MEMORIA TÉCNICA FINAL: AMEGO HOSTING

Administración de Sistemas Informáticos en Red (ASIX)

ÍNDICE

1. Introducción y Justificación del Proyecto

Hoy en día todo está prácticamente basado en internet, y si una empresa o un centro educativo no tiene una buena presencia online, es como si no existiera.

El problema es que las soluciones más usadas, como AWS o Azure, aunque son muy potentes, también son caras y bastante complicadas de gestionar si no eres una empresa grande.

Por eso nosotros hemos creado *AMEGO Hostings,* un proyecto que intenta simular una infraestructura de hosting real, pero montada en local.

La idea no es competir con estas plataformas, sino entender cómo funcionan por dentro y demostrar que se puede montar algo parecido con recursos más básicos, sin depender de servicios externos.

1.1. Contexto y Origen del Proyecto: AMEGO Hosting

1.1. Contexto y Origen del Proyecto: AMEGO Hosting

En el entorno actual, casi todo depende de la nube: páginas web, bases de datos, aplicaciones… todo está externalizado.

Esto tiene ventajas, pero también un problema bastante claro: dependes totalmente de otras empresas, pagas cuotas constantes muy elevadas y pierdes control sobre dónde están tus datos y cómo se gestiona la infraestructura.

Nosotros vimos que muchas veces se usa el “cloud” porque parece la única opción profesional, cuando en realidad detrás hay todo un sistema que se puede entender y replicar a pequeña escala.

También nos dimos cuenta de que normalmente no se enseña cómo funciona realmente un servidor o un centro de datos, porque todo viene ya montado y automatizado.

A partir de ahí nace AMEGO Hosting. El proyecto surge con la idea de demostrar que se puede montar un entorno tipo centro de datos funcional, seguro y automatizado usando únicamente recursos locales y Software Libre.

La intención no era solo montar un servidor, sino entender todo el sistema completo como si fuéramos un proveedor de internet, pero dentro de VirtualBox.

1.2. Análisis del Problema y Solución Propuesta

Durante la planificación estuvimos viendo los problemas más típicos que tienen las pequeñas empresas cuando contratan hosting.

Lo primero que detectamos fue el problema de la gestión. Muchos paneles de control están llenos de opciones que no se usan, y para alguien que no es técnico puede ser bastante confuso hacer cosas básicas.

Otro punto importante es la seguridad. En muchos hostings tradicionales varios usuarios comparten parte de la misma infraestructura, lo que puede generar riesgos si algo falla o si hay vulnerabilidades en el sistema compartido.

Para intentar solucionar todo esto utilizamos contenedores rootless con Podman para que cada servicio, como WordPress, funcione de manera independiente y no afecte al resto si algo falla.

Por último, creamos una automatización con PHP y Bash para simplificar la gestión. La idea es que el usuario no tenga que estar constantemente usando la terminal de Linux, sino que tenga un panel mucho más simple y directo para gestionar sus servicios.

1.3. Filosofía de Trabajo: El Método A.M.E.G.O.

Para darle un poco de orden a todo el proyecto, nosotros hemos creado una especie de “método propio” que resume cómo hemos montado toda la infraestructura. No es algo teórico inventado sin más, sino más bien la forma en la que hemos ido construyendo y organizando el sistema mientras lo hacíamos.

Este método se basa en cinco ideas principales:

Alta Disponibilidad (A)
Nosotros no hemos planteado el sistema como algo que pueda caerse fácilmente y ya está. La idea es que, si un servidor falla, haya forma de recuperarlo rápido.

Para eso hemos usado clones y snapshots en VirtualBox, de manera que si algo va mal en los servidores S3 o S4, se puedan restaurar o poner en funcionamiento en muy poco tiempo.

Mantenimiento Continuo (M)
También hemos intentado que el sistema no sea un “montaje puntual” que luego no se toca, sino algo que se pueda revisar y mantener fácilmente y para eso hemos organizado bien los logs y la estructura de carpetas, y hemos preparado el uso de imágenes de Podman de forma que se puedan actualizar sin romper los datos importantes.

Escalabilidad (E)
Otra parte importante es que el sistema pueda crecer sin tener que rehacerlo todo. Gracias

a la arquitectura de microservicios, podemos añadir más contenedores cuando haga falta. Además, el uso de NGINX como proxy inverso hace que este crecimiento sea bastante transparente para el usuario, sin que note cambios en cómo accede al servicio.

Gestión de Recursos (G)
También hemos tenido en cuenta que los recursos del sistema son limitados. Por eso hemos intentado controlar bien el uso de RAM y CPU en los servidores S1 y S2. Con herramientas de monitorización vamos vigilando que ningún proceso, especialmente las bases de datos, se coma todos los recursos y deje el sistema colgado.

Optimización (O)
Por último, hemos intentado ajustar todo lo posible para que el sistema vaya lo más rápido y eficiente que se pueda. Por ejemplo, configurando servicios como BIND9 y Postfix para reducir tiempos de respuesta, tanto en la resolución de DNS como en la entrega de correos.

1.4. Objetivos del Proyecto

En este proyecto nosotros hemos dividido los objetivos en tres bloques diferentes para tener claro qué queríamos conseguir en cada parte: la parte técnica, la parte de seguridad y la parte de aprendizaje.

1.4.1. Objetivos Técnicos

En la parte técnica, el objetivo principal ha sido montar una infraestructura que funcione como una red real dentro de un entorno local.

Para ello, hemos desplegado una red local bastante completa con direccionamiento dinámico usando Kea DHCP, de forma que los equipos no tengan que configurarse a mano.

También hemos configurado un servidor DNS con BIND9, creando tanto zonas directas como inversas para poder resolver nombres dentro de la red.

Otro punto importante ha sido la implementación de un sistema de correo completo utilizando Postfix y Dovecot, para simular un servicio de email funcional dentro de la infraestructura.

Además, hemos trabajado con contenedores de WordPress, orquestados mediante Podman, para poder desplegar servicios web de forma más organizada y automatizada.

1.4.2. Objetivos de Seguridad

En la parte de seguridad, lo que hemos intentado es que los servicios no estén todos mezclados ni expuestos de cualquier forma.

Para eso, hemos aislado los servicios más importantes dentro de una red interna privada, evitando accesos innecesarios desde fuera.

También hemos cifrado el tráfico utilizando certificados SSL gestionados a través del proxy inverso, para asegurar las comunicaciones entre el usuario y los servicios.

Por último, hemos aplicado políticas de firewall bastante restrictivas en el nodo S1, controlando qué tráfico puede entrar y salir del sistema.

1.4.3. Objetivos de Aprendizaje

Más allá de lo técnico, este proyecto también tenía una parte importante de aprendizaje.

Por un lado, nos ha servido para consolidar conocimientos de administración de sistemas Linux, especialmente en entornos *Ubuntu Server*, trabajando con servicios reales y configuraciones avanzadas.

Y por otro lado, también hemos aprendido a documentar todo el proceso de forma ordenada y comprensible, cómo se haría en un entorno profesional, donde no solo importa que funcione el sistema, sino también saber explicarlo bien.

1.5. Metodología y Planificación (Gantt)

Para organizar el desarrollo del proyecto nosotros hemos seguido una forma de trabajar bastante práctica, basada en ir construyendo todo por partes y comprobando que cada cosa funcionara antes de seguir avanzando.

No hemos intentado hacerlo todo de golpe, sino que hemos ido montando la infraestructura poco a poco, como si fuera un sistema real en producción.

La idea principal ha sido trabajar de forma progresiva, es decir primero entender qué queríamos montar, después implementarlo y finalmente probarlo y corregirlo.

Esto nos ha permitido no perdernos con la cantidad de servicios que tiene el proyecto y poder detectar errores a tiempo, el único problema que ha aparecido a sido que no podiamos trabajar simultáneamente en la misma maquina o editando los mismos ficheros, lo que hacíamos era cada uno se centra en un servicio o en una tarea en específico en otra máquina de pruebas y a partir de la máquina de pruebas si todo salió correctamente copiar los comandos y la configuración usada para aplicarlo en la máquina final/definitiva.

En general, hemos combinado varias formas de trabajo:

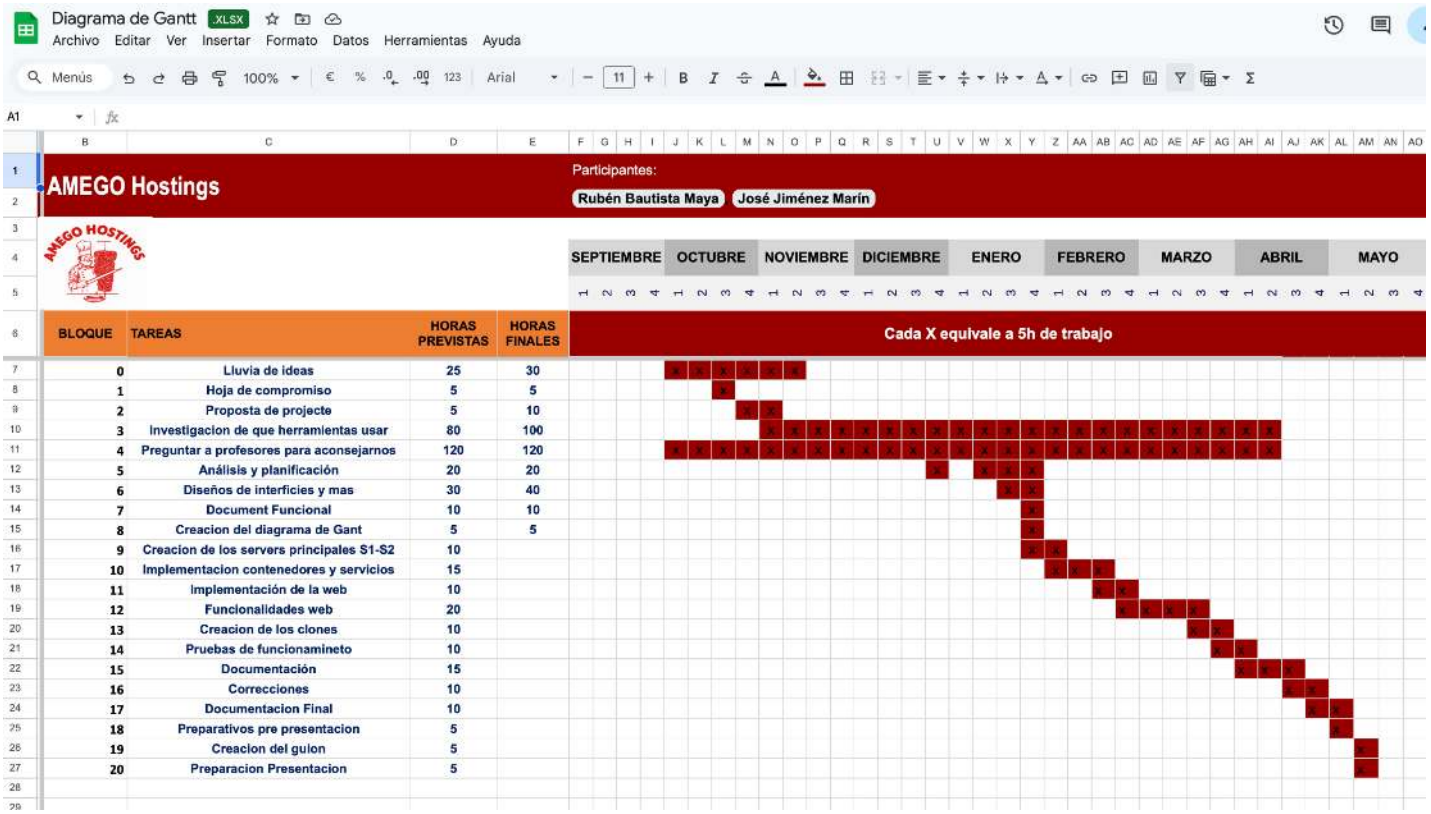
Por un lado, hemos seguido una planificación por fases (como se ve en el diagrama de Gantt), donde cada bloque del proyecto tenía unos objetivos claros.

Esto nos ha ayudado a repartir el trabajo a lo largo del tiempo y no dejarlo todo para el final.

Por otro lado, hemos utilizado un enfoque más iterativo en la parte técnica, es decir, cada vez que montábamos un servicio (DNS, DHCP, contenedores, web, etc.) lo probábamos directamente en el entorno de VirtualBox para ver si funcionaba correctamente antes de continuar con el siguiente.

También hemos trabajado de forma bastante colaborativa, repartiendo tareas entre los miembros del grupo según la parte del sistema (red, servidores, web, documentación…).

Esto ha hecho que el trabajo fuera más fluido y que cada uno pudiera centrarse en una parte concreta del proyecto.



BLOQUE	TAREAS	HORAS PREVISTAS	HORAS FINALES
0	Lluvia de ideas	25	30
1	Hoja de compromiso	5	5
2	Proposta de projecte	5	10
3	Investigacion de que herramientas usar	80	100
4	Preguntar a profesores para aconsejarnos	120	120
5	Análisis y planificación	20	20
6	Diseños de interficies y mas	30	40
7	Document Funcional	10	10
8	Creacion del diagrama de Gant	5	5
9	Creacion de los servers principales S1-S2	10	
10	Implementacion contenedores y servicios	15	
11	Implementación de la web	10	
12	Funcionalidades web	20	
13	Creacion de los clones	10	
14	Pruebas de funcionamineto	10	
15	Documentación	15	
16	Correcciones	10	
17	Documentacion Final	10	
18	Preparativos pre presentacion	5	
19	Creacion del guion	5	
20	Preparacion Presentacion	5	

2. Creación y configuración de máquinas virtuales (VirtualBox)

SUBÍNDICE

2.1 Introducción al sistema de virtualización

En el proyecto *AMEGO Hosting* hemos utilizado *VirtualBox* como herramienta principal de virtualización. Básicamente, lo que nos permite es crear varias máquinas virtuales dentro de un mismo ordenador físico, como si tuviéramos varios servidores reales funcionando a la vez.

Esto es muy útil porque nos da la posibilidad de simular una infraestructura completa sin necesidad de montar hardware físico ni gastar dinero en servidores reales y, es perfecto porque nos deja experimentar sin riesgos.

Gracias a VirtualBox podemos trabajar en un entorno bastante parecido al de un centro de datos real ya que creamos servidores, configuramos redes internas y desplegamos servicios que están completamente aislados entre sí, todo dentro del mismo sistema.

2.2 Motivo de uso en el proyecto

2.2.1 Hemos elegido VirtualBox porque:

Hemos decidido utilizar VirtualBox principalmente por varias razones bastante claras:

- Es una herramienta gratuita y bastante estable, que no requiere licencias ni costes adicionales.
- Ya la hemos utilizado anteriormente durante el ciclo formativo (SMX / ASIX), por lo que ya teníamos experiencia con ella.
- Permite crear redes internas complejas entre máquinas virtuales, algo clave para simular la infraestructura del proyecto.
- Facilita mucho la simulación de un entorno tipo hosting o datacenter real sin necesidad de hardware físico.
- Tiene una interfaz gráfica sencilla, lo que hace que la gestión de las máquinas virtuales sea más cómoda.

Gracias a todo esto, hemos podido recrear una arquitectura bastante similar a la de un proveedor de hosting real, pero dentro de un entorno controlado, seguro y totalmente adaptado a nuestro proyecto.

2.3 Creación de máquinas virtuales

2.3.1 Servidores principales

Hemos creado varias máquinas virtuales, y cada una tiene un papel concreto dentro de la infraestructura. La idea era simular un entorno real de servidores, donde cada máquina se encarga de una parte del sistema.

En concreto, hemos definido los siguientes servidores:

- *S1 → Servidor principal de infraestructura (DNS, DHCP y servicios base de red)*
- *S2 → Servidor de servicios (web, correo y aplicaciones principales)*
- *S3 y S4 → Servidores de backup y redundancia, pensados para recuperación y continuidad del sistema en caso de fallo*

S1
S1.png

S2
s2.png

2.4 Configuración de red en VirtualBox

2.4.1 Tipos de red utilizados

Durante el desarrollo del proyecto hemos trabajado principalmente con dos tipos de configuración de red dentro de VirtualBox, ya que necesitábamos separar bien la comunicación interna del acceso externo.

Por un lado hemos utilizado la *red interna (Internal Network***)**, que es la que hemos usado para la comunicación entre los distintos servidores del sistema. Y por otro lado hemos utilizado el *adaptador puente (Bridge),* que nos permite tener acceso desde el equipo anfitrión cuando ha sido necesario para pruebas o configuración.

Todos los servidores del proyecto están conectados a esta red interna, lo que permite que se comuniquen entre ellos de forma directa, pero sin estar expuestos al exterior.

Esto es importante porque nos ayuda a mantener el sistema aislado y controlado, simulando una red privada real como la que tendría una empresa o un centro de datos.

S1.3.png S1.4.png

2.5 Edición de máquinas virtuales

Durante el desarrollo del proyecto hemos tenido que ir modificando varias veces las máquinas virtuales en función de las necesidades que iban apareciendo con cada servicio.

Esto ha sido algo totalmente normal, ya que el sistema no estaba cerrado desde el principio y ha ido evolucionando a medida que añadíamos nuevas funcionalidades.

Los cambios más habituales que hemos realizado han sido los siguientes:

- Hemos aumentado la memoria RAM en algunas máquinas cuando los servicios lo requerían, sobre todo en momentos donde había más carga o procesos funcionando a la vez.
- Hemos cambiado la configuración de los adaptadores de red en varias ocasiones para poder hacer pruebas distintas y ajustar mejor la comunicación entre servidores.
- Hemos ajustado el número de CPUs asignadas a ciertas máquinas para mejorar el rendimiento en servicios más pesados.
- También hemos tenido que reconfigurar discos y redes internas cuando la estructura del sistema ha cambiado o se ha optimizado.
- En algunos casos hemos actualizado el sistema operativo o reinstalado servicios para corregir errores o mejorar la estabilidad.

En general, estos cambios han formado parte natural del proceso, ya que el proyecto ha ido creciendo poco a poco y cada nueva parte que se añadía requería ajustar la infraestructura para que todo funcionara correctamente.

S1
S1.1.png S1.2.png

S2
s2.1.png s2.2.png

2.6 Clonado de máquinas virtuales

Una parte bastante importante del proyecto ha sido el uso de clones de máquinas virtuales, ya que nos ha permitido ahorrar muchísimo tiempo durante la creación y pruebas de la infraestructura. En lugar de instalar y configurar todo desde cero cada vez, hemos podido duplicar máquinas ya preparadas y adaptarlas según la necesidad.

2.6.1 Tipos de clonación:

Durante el proyecto hemos trabajado con dos tipos de clonación:

- **Clon completo:** es una copia totalmente independiente de la máquina original. Una vez creado, no depende de la máquina base para funcionar, por lo que se puede modificar sin afectar al original.
- **Clon enlazado:** depende de la máquina base, ya que comparte parte de sus archivos. Es más rápido de crear y ocupa menos espacio, pero está ligado a la máquina principal.

2.6.2 Uso en el proyecto:

En nuestro caso hemos utilizado los clones principalmente para crear los servidores *S3 y S4*, que funcionan como servidores de backup dentro de la infraestructura.

La idea es poder simular un sistema de alta disponibilidad, donde si un servidor principal falla, otro pueda entrar en funcionamiento y asumir su papel sin que el servicio se caiga.

Además, el uso de clones nos ha permitido:

- Evitar tener que reinstalar sistemas operativos y servicios desde cero cada vez
- Probar configuraciones nuevas sin riesgo de romper la máquina original
- Mantener una coherencia entre servidores, ya que parten de la misma base y configuración

El único problema ha sido que no hemos podido hacer S3 y S4 hasta que S1 y S2 estuvieran acabados al completo y hasta bastante tarde del proyecto no hemos dispuesto de tener las 4 máquinas creadas.

2.7 Configuración de almacenamiento

Los discos virtuales de todas las máquinas del proyecto se han configurado siguiendo un mismo criterio para mantener la coherencia y optimizar el uso de recursos del equipo físico.

En todos los casos hemos utilizado discos en formato *VDI (VirtualBox Disk Image)*, que es el formato propio de VirtualBox y el más adecuado para este tipo de entorno.

Configuración utilizada

- **Tipo de disco:** VDI (VirtualBox Disk Image)
- **Asignación:** dinámica
- **Tamaño:** expandible según el uso real del sistema

Esta configuración nos ha resultado muy útil durante todo el desarrollo del proyecto por varias razones:

- El disco no ocupa todo el espacio desde el inicio, lo que ahorra almacenamiento en el equipo
- Crece únicamente cuando es necesario, en función del uso real del sistema
- Permite optimizar mejor los recursos del ordenador anfitrión
- Facilita la gestión de varias máquinas virtuales al mismo tiempo sin saturar el sistema

En general, esta forma de configuración ha sido clave para poder trabajar con varias máquinas virtuales de forma eficiente y sin problemas de espacio o rendimiento, que era unas de nuestras grandes inquietudes, el saber si los ordenadores de clase pudieses correr los 4 al mismo tiempo.

2.8 Seguridad y aislamiento

La seguridad ha sido un aspecto bastante importante dentro del diseño del entorno virtual, ya que la idea del proyecto es simular una infraestructura lo más parecida posible a un entorno real de hosting o datacenter.

Para ello hemos aplicado varias medidas básicas de seguridad y aislamiento:

- No todas las máquinas tienen acceso directo a Internet, evitando exposiciones innecesarias
- Se utiliza una red interna para la comunicación entre servidores, manteniendo el tráfico controlado
- Solo algunos servicios concretos tienen permisos para exponer puertos hacia el exterior
- Los roles de cada servidor están separados para evitar dependencias innecesarias entre servicios

En conjunto, esto nos permite simular un entorno real de centro de datos, donde no todos los servidores están accesibles públicamente, sino que existe una estructura interna organizada, segmentada y controlada.

2.9 Pruebas realizadas

Durante la configuración del entorno hemos ido haciendo varias pruebas para asegurarnos de que todo funcionaba como debía y que no había fallos en la red ni en los servicios.

Hemos comprobado, por ejemplo, que los servidores se veían entre ellos usando ping, lo que nos confirmó que la red interna estaba bien montada. También hemos verificado que el DHCP asigna las IPs correctamente, sin conflictos ni errores.

Otra prueba importante ha sido el acceso entre máquinas virtuales a los distintos servicios internos, como web, correo o DNS, para ver que todo respondía bien dentro de la infraestructura.

Además, hemos comprobado que los servidores clonados funcionaban correctamente como sistema de backup, asegurando que podían entrar en funcionamiento si hacía falta.

En general, todas estas pruebas nos han servido para validar que el sistema estaba bien configurado y que todas las máquinas podían comunicarse entre sí dentro de la red interna sin problemas de conectividad.

2.10 Problemas encontrados

Durante el proceso de creación y configuración de las máquinas virtuales no hemos tenido problemas graves, ya que ya teníamos experiencia previa trabajando con VirtualBox en clases de SMX y ASIX.

Aun así, sí que han ido apareciendo algunos pequeños errores típicos de este tipo de entornos al principio del montaje del sistema.

Por ejemplo, en algunas ocasiones las interfaces de red no estaban bien configuradas, lo que hacía que algunas máquinas no se vieran entre ellas dentro de la red interna.

También nos encontramos con casos en los que faltaba memoria RAM o recursos de CPU en algunas máquinas virtuales, lo que provocaba que fueran bastante lentas o poco estables durante las pruebas. Otro problema inicial fueron algunos conflictos de direcciones IP en la red interna, sobre todo en las primeras pruebas del DHCP ya que no teníamos claro que ips asignarles , y gracias a un consejo de Oscar Torrente nos recomendó ponerle una IP con un valor alto a nuestro S1 para que así no hayan problemas de que se le asigne nuestra IP a otra maquina y asi tenerla fija y no tener que editar cada vez que iniciemos la máquina la ip asignada.

Solución

Todos estos problemas se han ido solucionando de forma bastante directa y sin complicaciones importantes:

- Reconfigurando correctamente las redes dentro de VirtualBox

- Ajustando la asignación de recursos (RAM y CPU) según las necesidades de cada servidor
- Revisando la configuración del DHCP para evitar conflictos de IP
- Reiniciando las máquinas virtuales cuando ha sido necesario para aplicar cambios

En general, al estar acostumbrados a trabajar con este tipo de entornos, la resolución de los problemas ha sido rápida y no ha supuesto grandes dificultades durante el desarrollo del proyecto.

2.11 Conclusión

El uso de VirtualBox nos ha permitido montar una infraestructura bastante completa de servidores dentro de un entorno totalmente controlado y sin depender de hardware real.

Gracias a la creación, configuración y clonación de máquinas virtuales hemos podido simular un sistema de hosting bastante realista, con varios servicios separados, una red interna bien definida y una arquitectura que se puede ampliar si hace falta.

Todo esto nos ha servido para entender mejor cómo funcionan los entornos profesionales por dentro, no solo a nivel teórico sino también práctico, y para construir una base sólida sobre la que hemos desarrollado todo el proyecto AMEGO Hosting.

3.NGINX – Configuración de sites-available y sites-enabled

SUBÍNDICE

3.1 Introducción al servicio

3.1.1 Descripción del servicio

En este proyecto hemos utilizado *NGINX* principalmente como ***Proxy inverso***, es decir, como un punto intermedio entre el usuario y los servicios internos que tenemos dentro de la infraestructura.

Básicamente, en vez de que el usuario acceda directamente a cada servicio, NGINX se encarga de recibir las peticiones y redirigirlas al servicio correspondiente dentro del sistema.

Esto nos ha permitido centralizar el acceso a los distintos servicios, mejorar la organización general del proyecto y añadir una capa extra de seguridad dentro de la arquitectura de AMEGO Hosting.

3.1.2 Motivo de uso en el proyecto

Hemos elegido *NGINX* porque encaja muy bien con el tipo de infraestructura que queríamos montar.

Nos permite gestionar varios servicios web desde un único punto de entrada, lo cual es clave en un entorno tipo hosting donde no quieres exponer cada servicio por separado.

Además, nos facilita bastante la configuración de cosas como HTTPS, redirecciones y separación de servicios, por ejemplo entre:

- la página web principal
- el panel de administración
- el webmail

De esta forma, todo queda más ordenado y más fácil de mantener.

3.2 Instalación de NGINX

En Ubuntu Server 24.04, la instalación se realiza de forma directa desde repositorios oficiales:

sudo apt update && sudo apt install nginx -y

3.3 Configuración del servicio

3.3.1 Optimización de headers

proxy_headers_hash_max_size 1024;
proxy_headers_hash_bucket_size 128;

Explicación:
Estas directivas se han utilizado para evitar errores o warnings relacionados con el tamaño de los headers en NGINX. En un entorno como el nuestro, donde se manejan varias redirecciones y servicios a la vez, esto ayuda a que el proxy funcione de forma más estable y sin problemas de configuración.

3.3.2 Redirección HTTP → HTTPS

```
server {
    listen 80;
    server_name amegohostings.es admin.amegohostings.es webmail.amegohostings.es;
    return 301 https://$host$request_uri;
}
```

Explicación:
Hemos configurado una redirección global para que cualquier acceso HTTP sea redirigido automáticamente a HTTPS.

Esto es importante porque en un entorno real de hosting, toda la comunicación debe ir cifrada.

3.3.3 Web principal (HTTPS)

```
server {
    listen 443 ssl;
    server_name amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8082;
        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 https;
    }
}
```

Explicación:
Este bloque gestiona la web principal del proyecto.

Hemos configurado NGINX para que reciba las peticiones HTTPS y las redirija internamente al servidor backend que está en 10.0.1.2:8082.

Esto nos permite separar el frontend del backend y mejorar la seguridad.

3.3.4 Panel de admin

```
server {
    listen 443 ssl;
    server_name admin.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8084;
        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 https;
    }
}
```

Explicación:
Este bloque está destinado al panel de administración.

Lo hemos separado de la web principal para tener un acceso más controlado y simular un entorno real donde los administradores tienen una zona independiente.

3.3.5 Webmail (Roundcube)

```
server {
    listen 443 ssl;
    server_name webmail.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;
```

```
    location / {
        proxy_pass http://10.0.1.2:8081;
        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 https;

        proxy_read_timeout 90;
    }
}
```

Explicación:
Este bloque gestiona el servicio de correo web (Roundcube).

Hemos añadido un timeout mayor porque el acceso al correo puede tardar más que una web normal.

3.4 Activación de sitios (sites-enabled)

En NGINX, los sitios se activan mediante enlaces simbólicos desde sites-available a sites-enabled.

Esto permite tener configuraciones preparadas sin activarlas directamente hasta que estén listas.

Explicación:
Esto nos ayuda a trabajar de forma ordenada, activando o desactivando servicios sin borrar configuraciones.

3.5 Decisiones de diseño

Hemos tomado varias decisiones importantes:

Uso de NGINX como proxy inverso para centralizar accesos
Separación de servicios (web, admin, mail) en subdominios
Uso obligatorio de HTTPS en todo el sistema
Red interna para backend (10.0.1.2) para mayor seguridad
Separación de roles y servicios para simular hosting real

3.6 Pruebas realizadas

Verificar el acceso a:

https://amegohostings.es/

https://admin.amegohostings.es/
https://webmail.amegohostings.es/

3.7 Problemas encontrados

En general no hemos tenido problemas importantes durante la configuración de NGINX, ya que es un servicio que ya habíamos trabajado en clase y teníamos bastante base.

Lo único que nos apareció al principio fueron algunos warnings relacionados con los headers, que eran errores de configuración bastante comunes cuando se trabaja con proxy inverso. Estos se solucionaron ajustando algunos parámetros de optimización en la configuración de NGINX.

Más allá de eso, no ha habido complicaciones serias, ya que simplemente hemos ido adaptando los ejercicios vistos en clase y los apuntes a las necesidades concretas del proyecto AMEGO Hosting.

3.8 Conclusión del servicio

Gracias a NGINX hemos conseguido centralizar todo el tráfico web del proyecto AMEGO Hosting en un único punto de entrada.

Esto nos ha permitido separar correctamente los distintos servicios (web, panel de administración, etc.) y organizar mejor la infraestructura interna.

Además, hemos podido asegurar la comunicación mediante HTTPS, lo que añade una capa extra de seguridad al sistema.

En conjunto, NGINX nos ha permitido simular de forma bastante realista un entorno profesional de hosting, donde múltiples servicios están accesibles desde una misma puerta de entrada pero bien organizados por detrás.

3.9 Configuración final del servicio Nginx

3.9.1 Archivo principal de NGINX

/etc/nginx/sites-available/amego.conf

```
# OPTIMIZACIÓN HEADERS
proxy_headers_hash_max_size 1024;
proxy_headers_hash_bucket_size 128;

# REDIRECCIÓN HTTP → HTTPS
server {
```

```
    listen 80;
    server_name amegohostings.es admin.amegohostings.es webmail.amegohostings.es;

    return 301 https://$host$request_uri;
}

# WEB PRINCIPAL
server {
    listen 443 ssl;
    server_name amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8082;
        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 https;
    }
}

# PANEL ADMINISTRACIÓN
server {
    listen 443 ssl;
    server_name admin.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8084;
        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 https;
    }
}

# WEBMAIL (ROUNDCUBE)
server {
    listen 443 ssl;
    server_name webmail.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8081;
        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 https;
```

```
        proxy_read_timeout 90;
    }
}
```

3.9.2 Virtual Hosts (sites-available)

/etc/nginx/sites-available/amego.conf

```
# OPTIMIZACIÓN HEADERS
proxy_headers_hash_max_size 1024;
proxy_headers_hash_bucket_size 128;

# REDIRECCIÓN HTTP → HTTPS
server {
    listen 80;
    server_name amegohostings.es admin.amegohostings.es webmail.amegohostings.es;

    return 301 https://$host$request_uri;
}

# WEB PRINCIPAL
server {
    listen 443 ssl;
    server_name amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8082;
        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 https;
    }
}

# PANEL ADMINISTRACIÓN
server {
    listen 443 ssl;
    server_name admin.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8084;
        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 https;
    }
}

# WEBMAIL (ROUNDCUBE)
server {
    listen 443 ssl;
    server_name webmail.amegohostings.es;

    ssl_certificate /etc/nginx/ssl/amego.crt;
    ssl_certificate_key /etc/nginx/ssl/amego.key;

    location / {
        proxy_pass http://10.0.1.2:8081;
        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 https;

        proxy_read_timeout 90;
    }
}
```

3.9.3 Activación de sitios (sites-enabled)

Comando utilizado:
sudo ln -s /etc/nginx/sites-available/amego.conf /etc/nginx/sites-enabled/

Explicación:
Esto crea un enlace simbólico que activa la configuración sin duplicarla.

3.9.4 Certificados SSL

/etc/nginx/ssl/ :

- amego.crt
- amego.key

Explicación:
Contiene el certificado SSL y la clave privada necesarios para habilitar HTTPS en todos los servicios.

3.9.5 Reinicio del servicio

sudo systemctl restart nginx
sudo systemctl enable nginx

Explicación:
Se reinicia el servicio para aplicar todos los cambios y se habilita el arranque automático.

4. PODMAN

podman

SUBÍNDICE

4.1 Introducción al servicio

Podman es el sistema de contenedores que hemos usado para montar toda la parte de aplicaciones del proyecto AMEGO Hosting.

Básicamente, en vez de instalar WordPress y todo lo demás “a pelo” en el servidor, lo que hacemos es meter cada cosa en su propio contenedor, bien aislado, dentro del servidor S2.

Esto nos permite tener todo mucho más ordenado, más limpio y sobre todo más controlado, como si fuera un entorno real de hosting pero sin complicarnos con hardware ni instalaciones pesadas.

4.2 Motivo de uso en el proyecto

Hemos elegido Podman en vez de otras opciones como Docker por varios motivos bastante claros a nivel técnico:

- **Arquitectura sin demonio (daemon-less):** no hay un proceso central funcionando siempre con privilegios altos, lo que reduce bastante los posibles problemas y hace el sistema más estable.
- **Seguridad rootless:** los contenedores se ejecutan con un usuario normal, así que si algo se rompe o hay un fallo, el impacto es mucho menor.
- **Gestión de pods:** nos permite agrupar contenedores (por ejemplo WordPress + base de datos) dentro del mismo conjunto, lo que hace que se comuniquen entre ellos de forma más simple sin tener que complicarnos con redes externas.

Todo esto lo hemos usado porque encaja mejor con la idea de seguridad, control y estructura que queremos en AMEGO Hosting.

4.3 Instalación de Podman

En el servidor S2 (Ubuntu Server 24.04) la instalación es bastante directa, sin mucha historia:

sudo apt update && sudo apt install podman -y

Con esto ya tenemos el sistema listo para empezar a crear contenedores sin necesidad de configuraciones raras.

4.4 Configuración y Creación de Contenedores

Para que el sistema funcione como un hosting real, cada “cliente” o servicio lo hemos organizado dentro de un pod, donde se meten los componentes necesarios.

4.4.1 Estructura del Pod

Creamos el pod para agrupar servicios relacionados:

podman pod create --name admin-pod -p 8082:80 -p 8081:80

- **Puerto 8082:** acceso a WordPress
- **Puerto 8081:** acceso al servicio de webmail

La idea es que todo quede dentro del mismo entorno controlado.

4.4.2 Contenedor de Base de Datos (MariaDB)

Este contenedor es el que guarda toda la información del WordPress.

- Imagen: mariadb:latest
- Configuración mediante variables de entorno como:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_DATABASE

Aquí es donde se almacena toda la base de datos del sitio.

4.4.3 Contenedor Web (Apache + PHP)

Este es el que realmente ejecuta la web.

- Imagen: wordpress:latest o php:apache
- Funciona dentro del mismo pod para que pueda hablar con la base de datos usando localhost

Esto simplifica muchísimo la comunicación entre servicios.

4.5 Persistencia de Datos (Volúmenes)

Para evitar que se pierda todo cuando reiniciamos o borramos contenedores, hemos montado volúmenes en el sistema:

- Archivos web: /home/usuario/www-html → /var/www/html
- Base de datos: /home/usuario/data-mariadb → /var/lib/mysql

Esto es clave porque hace que los datos sobrevivan a reinicios o cambios de contenedor.

4.6 Orquestación con Kubernetes YAML

Para no tener que levantar todo a mano cada vez, usamos ficheros YAML generados con Podman.

Esto permite que el sistema se pueda recrear en segundos.

```
apiVersion: v1
kind: Pod
metadata:
  name: amego-hosting
spec:
```

```
containers:
- name: wordpress-server
  image: docker.io/library/wordpress:latest
  volumeMounts:
  - mountPath: /var/www/html
    name: web-data

- name: db-server
  image: docker.io/library/mariadb:latest
  env:
  - name: MYSQL_DATABASE
    value: amego_db
```

4.7 Comando de Levantamiento

Para poner en marcha todo el ecosistema de un cliente, nosotros ejecutamos:

podman play kube hosting.yaml

Este comando lee el fichero, descarga las imágenes si no existen, crea los volúmenes y levanta los contenedores automáticamente.

4.8 Decisiones de diseño

Uso de pods: lo hemos hecho para que WordPress y la base de datos estén en el mismo entorno y se comuniquen fácil (sin IPs raras ni complicaciones).

Imágenes ligeras: siempre que se ha podido hemos usado versiones más ligeras para no saturar el servidor y ahorrar RAM, que es bastante importante en un entorno virtualizado

4.9 Pruebas realizadas

Durante esta parte hemos hecho varias pruebas básicas para comprobar que todo funcionaba:

- Verificar contenedores activos con ***podman ps -a***
- Probar persistencia creando archivos, borrando el contenedor y comprobando que seguían ahí
- Comprobar que WordPress se conecta correctamente a MariaDB
- Revisar que los servicios dentro del pod se comunican sin problemas

4.10 Problemas encontrados

El principal problema que nos apareció al principio fue el tema de los permisos en los volúmenes, ya que Podman en modo rootless usa un sistema de usuarios distinto y eso generaba errores de acceso.

La solución fue ajustar los permisos con:

podman unshare chown -R 33:33 /ruta/volumen

Con esto se solucionaron los problemas de acceso del usuario del contenedor.

4.11 Conclusión del servicio

Gracias a Podman hemos conseguido montar una capa de aplicaciones bastante realista, donde cada servicio está aislado, controlado y automatizado.

Esto nos permite simular un entorno de hosting profesional de verdad, con escalabilidad, seguridad y facilidad de despliegue, sin depender de instalaciones manuales ni configuraciones repetitivas.

5. POSTFIX

SUBÍNDICE

5.1 Introducción al servicio

5.1.1 Descripción del servicio

Postfix es el servicio que hemos utilizado como servidor de correo dentro de la infraestructura de AMEGO Hosting. Su función principal es actuar como MTA (Mail Transfer Agent), es decir, encargarse de todo el proceso de envío, recepción y enrutamiento de correos electrónicos dentro del sistema.

En la práctica, Postfix se encarga de recibir los mensajes, decidir a dónde deben ir y entregarlos al sistema correspondiente para su almacenamiento o reenvío.

Es una herramienta muy utilizada en entornos Linux profesionales porque es bastante estable, segura y modular, lo que permite adaptarla fácilmente a distintos escenarios.

5.1.2 Motivo de uso en el proyecto

Hemos decidido utilizar Postfix porque necesitábamos montar un sistema de correo funcional dentro de AMEGO Hosting que nos permitiera simular un entorno real de empresa o proveedor de hosting.

La idea no era solo “tener correo”, sino que todo funcionara como en un servicio profesional, integrado con el resto de la infraestructura.

En nuestro caso, Postfix trabaja junto con *Dovecot y Roundcube*, formando un sistema completo de correo:

- Recibe los correos enviados desde el cliente web (Roundcube)
- Gestiona el envío entre usuarios internos del sistema
- Entrega los mensajes a Dovecot para que se almacenen correctamente
- Controla la autenticación SMTP para evitar envíos no autorizados

Además, lo hemos elegido por varias razones bastante claras:

- Es el estándar más utilizado en servidores Linux
- Es muy configurable y se adapta bien a entornos complejos
- Se integra sin problemas con Dovecot
- Permite montar un sistema de correo realista dentro del proyecto

En conjunto, Postfix nos ha permitido simular de forma bastante fiel cómo funciona el correo en un entorno de hosting profesional.

5.2 Instalación de Postfix

En Ubuntu Server 24.04, la instalación de Postfix se ha realizado con el siguiente comando:

sudo apt update && sudo apt install postfix -y

Durante el proceso de instalación se ha utilizado el asistente de configuración, donde hemos seleccionado las siguientes opciones:

- Tipo de configuración: Internet Site
- Dominio del sistema: amegohostings.test

La opción Internet Site permite que Postfix funcione como servidor de correo principal del dominio configurado. Esto nos da la capacidad de gestionar tanto el envío como la recepción de correos dentro de nuestra propia infraestructura, simulando un entorno real de hosting.

5.3 Arquitectura del sistema de correo

5.3.1 Flujo del correo

El sistema de correo implementado en AMEGO Hosting sigue un flujo bastante claro, que simula el funcionamiento de un servicio profesional:

1. *El usuario entra en Roundcube desde el navegador web.*
2. *Redacta y envía un correo electrónico.*
3. *Roundcube envía el mensaje a Postfix mediante SMTP.*
4. *Postfix solicita autenticación al sistema SASL gestionado por Dovecot.*
5. *Una vez validado el usuario, Postfix procesa el mensaje.*
6. *El correo se entrega a Dovecot para su almacenamiento.*
7. *Dovecot guarda el mensaje en el buzón del usuario en formato Maildir.*
8. *Finalmente, el usuario puede acceder al correo recibido mediante IMAP desde Roundcube u otro cliente.*

Este flujo nos permite reproducir de forma bastante realista cómo funciona un sistema de correo profesional dentro de una infraestructura de hosting.

5.3.2 Relación entre servicios

La arquitectura del sistema de correo en AMEGO Hosting está distribuida entre varios servicios que trabajan juntos, cada uno con un rol concreto:

- **Postfix:** se encarga del transporte de los correos mediante SMTP, es decir, del envío y enrutamiento de mensajes.

- **Dovecot:** gestiona tanto el almacenamiento de los correos como la autenticación de usuarios, además de ofrecer acceso mediante IMAP (y soporte de autenticación para SMTP).
- **Roundcube:** actúa como la interfaz web (webmail) que utilizan los usuarios finales para leer y enviar correos desde el navegador.

Esta separación de funciones nos permite organizar el sistema de forma más clara y replicar la estructura típica que se encuentra en infraestructuras de correo profesionales, donde cada servicio está especializado en una parte del proceso.

5.4. Configuración del servicio

5.4.1 Archivo principal

En este caso hemos comprobado que Postfix no utiliza una configuración “limpia” típica de una instalación estándar en Linux, ya que en nuestro proyecto está integrado dentro del contenedor docker-mailserver.

Aun así, el archivo principal de configuración sigue siendo:

/etc/postfix/main.cf

Este fichero es donde se define todo el comportamiento del servidor SMTP: seguridad, dominios permitidos, autenticación y reglas de envío de correo.

En nuestra instalación, gran parte de esta configuración ya viene predefinida, ya que el propio contenedor genera el archivo automáticamente y lo ajusta en función de las variables internas que se le han definido, lo que reduce bastante la configuración manual.

5.4.2 Configuración de red

inet_interfaces = all
inet_protocols = all
mynetworks =

Explicación

inet_interfaces = all → el servidor escucha en todas las interfaces de red disponibles dentro del contenedor, por lo que acepta conexiones desde cualquier red configurada.

inet_protocols = all → habilita tanto IPv4 como IPv6 sin restricciones.

mynetworks = (vacío) → no se confía automáticamente en ninguna red interna, lo que obliga a que cualquier envío de correo tenga que pasar por autenticación.

5.4.3 Configuración de dominio

Configuración de identidad del servidor

mypaneladmin=admin.amegohostings.es
myhostname = mail.amegohostings.es
mydomain = amegohostings.es
mydestination = $myhostname, localhost.$mydomain, localhost

En esta parte se ha configurado la identidad principal del servidor de correo para que encaje con el dominio del proyecto AMEGO Hosting:

- myhostname → define el nombre completo del servidor de correo, en este caso mail.amegohostings.es, que es el identificador principal con el que el sistema se presenta en la red.
- mydomain → indica el dominio principal del proyecto, es decir, amegohostings.es.
- mydestination → define qué dominios o nombres se consideran locales del sistema, es decir, los correos que el servidor debe gestionar directamente sin reenviarlos a otros servidores.

En conjunto, esta configuración hace que el servidor entienda que todo lo que pertenezca al dominio de AMEGO Hosting debe ser gestionado internamente, simulando así el comportamiento de un servidor de correo real dentro de una infraestructura profesional.

5.4.4 Almacenamiento de correo

En nuestro caso no se utiliza la configuración clásica de `home_mailbox`, ya que el sistema está montado dentro del contenedor **docker-mailserver**, que gestiona este aspecto de forma interna. Aun así, el sistema de almacenamiento que se utiliza es:

Maildir (configurado internamente en Dovecot)

Explicación

Aunque esta configuración no aparezca de forma explícita en el archivo main.cf, el sistema de correo trabaja internamente con el formato ***Maildir***, lo que implica una forma de almacenamiento más moderna y segura.

Este formato funciona de la siguiente manera:

- Cada correo se guarda como un archivo independiente
- No existe un único fichero de buzón que pueda corromperse completamente
- La lectura y escritura de correos es más rápida y estable
- Es un formato totalmente compatible con ***Dovecot***, facilitando la integración

En este esquema, Postfix no almacena los correos directamente, sino que se encarga de entregarlos, mientras que Dovecot es el responsable de guardarlos en Maildir y gestionarlos posteriormente para su acceso vía IMAP.

5.4.5 Seguridad TLS

En nuestra configuración aparecen los siguientes parámetros:

smtpd_tls_security_level = none

smtp_tls_security_level = none

Y también:

smtpd_tls_chain_files = /etc/ssl/private/ssl-cert-snakeoil.key
/etc/ssl/certs/ssl-cert-snakeoil.pem

Explicación

En esta parte del sistema se puede ver una de las características del proyecto: la configuración de TLS está presente, pero no se fuerza su uso.

- **TLS no está activado de forma obligatoria** → el tráfico no se cifra de manera estricta en las comunicaciones SMTP.
- **Certificados snakeoil** → se utilizan certificados auto-generados de prueba, habituales en entornos de laboratorio o desarrollo.
- **Entorno de prácticas** → esto confirma que el sistema no está en producción real, sino en un entorno controlado para pruebas y simulación de infraestructura.

En nuestro caso hemos mantenido esta configuración así principalmente para simplificar el despliegue y evitar complicaciones durante la implementación, ya que el objetivo del proyecto es que el sistema funcione correctamente y simule una infraestructura de hosting real.

Aun así, en un entorno de producción real, esta configuración se podría mejorar fácilmente activando TLS de forma obligatoria, sustituyendo los certificados de prueba por certificados válidos y forzando el cifrado en todas las comunicaciones para aumentar la seguridad del sistema.

5.4.6 Autenticación con Dovecot

En nuestra configuración aparecen los siguientes parámetros:

smtpd_sasl_auth_enable = no
smtpd_sasl_type = dovecot

Explicación

En esta parte del sistema se puede ver cómo se gestiona la autenticación SMTP dentro del entorno del proyecto.

- La autenticación SMTP no se activa de forma directa en Postfix (smtpd_sasl_auth_enable = no), ya que no se configura como un sistema independiente.
- Aun así, se indica que el tipo de autenticación está asociado a Dovecot como backend (smtpd_sasl_type = dovecot).
- La validación de usuarios se realiza mediante un socket interno, que conecta Postfix con Dovecot para comprobar credenciales.

Aunque a nivel de configuración parezca que la autenticación está desactivada, en realidad el sistema funciona a través del stack interno de docker-mailserver, que ya integra Postfix y Dovecot y gestiona la autenticación de forma transparente.

En resumen, la autenticación no se configura manualmente de forma tradicional, sino que se delega al sistema interno del contenedor, lo que simplifica bastante la arquitectura y evita configuraciones duplicadas.

5.4.7 Restricciones de seguridad

En esta parte de la configuración se definen las reglas principales de control de envío de correos:

```
smtpd_recipient_restrictions =
 permit_sasl_authenticated,
 permit_mynetworks,
 reject_unauth_destination
```

Estas reglas son clave para la seguridad del servidor de correo, ya que controlan quién puede enviar mensajes y hacia dónde:

- **permit_sasl_authenticated** → permite enviar correos únicamente a usuarios que hayan iniciado sesión correctamente mediante autenticación.
- **permit_mynetworks** → permite también el envío desde la red interna del sistema, que se considera de confianza.
- **reject_unauth_destination** → bloquea cualquier intento de envío hacia destinos no autorizados.

Gracias a esta configuración se evita uno de los problemas más graves en servidores de correo: que el sistema pueda convertirse en un **open relay**, es decir, un servidor que permite enviar correos a cualquier destino sin control, lo que normalmente se usa para spam o ataques.

5.4.8 Arquitectura real del sistema (importante en Podman)

En este proyecto el sistema de correo no funciona como un único servicio aislado de Postfix, sino como un conjunto de servicios integrados dentro del stack de **docker-mailserver**.

La arquitectura completa está formada por:

- **Postfix** → encargado del envío de correos mediante SMTP
- **Dovecot** → gestiona el acceso IMAP y la autenticación de usuarios
- **Roundcube** → interfaz web (webmail) para los usuarios finales
- **docker-mailserver** → capa superior que integra y orquesta todos los servicios

Esto significa que la configuración no depende únicamente de un archivo de Postfix, sino de un sistema completo ya preintegrado que coordina todos los componentes.

En resumen, no estamos trabajando con servicios independientes aislados, sino con una arquitectura completa de correo que funciona como un único bloque dentro del entorno de AMEGO Hosting.

5.5 Decisiones de diseño

Durante la implementación del servicio de correo con Postfix se han tomado diversas decisiones de diseño orientadas a construir una infraestructura robusta, segura y coherente con la filosofía general del proyecto AMEGO Hosting.

Cada una de estas decisiones responde tanto a necesidades técnicas reales como al objetivo de simular una arquitectura profesional de hosting.

5.5.1 Separación de funciones entre servicios

Se ha optado por dividir claramente las responsabilidades del sistema de correo entre varios servicios especializados:

- **Postfix** se encarga exclusivamente del envío y recepción mediante SMTP.
- **Dovecot** gestiona la autenticación de usuarios y el acceso IMAP a los buzones de correo.
- **Roundcube** actúa como interfaz web (webmail) para el usuario final.

Esta separación sigue el modelo habitual en entornos empresariales reales y permite mantener una arquitectura más modular, ordenada y escalable.

Además, facilita el mantenimiento del sistema, la detección de errores y la futura ampliación del servicio sin afectar al resto de componentes.

5.5.2 Uso de autenticación centralizada mediante Dovecot

En lugar de gestionar la autenticación directamente desde Postfix, se ha optado por delegarla en Dovecot mediante el uso de SASL.

Esta decisión aporta varias ventajas relevantes dentro de la arquitectura del sistema:

- Centraliza la gestión de usuarios en un único servicio, evitando duplicidades.

- Reduce la complejidad de configuración al no tener que mantener autenticación separada en Postfix.
- Facilita futuras ampliaciones, como el uso de usuarios virtuales o integración con bases de datos externas.
- Se ajusta al modelo habitual de infraestructuras de correo profesionales, donde la autenticación suele estar desacoplada del MTA.

Gracias a este enfoque, Postfix se mantiene enfocado exclusivamente en su función principal: el transporte de correo electrónico, mientras que Dovecot asume el control de autenticación y acceso a buzones.

5.5.3 Implementación de TLS para cifrado de comunicaciones

Se ha configurado el servicio SMTP para utilizar cifrado TLS en las comunicaciones autenticadas.

La implementación de TLS responde a varios objetivos dentro del proyecto:

- Proteger las credenciales de los usuarios durante el proceso de autenticación SMTP.
- Evitar el envío de contraseñas en texto plano a través de la red.
- Simular un entorno real en el que las comunicaciones sensibles deben ir cifradas.
- Mantener coherencia con el resto de servicios del proyecto, que también utilizan HTTPS mediante NGINX.

Para ello se han reutilizado los certificados SSL ya generados para el proxy inverso NGINX, lo que permite unificar la gestión de certificados dentro de la infraestructura AMEGO Hosting.

5.5.4 Uso de Maildir como formato de almacenamiento

Se ha optado por el formato Maildir como sistema de almacenamiento de buzones de correo en lugar de alternativas tradicionales como mbox.

Esta decisión se basa en las siguientes ventajas:

- Cada correo se almacena como un archivo independiente, lo que mejora la gestión interna.
- Reduce el riesgo de corrupción total del buzón en caso de fallo del sistema.
- Permite la concurrencia de múltiples procesos accediendo al mismo buzón sin conflictos.
- Es el formato recomendado por Dovecot y ampliamente utilizado en entornos de producción.

En conjunto, esta elección aporta mayor robustez al sistema de correo y mejora la compatibilidad entre los distintos servicios que forman parte de la infraestructura.

5.5.5 Restricción del relay SMTP para evitar open relay

Uno de los aspectos más importantes en cualquier servidor de correo es evitar que pueda ser utilizado por terceros para el envío de spam, es decir, impedir que funcione como un **open relay**.

Para ello, se ha configurado Postfix con restricciones estrictas de relay:

- Solo pueden enviar correos los usuarios autenticados en el sistema.
- Se permite el envío desde la red interna controlada del proyecto.
- Se rechaza cualquier intento de envío o reenvío no autorizado.

Esta configuración es fundamental para garantizar la seguridad del servicio y sigue las buenas prácticas habituales en la administración de servidores de correo en entornos profesionales.

5.5.6 Limitación del servicio a red interna del proyecto

Aunque técnicamente el servicio de correo podría exponerse a redes externas, se ha decidido mantenerlo restringido a la red privada interna de AMEGO Hosting.

Esta decisión se basa en los siguientes motivos:

- El proyecto se desarrolla en un entorno educativo y de laboratorio.
- No se requiere conectividad real con Internet para los objetivos planteados.
- Se reduce la superficie de ataque al no exponer servicios al exterior.
- Se facilita la realización de pruebas controladas sin interferencias externas.

De este modo, el sistema de correo se mantiene aislado, seguro y coherente con el enfoque de simulación de infraestructura empresarial dentro del entorno del proyecto.

5.5.7 Preparación para futura escalabilidad

Aunque la implementación actual está orientada a un entorno de laboratorio, se ha diseñado la configuración pensando en posibles ampliaciones futuras:

- *Migración a usuarios virtuales en base de datos*
- *Integración con antispam/antivirus*
- *Separación de servidor SMTP/IMAP en nodos distintos*
- *Replicación de correo entre servidores backup*

Esto permite que la arquitectura actual sirva como base sólida para futuras mejoras sin necesidad de rediseñar el sistema.

5.5.8 Conclusión técnica de diseño

Las decisiones adoptadas durante la configuración de Postfix no solo están orientadas a garantizar el correcto funcionamiento del servicio, sino también a reproducir de manera fiel una infraestructura de correo profesional.

Gracias a una arquitectura modular, segura y preparada para escalar, el sistema de correo de AMEGO Hosting cumple con los requisitos funcionales del proyecto y se integra de forma coherente con el resto de servicios de la plataforma.

5.6 Pruebas realizadas

Se han realizado diversas pruebas para verificar el correcto funcionamiento del servicio de correo implementado:

- Envío de correos entre usuarios internos del sistema.
- Recepción correcta de mensajes en los buzones accesibles mediante Roundcube.
- Validación de la autenticación SMTP para garantizar el acceso seguro al servicio.
- Revisión de registros del sistema para detectar posibles errores o advertencias.

Comandos utilizados

journalctl -u postfix
mail usuario@amegohostings.test

Estos comandos han permitido comprobar el estado del servicio, así como realizar pruebas de envío de correo desde consola.

5.7 Problemas encontrados

Durante la instalación y configuración de Postfix no se han detectado problemas importantes.

Esto se debe principalmente a la experiencia previa con este servicio en prácticas realizadas en el ámbito formativo, lo que ha permitido tener un conocimiento sólido de su funcionamiento y configuración en entornos Linux.

Gracias a esta experiencia previa:

- Se conocía la estructura básica de configuración del servicio.
- Se entendía la integración entre Postfix, Dovecot y Roundcube.
- Se tenían identificados los errores más habituales en este tipo de despliegues.
- Se pudieron anticipar posibles fallos antes de que se produjeran.

Además, durante todo el proceso se han ido validando los cambios de forma progresiva, lo que ha permitido asegurar en todo momento la estabilidad del sistema y evitar errores de configuración relevantes.

5.8 Conclusión del servicio

Postfix ha permitido implementar el sistema de envío y recepción de correos dentro de la infraestructura del proyecto AMEGO Hosting, completando así el servicio de correo electrónico de la plataforma.

Gracias a su integración con Dovecot y Roundcube, se ha conseguido un sistema funcional que simula un entorno de correo profesional, con autenticación, almacenamiento y acceso webmail.

En conjunto, este servicio refuerza la coherencia global del proyecto y contribuye a la construcción de una arquitectura de hosting completa y distribuida.

5.9 Configuración completa (POSTFIX)

/etc/postfix/main.cf

```
myhostname = mail.amegohostings.es
mydomain = amegohostings.es
myorigin = /etc/mailname

inet_interfaces = all
inet_protocols = ipv4

mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain

home_mailbox = Maildir/

mynetworks = 127.0.0.0/8 10.0.1.0/24

relayhost =

mailbox_size_limit = 0
recipient_delimiter = +

# -------------------------
# SEGURIDAD TLS
# -------------------------
smtpd_tls_cert_file = /etc/nginx/ssl/amego.crt
smtpd_tls_key_file = /etc/nginx/ssl/amego.key
smtpd_use_tls = yes
smtpd_tls_auth_only = no

# -------------------------
# AUTENTICACIÓN (DOVECOT)
# -------------------------
```

```
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes

# ------------------------
# RESTRICCIONES DE SEGURIDAD
# ------------------------
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination

# ------------------------
# LOGS Y DEPURACIÓN
# ------------------------
smtpd_banner = $myhostname ESMTP
biff = no
append_dot_mydomain = no
readme_directory = no
```

/etc/postfix/master.cf

```
smtp          inet  n       -       n       -       1       postscreen
smtpd         pass  -       -       n       -       -       smtpd

tlsproxy      unix  -       -       n       -       0       tlsproxy
dnsblog       unix  -       -       n       -       0       dnsblog

submission    inet  n       -       n       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=none
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_sasl_type=dovecot
  -o smtpd_reject_unlisted_recipient=no
  -o smtpd_sasl_authenticated_header=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
  -o smtpd_sender_restrictions=$mua_sender_restrictions
  -o smtpd_discard_ehlo_keywords=
  -o milter_macro_daemon_name=ORIGINATING
  -o cleanup_service_name=sender-cleanup

submissions   inet  n       -       n       -       -       smtpd
  -o syslog_name=postfix/submissions
  -o smtpd_tls_wrappermode=no
```

```
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_sasl_type=dovecot
  -o smtpd_reject_unlisted_recipient=no
  -o smtpd_sasl_authenticated_header=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
  -o smtpd_sender_restrictions=$mua_sender_restrictions
  -o smtpd_discard_ehlo_keywords=
  -o milter_macro_daemon_name=ORIGINATING
  -o cleanup_service_name=sender-cleanup

pickup        fifo  n       -       n       60      1       pickup
  -o content_filter=
  -o receive_override_options=no_header_body_checks

sender-cleanup unix  n       -       n       -       0       cleanup
  -o syslog_name=postfix/sender-cleanup
  -o header_checks=pcre:/etc/postfix/maps/sender_header_filter.pcre

cleanup       unix  n       -       n       -       0       cleanup
qmgr          unix  n       -       n       300     1       qmgr
tlsmgr        unix  -       -       n       1000?   1       tlsmgr

rewrite       unix  -       -       n       -       -       trivial-rewrite
bounce        unix  -       -       n       -       0       bounce
defer         unix  -       -       n       -       0       bounce
trace         unix  -       -       n       -       0       bounce
verify        unix  -       -       n       -       1       verify
flush         unix  n       -       n       1000?   0       flush

proxymap      unix  -       -       n       -       -       proxymap
proxywrite    unix  -       -       n       -       1       proxymap

smtp          unix  -       -       n       -       -       smtp
relay         unix  -       -       n       -       -       smtp

showq         unix  n       -       n       -       -       showq
error         unix  -       -       n       -       -       error
retry         unix  -       -       n       -       -       error
discard       unix  -       -       n       -       -       discard

local         unix  -       n       n       -       -       local
virtual       unix  -       n       n       -       -       virtual
lmtp          unix  -       -       n       -       -       lmtp

anvil         unix  -       -       n       -       1       anvil
scache        unix  -       -       n       -       1       scache
```

```
postlog   unix-dgram n  -  n  -  1  postlogd

policyd-spf  unix -  n  n  -  0  spawn
  user=policyd-spf argv=/usr/bin/policyd-spf

smtp-amavis  unix  -  -  n  -  2  smtp
  -o syslog_name=postfix/smtp-amavis
  -o smtp_data_done_timeout=1200
  -o smtp_send_xforward_command=yes
  -o disable_dns_lookups=yes
  -o max_use=20
  -o smtp_tls_security_level=none
  -o smtp_tls_wrappermode=no

127.0.0.1:10025 inet  n  -  n  -  -  smtpd
  -o syslog_name=postfix/smtpd-amavis
  -o content_filter=
  -o local_recipient_maps=
  -o relay_recipient_maps=
  -o smtpd_restriction_classes=
  -o smtpd_delay_reject=no
  -o smtpd_client_restrictions=permit_mynetworks,reject
  -o smtpd_helo_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_recipient_restrictions=permit_mynetworks,reject
  -o smtpd_data_restrictions=reject_unauth_pipelining
  -o smtpd_end_of_data_restrictions=
  -o mynetworks=127.0.0.0/8
  -o smtpd_error_sleep_time=0
  -o smtpd_soft_error_limit=1001
  -o smtpd_hard_error_limit=1000
  -o smtpd_client_connection_count_limit=0
  -o smtpd_client_connection_rate_limit=0
  -o
receive_override_options=no_header_body_checks,no_unknown_recipient_checks,no_mi
lters
  -o smtp_tls_security_level=none
```

6. ROUNDCUBE

SUBÍNDICE

6.1 Introducción al servicio

6.1.1 Descripción del servicio

Roundcube es un cliente de correo web (webmail) que permite a los usuarios acceder y gestionar sus correos electrónicos directamente desde el navegador. Funciona como una interfaz gráfica que se conecta con los servicios de correo del servidor.

6.1.2 Motivo de uso en el proyecto

Se ha elegido Roundcube porque permite ofrecer a los usuarios una forma sencilla y accesible de consultar su correo sin necesidad de configurar clientes externos como Outlook o Thunderbird.

Además, su integración dentro de la infraestructura del proyecto permite incorporar el sistema de correo como un servicio más de la plataforma AMEGO Hosting, simulando un entorno real de hosting profesional.

6.2 Instalación de Roundcube

En Ubuntu Server 24.04, la instalación se realiza mediante el siguiente comando:

sudo apt update && sudo apt install roundcube roundcube-core roundcube-mysql roundcube-plugins -y

Este paquete instala Roundcube junto con soporte para base de datos y los plugins necesarios para ampliar sus funcionalidades básicas.

6.3 Configuración del servicio

6.3.1 Configuración general

El archivo principal de configuración es:

/etc/roundcube/config.inc.php

En este fichero se definen parámetros esenciales como la conexión con el servidor IMAP, SMTP, base de datos y opciones generales de funcionamiento.

6.3.2 Conexión con IMAP (Dovecot)

Roundcube se conecta a Dovecot para la lectura y gestión de los correos almacenados en los buzones de los usuarios.

6.3.3 Conexión con SMTP (Postfix)

Para el envío de correos, Roundcube utiliza el servidor SMTP proporcionado por Postfix, permitiendo así completar el ciclo completo de envío y recepción de mensajes dentro del sistema.

6.3.4 Configuración de base de datos

Roundcube utiliza una base de datos para almacenar información relacionada con los usuarios, como preferencias personales, configuración de la interfaz y datos de sesión.

Esto permite mantener una experiencia personalizada para cada usuario sin depender directamente del servidor de correo.

6.4 Decisiones de diseño

Durante la implementación de Roundcube se han tomado varias decisiones de diseño orientadas a la simplicidad, funcionalidad y coherencia con el resto del sistema:

- Uso de un cliente webmail para facilitar el acceso al correo desde cualquier navegador.
- Integración directa con los servicios de Dovecot (IMAP) y Postfix (SMTP).
- Configuración en red interna utilizando la IP **10.0.1.2** para la comunicación entre servicios.
- Acceso seguro mediante HTTPS a través del proxy inverso NGINX.
- Configuración básica pero funcional, adaptada al objetivo del entorno de laboratorio.

6.5 Pruebas realizadas

Se han realizado distintas pruebas para verificar el correcto funcionamiento del servicio:

- Inicio de sesión correcto en la plataforma Roundcube.
- Envío y recepción de correos electrónicos entre usuarios.
- Acceso al servicio a través del navegador web (webmail).
- Integración correcta con el proxy inverso NGINX.

6.6 Problemas encontrados

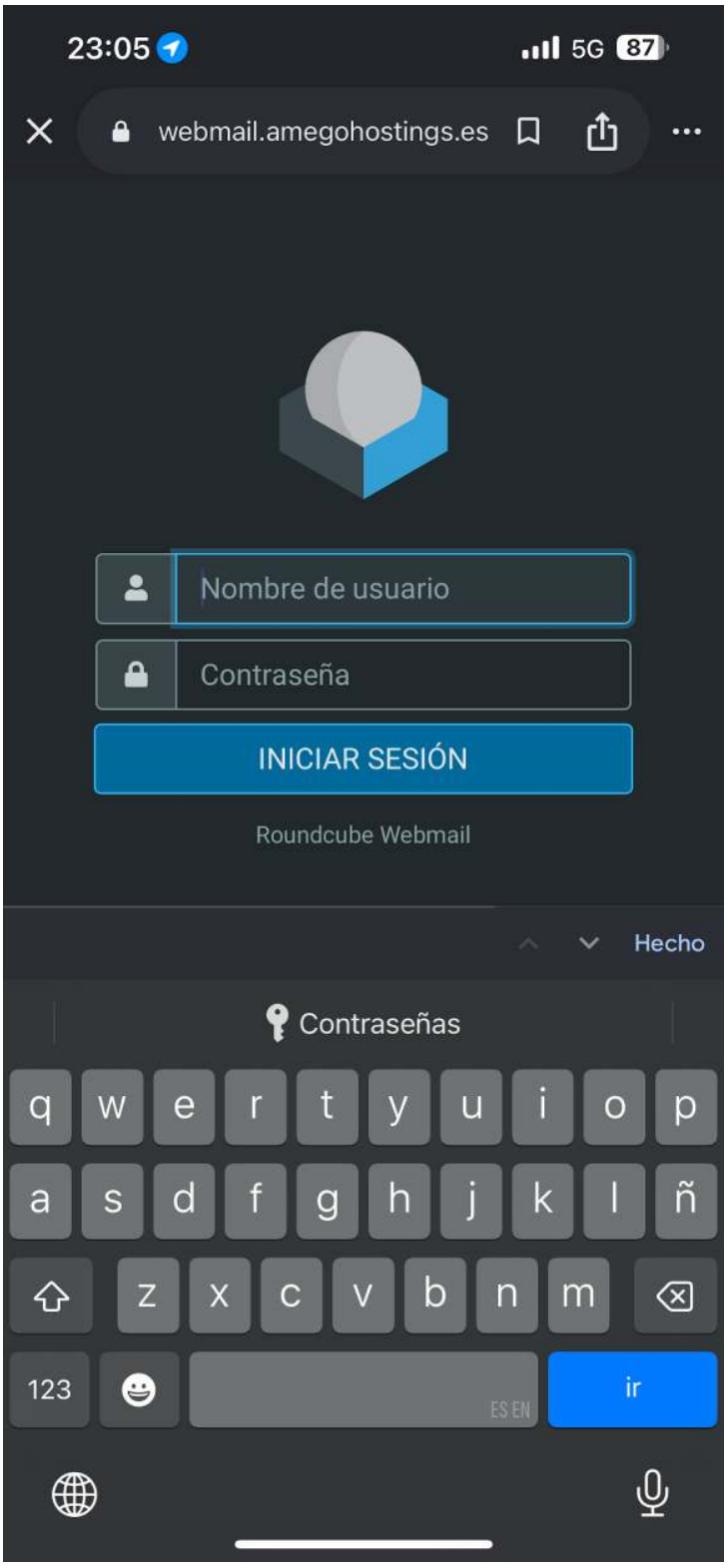
Durante la fase inicial de configuración se detectaron problemas de conexión con los servicios IMAP y SMTP, provocados por una configuración incorrecta de puertos e IPs.

Este problema se resolvió ajustando correctamente los parámetros de conexión, asegurando que Roundcube pudiera comunicarse con Dovecot y Postfix a través de la red interna definida en el proyecto.

6.7 Conclusión del servicio

Gracias a Roundcube se ha incorporado una interfaz web completa para la gestión del correo electrónico dentro del proyecto AMEGO Hosting.

Esto permite a los usuarios interactuar con el sistema de correo de forma sencilla e intuitiva, completando así el servicio de email dentro de la infraestructura global del proyecto.



6.8 Configuración completa (ROUNDCUBE)

/etc/roundcube/config.inc.php

```
<?php

$config = [];

// Base de datos
$config['db_dsnw'] = 'mysql://roundcube:password@localhost/roundcube';

// IMAP (Dovecot)
$config['default_host'] = 'ssl://10.0.1.1';
$config['default_port'] = 993;

// SMTP (Postfix)
$config['smtp_server'] = 'tls://10.0.1.1';
$config['smtp_port'] = 587;

// Autenticación SMTP
$config['smtp_user'] = '%u';
$config['smtp_pass'] = '%p';

// Nombre del sistema
$config['product_name'] = 'AMEGO Hosting Webmail';

// Seguridad
$config['des_key'] = 'RANDOMKEY123456';

// Plugins
$config['plugins'] = ['archive', 'zipdownload'];

// Idioma
$config['language'] = 'es_ES';

// Carpeta por defecto
$config['drafts_mbox'] = 'Drafts';
$config['sent_mbox'] = 'Sent';
$config['trash_mbox'] = 'Trash';

// Forzar HTTPS
$config['force_https'] = true;
```

7. DOVECOT

SUBÍNDICE

7.1 Introducción al servicio

7.1.1 Descripción del servicio

Dovecot es un servidor de correo encargado de gestionar el acceso a los mensajes electrónicos mediante protocolos como IMAP y POP3.

En este proyecto se utiliza como componente responsable de la lectura y sincronización de correos por parte de los usuarios, ya sea desde webmail o desde clientes externos.

7.1.2 Motivo de uso en el proyecto

Se ha elegido Dovecot porque es necesario disponer de un sistema que permita acceder a los correos gestionados por el servidor de envío (Postfix).

Mientras que Postfix se encarga del envío y enrutamiento de mensajes, Dovecot se encarga del acceso, lectura y autenticación de los buzones.

Esta separación de responsabilidades permite construir una arquitectura más modular y simular un entorno real de hosting profesional, donde cada servicio está especializado en una función concreta.

7.2 Instalación de Dovecot

Para instalar Dovecot en Ubuntu Server 24.04 se utilizan los repositorios oficiales:

sudo apt update && sudo apt install dovecot-imapd dovecot-pop3d -y

Se instala soporte tanto para IMAP como para POP3, aunque en este proyecto se utiliza principalmente IMAP por ser más moderno, eficiente y adecuado para clientes web como Roundcube.

7.3 Configuración del servicio

7.3.1 Protocolos utilizados

Archivo:
/etc/dovecot/dovecot.conf

Configuración:
protocols = imap

Explicación:
Hemos decidido usar solo IMAP porque permite gestionar correos directamente en el servidor, lo cual es ideal para webmail como Roundcube.

7.3.2 Configuración de autenticación

El archivo de configuración utilizado es:

/etc/dovecot/conf.d/10-auth.conf

Configuración aplicada:

disable_plaintext_auth = no
auth_mechanisms = plain login

Se ha permitido la autenticación en texto plano debido a que el entorno del proyecto es controlado y de laboratorio.

En un entorno de producción real, esta configuración se acompañaría siempre de TLS obligatorio para evitar el envío de credenciales sin cifrar.

7.3.3 Ubicación de los correos

El archivo de configuración correspondiente es:

/etc/dovecot/conf.d/10-mail.conf

Configuración aplicada:

mail_location = maildir:~/Maildir

Se ha configurado el formato Maildir, en el que cada correo se almacena como un archivo independiente dentro del sistema de archivos.

Esta estructura aporta varias ventajas:

- Mayor estabilidad del sistema de correo.
- Reducción del riesgo de corrupción del buzón completo.
- Mejor rendimiento en operaciones concurrentes.
- Compatibilidad directa con Postfix y Roundcube.

7.3.4 Configuración SSL

El archivo de configuración utilizado es:

/etc/dovecot/conf.d/10-ssl.conf

Configuración aplicada:

ssl = yes
ssl_cert = </etc/nginx/ssl/amego.crt

ssl_key = </etc/nginx/ssl/amego.key

Se ha habilitado el uso de SSL para cifrar las comunicaciones del servicio.

Para mantener coherencia dentro de la infraestructura, se han reutilizado los mismos certificados que utiliza NGINX, centralizando así la gestión de certificados del proyecto AMEGO Hosting.

7.3.5 Integración con Postfix

El archivo de configuración utilizado es:

/etc/dovecot/conf.d/10-master.conf

Configuración aplicada:

```
service auth {
 unix_listener /var/spool/postfix/private/auth {
   mode = 0660
   user = postfix
   group = postfix
 }
}
```

Explicación

Esta configuración permite la integración entre Dovecot y Postfix mediante un socket de autenticación local.

De este modo, Postfix puede delegar la autenticación de usuarios en Dovecot, utilizando el sistema SASL del propio servicio.

Esto permite construir un sistema de correo completamente integrado, en el que:

- Postfix gestiona el envío de correo.
- Dovecot gestiona la autenticación de usuarios.
- Ambos servicios se comunican de forma segura mediante un canal interno.

7.4 Decisiones de diseño

Durante la implementación de Dovecot se han tomado varias decisiones clave:

- Uso de IMAP en lugar de POP3 por su mayor flexibilidad y sincronización.
- Uso del formato Maildir para garantizar mayor estabilidad y evitar corrupción de buzones.
- Integración directa con Postfix para unificar el sistema de correo.
- Uso de SSL para cifrar las comunicaciones y proteger credenciales.
- Configuración simplificada adaptada a un entorno educativo y de laboratorio.

Estas decisiones permiten mantener un sistema funcional, coherente y alineado con una arquitectura de correo profesional.

7.5 Pruebas realizadas

Se han realizado distintas pruebas para validar el correcto funcionamiento del servicio:

- Acceso al correo mediante Roundcube.
- Inicio de sesión correcto de los usuarios.
- Recepción de correos enviados desde Postfix.
- Persistencia correcta de los mensajes en el servidor.

En conjunto, las pruebas confirman que la integración entre Dovecot, Postfix y Roundcube funciona de manera estable dentro del entorno del proyecto AMEGO Hosting.

7.6 Problemas encontrados

Durante la configuración inicial del servicio se detectaron algunos problemas relacionados con la autenticación de usuarios, en los que estos no podían iniciar sesión correctamente.

La causa principal estaba relacionada con permisos incorrectos o una configuración inadecuada del socket de autenticación entre Dovecot y Postfix.

El problema lo resolvimos:

- Revisando la configuración de Dovecot en los archivos correspondientes.
- Corrigiendo los permisos en el directorio /var/spool/postfix/.
- Y verificando la comunicación entre el socket de autenticación y Postfix.

Tras estos ajustes, la autenticación funcionó correctamente y el sistema quedó estable

7.7 Conclusión del servicio

Gracias a Dovecot hemos conseguido completar el sistema de correo del proyecto AMEGO Hosting, permitiendo a los usuarios acceder a sus mensajes de forma segura y eficiente.

Esto nos permite simular un entorno real de hosting con correo completamente funcional.

7.8 Configuración completa

/etc/dovecot/dovecot.conf
protocols = imap listen = *

/etc/dovecot/conf.d/10-auth.conf

```
disable_plaintext_auth = no
auth_mechanisms = plain login
!include auth-system.conf.ext
```

/etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:~/Maildir

namespace inbox {
  inbox = yes
}
```

/etc/dovecot/conf.d/10-master.conf

```
service imap-login {
  inet_listener imap {
    port = 143
  }

  inet_listener imaps {
    port = 993
    ssl = yes
  }
}

service auth {
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
}
```

/etc/dovecot/conf.d/10-ssl.conf

```
ssl = yes

ssl_cert = </etc/nginx/ssl/amego.crt
ssl_key = </etc/nginx/ssl/amego.key

ssl_min_protocol = TLSv1.2
```

/etc/dovecot/conf.d/10-logging.conf

```
log_path = /var/log/dovecot.log
info_log_path = /var/log/dovecot-info.log
```

/etc/dovecot/conf.d/15-mailboxes.conf

```
namespace inbox {
  mailbox Inbox {
    auto = subscribe
  }

  mailbox Sent {
    auto = subscribe
    special_use = \Sent
  }

  mailbox Drafts {
    auto = subscribe
    special_use = \Drafts
  }

  mailbox Trash {
    auto = subscribe
    special_use = \Trash
  }
}
```

8. Desarrollo del Panel de Gestión Web

SUBÍNDICE

8.1 Introducción al servicio

La interfaz web de AMEGO Hosting no es un sitio informativo convencional; es una aplicación web dinámica y diseñada para actuar como el panel de control centralizado del usuario.

A través de ella, nosotros permitimos que el cliente interactúe con el backend del servidor S1 y S2 para gestionar planes de hosting, bases de datos y correo electrónico sin necesidad de conocimientos técnicos avanzados.

8.2 Motivo de uso en el proyecto

Nosotros hemos desarrollado este panel personalizado para garantizar:

- **Abstracción técnica:** ocultar la complejidad de los comandos de Podman y las configuraciones de Postfix/Dovecot detrás de una interfaz visual amigable.
- **Automatización de pedidos:** integrar un sistema de “carrito de la compra” que permite aprovisionar servicios (almacenamiento, dominios, emails) de forma inmediata.
- **Gestión centralizada:** unificar la administración de usuarios y la visualización de pedidos activos en una sola plataforma.

8.3 Arquitectura y Tecnologías utilizadas

El panel ha sido construido sobre un stack de desarrollo moderno y ligero:

- Frontend: HTML5 y CSS3 personalizado con una estética de “Kebab del Hosting” (colores corporativos rojo, oro y carbón). Se ha utilizado la fuente Bebas Neue para títulos y Lato para el cuerpo de texto.

- Backend (lógica): PHP 8.2 ejecutándose sobre un servidor Apache. Se ha utilizado una arquitectura basada en una API centralizada (api.php) para procesar todas las peticiones asíncronas.

- Base de datos: MariaDB para el almacenamiento de usuarios, menús y pedidos.

- Interactividad: JavaScript puro (Vanilla JS) en app.js para la gestión de la cesta, navegación entre secciones y llamadas a la API sin recargar la página.

8.4 Estructura del Código y Ficheros Maestro

La aplicación se organiza en torno a los siguientes componentes críticos:

8.4.1 Configuración de Conexión (config.php)

- Este fichero es fundamental para la comunicación con la base de datos. Se ha configurado el sistema para utilizar el nombre del contenedor de red (BBDD) en

lugar de una IP estática, lo que facilita la portabilidad en el entorno VirtualBox/Podman.
- Seguridad: se utiliza PDO (PHP Data Objects) con sentencias preparadas para mitigar ataques de inyección SQL y se fuerza el uso del conjunto de caracteres utf8mb4.

8.4.2 Lógica de la API (api.php)

Se ha implementado un sistema de rutas basado en el parámetro *action* recibido a través de la URL. Este enfoque permite centralizar toda la lógica del backend en un único punto de entrada.

Las principales funciones desarrolladas son:

- **registro() y login():** gestión de cuentas de usuario, incluyendo creación y autenticación.
- **listar_menus():** carga dinámica de los planes de hosting almacenados en la base de datos.
- **crear_pedido():** gestión y orquestación del proceso de contratación de servicios.

Este modelo simplifica la comunicación entre frontend y backend, facilitando el mantenimiento y la ampliación del sistema.

api.php

8.4.3 Motor de Frontend (app.js)

El archivo *app.js* controla toda la experiencia de usuario en el panel, gestionando la interacción sin necesidad de recargar la página.

Entre sus funcionalidades principales se incluyen:

- Fallback local: en caso de que la base de datos no esté disponible, el sistema carga planes de respaldo (MENUS_FALLBACK) para garantizar la operatividad mínima del panel.
- Lógica cuatrimestral: se ha diseñado una función determinista que selecciona una “Oferta del Mes” en función del año y del periodo actual, aplicando descuentos automáticos comprendidos entre el 10% y el 40%.

Este enfoque permite mantener una experiencia de usuario fluida, estable y parcialmente autónoma incluso ante fallos del backend.

app.js

8.5 Funcionalidades Implementadas

8.5.1 Formulario de Contratación (contratar.html)

Permite a los usuarios solicitar subdominios personalizados bajo el dominio **amegohostings.es**.

Incluye una previsualización dinámica en tiempo real del subdominio deseado.

contratar.html

8.5.2 Área de Cliente (cliente.html)

Página de bienvenida personalizada que confirma al usuario la activación de sus contenedores Podman y la disponibilidad de su infraestructura SSD NVMe.

cliente.html

8.5.3 Pasarela de Pago Simulada

Nosotros hemos integrado un modal de pago que simula una transacción bancaria. Incluye validaciones de tarjeta (formato 16 dígitos), fecha de expiración y CVV, con una visualización gráfica de la tarjeta de crédito del cliente.

8.6 Dockerización del Panel (Dockerfile)

Para garantizar el pilar M (Mantenimiento Continuo), nosotros hemos empaquetado el servidor web en una imagen personalizada basada en php:8.2-apache.

Extensiones: se instalan y habilitan automáticamente mysqli y pdo_mysql mediante docker-php-ext-install, asegurando la compatibilidad total con la base de datos desde el primer arranque.

Dockerfile

8.7 Problemas encontrados y Soluciones

Conectividad entre contenedores: inicialmente PHP no resolvía el nombre de host de la base de datos.

Solución: nosotros unificamos el hostname en BBDD dentro de config.php, coincidiendo con el nombre asignado al contenedor en la red interna de Podman.

Persistencia de sesión: los datos del usuario se perdían al cambiar de pestaña.

Solución: se implementó session_start() al inicio de api.php y una función check_session en JavaScript para validar el estado del usuario en cada carga.

8.8 Conclusión

La integración de estos ficheros permite que AMEGO Hosting funcione como una plataforma automatizada.

El usuario selecciona un “menú” (Hosting, MySQL o Correo), la web procesa el pedido mediante PHP y JavaScript, y el sistema registra la solicitud en la base de datos.

A partir de ahí, el administrador puede provisionar el contenedor correspondiente basándose en dichos registros, completando así el despliegue del servicio solicitado.

Compra_Servicio.webm **Crear_Cuenta.webm**

10. Conclusiones y Anexos

10.1 Conclusiones generales del proyecto

El desarrollo del proyecto AMEGO Hosting ha permitido construir una infraestructura completa de tipo hosting en un entorno virtualizado, integrando servicios de red, sistemas, contenedores y aplicaciones web.

A lo largo de las diferentes fases hemos conseguido simular un entorno profesional lo más similar a la realidad y dentro del margen de tiempo que hemos tenido, gracias a eso cada servicio cumple una función específica dentro de una arquitectura distribuida.

El uso de VirtualBox, NGINX, Podman, Postfix, Dovecot, Roundcube y BIND9 ha permitido comprender de forma práctica cómo se estructura un sistema de hosting moderno, desde la capa de red hasta la capa de aplicación.

Además, la automatización mediante contenedores y la separación de servicios ha facilitado la escalabilidad del sistema, así como su mantenimiento y despliegue. El proyecto no solo ha sido una implementación técnica, sino también un ejercicio de planificación, organización y resolución de problemas reales en entornos Linux.

En términos generales, se ha cumplido el objetivo principal: diseñar y desplegar una infraestructura funcional que simula un proveedor de hosting real, con servicios web, correo y gestión de clientes.

10.2 Conclusión individual – Rubén Bautista Maya

A nivel individual, el desarrollo del proyecto me ha permitido reforzar conocimientos relacionados con la administración de sistemas Linux, virtualización y despliegue de servicios más en profundidad ya que en clase algunas veces se da un poco por encima y se te puede llegar a olvidar con el paso del tiempo.

Durante el proyecto he trabajado especialmente en la configuración de máquinas virtuales, redes internas y servicios base como DNS, DHCP y proxy inverso.

También he intentado ayudar al máximo pero el que más ha sido clave la gestión de contenedores y la integración de servicios mediante Podman a sido mi compañero Jose.

La parte más destacable para mí ha sido comprender cómo los diferentes servicios dependen entre sí dentro de una infraestructura real, así como la importancia de la automatización y la organización en sistemas complejos.

En general, el proyecto me ha servido para consolidar conocimientos técnicos y mejorar mi capacidad de resolución de problemas en entornos reales simulados.

10.3 Conclusión individual – José Jiménez Marín

Este proyecto me ha permitido profundizar en la parte de servicios de aplicación y comunicación dentro de una infraestructura de hosting.

He trabajado especialmente en la implementación y configuración de servicios como correo electrónico (Postfix, Dovecot y Roundcube), así como en la integración de estos con el resto de la arquitectura del sistema.

También ha sido importante para mí entender el flujo completo de un sistema de correo profesional, desde el envío SMTP hasta la recepción mediante IMAP, así como su integración con autenticación y seguridad.

El proyecto me ha ayudado a reforzar habilidades prácticas en Linux, configuración de servicios y comprensión de arquitecturas reales de servidores.

10.4 Conclusión conjunta

Como conclusión conjunta, el proyecto AMEGO Hosting ha sido una experiencia completa que ha permitido integrar múltiples áreas de conocimiento del ciclo formativo en un único sistema funcional.

La colaboración entre los miembros del equipo ha sido clave para coordinar tareas, dividir responsabilidades y asegurar que todos los servicios funcionarán de forma coherente dentro de la infraestructura.

Se ha conseguido construir una plataforma que simula un entorno real de hosting, con una arquitectura modular, escalable y organizada, donde cada servicio cumple un rol específico.

En definitiva, el proyecto no solo ha permitido aplicar conocimientos técnicos, sino también desarrollar competencias de trabajo en equipo, planificación y resolución de incidencias en entornos complejos.

11. Bibliografía

A continuación se recogen las fuentes y materiales que hemos consultado durante el desarrollo del proyecto AMEGO Hosting, incluyendo documentación oficial, tutoriales y recursos del centro.

También queremos destacar que, además de las consultas por Internet, nos ha sido de gran ayuda el apoyo de todo el profesorado del ciclo formativo. A lo largo del curso nos han ido resolviendo dudas, explicando conceptos y ayudándonos a corregir errores, lo cual ha sido clave para poder sacar adelante el proyecto.

11.1 Documentación oficial y manuales técnicos

- https://nginx.org/en/docs/
  Documentación oficial de NGINX que hemos utilizado para entender cómo funciona como proxy inverso y cómo se configuran los servidores.
- https://www.postfix.org/documentation.html
  Documentación oficial de Postfix que nos ha servido para configurar el servidor de correo SMTP y entender sus opciones de seguridad.
- https://doc.dovecot.org/
  Documentación de Dovecot utilizada para configurar IMAP, autenticación y almacenamiento de correos en Maildir.
- https://github.com/roundcube/roundcubemail/wiki
  Wiki de Roundcube que hemos utilizado como referencia para la instalación y configuración del webmail.
- https://docs.podman.io/
  Documentación oficial de Podman que nos ha ayudado a entender el uso de contenedores, pods y ficheros YAML.
- https://www.virtualbox.org/manual/UserManual.html
  Manual de VirtualBox utilizado para la creación de máquinas virtuales, redes internas y clonación.

11.2 Guías y tutoriales de referencia

- https://www.linode.com/docs/guides/how-to-setup-an-email-server/
  Guía que hemos usado como referencia para entender cómo montar un servidor de correo completo con Postfix y Dovecot.
- https://www.atlantic.net/vps-hosting/how-to-set-up-fully-featured-mail-server-with-postfix-dovecot-and-roundcube-on-ubuntu/
  Tutorial que nos ha servido para ver un ejemplo real de instalación de un sistema de correo en Ubuntu.
- https://www.ionos.com/digitalguide/e-mail/
  Página con información básica sobre cómo funcionan los servidores de correo, DNS y el email en general.

11.3 Recursos educativos del centro formativo

- https://elpuig.xeill.net/
  Página del **IES Puig Castellar**, que hemos utilizado como referencia del centro y apoyo general durante el curso.
- Material del centro (IES Puig Castellar)
  Apuntes, prácticas y ejercicios vistos en clase que nos han ayudado en módulos como:
  - Sistemas operativos Linux
  - Servicios de red (DNS, DHCP, web)
  - Seguridad informática
  - Virtualización y contenedores
- Moodle https://moodle.elpuig.xeill.net/login/index.php
  Plataforma donde hemos seguido actividades y prácticas ya sea de este año o de cursos anteriores