.249388 segundos
Tiempo de ejecución 7: 24.388998 segundos
Media: 24.55557742857142857142 segundos
Desviación estándar: .27711931631589262978 segundos
```

En esta configuración se observa un incremento significativo en el tiempo del test debido a la latencia de la red y el acceso a un servidor remoto. Además no podemos esperar mucho de un servicio gratuito.

Sentencias ORM:

Para realizar el último apartado podemos utilizar como base de datos la MV1 o neon de nuevo, vamos a comparar tiempos; para ello también es necesario modificar el archivo de pruebas para usar ORM, el resultado es el siguiente:

```

import time
import os
import django

# Configurar el entorno de Django
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'votoSite.settings')
django.setup()

from votoApp.models import Censo

def realizar_búsquedas():
    # Medir el tiempo de inicio
    start_time = time.time()

    # Obtener las primeras 1000 entradas de la tabla 'censo'
    censos = Censo.objects.all()[:1000] # O usando `.filter()` si necesitas aplicar algún filtro

    # Realizar búsquedas una por una
    for censo in censos:
        # Obtener el censo usando el 'numeroDNI'
        try:
            censo_encontrado = Censo.objects.get(numeroDNI=censo.numeroDNI)
            # Realiza cualquier procesamiento con censo_encontrado si es necesario
        except Censo.DoesNotExist:
            # Si no se encuentra un censo, manejar el caso de error
            print(f"Censo con numeroDNI {censo.numeroDNI} no encontrado.")

    # Medir el tiempo de finalización
    end_time = time.time()

    # Mostrar los resultados
    print(f"Tiempo invertido en buscar las 1000 entradas una a una: {end_time - start_time:.6f} segundos")
realizar_búsquedas()

```

Tras ejecutar el script de tiempos modificado en la MV1, se ha modificado el env, como visto antes:

```

3. Base de datos en VM1 con read_1000_entries_from_db_modified.py
Tiempo de ejecución 1: 0.757448 segundos
Tiempo de ejecución 2: 0.734979 segundos
Tiempo de ejecución 3: 0.747319 segundos
Tiempo de ejecución 4: 0.763862 segundos
Tiempo de ejecución 5: 0.746675 segundos
Tiempo de ejecución 6: 0.751223 segundos
Tiempo de ejecución 7: 0.762511 segundos
Media: .75200242857142857142 segundos
Desviación estándar: .00942227136488168546 segundos

```

Cabe destacar que es ligeramente más lento que su test homólogo ya que tiene la sobrecarga de utilizar ORM (object relational mapping).

Tras ejecutar el script de tiempos modificado en neon, se ha modificado el env, como visto antes:


```
Tiempo de ejecución 1: 61.046521 segundos
Tiempo de ejecución 2: 59.964181 segundos
Tiempo de ejecución 3: 60.331365 segundos
Tiempo de ejecución 4: 60.626618 segundos
Tiempo de ejecución 5: 61.194774 segundos
Tiempo de ejecución 6: 61.865202 segundos
Tiempo de ejecución 7: 62.143140 segundos
Media: 61.024543000000000000 segundos
Desviación estándar: .73198712370486018386 segundos
```

Esta es la configuración con la mayor latencia, combinando la penalización del ORM y el acceso remoto. En comparación a la más rápida es unas 280 veces más lenta; lo que supone un notable impacto.

***cuestión interna de la pregunta 4**

Question 1: En el ejercicio anterior 7 repeticiones de cada medida proporcionan una estimación fiable? Porqué?.

7 repeticiones proporcionan una estimación fiable. En términos generales, el número de mediciones debe ser suficiente para obtener una estimación representativa del tiempo medio real de lectura.

- Si la desviación estándar es baja, significa que las mediciones son consistentes y 7 repeticiones pueden ser suficientes.

- Si la desviación estándar es alta, indica una alta variabilidad en los tiempos de lectura, lo que sugiere que el número de repeticiones es insuficiente.

En este caso, los resultados obtenidos muestran que en Neon la desviación estándar es considerablemente mayor, lo que sugiere que factores externos (carga de la red, congestión del servidor) están afectando los tiempos de lectura. Para obtener una mejor estimación en este caso, sería recomendable aumentar el número de repeticiones.

Conclusiones

- La lectura desde una base de datos local (VM1) es significativamente más rápida que desde un servidor remoto (Neon).
- El uso de ORM introduce una penalización en el tiempo de lectura respecto a las consultas SQL directas.
- La variabilidad en los tiempos de respuesta es mayor en conexiones remotas, lo que sugiere la necesidad de realizar más mediciones.

Cuestión número 5:

Ejercicio 5: ejecuta los test proporcionados con el proyecto P1-base y comprueba que no devuelven errores. Adjunta en la memoria una captura de pantalla en la que se muestre el resultado de ejecutar los test incluyendo el comando utilizado para lanzarlos. En esta ejecución la base de datos debe estar en VM1 y la aplicación web en VM2.

Realizamos los ajustes necesarios, y tras arrancar las VM1 y VM2; dentro de la VM2 procedemos a ejecutar los tests que se encuentran en P1-base, están en votoApp y son tests_models y tests_views. Los hemos ejecutado con estos comandos, dando los dos correctos:

python3 manage.py test votoApp.tests_models --verbosity 2

python3 manage.py test votoApp.tests_views --verbosity 2

```
(p1_env) si2@si2-ubuntu-vn-2:~/repo/pibase/P1/P1-base$ python3 manage.py test votoApp.tests_models --verbosity 2
Found 9 test(s).
Creating test database for alias 'default' ('test_voto')...
Operations to perform:
  Synchronize unmigrated apps: messages, staticfiles
  Apply all migrations: admin, auth, contenttypes, sessions, votoApp
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
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 sessions.0001_initial... OK
  Applying votoApp.0001_initial... OK
System check identified no issues (0 silenced).
test_00_censo_creation (votoApp.tests_models.CensoModelTest.test_00_censo_creation)
Test that a Censo instance is created correctly. ... ok
test_01_unique_numeroONI (votoApp.tests_models.CensoModelTest.test_01_unique_numeroONI)
Test that numeroONI is unique. ... ok
test_registrar_voto_invalid (votoApp.tests_models.RegistrarVotoTests.test_registrar_voto_invalid) ... Error: Registrando voto: null value in column 'censo_id' of relation 'voto' violates not-null constraint
DETAIL:  Falling row contains (1, C001, M001, P001, Candidato A, 2025-02-17 17:11:45.021071+00, 000, null).

ok
test_registrar_voto_valid (votoApp.tests_models.RegistrarVotoTests.test_registrar_voto_valid) ... ok
test_verificar_censo_invalid (votoApp.tests_models.VerificarCensoTests.test_verificar_censo_invalid) ... ok
test_verificar_censo_valid (votoApp.tests_models.VerificarCensoTests.test_verificar_censo_valid) ... ok
test_01_voto_creation (votoApp.tests_models.VotoModelTest.test_01_voto_creation)
Test that a Voto instance is created correctly ... ok
test_02_default_codigo_respuesta (votoApp.tests_models.VotoModelTest.test_02_default_codigo_respuesta)
Test that the default codigoRespuesta is RESPUESTA_OK. ... ok
test_03_unique_constraint (votoApp.tests_models.VotoModelTest.test_03_unique_constraint)
Test that unique constraint ... ok

-----
Ran 9 tests in 1.486s

OK
Destroying test database for alias 'default' ('test_voto')...
(p1_env) si2@si2-ubuntu-vn-2:~/repo/pibase/P1/P1-base$
```

Y ahora el de views:

```
(p1_env) si2@si2-ubuntu-vn-2:~/repo/pibase/P1/P1-base$ python3 manage.py test votoApp.tests_views --verbosity 2
Found 9 test(s).
Creating test database for alias 'default' ('test_voto')...
Operations to perform:
  Synchronize unmigrated apps: messages, staticfiles
  Apply all migrations: admin, auth, contenttypes, sessions, votoApp
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
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 sessions.0001_initial... OK
  Applying votoApp.0001_initial... OK
System check identified no issues (0 silenced).
test_00_aportarinfo_censo_valid_submission (votoApp.tests_views.VotoCensoViewsTest.test_00_aportarinfo_censo_valid_submission) ... ok
test_01_aportarinfo_censo_invalid_submission (votoApp.tests_views.VotoCensoViewsTest.test_01_aportarinfo_censo_invalid_submission) ... ok
test_05_aportarinfo_voto_with_valid_censo_data (votoApp.tests_views.VotoCensoViewsTest.test_05_aportarinfo_voto_with_valid_censo_data) ... ok
test_06_aportarinfo_voto_without_censo_data (votoApp.tests_views.VotoCensoViewsTest.test_06_aportarinfo_voto_without_censo_data) ... ok
test_10_testbd_valid_submission (votoApp.tests_views.VotoCensoViewsTest.test_10_testbd_valid_submission) ... ok
test_15_delvoto_valid_deletion (votoApp.tests_views.VotoCensoViewsTest.test_15_delvoto_valid_deletion) ... ok
test_20_delvoto_invalid_id (votoApp.tests_views.VotoCensoViewsTest.test_20_delvoto_invalid_id) ... ok
test_25_getvotos_valid_idProcesoElectoral (votoApp.tests_views.VotoCensoViewsTest.test_25_getvotos_valid_idProcesoElectoral) ... ok
test_30_testbd_invalid_submission (votoApp.tests_views.VotoCensoViewsTest.test_30_testbd_invalid_submission) ... ok

-----
Ran 9 tests in 0.607s

OK
Destroying test database for alias 'default' ('test_voto')...
(p1_env) si2@si2-ubuntu-vn-2:~/repo/pibase/P1/P1-base$
```

Cuestión número 6:

Ejercicio 6: ejecuta los test proporcionados con el proyecto P1-ws-server aplicación votoAppWSServer y comprueba que no devuelven errores. Adjunta en la memoria una captura de pantalla en la que se muestre el resultado de ejecutar los test junto con el comando utilizado. En esta ejecución la base de datos debe estar en VM1 y la aplicación web en VM2.

Realizamos los cambios necesarios, arrancamos las VM1 y VM2; dentro de la VM2 procedemos a ejecutar los tests que se encuentran en P1-ws-server, están en votoAppWSServer en tests_views. Los hemos ejecutado con este comando, obteniendo los resultados correctamente:

```
python3 manage.py test votoAppWSServer.test_views --verbosity 2
```

```
si2@si2-ubuntu-vm-2:~/repo/pibase/P1/P1-ws-server$ python3 manage.py test votoAppWSServer.test_views --verbosity 2
Found 6 test(s).
Creating test database for alias 'default' ('test_voto')...
Operations to perform:
  Synchronize unmigrated apps: messages, rest_framework, staticfiles
  Apply all migrations: admin, auth, contenttypes, sessions, votoAppWSServer
Synchronizing apps without migrations:
  Creating tables...
  Running deferred SQL...
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 sessions.0001_initial... OK
  Applying votoAppWSServer.0001_initial... OK
System check identified no issues (0 silenced).
test_01_censo_check_valid_data (votoAppWSServer.test_views.ApiViewTest.test_01_censo_check_valid_data) ... ok
test_02_censo_check_invalid_data (votoAppWSServer.test_views.ApiViewTest.test_02_censo_check_invalid_data) ... ok
test_05_list_votos (votoAppWSServer.test_views.ApiViewTest.test_05_list_votos) ... ok
test_10_voto_store (votoAppWSServer.test_views.ApiViewTest.test_10_voto_store) ... ok
test_20_delete_existing_voto (votoAppWSServer.test_views.ApiViewTest.test_20_delete_existing_voto)
Test deleting an existing Voto object. ... ok
test_21_delete_nonexistent_voto (votoAppWSServer.test_views.ApiViewTest.test_21_delete_nonexistent_voto)
Test attempting to delete a Voto object that does not exist. ... ok
.....
Ran 6 tests in 0.059s

OK
```

Cuestión número 7:

Ejercicio 7: conéctate a los diferentes endpoints usando el navegador (<http://<hostname>:28000/restapiserver/xxxx/>) y muestra las pantallas resultantes de usar cada uno de ellos. Suministra al sistema datos válidos e inválidos. Junto a cada imagen debes incluir: (1) el URL al que te conectas, (2) el método usado (POST/GET/DELETE) y (3) en caso de usar el método POST los datos enviados en la petición.

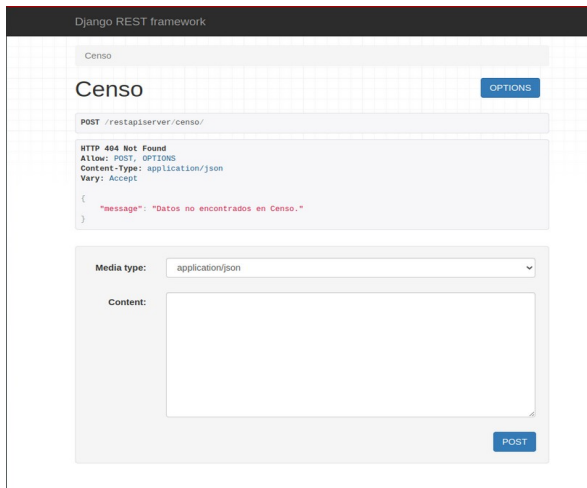
<http://localhost:28000/restapiserver/censo/>

POST, datos incorrectos.

Content:

```
{
```

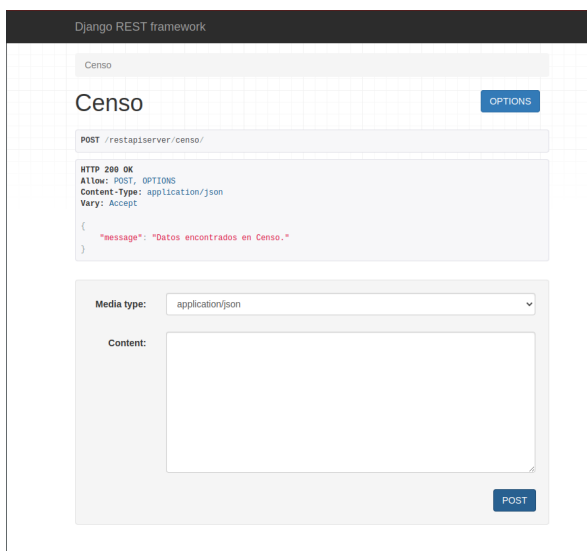
```
"numeroDNI": "12345678",
"nombre": "Juan",
"fechaNacimiento": "19900101",
"codigoAutorizacion": "123"
}
```



POST, datos correctos.

Content:

```
{
  "numeroDNI": "99799799D",
  "nombre": "Luisa Coll Torres",
  "fechaNacimiento": "05/04/70",
  "codigoAutorizacion": "508"
}
```



<http://localhost:28000/restapiserver/voto/>

POST, datos incorrectos.

Content:

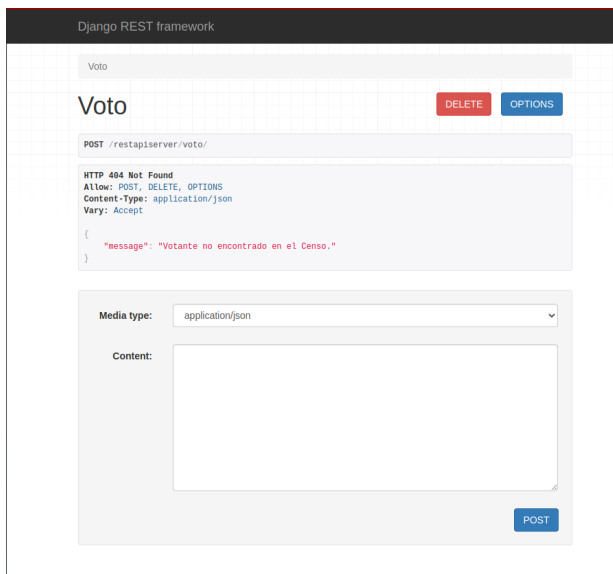
```
{
  "idCircunscripcion": "123",

```

```

"idMesaElectoral": "456",
"idProcesoElectoral": "789",
"nombreCandidatoVotado": "Candidato A",
"censo_id": "99999999A"
}

```



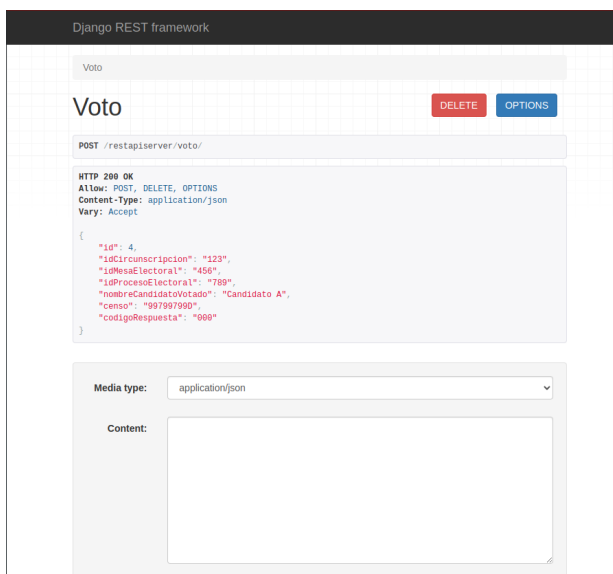
POST, datos correctos.

Content:

```

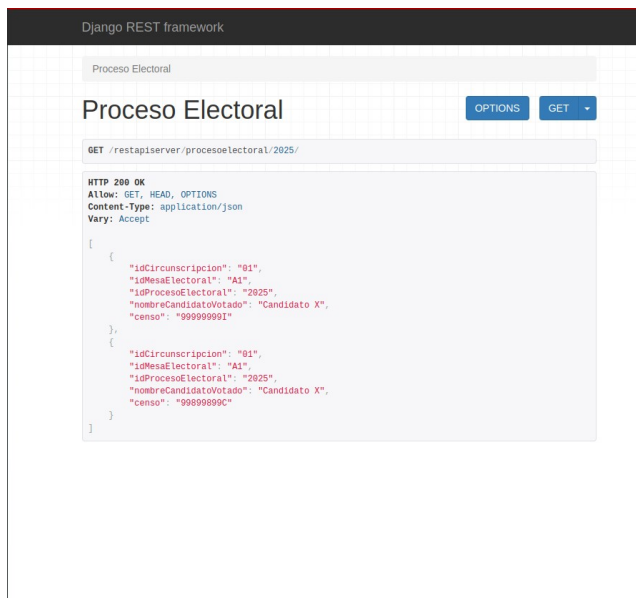
{
"idCircunscripcion": "123",
"idMesaElectoral": "456",
"idProcesoElectoral": "789",
"nombreCandidatoVotado": "Candidato A",
"censo_id": "99799799D"
}

```



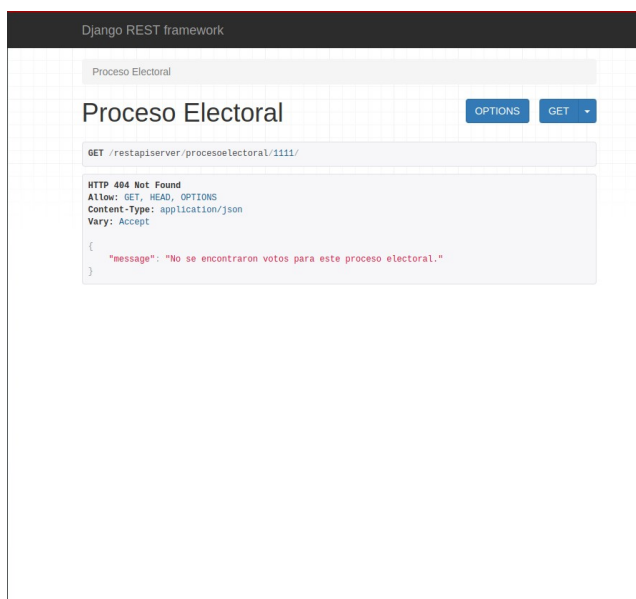
<http://localhost:28000/restapiserver/procesoelectoral/2025/>

GET, datos correctos(idProcesoElectoral).



<http://localhost:28000/restapiserver/procesoelectoral/1111/>

GET, datos incorrectos.



<http://localhost:28000/restapiserver/voto/4/>

DELETE, datos correctos.

Django REST framework

Voto / Voto

Voto

DELETE /restapiserver/voto/4/

HTTP 200 OK
Allow: POST, DELETE, OPTIONS
Content-Type: application/json
Vary: Accept
{
 "message": "Voto eliminado correctamente."
}

Media type: application/json

Content:

POST

<http://localhost:28000/restapiserver/voto/1234/>

DELETE, datos incorrectos.

Django REST framework

Voto / Voto

Voto

DELETE /restapiserver/voto/1234/

HTTP 404 Not Found
Allow: POST, DELETE, OPTIONS
Content-Type: application/json
Vary: Accept
{
 "message": "Voto no encontrado."
}

Media type: application/json

Content:

POST

Cuestión número 8:

Ejercicio 8: ejecuta los test proporcionados con el proyecto P1-ws-client aplicación votoAppWSclient y comprueba que no devuelven errores. Adjunta en la memoria una captura de pantalla en la que se muestre el resultado de ejecutar los test. En esta ejecución la base de datos debe estar en VM1 y la aplicación web en VM2.

Para ejecutar el estado propuesto, el cliente debe tener este entorno; ya que el servidor está en la misma MV, la MV2; la url de la base de datos es la misma que la del servidor.

```
# IMPORTANT: this file should not be in a repository
# To remove a file named env from a Git repository
# but keep it in the source (local system), follow these steps:
# Remove the file from Git tracking but keep it locally
## git rm --cached env
# Add 'env' to .gitignore (so it's not tracked again)
## echo "env" >> .gitignore
# Commit the changes
## git commit -m "Removed env from Git tracking and added to .gitignore"
# Push the changes to the remote repository
## git push
# use sqlite 3
##DATABASE_SERVER_URL=sqlite:///db.sqlite3
# use postgres
#DATABASE_SERVER_URL='postgresql://neondb_owner:npg_6WJxtXRPzZ8D@ep-steep-pine-a8gb1b6
DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@172.27.239.131:15432/voto'
RESTAPIBASEURL='http://localhost:8000/restapiserver/'
# The client does not need to store data in any database
# so let us define a sqlite in order to avoid warning messages
DEBUG=True
SECRET_KEY = 'django-insecure-alczftn)j1#$v%xmK@5j(n*px43c8kxgi_ua4%khc+t7g_)s9d'
```

Como podemos observar en el env del servidor, la url es la misma:

```

# IMPORTANT: this file should not be in a repository
# To remove a file named env from a Git repository
# but keep it in the source (local system), follow these steps:
# Remove the file from Git tracking but keep it locally
## git rm --cached env
# Add 'env' to .gitignore (so it's not tracked again)
## echo "env" >> .gitignore
# Commit the changes
## git commit -m "Removed env from Git tracking and added to .gitignore"
# Push the changes to the remote repository
## git push
# use sqlite 3
#DATABASE_SERVER_URL=sqlite:///db.sqlite3
# use postgres
DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@172.27.239.131:15432/voto'
# The client does not need to store data in any database
# so let us define a sqlite in orden to avoid warning messages
DEBUG=True
SECRET_KEY = 'django-insecure-alczftn)jl#$v%xmkg@5j(n*px43c8kxgi_ua4%khc+t7g_)s9d'

```

Para poder continuar con el ejercicio, gunicorn debe estar corriendo y ejecutando el servidor, como se puede ver en la captura:

```

si2@si2-ubuntu-vm-2:~$ sudo systemctl status gunicorn
● gunicorn.service - Gunicorn WSGI Application Server
   Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-02-23 19:18:39 CET; 8s ago
     Main PID: 12673 (gunicorn)
        Tasks: 2 (limit: 1672)
      Memory: 42.9M (peak: 43.1M)
         CPU: 557ms
    CGroup: /system.slice/gunicorn.service
            └─12673 /home/si2/p1_env/bin/python3 /home/si2/p1_env/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application
              12674 /home/si2/p1_env/bin/python3 /home/si2/p1_env/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application

feb 23 19:18:39 si2-ubuntu-vm-2 systemd[1]: Started gunicorn.service - Gunicorn WSGI Application Server.
feb 23 19:18:40 si2-ubuntu-vm-2 gunicorn[12673]: [2025-02-23 19:18:40 +0100] [12673] [INFO] Starting gunicorn 20.1.0
feb 23 19:18:40 si2-ubuntu-vm-2 gunicorn[12673]: [2025-02-23 19:18:40 +0100] [12673] [INFO] Listening at: http://0.0.0.0:8000 (12673)
feb 23 19:18:40 si2-ubuntu-vm-2 gunicorn[12673]: [2025-02-23 19:18:40 +0100] [12673] [INFO] Using worker: sync
feb 23 19:18:40 si2-ubuntu-vm-2 gunicorn[12674]: [2025-02-23 19:18:40 +0100] [12674] [INFO] Booting worker with pid: 12674
si2@si2-ubuntu-vm-2:~$

```

Tras esto ya tenemos todos los pasos, ahora solo queda entrar en el entorno virtual y ejecutar el test_views.py:

```

(p1_env) si2@si2-ubuntu-vm-2:~/P1-ws-client$ python3 manage.py test votoAppWSClient.test_views --verbosity 2
Found 8 test(s).
Creating test database for alias 'default' ('test_voto')...
Operations to perform:
  Synchronize unmigrated apps: messages, staticfiles, votoAppWSClient
  Apply all migrations: admin, auth, contenttypes, sessions
  Synchronizing apps without migrations:
    Creating tables...
    Running deferred SQL...
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 sessions.0001_initial... OK
System check identified no issues (0 silenced).
test_015_aportarInfo_censo_invalid_post (votoAppWSClient.test_views.VotingViewsTest.test_015_aportarInfo_censo_invalid_post)
check invalid censo entry ... deleting votes
DELETE 1
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_016_aportarInfo_censo_valid_post (votoAppWSClient.test_views.VotingViewsTest.test_016_aportarInfo_censo_valid_post)
Check Censo information ... deleting votes
DELETE 0
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_02_store_voto_valid_post (votoAppWSClient.test_views.VotingViewsTest.test_02_store_voto_valid_post)
Create and save a 'voto' ... deleting votes
DELETE 0
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_033_delvoto_invalid_post (votoAppWSClient.test_views.VotingViewsTest.test_033_delvoto_invalid_post)
Test deleting an existing Voto object. ... deleting votes
DELETE 1
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_034_delvoto_valid_post (votoAppWSClient.test_views.VotingViewsTest.test_034_delvoto_valid_post)
Test deleting an existing Voto object. ... deleting votes
DELETE 1
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_04_getVotos_post (votoAppWSClient.test_views.VotingViewsTest.test_04_getVotos_post)
Test deleting an existing Voto object. ... deleting votes
DELETE 0
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_10_testdb_post (votoAppWSClient.test_views.VotingViewsTest.test_10_testdb_post) ... deleting votes
DELETE 3
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
test_11_testdb_invalid_post (votoAppWSClient.test_views.VotingViewsTest.test_11_testdb_invalid_post) ... deleting votes
DELETE 1
This test only works if (1) http://localhost:8000/restapiserver/
server is up and running and (2) the database is populated.
Very likely in the future this test should be done with mocks
and not real calls to the server.
ok
-----
Ran 8 tests in 2.577s
OK
Destroying test database for alias 'default' ('test_voto')...
(p1_env) si2@si2-ubuntu-vm-2:~/P1-ws-client$

```

Como no se ve por completo, aquí esta lo más importante:

Ejecución correcta y aprobada de todas las pruebas:

```

-----
Ran 8 tests in 2.577s

OK
Destroying test database for alias 'default' ('test_voto')...
(p1_env) si2@si2-ubuntu-vm-2:~/P1-ws-client$

```

Comando en cuestión y se puede observar su ejecución en la MV2:

```

(p1_env) si2@si2-ubuntu-vm-2:~/P1-ws-client$ python3 manage.py test votoAppWSClient.test_views --verbosity 2
Found 8 test(s).
Creating test database for alias 'default' ('test_voto')...
Operations to perform:

```


Cuestión número 9:

Ejercicio 9: incluya en la memoria de la practica pruebas de que se ha desplegado las aplicaciones web P1-ws-client y P1-ws-server y de que estas funcionan. Al menos se debe incluir una copia del contenido del #chero env así como una captura de pantalla en la que se muestre el resultado de interaccionar con cada uno de los #endpoints# del cliente. Las capturas deben mostrar el URL al que se conecta el Navegador.

Hemos realizado las pruebas justo como indican, con el uso de dos pc y las tres MV; Para probar esto, iniciamos la base de datos como siempre y el servidor, con el mismo env que antes, ya que se están ejecutando en el mismo ordenador; gunicorn también será iniciado:

```
# IMPORTANT: this file should not be in a repository
# To remove a file named env from a Git repository
# but keep it in the source (local system), follow these steps:
# Remove the file from Git tracking but keep it locally
## git rm --cached env
# Add 'env' to .gitignore (so it's not tracked again)
## echo "env" >> .gitignore
# Commit the changes
## git commit -m "Removed env from Git tracking and added to .gitignore"
# Push the changes to the remote repository
## git push
# use sqlite 3
#DATABASE_SERVER_URL=sqlite:///db.sqlite3
# use postgres
DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@172.27.239.131:15432/voto'
# The client does not need to store data in any database
# so let us define a sqlite in orden to avoid warning messages
DEBUG=True
SECRET_KEY = 'django-insecure-alczftn)j1#$v%xmk@5j(n*px43c8kxgi_ua4%khc+t7g_)s9d'
```

Posteriormente vemos el enviroment del cliente, en el que encontramos el url de la APIREST, ya que el cliente y el el servidor están ejecutando distintas Pcs, debemos indicar en el cliente la IP del ordenador donde se esta ejecutando el servidor.

```
P1 > P1-ws-client > env
1 # IMPORTANT: this file should not be in a repository
2 # To remove a file named env from a Git repository
3 # but keep it in the source (local system), follow these steps:
4 # Remove the file from Git tracking but keep it locally
5 ## git rm --cached env
6 # Add 'env' to .gitignore (so it's not tracked again)
7 ## echo "env" >> .gitignore
8 # Commit the changes
9 ## git commit -m "Removed env from Git tracking and added to .gitignore"
10 # Push the changes to the remote repository
11 ## git push
12 # use sqlite 3
13 ##DATABASE_SERVER_URL=sqlite:///db.sqlite3
14 # use postgres
15 #DATABASE_SERVER_URL='postgres://neondb_owner:npg_6WJxtXRPzZ8D@ep-steep-pine-a8gb1b6g-pooler.eastus2.azure.neon.tech/neondb?sslmode=require'
16 DATABASE_SERVER_URL='postgres://alumnodb:alumnodb@172.27.239.131:15432/voto'
17 RESTAPIBASEURL='http://172.27.239.131:28000/restapiserver/'
18 # The client does not need to store data in any database
19 # so let us define a sqlite in order to avoid warning messages
20 DEBUG=True
21 SECRET_KEY = 'django-insecure-alczftnjl#sV$xmK@5j(n*px43c8kxgi_ua4%khc+t7g_)s9d'
22
```

Tras tener el enviroment podemos comprobar que gunicorn está corriendo en nuestra MV3

```
(venv) si2@si2-ubuntu-vm-3:~$ sudo systemctl status gunicorn
● gunicorn.service - Gunicorn WSGI Application Server
   Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-02-23 19:25:58 CET; 4s ago
     Main PID: 7791 (gunicorn)
        Tasks: 2 (limit: 1672)
      Memory: 47.5M (peak: 47.7M)
         CPU: 628ms
    CGroup: /system.slice/gunicorn.service
            └─7791 /home/si2/venv/bin/python3 /home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application
              7795 /home/si2/venv/bin/python3 /home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application

feb 23 19:25:58 si2-ubuntu-vm-3 systemd[1]: Started gunicorn.service - Gunicorn WSGI Application Server.
feb 23 19:25:58 si2-ubuntu-vm-3 gunicorn[7791]: [2025-02-23 19:25:58 +0100] [7791] [INFO] Starting gunicorn 20.1.0
feb 23 19:25:58 si2-ubuntu-vm-3 gunicorn[7791]: [2025-02-23 19:25:58 +0100] [7791] [INFO] Listening at: http://0.0.0.0:8000 (7791)
feb 23 19:25:58 si2-ubuntu-vm-3 gunicorn[7791]: [2025-02-23 19:25:58 +0100] [7791] [INFO] Using worker: sync
feb 23 19:25:58 si2-ubuntu-vm-3 gunicorn[7795]: [2025-02-23 19:25:58 +0100] [7795] [INFO] Booting worker with pid: 7795
(venv) si2@si2-ubuntu-vm-3:~$
```

Ya tenemos todo preparado y solo queda abrir el navegador en el pc del cliente con la ruta localhost:38000:

←

→

↻

127.0.0.1:38000/votoAppWSClient/

Introduzca la Informacion Censo de la Persona que Vota (votoSite)

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Procedemos a probar la aplicación rellenando los campos, en este caso primero vamos a probar datos aleatorios:

← → ↻ 127.0.0.1:38000/votoAppWSClient/

Introduzca la Informacion Censo de la Persona que Vota (votoSite)

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

La respuesta deberá ser de error:

← → ↻ 127.0.0.1:38000/votoAppWSClient/

¡Error: Votante no registrado en el Censo!

Ahora si procedemos a introducir alguien registrado en el censo:

Aportar Información Censo (x) +

← → ↻ 127.0.0.1:38000/votoAppWSClient/

Introduzca la Informacion Censo de la Persona que Vota (votoSite)

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Tras enviar la información, vemos como si que estaba registrado y la aplicación nos ha respondido

Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

ID Proceso Electoral:

ID Circunscripcion:

ID Mesa Electoral:

Nombre Candidato Votado:

Tras rellenar los campos, el voto queda registrado:

Voto Registrado con Éxito (votoSite)

Id: 31

Codigo Respuesta: 000

Marca Tiempo :

Id Circunscripcion : 2

Id Mesa Electoral : 2

Id Proceso Electoral : 2

Nombre Candidato Votado: 2

Screenshot captured
You can paste the image from the clipboard.

Vamos a comprobar si realmente se ha quedado registrado:

Test Base de Datos: Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

ID Proceso Electoral:

ID Circunscripcion:

ID Mesa Electoral:

Nombre Candidato Votado:

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Test Base de Datos: Borrado de Voto

Introduzca el ID del voto a Borrar:

ID del Voto:

Test Base de Datos: Listado de Votos

Introduzca el ID del proceso electoral:

ID del Proceso Electoral:

Podemos observar que sí ha quedado registrado y se muestra la información:



localhost:38000/votoAppWSClient/testbd/getvotos/

Votos Registrados (votoSite)

id	IdCircunscripcion	IdMesaElectoral	Candidato Votado	Marca Tiempo	Codigo Respuesta
----	-------------------	-----------------	------------------	--------------	------------------

Ahora vamos a probar y borrarlo:

Test Base de Datos: Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

ID Proceso Electoral:

ID Circunscripcion:

ID Mesa Electoral:

Nombre Candidato Votado:

Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Test Base de Datos: Borrado de Voto

Introduzca el ID del voto a Borrar:

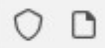
ID del Voto:

Test Base de Datos: Listado de Votos

Introduzca el ID del proceso electoral:

ID del Proceso Electoral:

Lo borramos y da éxito, lo que demuestra que si se había registrado y ya no está:



localhost:38000/votoAppWSClient/testbd/delvoto/

¡Voto eliminado correctamente!

Las demás pruebas se realizan en el test por lo que no es necesario probar más funcionalidades.