
Proyecto Intermodular

DESPLIEGUE DE BITKEYS

**ALEJANDRO RICO SÁNCHEZ
ADRIÁN GUTIÉRREZ GALVAÑ**

26/02/2026



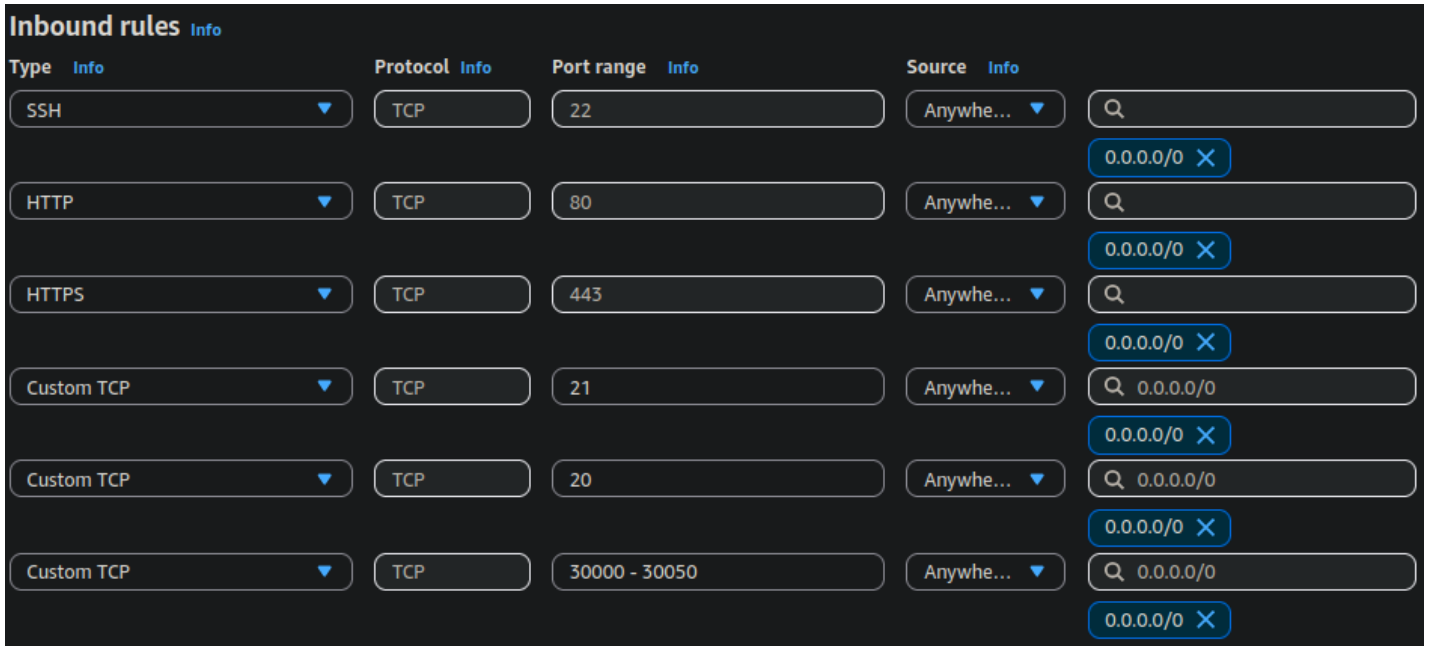
Índice

1. Instancia AWS.....	3
1.1. Grupo de seguridad.....	3
1.2. Instancia.....	3
1.3. IP Elástica.....	4
2. Usuarios.....	5
2.1. Usuarios, directorios y permisos.....	5
2.2. Mensaje de bienvenida SSH.....	5
3. Configuración SSH.....	6
3.1. Claves.....	6
3.2. Seguridad.....	7
4. Certificados para HTTPS.....	8
4.1. LetsEncrypt.....	8
5. Apache.....	9
5.1. Virtual Hosts.....	9
6. Servidor FTP.....	11
6.1. vsftpd.conf.....	11
6.2. vsftpd.userlist.....	12
6.3. Comprobación vsftpd.....	12
7. Despliegue de la aplicación.....	13
7.1. APP.....	13
7.2. Test.....	21
7.3. Backup.....	23
8. Arquitectura escalable AWS.....	24
8.1. Subred VPC.....	24
8.2. Load Balancer.....	25
8.3. Capa de datos.....	26
8.4. Capa de aplicación.....	27
8.5. Despliegue.....	28

1. Instancia AWS

1.1. Grupo de seguridad

Reglas de entrada del grupo de seguridad web-ftp-project-sg, que se utilizará para la instancia EC2 que contenga la aplicación.

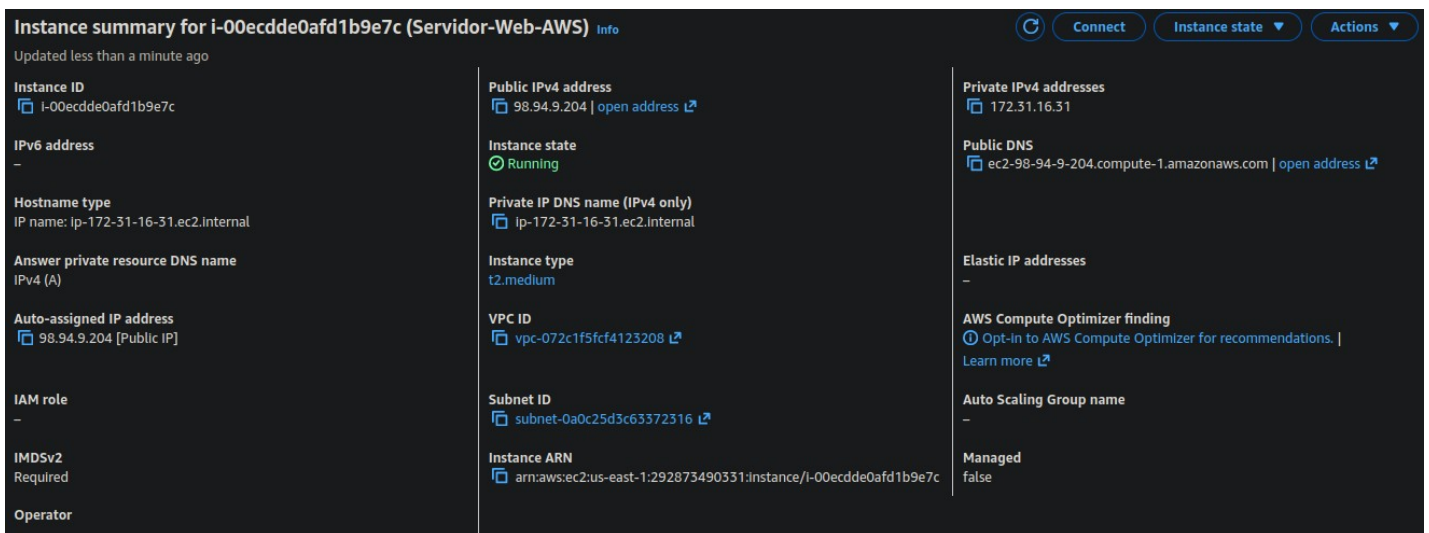


The screenshot shows the 'Inbound rules' configuration for a security group. It lists six rules, each with a Type, Protocol, Port range, and Source. The Source for all rules is 'Anywhere...'. The rules are:

Type	Protocol	Port range	Source
SSH	TCP	22	Anywhere...
HTTP	TCP	80	Anywhere...
HTTPS	TCP	443	Anywhere...
Custom TCP	TCP	21	Anywhere...
Custom TCP	TCP	20	Anywhere...
Custom TCP	TCP	30000 - 30050	Anywhere...

1.2. Instancia

Creamos una instancia EC2 llamada Servidor-Web-AWS en la que ponemos ese grupo de seguridad y un par de claves .pem generado.



The screenshot shows the 'Instance summary' for the EC2 instance 'i-00ecdde0afd1b9e7c (Servidor-Web-AWS)'. The instance is in a 'Running' state. Key details include:

Property	Value
Instance ID	i-00ecdde0afd1b9e7c
Public IPv4 address	98.94.9.204 open address
Private IPv4 addresses	172.31.16.31
Instance state	Running
Private IP DNS name (IPv4 only)	ip-172-31-16-31.ec2.internal
Instance type	t2.medium
VPC ID	vpc-072c1f5fcf4123208
Subnet ID	subnet-0a0c25d3c63372316
Instance ARN	arn:aws:ec2:us-east-1:292873490331:instance/i-00ecdde0afd1b9e7c
Auto-assigned IP address	98.94.9.204 [Public IP]
Public DNS	ec2-98-94-9-204.compute-1.amazonaws.com open address
Managed	false


1.3. IP Elástica

Generamos una IP elástica y la asociamos con la instancia EC2. La IP de nuestro EC2 es: 3.213.82.57.

Elastic IP address: 3.213.82.57

Resource type
Choose the type of resource with which to associate the Elastic IP address.

Instance
 Network Interface

 If you associate an Elastic IP address with an Instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. [Learn more](#)

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

Instance

Private IP address
The private IP address with which to associate the Elastic IP address.

Reassociation
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.

Allow this Elastic IP address to be reassociated

2. Usuarios

2.1. Usuarios, directorios y permisos

Creamos los usuarios app y backup y la estructura de directorios necesaria. Para ello pondremos los siguiente comandos:

app

```
sudo adduser --disabled-password --gecos "" app
sudo mkdir -p /home/app/ftp/www
sudo mkdir -p /home/app/logs
sudo chown -R app:app /home/app/ftp/www
sudo chown -R app:app /home/app/logs
sudo chmod 755 /home/app/ftp/www
```

backup

```
sudo adduser --disabled-password --gecos "" backup
sudo mkdir -p /home/backup/ftp/fixers
sudo chown -R backup:backup /home/backup/ftp/fixers
sudo chmod 755 /home/backup/ftp/fixers
```

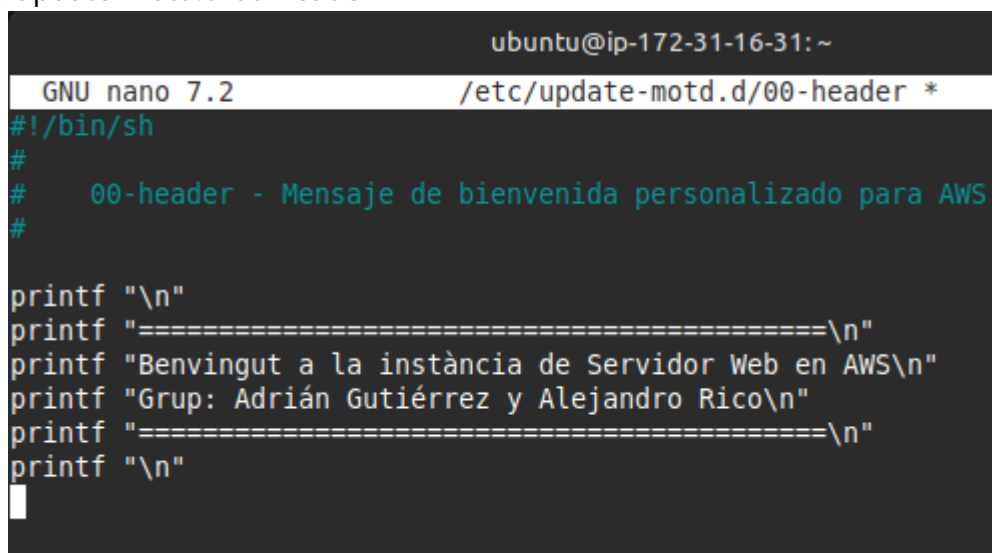
test

```
sudo mkdir -p /home/app/ftp/test
sudo chown -R app:app /home/app/ftp/test
```

2.2. Mensaje de bienvenida SSH

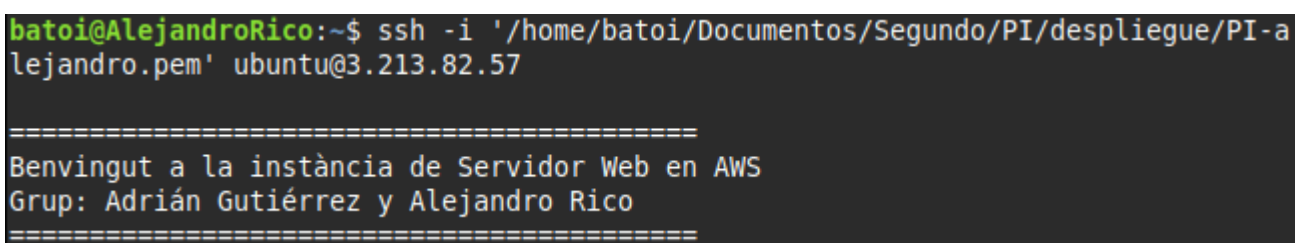
Configuramos el mensaje de bienvenida del usuario ubuntu al conectarse por SSH a la instancia. Para ello editamos e archivo que se encuentra en la siguiente ruta:

```
sudo nano /etc/update-motd.d/00-header
```



```
ubuntu@ip-172-31-16-31: ~
GNU nano 7.2 /etc/update-motd.d/00-header *
#!/bin/sh
#
# 00-header - Mensaje de bienvenida personalizado para AWS
#
printf "\n"
printf "=====\n"
printf "Benvingut a la instància de Servidor Web en AWS\n"
printf "Grup: Adrián Gutiérrez y Alejandro Rico\n"
printf "=====\n"
printf "\n"
```

Nos desconectamos y volvemos a conectar para ver si funciona correctamente



```
batoi@AlejandroRico:~$ ssh -i '/home/batoi/Documentos/Segundo/PI/despliegue/PI-a-alejandro.pem' ubuntu@3.213.82.57
=====  
Benvingut a la instància de Servidor Web en AWS  
Grup: Adrián Gutiérrez y Alejandro Rico  
=====
```

3. Configuración SSH

3.1. Claves

Añadimos las tres claves públicas (Alejandro, Adrián y Rubén) al archivo de configuración `authorized_keys` para que podamos conectarnos a la instancia.

Para generar las claves públicas se han extraído a partir de los archivos `.pem` que contienen la clave privada con el comando `ssh-keygen -y -f 'Privada.pem' > 'Publica.pub'`

Clave pública Alejandro:

```
ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAQDOqJcEkd9jgnip/+aF/m2Ua7xVTtWCWZDFTdGo8HD6ADZxS8gqV38qY/YuLapp5qoRySAQsKeMZDWBGRrM8ZuUp9OWfmg63k8dRjMy5GEiJHs6748Kj9FiaTp9/PePthuECYQLAWLAtyCnKit+PimufaDdaJl4sK84Fz4rY8tHyKcxaAvfD7i61oWPph/7EAnS34xTWZcF7rUvR0dL3W4MZKDr5gutSRW6GxpvLaS3m5zlwetxTIgjOOJqM445ir2ZCmosXmxiKSqgDg4GAECUm0W2g+/19KKYXp/xBpvNLKvs9oHEWllegtgB0l4IzIk0HNealmaxbDlBDpBHhCHv
```

Clave pública Adrián:

```
ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAQADH3/mLPH+QScQxLdFCp+j1Zp52EfrIujq4qEw7g3luu5SOWCsxuGpaMUN8yxSwCgEAg4BbBLU5EijhHUcV6CSQyWR8WRHMI4w5GlvbLYBNaR+uCCABfLLbTterw2/4kzKWLM9KUZPq1204ulALDGLh/3eLoJGtFKwcvDjfp+zuoH1GGJrP3RdpfIT2E+ovNQGo69xgH+hUueKxOvjRMKbs0FB3CABsDajuMIulRnFxaowa/HofalqOaMaoqWXeqq9Ow+im1qAmZnKAl5oz7nHJA1Njm105VutBylahXr6q4BMJMrPXuJMoyNHVYy509eJ0TYvKOVVSPFtsnZTXowP
```

Clave pública Rubén:

```
ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAQADGQVK40w8lbCmO/vWRelnC64bpYpQ5EXbyIWiGG8Xx7ir0gXluw6Kp3Vakm2xJp0KzpD8d4guYIBr038Vvyxe/wR69ClSKVMEEWs1MfyIuNv6J5V9kwFAHZlG/IrJ5CaJVChnKxPpeiP5n08jKUZK+zsA+6SlgFM0J9XtmTGNg0g4skqkC9PZ8f4omOBobZ0yZ1dxi1Czb1HgzLDlnyYlWi4Mdz6uB0pD4G6J/oBEmv8Rx1CILDG3me45dT7TLLJ04g3vmVr2qPKGmPMiSeoPFKbYJU5P6uSr6okx2jEXHCF/8LhKxmjybh3liwGU7t3vX6PUyEGDoQqz0pfhbX6mr
```

Modificamos el archivo `authorized_keys` para dejarlo de la siguiente manera:

```
ubuntu@ip-172-31-16-31:~$ cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDOqJcEkd9jgnip/+aF/m2Ua7xVTtWCWZDFTdGo8HD6ADZxS8gqV38qY/YuLapp5qoRySAQsKeMZDWBGRrM8ZuUp9OWfmg63k8dRjMy5GEiJHs6748Kj9FiaTp9/PePthuECYQLAWLAtyCnKit+PimufaDdaJl4sK84Fz4rY8tHyKcxaAvfD7i61oWPph/7EAnS34xTWZcF7rUvR0dL3W4MZKDr5gutSRW6GxpvLaS3m5zlwetxTIgjOOJqM445ir2ZCmosXmxiKSqgDg4GAECUm0W2g+/19KKYXp/xBpvNLKvs9oHEWllegtgB0l4IzIk0HNealmaxbDlBDpBHhCHv Alejandro
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADH3/mLPH+QScQxLdFCp+j1Zp52EfrIujq4qEw7g3luu5SOWCsxuGpaMUN8yxSwCgEAg4BbBLU5EijhHUcV6CSQyWR8WRHMI4w5GlvbLYBNaR+uCCABfLLbTterw2/4kzKWLM9KUZPq1204ulALDGLh/3eLoJGtFKwcvDjfp+zuoH1GGJrP3RdpfIT2E+ovNQGo69xgH+hUueKxOvjRMKbs0FB3CABsDajuMIulRnFxaowa/HofalqOaMaoqWXeqq9Ow+im1qAmZnKAl5oz7nHJA1Njm105VutBylahXr6q4BMJMrPXuJMoyNHVYy509eJ0TYvKOVVSPFtsnZTXowP Adrián
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADGQVK40w8lbCmO/vWRelnC64bpYpQ5EXbyIWiGG8Xx7ir0gXluw6Kp3Vakm2xJp0KzpD8d4guYIBr038Vvyxe/wR69ClSKVMEEWs1MfyIuNv6J5V9kwFAHZlG/IrJ5CaJVChnKxPpeiP5n08jKUZK+zsA+6SlgFM0J9XtmTGNg0g4skqkC9PZ8f4omOBobZ0yZ1dxi1Czb1HgzLDlnyYlWi4Mdz6uB0pD4G6J/oBEmv8Rx1CILDG3me45dT7TLLJ04g3vmVr2qPKGmPMiSeoPFKbYJU5P6uSr6okx2jEXHCF/8LhKxmjybh3liwGU7t3vX6PUyEGDoQqz0pfhbX6mr Rubén
```

3.2. Seguridad

Hacemos una copia de seguridad de la configuración ssh por si acaso con:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
```

Modificamos el archivo de configuración para permitir solo conexiones con claves y no con contraseñas.

Accedemos al archivo con `sudo nano /etc/ssh/sshd_config` y modificamos los atributos relevantes.

```
ubuntu@ip-172-31-16-31: ~  
GNU nano 7.2 /etc/ssh/sshd config *  
PermitRootLogin no  
PasswordAuthentication no  
PubkeyAuthentication yes
```

4. Certificados para HTTPS

4.1. LetsEncrypt

Generamos certificados con letsEncrypt.

app

```
sudo mkdir -p /etc/letsencrypt/live/app.projecteGrupG1.es
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/letsencrypt/live/app.projecteGrupG1.es/privkey.pem \
  -out /etc/letsencrypt/live/app.projecteGrupG1.es/fullchain.pem \
  -subj "/C=ES/ST=Madrid/L=Madrid/O=ProjecteGrupG1/CN=app.projecteGrupG1.es"
```

backup

```
sudo mkdir -p /etc/letsencrypt/live/backup.projecteGrupG1.es
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/letsencrypt/live/backup.projecteGrupG1.es/privkey.pem \
  -out /etc/letsencrypt/live/backup.projecteGrupG1.es/fullchain.pem \
  -subj "/C=ES/ST=Madrid/L=Madrid/O=ProjecteGrupG1/CN=backup.projecteGrupG1.es"
```

test

```
sudo mkdir -p /etc/letsencrypt/live/test.projecteGrupG1.es
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/letsencrypt/live/test.projecteGrupG1.es/privkey.pem \
  -out /etc/letsencrypt/live/test.projecteGrupG1.es/fullchain.pem \
  -subj "/C=ES/ST=Madrid/L=Madrid/O=ProjecteGrupG1/CN=test.projecteGrupG1.es"
```


5. Apache

5.1. Virtual Hosts

app.projecteGrupG1.es

```
ubuntu@ip-172-31-16-31: ~
GNU nano 7.2 /etc/apache2/sites-available/app.projecteGrupG1.es.conf *
<VirtualHost *:80>
  ServerName app.projecteGrupG1.es
  Redirect permanent / https://app.projecteGrupG1.es/
</VirtualHost>

<VirtualHost *:443>
  ServerName app.projecteGrupG1.es
  DocumentRoot /home/app/ftp/www

  SSLEngine on
  SSLCertificateFile /etc/letsencrypt/live/app.projecteGrupG1.es/fullchain.pem
  SSLCertificateKeyFile /etc/letsencrypt/live/app.projecteGrupG1.es/privkey.pem

  ErrorLog /home/app/logs/error.log
  CustomLog /home/app/logs/access.log combined

  ProxyPreserveHost On
  ProxyPass / http://127.0.0.1:8000/
  ProxyPassReverse / http://127.0.0.1:8000/

  RequestHeader set X-Forwarded-Proto "https"
  RequestHeader set X-Forwarded-Port "443"
  RequestHeader set X-Real-IP "%{REMOTE_ADDR}s"
  RequestHeader set X-Forwarded-For "%{REMOTE_ADDR}s"
</VirtualHost>
```

backup.projecteGrupG1.es

```
ubuntu@ip-172-31-16-31: ~
GNU nano 7.2 /etc/apache2/sites-available/backup.projecteGrupG1.es.conf *
<VirtualHost *:80>
  ServerName backup.projecteGrupG1.es
  Redirect permanent / https://backup.projecteGrupG1.es/
</VirtualHost>

<VirtualHost *:443>
  ServerName backup.projecteGrupG1.es
  DocumentRoot /home/backup/ftp/fitxers

  SSLEngine on
  SSLCertificateFile /etc/letsencrypt/live/backup.projecteGrupG1.es/fullchain.pem
  SSLCertificateKeyFile /etc/letsencrypt/live/backup.projecteGrupG1.es/privkey.pem

  <Directory /home/backup/ftp/fitxers>
    Options Indexes FollowSymLinks
    AuthType Basic
    AuthName "Acceso Restringido - Backups"
    AuthUserFile /etc/apache2/.htpasswd_backup
    Require valid-user
  </Directory>

  ErrorLog /var/log/apache2/backup_error.log
  CustomLog /var/log/apache2/backup_access.log combined
</VirtualHost>
```

Además del VirtualHost para backup, creamos credenciales para acceder por contraseña con sudo htpasswd -c /etc/apache2/.htpasswd_backup app y le ponemos 12345
Para loguearnos: User → app; Password → 12345

test.projecteGrupG1.es

```
ubuntu@ip-172-31-16-31:~  
GNU nano 7.2 /etc/apache2/sites-available/test.projecteGrupG1.es.conf *  
<VirtualHost *:80>  
  ServerName test.projecteGrupG1.es  
  Redirect permanent / https://test.projecteGrupG1.es/  
</VirtualHost>  
  
<VirtualHost *:443>  
  ServerName test.projecteGrupG1.es  
  DocumentRoot /home/app/ftp/test  
  
  SSLEngine on  
  SSLCertificateFile /etc/letsencrypt/live/test.projecteGrupG1.es/fullchain.pem  
  SSLCertificateKeyFile /etc/letsencrypt/live/test.projecteGrupG1.es/privkey.pem  
  
  ProxyPreserveHost On  
  # El entorno test correrá en el puerto 8001  
  ProxyPass / http://127.0.0.1:8001/  
  ProxyPassReverse / http://127.0.0.1:8001/  
  
  RequestHeader set X-Forwarded-Proto "https"  
  RequestHeader set X-Forwarded-Port "443"  
  
  ErrorLog /var/log/apache2/test_error.log  
  CustomLog /var/log/apache2/test_access.log combined  
</VirtualHost>
```

Hecho esto, configuramos el archivo hosts para que las ip apunten a nuestros nombres.

```
batoi@AlejandroRico: ~/Documentos/Segundo/PI/despliegue  
GNU nano 7.2 /etc/hosts *  
127.0.0.1 localhost  
127.0.1.1 batoi-SSD-2024  
  
# Pagina PI  
3.213.82.57 app.projecteGrupG1.es  
3.213.82.57 backup.projecteGrupG1.es  
3.213.82.57 test.projecteGrupG1.es
```

6. Servidor FTP

6.1. vsftpd.conf

Instalamos vsftpd y configuramos el archivo vsftpd.conf para que cumpla con los requisitos de puertos, la ip de nuestra máquina y que podamos permitir el uso a ciertos usuarios.

```
ubuntu@ip-172-31-16-31: ~
GNU nano 7.2 /etc/vsftpd.conf
# =====
# CONFIGURACIÓN VSFTPD - ProjecteGrupG1
# =====

# === Acceso Básico ===
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022

# === Encarcelar usuarios (Chroot) ===
chroot_local_user=YES
allow_writeable_chroot=YES

# === Modo Pasivo (Requisito 3.3: puertos 30000-30050) ===
pasv_enable=YES
pasv_min_port=30000
pasv_max_port=30050
pasv_address=3.213.82.57

# === Seguridad ===
force_local_logins_ssl=NO
force_local_data_ssl=NO
ssl_enable=NO
pam_service_name=vsftpd
secure_chroot_dir=/var/run/vsftpd/empty

# === Logs ===
xferlog_enable=YES
xferlog_file=/var/log/vsftpd.log
xferlog_std_format=YES
log_ftp_protocol=YES

# === Restringir usuarios a lista específica ===
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=YES

listen=YES
listen_ipv6=NO
```

6.2. vsftpd.userlist

Permitimos que el usuario ubuntu pueda utilizar el servicio ftp

```
ubuntu@ip-172-31-16-31:~$ sudo cat /etc/vsftpd.userlist
ubuntu
```

Creamos contraseñas para vsftpd para los usuarios app y backup

```
sudo passwd app → app1234
```

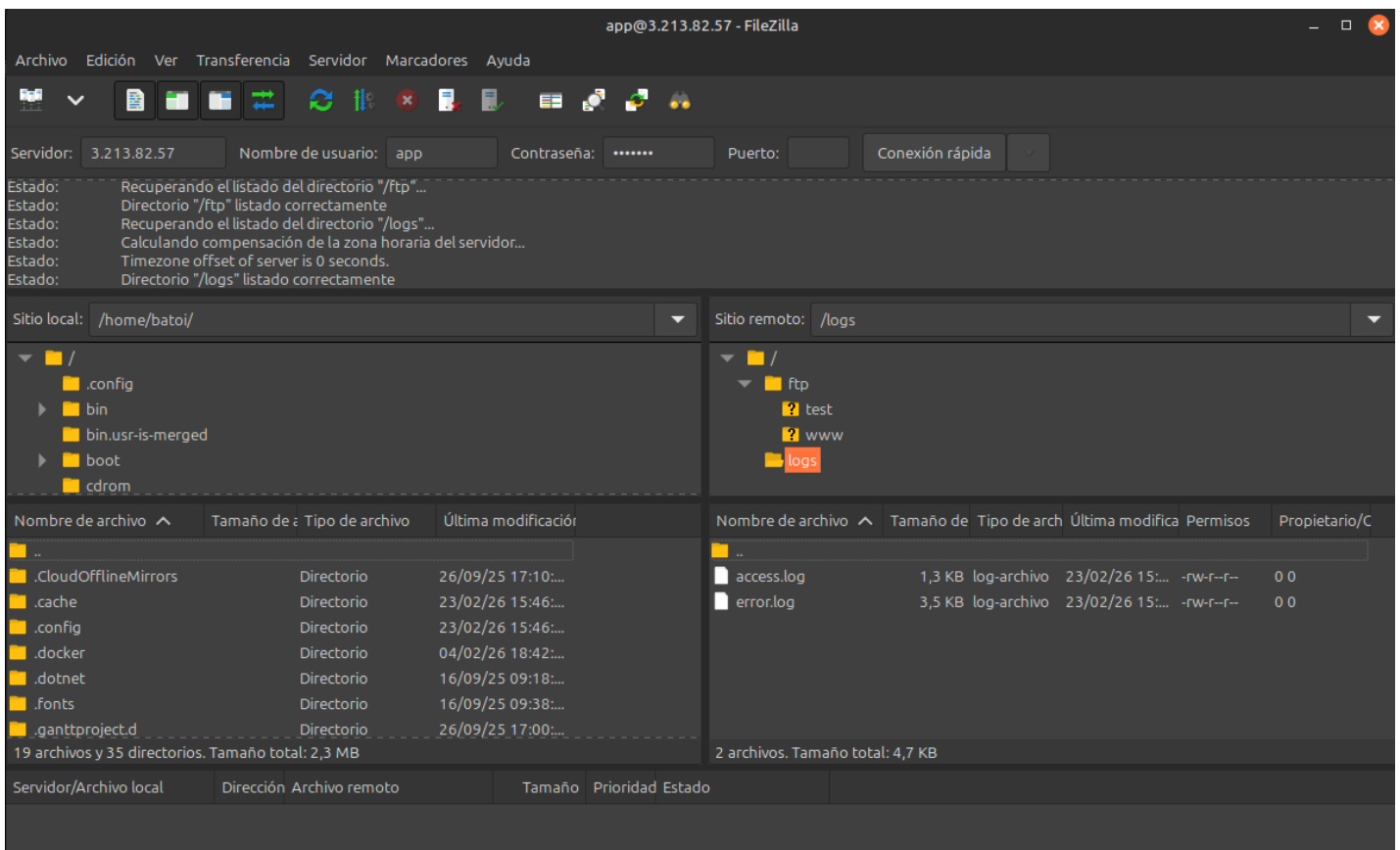
```
sudo passwd backup → backup1234
```

6.3. Comprobación vsftpd

Vemos el estado del servicio vsftpd para comprobar que esta active (running) y nos conectamos con app desde filezilla para comprobar que funcione correctamente.

```
ubuntu@ip-172-31-16-31:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled; preset: e
   Active: active (running) since Mon 2026-02-23 14:39:18 UTC; 9s ago
     Main PID: 3905 (vsftpd)
        Tasks: 1 (limit: 4665)
       Memory: 712.0K (peak: 1.5M)
          CPU: 8ms
       CGroup: /system.slice/vsftpd.service
              └─3905 /usr/sbin/vsftpd /etc/vsftpd.conf

Feb 23 14:39:18 ip-172-31-16-31 systemd[1]: Starting vsftpd.service - vsftpd FT
Feb 23 14:39:18 ip-172-31-16-31 systemd[1]: Started vsftpd.service - vsftpd FTP
```



The screenshot shows the FileZilla interface with the following details:

- Server: 3.213.82.57
- Username: app
- Connection: Successful
- Local site: /home/batoi/
- Remote site: /logs
- Remote directory listing:

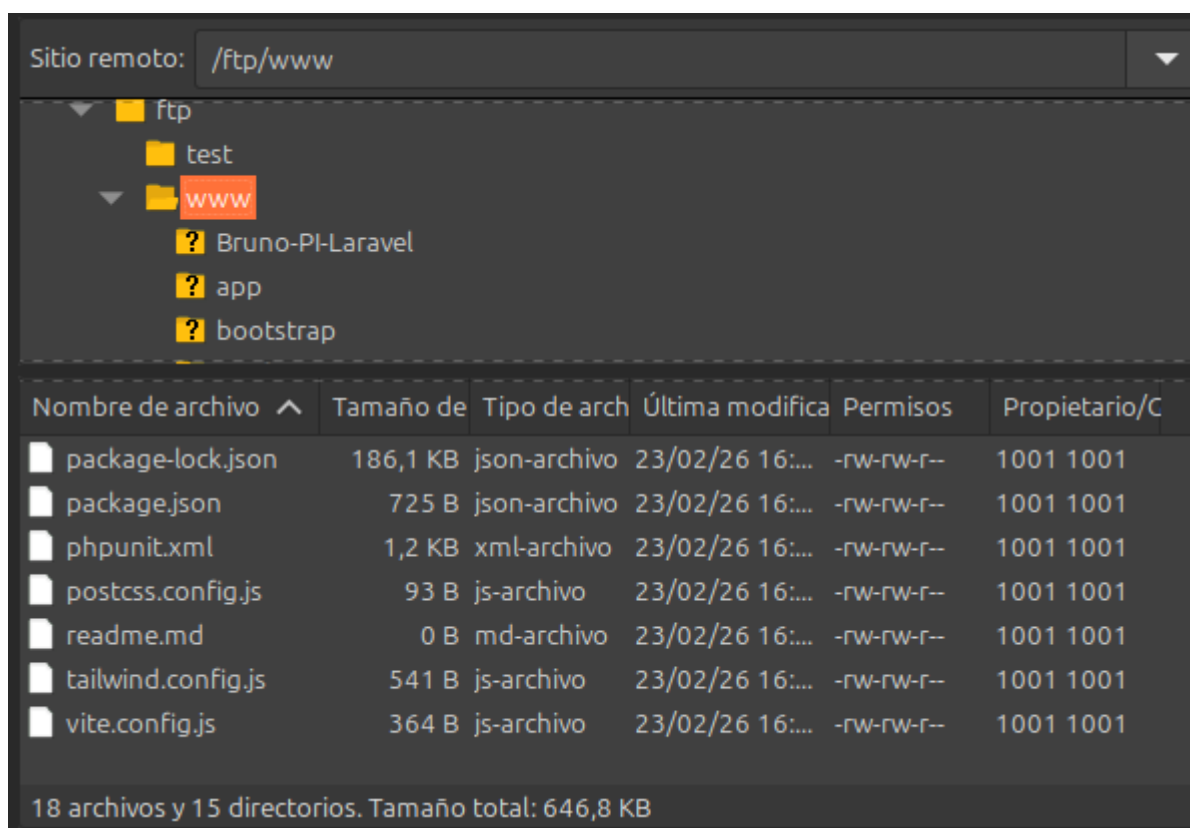
Nombre de archivo	Tamaño de	Tipo de arch	Última modifica	Permisos	Propietario/C
..					
access.log	1,3 KB	log-archivo	23/02/26 15:...	-rw-r--	0 0
error.log	3,5 KB	log-archivo	23/02/26 15:...	-rw-r--	0 0

7. Despliegue de la aplicación

7.1. APP

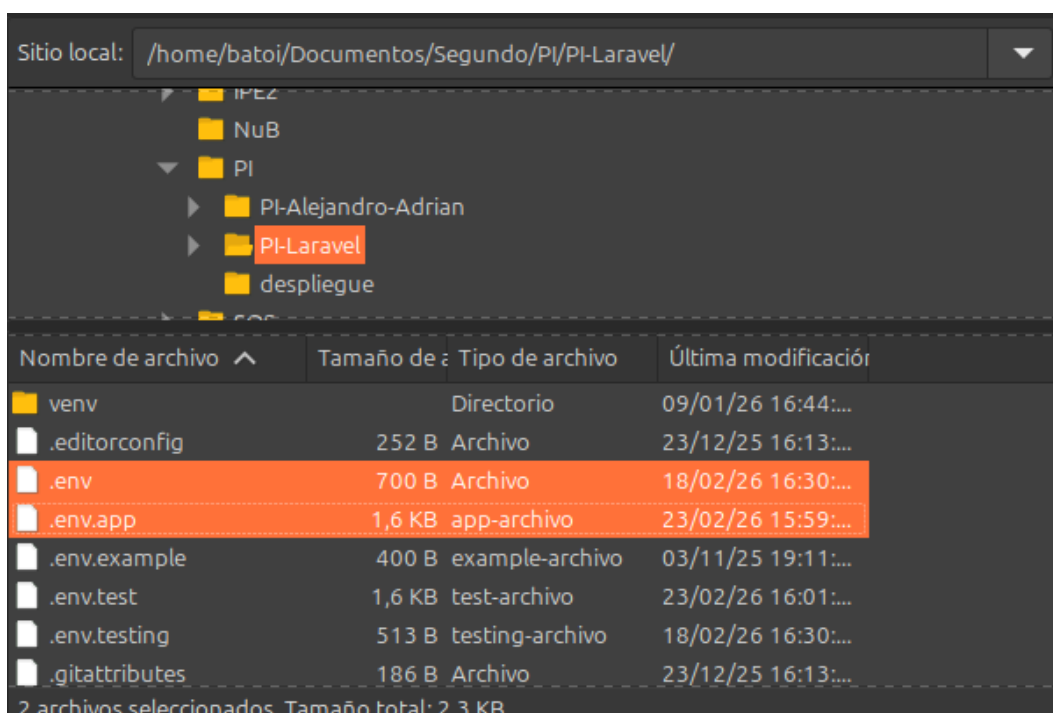
Para subir los archivos a la máquina virtual, lo haremos desde git en vez de el ftp que acabamos de configurar para que sea mas sencillo poder actualizar el proyecto y luego, por ftp enviaremos los archivos .env necesarios con las credenciales, ya que estos están dentro den .gitignore y no se copiarán.

Hacemos un git clone de nuestro repositorio en app/ftp/www de lo que sería nuestro proyecto completo y vemos los siguientes archivos:



Ahora movemos a www los archivos .env y .env.app que sería los necesarios para el despliegue en producción de nuestra aplicación. Esto lo haremos desde filezilla.

Seleccionamos los dos archivos que nos interesan y los arrastramos al directorio de nuestra máquina virtual.



```

ubuntu@ip-172-31-16-31:/$ sudo ls -la /home/app/ftp/www/
total 776
drwxr-xr-x 19 app app 4096 Feb 23 15:36 .
drwxr-xr-x  4 app app 4096 Feb 19 18:14 ..
-rw-rw-r--  1 app app  252 Feb 23 15:27 .editorconfig
-rw-rw-r--  1 app app  730 Feb 23 15:36 .env
-rw-r--r--  1 app app 1596 Feb 23 15:36 .env.app

```

Ahora que todo parece estar listo, hacemos `docker compose up -d` para construir y levantar los contenedores por primera vez. Sorprendentemente no ha fallado por ahora y tenemos lo siguiente:

```

ubuntu@ip-172-31-16-31: /home/app/ftp/www
=> [stage-0 4/6] RUN usermod -u 1000 www-data && groupmod -g 1000 www-da 0.3s
=> [stage-0 5/6] RUN mkdir -p /var/www/html /home/www-data/.npm && c 0.2s
=> [stage-0 6/6] WORKDIR /var/www/html 0.1s
=> exporting to image 10.1s
[+] up 91/91ting layers 8.4s
✓ Image redis:7-alpine Pulled 5.7ss
✓ Image n8nio/n8n:latest Pulled 64.0s
✓ Image pi-laravel-app Built 204.2s
✓ Image phpmyadmin:latest Pulled 38.8s
✓ Image node:20 Pulled 49.1s
✓ Image nginx:1.27-alpine Pulled 5.5ss
✓ Image mysql:8.4 Pulled 42.8s
✓ Network www_default Created 0.0s
✓ Volume www_n8n_data Created 0.0s
✓ Volume www_db_data Created 0.0s
✓ Container pi-laravel-redis Created 0.3s
✓ Container pi-laravel-n8n Created 0.3s
✓ Container pi-laravel-db Created 0.3s
✓ Container pi-laravel-app Created 0.1s
✓ Container pi-laravel-phpmyadmin Created 0.1s
✓ Container pi-laravel-nginx Created 0.1s
✓ Container pi-laravel-vite Created 0.1s

```

Ahora tenemos que instalar las dependencias necesarias de nuestro proyecto para que todo funcione correctamente.

Añadimos en el `.gitignore` en local el archivo para evitar pisarlo en un futuro igual que con los `.env`.

Debido a que en local tenemos contenedores en docker con nginx y en remoto servimos la aplicación a través de apache y que tenemos en el proyecto mezcla de laravel para backend (+ vistas blade para frontend iniciales que ya no se utilizan) y vue para el frontend, debemos configurar alguna ruta de `routes/web.php` y `routes/api.php` junto con `nginx/default.conf` para que las rutas no entren en bucles cíclicos y funcionen correctamente.

```
ubuntu@ip-172-31-16-31:/home/app/ftp/www$ cat nginx/default.conf
server {
    listen 80;
    server_name _;

    root /var/www/html/public;
    index index.php index.html;

    # API → Laravel (PRIMERO)
    location ^~ /api {
        try_files $uri $uri/ /index.php?$query_string;
    }

    # PHP-FPM
    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass app:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $document_root;
    }

    # Assets (muy importante antes que SPA fallback)
    location /assets/ {
        try_files $uri =404;
        expires 7d;
        access_log off;
    }

    # SPA fallback SOLO para lo que no sea API ni archivo real
    location / {
        try_files $uri $uri/ /index.html;
    }

    location ~ /\. {
        deny all;
    }
}
```

Arquitectura de la aplicación en producción:

Aquí se muestra un diagrama general del despliegue en el que podemos ver el flujo de la información desde que el usuario accede a la aplicación hasta que se obtiene una respuesta desde la base de datos.

El cliente se conecta a través del protocolo HTTPS al servidor Apache, que actúa como reverse proxy y se encarga de gestionar el certificado SSL.

El servidor Apache redirige las peticiones al servidor Nginx, que se encuentra ejecutándose dentro de un contenedor Docker.

Nginx analiza el tipo de petición recibida:

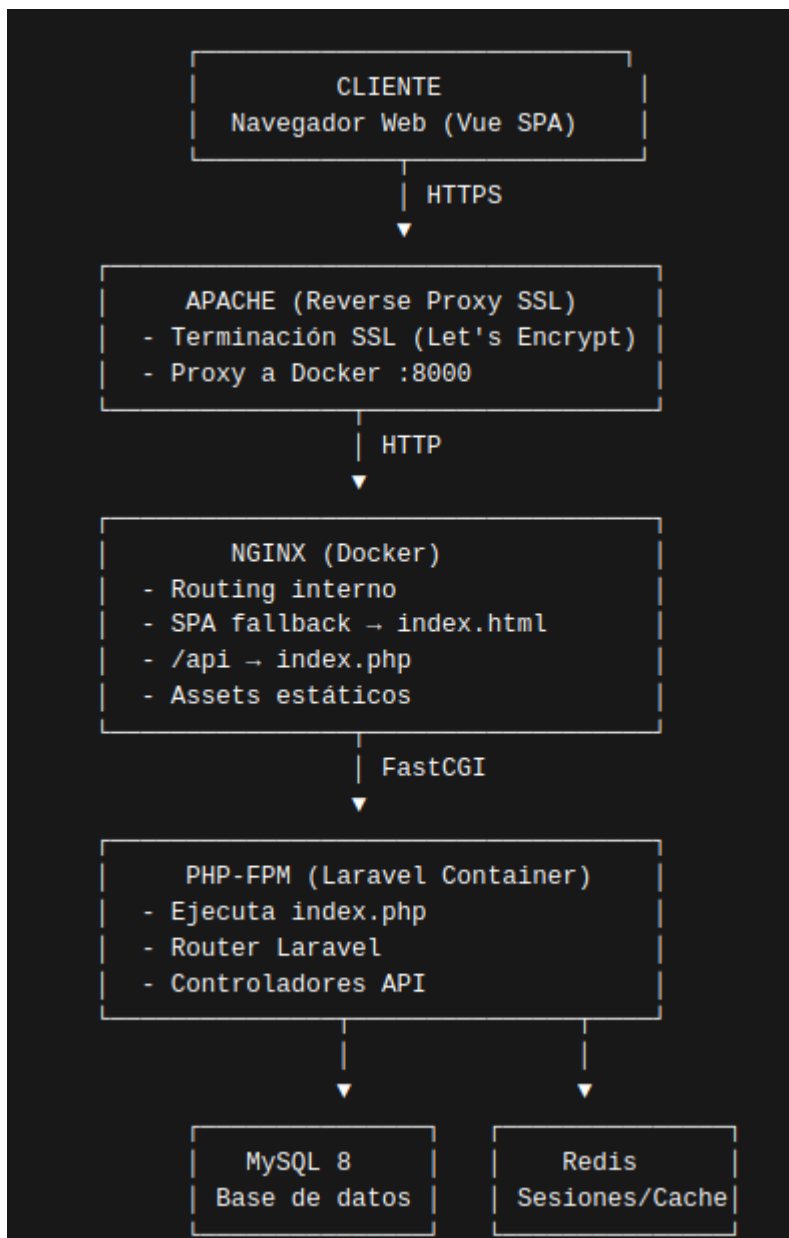
Si es una ruta de la API (/api/*), la redirige a PHP-FPM, donde se ejecuta Laravel.

Si es una ruta de la aplicación web (como /login o /register), devuelve el archivo index.html para que Vue gestione la navegación.

Si es un recurso estático (como archivos JavaScript o CSS), lo sirve directamente.

Cuando la petición se dirige a Laravel, este procesa la lógica de negocio y se comunica con MySQL para obtener o almacenar datos, y con Redis para gestionar sesiones o caché.

Finalmente, la respuesta vuelve siguiendo el mismo recorrido hasta llegar al cliente.



Arquitectura Lógica SPA + API:

Aquí se muestra un diagrama lógico del funcionamiento interno de la aplicación, donde se aprecia la separación entre frontend y backend.

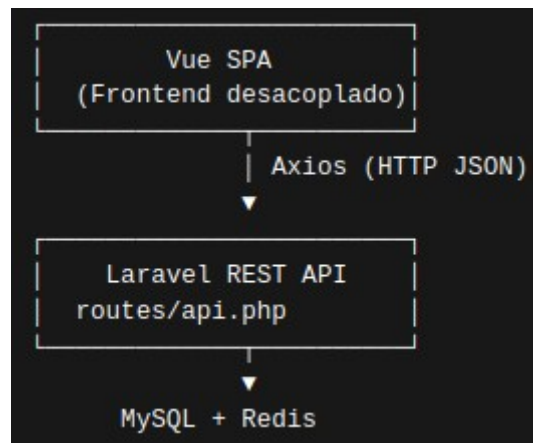
El cliente carga una aplicación desarrollada en Vue, que funciona como una Single Page Application (SPA). Cuando el usuario interactúa con la aplicación, Vue realiza peticiones HTTP mediante Axios a la API REST desarrollada en Laravel.

Laravel recibe estas peticiones a través de sus rutas definidas en `routes/api.php`, ejecuta la lógica correspondiente en los controladores y consulta la base de datos MySQL cuando es necesario.

Una vez procesada la información, Laravel devuelve una respuesta en formato JSON, que Vue interpreta y muestra dinámicamente en la interfaz sin necesidad de recargar la página completa.

Este modelo permite separar claramente la parte visual (frontend) de la lógica de negocio y acceso a datos (backend), facilitando el mantenimiento y la escalabilidad del sistema.

A continuación se muestran algunas de las vistas de la aplicación web para demostrar que todo funciona correctamente y no es solo la vista principal.



Mewgenics

Ya disponible

Últimas novedades





Baldur's Gate III

40.00€ **36.00€**

Sumérgete en un RPG épico ambientado en el universo de Dungeons & Dragons. Baldur's Gate III ofrece una narrativa rica, decisiones que influyen directamente en la historia y un sistema de combate por turnos profundo y estratégico.

Plataforma:

- PlayStation 5
- Steam
- Xbox Series X

Añadir al carrito 🛒 ⚡

Comentarios

Adrián

★ 4

Tenetur quia dolorempue et molestiae non et totam fuga perspiciatis nihil enim.



Buscar...



Finalizar Compra

Resumen del Pedido

Celeste 10.00 €
1 x 10.00 €
Plataforma: Steam

Total: 10.00 €

Método de Pago

Tarjeta guardada Nueva tarjeta

Selecciona tu tarjeta

**** * 1113 - ADMIN USER

Volver al carrito

Pagar 10.00 €

7.2. Test

Para desplegar los tests, podemos realizar una copia del despliegue realizado para producción en app y modificar el archivo .env para que coincida con el .env.test que se ha generado anteriormente. De esta manera podemos aprovechar todos los ajustes realizados para que funcione bien la web y tener así un entorno real para realizar pruebas con la aplicación desplegada en un entorno real. Esto es interesante debido a que tenemos añadidas opciones en nuestra aplicación para poder modificar de forma manual productos.

Debido a que no tenemos demasiado espacio en la máquina virtual de AWS, no vamos a duplicar los archivos, pero en nuestro caso, al querer una instancia igual a la de producción pero con una base de datos distinta para poder hacer las pruebas dejando solo este .env y el resto igual debería ser suficiente:

```
# =====
# APP - Configuración de Test
# =====
APP_NAME=ProjecteGrupG1
APP_ENV=testing
APP_KEY=base64:Y776kFynd6PFMSIWQwTzpB7nc8x9JteYYcy2E2rIc5w=
APP_DEBUG=false
APP_URL=https://test.projecteGrupG1.es

LOG_CHANNEL=stack
LOG_LEVEL=error

# =====
# BASE DE DATOS - Test
# =====
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=pi_laravel_test
DB_USERNAME=pi
DB_PASSWORD=pi

# =====
# CACHE / COLAS / SESIONES - Test
# =====
BROADCAST_DRIVER=log
CACHE_DRIVER=redis
FILESYSTEM_DISK=local
QUEUE_CONNECTION=redis
SESSION_DRIVER=redis
SESSION_LIFETIME=120

REDIS_HOST=redis
REDIS_PASSWORD=null
REDIS_PORT=6379
```

```
# =====
# GOOGLE AUTH - Test
# =====
GOOGLE_CLIENT_ID=782388634943-
00eldr0npg8i37ndph980g76m4844oku.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
GOOGLE_REDIRECT_URI=https://test.projecteGrupG1.es/api/auth/google/callback

# =====
# FRONTEND (Vue/Vite) - Test
# =====
FRONTEND_URL=https://test.projecteGrupG1.es

# =====
# PROXY / HEADERS - Importante detrás de Apache
# =====
TRUSTED_PROXIES=*
```

Hemos cambiado estas lineas:

APP_ENV=testing ← para tener el entorno de test

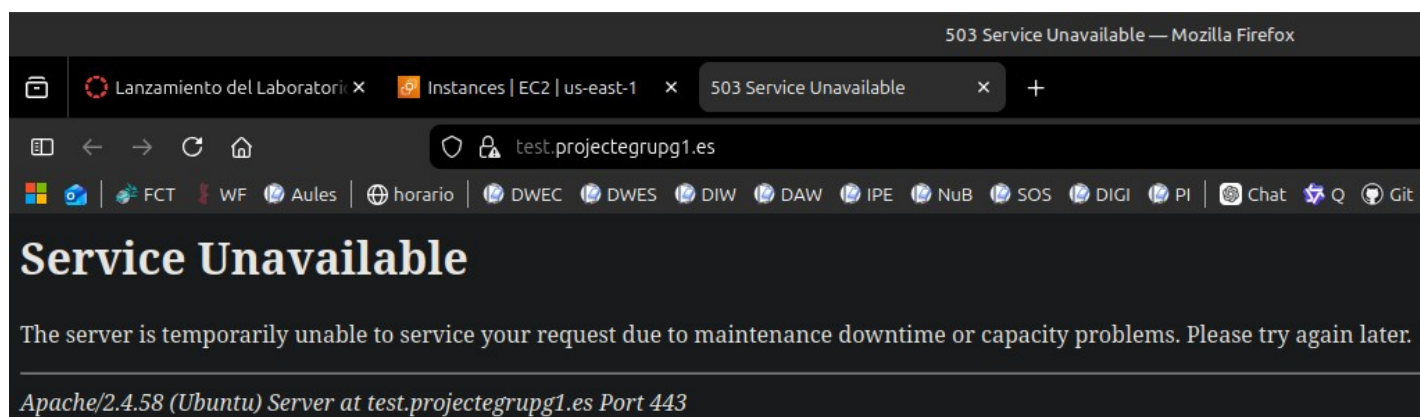
APP_URL=<https://test.projecteGrupG1.es> ← la dirección de acceso a la web

DB_DATABASE=pi_laravel_test ← para tener una nueva base de datos (habría que modificar los controladores para redirigirlos a esa nueva base de datos de test)

GOOGLE_REDIRECT_URI=<https://test.projecteGrupG1.es/api/auth/google/callback> ← para poder seguir logueando desde google en el entorno de test

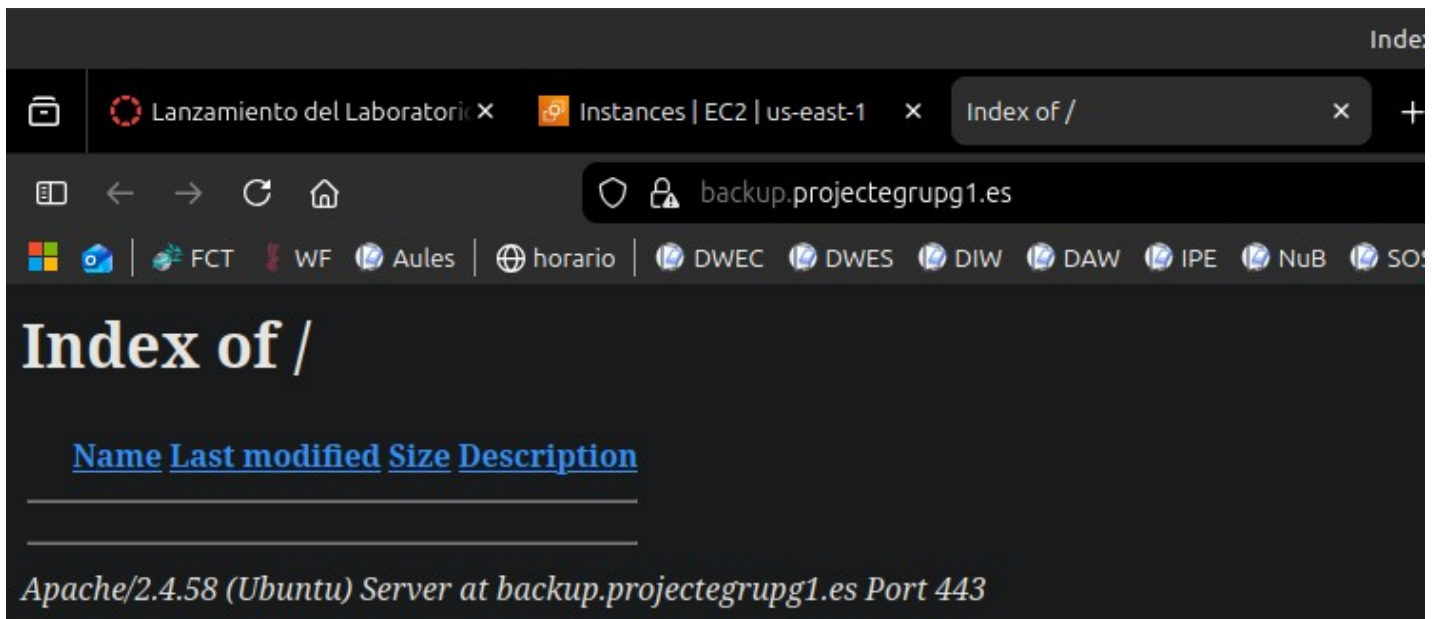
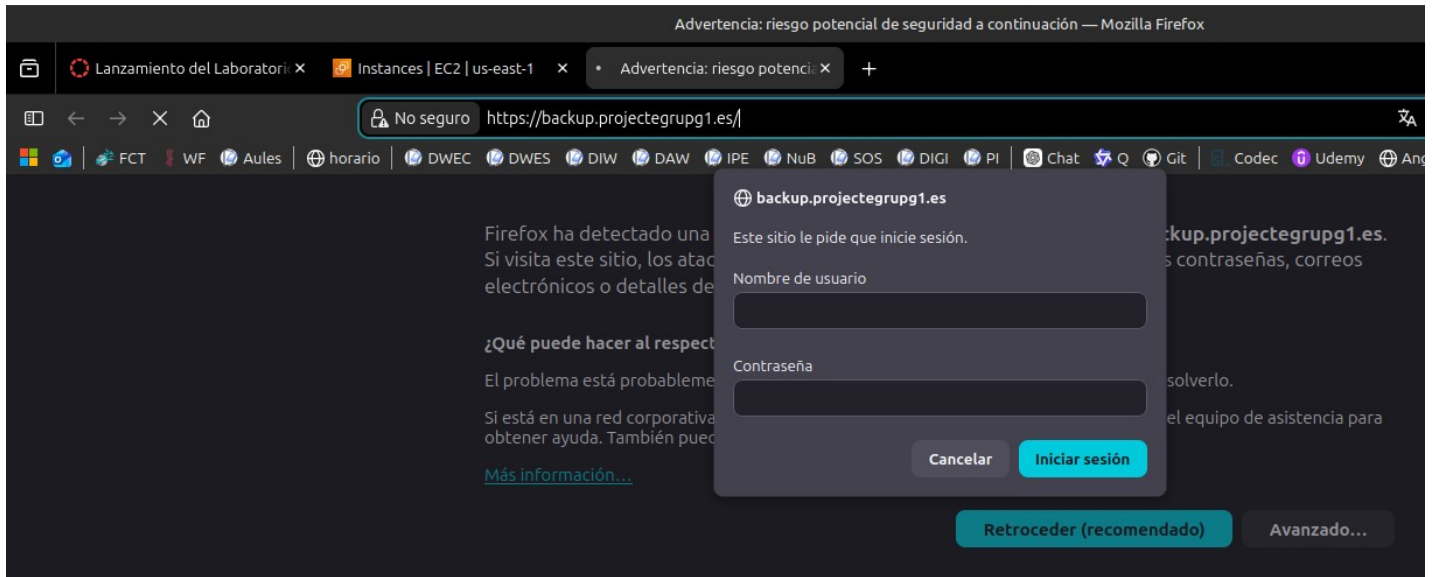
FRONTEND_URL=<https://test.projecteGrupG1.es> ← para que vue pueda servir el frontend en test.

Ahora mismo el entorno está vacío y sin levantar docker ni nada, pero esta vista nos dice que debería estar todo configurado para poder funcionar correctamente.



7.3. Backup

El entorno de backup está ya preparado para poder almacenar ahí los datos que consideremos oportunos. Si entramos desde la web a <https://backup.projectegrupg1.es/> vemos que nos pide las credenciales como se ha configurado anteriormente para poder acceder al sitio en el que tendremos almacenadas las copias de seguridad.



8. Arquitectura escalable AWS

8.1. Subred VPC

Vamos ahora a realizar una vpc para separar aún mas las responsabilidades de cada servicio de la aplicación y a subir la base de datos a AWS para que no la gestione docker. Para ello empezaremos creando una VPC.

The screenshot shows the AWS VPC console for vpc-03004d3b27a2849cf. The details are as follows:

VPC ID vpc-03004d3b27a2849cf	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-0563ac16e0c485134	Main route table rtb-0b04e80e94316d3af
Main network ACL acl-043bb8a898e74932c	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -
IPv6 CIDR (Network border group) -	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups Failed to load rule groups	Owner ID 292873490331
Encryption control ID -	Encryption control mode -		

Creamos diversas subredes publicas y privadas para la aplicación y privadas para la base de datos:

Subnet Name	Subnet ID	State	VPC	Block Public Access	IPv4 CIDR
app-pub-1	subnet-062998117569f9854	Available	vpc-03004d3b27a2849cf	Off	10.0.1.0/24
app-pub-2	subnet-0dfe62369ce09e303	Available	vpc-03004d3b27a2849cf	Off	10.0.2.0/24
app-priv-1	subnet-090ca4ddc2022a283	Available	vpc-03004d3b27a2849cf	Off	10.0.11.0/24
app-priv-2	subnet-09b204be2fd867a12	Available	vpc-03004d3b27a2849cf	Off	10.0.12.0/24
bd-priv-1	subnet-071c1f38a83d182f4	Available	vpc-03004d3b27a2849cf	Off	10.0.21.0/24
bd-priv-2	subnet-092f08e0106acb854	Available	vpc-03004d3b27a2849cf	Off	10.0.22.0/24

Creamos una Internet Gateway y la enlazamos a la VPC.

The screenshot shows the 'Attach to VPC' page for Internet Gateway igw-0786e061c57dd7598. The 'Available VPCs' section shows a search bar with 'vpc-03004d3b27a2849cf' entered.

Y ahora hacemos una NAT Gateway en una de las subredes públicas.

The screenshot shows the AWS NAT Gateway console for nat-1fe9ef761a56c463e. The details are as follows:

NAT gateway ID nat-1fe9ef761a56c463e	Availability mode Regional	State Available	State message -
NAT gateway ARN arn:aws:ec2:us-east-1:292873490331:natgateway/nat-1fe9ef761a56c463e	Connectivity type Public	Created Wednesday, February 25, 2026 at 16:11:44 GMT +1	Deleted -
VPC vpc-03004d3b27a2849cf / vpc-bitkeys	Method of EIP allocation Manual		

Associated IP addresses (1)

IP address	Status	Availability Zone	Allocation ID	Association ID
44.206.43.168	Succeeded	use1-az1 (us-east-1a)	eipalloc-062fcc1f13c60177f	eipassoc-0c2f6db6b3e801f9e

Creamos una tabla de rutas y asociamos a ella las dos subredes públicas de la aplicación

rtb-0d6c5446670040fb9 / bitkeys-public Actions

Details Info

Route table ID rtb-0d6c5446670040fb9	Main No	Explicit subnet associations 2 subnets	Edge associations -
VPC vpc-03004d3b27a2849cf vpc-bitkeys	Owner ID 292873490331		

Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2) Both Edit routes

Filter routes

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0786e061c57dd7598	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

Creamos otra tabla de rutas para gestionar las subredes privadas.

rtb-0db981a3f093a1a15 / bitkeys-app-private Actions

Details Info

Route table ID rtb-0db981a3f093a1a15	Main No	Explicit subnet associations 2 subnets	Edge associations -
VPC vpc-03004d3b27a2849cf vpc-bitkeys	Owner ID 292873490331		

Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2) Both Edit routes

Filter routes

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	nat-1fe9ef761a56c463e	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

Otra tabla de rutas para la base de datos.

rtb-0759f9fa83bc8b679 / bitkeys-bd-private Actions

Details Info

Route table ID rtb-0759f9fa83bc8b679	Main No	Explicit subnet associations 2 subnets	Edge associations -
VPC vpc-03004d3b27a2849cf vpc-bitkeys	Owner ID 292873490331		

Subnet associations | Edge associations | Route propagation | Tags

Explicit subnet associations (2) Edit subnet associations

Find subnet association

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
bd-priv-2	subnet-092f08e0106acb854	10.0.22.0/24	-
bd-priv-1	subnet-071c1f38a83d182f4	10.0.21.0/24	-

8.2. Load Balancer

Creamos primero un target group

TG-bitkey-app Actions

Details

arn:aws:elasticloadbalancing:us-east-1:292873490331:targetgroup/TG-bitkey-app/565fdb86abe6408c

Target type Instance	Protocol : Port HTTP : 8000	Protocol version HTTP1	VPC vpc-03004d3b27a2849cf
IP address type IPv4	Load balancer None associated		

0 Total targets	0 Healthy	0 Unhealthy	0 Unused	0 Initial	0 Draining
	0 Anomalous				

Creamos un nuevo grupo de seguridad para el load balancer

sg-0dbcd03c623becaaa - load-balancer Actions

Details

Security group name load-balancer	Security group ID sg-0dbcd03c623becaaa	Description sg de load balancer	VPC ID vpc-03004d3b27a2849cf
Owner 292873490331	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing](#) | [VPC associations](#) | [Related resources - new](#) | [Tags](#)

Inbound rules (2) Manage tags Edit inbound rules

Search

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0df831f630c33e984	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-0e69fa4c6ada8dbb7	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

Creamos el load balancer

bitkeys-LB Actions

Details

Load balancer type Application	Status Provisioning	VPC vpc-03004d3b27a2849cf	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0dfe62369ce09e303 us-east-1b (use1-az2) subnet-062998117569f9854 us-east-1a (use1-az1)	Date created February 25, 2026, 16:35 (UTC+01:00)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:292873490331:loadbalancer/app/bitkeys-LB/57988dca37a274b0

DNS name info
bitkeys-LB-160498095.us-east-1.elb.amazonaws.com (A Record)

[Listeners and rules](#) | [Network mapping](#) | [Resource map](#) | [Security](#) | [Monitoring](#) | [Integrations](#) | [Attributes](#) | [Capacity](#) | [Tags](#)

Listeners and rules (1) Info Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
HTTP:80	Forward to target group TG-bitkey-app : 1 (100%) Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable

8.3. Capa de datos

Creamos un subnet group con las subredes de la base de datos

bitkeys-db

Subnet group details

VPC ID
vpc-03004d3b27a2849cf

ARN
arn:aws:rds:us-east-1:292873490331:subgrp:bitkeys-db

Supported network types
IPv4

Description
database de bitkeys

Subnets (2)

Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1b	bd-priv-2	subnet-092f08e0106acb854	10.0.22.0/24
us-east-1a	bd-priv-1	subnet-071c1f38a85d182f4	10.0.21.0/24

Generamos un grupo de seguridad para el puerto 3306 que pueda conectarse solo desde nuestro load balancer

The screenshot shows the AWS IAM console for a security group named 'db-sg'. The details section includes: Security group name (db-sg), Security group ID (sg-00c738ce1f8125e01), Description (database security group), VPC ID (vpc-03004d3b27a2849cf), Owner (292873490331), Inbound rules count (1 Permission entry), and Outbound rules count (1 Permission entry). The inbound rules section shows a single rule for MySQL/Aurora on TCP port 3306, with source 'sg-0dbcd03c623becaaa...'.

Creamos la base de datos

The screenshot shows the AWS IAM console for a database instance named 'database-bitkeys-instance-1'. The related section shows a table with columns: DB identifier, Status, Role, Engine, Region, Size, and Recommendations. The instance 'database-bitkeys-instance-1' is listed as 'Available' with role 'Writer ins...', engine 'Aurora My...', region 'us-east-1a', and size 'db.t3.med...'.

8.4. Capa de aplicación

Creamos una Launch template con as órdenes necesarias para generar nuevas instancias en caso de tener una carga elevada en nuestra EC2.

The screenshot shows the AWS IAM console for a launch template named 'scaling-template (lt-016d0baf1c7e28c5e)'. The launch template details section includes: Launch template ID (lt-016d0baf1c7e28c5e), Launch template name (scaling-template), Default version (1), and Owner (arn:aws:sts::292873490331:assumed-role/vocla bs/user4424895=ALEJANDRO_RICO_SANCHEZ). The launch template version details section shows version 1 (Default) with description '-', date created '2026-02-25T16:31:41.000Z', and created by 'arn:aws:sts::292873490331:assumed-role/vocla bs/user4424895=ALEJANDRO_RICO_SANCHEZ'. The instance details section includes: AMI ID (ami-0b6c6ebd2801a5cb), Instance type (t2.small), Availability Zone (-), Availability Zone Id (-), Key pair name (PI-alejandro), Security groups (-), and Security group IDs (sg-00c738ce1f8125e01).

Creamos el Auto Scaling Group

The screenshot shows the AWS Management Console for an Auto Scaling Group named 'Bitkeys-ASG'. The 'Capacity overview' section displays a desired capacity of 2 instances, scaling limits from 2 to 3, and a status of 'Updating capacity'. The 'Launch template' section shows it is using the 'scaling-template' with an AMI ID of 'ami-0b6c6ebed2801a5cb', instance type 't2.small', and security group 'sg-00c738ce1f8125e01'. The owner is 'arn:aws:sts::292873490331:assumed-role/voclabs/user4424895=ALEJANDRO_RICO_SANCHEZ' and it was created on Feb 25, 2026 at 17:31:41 GMT+0100.

8.5. Despliegue

Vamos a aprovechar el directorio que tenemos creado en `home/app/ftp/test` que estaba vacío para desplegar la versión de la aplicación que utiliza la base de datos de AWS y no la de docker, para ello tenemos que modificar el archivo `.env` y `docker-compose.yml`.

Los archivos han quedado de la siguiente manera:

```
/home/app/ftp/test$ cat .env
```

```
# =====  
# APP - Configuración de Test  
# =====  
APP_NAME=ProjecteGrupG1  
APP_ENV=test  
APP_KEY=base64:Y776kFynd6PFMSIWQwTzpB7nc8x9JteYYcy2E2rIc5w=  
APP_DEBUG=true  
APP_URL=https://test.projecteGrupG1.es  
  
LOG_CHANNEL=stack  
LOG_LEVEL=debug  
  
# =====  
# BASE DE DATOS - Test (RDS)  
# =====  
DB_CONNECTION=mysql  
DB_HOST=database-bitkeys.cluster-cuz1xqqg4dmp.us-east-1.rds.amazonaws.com  
DB_PORT=3306  
DB_DATABASE=pi_laravel  
DB_USERNAME=pi  
DB_PASSWORD=12345678
```

```
# =====
# CACHE / COLAS / SESIONES - Test
# =====
BROADCAST_DRIVER=log
CACHE_DRIVER=redis
FILESYSTEM_DISK=local
QUEUE_CONNECTION=redis
SESSION_DRIVER=redis
SESSION_LIFETIME=120

REDIS_HOST=redis
REDIS_PASSWORD=null
REDIS_PORT=6379

# =====
# GOOGLE AUTH - Test
# =====
GOOGLE_CLIENT_ID=782388634943-
00eldr0npg8i37ndph980g76m4844oku.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
GOOGLE_REDIRECT_URI=https://test.projecteGrupG1.es/api/auth/google/callback

# =====
# FRONTEND (Vue/Vite) - Test
# =====
FRONTEND_URL=https://test.projecteGrupG1.es

# =====
# PROXY / HEADERS - Importante detrás de Apache
# =====
TRUSTED_PROXIES=*
```

Vemos que en el .env se han modificado las credenciales de la base de datos para que coincidan con el endpoint de la RDS de AWS. De esta forma se utiliza la base de datos gestionada por AWS y no por docker.

```
/home/app/ftp/test$ cat docker-compose.yml
```

```
services:
```

```
  app:
```

```
    build:
```

```
      context: .
```

```
      dockerfile: Dockerfile
```

```
    args:
```

```
      WWWUSER: "${WWWUSER:-1000}"
```

```
      WWWGROUP: "${WWWGROUP:-1000}"
```

```
  image: pi-laravel-app-test-test
```

```
  container_name: pi-laravel-app-test-test
```

```
  working_dir: /var/www/html
```

```
  volumes:
```

```
    - ./var/www/html:cached
```

```
    - ./vendor:/var/www/html/vendor
```

```
    - ./node_modules:/var/www/html/node_modules
```

```
  environment:
```

```
    APP_ENV: local
```

```
    APP_DEBUG: "true"
```

```
    APP_URL: http://localhost
```

```
    REDIS_HOST: redis
```

```
  depends_on:
```

```
    - redis
```

```
  web:
```

```
    image: nginx:1.27-alpine
```

```
    container_name: pi-laravel-nginx-test-test
```

```
    ports:
```

```
      - "8001:80"
```

```
    volumes:
```

```
      - ./var/www/html:cached
```

```
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf:ro
```

```
    depends_on:
```

```
      - app
```

```
  redis:
```

```
    image: redis:7-alpine
```

```
    container_name: pi-laravel-redis-test-test
```

```
    ports:
```

```
      - "6380:6379"
```

```
  vite:
```

```
    image: node:20
```

```
    container_name: pi-laravel-vite-test-test
```

```
    working_dir: /var/www/html
```

```
    volumes:
```

```
- ./var/www/html:cached
- ./node_modules:/var/www/html/node_modules
environment:
  VITE_DEV_SERVER_HOST: 0.0.0.0
  VITE_DEV_SERVER_PORT: 5173
command: npm run dev --host 0.0.0.0
ports:
  - "5174:5173"
depends_on:
  - app
```

```
n8n:
  image: n8nio/n8n:latest
  container_name: pi-laravel-n8n-test-test
  restart: unless-stopped
  ports:
    - "5679:5678"
  environment:
    - N8N_HOST=localhost
    - N8N_PORT=5678
    - N8N_PROTOCOL=http
    - NODE_ENV=production
    - WEBHOOK_URL=http://localhost:5678/
    - GENERIC_TIMEZONE=Europe/Madrid
    - DB_TYPE=sqlite
  volumes:
    - n8n_data:/home/node/.n8n
```

```
volumes:
  n8n_data:
```

En el docker-compose.yml se ha eliminado el servicio db y el volume db, ya que no es necesario que docker lo levante porque se conectará a AWS gracias a la configuración del .env.

A continuación se mostrarán un par de imágenes de la web desplegada en test para demostrar que funciona y tiene datos cargados desde la RDS:

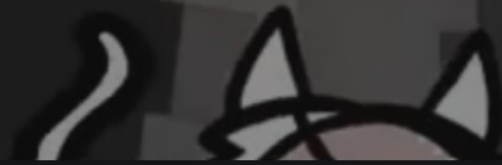


Buscar...



Mewgenics

Ya disponible





Deep Rock Galactic Survivor

20.00€ **19.00€**

Un spin-off de acción y supervivencia ambientado en el universo de Deep Rock Galactic.

Plataforma:

🎮 Steam

Añadir al carrito 🛒



Cesta



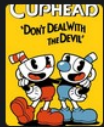
Deep Rock Galactic Survivor
Steam

19.00 €
~~20.00 €~~
1



Cuphead
Nintendo Switch

6.80 €
~~8.00 €~~
1



Cuphead
Xbox Series X

6.80 €
~~8.00 €~~
1

Resumen

Artículos 3

Total 32.60 €

Comprar