UNIVERSIDAD AUTONOMA DE MADRID		**Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2**			
Grupo	**2311**	**Práctica**	3	**Fecha**	04/05/2025
Alumno/a		Gómez, Hernández, Marco			
Alumno/a		Haya, de la Vega, Juan			

Práctica 3: Uso de apache como balanceador de carga de la aplicación web Django replicada

Ejercicio número 1:

Incluya en la memoria de la práctica pruebas de que se ha configurado y ejecutado el balanceador de carga de apache. Se debe incluir capturas de la ejecucion sudo systemctl status apache2 así como sudo apachectl congfigtest junto capturas de la interfaz de administración.

Configuramos el fichero /etc/apache2/sites-available/000-default.conf (la ip de los BalancerMember es la de hacer ifconfig en el host pública, y la ServerName, la de hacer ifconfig en la VM1 pública):

```
  GNU nano 7.2                    /etc/apache2/sites-available/000-default.conf
ProxyRequests Off

<Proxy balancer://SI2Cluster>
  # Miembros del cluster (instancias de la app Django)
  BalancerMember http://192.168.1.140:18000 route=Instance01
  BalancerMember http://192.168.1.140:28000 route=Instance02
  BalancerMember http://192.168.1.140:38000 route=Instance03

  # Habilitar sticky session con la cookie ROUTEID
  ProxySet stickysession=ROUTEID
</Proxy>

Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED

# Configurar la ubicacion para balanceo de carga
<Location /P1base>
  Require all granted
  ProxyPass balancer://SI2Cluster
  ProxyPassReverse balancer://SI2Cluster
</Location>

# Configuracion del administrador del balanceador de carga
<Location /balancer-manager>
  SetHandler balancer-manager
</Location>

# Configuracion para manejar las redirecciones de Django correctamente
RewriteEngine On

# Redirecciones con /P1base se manejen correctamente
RewriteCond %{REQUEST_URI} !^/P1base
RewriteCond %{REQUEST_URI} !^/balancer-manager
RewriteRule ^(.*)$ /P1base$1 [R,L]

# Ajustar encabezado Location para evitar afectaciones
```

```
Header edit Location ^/(?!balancer-manager)([^/]) /P1base/$1

# Definir el nombre del servidor (ajústalo según tu configuración)
ServerName 10.0.2.15
```

Ahora, tras hacer un sudo systemctl restart apache2, ejecutamos los comandos solicitados y vemos que la

sintaxis del fichero de configuración es correcta y que apache está activo y funcionado:

```
si2@si2-ubuntu-vm-3:~$ sudo apachectl configtest
Syntax OK
si2@si2-ubuntu-vm-3:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-04-13 19:52:16 CEST; 4min 2s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 4260 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 4264 (apache2)
      Tasks: 55 (limit: 1670)
     Memory: 5.6M (peak: 5.9M)
        CPU: 71ms
     CGroup: /system.slice/apache2.service
             ├─4264 /usr/sbin/apache2 -k start
             ├─4265 /usr/sbin/apache2 -k start
             └─4266 /usr/sbin/apache2 -k start

Apr 13 19:52:16 si2-ubuntu-vm-3 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Apr 13 19:52:16 si2-ubuntu-vm-3 systemd[1]: Started apache2.service - The Apache HTTP Server.
```

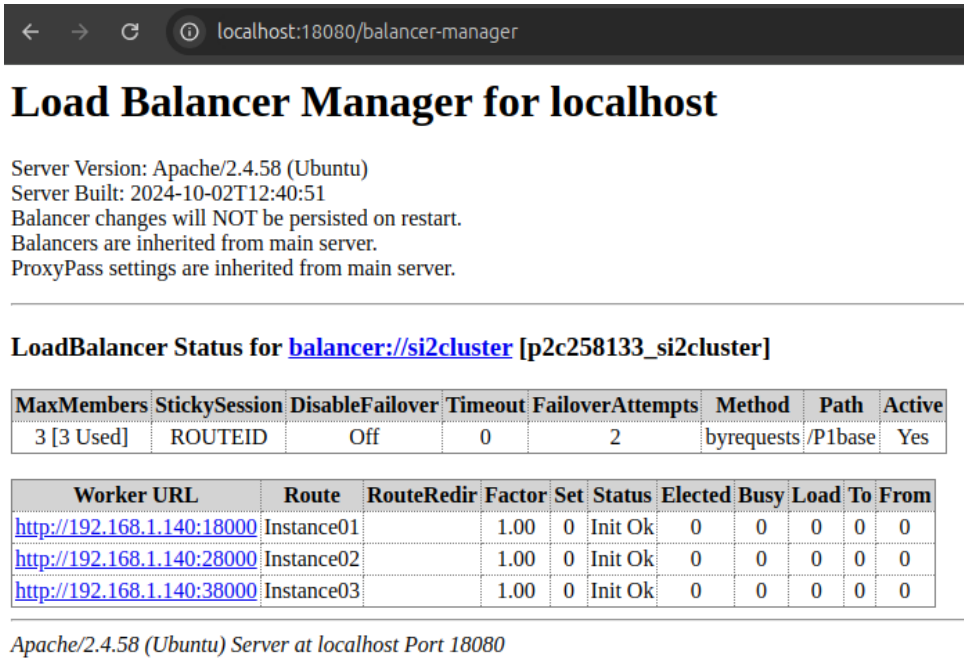
Como como extra, probamos a ejecutar journalctl -xeu apache2.service, y nos sale que no hay entradas (en los logs del servicio de apache):

```
~
~
-- No entries --
```

Además, también hemos ejecutado sudo tail -n 50 /var/log/apache2/error.log, en lo que vemos que apache tras el restart está funcionando correctamente (no hay errores en el log de errores):

```
[Sun Apr 13 19:52:16.886379 2025] [mpm_event:notice] [pid 2428:tid 125755108022144] AH00492: caught SIGWINCH, shut
ting down gracefully
[Sun Apr 13 19:52:16.967412 2025] [mpm_event:notice] [pid 4264:tid 130276891211648] AH00489: Apache/2.4.58 (Ubuntu
) configured -- resuming normal operations
[Sun Apr 13 19:52:16.967546 2025] [core:notice] [pid 4264:tid 130276891211648] AH00094: Command line: '/usr/sbin/a
pache2'
```

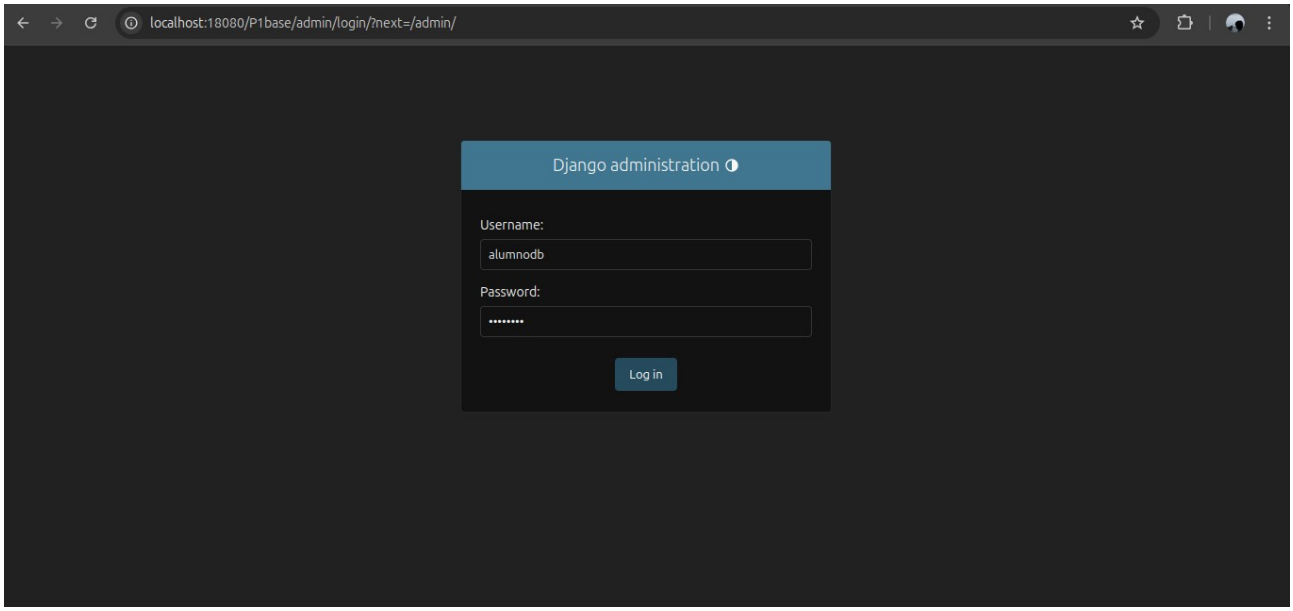
Si probamos a acceder a la consola del balanceador, vemos que todo sale como debería al estar apache funcionando:

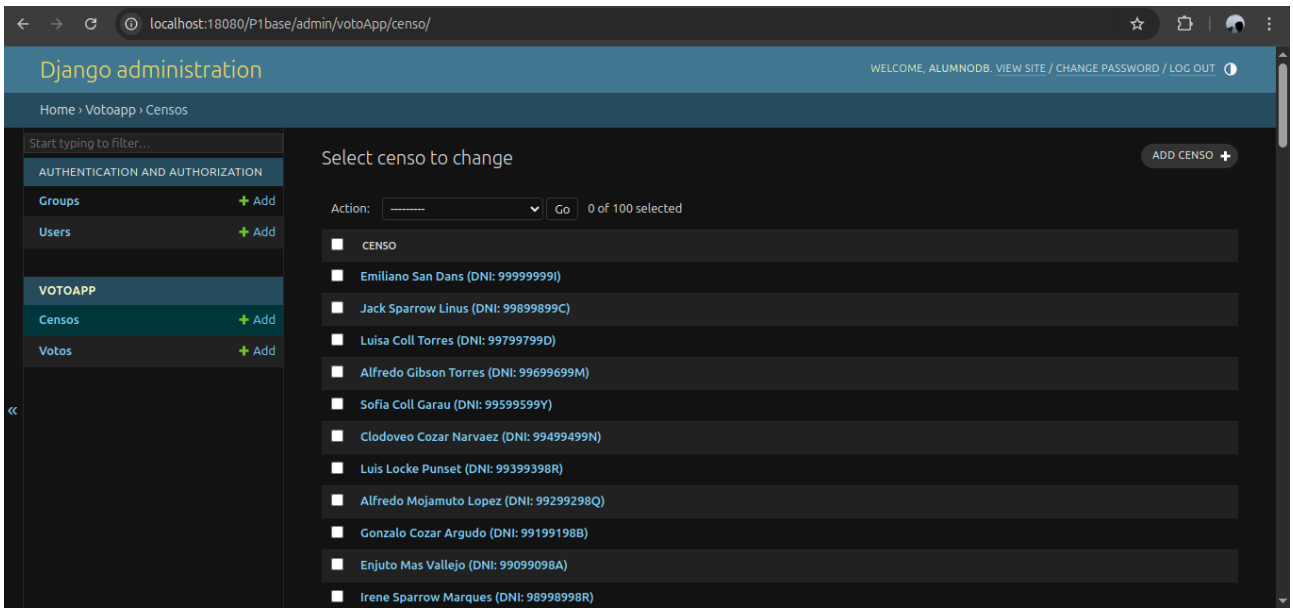


Para acceder a la interfaz de administración, cremos un superusuario:

```
si2@1:~/repo/p1base/p1/P1-base$ python manage.py createsuperuser
Username (leave blank to use 'si2'): alumnodb
Email address:
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
si2@1:~/repo/p1base/p1/P1-base$
```

Accedemos ahora a la interfaz de administración de django para comprobar que todo está funcionando correctamente (está además la base de datos poblada correctamente):



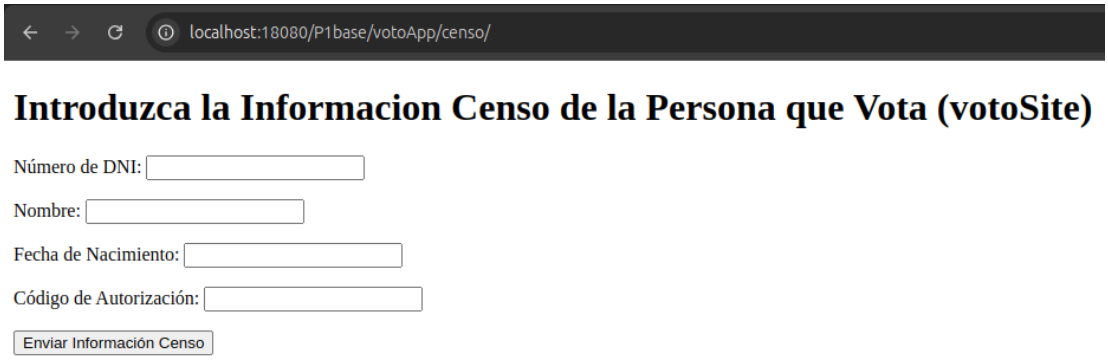


Para que nos dejase hacer el login correctamente, hemos tenido que iniciar sesión en django directamente desde el puerto 18000 (sin pasar por el balanceador) y luego con las cookies sessionid, csrftoken y ROUTEID que se nos generan, hacer lo que sale en la última captura anterior (enviarla al acceder a ese endpoint a través del balanceador de apache) (ya en el puerto 18080, pasando por el balanceador). No nos dejaba iniciar sesión en la interfaz de admin de django directamente desde apache probablemente porque como se añade P1base al path, no se nos retornaba la cookie sessionid (que tendría un path que retorna la cookie sólo a paths sin P1base o algo por el estilo) tras introducir los datos de inicio de sesión correctamente.

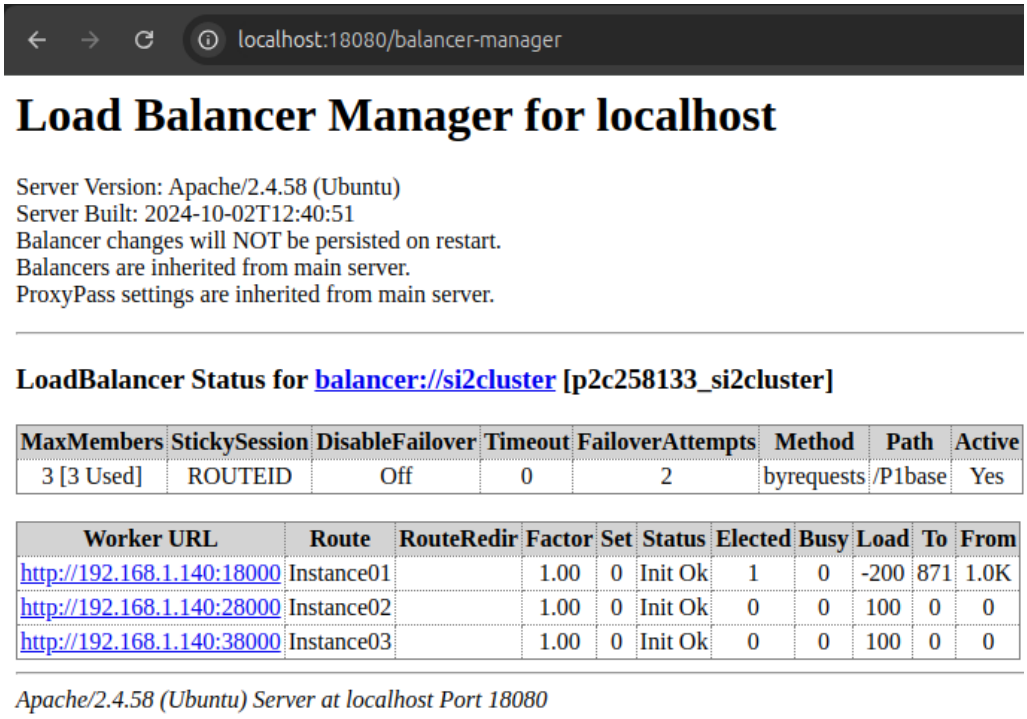
Ejercicio número 2:

Con únicamente la instancia de la MV1 en ejecución, acceda por el balanceador a la aplicación, describa los valores que se muestran por la consola del balanceador y el redireccionamiento adjuntando una captura del balanceador.

Apagamos las otras dos VM (2 y 3) para que las instancias de la aplicación en ellas no estén en ejecución, dejando sólo la VM1; y accedemos a la aplicación desde el balanceador:



Y si probamos ahora a acceder a la consola del balanceador, se nos muestra la siguiente pantalla:



Vamos a explicar las distintas columnas de la tabla de arriba y sus valores (aqunque no nos es útil para ver que la solicitud hecha se redirige a la instancia 1, pero lo ponemos como explicación extra de la captura):

MaxMembers indica el número máximo de instancias (nodos o workers) que pueden estar activamente asociadas al grupo de balanceo que gestiona este contexto. El valor “3 [3 Used]” significa que hay tres miembros asignados y que actualmente los tres están en uso para servir peticiones, lo que sugiere que todos los nodos disponibles están operativos y participando activamente en el balanceo.

StickySession define si se está utilizando el mecanismo de sesión persistente, también conocido como sesión “pegajosa”. En este caso, el valor “ROUTEID” (cookie) indica que se están utilizando identificadores de ruta específicos asociados a las sesiones, de modo que una vez que un cliente ha sido asignado a un nodo, las siguientes peticiones se redirigen al mismo nodo, manteniendo la sesión de usuario estable.

DisableFailover hace que apache no intente redirigir solicitudes fallidas a otros servidores si está en On (aunque aquí está en Off). Se usa en situaciones donde repetir solicitudes puede causar problemas, como en transacciones críticas. Es una medida para evitar duplicaciones o efectos no deseados.

Timeout tiene un valor de 0, lo que en este contexto puede significar que no se ha configurado un tiempo máximo de espera para el failover, o que se permite indefinidamente esperar respuesta del nodo antes de considerar redirigir la petición a otro nodo (considrerar que ha fallado). Sin embargo, este valor depende de la configuración más general del balanceador.

FailoverAttempts define cuántas veces intentará apache reenviar una solicitud fallida a otros servidores del balanceador (en este caso está en 2). Es útil cuando un servidor no responde y se desea que la solicitud se redirija automáticamente a otro. Solo aplica si la solicitud no ha sido completamente enviada.

Method se refiere al criterio utilizado para distribuir las peticiones entre los nodos. “byrequests” indica que la lógica de balanceo se basa en el número de peticiones: el sistema intenta repartir de manera uniforme las solicitudes entre los miembros disponibles, equilibrando la carga de trabajo.

Path muestra el contexto o ruta del servidor que se está balanceando. En este caso, “/P1base” es la ruta a la raíz de nuestro proyecto Django.

Finalmente, **Active** con el valor “Yes” señala que este grupo de balanceo y su configuración están activos y siendo utilizados por Apache en este momento.

Vamos a explicar ahora las distintas columnas de la tabla de abajo, y el sentido de los valores que toman para cada instancia (las que cambian entre la instancia activa y las otras son Elected, Load, To y From; que es lo que nos importa para este ejercicio sobre todo; lo otro es información y explicación extra de la captura):

La columna **"Worker URL"** indica la dirección del servidor o instancia a la que el balanceador redirigirá el tráfico. En este caso, hay tres instancias configuradas, que son las dlas tres VM, por eso tienen la misma ip (la del host) pero distintos puertos (el de gunicorn de cada VM visto desde el host).

"Route" representa el identificador lógico o nombre de esa instancia dentro del clúster, usado para mantener sesiones persistentes (conocido como *sticky sessions*). Aquí tenemos Instance01, Instance02 e Instance03, lo cual facilita la identificación de cada instancia.

La columna **"RouteRedir"** se usaría si existiera una redirección de sesión en caso de que una instancia no estuviera disponible, pero como está vacía, significa que no hay rutas alternativas definidas para redirección de sesiones en caso de fallo.

"Factor" es el peso de la instancia en el balanceo de carga. Un valor de 1.00 en todas las instancias indica que cada una tiene el mismo peso o prioridad en la distribución de tráfico. Si un valor fuera mayor, esa instancia recibiría más peticiones proporcionalmente.

"Set" indica si el worker pertenece a un subconjunto de balanceo. Aquí todos están en el set 0, lo que sugiere que no hay agrupaciones diferenciadas (como por zona geográfica o tipo de servicio).

La columna **"Status"** muestra el estado actual del worker. "Init Ok" indica que el balanceador considera disponible ese servidor para recibir tráfico (pese a que no estén activos, como es el caso de las instancias 2 y 3, que también tienen Init Ok).

"Elected" refleja cuántas veces esa instancia ha sido seleccionada para manejar una petición desde que se inició el balanceador. Por ejemplo, Instance01 ha sido elegida una vez (para la solicitud recién realizada), mientras que las otras aún no han manejado tráfico.

"Busy" representa el número de conexiones activas actualmente en curso hacia ese worker. Aquí todas están en cero, lo que indica que en el momento de la consulta no había solicitudes en proceso (no está la de la instancia 1 en 1 porque al recargar la consola del balanceador la petición a la app ya se había hecho y la conexión se había cerrado).

"Load" es una métrica interna del balanceador que indica la carga relativa actual. El valor puede ser positivo o negativo, y es usado para determinar cuál worker debería ser elegido a continuación. Se escoge la instancia con más Load en las siguientes peticiones si es posible (para distribuir la carga lo más equitativamente posible). En este caso, Instance01 tiene un valor de -200, y las otras dos tienen 100. Esto hace que en las siguientes dos peticiones se vaya a escoger antes a cualquiera de las instancias 2 o 3 antes que a la 1 si es posible, y se ha asignado para compensar a la instancia 1 por la carga que acaba de asumir en la petición que hemos hecho.

Finalmente, **"To"** y **"From"** se refieren al número de bytes transferidos hacia y desde cada instancia. Por ejemplo, Instance01 ha enviado 871 bytes al cliente y ha recibido aproximadamente 1.0K bytes del balanceador, lo que también concuerda con que fue la única que manejó una petición ya que las otras instancias tienen 0 en ambos campos.

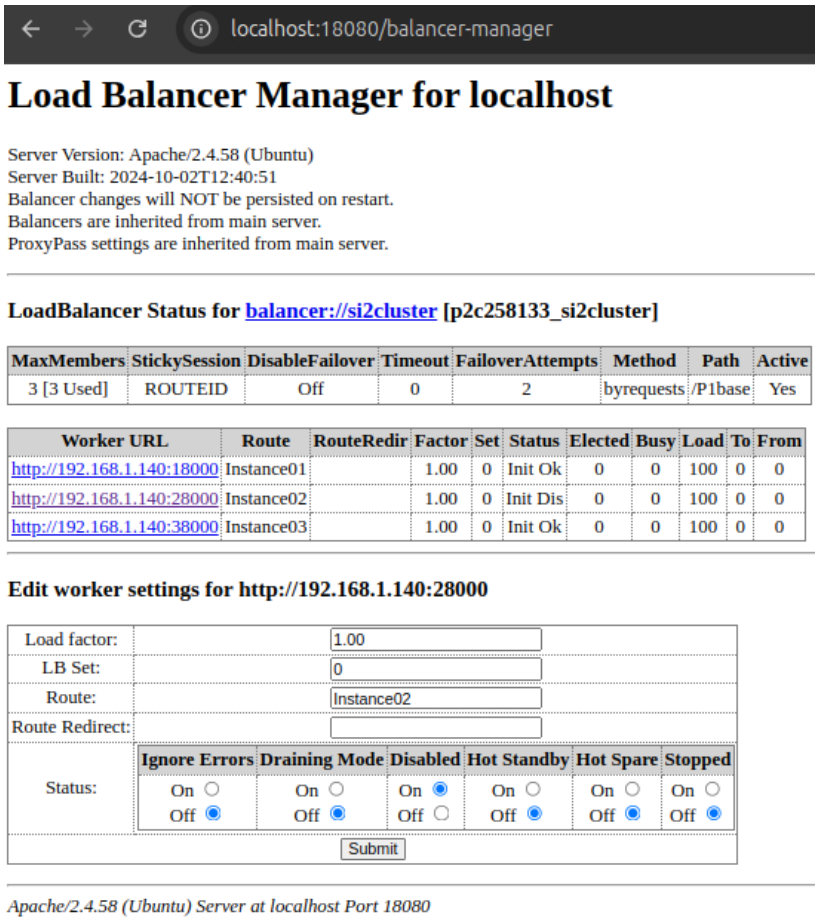
En la consola del balanceador se muestra también la versión del server, cuándo fue construido… (ver imágen), pero esto no es información que nos sea demasiado útil en este apartado.

Ejercicio número 3:

Pruebe a registrar un voto individualmente en cada instancia. Para ello, desde la consola de balanceo, fuerce el uso de una única instancia, compruebe que los votos creados con una instancia, son accesibles desde el resto de instancias.

Se debe destacar, que en los ficheros /etc/hostname de cada VM, hemos puesto 1, 2 y 3 (número de la VM), para poder distinguir desde qué instancia se guardó cada voto (atributo nuevo instancia, de voto).

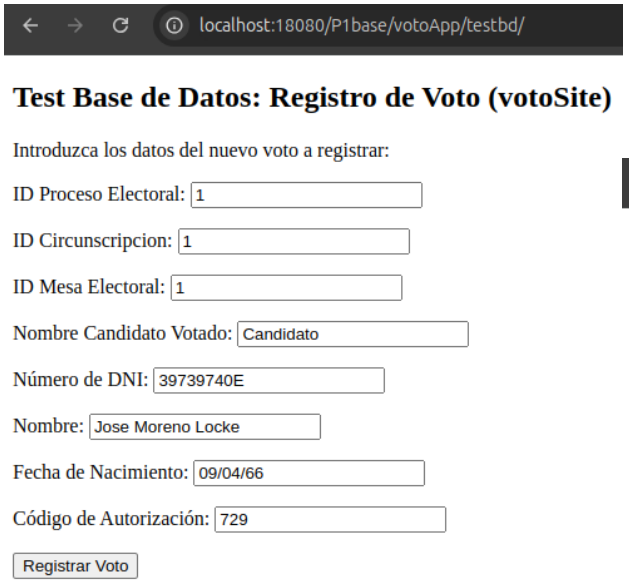
Primero, se destaca que para deshabilitar las dos instancias que no son en las que vamos a registrar el voto, hacemos algo equivalente a esto pero para la instancia que corresponda (se selecciona haciendo click en cada Worker URL) (no lo ponemos todo el rato para que no haya demasiadas capturas):

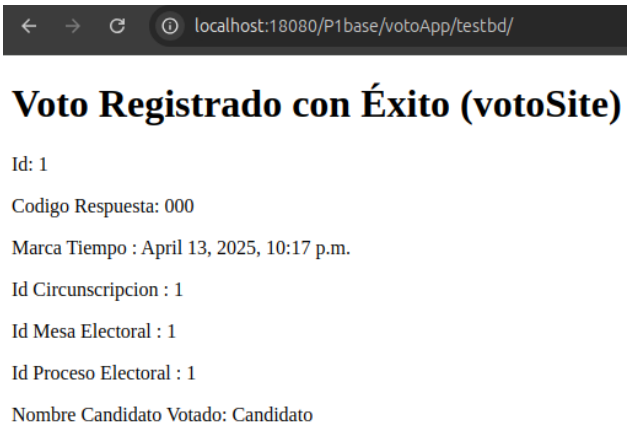


Antes de comenzar, buscaremos tres censos válidos (uno para el voto de cada instancia) con DBeaver:

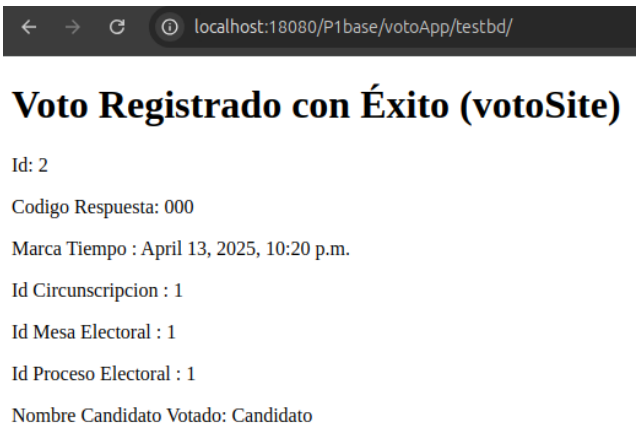
numeroDNI	nombre	fechaNacimiento	anioCenso	codigoAutorizacion
39739740E	Jose Moreno Locke	09/04/66	2025	729
83583583L	Restituta Sparrow Martinez	31/08/73	2025	535
67867868T	Randall Martinez Mojamuto	04/06/96	2025	056

Primero, registraremos un voto desde la instancia 1, con lo que tendremos que desactivar las otras dos instancias como se ha mencionado:

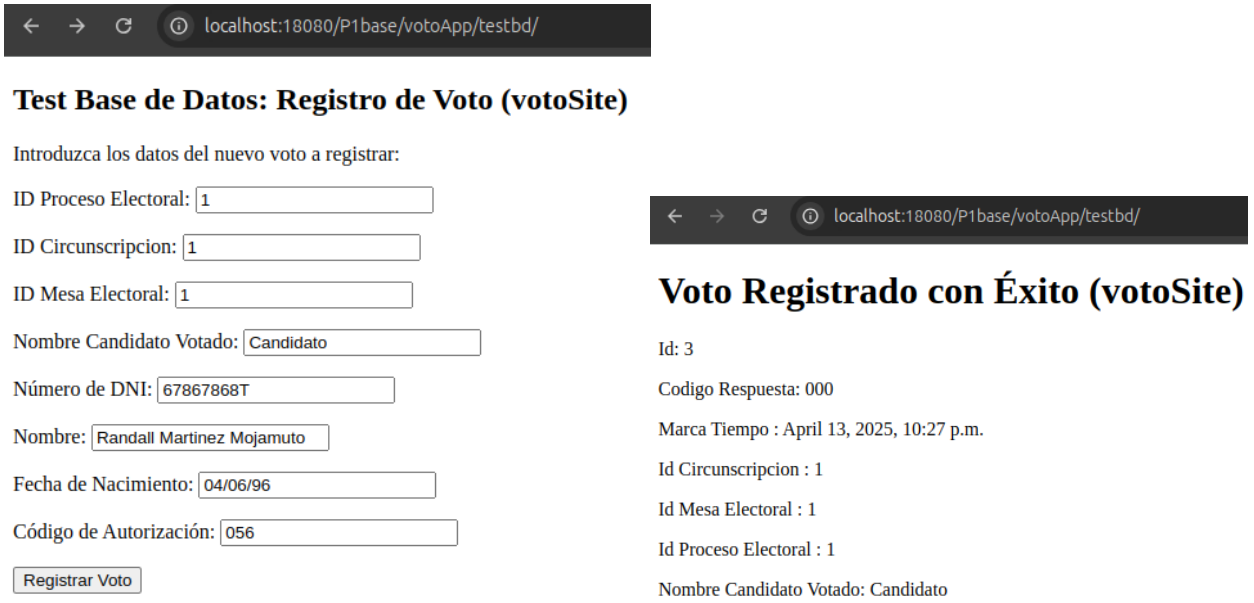




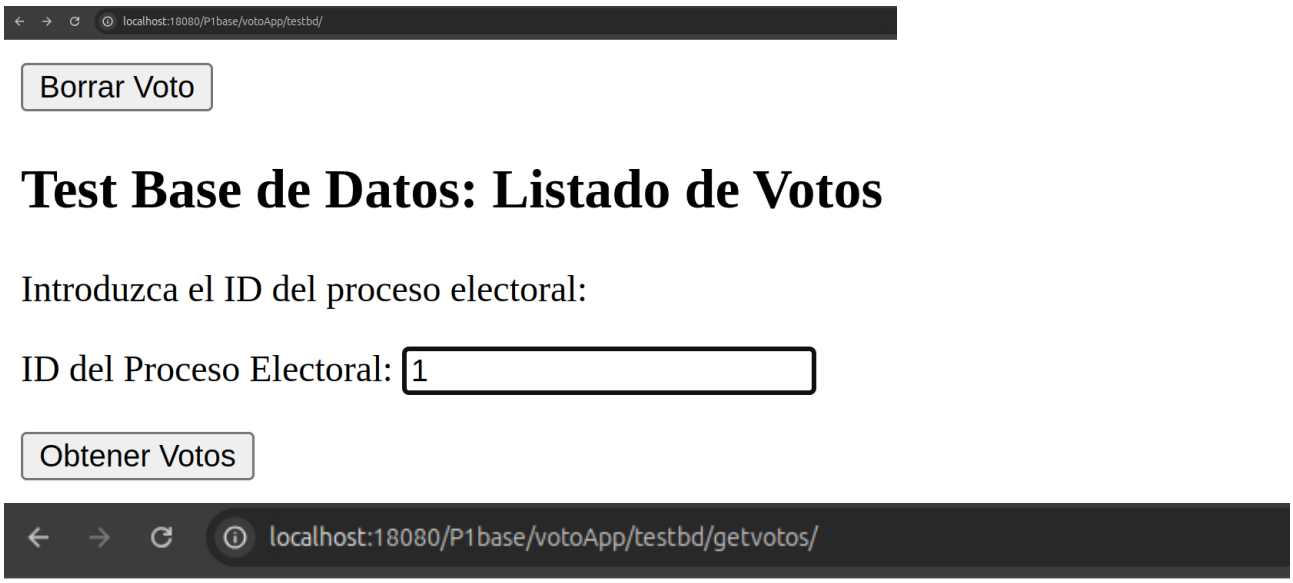
Segundo, registraremos un voto desde la instancia 2, con lo que tendremos que desactivar las otras dos instancias como se ha mencionado:



Tercero, registraremos un voto desde la instancia 3, con lo que tendremos que desactivar las otras dos instancias como se ha mencionado:



Ahora, probaremos desde cada una de las tres instancias a obtener los votos del proceso electoral 1 (el de todos los votos registrados) (para hacer esto de con una instancia específica hacemos lo mismo que antes: desactivar como se mencionó las otras dos instancias y activar esa). Nos sale que sea la instancia que sea desde la que listemos los votos, salen los tres registrados (cada uno en una instancia), lo que muestra que los votos registrados desde una instancia son accesibles desde las otras (en los tres casos se muestra lo mismo):



id	IdCircunscripcion	IdMesaElectoral	Candidato Votado	Marca Tiempo	Codigo Respuesta	Instancia
1	1	1	Candidato	April 13, 2025, 10:17 p.m.	000	1
2	1	1	Candidato	April 13, 2025, 10:20 p.m.	000	2
3	1	1	Candidato	April 13, 2025, 10:27 p.m.	000	3

Ejercicio número 4:

Suprimiendo la afinidad de sesión, (eliminando las conguraciones de sticky session) accede a la aplicación a través de la URL del balanceador. Complete el voto con datos de censo correcto, repita los votos hasta que uno falle por la afinidad de sesión.

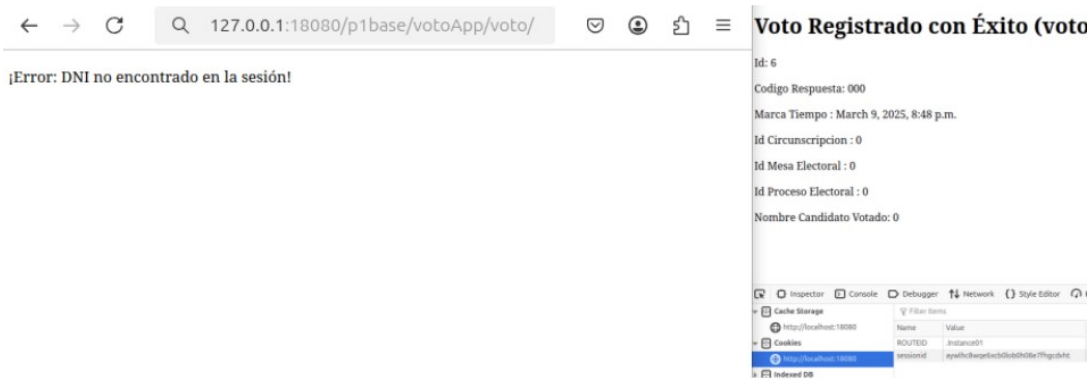


Figure 7: Error sesion

A continuación, verica que el balanceador de carga esté utilizando ProxySet stickysession=ROUTEID para mantener la afinidad de sesión. La cookie ROUTEID se generará automáticamente, y se usará para asegurar que las solicitudes de este cliente se dirijan siempre a la misma instancia de servidor, de acuerdo con la conguración del balanceador.

Explique en qué y cómo afecta el uso de ProxySet stickysession con respecto a errores.

Para eliminar las configuraciones de sticky session, comentamos las líneas de ProxySet y Header add (las que tienen que ver con la sticky session) del fichero /etc/apache2/sites-available/000-default.conf:

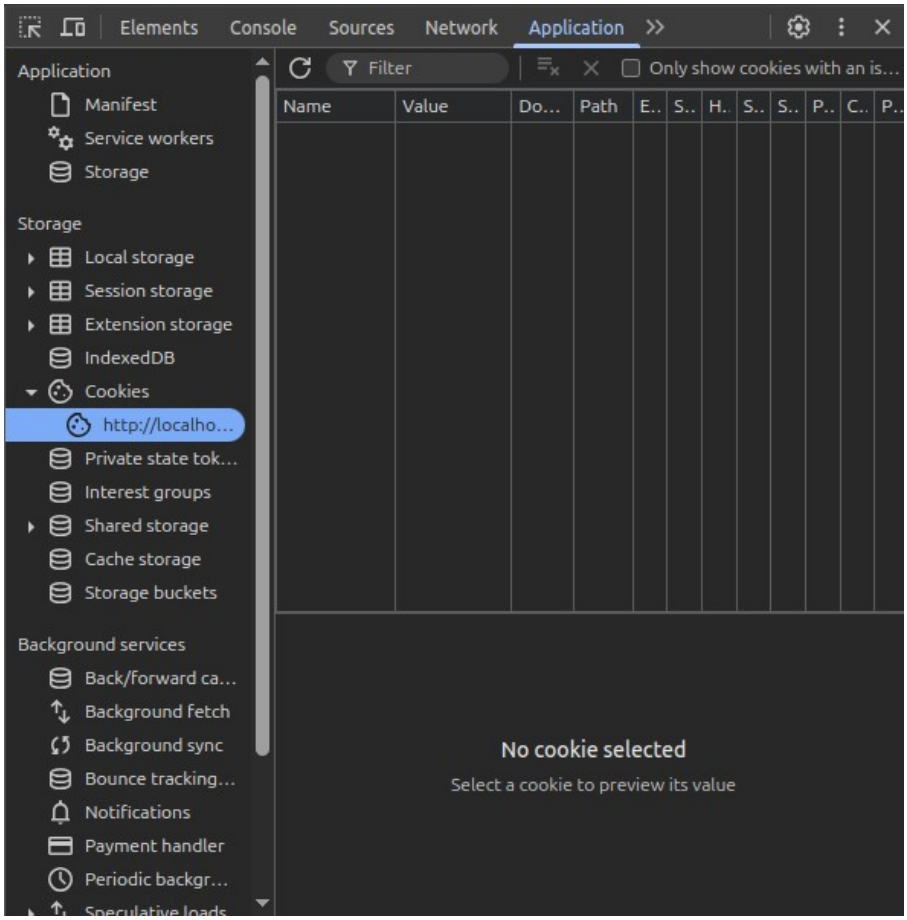
```
  # Habilitar sticky session con la cookie ROUTEID
  # ProxySet stickysession=ROUTEID
</Proxy>

# Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
```

Ahora, para asegurarnos de que se ha aplicado, hacemos un sudo systemctl restart apache2 para reiniciar apache. Si accedemos ahora a la consola del balanceador, en la columna StickySession, donde antes salía ROUTEID, ahora pone (None):

StickySession
(None)

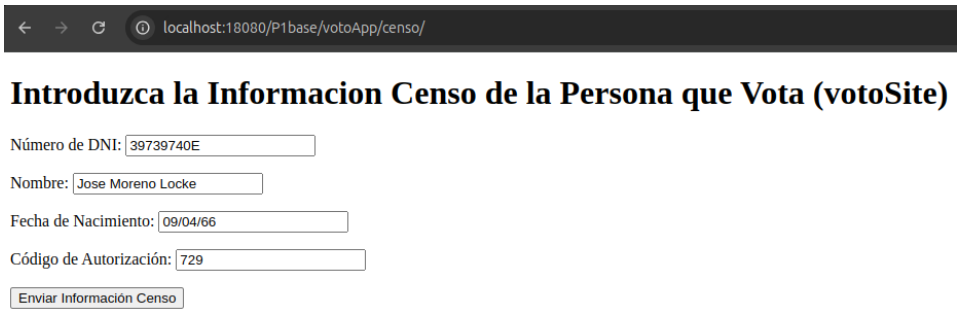
Ahora, comprobamos en el navegador que no hay ninguna cookie guardada y vemos que así es:



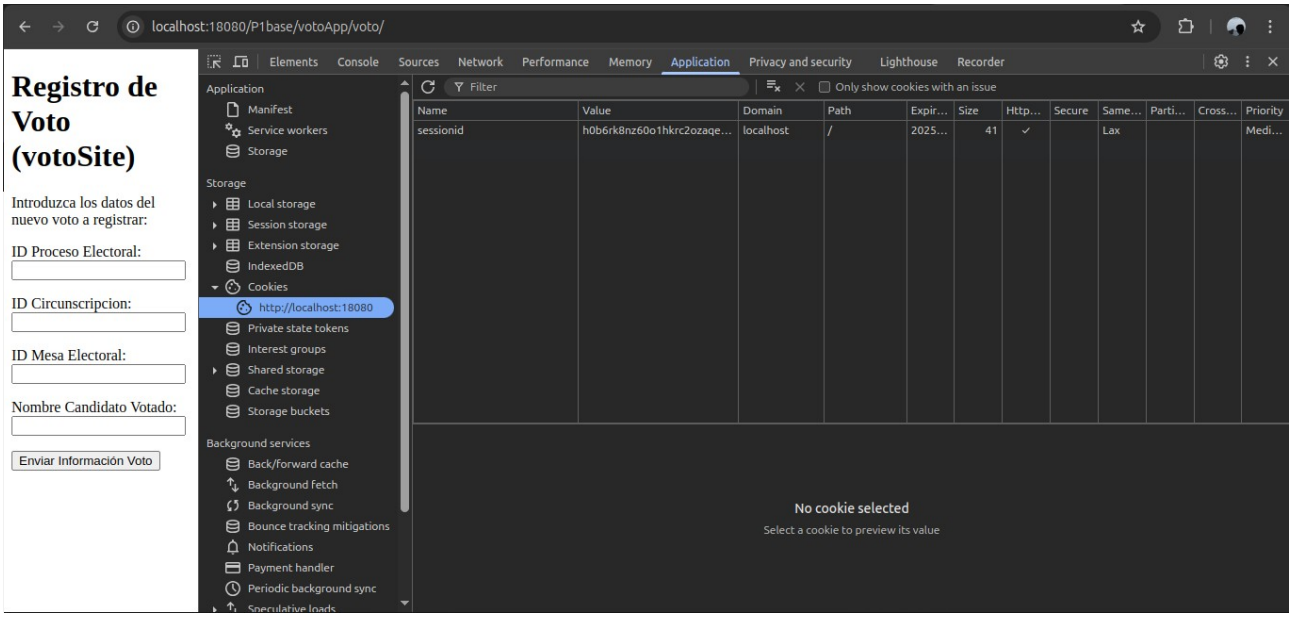
Además, no hace falta mencionar que hemos vuelto a habilitar todas las instancias (con respecto al ejercicio anterior), y que hemos borrado los votos registrados en anteriores apartados.

Procederemos ahora a intentar registrar votos verificando el censo desde el endpoint de censo y registrando el voto desde el endpoint de voto (no testbd), ya que estos son los endpoints desde los que se maneja el id del censo verificado como un dato de sesión para luego poder registrar el voto (debiendo fallar si verificamos el censo en una instancia y luego en otra intentamos registrar el voto, porque el id del censo verificado está en la caché de la primera instancia, a la que la segunda no tiene acceso). Utilizamos los censos válidos que vimos en DBeaver en el anterior ejercicio.

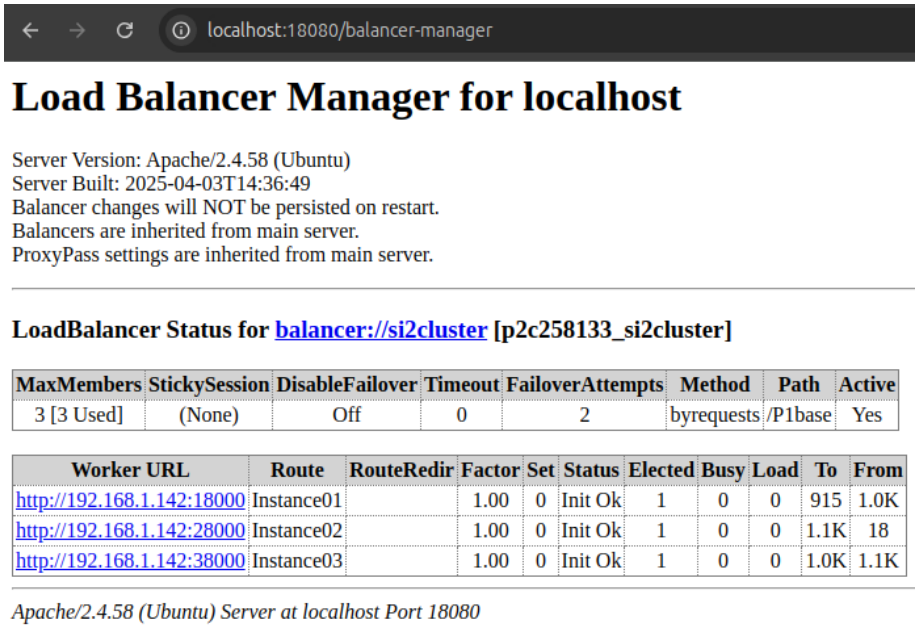
Primero, sin tener cookies, verificamos el censo:



Y vemos que se genera una cookie con el id de la sesión (para identificar los datos de la sesión, que es el id del censo verificado, gurdados en la caché cierta instancia):

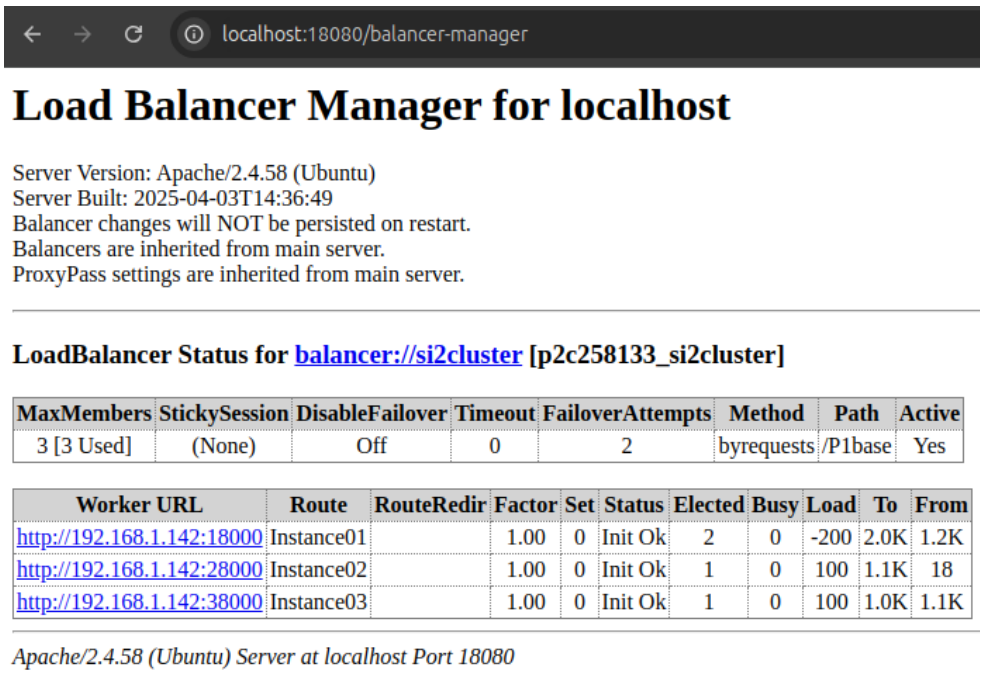


Se puede ver aquí que las peticiones hechas las han atendido las instancias 1, 2 y 3 (en orden, por lo de Load y que ante igualdad de Load para el balanceo de carga explicado antes y que también se explicará en el futuro se coge la instancia con un menor índice). Debido a esto, sabemos que es la instancia 2 la que tiene en su caché el id del censo verificado como datos de sesión (porque la segunda petición es el post de censo, donde se generan los datos de sesión, y es la que se ha hecho a la instancia 2):



Como se ha redireccionado al endpoint voto, sabemos que el censo se ha verificado correctamente y que el id de este está gurdado en la caché de una de las instancias (la 2), para que ahora al registrar el voto, se pueda poner el id (DNI) del censo que lo registra:

Podemos ver que aún habiendo sido el censo verificado correctamente, no se encuantra el id del censo (DNI) en la sesión. Esto es, como hemos explicado, porque ha sido otra instancia distinta la que se ha usado para verificar el censo (la 2, que en su caché tiene los datos de sesión (id/DNI del censo) para poder resgistrar el voto) y la que ha tratado de registrar el voto (la 1 (ver siguiente captura en la que es la única en la que ha aumentado su Elected), en su caché no tiene ningún dato de sesión).



Ahora descomentaremos las configuraciones de sticky session comentadas antes del fichero /etc/apache2/sites-available/000-default.conf:

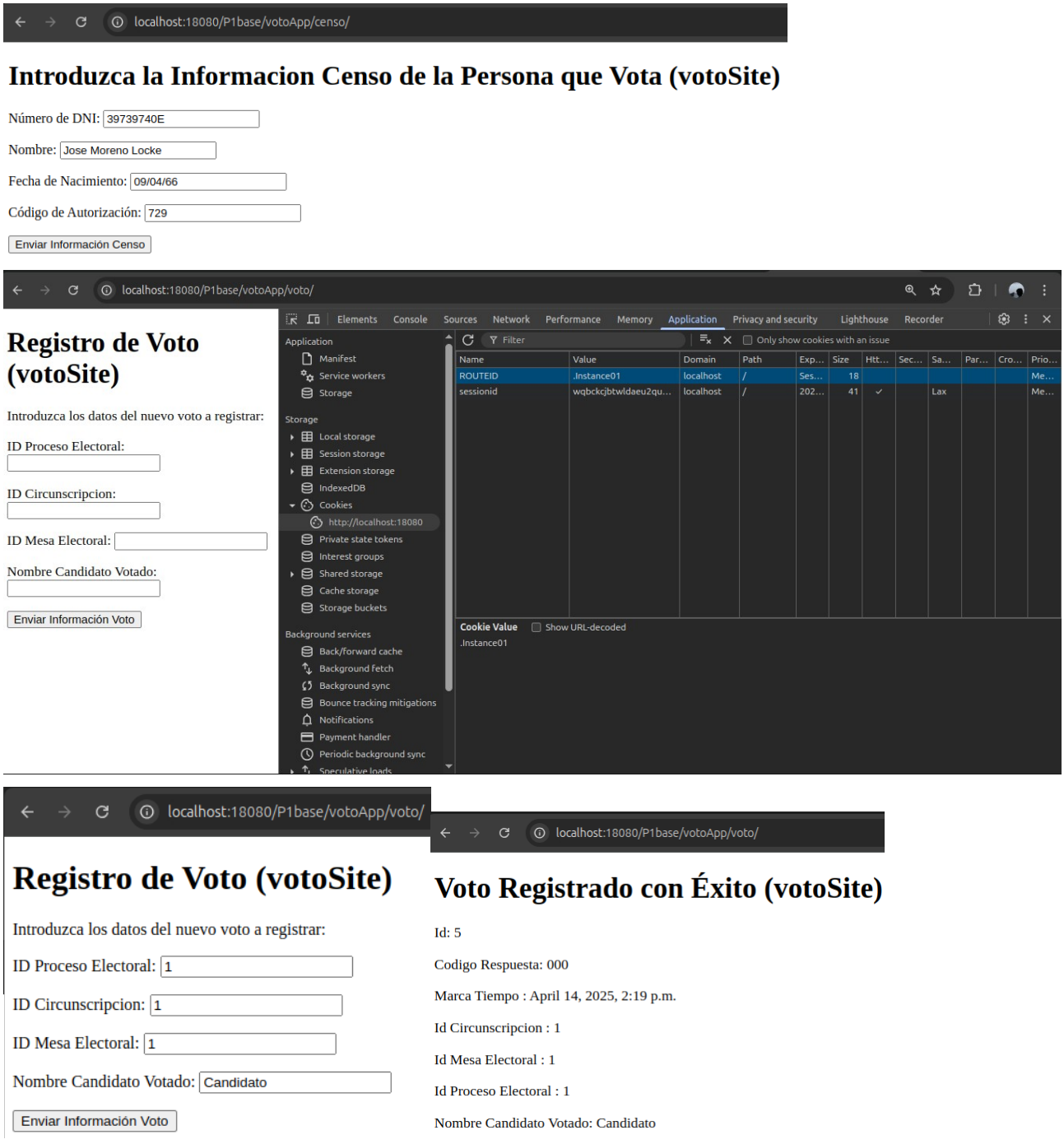
```
  # Habilitar sticky session con la cookie ROUTEID
  ProxySet stickysession=ROUTEID
</Proxy>

Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
```

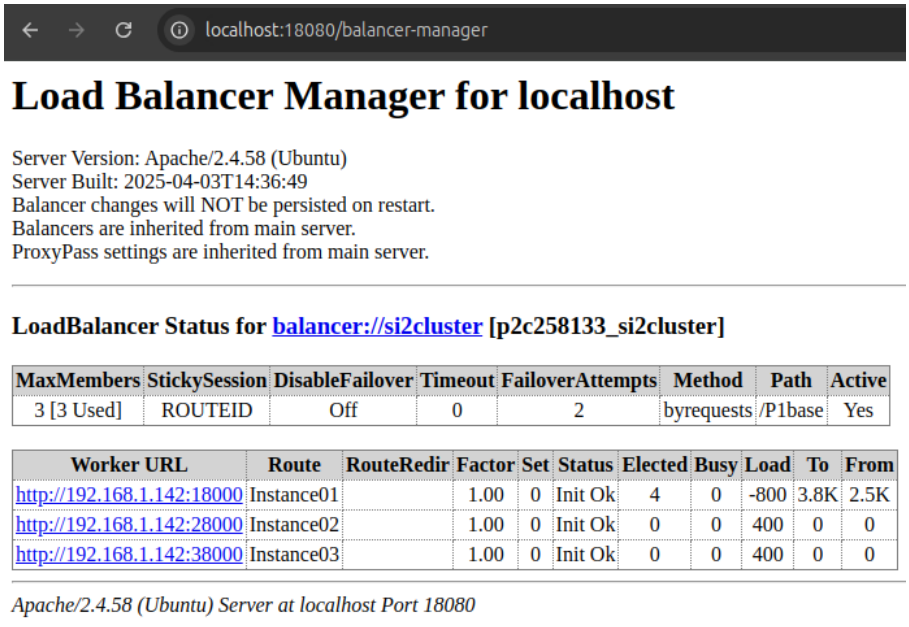
Hacemos otro sudo systemctl restart apache2 para reiniciar apache y comprobamos que en la consola del balanceador, en la columna StickySession, vuelve a poner ROUTEID:

StickySession
ROUTEID

Volvemos a eliminar las cookies igual que antes, e intentamos registrar el voto de la misma manera (mismos datos), pudiendo ver ahora que se genera una cookie ROUTEID, y que el voto se registra correctamente (habiendo elimiado el voto anterior antes). Esto ocurre por lo ya comentado: ahora la misma instancia que verifica el censo y que guarda en su caché el id de este, es la que registra el voto (la 1):



En la consola del balanceador podemos ver que todas las peticiones ahora se han hecho a la misma instancia (la 1) (sticky sessions):



En cuanto a la explicación de en qué y cómo afecta el uso de ProxySet stickysession con respecto a errores, ya se ha explicado a lo largo de este ejercicio, pero vamos a volver a comentarlo pero en más detalle: cuando verificamos un censo desde el endpoint censo (y se verifica correctamente), el servidor almacena en su caché (configurado que sea la caché en vez de la BD en el settings.py, como se mencionó en el enunciado) el id/DNI del censo verificado y asigna el id de esa sesión como cookie al cliente (para que la envíe en las próximas solicitudes). Tras esto, ya sea redireccionado automáticamente o accediendo manualmente al endpoint voto, al enviar el formulario del voto, el navegador enviará la cookie con el id de la sesión. En caso de que la instancia que tiene los datos de la sesión en su caché no sea la que se use para atender este envío del formulario del registro del voto con la cookie del id de la sesión (cosa que puede suceder si la sticky session no está configurada con la directiva mencionada), no se podrá registrar el voto al no tener el id/DNI del censo verificado previamente esta instancia (porque esta instancia no tiene en su caché los datos de la sesión de la otra instancia), haciendo que salga el fallo de que el DNI del censo verificado no está en la sesión, como se ha visto. Si activamos las sticky sessions mediante la directiva, aseguramos con una cookie (ROUTEID) que identifica la instancia que atendió a la primera petición de la sesión del cliente, que se usará esa misma instancia (que será la que en su caché tendrá todos los datos de esta sesión) para las siguientes solicitudes de la sesión, haciendo que no suceda el error comentado.

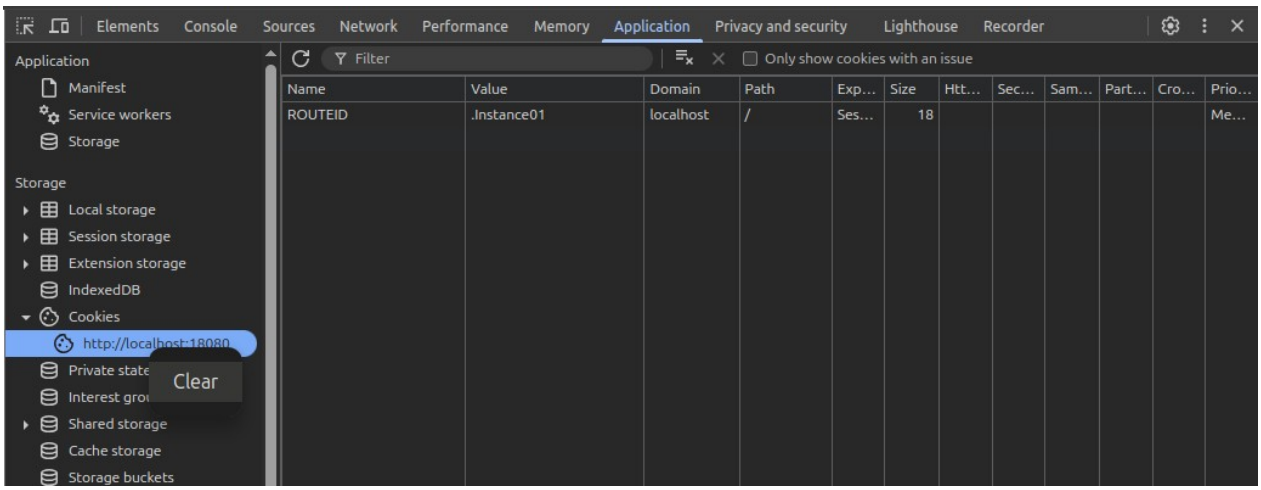
Ejercicio número 5:

Vericar que el balanceador de carga maneja el failover correctamente, redirigiendo el tráfico a una nueva instancia del servidor si una de las instancias cae.

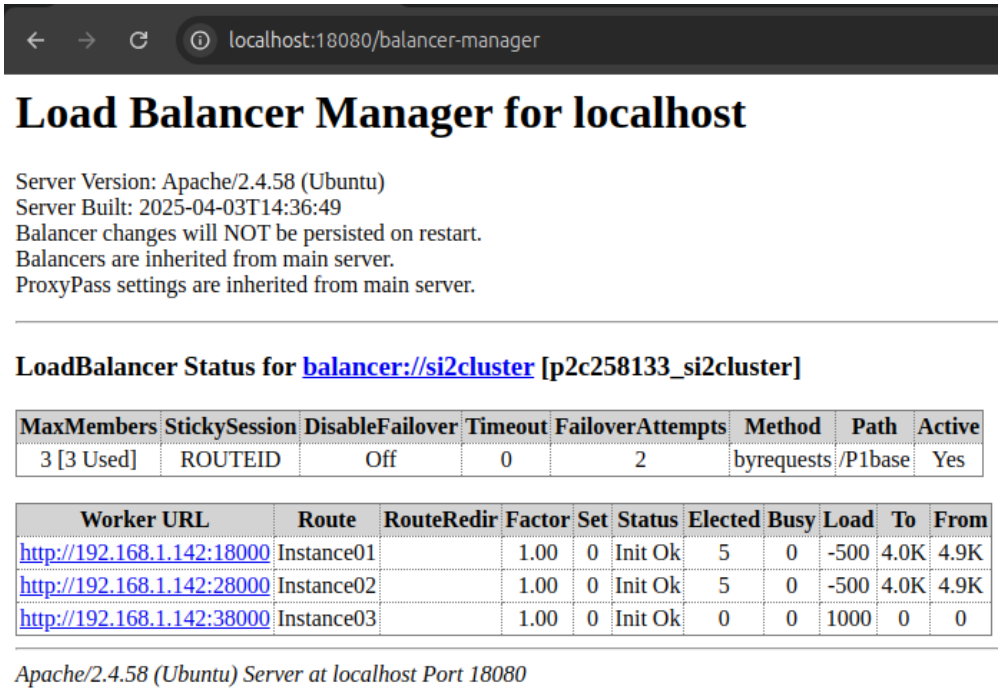
- **Identificar la instancia con menos elecciones: Utilizando el gestor de balanceo de carga (accediendo a la página /balancer-manager), observa el número de "elecciones" o "requests" para cada instancia. Esta métrica indicará cuántas solicitudes ha recibido cada servidor.**
- **Parar la instancia del clúster con menos elecciones; para ello bastará con parar el demonio de gunicorn.**
- **Realiza peticiones nuevamente accediendo a la URL del balanceador. Accede a la página /balancer-manager para vericar que la instancia detenida ha sido marcada como "errónea" y que el balanceador está redirigiendo todas las peticiones a la instancia restante.**
- **Demuestra borrando cookies y accediendo varias veces que el balanceador redirige a la instancia que no está caida.**

Se destaca que todo el rato se ha accedido (GET) al endpoint censo para realizar peticiones; y que, para no repetirnos y que no quede demasiado extenso el documento, no se mostrarán todo el rato capturas de este endpoint para demostrar que hacemos peticiones. Para reiniciar los datos de la consola del balanceador, hacemos un sudo systemctl restart apache2. Para forzar que haya una instancia con un Elected menor (y Load mayor) que las otras, hacemos varias peticiones seguidas sin borrar las cookies (para que se hagan todas a la misma instancia por las sticky sessions), luego borramos las cookies, y volvemos a hacer varias peticiones sin borrarlas (para que se hagan todas a la misma instancia por las sticky sessions, pero siendo

distinta a la instancia de la serie de peticiones anterior (por lo de Load mayor en las instancias poco utilizadas, que va a hacer que no se siga utilizando una instancia que recientemente ha estado más cargada si es posible, y en este caso lo es porque hemos borrado la cookie de sticky session)). Para borrar las cookies, lo hacemos así:



Tras hacer todo esto que hemos dicho, con 5 peticiones a cada instancia (5 peticiones antes de borrar la cookie de sticky session ROUTEID), nos queda todo como queremos, teniendo la Instancia 3 0 en Elected (y 1000 en Load), y las otras dos 5 (y -500 en Load):



Ahora, sabemos que por el reparto equitativo de peticiones que intenta hacer el balanceador (mediante Load, como se explicó en uno de los apartados anteriores), las peticiones que no tengan la cookie ROUTEID (que no sean de una sesión sticky) se intentaran redirigir a la instancia 3 para compensar que esta no ha estado cargada recientemente en comparación con las otras.

Con esto en mente, pararemos la instancia 3 (parando la VM3 completa y no sólo el demonio de gunicorn, para que la máquina no acepte conexiones y no confunda al balanceador). Aunque es inútil porque vamos a parar la VM3 completa, como se dice en el enunciado, también paramos el demonio de gunicorn antes:

```
si2@3:~$ sudo systemctl stop gunicorn
si2@3:~$ sudo systemctl status gunicorn
○ gunicorn.service - Gunicorn WSGI Application Server
     Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
     Active: inactive (dead) since Tue 2025-04-29 13:37:41 CEST; 45s ago
   Duration: 2min 44.445s
    Process: 701 ExecStart=/home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application (code=exited, >
   Main PID: 701 (code=exited, status=0/SUCCESS)
        CPU: 1.201s

Apr 29 13:35:00 3 gunicorn[701]: [2025-04-29 13:34:59 +0200] [701] [INFO] Listening at: http://0.0.0.0:8000 (701)
Apr 29 13:35:00 3 gunicorn[701]: [2025-04-29 13:34:59 +0200] [701] [INFO] Using worker: sync
Apr 29 13:35:00 3 gunicorn[936]: [2025-04-29 13:35:00 +0200] [936] [INFO] Booting worker with pid: 936
Apr 29 13:37:40 3 gunicorn[701]: [2025-04-29 13:37:40 +0200] [701] [INFO] Handling signal: term
Apr 29 13:37:40 3 systemd[1]: Stopping gunicorn.service - Gunicorn WSGI Application Server...
Apr 29 13:37:40 3 gunicorn[936]: [2025-04-29 13:37:40 +0200] [936] [INFO] Worker exiting (pid: 936)
Apr 29 13:37:40 3 gunicorn[701]: [2025-04-29 13:37:40 +0200] [701] [INFO] Shutting down: Master
Apr 29 13:37:41 3 systemd[1]: gunicorn.service: Deactivated successfully.
Apr 29 13:37:41 3 systemd[1]: Stopped gunicorn.service - Gunicorn WSGI Application Server.
Apr 29 13:37:41 3 systemd[1]: gunicorn.service: Consumed 1.201s CPU time.
lines 1-18/18 (END)
```

Probamos ahora a hacer una petición habiendo borrado las cookies, y nos sale lo siguiente (imágenes), cosa que muestra que sí que se nos redirige a la página correcta. Viendo la consola del balanceador, se puede apreciar que sí que se ha detectado la instancia 3 como errónea (Status a Init Err). Podemos ver también que Elected de la instancia 3 es 1, y de la 1 es 6 (1 más en ambos); y que en ambos a crecido el número en la columna To, pero sólo en la instancia 1 ha crecido el número en la columna From. Además, los Load se han reajustado en base a las peticiones hechas a cada instancia, pero esto ahora no nos importa demasiado más alla de ver que la instancia 3 sigue siendo la que se debería elegir al ser la del Load más alto (menos usada anteriormente en comparación con las otras). Aunque es obvio, estos resultados se deben a que se ha intentado hacer esta petición a la instancia 3, no ha devuelto nada al estar inactiva (y se ha marcado como inactiva), y entonces el balanceador la ha redirigido a la instancia 1, que sí que ha devuelto la respuesta correspondiente:

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

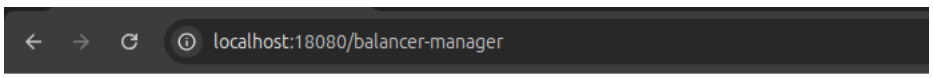
Número de DNI:

Nombre:

Fecha de Nacimiento:

Código de Autorización:

Enviar Información Censo



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

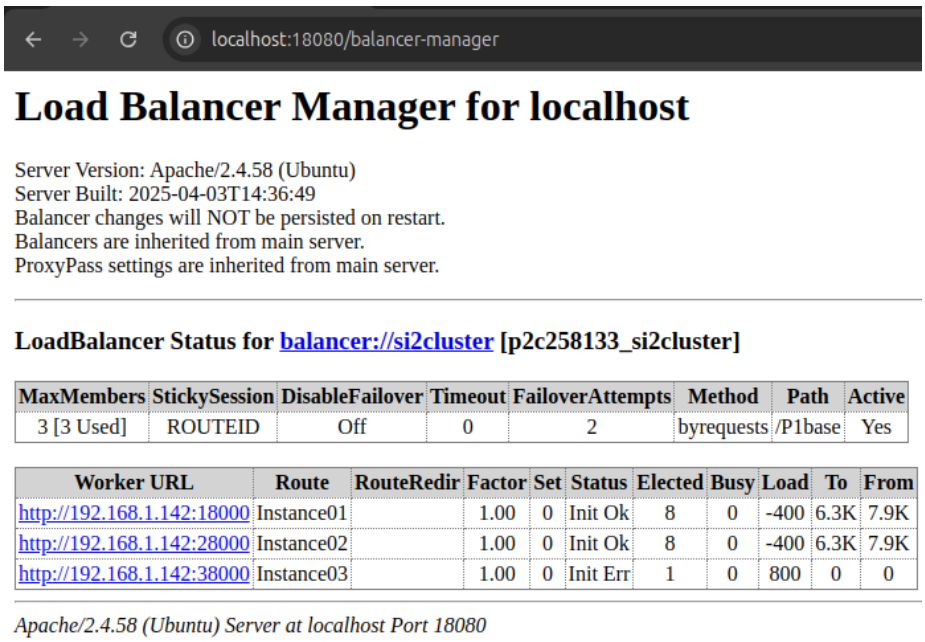
MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	6	0	-500	4.8K	5.9K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	5	0	-300	4.0K	4.9K
http://192.168.1.142:38000	Instance03		1.00	0	Init Err	1	0	800	0	0

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Si volvemos a hacer peticiones borrando las cookies antes de cada una, apreciamos que todas devuelven

una respuesta correcta, que la instancia 3 sigue marcada como errónea (Init Err) y que ya ni siquiera se intenta hacerle peticiones (no aumenta Elected y To en la instancia 3, sino en las otras) sino que se redirigen a las otras instancias, incluso aunque le tocase a la instancia 3 por equilibración de carga (Load mayor que el del resto de instancias):



Ejercicio número 6:

Vericar que el balanceador de carga reactiva correctamente una instancia previamente detenida (failback).

- **Inicie la instancia detenida. Para ello simplemente vuelva a lanzar el servicio de gunicorn y compruebe en los logs que se inicia correctamente, como medida adicional, prueba a acceder directamente a la instancia desde el navegador sin pasar por el balanceador y compruebe que cargue la aplicación.**
- **Accede nuevamente a la página /balancer-manager del balanceador de carga y verifica que la instancia previamente caída se haya reactivado y esté ahora en funcionamiento.**
- **Vuelve a realizar una votación completando los datos del censo como en ejercicios anteriores.**
- **Observar cómo el balanceador distribuye las solicitudes entre las instancias activas. Si el failback está funcionando correctamente, la carga debe distribuirse entre las instancias disponibles, incluyendo la recién reactivada.**
- **Incluye las evidencias del proceso de reactivación de la instancia, así como cualquier cambio visible en la consola del balanceador de carga.**

Iniciamos la instancia 3 (volvemos a encender la VM3, y lanzamos el servicio de gunicorn) que había sido detenida en el anterior ejercicio y comprobamos que está activa mediante el comando status (en esta captura se hace el start también), viendo los logs y accediendo en el navegador al endpoint censo desde gunicorn de la VM3 sin el balanceador (en el puerto 38000):

```
si2@3:~$ sudo systemctl start gunicorn
si2@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 Tue 2025-04-29 13:41:55 CEST; 2s ago
   Main PID: 1558 (gunicorn)
      Tasks: 2 (limit: 1670)
     Memory: 42.8M (peak: 43.1M)
        CPU: 664ms
     CGroup: /system.slice/gunicorn.service
             ├─1558 /home/si2/venv/bin/python3 /home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:applic>
             └─1559 /home/si2/venv/bin/python3 /home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:applic>

Apr 29 13:41:55 3 systemd[1]: Started gunicorn.service - Gunicorn WSGI Application Server.
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Starting gunicorn 20.1.0
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Listening at: http://0.0.0.0:8000 (1558)
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Using worker: sync
Apr 29 13:41:55 3 gunicorn[1559]: [2025-04-29 13:41:55 +0200] [1559] [INFO] Booting worker with pid: 1559
lines 1-16/16 (END)
```

```
si2@3:~$ journalctl -u gunicorn -f
Hint: You are currently not seeing messages from other users and the system.
      Users in groups 'adm', 'systemd-journal' can see all messages.
      Pass -q to turn off this notice.
Apr 29 13:35:00 3 gunicorn[701]: [2025-04-29 13:34:59 +0200] [701] [INFO] Listening at: http://0.0.0.0:8000 (701)
Apr 29 13:35:00 3 gunicorn[701]: [2025-04-29 13:34:59 +0200] [701] [INFO] Using worker: sync
Apr 29 13:35:00 3 gunicorn[936]: [2025-04-29 13:35:00 +0200] [936] [INFO] Booting worker with pid: 936
Apr 29 13:37:40 3 gunicorn[701]: [2025-04-29 13:37:40 +0200] [701] [INFO] Handling signal: term
Apr 29 13:37:40 3 gunicorn[936]: [2025-04-29 13:37:40 +0200] [936] [INFO] Worker exiting (pid: 936)
Apr 29 13:37:40 3 gunicorn[701]: [2025-04-29 13:37:40 +0200] [701] [INFO] Shutting down: Master
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Starting gunicorn 20.1.0
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Listening at: http://0.0.0.0:8000 (1558)
Apr 29 13:41:55 3 gunicorn[1558]: [2025-04-29 13:41:55 +0200] [1558] [INFO] Using worker: sync
Apr 29 13:41:55 3 gunicorn[1559]: [2025-04-29 13:41:55 +0200] [1559] [INFO] Booting worker with pid: 1559
```

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

Número de DNI:

Nombre:

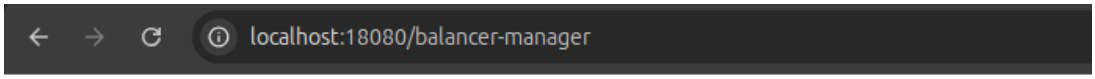
Fecha de Nacimiento:

Código de Autorización:

Enviar Información Censo

Ahora, sabemos que la instancia que el balanceador debería intentar usar es la 3 (que antes estaba inactiva y acabamos de reactivar) por lo justificado en anteriores apartados de que se intentará coger la instancia con mayor Load si está disponible por la equilibración de la carga de trabajo que hace el balanceador ya descrita (tener en cuenta que estamos en el estado del balanceador de la úlima imágen del anterior apartado). Para comprobar esto, antes de registrar un voto, vamos a probar a hacer varias peticiones al endpoint censo eliminando las cookies antes de hacerlas igual que en el ejercicio anterior con el fin comprobar que las solicitudes se distribuyen entre las instancias activas, para lo que debemos evitar las sticky sessions (por eso eliminamos las cookies) (con sticky sessions, una vez hagamos una petición, nuestras peticiones siguientes irán todo el rato a la misma instancia; y no queremos eso, sino que se distribuyan entre las instancias de la manera equitativa en base a la carga de trabajo relativa que han tenido estas instancias).

Tras hacer una petición habiendo borrado las cookies vemos que la instancia 3 ya ha podido responderla (From es mayor que 0 y Status es Init Ok, con lo que ha respondido; y To y Elected han aumentado, con lo que se le ha enviado la petición) y que se ha vuelto a marcar su Status como Init Ok (activa) en vez de Init Err:



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

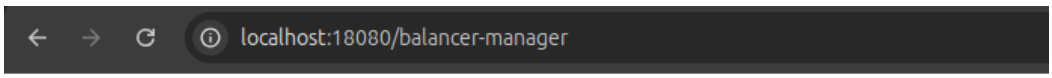
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	8	0	-300	6.3K	7.9K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	8	0	-300	6.3K	7.9K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	2	0	600	798	1.0K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Si seguimos haciendo peticiones borrando las cookies antes de cada una observamos que Load se va equilibrando entre las 3 instancias al hacer más peticiones, y el Elected de la instancia 3 (peticiones hechas a la misma) se acerca un poco más a los de las otras instancias:



Load Balancer Manager for localhost

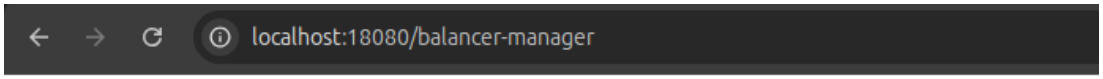
Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	8	0	0	6.3K	7.9K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	8	0	0	6.3K	7.9K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	5	0	0	3.1K	3.9K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

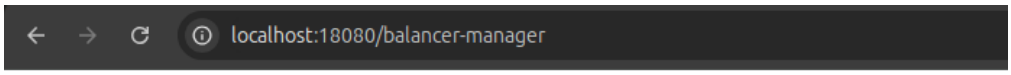
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	11	0	0	8.7K	11K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	11	0	0	8.7K	11K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	8	0	0	5.5K	6.9K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Si hacemos otras dos peticiones borrando las cookies antes de cada una a partir de este estado de Load equilibrado (0 todas), podemos ver que Load se queda desigual ya que las atienden las instancias 1 y 2, teniendo que ser la próxima la 3 (por la equilibración de carga):



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	12	0	-100	9.5K	12K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	12	0	-100	9.5K	12K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	8	0	200	5.5K	6.9K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Ahora, probaremos a registrar un voto desde los endpoints censo y voto (es decir, sin eliminar las cookies, con sticky sessions), y sabemos que este registro lo debería comenzar la instancia 3 (obtención del formulario del censo) por lo comentado en el anterior párrafo (de la equilibración de carga con Load), y luego también continuarlo por las sticky sessions. Se destaca que usamos un censo válido de los votos en los antriores apartados, y eliminamos los votos de este:

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

Número de DNI: 39739740E

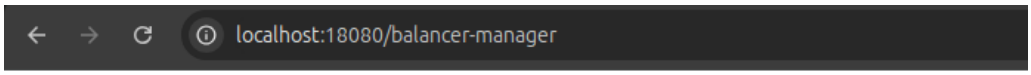
Nombre: Jose Moreno Locke

Fecha de Nacimiento: 09/04/66

Código de Autorización: 729

Enviar Información Censo

Se verifica el censo correctamente y se nos redirige al endpoint voto, y vemos en la consola del balanceador que la instancia 3 es la que ha hecho las dos peticiones (enviar los datos del censo y obtener el formulario de voto) (+2 en Elected y los cambios correspondientes en las otra columnas) como habíamos dicho que debería pasar:



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

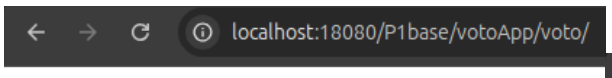
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	12	0	100	9.5K	12K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	12	0	100	9.5K	12K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	10	0	-200	7.4K	8.0K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Ahora, registramos el voto, y observamos que todo funciona correctamente:



Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

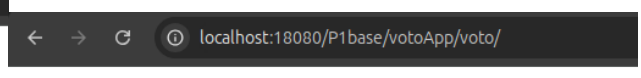
ID Proceso Electoral: 1

ID Circunscripcion: 1

ID Mesa Electoral: 1

Nombre Candidato Votado: Candidato

Enviar Información Voto



Voto Registrado con Éxito (votoSite)

Id: 3007

Codigo Respuesta: 000

Marca Tiempo : April 29, 2025, 2:01 p.m.

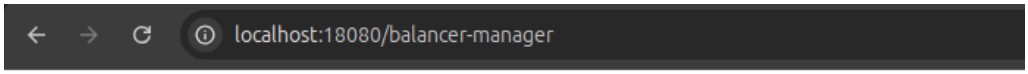
Id Circunscripcion : 1

Id Mesa Electoral : 1

Id Proceso Electoral : 1

Nombre Candidato Votado: Candidato

Ahora, para demostrar la justificación que hicimos antes de por qué hemos hecho lo de hacer peticiones sin cookies para enseñar el balanceo de carga (en vez de registrar un voto por censo y voto), podemos ver que esta petición de registrar el voto la ha llevado la instancia 3 (la que había verificado el censo) (+1 en Elected y los cambios correspondientes en las otra columnas) por el tema de las sticky sessions, incluso aunque tenía un Load menor que las otras instancias (por balanceo de carga no se debería escogido la instancia 3, pero era necesario por las sticky sessions):



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	12	0	200	9.5K	12K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	12	0	200	9.5K	12K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	11	0	-400	8.5K	8.4K

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

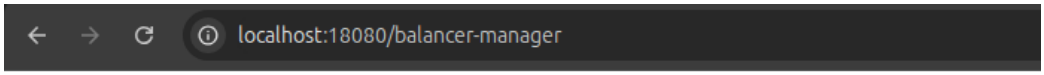
Ejercicio número 7:

Verificar el comportamiento del balanceador y el proceso de failover en el transcurso de una sesión activa.

- **Abre un navegador y comienza una votación completando el formulario.**
- **Justo cuando la pantalla pida completar la información sobre el censo, observa la instancia que ha procesado la solicitud y detén esa instancia.**
- **Completar los datos del censo de modo que el voto fuera válido, y enviar la petición.**
- **Observar la instancia del cluster que procesa el voto, y razonar las causas por las que se rechaza la petición de registro de voto.**

Se destaca antes de nada que cuando pararemos la instancia será tras verificar el censo aunque en el enunciado se entienda que es antes de enviar el formulario de censo. Esto lo hacemos así porque el profesor nos ha dicho que esta es la manera correcta de hacerlo.

Primero, para mayor claridad y para que se nos reinicien las columnas de la consola del balanceador, hacemos un sudo systemctl restart apache2:



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

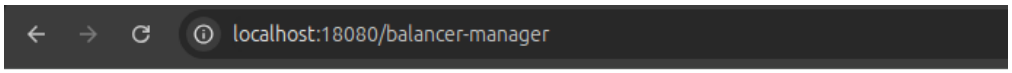
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	0	0	0	0	0
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	0	0	0	0	0
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	0	0	0	0	0

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

También hay que destacar que hemos borrado los votos que había. Para que se seleccione la instancia 2 para esta sesión (ya que no podemos permitir que se seleccione la instancia 1 porque vamos a parar la VM y en la VM1 está también el balanceador), hacemos un get de censo, que se enviará a la instancia 1; y luego borramos las cookies porque sabemos que por balanceo de carga la instancia 1 ya no se cogerá para la siguiente petición:



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

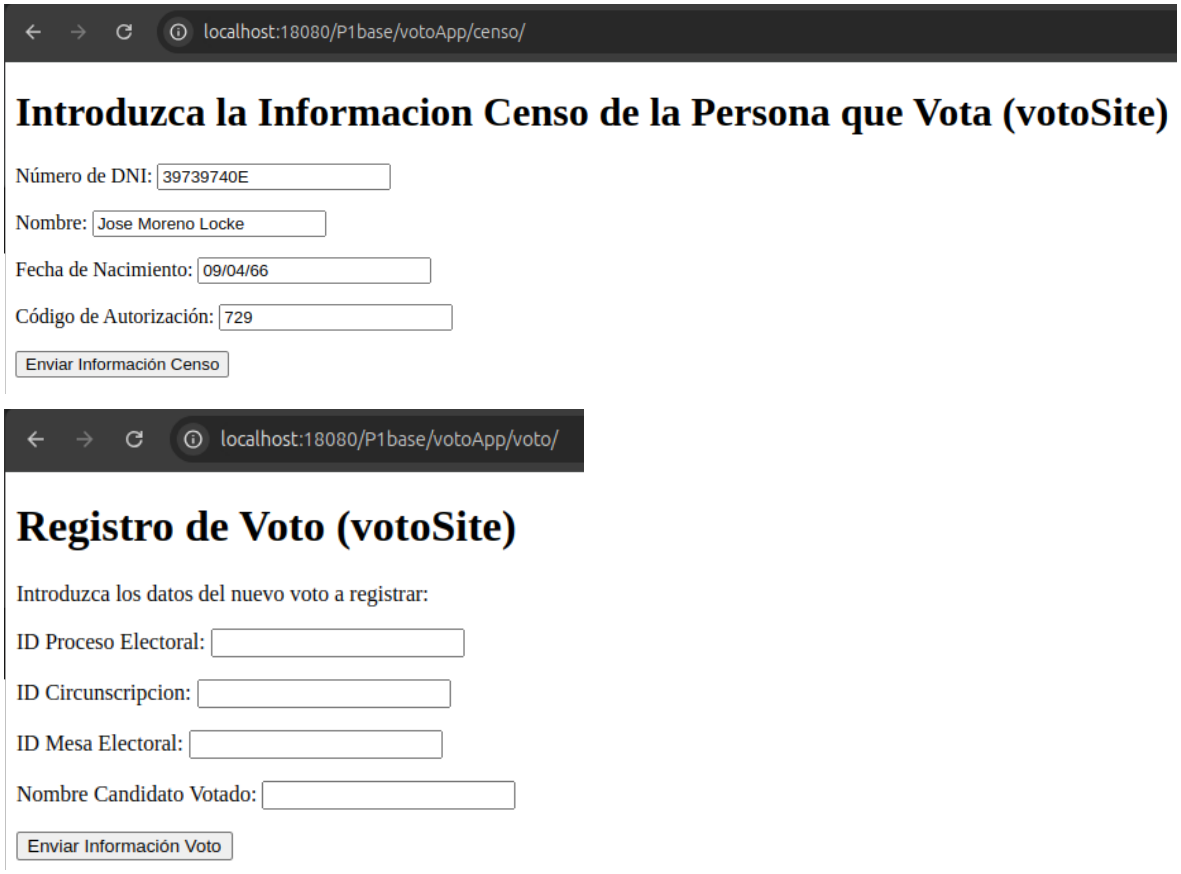
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

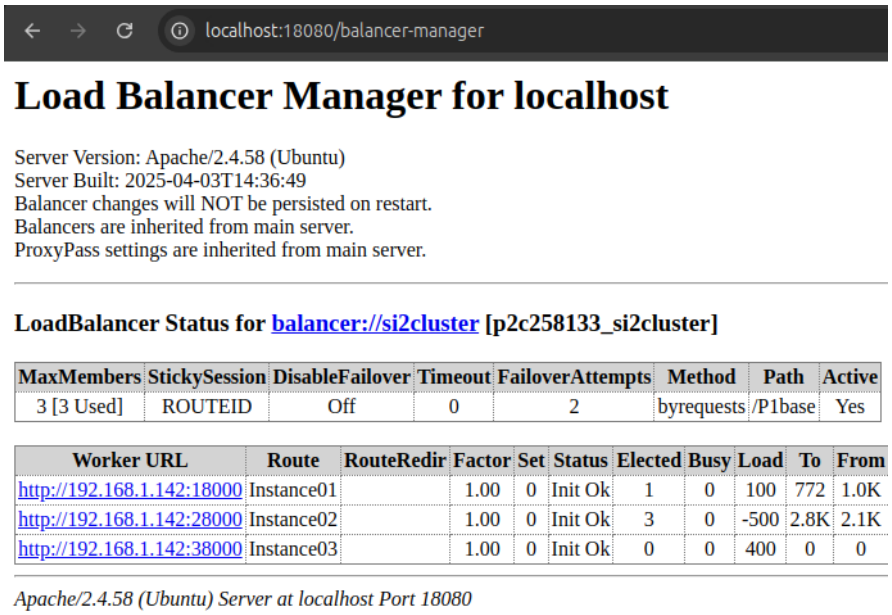
Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	1	0	-200	772	1.0K
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	0	0	100	0	0
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	0	0	100	0	0

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

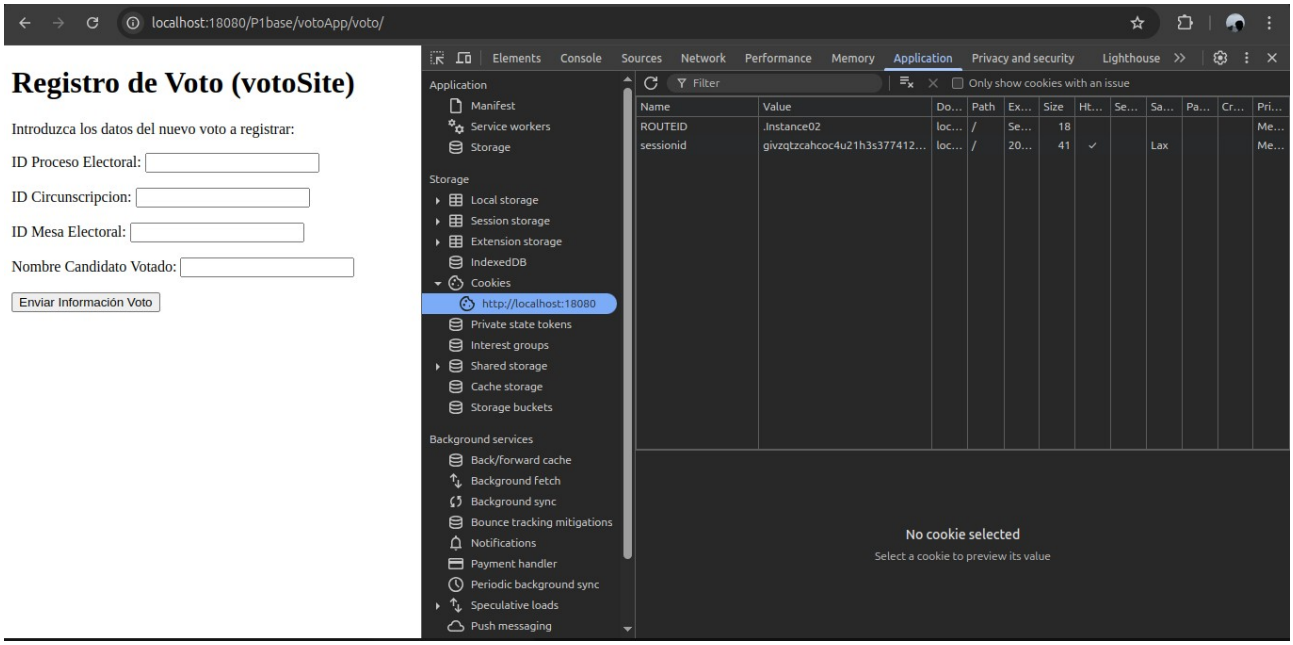
Ahora, sabiendo que las sticky sessions están activadas, introduciremos un censo que sabemos que es válido (por los vistos en DBeaver en apartados anteriores) para que nos salga el formulario de voto:



Y ahora podemos ver en la consola del balanceador que por las sticky sessions, se estaba usando para esta sesión la instancia 2 (las nuevas peticiones iban a ella):



También podemos ver que esta es la instancia que se ha escogido para la sesión en la cookie del navegador ROUTEID (además de que se ve la cookie del id de la sesión que se generó tras validar el censo, sesión almacenada en la caché de la instancia 2):



Ahora, pararemos la instancia 2 (parando la VM2 completa y no sólo el demonio de gunicorn, para que la máquina no acepte conexiones y no confunda al balanceador). Aunque es inútil porque vamos a parar la VM2 completa, como se dice en el enunciado, también paramos el demonio de gunicorn antes:

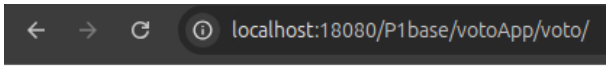
```
si2@2:~$ sudo systemctl stop gunicorn
si2@2:~$ sudo systemctl status gunicorn
○ gunicorn.service - Gunicorn WSGI Application Server
     Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
     Active: inactive (dead) since Tue 2025-04-29 14:29:48 CEST; 6s ago
   Duration: 1h 28min 37.739s
    Process: 698 ExecStart=/home/si2/venv/bin/gunicorn --workers 1 --bind 0.0.0.0:8000 votoSite.wsgi:application (code=exited, >
   Main PID: 698 (code=exited, status=0/SUCCESS)
        CPU: 3.504s

Apr 29 13:01:16 2 gunicorn[698]: [2025-04-29 13:01:16 +0200] [698] [INFO] Listening at: http://0.0.0.0:8000 (698)
Apr 29 13:01:16 2 gunicorn[698]: [2025-04-29 13:01:16 +0200] [698] [INFO] Using worker: sync
Apr 29 13:01:16 2 gunicorn[1052]: [2025-04-29 13:01:16 +0200] [1052] [INFO] Booting worker with pid: 1052
Apr 29 14:29:48 2 systemd[1]: Stopping gunicorn.service - Gunicorn WSGI Application Server...
Apr 29 14:29:48 2 gunicorn[698]: [2025-04-29 14:29:48 +0200] [698] [INFO] Handling signal: term
Apr 29 14:29:48 2 gunicorn[1052]: [2025-04-29 14:29:48 +0200] [1052] [INFO] Worker exiting (pid: 1052)
Apr 29 14:29:48 2 gunicorn[698]: [2025-04-29 14:29:48 +0200] [698] [INFO] Shutting down: Master
Apr 29 14:29:48 2 systemd[1]: gunicorn.service: Deactivated successfully.
Apr 29 14:29:48 2 systemd[1]: Stopped gunicorn.service - Gunicorn WSGI Application Server.
Apr 29 14:29:48 2 systemd[1]: gunicorn.service: Consumed 3.504s CPU time.
lines 1-18/18 (END)
```

Si ahora rellenamos el formulario de voto y lo enviamos, vemos que alguna de las instancias activas procesa nuestra solicitud, que se marca la instancia 2 como inactiva (Init Err en Status), y que la petición que se le intenta hacer a la instancia 2 se reenvía tras fallar a la instancia 3 (Elected aumenta en 1 en ambos, siendo 1 en la instancia 3 y 4 en la instancia 2).

En cuanto al resultado, se puede apreciar que el servidor no ha encontrado el id del censo verificado en su sesión. Esto, como ya se ha explicado en otros contextos en los antriores apartados, se debe a que:

El servidor almacena en su caché (configurado que sea la caché en vez de la BD en el settings.py, como se mencionó en el enunciado) el id/DNI del censo verificado y asigna el id de esa sesión como cookie al cliente (para que la envíe en las próximas solicitudes). Tras esto, ya sea redireccionado automáticamente o accediendo manualmente al endpoint voto, al enviar el formulario del voto, el navegador enviará la cookie con el id de la sesión. En caso de que la instancia que tiene los datos de la sesión en su caché (que en este caso es la 2, que está inactiva) no sea la que se use para atender este envío del formulario del registro del voto con la cookie del id de la sesión (la sesión la tiene guardada en su caché la instancia 2 pero por la caída de esta, el registro del voto lo intenta hacer la instancia 3), no se podrá registrar el voto al no tener el id/DNI del censo verificado previamente esta instancia (porque esta instancia no tiene en su caché los datos de la sesión de la otra instancia), haciendo que salga el fallo de que el DNI del censo verificado no está en la sesión, como se ha visto.



Registro de Voto (votoSite)

Introduzca los datos del nuevo voto a registrar:

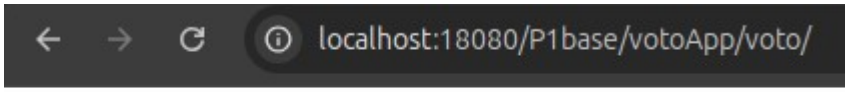
ID Proceso Electoral: 1

ID Circunscripcion: 1

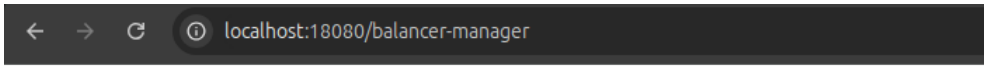
ID Mesa Electoral: 1

Nombre Candidato Votado: Candidato

Enviar Información Voto



¡Error: DNI no encontrado en la sesión!



Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

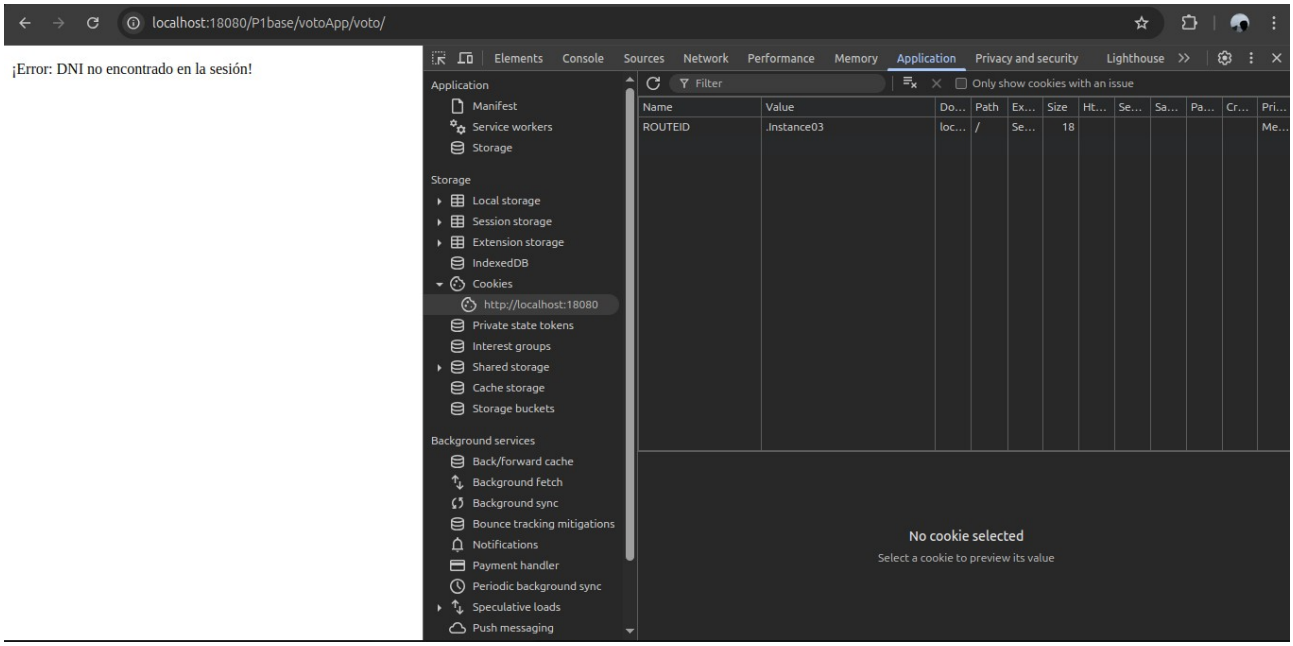
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	1	0	300	798	1.0K
http://192.168.1.142:28000	Instance02		1.00	0	Init Err	4	0	-700	2.7K	2.1K
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	1	0	400	1.1K	211

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Si miramos las cookies, como curiosidad, se puede ver que la de las sticky sessions ROUTEID se ha cambiado de la instancia 2 a la 3 para que ahora sea esta (una que sí que esté activa) la que gestione esta sesión. Además, se ha eliminado la cookie del id de la sesión ya que tras fallar o completarse exitosamente el registro del voto, ya no se necesita la sesión con el id del censo verificado:

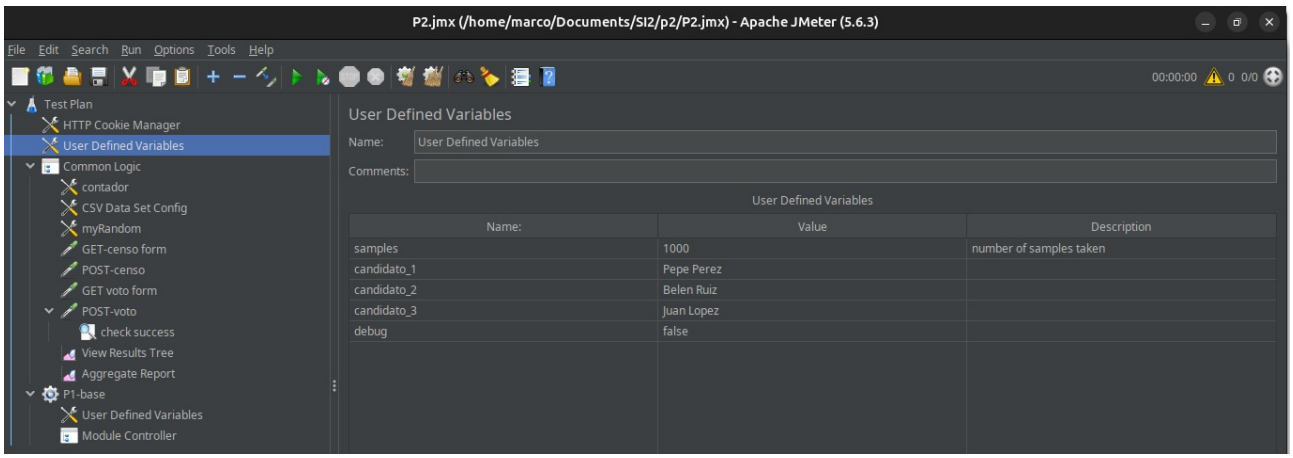


Ejercicio número 8:

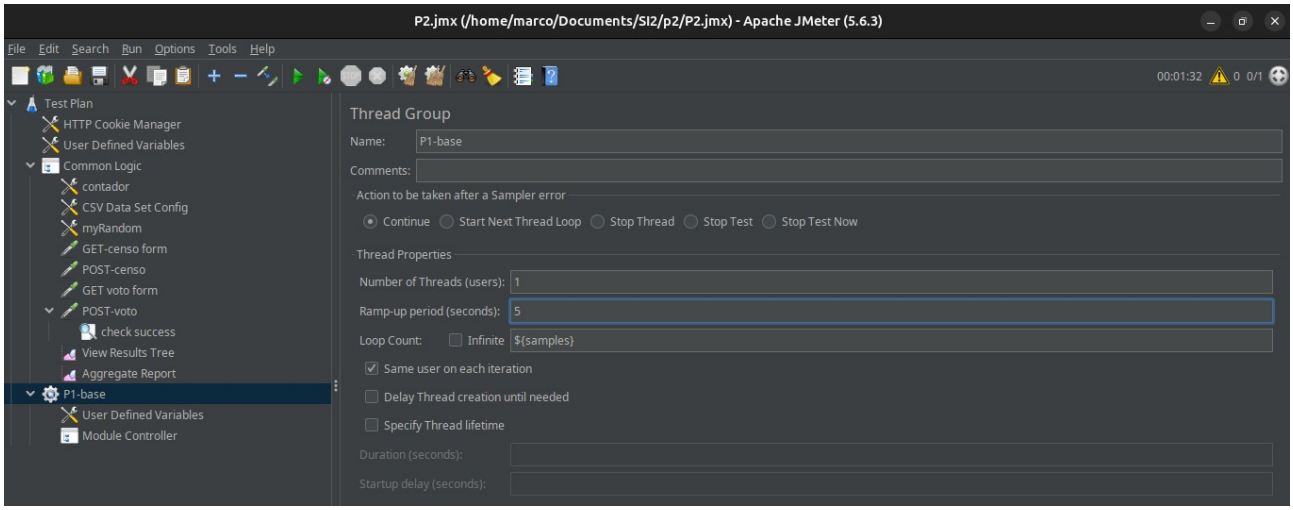
Pruebas de carga con Jmeter

- **Abre el archivo de pruebas P2.jmx y modifica el ciclo de pruebas para realizar 1000 pruebas en un solo hilo contra la IP del clúster y la nueva URL de la aplicación (Asegúrate de eliminar cualquier voto previo antes de iniciar el ciclo de pruebas para evitar interferencias).**
- **Limpia los datos de apache de forma que ambas instancias empiecen con 0 datos.**
- **Después de ejecutar las pruebas, verifica cómo se distribuyen los votos entre las instancias del clúster.**
- **deduce qué algoritmo de reparto parece estar utilizando el balanceador de carga. Algunas preguntas clave a responder son:**
  - **¿El balanceador distribuye las solicitudes de manera equitativa?**
  - **¿Hay alguna instancia que reciba signicativamente más o menos solicitudes que otras?**
  - **¿El balanceador parece seguir un algoritmo de round-robin o un algoritmo de afinidad de sesión basado en la cookie ROUTEID?**
- **Incluye las evidencias de los resultados obtenidos en JMeter y cualquier comentario sobre el comportamiento observado durante las pruebas.**

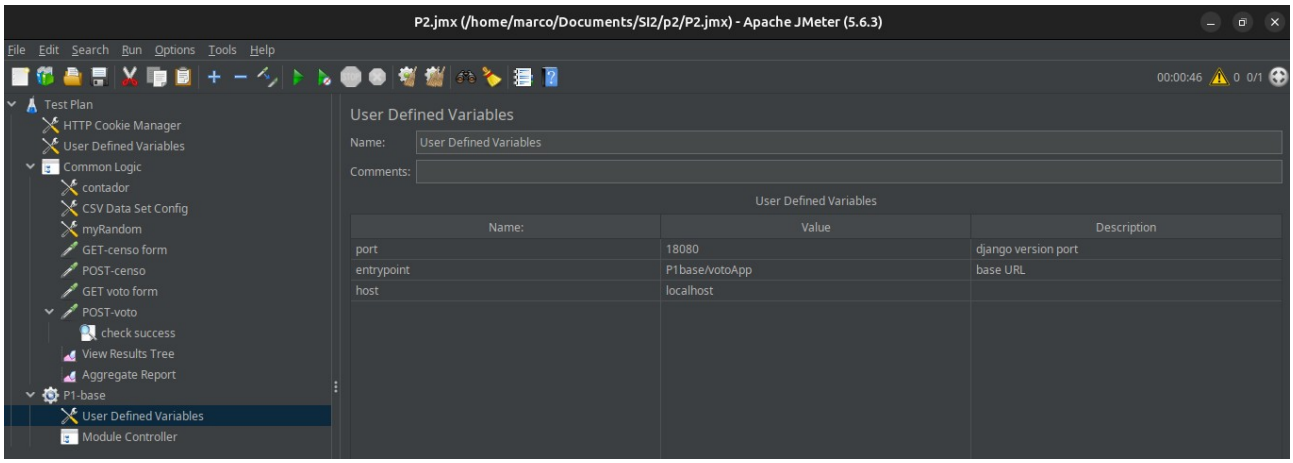
Con todas las instancias activas de nuevo (sudo systemctl start gunicorn ejecutado en la instancia 2 que estaba inactiva para activarla, tras encender la VM2), primero, en el fichero de pruebas P2.jmx (que tiene sólo el Thread Group P1-base y no los otros porque en el balanceador sólo usamos P1-base), ponemos samples a 1000 (loop count):



Ahora ponemos number of threads a 1, Ramp-up a 5 segundos (pequeño en comparación con el tiempo de la ejecución, que es un minuto y medio aproximadamente):



Y ponemos el puerto como 18080 (puerto de apache (balanceador) de la VM1), y en el entrypoint, antes de votoApp añadimos P1base/:



Hacemos ahora un populate para limpiar y repoblar la base de datos:

```
si2@1:~/repo/p1base/p1/P1-base$ python manage.py populate
Database cleaned successfully
Censo objects created successfully
si2@1:~/repo/p1base/p1/P1-base$
```

Le hacemos un restart a apache para que las instancias comiencen con 0 datos (se muestra también una captura en la que se ve que tienen 0 datos):

```
si2@1:~/repo/p1base/p1/P1-base$ sudo systemctl restart apache2
si2@1:~/repo/p1base/p1/P1-base$
```

localhost:18080/balancer-manager

Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

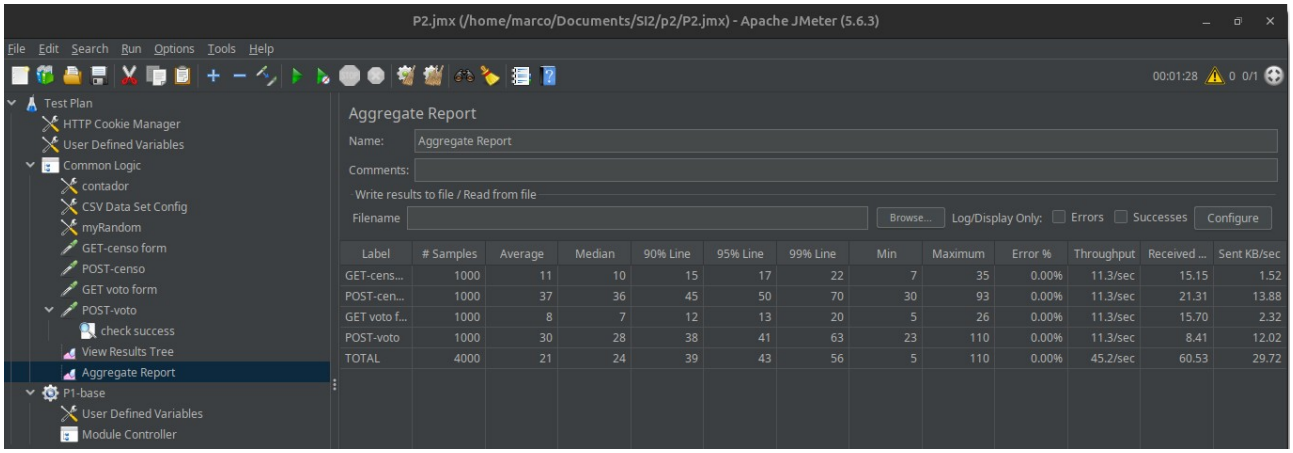
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

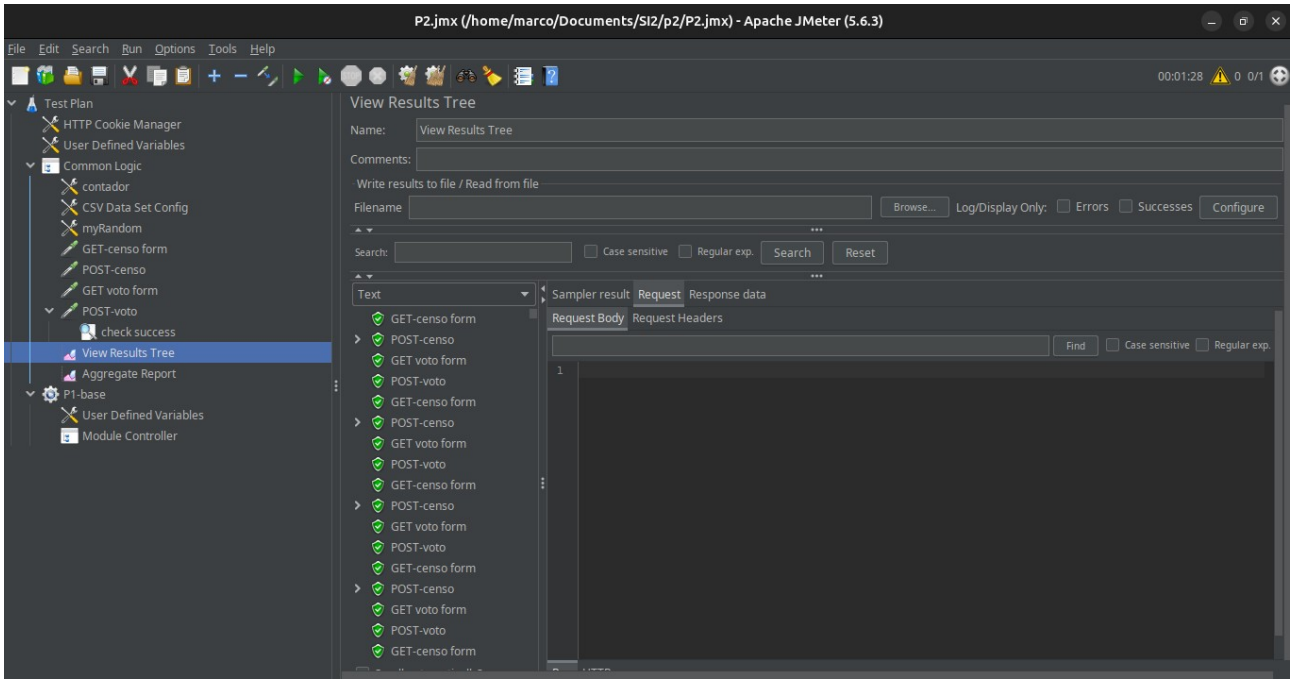
Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	0	0	0	0	0
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	0	0	0	0	0
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	0	0	0	0	0

Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Ahora ejecutamos el test (con interfaz gráfica ya que los tiempos no nos importan demasiado, sino que nos importa que se hagan las peticiones para ver cómo se distribuyen), y en el aggregate report nos sale lo siguiente. Podemos apreciar que no falló ninguna petición gracias en parte a la configuración de cookies que tenemos en jmeter que hace que se puedan usar correctamente. Se puede ver que se hicieron mil peticiones de cada tipo (las 4 peticiones distintas (un regsitro de voto) por cada una de las 1000 iteraciones del bucle). En cuanto a los tiempos de respuesta, aunque no nos importa demasiado para este apartado, se puede apreciar que los de los post son más altos que los de los get ya que implican una petición de mayor tamaño al enviar datos, y un mayor procesamiento en el servidor:



Y en el results tree, sale esto (todas las peticiones dan resultados correctos):



Mirando ahora la consola del balanceador, podemos ver que se hicieron más o menos las mismas peticiones a todas las instancias (Elected parecidos; también se ve en To y From iguales). Se han distribuido correctamente los votos entre las instancias.

- ¿El balanceador distribuye las solicitudes de manera equitativa?

  Sí, como se puede ver.

- ¿Hay alguna instancia que reciba signicativamente más o menos solicitudes que otras?

  No, todas han recibido aproximadamente las mismas solicitudes que las otras.

- ¿El balanceador parece seguir un algoritmo de round-robin o un algoritmo de afinidad de sesión basado en la cookie ROUTEID?

  El balanceador sigue un algoritmo round-robbin para realizar el reparto equitativo de las solicitudes mediante Load (ver explicación de cómo lo usa el balanceador para equilibrar la carga, explicado en los anteriores apartados) (Load más alto indica que la instancia ha sido menos usada anteriormente en comparación con las otras, con lo que se intenta seleccionar para la siguiente solicitud si no hay impedimentos con la afinidad de sesión (ROUTEID)), seimpre que no haya impedimentos por las sticky sessions. Sin embargo tenemos puesto en jmeter que en cada iteración del bucle (en la que se registra un voto haciendo las 4 peticiones) se mantengan las cookies, pero tras esta se eliminen, con lo que se hacen 4 solicitudes seguidas a la misma instancia (por la cookie ROUTEID de afinidad de sesión), y luego ya se distrubuye mediante el balanceo de carga a qué instancia se harán las siguientes 4 (otra iteración, otra instancia).

  Resumiendo, tiene prioridad la afinidad de sesión con ROUTEID para seleccionar instancia, pero si no hay cookie ROUTEID, las peticiones se distribuyen equitativamente mediante round-robbin.

  Podemos comprobar lo del round-robbin también en la consola del balanceador viendo que en la columna Method pone byrequests, lo que indica que se intentará balancear la carga de manera equitativa en base a las peticiones que ha hecho cada instancia.

(Se destaca que en en la consola del balanceador aparece que se han hecho 5000 peticiones (en total) mientras que en jmeter 4000 porque la de POST de censo redirecciona a la url de voto, con lo que se hace una petición extra en esta redireccion por cada una de las 1000 iteraciones del bucle, dando 1000 extras en total)

localhost:18080/balancer-manager

Load Balancer Manager for localhost

Server Version: Apache/2.4.58 (Ubuntu)
Server Built: 2025-04-03T14:36:49
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

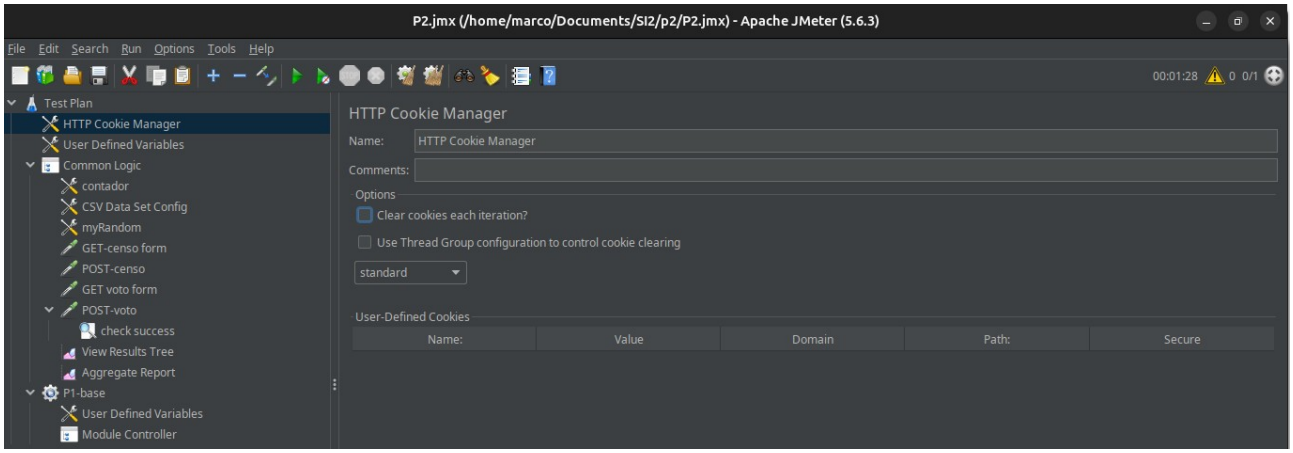
LoadBalancer Status for balancer://si2cluster [p2c258133_si2cluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
3 [3 Used]	ROUTEID	Off	0	2	byrequests	/P1base	Yes

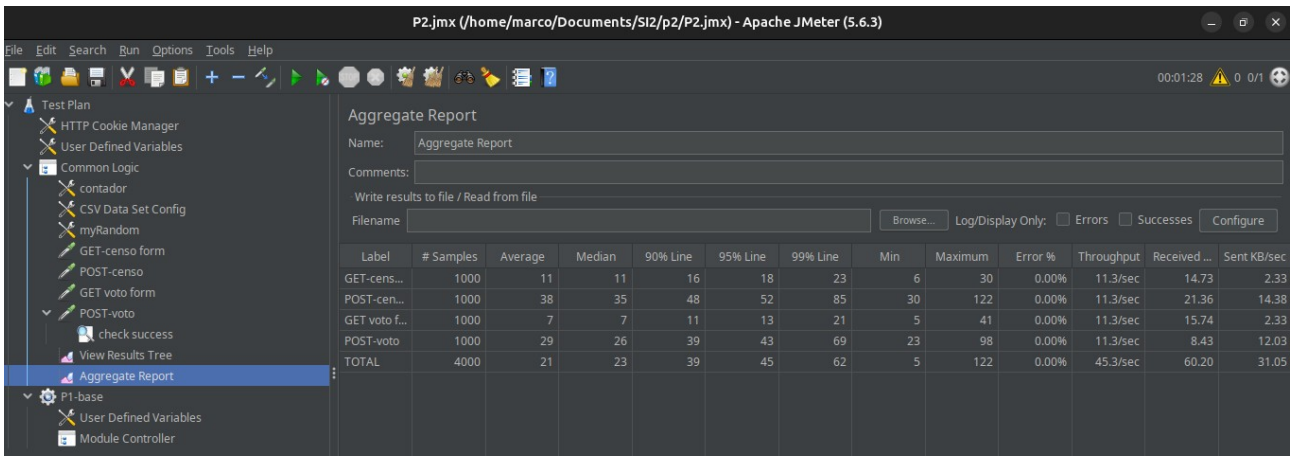
Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://192.168.1.142:18000	Instance01		1.00	0	Init Ok	1670	0	-1000	1.0M	1.2M
http://192.168.1.142:28000	Instance02		1.00	0	Init Ok	1665	0	500	1.0M	1.2M
http://192.168.1.142:38000	Instance03		1.00	0	Init Ok	1665	0	500	1.0M	1.2M

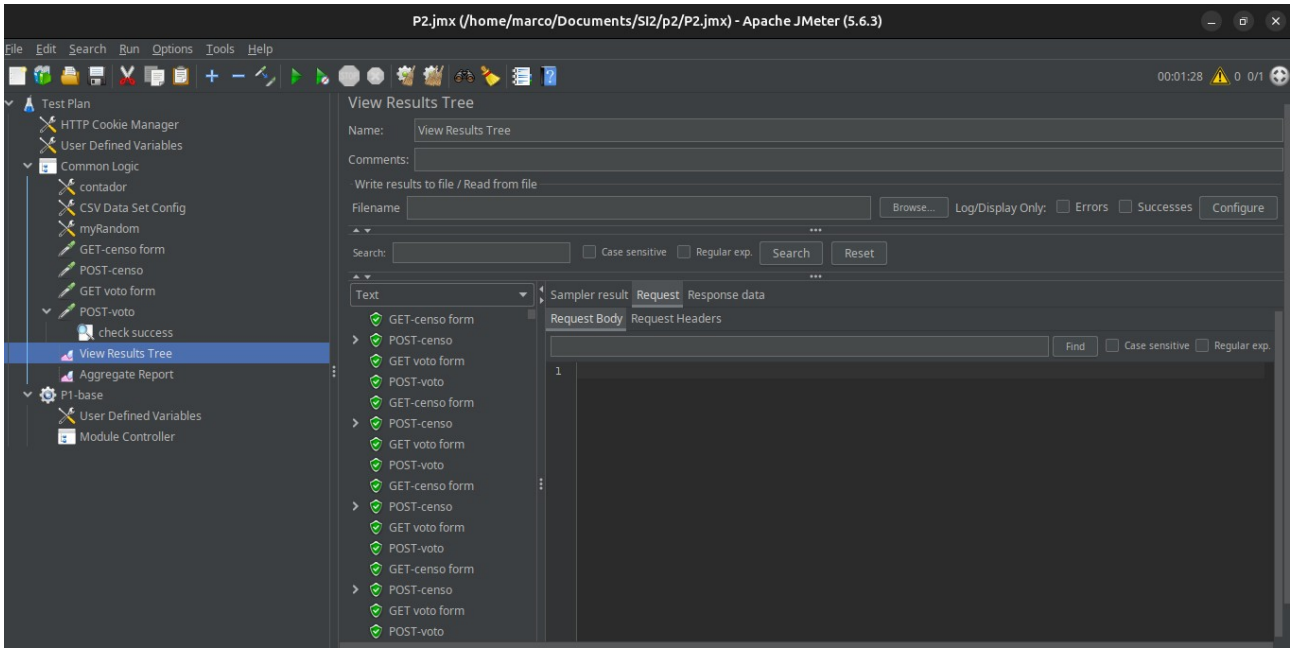
Apache/2.4.58 (Ubuntu) Server at localhost Port 18080

Como prueba extra, quitaremos la opción de que se eliminen las cookies tras cada iteración (registro de voto, las 4 peticiones), que antes estaba activada:



Ahora hacemos un restart a apache para que las instancias empiecen con 0 datos y un populate para reiniciar los votos (igual que antes). Luego, volvemos a ejecutar el test, y nos salen unos resultados de jmeter similares a los anteriores (ver mismos comentarios):





Pero ahora todas las peticiones han ido a parar a la instancia 1 porque es la que se usó en la primera petición, con lo que por afinidad de sesión que se mantenía entre iteraciones (la cookie ROUTEID con valor de la instancia 1 (usada en la primera iteración) se ha enviado en todas las peticiones), todas las peticiones se han hecho a la primera instancia. Si volvemos a responder a las preguntas de antes (estas respuestas de ahora no son las válidas, sino las de antes), tendríamos esto:

- ¿El balanceador distribuye las solicitudes de manera equitativa?

  No, como se puede ver.

- ¿Hay alguna instancia que reciba signicativamente más o menos solicitudes que otras?

  Si, la primera (todas: 5000 y las otras 0).

- ¿El balanceador parece seguir un algoritmo de round-robin o un algoritmo de afinidad de sesión basado en la cookie ROUTEID?

  Parece seguir un algoritmo basado sólo en ROUTEID, como hemos explicado, pero nosotros sabemos que no es así del todo por las preguntas respondidas antes (ver justificaciones), que son las verdaderamente válidas.