PROYECTO INTERMODULAR: SEGURIDAD Y ALTA DISPONIBILIDAD

Información necesaria para el desarrollo de las actividades del Proyecto Intermodular correspondientes al módulo de SAD en el curso 25-26

ENERO DE 2026

IES MIGUEL HERRERO PEREDA
Natalia Isidro Blázquez

Tabla de contenido

1. INTRODUCCIÓN

En este documento puedes encontrar la información que necesitas para desarrollar la parte del Proyecto Intermodular correspondiente al módulo de SAD para el curso 25-26. Estas tareas parten del servicio web desplegado en AWS para tu proyecto, que se utilizará como base para evolucionar la arquitectura hacia una solución con **mayor disponibilidad**, **redundancia** y **entrada segura**.

En SAD se evaluará específicamente la implantación de:

- un **proxy inverso** como componente de entrada del servicio,
- un **clúster** con **servidores redundantes** que garantice continuidad ante caída de nodos,
- un **balanceador de carga** que actúe además como **proxy inverso**, incluyendo **gestión de certificados** y reparto de conexiones entre servidores o clústeres.

A continuación, encontrarás tres apartados:

1. Enunciado con las instrucciones de lo que debes desarrollar.
2. Nota aclaratoria.
3. La lista de cotejo que se utilizará para evaluar. Se recomienda usarla como **guía de autoevaluación** antes de la entrega final.
4. Corrección

2. ENUNCIADO: Seguridad y alta disponibilidad

Como parte del Proyecto Intermodular del ciclo de Administración de Sistemas Informáticos en Red (ASIR), tu equipo deberá diseñar, desplegar, configurar y documentar una solución que permita publicar el servicio web del proyecto con un enfoque de **alta disponibilidad**, utilizando técnicas de **redundancia**, **balanceo** y **proxy inverso**, y aplicando buenas prácticas de **seguridad en la entrada del servicio**: endurecimiento y configuración de accesos, seguros así como mecanismos de redundancia y duplicidad de la información y balanceo.

En SAD se evaluarán mediante las actividades descritas a continuación, los siguientes criterios de evaluación:

RA 3: Implanta técnicas seguras de acceso remoto a un sistema informático, interpretando y aplicando el plan de seguridad

e) Se ha implantado un servidor como pasarela de acceso a la red interna desde ubicaciones remotas.

f) Se han identificado y configurado los posibles métodos de autenticación en el acceso de usuarios remotos a través de la pasarela.

g) Se ha instalado, configurado e integrado en la pasarela un servidor remoto de autenticación.

RA 5: Implanta servidores "proxy", aplicando criterios de configuración que garanticen el funcionamiento seguro del servicio.

h) Se ha configurado un servidor "proxy" en modo inverrso.

i) Se ha elaborado documentación relativa a la instalación, configuración y uso de servidores "proxy".

RA 6: Implanta soluciones de alta disponibilidad empleando técnicas de virtualización y configurando los entornos de prueba.

d) Se ha implantado un servidor redundante que garantice la continuidad de servicios en casos de caída del servidor principal, formando un clúster de servidores.

e) Se ha implantado un balanceador de carga y proxy inverso a la entrada de la red interna, gestionando certificados y repartiendo las conexiones entre los servidores de uno o varios clústeres de servicios.

f) Se han implantado sistemas de almacenamiento redundante sobre servidores y dispositivos específicos.

g) Se ha evaluado la utilidad de los sistemas de "clusters" para aumentar la fiabilidad y productividad del sistema.

NOTA: Hay una nota aclaratoria después del enunciado que es importante que sea tenida en cuenta.

Tareas a realizar:

1. Implantación de servidor redundante (clúster)

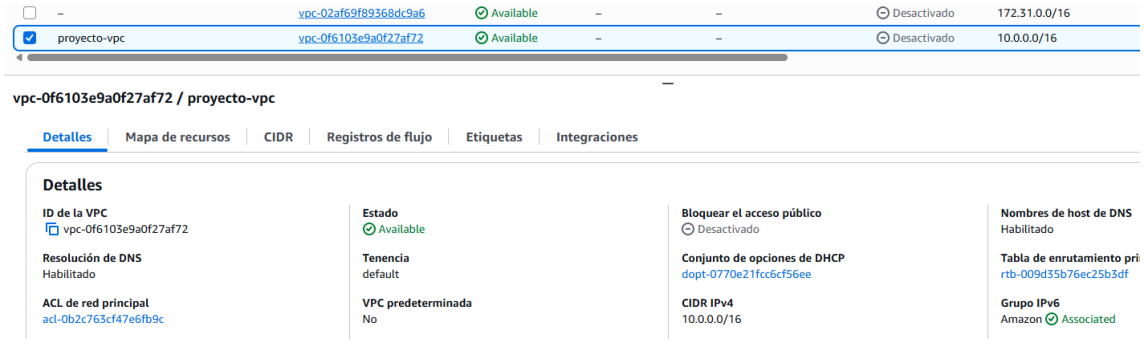
A partir de alguno de los servidores web existentes, se deberá implantar un segundo servidor (o más, si se desea) con el mismo rol o función, de forma que el servicio quede soportado por un **conjunto de servidores redundantes**.

La solución deberá estar orientada a garantizar la **continuidad del servicio** ante la caída de uno de los servidores.

El equipo deberá:

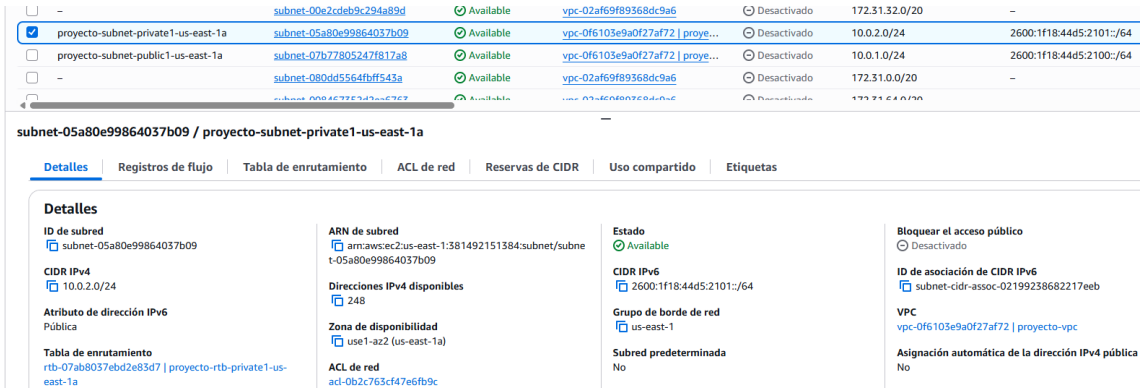
- implantar la redundancia creando el nuevo(s) servidor(es) y asegurando que el servicio web es funcional en todos los nodos,
- definir la estructura del clúster (nodos, función de cada uno y relación entre ellos),
- realizar pruebas de caída de uno de los servidores y evidenciar cómo se mantiene la continuidad del servicio mediante el nodo o nodos restantes.

Creamos una vpc donde estará toda la estructura de aws



Configurar subredes públicas y/o privadas según tu diseño.

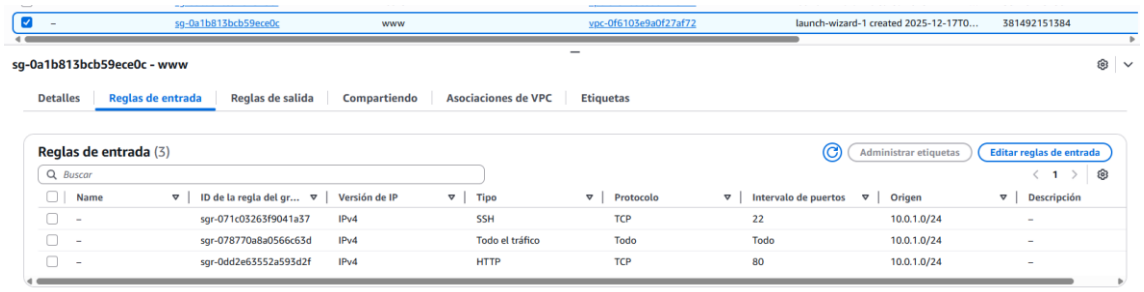
Vamos a crear 2 redes una publica y una privada. En la red pública estará expuesto nuestro proxy inverso y router a la vez que nos permitir que los servidores web estén situados en la red interna y que tengan conexión a internet.

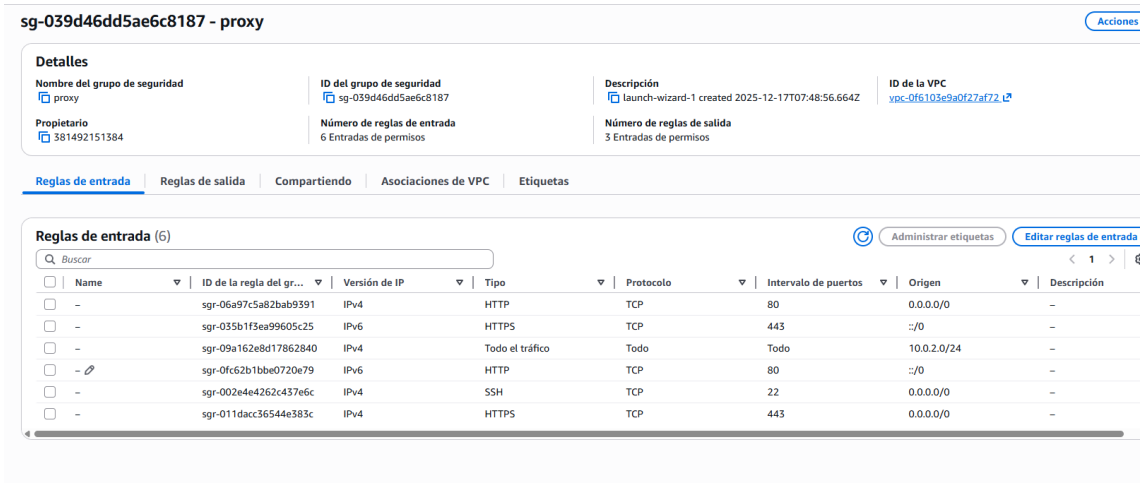


Usar grupos de seguridad bien configurados para proteger el servidor web.

En los grupos de seguridad configuraremos que solo se podrá acceder por ssh desde un equipo al proxy y desde el proxy a la red interna.

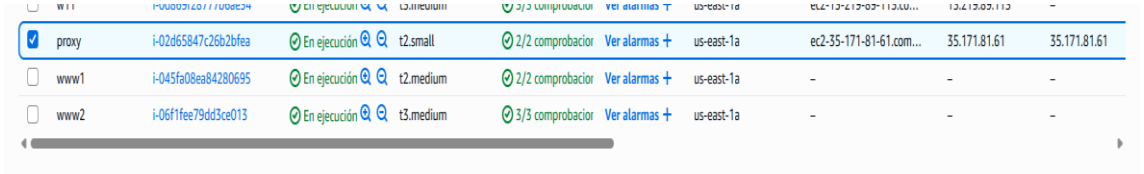
También que cualquiera se pueda conectar al proxy por https y que solo se pueda conectarse por http desde el proxy a los 2 servidores.





Desplegar una o varias instancias EC2 donde se alojará el servidor web.

Ahora vamos a crear la instancia que ya hemos hablado antes. También habrá que asignarles sus grupos de seguridad.



Instalación del servidor web Apache

Se ha instalado el servidor web **Apache** en la instancia EC2 de Ubuntu mediante el gestor de paquetes apt, ya que es una solución robusta, ampliamente documentada y adecuada para entornos educativos y empresariales.

Comandos ejecutados:

1. sudo apt update && sudo apt upgrade -y

```
root@debian:/etc/apache2# sudo apt update && sudo apt upgrade -y
Obj:1 http://deb.debian.org/debian trixie InRelease
Des:2 http://deb.debian.org/debian trixie-updates InRelease [47,3 kB]
Obj:3 http://security.debian.org/debian-security trixie-security InRelease
Descargados 47,3 kB en 0s (451 kB/s)
Se pueden actualizar 26 paquetes. Ejecute «apt list --upgradable» para verlos.
Upgrading:
  base-files  busybox      e2fsprogs       libc-bin      libc-l10n  libc6-dev  lib
```

2. sudo apt install apache2 -y

```
root@debian:/etc/apache2# sudo apt install apache2 -y
apache2 ya está en su versión más reciente (2.4.66-1~deb13u1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
root@debian:/etc/apache2#
```

3. sudo systemctl start apache2
4. sudo systemctl status apache2

```
root@debian:/etc/apache2# systemctl start apache2
root@debian:/etc/apache2# 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 2026-01-11 09:19:43 CET; 5min ago
 Invocation: 55028429369d45eaa396e881ddcc229a
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 1034 (apache2)
      Tasks: 55 (limit: 4537)
     Memory: 7.7M (peak: 8.2M)
        CPU: 76ms
     CGroup: /system.slice/apache2.service
             ├─1034 /usr/sbin/apache2 -k start
             ├─1037 /usr/sbin/apache2 -k start
             └─1038 /usr/sbin/apache2 -k start

ene 11 09:19:43 debian systemd[1]: Starting apache2.service - The Apache HTTP Server...
ene 11 09:19:43 debian apachectl[980]: AH00558: apache2: Could not reliably determine the server's ful
ene 11 09:19:43 debian systemd[1]: Started apache2.service - The Apache HTTP Server.
root@debian:/etc/apache2#
```

Verificación:

El servicio se inicia correctamente y permanece activo tras reinicios.
Se ha verificado que los puertos **80 (HTTP)** y **443 (HTTPS)** están abiertos en el **grupo de seguridad de AWS**.

---

2. Proxy inverso como punto de entrada del servicio

Se deberá configurar un **servidor proxy en modo inverso** que actúe como punto de entrada al servicio web del proyecto.

El proxy inverso deberá recibir las peticiones de los clientes y reenviarlas al/los servidor/es backend, de acuerdo con la arquitectura adoptada por el equipo.

El equipo deberá:

- configurar el proxy inverso de forma funcional,
- justificar brevemente la utilidad del proxy inverso dentro del contexto del proyecto (publicación del servicio, organización de accesos, control de entrada u otros),
- realizar pruebas de acceso al servicio a través del proxy inverso y aportar evidencias verificables.

Se han configurado dos sitios virtuales para simular distintas áreas de la empresa:

- **valles.ddns.net**: Pagina web publica.
- **Valles.ddns.net/mipagina**: Zona administrativa y restringida.

Estructura de directorios:

Esta esta dividida en 2. Una para la pagina de cara al público en /var/www/html/wordpres y la otra con un alias en /var/www/mipagina.

Para listar los directorios vamos a usar el comando tree; por lo cual primero vamos a instalarlo con sudo apt install tree .

Una vez instalado vamos a usar el comando tree -L 3 (directorio). -L 3 sirve para indicar la profundidad del comando.

```
root@debian:/var/www/html# tree -L 3 /var/www/
/var/www/
├── html
│   ├── index.html
│   └── wordpress
│       ├── CPbase
│       ├── index.php
│       ├── license.txt
│       ├── readme.html
│       ├── wp-activate.php
│       ├── wp-admin
│       ├── wp-blog-header.php
│       ├── wp-comments-post.php
│       ├── wp-config.php
│       ├── wp-config-sample.php
│       ├── wp-content
│       ├── wp-cron.php
│       ├── wp-includes
│       ├── wp-links-opml.php
│       ├── wp-load.php
│       ├── wp-login.php
│       ├── wp-mail.php
│       ├── wp-settings.php
│       ├── wp-signup.php
│       ├── wp-trackback.php
│       └── xmlrpc.php
└── mipagina
    ├── index.html
    └── restringida
        └── index.html

9 directories, 20 files
root@debian:/var/www/html#
```

Archivos de configuración:

/etc/apache2/sites-available/destileria.conf

```
<VirtualHost *:80>

#Configuracion basica del servidor
    ServerName valles.ddns.net
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/wordpress
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm

    <Directory /var/www/html/wordpress>
        AllowOverride All
    </Directory>


#Utilizacion de alias
    Alias "/mipagina" "/var/www/mipagina"

    <Directory "/var/www/mipagina">
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>
```

Activación:

bash

1. sudo a2ensite ConfFinal.conf

```
root@ip-10-0-2-31:/home/ubuntu# sudo a2ensite ConfFinal.conf
Site ConfFinal already enabled
root@ip-10-0-2-31:/home/ubuntu#
```

2. sudo a2dissite 000-default.conf

```
root@ip-10-0-2-31:/home/ubuntu# sudo a2dissite 000-default.conf
Site 000-default already disabled
root@ip-10-0-2-31:/home/ubuntu#
```

3. sudo systemctl reload apache2

```
ubuntu@ip-10-0-2-31:~$ sudo systemctl reload apache2
ubuntu@ip-10-0-2-31:~$
```

1.1.1 Creacion del proxy y balanceo de cargas

Proxys utilizados

Debido a distintos problemas hemos utilizado y conseguido que funcionen 3 proxys diferentes.
2 de ellos son servidores webs hechos proxy (Apache y Nginx) y el otro es un proxy directamente (Caddy).

Configuración

- **Caddy**

Este es el más fácil de configurar de los 3. Primero instalamos Caddy con ***sudo apt install Caddy***.

```
root@debian:/home/usuario# apt install caddy
Installing:
  caddy

Installing dependencies:
  libnspr4  libnss3  libnss3-tools

Summary:
  Upgrading: 0, Installing: 4, Removing: 0, Not Upgrading: 0
  Download size: 12,7 MB
  Space needed: 50,6 MB / 13,8 GB available

Continue? [S/n] s
Des:1 http://deb.debian.org/debian trixie/main amd64 libnspr4 amd64 2:4.36-1 [110 kB]
Des:2 http://deb.debian.org/debian trixie/main amd64 libnss3 amd64 2:3.110-1 [1.393 kB]
Des:3 http://deb.debian.org/debian trixie/main amd64 libnss3-tools amd64 2:3.110-1 [1.086 kB]
Des:4 http://deb.debian.org/debian trixie/main amd64 caddy amd64 2.6.2-12+b3 [10,1 MB]
Descargados 12,7 MB en 1s (10,2 MB/s)
Seleccionando el paquete libnspr4:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 72932 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../libnspr4_2%3a4.36-1_amd64.deb ...
Desempaquetando libnspr4:amd64 (2:4.36-1) ...
Seleccionando el paquete libnss3:amd64 previamente no seleccionado.
Preparando para desempaquetar .../libnss3_2%3a3.110-1_amd64.deb ...
Desempaquetando libnss3:amd64 (2:3.110-1) ...
Seleccionando el paquete libnss3-tools previamente no seleccionado.
Preparando para desempaquetar .../libnss3-tools_2%3a3.110-1_amd64.deb ...
Desempaquetando libnss3-tools (2:3.110-1) ...
Seleccionando el paquete caddy previamente no seleccionado.
Preparando para desempaquetar .../caddy_2.6.2-12+b3_amd64.deb ...
Desempaquetando caddy (2.6.2-12+b3) ...
Configurando libnspr4:amd64 (2:4.36-1) ...
Configurando libnss3:amd64 (2:3.110-1) ...
Configurando libnss3-tools (2:3.110-1) ...
Configurando caddy (2.6.2-12+b3) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/caddy.service' → '/usr/lib/systemd/system/caddy.service'.
Progreso: [ 88%] [
```

Una vez instalado modificaremos el fichero /etc/Caddy/Caddyfile.

Allí pondremos dentro de las llaves reverse_proxy IP/IPs

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 2
  GNU nano 8.4
:80 {
        reverse_proxy 10.0.2.31 10.0.2.48
}
```

Los 2 puntos ochenta significa que se usara http. Ya que Caddy de base fuerza a usar https.

Si pones solo una IP de redirigirá al servidor; pero si pones más de una está permitir a que el proxy sea también balanceador de cargas.

Por último, reiniciamos el servicio y con eso ya funciona.

- **Apache**

  Paso 1: Habilitar los módulos necesarios

Ejecuta:

```
root@debian:/etc/caddy# sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod lbmethod_byrequests
sudo a2enmod slotmem_shm
Enabling module proxy.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

Reinicia Apache para aplicar los cambios:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 3

root@debian:/etc/caddy# systemctl reload apache2
root@debian:/etc/caddy#
```

Paso 2: Crear un archivo de configuración del sitio

Editar el fichero de apache para que funcione como proxy

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 3

 GNU nano 8.4                                                    000-
<VirtualHost *:80>
    ServerName valles.ddns.net

    # Definir el grupo de servidores backend (balanceador)
    <Proxy "balancer://servidores">
        BalancerMember http://10.0.2.31:80
        BalancerMember http://10.0.2.106:80
    </Proxy>

    # Enviar todo el tráfico al balanceador
    ProxyPass "/" "balancer://servidores/"
    ProxyPassReverse "/" "balancer://servidores/"

    # Opcional: panel de estado del balanceador (solo para admins)
    <Location "/balancer-manager">
        SetHandler balancer-manager
        Require local
    </Location>
</VirtualHost>
```

Paso 3: Recargamos el servicio.

```
root@debian:/etc/apache2/sites-available# systemctl reload apache2
root@debian:/etc/apache2/sites-available#
```

- **Nginx**

Paso 1: instalación y crear un archivo de configuración del sitio

Instalamos nginx con ***sudo apt install nginx***

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 3

root@debian:/etc/apache2/sites-available# sudo apt install nginx
Installing:
  nginx

Installing dependencies:
  nginx-common

Paquetes sugeridos:
  fcgiwrap  nginx-doc

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
  Download size: 717 kB
  Space needed: 1.891 kB / 13,8 GB available

Continue? [S/n] s
Des:1 http://deb.debian.org/debian trixie/main amd64 nginx-common all 1.26.3-3+deb13u1 [109 kB]
Des:2 http://deb.debian.org/debian trixie/main amd64 nginx amd64 1.26.3-3+deb13u1 [609 kB]
Descargados 717 kB en 0s (5.605 kB/s)
Preconfigurando paquetes ...
Seleccionando el paquete nginx-common previamente no seleccionado.
(Leyendo la base de datos ... 73038 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../nginx-common_1.26.3-3+deb13u1_all.deb ...
Desempaquetando nginx-common (1.26.3-3+deb13u1) ...
Seleccionando el paquete nginx previamente no seleccionado.
Preparando para desempaquetar .../nginx_1.26.3-3+deb13u1_amd64.deb ...
Desempaquetando nginx (1.26.3-3+deb13u1) ...
Configurando nginx-common (1.26.3-3+deb13u1) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/nginx.service' → '/usr/lib/systemd/system/nginx.service'.

Progreso: [ 56%] [
```

Configuramos el fichero de */etc/nginx/sites-available*

```
  GNU nano 8.4
upstream mis_apps {
    server 10.0.2.31:80;
    server 10.0.2.106:80;
    # Método de balanceo por defecto: round-robin
}

server {
    listen 80;
    server_name valles.ddns.net;

    location / {
        proxy_pass http://valles.ddns.net;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    }
}
```

Paso 2: Recargamos el servicio

```
root@debian:/etc/nginx/sites-available# sudo systemctl reload nginx
root@debian:/etc/nginx/sites-available#
```

1.1.2 Autenticación básica en zona administrativa

Se ha protegido el sitio admin.destileria.local mediante autenticación HTTP básica.

Pasos:

1. sudo apt install apache2-utils -y

```
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$ sudo apt install apache2-utils -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2-utils is already the newest version (2.4.58-1ubuntu8.8).
apache2-utils set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 46 not upgraded.
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$
```

2. sudo htpasswd -c /etc/apache2/.htpasswd usuario1.
    a. Importante una vez que has creado el primer usuario para crear el siguiente hay que quitar el -c.

```
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$ sudo htpasswd -c /etc/apache2/.htpasswd usuario1
New password:
Re-type new password:
Adding password for user usuario1
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$ sudo htpasswd  /etc/apache2/.htpasswd usuario2
New password:
Re-type new password:
Adding password for user usuario2
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$
```

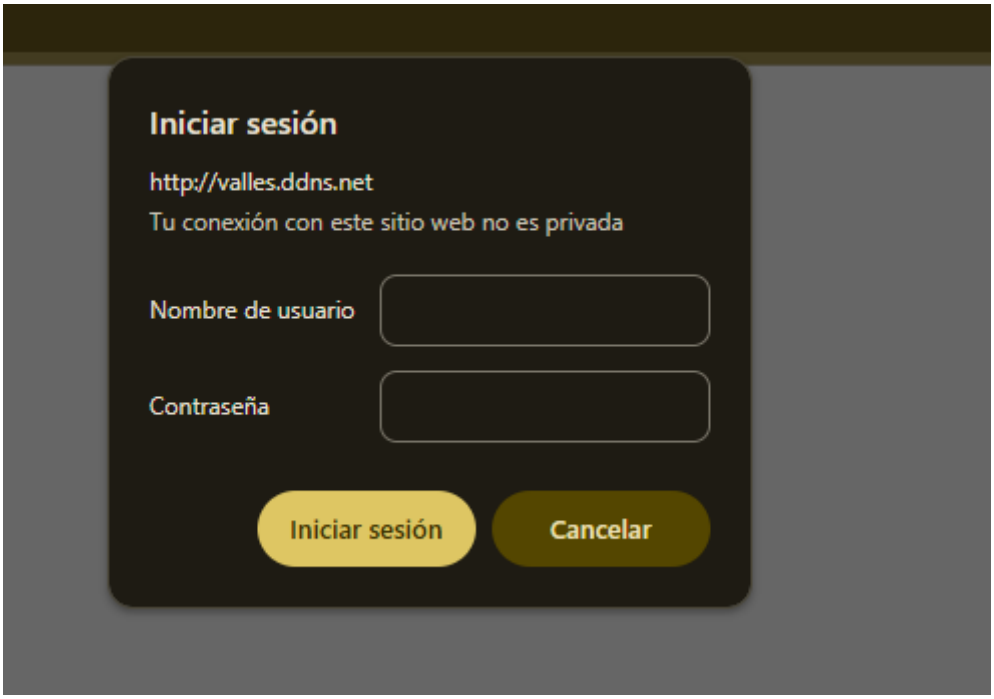
Configuración añadida en admin.destileria.conf:

```
#Autenticacion basica
    <Directory "/var/www/mipagina">
    AuthType Basic
    AuthName "Área restringida"
    AuthUserFile /etc/apache2/.htpasswd
    Require valid-user
    </Directory>
```

1. sudo systemctl reload apache2

```
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$ sudo systemctl reload apache2
ubuntu@ip-10-0-2-31:/etc/apache2/sites-available$
```

Comprobación:



(Criterio evaluado: RA5 – h)

---

3. Balanceador de carga y proxy inverso con certificados

Se deberá implantar un componente de entrada que actúe como **balanceador de carga y proxy inverso**, situado a la entrada de la red interna/servicio, con capacidad para:

- gestionar **certificados digitales** para el acceso seguro,
- repartir conexiones entre los servidores del clúster (o entre varios clústeres, si el diseño lo contempla),
- mantener el servicio operativo cuando alguno de los servidores backend no esté disponible.

El equipo deberá:

- definir la estrategia de balanceo y justificar su elección (por ejemplo, reparto simple, comprobación de estado, criterio de distribución, etc.),
- configurar el acceso seguro mediante certificados, evidenciando la correcta aplicación de HTTPS/TLS según el diseño,
- demostrar mediante pruebas que las conexiones se distribuyen entre los nodos backend y que la solución tolera la caída de un servidor sin pérdida del servicio.

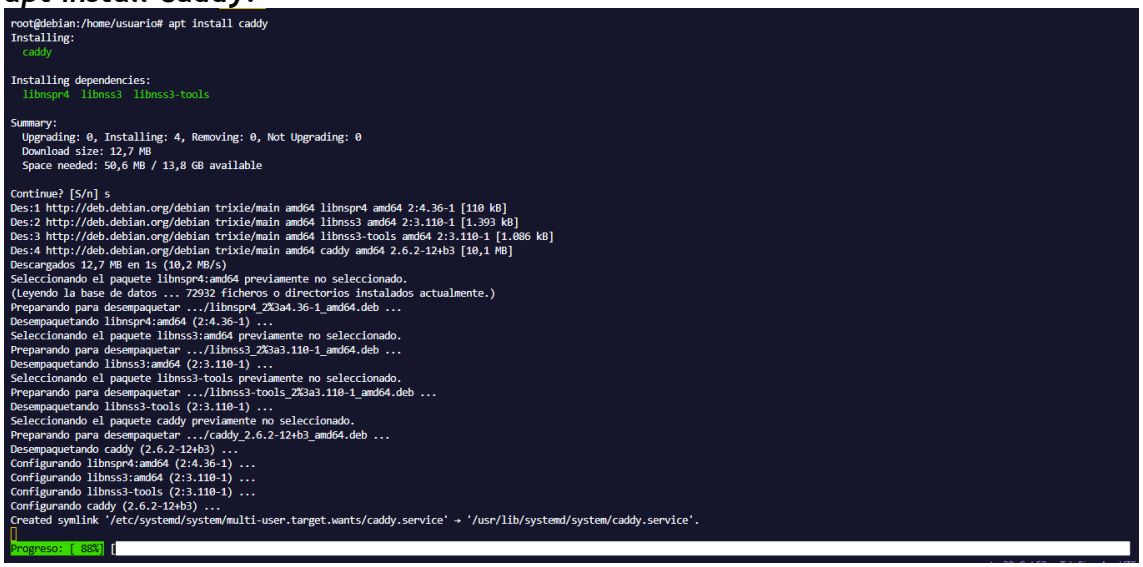
Proxys utilizados

Debido a distintos problemas hemos utilizado y conseguido que funcionen 3 proxys diferentes.
2 de ellos son servidores webs hechos proxy (Apache y Nginx) y el otro es un proxy directamente (Caddy).

Configuración

Caddy

Este es el más fácil de configurar de los 3. Primero instalamos Caddy con ***sudo apt install Caddy***.



Una vez instalado modificaremos el fichero /etc/Caddy/Caddyfile.
Allí pondremos dentro de las llaves reverse_proxy IP/IPs

Los 2 puntos ochenta significa que se usara http. Ya que Caddy de base fuerza a

usar https.
Si pones solo una IP de redirigirá al servidor; pero si pones más de una está permitir a que el proxy sea también balanceador de cargas.
Por último, reiniciamos el servicio y con eso ya funciona.

Apache

Paso 1: Habilitar los módulos necesarios

Ejecuta:

```
root@debian:/etc/caddy# sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod lbmethod_byrequests
sudo a2enmod slotmem_shm
Enabling module proxy.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

Reinicia Apache para aplicar los cambios:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 3

root@debian:/etc/caddy# systemctl reload apache2
root@debian:/etc/caddy#
```

Paso 2: Crear un archivo de configuración del sitio

Editar el fichero de apache para que funcione como proxy

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 3

 GNU nano 8.4                                                    000-
<VirtualHost *:80>
    ServerName valles.ddns.net

    # Definir el grupo de servidores backend (balanceador)
    <Proxy "balancer://servidores">
        BalancerMember http://10.0.2.31:80
        BalancerMember http://10.0.2.106:80
    </Proxy>

    # Enviar todo el tráfico al balanceador
    ProxyPass "/" "balancer://servidores/"
    ProxyPassReverse "/" "balancer://servidores/"

    # Opcional: panel de estado del balanceador (solo para admins)
    <Location "/balancer-manager">
        SetHandler balancer-manager
        Require local
    </Location>
</VirtualHost>
```

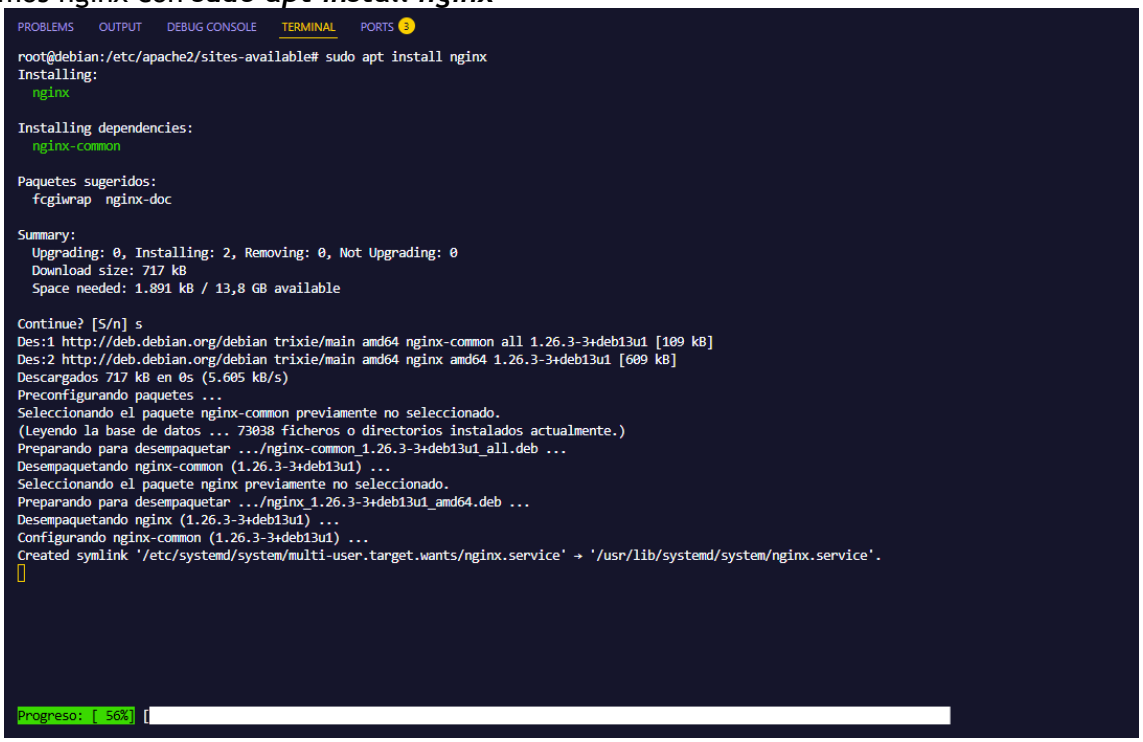
Paso 3: Recargamos el servicio.

```
root@debian:/etc/apache2/sites-available# systemctl reload apache2
root@debian:/etc/apache2/sites-available#
```

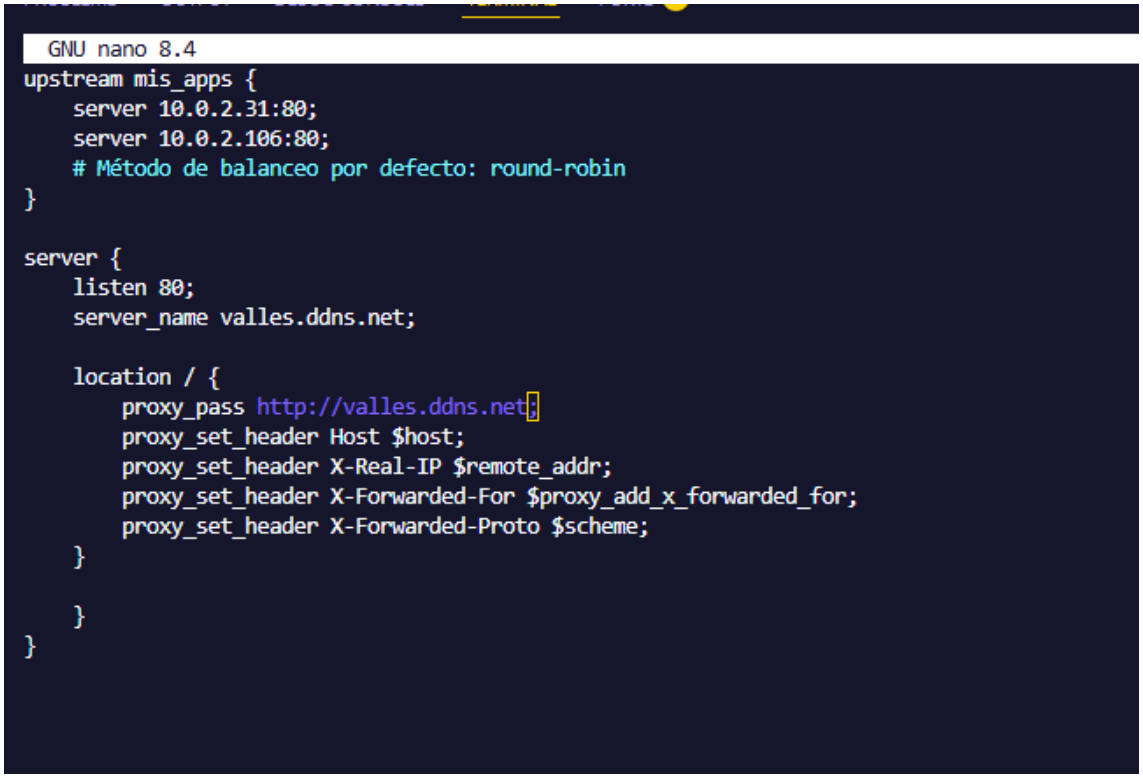
Nginx

Paso 1: instalación y crear un archivo de configuración del sitio

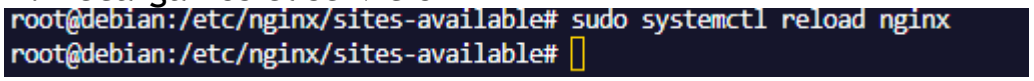
Instalamos nginx con ***sudo apt install nginx***



Configuramos el fichero de */etc/nginx/sites-available*



Paso 2: Recargamos el servicio



(Criterio evaluado: RA3 – e, f, g / RA6 – e)

---

4. Integración con el proyecto intermodular

El servicio web desplegado en AWS deberá:

- Estar documentado e integrarse con el resto de los módulos del proyecto.
- Estar alineado con el contexto de la empresa ficticia.
- Presentar una solución realista, funcional y coherente con un entorno cloud profesional.

(Criterio evaluado: RA5 – i)

HTTPS y TLS en Nginx

Nginx no gestiona automáticamente los certificados. Debes obtenerlos y configurarlos tú.

1. Obtención de certificados:
Usa Certbot exactamente igual que en apache:

```
root@debian:~# sudo apt install certbot python3-certbot-nginx
certbot ya está en su versión más reciente (4.0.0-2).
Installing:
  python3-certbot-nginx

Installing dependencies:
  python3-pyparsing

Paquetes sugeridos:
  python-certbot-nginx-doc  python-pyparsing-doc

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
  Download size: 212 kB
  Space needed: 865 kB / 13,6 GB available

Des:1 http://deb.debian.org/debian trixie/main amd64 python3-pyparsing all 3.1.2-1 [146 kB]
Des:2 http://deb.debian.org/debian trixie/main amd64 python3-certbot-nginx all 4.0.0-2 [65,7 kB]
Descargados 212 kB en 0s (2.309 kB/s)
Seleccionando el paquete python3-pyparsing previamente no seleccionado.
(Leyendo la base de datos ... 74899 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../python3-pyparsing_3.1.2-1_all.deb ...
Desempaquetando python3-pyparsing (3.1.2-1) ...
Seleccionando el paquete python3-certbot-nginx previamente no seleccionado.
Preparando para desempaquetar .../python3-certbot-nginx_4.0.0-2_all.deb ...
Desempaquetando python3-certbot-nginx (4.0.0-2) ...
Configurando python3-pyparsing (3.1.2-1) ...
Configurando python3-certbot-nginx (4.0.0-2) ...
root@debian:~#
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
root@debian:~# sudo certbot --nginx -d valles.ddns.net
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for valles.ddns.net
```

Esto:

- **Obtiene el certificado de Let's Encrypt.**
- **Configura automáticamente el servidor para HTTPS.**
- **Forza redirección HTTP → HTTPS.**

Almacenamiento de Certificados y Claves
Los certificados generados por Certbot se almacenan en /etc/letsencrypt/live/valles.ddns.net/:

```
/etc/letsencrypt/live/valles.dns.net/
├── cert.pem          # Certificado público
├── chain.pem         # Cadena de confianza
├── fullchain.pem     # Certificado + cadena
└── privkey.pem       # Clave privada
```

Ubicación por defecto:

- **Configuración principal: /etc/nginx/nginx.conf**
- **Servidores: /etc/nginx/sites-available/**
- **Certificados: /etc/letsencrypt/live/valles.dns.net/**

Forzar a los usuarios a usar HTTPS
Con Certbot, esto se hace automáticamente.

Monitorización del servicio web
Se ha implementado un script de monitorización ejecutado cada 5 minutos mediante cron.
Script: ~/check_apache.sh

```
ubuntu@ip-10-0-2-31:~$ cat check_apache.sh
#!/bin/bash
if systemctl is-active --quiet apache2; then
    echo "$(date): Apache OK" >> /var/log/apache_monitor.log
else
    echo "$(date): Apache CAÍDO" >> /var/log/apache_monitor.log
    sudo systemctl restart apache2
fi
ubuntu@ip-10-0-2-31:~$
```

Cron:
Utilizamos el cron para programar él una ejecución automática del script de monitorización

```
ubuntu@ip-10-0-2-31:~$ (crontab -l 2>/dev/null; echo "*/5 * * * * /home/ubuntu/check_apache.sh") | crontab -
ubuntu@ip-10-0-2-31:~$
```

Comprobación:
Vamos al fichero de /var/log. Allí aparecerá el fichero apache_monitor.log

```
root@ip-10-0-2-31:/home/ubuntu# ./check_apache.sh
root@ip-10-0-2-31:/home/ubuntu# ls /var/log/
README              apache_monitor.log  auth.log.2.gz          cloud-init.log    dmesg.1.gz  dpkg.log.1  landscape          private     unattended-upgrades
alternatives.log    apport.log          btmp                   cloud-init.log.1  dmesg.2.gz  journal     lastlog            syslog      wtmp
alternatives.log.1  apt                 btmp.1                 dist-upgrade      dmesg.3.gz  kern.log    php8.3-fpm.log     syslog.1
amazon              auth.log            chrony                 dmesg             dmesg.4.gz  kern.log.1  php8.3-fpm.log.1   syslog.2.gz
apache2             auth.log.1          cloud-init-output.log  dmesg.0           dpkg.log    kern.log.2.gz  php8.3-fpm.log.2.gz  sysstat
root@ip-10-0-2-31:/home/ubuntu# cat /var/log/apache_monitor.log
Sun Jan 11 09:41:48 UTC 2026: Apache OK
root@ip-10-0-2-31:/home/ubuntu#
```

Análisis de registros (logs)
Los logs se almacenan en /var/log/apache2/. Se han realizado análisis básicos:
Comandos usados:
 # IPs más frecuentes

```
root@ip-10-0-2-31:/home/ubuntu# awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr | head -5
    257 10.0.1.82
      1 ::1
root@ip-10-0-2-31:/home/ubuntu#
```

 # Errores 404

```
root@ip-10-0-2-31:/home/ubuntu# grep " 404 " /var/log/apache2/error.log
root@ip-10-0-2-31:/home/ubuntu#
```

 # Tráfico del día

```
root@ip-10-0-2-31:/home/ubuntu# grep "$(date '+%d/%b/%Y')" /var/log/apache2/access.log | wc -l
258
root@ip-10-0-2-31:/home/ubuntu#
```

Conclusiones:
“No se detectaron accesos maliciosos,errores, ni intentos de explotación. El tráfico es coherente con un entorno de prueba controlado.”

3. NOTA ACLARATORIA: sobre el concepto de *clúster* y *alta disponibilidad* en el Proyecto Intermodular

En el contexto del Proyecto Intermodular del ciclo de **Administración de Sistemas Informáticos en Red (ASIR)**, y en particular en las tareas asociadas al módulo de **Seguridad y Alta Disponibilidad**, es importante aclarar el alcance de los términos *clúster* y *alta disponibilidad* utilizados en los enunciados.

Sobre el concepto de clúster

A efectos de este proyecto, se considera **clúster de servidores** a un conjunto de **dos o más servidores** que prestan **el mismo servicio** (por ejemplo, un servicio web), de manera que el servicio **no dependa de un único servidor** para su funcionamiento.

No se exige la implantación de soluciones de clúster avanzadas basadas en software específico de conmutación (como sistemas de clúster con estado compartido o direcciones IP flotantes). Es suficiente con que:

- existan varios servidores con el mismo rol,
- todos ellos sean capaces de prestar el servicio de forma independiente,
- y pueda demostrarse que, ante la caída de uno de los servidores, el servicio puede seguir siendo ofrecido por otro nodo del clúster.

Sobre el concepto de alta disponibilidad

En este proyecto, la **alta disponibilidad** se alcanza de forma progresiva y está directamente relacionada con la **arquitectura global** del sistema:

- En una primera fase, la continuidad del servicio se garantiza mediante la **redundancia de servidores** (existencia de varios nodos con el mismo servicio).
- En una fase posterior, la **gestión automática de la disponibilidad** se realiza mediante un **proxy inverso y/o balanceador de carga**, que actúa como punto de entrada al servicio y se encarga de distribuir las peticiones entre los distintos nodos disponibles.

Por tanto, la **alta disponibilidad real desde el punto de vista del usuario** no recae en los servidores del clúster, sino en el **componente de entrada (proxy/balanceador)**, que permite:

- repartir las conexiones entre varios servidores,
- dejar de enviar tráfico a un servidor que no esté disponible,
- y mantener el acceso al servicio sin intervención del usuario final.

Consideraciones importantes

- El proyecto **no exige** la implantación de soluciones de clúster complejas o de nivel empresarial.
- El objetivo es comprender y aplicar los **principios básicos de redundancia, balanceo y entrada segura al servicio**, utilizando una arquitectura realista y coherente.
- Se valorará especialmente la **comprensión del diseño**, la **justificación de las decisiones adoptadas** y la **demostración práctica** del funcionamiento del sistema ante escenarios de fallo.

4. EVALUACIÓN: Lista de cotejo para evaluación del proyecto en SAD

Escala de valoración

- **0 – No alcanzado:** No se cumple el criterio o se hace de forma incorrecta.
- **1 – Básico:** Se cumple de forma mínima o incompleta.
- **2 – Adecuado:** Se cumple correctamente.
- **3 – Avanzado:** Se cumple de forma completa, bien integrada y justificada

Tareas	Peso	Nivel 0 – No alcanzado	Nivel 1 – Básico	Nivel 2 – Adecuado	Nivel 3 – Avanzado	Evidencias mínimas a capturar
1. Servidor redundante / clúster de servidores	**2G,5%**	No hay redundancia: existe un único servidor o el secundario no presta el servicio.	Se crea un segundo servidor, pero no está alineado (contenido/configuración) o no se demuestra continuidad.	Redundancia operativa: al menos dos nodos funcionales y prueba clara de continuidad ante caída de uno.	Clúster bien planteado e integrado: nodos coherentes, pruebas completas de fallo/recuperación y explicación clara del diseño.	Diagrama simple del clúster, evidencias de servicio funcionando en cada nodo (capturas/curl), prueba de caída (servicio sigue disponible), configuración o inventario de recursos implicados.

Tareas	Peso	Nivel 0 – No alcanzado	Nivel 1 – Básico	Nivel 2 – Adecuado	Nivel 3 – Avanzado	Evidencias mínimas a capturar
2.Proxy inverso	**16,8%**	No existe proxy inverso o no funciona (no enruta / error).	Proxy inverso configurado de forma mínima, con pruebas incompletas o configuración poco coherente.	Proxy inverso funcional: enruta correctamente al backend y se aportan pruebas verificables.	Proxy inverso robusto: configuración clara, bien integrada con el servicio del proyecto y evidencias de control/observabilidad (logs/monitorización) y decisiones justificadas.	Fragmentos/capturas de configuración del proxy, pruebas de acceso pasando por el proxy (curl/cabeceras/URL), evidencia de logs del proxy, explicación breve del rol del proxy en la arquitectura.
3.Balanceador de carga + proxy inverso con certificados	**45,2%**	No hay balanceo o no gestiona certificados; el servicio no cumple la función de entrada.	Balanceo o TLS parcial: reparte de forma limitada o el certificado no está bien gestionado; tolerancia a fallos no demostrada.	Balanceador + proxy inverso operativo: reparte conexiones entre nodos y gestiona certificados correctamente; prueba de tolerancia a fallos superada.	Solución avanzada: balanceo coherente (health checks o equivalente), TLS bien aplicado (buenas prácticas), alta disponibilidad demostrada con pruebas repetibles y explicación/justificación del diseño.	Evidencia de reparto de carga (peticiones repetidas mostrando alternancia o distribución), evidencia de HTTPS/TLS (info de certificado/validación), prueba de caída de un backend manteniendo servicio, configuración relevante del balanceo y del TLS.

Tareas	Peso	Nivel 0 – No alcanzado	Nivel 1 – Básico	Nivel 2 – Adecuado	Nivel 3 – Avanzado	Evidencias mínimas a capturar
4. Integración con el proyecto intermodular	**8,5%**	El sistema no se documenta o no se integra con el resto de los módulos del proyecto, no guarda relación con el contexto de la empresa ficticia o la solución propuesta es incoherente, incompleta o no funcional en un entorno cloud.	El sistema está mínimamente documentado o muestra una integración mínima o parcial con otros módulos del proyecto. La relación con la empresa ficticia es superficial y la solución presenta limitaciones de realismo o coherencia en un entorno cloud profesional.	El sistema está correctamente documentado y se integra correctamente con el resto de los módulos del proyecto, está alineado con el contexto de la empresa ficticia y ofrece una solución funcional, realista y coherente con un entorno cloud profesional.	El sistema está ampliamente documentado y se integra de forma completa y consistente con todos los módulos del proyecto, está claramente contextualizado en la empresa ficticia y presenta una solución profesional, bien justificada, realista y plenamente coherente con un entorno cloud real.	Documento o apartado explicativo donde se describa la integración con el resto de los módulos del proyecto. Referencia clara al contexto de la empresa ficticia Capturas de pantalla o enlaces que demuestre que el servicio web está desplegado y es accesible en AWS. Descripción funcional breve del sistema