

S.I.G.P.D.

Administración de Sistemas Operativos JurassiCode

Rol	Apellido	Nombre	C.I	Email
Coordinador	Fianza	Ignacio	5.690.153-1	businessignaciofianza@gmail.com
Sub-Coordinador	Benítez	Sebastián	5.652.044-4	sebastianbenitez2505@gmail.com
Integrante 1	Fleitas	Joaquín	5.570.982-3	joacolambru7@gmail.com
Integrante 2	Paz	Tomás	5.700.344-1	tomaslautaropaz@gmail.com

Docente: Martínez, Santiago

**Fecha de
culminación
10/11/2025**

TERCERA ENTREGA

I.S.B.O.

3MI



ÍNDICE

ÍNDICE	2
Script bash gestor de usuarios y grupos	3
Manual de instalación y justificación de Fedora Server	3
Instalación y Configuración de la Red en Fedora Server	6
Gestión y administración de paquetes del S.O	8
Configuración del servicio SSH en el servidor	14
Instalación y justificación de Docker	18
Desarrollo del Script interactivo para la gestión de reglas de red y firewall en Fedora	21
Rutinas de backup y automatización de las mismas	23
Implementación del servidor MySQL y Apache en docker	26
Gestión y administración de contenedores para despliegue final de aplicaciones	29
Scripts del servidor	34
Credenciales de los servidores y .ova	35
Hoja Testigo	36



Script bash gestor de usuarios y grupos

Para acceder a la descarga del script gestor de usuarios y grupos, ir al final del documento, en la sección “[Scripts del servidor](#)”

Manual de instalación y justificación de Fedora Server

Sistema Operativo elegido: Fedora Server

¿Por qué elegimos este sistema operativo?

Para el desarrollo del proyecto se optó por utilizar Fedora Server como sistema operativo base, fundamentado en criterios técnicos y educativos:

- Actualización constante: Fedora es una distribución conocida por incorporar las versiones más recientes del software y tecnologías en el ámbito Linux, proporcionando un entorno moderno, seguro y alineado con estándares actuales de la industria.
- Enfoque en servidores: Fedora Server está específicamente orientado a entornos de servidor, adecuándose eficazmente a los objetivos del proyecto relacionados con administración de usuarios, grupos y permisos en sistemas multiusuario.
- Facilidad de gestión y administración: Incluye herramientas como Cockpit, una interfaz web intuitiva para la administración del sistema, manteniendo además el potencial de la administración mediante la línea de comandos, facilitando la progresión desde prácticas básicas a complejas.
- Documentación clara y comunidad activa: Cuenta con una comunidad robusta y organizada, así como con una documentación oficial completa, favoreciendo la resolución de problemas y el aprendizaje autónomo.
- Proyección profesional: Fedora constituye la base de Red Hat Enterprise Linux (RHEL), uno de los sistemas operativos más utilizados en ambientes empresariales, ofreciendo experiencia práctica aplicable en contextos profesionales.



Manual Completo de Instalación de Fedora Server

Aplicable a máquinas físicas y virtuales (VirtualBox, VMware)

Parte 1: Requisitos previos

1. Descargar Fedora Server
 - Acceder al sitio oficial: <https://fedoraproject.org/server/download/>
 - Seleccionar:
 - Arquitectura: x86_64
 - Opción: Imagen ISO completa (no netinstall)
2. Preparar el entorno
 - Para instalación en equipo físico:
 - Pendrive mínimo de 8 GB
 - Herramienta para crear USB booteable:
 - Windows: Rufus
 - Linux/macOS: balenaEtcher
 - Para instalación en máquina virtual:
 - Programa de virtualización:
 - VirtualBox
 - VMware Workstation Player

Parte 2: Crear medio de instalación

- En equipo físico (USB booteable):
 - Abrir Rufus.
 - Seleccionar el dispositivo USB y la ISO descargada.
 - Configurar sistema de archivos como FAT32.
 - Seleccionar esquema de partición MBR (BIOS) o GPT (UEFI).
 - Iniciar y esperar la finalización del proceso.
- En máquina virtual (VirtualBox):
 - Crear nueva máquina virtual.
 - Nombre: Fedora Server
 - Tipo: Linux
 - Versión: Fedora (64-bit)
 - Asignar memoria (mínimo 2048 MB, recomendado 4096 MB).
 - Crear disco virtual (VDI, tamaño dinámico, mínimo 20 GB, recomendado 30 GB).
 - Agregar la ISO descargada en configuración de almacenamiento.



Parte 3: Iniciar instalación

En equipo físico:

- Insertar USB, encender PC y seleccionar dispositivo de arranque.

En máquina virtual:

- Iniciar máquina virtual.

En ambos casos seleccionar: Install Fedora Server.

Parte 4: Proceso de instalación

1. Elegir idioma:

- Seleccionar Español (Uruguay / Latinoamérica / España (Según preferencia)).
- Continuar.

2. Configuración del sistema:

- Destino de instalación: seleccionar disco, particionado automático recomendado.
- Configuración de red y nombre de host: activar red, asignar nombre (ejemplo: jurassiserver.local).
- Fecha y hora: América/Montevideo.
- Contraseña usuario root: establecer contraseña segura.
- Usuario adicional: crear y asignar privilegios administrativos (opcional pero recomendable).

3. Comenzar instalación:

Iniciar la instalación y esperar a que finalice.

Parte 5: Finalizar e iniciar el sistema:

- Reiniciar el sistema retirando previamente el medio de instalación.

Parte 6: Primer arranque y configuración post-instalación:

- Iniciar sesión con usuario creado o root.
- Actualizar sistema ejecutando en terminal:

```
sudo dnf update -y
```

Consejos finales:

Fedora Server no incluye entorno gráfico por defecto, administrándose vía terminal o interfaz web Cockpit.

Para mejorar la compatibilidad en VirtualBox, se recomienda instalar Guest Additions.



Instalación y Configuración de la Red en Fedora Server

PASO 1: Instalación de herramientas de configuración de red (nmtui)

Fedora Server incluye por defecto NetworkManager, que permite gestionar interfaces de red de manera eficiente.

Una de las formas más simples y seguras de configurar la red en modo texto es a través de la utilidad nmtui.

En caso de que no esté instalada, puede instalarse ejecutando:

```
sudo dnf install -y NetworkManager-tui
```

Una vez instalada, se puede lanzar el asistente con el comando:

```
nmtui
```

Este menú interactivo permite:

- Editar conexiones
- Activar/desactivar interfaces
- Establecer una IP estática
- Configurar DNS

PASO 2: Configurar la IP estática en el servidor

Para que el servidor sea visible fácilmente desde la red y no cambie su IP entre reinicios, es necesario asignarle una dirección IP estática.

1. Ejecutar:

```
sudo nmtui
```

2. Seleccionar: Editar una conexión
3. Elegir la interfaz de red (por ejemplo ens33, enp0s3, etc.)
4. Cambiar el método de dirección IPv4 a: Manual
5. Ingresar los siguientes valores:
 - Dirección IP: por ejemplo 192.168.1.100/24
 - Puerta de enlace (gateway): por ejemplo 192.168.1.1
 - DNS: por ejemplo 1.1.1.1, 8.8.8.8
6. Aceptar los cambios y luego seleccionar Activar una conexión para reiniciar la interfaz.



PASO 3: Verificar conectividad

Luego de configurar la red, se puede verificar la conectividad con:

```
#Ver estado de las interfaces
ip a
#Probar acceso a internet
ping -c 4 8.8.8.8
#Probar resolución DNS
ping -c 4 google.com
```

También es útil hacer un ping desde otra máquina para confirmar que el servidor es visible en la red.

PASO 4: Configurar los puertos del servidor

Para que el servidor pueda enviar y recibir información a través de la red, es necesario asegurarse de que los puertos de los servicios estén abiertos en el firewall.

Ver puertos abiertos actualmente:

```
sudo firewall-cmd --list-all
```

Abrir puertos comunes:

- SSH (puerto 22):

```
#SSH
sudo firewall-cmd --add-service=ssh --permanent
```

- HTTP (puerto 80) y HTTPS (puerto 443):

```
#HTTP
sudo firewall-cmd --add-service=http --permanent
#HTTPS
sudo firewall-cmd --add-service=https --permanent
```

- Otro puerto personalizado: (por ejemplo, para un servidor de desarrollo)

```
#Puerto 3000
sudo firewall-cmd --add-port=3000/tcp --permanent
```

Aplicar cambios del firewall:

```
sudo firewall-cmd --reload
```



PASO 5: Confirmar que los servicios escuchan en los puertos esperados

Usar ss:

```
sudo ss -tuln
```

- -t: solo conexiones TCP
- -u: también incluye UDP
- -l: solo servicios que están escuchando (no conexiones activas)
- -n: no resuelve nombres (muestra IPs y puertos en número, no en nombre como ssh o http)

Gestión y administración de paquetes del S.O

Introducción

Para llevar adelante el deploy inicial de la aplicación web de JurassiDraft, se optó por instalar y configurar manualmente los paquetes necesarios a través del gestor de paquetes dnf de Fedora Server. Esta decisión se basa en la necesidad de contar con un entorno de ejecución controlado, seguro y ajustado a buenas prácticas de administración de servidores Linux.

Aunque XAMPP podría ser una solución rápida para entornos de desarrollo, presenta importantes limitaciones en servidores como Fedora Server:

- Está diseñado para uso local y entornos gráficos, lo cual no aplica en servidores sin GUI.
- Corre servicios fuera del control de systemd, dificultando la administración de procesos.
- Ejecuta Apache y MySQL como root, una mala práctica en entornos de producción.



- No se integra con el firewall ni las políticas de red de Fedora.
- Por estas razones, se descartó XAMPP en favor de una instalación modular y nativa con dnf, que permite mejor integración, seguridad y control.

Instalación de Apache, PHP y MySQL

A continuación se detalla el procedimiento para instalar LAMP en Fedora Server, junto con las extensiones PHP que podría requerir la aplicación para funcionar a futuro.

AVISO:

La siguiente sección documenta la instalación inicial de Apache, PHP y MySQL directamente sobre Fedora Server, utilizada durante las primeras etapas del proyecto para validar la configuración del entorno LAMP y las dependencias del sistema.

Posteriormente, este método fue reemplazado por una implementación completa mediante contenedores Docker, descrita en la sección [“Gestión y administración de contenedores para despliegue final de aplicaciones”](#), donde se integran Laravel y MySQL dentro de un entorno automatizado con Docker Compose.

PASO 1: Actualizar el sistema

Antes de instalar cualquier componente, es recomendable asegurarse de que el sistema esté completamente actualizado. Esto previene conflictos de dependencias y asegura compatibilidad con las versiones más recientes.

```
sudo dnf update -y
```

PASO 2: Instalar Apache (servidor web)

Apache es el servidor encargado de procesar las peticiones HTTP, servir los archivos HTML, CSS y JS, y ejecutar scripts PHP.

```
sudo dnf install -y httpd
```

Una vez instalado, se activa y configura para que inicie automáticamente con el sistema con:

```
sudo systemctl enable --now httpd
```



PASO 3: Instalar PHP y extensiones necesarias

PHP es el lenguaje backend que se eligió para ser utilizado en el proyecto. Para que funcione correctamente con MySQL y otras funcionalidades del sistema, se instalan también varias extensiones específicas:

```
sudo dnf install -y php php-cli php-mysqlcli php-pdo php-mbstring php-xml
```

Explicación de cada módulo:

- **php**: núcleo de PHP.
- **php-cli**: permite ejecutar scripts desde consola, útil para pruebas o mantenimiento.
- **php-mysqlcli**: permite a PHP conectarse a la base de datos MySQL, esencial para las funciones del sistema.
- **php-pdo**: proporciona una interfaz moderna y segura para trabajar con bases de datos desde PHP.
- **php-mbstring**: permite trabajar correctamente con textos en UTF-8 y caracteres especiales.
- **php-xml**: proporciona soporte para trabajar con XML. Aunque todavía no sabemos si el proyecto va a utilizarlo directamente, se instala como medida preventiva para futuras integraciones.

PASO 4: Instalar MySQL Server

Para gestionar y brindar los servicios de base de datos en nuestro proyecto, se procede a la instalación del motor MySQL Server, utilizando el repositorio oficial de Oracle para Fedora.

Para instalar el mismo, se debe primero descargar el repositorio usando:

```
wget  
https://dev.mysql.com/get/mysql184-community-release-fc42-3.noarch.  
rpm
```

luego, se instala el repositorio con el comando:

```
sudo dnf install mysql184-community-release-fc42-3.noarch.rpm
```

Una vez instalado el repositorio, se debe proceder a instalar MySQL Server, teniendo el repositorio instalado en nuestro servidor, ejecutamos



```
sudo dnf install -y mysql-server
```

Configuración del servidor web Apache en Fedora Server

Apache (httpd) ya fue instalado previamente con dnf, pero solo la instalación podría no ser suficiente, se podría necesitar realizar alguna configuración o ajuste para permitir que el servicio funcione correctamente, sea accesible desde otros equipos, y pueda estar listo para producción.

- Verificar estado del servicio

Se tiene que confirmar que Apache esté corriendo y configurado para iniciarse automáticamente al encender el servidor:

```
sudo systemctl status httpd
```

Se debería ver algo como active (running). Si no está activo:

```
sudo systemctl start httpd
```

Y para que arranque solo en cada reinicio:

```
sudo systemctl enable httpd
```

- Confirmar ruta del directorio raíz (DocumentRoot)

Apache, por defecto en Fedora, sirve los archivos desde:

```
/var/www/html/
```

Eso significa que si se accede a `http://IP_DEL_SERVIDOR`, el navegador va a mostrar el contenido de esa carpeta, entonces los archivos del proyecto tienen que estar copiados en la misma, en caso de que no estén, se deben mover con:

```
sudo cp -r /var/www/html/
```

- Ajustar permisos

Asegurarse de que Apache pueda leer los archivos del proyecto:

```
sudo chown -R apache:apache /var/www/html/jurassidraft  
sudo chmod -R 755 /var/www/html/jurassidraft
```



Configuración y gestión de MySQL Server

Una vez instalado MySQL utilizando el repositorio oficial provisto por Oracle (ver sección de instalación), es fundamental realizar su configuración inicial para garantizar el correcto funcionamiento del sistema, su seguridad y su disponibilidad.

- Verificar el estado del servicio

Luego de la instalación, es necesario asegurarse de que el servicio de MySQL esté activo y habilitado para iniciarse automáticamente con el sistema:

```
sudo systemctl start mysqld  
sudo systemctl enable mysqld
```

Para verificar el estado actual del servicio:

```
sudo systemctl status mysqld
```

Se espera que el estado sea active (running). En caso contrario, se deben revisar los registros del sistema.

- Recuperar la contraseña temporal del usuario root

MySQL genera una contraseña aleatoria temporal para el usuario root durante la instalación. Esta contraseña puede consultarse en el archivo de registro correspondiente:

```
sudo grep 'temporary password' /var/log/mysqld.log
```

El resultado mostrará una línea similar a:

```
[Note] A temporary password is generated for root@localhost:  
Abcdefg123!
```

- Configurar la instalación inicial de MySQL

Se recomienda ejecutar el script interactivo de configuración segura de MySQL, el cual permite modificar parámetros críticos de seguridad, como la contraseña del administrador, y deshabilitar configuraciones inseguras.

```
sudo mysql_secure_installation
```

Durante la ejecución, se solicitará:

- Cambiar la contraseña del usuario root.
- Definir el nivel de seguridad de contraseñas.
- Eliminar usuarios anónimos.
- Deshabilitar el acceso remoto del usuario root.
- Eliminar la base de datos de prueba.
- Recargar las tablas de privilegios.



- Ingresar a MySQL

Una vez finalizada la configuración segura, se puede acceder al cliente MySQL con el siguiente comando:

```
sudo mysql -u root -p
```

Tras ingresar la contraseña correspondiente, se accede al entorno de gestión SQL de MySQL, donde se pueden escribir comandos sql, se recomienda crear una base de datos y un nuevo usuario (no root) para la misma, siguiendo:

```
CREATE DATABASE jurassidraft
CREATE USER 'jurassi_user'@'localhost' IDENTIFIED BY
'una_contraseña_segura';
GRANT ALL PRIVILEGES ON jurassidraft.* TO
'jurassi_user'@'localhost';
FLUSH PRIVILEGES;
```

Se puede salir del cliente con exit y si se quiere probar la conexión con el nuevo usuario, luego de haber salido, se ejecuta desde la terminal:

```
mysql -u jurassi_user -p
```

Si la conexión es exitosa, significa que la base de datos está correctamente configurada para ser utilizada por la aplicación web.

Gestión del firewall y apertura de puertos

Una vez instalados y configurados los servicios del servidor, es necesario garantizar que puedan ser accedidos desde la red. Para ello, se debe permitir explícitamente el tráfico a través de los puertos utilizados, mediante la herramienta de configuración del firewall incorporada en Fedora: firewalld.

- Verificar servicios permitidos por el firewall

Para consultar los servicios actualmente habilitados:

```
sudo firewall-cmd --list-services
```

- Habilitar servicios necesarios para el funcionamiento del servidor

Se deben permitir los siguientes servicios de forma permanente:

```
sudo firewall-cmd --add-service=http --permanent
#Puerto 80 - Servidor web Apache
sudo firewall-cmd --add-service=https --permanent
#Puerto 443 - HTTPS (si aplica)
sudo firewall-cmd --add-service=ssh --permanent
#Puerto 22 - Acceso SSH al servidor
```



Finalmente, se aplica la configuración:

```
sudo firewall-cmd --reload
```

Configuración del servicio SSH en el servidor

PASO 1: Instalar servicio

Para instalar el servicio de ssh en Fedora se usa el comando:

```
sudo dnf install openssh-server
```

En muchos casos ya viene preinstalado, pero si no es así, este comando lo instalará.

PASO 2: Iniciar servicio

El servicio de ssh en fedora se habilita y se inicia con los siguientes comandos:

```
sudo systemctl enable sshd
sudo systemctl start sshd

#Comprobar estado del ssh
systemctl status sshd
```

Luego de realizados estos dos comandos, se puede comprobar el estado del mismo con el comando indicado abajo del comentario. (Si el sistema ya tenía SSH preinstalado, probablemente ya esté corriendo, y se puede omitir este paso).

PASO 3: Configuración del firewall para habilitar SSH en Fedora

Para verificar y, en caso necesario, permitir las comunicaciones a través del servicio SSH en Fedora, se emplean los siguientes comandos:

```
sudo firewall-cmd --list-services
#Para ver los servicios
sudo firewall-cmd --add-service=ssh --permanent
#Para agregar ssh a la lista de servicios que el firewall permite, de
manera permanente
sudo firewall-cmd --reload
#Para aplicar los cambios inmediatamente
```

Si el servicio SSH fue instalado junto con el sistema operativo, es muy probable que ya se encuentre habilitado en el firewall por defecto, por lo que este procedimiento puede no ser necesario.



PASO 4: Revisión de la configuración de SSH

En caso de ser necesario, la configuración del servicio SSH puede revisarse y modificarse accediendo al archivo ubicado en:

```
/etc/ssh/sshd_config
```

Para un uso básico del servicio, no es imprescindible realizar cambios en este archivo. No obstante, en situaciones que requieran ajustes específicos —como cambiar el puerto por defecto, deshabilitar el acceso mediante el usuario root, o aplicar políticas de seguridad más estrictas—, estas modificaciones deben efectuarse en dicho archivo.

PASO 5: Conexión a un servidor remoto mediante SSH

Para establecer una conexión SSH desde otro equipo, se debe ejecutar el siguiente comando en la terminal del cliente:

```
ssh usuario@ip_del_equipo
```

Al realizar la conexión por primera vez, es posible que el sistema muestre un mensaje similar al siguiente:

```
The authenticity of host 'ip_del_equipo' can't be established.  
ED25519 key fingerprint is SHA256:huella_digital.  
This key is not known by any other names.  
Are you sure you want to continue connecting  
(yes/no/[fingerprint])?
```

Ante este mensaje, se debe confirmar la conexión escribiendo “yes”. A continuación, el sistema solicitará la contraseña del usuario remoto. Una vez ingresada correctamente, se establecerá la conexión con el servidor.



PASO 6: Conexión SSH sin contraseña (mediante autenticación con clave pública)

PASO 6.1: Generación del par de claves en Windows

Para establecer una conexión SSH sin necesidad de ingresar contraseña, se puede utilizar un par de claves criptográficas (clave pública y clave privada). Este mecanismo mejora la seguridad y facilita el acceso automatizado a servidores remotos.

En sistemas Windows, se puede generar el par de claves utilizando PowerShell. Para ello, ejecutar el siguiente comando:

```
ssh-keygen -t ed25519
```

Durante el proceso, se solicitará una ubicación para guardar las claves. Se recomienda aceptar la ruta por defecto:

```
C:\Users\TU_USUARIO\.ssh
```

Opcionalmente, se puede establecer una passphrase (contraseña adicional) para proteger la clave privada. Aunque no es obligatorio, se recomienda su uso en entornos donde se requiere un nivel de seguridad elevado.

PASO 6.2: Copiar la clave pública al servidor Linux

Una vez generado el par de claves en el cliente Windows, se debe transferir la clave pública al servidor Linux. Este procedimiento se realiza de forma manual, siguiendo los pasos detallados a continuación:

- **Acceder al servidor Linux mediante SSH**

Inicialmente, se debe establecer una conexión SSH al servidor utilizando usuario y contraseña.

- **Crear el directorio de claves SSH**

Ejecutar el siguiente comando para crear el directorio `.ssh` en el home del usuario (si aún no existe).

```
mkdir -p ~/.ssh
```

- **Obtener la clave pública en Windows**

Desde PowerShell en el equipo cliente, visualizar la clave pública generada previamente con el siguiente comando:

```
cat $env:USERPROFILE\.ssh\id_ed25519.pub
```




- **Copiar la clave pública al servidor**

Volver al servidor Linux y abrir el archivo `authorized_keys` con el editor `nano`:

```
nano ~/.ssh/authorized_keys
```

- **Asignar los permisos adecuados**

Finalmente, asegurarse de que tanto la carpeta como el archivo tengan los permisos correctos, ejecutando:

```
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/authorized_keys
```

Estos pasos permiten que futuras conexiones SSH desde el equipo cliente se realicen sin necesidad de ingresar la contraseña, siempre que se utilice la clave privada correspondiente.

PASO 6.3: Conectarse sin contraseña (o utilizando *passphrase* y clave privada)

Una vez que la clave pública ha sido correctamente copiada al servidor y configurados los permisos, se puede establecer la conexión SSH desde el cliente sin necesidad de ingresar la contraseña del usuario remoto.

Para ello, desde PowerShell (o cualquier terminal) en el equipo cliente, ejecutar:

```
ssh usuario@ip_del_equipo
```

Si el proceso se completó correctamente y no se configuró una *passphrase*, el sistema establecerá la conexión directamente, sin solicitar credenciales.

En caso de haberse definido una *passphrase* durante la generación de la clave privada, se pedirá ingresarla para completar la autenticación.

PASO 6.4: Deshabilitar el inicio de sesión por contraseña (opcional pero altamente recomendable)

Para reforzar la seguridad del servidor, se recomienda desactivar el inicio de sesión SSH mediante contraseña y permitir únicamente la autenticación a través de clave pública.



Pasos a seguir en el servidor Linux:

1. Editar el archivo de configuración de SSH:

```
sudo nano /etc/ssh/sshd_config
```

2. Modificar o agregar las siguientes directivas:

```
PasswordAuthentication no  
PermitRootLogin no
```

- PasswordAuthentication no, desactiva el uso de contraseñas para la autenticación SSH.
- PermitRootLogin no, impide el inicio de sesión directo como usuario root, lo que añade una capa adicional de seguridad.

3. Reiniciar el servicio SSH para aplicar los cambios.

```
sudo systemctl restart sshd
```

A partir de este momento, el servidor solo aceptará conexiones desde clientes que presenten una clave privada autorizada. Esta práctica reduce significativamente el riesgo de ataques de fuerza bruta y accesos no autorizados.

Instalación y justificación de Docker

Justificación

Dentro del desarrollo del sistema JurassiDraft, que busca gestionar digitalmente partidas del juego de mesa Draftosaurus mediante una plataforma web, se requiere un entorno de desarrollo estable, reproducible y seguro. Aquí es donde entra Docker.

Docker es una plataforma de virtualización ligera que permite armar paquetes de aplicaciones y sus dependencias en contenedores, garantizando que funcionen de forma idéntica sin importar el entorno en el que se ejecuten.

Ahora, nos preguntamos por qué usar docker, estas son las respuestas que encontramos:

- Entornos consistentes: Todos los integrantes del equipo podemos trabajar con la misma configuración sin conflictos entre versiones de PHP, MySQL o Apache.



- Facilidad para pruebas y despliegue: Se pueden levantar entornos aislados para probar nuevas funcionalidades sin afectar el proyecto principal.
- Aislamiento: Permite tener separados los servicios del backend (PHP, MySQL) del frontend, simplificando la depuración y el mantenimiento.
- Escalabilidad futura: Si se busca o se desea desplegar el sistema en otro servidor, en una pc para testing o directamente en la nube, Docker facilita la transición a diversas plataformas, sin importar el cómo y el cuando.
- Integración continua: Se puede automatizar el testing o el despliegue con herramientas como GitHub Actions, integradas a los contenedores.

Instalación de Docker en Fedora

Docker no se incluye por defecto en los repositorios principales de Fedora, por lo que es necesario configurar el repositorio oficial de Docker e instalar los componentes necesarios manualmente. A continuación, se detalla el procedimiento completo y actualizado.

PASO 1: Actualizar el sistema

Antes de comenzar, se recomienda actualizar todos los paquetes del sistema para evitar conflictos de dependencias:

```
sudo dnf update -y
```

PASO 2: Instalar dnf-plugins-core y configurar el repositorio oficial

Primero se debe instalar el paquete dnf-plugins-core, que permite gestionar repositorios externos con dnf.:

```
sudo dnf install -y dnf-plugins-core
```

Luego, se añade el repositorio oficial de Docker para Fedora:

```
sudo dnf config-manager --add-repo  
https://download.docker.com/linux/fedora/docker-ce.repo
```

Nota: En algunos sistemas puede aparecer como dnf-3, pero dnf funciona correctamente en versiones modernas de Fedora.



PASO 3: Instalar Docker Engine

Para instalar la última versión estable de Docker y sus componentes asociados, ejecutar:

```
sudo dnf install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin
```

PASO 4: Iniciar y habilitar Docker

Una vez instalado, el servicio de Docker no se inicia automáticamente. Para arrancarlo y hacer que se ejecute al iniciar el sistema, ejecutar:

```
sudo systemctl enable docker  
sudo systemctl start docker
```

PASO 5: Verificar instalación

Para comprobar que Docker funciona correctamente, ejecutar el siguiente comando:

```
sudo docker run hello-world
```

Este comando descarga una imagen de prueba desde Docker Hub, la ejecuta en un contenedor y muestra un mensaje de confirmación. Si aparece dicho mensaje, la instalación fue exitosa.

PASO 6: (Opcional) Usar Docker sin sudo

Docker crea un grupo llamado docker, pero no agrega automáticamente al usuario actual. Si se quiere ejecutar Docker sin anteponer sudo, se debe ejecutar:

```
sudo usermod -aG docker $USER
```

Luego, es necesario cerrar sesión y volver a iniciar sesión para que el cambio tenga efecto. (Agregar un usuario al grupo docker le otorga privilegios equivalentes a root, por lo tanto, debe hacerse únicamente en entornos controlados o de desarrollo.)

PASO 7: Confirmar versiones instaladas

```
docker --version  
docker compose version
```



SEGUNDA ENTREGA

Desarrollo del Script interactivo para la gestión de reglas de red y firewall en Fedora

Introducción

Programamos un script interactivo hecho en bash, usando Dialog, para gestionar el firewall y las reglas de red usando FirewallD. La idea principal fue simplificar el manejo del firewall mediante una interfaz relativamente intuitiva, de forma que cualquier usuario con conocimientos básicos logre administrar zonas, reglas e interfaces sin tener que memorizar comandos largos o complicados.

Herramientas utilizadas

- FirewallD: Sistema de gestión de firewall dinámico presente por defecto en todas las distribuciones de la familia RHEL, aunque FirewallD está disponible en casi todas las distribuciones existentes, siendo necesario solamente instalarlo.
- Dialog: Utilidad que posibilita la construcción de interfaces basadas en texto dentro de la terminal. Nos fue de gran utilidad al ofrecer menús y cuadros de diálogo más amigables con el usuario final.

Funciones implementadas

- 1. Listar zonas disponibles y cambiar la zona de una interfaz:**
Permite al usuario seleccionar una zona de entre las disponibles (ej. drop, public, etc.) y asignarla a una interfaz como enp0s3. Esto afecta directamente el comportamiento del tráfico entrante en esa interfaz.
- 2. Bloquear completamente una IP o red:**
Agrega una regla de tipo rich rule que bloquea todo el tráfico proveniente de la IP o red especificada, usando la acción drop.
- 3. Permitir o bloquear un puerto para una IP o red concreta:**
Permite aplicar reglas específicas por puerto y protocolo (TCP o UDP) según el origen del tráfico. Se puede permitir o denegar dicho acceso.
- 4. Eliminar reglas asociadas a una IP:**
Función añadida para limpiar todas las reglas relacionadas con una IP. Busca entre las reglas rich existentes y remueve aquellas que correspondan.
- 5. Selector de modo (runtime o permanent):**
Cada acción permite elegir si la regla se aplica de forma temporal (solo en memoria hasta reinicio) o permanente (se mantiene al reiniciar).



Pruebas realizadas

Se realizaron diversas pruebas para validar el funcionamiento del script:

- Se asignó la zona public a la interfaz enp0s3, permitiendo tráfico normal.
 - Se cambió a la zona drop y se comprobó que no se aceptaba ningún tipo de conexión entrante.
 - Se bloqueó completamente el acceso desde una IP específica y se verificó desde otra máquina con Windows mediante SSH.
 - Se bloqueó el puerto 3306 (MySQL) solo para dos IPs, y se comprobó que otras IPs mantenían el acceso.
 - Se permitió el puerto 22 (SSH) selectivamente a una IP, mientras se bloqueó para otra.
 - Se removieron reglas usando la función "eliminar reglas por IP" y se confirmó que se restablece la conectividad.
- Se verificó el funcionamiento de los modos runtime y permanent, y la necesidad de usar --reload tras cambios permanentes.

Instalación de dependencias

Para ejecutar el script es necesario tener dialog instalado. El siguiente comando lo instala en Fedora:

```
sudo dnf install -y dialog
```



Rutinas de backup y automatización de las mismas

Introducción

Un backup o copia de seguridad es, en términos simples, una copia de los datos importantes que se guardan en otro lugar para poder recuperarlos en caso de pérdida o falla. Las rutinas de backup no son otra cosa que la planificación organizada de esas copias: qué se guarda, con qué frecuencia y dónde.

El objetivo principal es garantizar que, ante cualquier imprevisto, la información (o la mayor cantidad de información posible) no se pierda de forma definitiva.

En teoría, una buena política de respaldo debe considerar:

- Frecuencia de las copias: pueden ser diarias, semanales, mensuales, según la importancia y el nivel de cambio de los datos.
- Tipos de respaldo:
 - Completo: se guarda todo, ocupa más espacio pero es el más seguro.
 - Incremental: guarda solo lo que cambió desde el último backup, más rápido y eficiente.
 - Diferencial: guarda lo que cambió desde el último backup completo, un punto medio.
- Diversificación de medios: no basta con tener todo en un mismo disco; lo ideal es seguir la regla 3-2-1:
 - 3 copias de la información.
 - Esas 3 copias, tenerlas en 2 medios diferentes (por ejemplo, disco y nube).
 - Al menos 1 copia fuera de la ubicación principal.

Implementación básica

En nuestro proyecto, la rutina definida combina:

- Backup diario de la base de datos y archivos web (script backup.sh).
- Registro en logs para auditoría.
- Sincronización remota semanal hacia otro servidor mediante rsync (script sync_backups.sh).
- Automatización con cron en horarios de baja carga (03:00 y 04:00).
- Interfaz dialog para ejecutar backups manuales, ver historiales y configurar horarios de cron.

Consideramos que para el alcance y el tamaño del proyecto es más que suficiente.



Medios de respaldo a mediano/largo plazo

En nuestro proyecto, como respaldo a mediano/largo plazo pensamos en enviar los backups a un servidor externo, que podría oficial como NAS (Network Attached Storage), pero analizamos e investigamos sobre otras opciones, tales como:

- Cintas Magnéticas (LTO, Linear Tape-Open)
 - Pros
 - Muy alta durabilidad (10-30 años).
 - Bajo costo por terabyte si se consideran grandes volúmenes de datos.
 - Estándar en empresas y bancos para archivo e histórico.
 - Contras
 - No es práctico para proyectos chicos, o de bajo volumen de datos.
 - Un mal ambiente (humedad, frío o calor extremo) podría dañar o percutir más rápido las cintas.
- Almacenamiento en la Nube (AWS S3 Glacier, Google Coldline, Azure Archive)
 - Pros
 - Alta disponibilidad global.
 - Se paga por uso.
 - Muy bajas posibilidades de pérdidas de datos.
 - Contras
 - Costo de mantenimiento recurrente (siempre hay que pagar aunque no se guarde nada si se quiere mantener los archivos).
 - En algunos casos, para obtener el archivo una vez guardado se puede demorar horas.



Plan práctico de respaldos a largo plazo

Definimos un plan a futuro para mantener el orden y la estabilidad en el sistema, pensado a largo plazo:

- Mantener backups diarios en disco local.
- Replicar semanalmente en un NAS o servidor remoto.
- Mensualmente, anualmente, o cuando el volumen de datos sea el suficiente como para justificarlo, generar un tape backup LTO o copia en la nube.
- Revisar anualmente las políticas de retención:
 - Copias diarias: retener 30 días.
 - Copias semanales: retener 6 meses.
 - (Si se realizan) Copias en nube: retener 5-10 años

Alta disponibilidad

Alta disponibilidad significa que un servicio sigue en funcionamiento aún ante fallos de hardware, software o red. No se trata solo de tener copias, sino de intentar minimizar el tiempo que el usuario no puede acceder al servicio.

Estrategias

- Nivel base (lo que hemos logrado hasta hoy)
 - Contenedores de docker con --restart=always
 - Backups periódicos y posibilidad de restauración manual.
- Nivel Intermedio
 - Replicación de base de datos MySQL.
 - Balanceador de carga con Nginx o HAProxy para los servicios web.
 - Monitorización y alertas (ejemplo: Zabbix).
- Nivel Avanzado (Pensando a nivel enterprise)
 - Escalado automático según carga.
 - Storage distribuido o completamente en la nube.
 - Failover geográfico entre datacenters.

Plan práctico

- Corto plazo: Mantener las 2 VMs sincronizadas y con cron configurado.
- Mediano plazo: Investigar implementación de Zabbix y replicación de MySQL
- Largo plazo: Investigar implementación de servicios en la nube como S3 o similares.



Implementación del servidor MySQL y Apache en docker

Introducción

Siguiendo el procedimiento de instalación detallado previamente ([Instalación y justificación de Docker](#)), pasamos ahora a la implementación de los servicios de base de datos y servidor web mediante contenedores Docker.

AVISO:

La presente sección corresponde a una fase intermedia del proyecto, donde se implementan por separado los servicios de MySQL y Apache + PHP dentro de contenedores individuales, con el objetivo de validar su funcionamiento en Docker. Esta implementación fue posteriormente reemplazada y mejorada en la etapa final del proyecto, donde ambos servicios se integraron y automatizaron mediante Docker Compose, descrito en la sección “[Gestión y administración de contenedores para despliegue final de aplicaciones](#)”.

Uso de Docker Hub

Para la implementación recurrimos a Docker Hub, plataforma oficial de distribución de imágenes de contenedores. Docker Hub ofrece repositorios mantenidos tanto por la comunidad como por proveedores oficiales, lo cual garantiza la actualización y seguridad de las imágenes.

En este caso, se seleccionaron imágenes oficiales para asegurar compatibilidad y soporte a largo plazo:

- mysql:oraclelinux9: Servidor de base de datos MySQL.
- php:8.4-apache: Servidor Apache con soporte integrado para PHP.



Implementación de MySQL

Luego de haber bajado de Docker Hub la imagen del contenedor con el comando

```
docker pull mysql:oraclelinux9
```

Levantamos el contenedor con el comando:

```
docker run -d \  
  --name mysql \  
  -e MYSQL_ROOT_PASSWORD=JurassiCode \  
  -p 3306:3306 \  
  --restart=always \  
  mysql:oraclelinux9
```

- Se usó -d para que el contenedor funcionara en segundo plano
- Se definió el nombre del contenedor como mysql.
- La contraseña del usuario root se estableció mediante variable de entorno.
- El puerto 3306 se expuso para permitir conexiones externas.
- La política --restart=always asegura que el servicio se reinicie automáticamente en caso de caída o reinicio del servidor.
- No se mapearon volúmenes, ya que para esta primera etapa se decidió que los datos residan directamente en el contenedor, simplificando la gestión y confiando en los backups completos del sistema.

Posteriormente se creó la base de datos draftosaurus y se importaron las tablas desde un archivo draftosaurus.sql mediante:

```
docker exec -i mysql \  
sh -c 'exec mysql -uroot -p"JurassiCode" draftosaurus' <  
/home/jurasssiuser/PROYECTO/dump/draftosaurus.sql
```



Implementación de Apache + PHP

Luego de haber bajado de Docker Hub la imagen del contenedor con el comando

```
docker pull php8.4-apache
```

Se configuró de la siguiente forma:

```
docker run -d \  
  --name apache_php \  
  -p 8080:80 \  
  -v /proyecto/docker:/var/www/html \  
  --restart=always \  
  php:8.4-apache
```

- Se expuso el puerto 8080 en el host, redirigido al puerto 80 del contenedor.
- Mediante el parámetro -v, se vinculó la carpeta /proyecto/docker del host con la ruta /var/www/html del contenedor, permitiendo que el desarrollo y la actualización de archivos web se realicen directamente en el servidor, sin necesidad de entrar en el contenedor.

En la carpeta /proyecto/docker se implementó un archivo index.php con Bootstrap, mostrando un mensaje de bienvenida y la hora del servidor, validando así la correcta integración entre Apache y PHP.



TERCERA ENTREGA

Gestión y administración de contenedores para despliegue final de aplicaciones

El entorno final se compone de dos servicios principales:

- **Aplicación Laravel (Backend + Frontend)**
- **Base de datos MySQL**

Ambos se ejecutan en contenedores conectados mediante una red interna.

El servicio app (Imágen bitnami/laravel) se encarga de servir la aplicación web, mientras que el servicio db (Imágen mysql) gestiona todo el sistema de base de datos.

Los datos de la base se almacenan en un **volumen persistente** para evitar su pérdida ante reinicios o actualizaciones.

El sistema se desplegó sobre Fedora Server.

Antes de iniciar el montaje de los contenedores, se instaló y configuró Docker siguiendo los pasos mencionados más arriba, en la sección: “Instalación y justificación de Docker”.

Dentro del directorio ~/jurassidraft/ se organizó el entorno con la siguiente estructura:

```
jurassidraft/  
├─ docker-compose.yml  
├─ mysql/  
│   └─ init/  
│       └─ jurassidraft.sql  
└─ web/  
    ├─ .env  
    ├─ artisan  
    ├─ app/  
    ├─ routes/  
    ├─ resources/  
    └─ ...
```



Breve descripción de los subdirectorios y archivos

- **docker-compose.yml**: define los servicios de la aplicación y la base de datos, redes y volúmenes.
- **mysql/init/jurassidraft.sql**: contiene el dump inicial de la base de datos, importado automáticamente la primera vez que se crea el volumen.
- **web/**: contiene el código fuente del proyecto Laravel, incluyendo controladores, vistas, modelos y configuración.

El archivo de despliegue (docker-compose.yml) contiene la siguiente configuración:

```
services:
  db:
    image: mysql
    container_name: jurassidraft-mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: JurassiCode
      MYSQL_DATABASE: jurassidraft
      MYSQL_USER: jurassiuser
      MYSQL_PASSWORD: jurassicode
    volumes:
      - db_data:/var/lib/mysql
      - ./mysql/init:/docker-entrypoint-initdb.d:ro,Z
    networks: [jurassinet]

  app:
    image: bitnami/laravel
    container_name: jurassidraft-laravel
    user: root
    restart: always
    depends_on: [db]
    ports:
      - "8000:80"
    environment:
      LARAVEL_DATABASE_TYPE: mysql
      LARAVEL_DATABASE_HOST: db
      LARAVEL_DATABASE_PORT_NUMBER: 3306
      LARAVEL_DATABASE_NAME: jurassidraft
      LARAVEL_DATABASE_USER: jurassiuser
      LARAVEL_DATABASE_PASSWORD: jurassicode
    command: ["bash", "-c", "chmod -R 777 /app && php artisan serve --host=0.0.0.0 --port=80"]
```



```
volumes:
  - ./web:/app:Z
networks: [jurassinet]

volumes:
  db_data:

networks:
  jurassinet:
    driver: bridge
```

En este caso, optamos por no usar un dockerfile, ya que las imágenes utilizadas ya contienen las configuraciones necesarias para ejecutar el proyecto. Sin embargo es importante aclarar que previo a la creación de los contenedores, hay que instalar dependencias de /web, y son prerequisites dentro del propio servidor:

- Composer + PHP
- Node.js
- Git

Previamente, antes de crear los contenedores, se debe clonar el repositorio de GitHub del proyecto, con el siguiente comando:

```
cd ~/jurassidraft/web/
git clone https://github.com/JurassiCode/PROYECTO.git
```

Luego, tenemos que ejecutar el siguiente comando para instalar dependencias del proyecto:

```
cp .env.example .env && composer install && npm install && npm run
build && php artisan key:generate
```

Luego, hay que editar el .env (vim ~/jurassidraft/web/.env) y cambiar las variables correspondientes a la base de datos:

```
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=jurassidraft
DB_USERNAME=jurassiuser
DB_PASSWORD=jurassicode
```



Sin esos comandos ejecutados, el contenedor de Laravel no puede iniciar correctamente.

En caso de no tener node.js instalados ni composer ni git, se instalan con los siguientes comandos:

Node.js

```
curl -fsSL https://rpm.nodesource.com/setup_20.x | sudo bash -  
sudo dnf install -y nodejs
```

Composer + PHP

```
sudo dnf install -y php-cli php-json php-mbstring php-xml php-zip php-pdo  
php-tokenizer composer
```

Git

```
sudo dnf install -y git
```

Una vez ejecutados los comandos previos y configuradas las dependencias, se procede a levantar el entorno mediante Docker Compose.

```
cd ~/jurassidraft  
docker compose up -d
```

Durante este proceso:

- Docker descarga las imágenes necesarias (mysql y bitnami/laravel) si no habían sido descargadas previamente.
- Se crean los contenedores y la red interna jurassinet.
- El contenedor **MySQL** ejecuta automáticamente el archivo `mysql/init/jurassidraft.sql`, el cual inicializa la base de datos con la estructura y datos definidos por el proyecto.
Este proceso solo se realiza **la primera vez** que se crea el volumen `db_data`.

Una vez iniciado todo el entorno, el sistema queda accesible desde el navegador web en la dirección:

`http://IP_DEL_SERVIDOR:8000`

Los datos de la DB se almacenan dentro del volumen `db_data`, el cual se mantiene persistente incluso si los contenedores se detienen o el servidor se reinicia.



De este modo, no es necesario importar el dump nuevamente, evitando pérdida de información o reconfiguraciones innecesarias.

Por otro lado, los archivos de la aplicación Laravel (código fuente, vistas, controladores, hojas de estilo, scripts, etc.) se almacenan fuera de los contenedores, en el directorio `~/jurassidraft/web` del servidor.

Este directorio se monta dentro del contenedor de la aplicación mediante la instrucción del docker compose:

```
volumes:  
  - ./web:/app:Z
```

Gracias a este mapeo, cualquier modificación realizada en los archivos del proyecto (por ejemplo, cambios en las vistas Blade, controladores, etc.) se refleja inmediatamente dentro del contenedor sin necesidad de reconstruir la imagen ni reiniciar el entorno.

Además, el contenido del proyecto se mantiene completamente persistente, ya que reside en el sistema de archivos del servidor anfitrión, no dentro del contenedor.

Ambos contenedores cuentan con la directiva `restart: always`, garantizando que, en caso de reinicio del sistema operativo o del servicio Docker, el entorno completo (incluyendo la aplicación Laravel y la base de datos) se levanten automáticamente sin intervención manual.

Administración y mantenimiento

Para la administración cotidiana del entorno se utiliza el script de docker, accesible desde los scripts del servidor, en él se puede:

- Levantar ambos contenedores
- Apagar ambos contenedores
- Levantar SOLO la base de datos
- Consultar logs de ambos contenedores

En caso de querer consultar el estado de ambos contenedores, ejecutar:

```
docker ps
```



En caso de querer acceder a los contenedores, una vez encendidos, ejecutar:

Para Laravel:

```
docker exec -it jurassidraft-laravel bash
```

Para MySQL:

```
docker exec -it jurassidraft-mysql bash
```

En caso de querer eliminar ambos contenedores, con todos sus datos, ejecutar:

```
docker compose down -v
```

Scripts del servidor

En esta entrega, se modificaron todos los scripts, implementando el uso de logs para cada acción realizada mediante los mismos. Además, se crearon dos nuevos scripts, menu.sh, ubicado en ~/scripts/menu.sh, que cumple el fin de ser el que “concentra” o “agrupa” todos los scripts. Desde el mismo además, se pueden consultar los logs de todo lo realizado mediante los otros scripts.

El siguiente script creado es docker.sh, usado para gestionar todo lo relacionado al servicio de Docker y contenedores, desde el mismo se pueden realizar acciones como: Levantar ambos contenedores del proyecto, apagar ambos contenedores del proyecto, levantar y apagar SOLO el contenedor de la base de datos y consultar logs de ambos contenedores.

Para acceder a la descarga de los scripts del servidor, acceder al siguiente link:

[📁 Administración de sistemas operativos](#)

En caso de querer acceder a los scripts dentro del propio servidor, acceder al servidor principal (no el que recibe los backups), a la carpeta ~/scripts. Dentro de la misma se encuentran todos los scripts correspondientes, requeridos durante el transcurso del proyecto.



Credenciales de los servidores y .ova

El archivo .ova con ambos servidores (Servidor principal y servidor que recibe archivos de respaldo) se encuentra tanto en el pendrive entregado en coordinación, como en el siguiente link:

Administración de sistemas operativos

- Las credenciales de los servidores son las siguientes:
- Para ambos servidores, el usuario es: jurassiuser
- Para ambos servidores, la contraseña del usuario es: jurassicode
- Para ambos servidores, la contraseña del usuario root es: JurassiCode
- En caso de ser necesarios, el usuario y contraseña del servidor MySQL son:
 - Para usuario root, contraseña JurassiCode
 - Para el usuario jurassiuser, la contraseña es jurassicode
 - La base de datos utilizada es jurassidraft



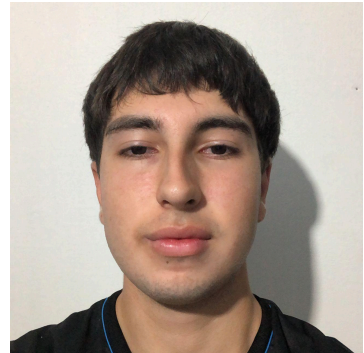
Hoja Testigo

Los integrantes del grupo certificamos que la presente carpeta contiene todo el material solicitado para la evaluación del proyecto en la asignatura Administración de sistemas operativos, cumpliendo con los lineamientos establecidos por la institución.



FIANZA, IGNACIO

COORDINADOR



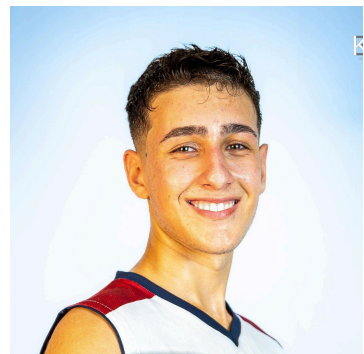
BENÍTEZ, SEBASTIÁN

SUBCOORDINADOR



FLEITAS, JOAQUÍN

INTEGRANTE 1



PAZ, TOMÁS

INTEGRANTE 2