

Guía de configuración de VPS para Spring Boot, MySQL y Angular

1. Configuración inicial del servidor

1.1 Actualizar el sistema

`sudo apt update`

`sudo apt upgrade -y`

1.2 Crear un usuario no-root

Reemplaza "tuusuario" con el nombre que prefieras

`sudo adduser tuusuario`

Seguir las instrucciones y establecer una contraseña segura

Se puede dejar en blanco el resto de información personal

Añadir al grupo sudo para tener privilegios administrativos

`sudo usermod -aG sudo tuusuario`

Si estás usando autenticación por clave SSH, copia las claves

`sudo mkdir -p /home/tuusuario/.ssh`

`sudo cp ~/.ssh/authorized_keys /home/tuusuario/.ssh/`

`sudo chown -R tuusuario:tuusuario /home/tuusuario/.ssh`

`sudo chmod 775 /home/tuusuario/.ssh`

`sudo chmod 664 /home/tuusuario/.ssh/authorized_keys`

Es recomendable verificar que puedes iniciar sesión con el nuevo usuario antes de cerrar.

1.3 Instalar herramientas básicas

```
sudo apt install -y git curl wget htop
```

1.4 Configurar firewall

Instalar UFW

```
sudo apt install -y ufw
```

Configurar reglas básicas

```
sudo ufw allow ssh
```

```
sudo ufw allow http
```

```
sudo ufw allow https
```

Activar el firewall

```
sudo ufw enable
```

Verificar el estado

```
sudo ufw status
```

2. Instalación de Docker

2.1 Instalar Docker

Instalar dependencias

```
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
```

Agregar clave GPG de Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Agregar repositorio Docker

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
$(lsb_release -cs) stable"
```

Actualizar e instalar Docker

```
sudo apt update
```

```
sudo apt install -y docker-ce
```

Agregar tu usuario al grupo docker

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

Es recomendable cerrar sesión y volver a iniciar para que el cambio haga efecto.

2.2 Instalar Docker Compose

Instalar Docker Compose

```
sudo apt install -y docker-compose
```

3. Instalación y configuración de Nginx

3.1 Instalar Nginx

```
sudo apt install -y nginx
```

```
sudo systemctl enable nginx
```

```
sudo systemctl start nginx
```

3.2 Configurar directorio para la aplicación

Crear directorio para la aplicación

```
sudo mkdir -p /var/www/tuaplicacion/frontend
```

```
sudo chown -R $USER:$USER /var/www/tuaplicacion
```

Se cambia el puerto al 8080 para que no entre en conflicto con Apache que también lo teníamos instalado. Se deshabilita el Apache Tomcat para que el front tire de NGINX.

4. Configuración para desplegar la aplicación

4.1 Crear archivo docker-compose.yml

```
mkdir -p ~/app-deployment
```

```
cd ~/app-deployment
```

Crear el archivo docker-compose.yml:

```
version: '3'

services:
  # Servicio de base de datos MySQL
  mysql:
    image: mysql:8.0 # Imagen oficial MySQL versión 8.0
    restart: always # Reinicia automáticamente si el contenedor falla
    environment:
      # Contraseña para el usuario root (administrador de todo MySQL)
      MYSQL_ROOT_PASSWORD: rootpassword # Cambiar a una contraseña segura
      # Nombre de la base de datos para la aplicación
      MYSQL_DATABASE: app_db
      # Usuario normal con permisos solo para la base de datos app_db
      MYSQL_USER: user # Cambiar a un usuario personalizado
      # Contraseña para el usuario normal
      MYSQL_PASSWORD: password # Cambiar a una contraseña segura
    volumes:
      # Guardar datos persistentes aunque se elimine el contenedor
      - mysql-data:/var/lib/mysql
    ports:
      # Exponer MySQL solo localmente por seguridad
      - "127.0.0.1:3306:3306" # Solo accesible desde el servidor

# Herramienta de administración de la base de datos
```

adminer:

image: adminer *# Herramienta ligera para administrar bases de datos*

restart: always

ports:

Exponer en puerto 8081 localmente

- "127.0.0.1:8081:8080"

Aplicación Spring Boot

backend:

build: ./backend *# Construir imagen a partir del Dockerfile en carpeta backend*

restart: always

depends_on:

- mysql *# Esperar a que MySQL esté listo antes de iniciar*

environment:

Configuración para conectar a MySQL

"mysql" es el nombre del servicio definido arriba

SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/app_db

Usuario normal (no root) para la conexión

SPRING_DATASOURCE_USERNAME: user

Misma contraseña del usuario normal

SPRING_DATASOURCE_PASSWORD: password

ports:

Exponer API solo localmente por seguridad

- "127.0.0.1:8080:8080" *# Solo accesible desde el servidor*

Definición de volúmenes persistentes

volumes:

mysql-data: *# Almacenamiento persistente para MySQL*

4.2 Configurar Nginx para la aplicación

Crear configuración de Nginx:

```
sudo nano /etc/nginx/sites-available/reto4
```

Contenido del archivo:

```
server {  
    listen 80;  
    server_name tudominio.com; # Cambiar a tu dominio real  
  
    # Frontend  
    location / {  
        root /var/www/reto4/frontend;  
        try_files $uri $uri/ /index.html;  
    }  
  
    # Backend API  
    location /api/ {  
        proxy_pass http://localhost:8080/  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto $scheme;  
    }  
  
    # Adminer (administración de BD)  
    location /adminer/ {  
        proxy_pass http://localhost: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;  
    }  
}
```

```
}
```

Activar la configuración:

```
sudo ln -s /etc/nginx/sites-available/reto4 /etc/nginx/sites-enabled/
```

```
sudo nginx -t # Verificar la configuración
```

```
sudo systemctl reload nginx
```

5. SSL/HTTPS con Certbot (Let's Encrypt)

5.1 Instalar Certbot

```
sudo apt install -y certbot python3-certbot-nginx
```

5.2 Obtener certificado SSL

```
sudo certbot --nginx -d tudominio.com
```

Certbot

```
ssl_certificate_key /etc/letsencrypt/live/algarciasi.com/privkey.pem; # managed by  
Certbot
```

```
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
```

```
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```

```
}
```

```
server {
```

```
    if ($host = www.algarciasi.com) {
```

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

```
    } # managed by Certbot
```

```
    if ($host = algarciasi.com) {
```

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

```
    } # managed by Certbot
```

```
listen 80;

server_name algarciasi.com www.algarciasi.com;

return 404; # managed by Certbot

}
```

6. Preparar el despliegue de la aplicación

6.1 Directorio para Spring Boot

Crear Dockerfile para Spring Boot:

```
nano ~/app-deployment/backend/Dockerfile
```

Contenido:

dockerfile

```
FROM openjdk:21-jdk-slim
WORKDIR /app
COPY target/*.jar app.jar
# Activar perfil de producción
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod", "app.jar"]
```

6.3 Configurar Spring Boot para entornos múltiples

En tu proyecto Spring Boot, usa archivos YAML para mejor compatibilidad con distintos entornos:

- Crea un archivo `application.yml` en la carpeta `src/main/resources` para desarrollo local:

```
spring:
  application:
    name: SGVE-SpringBoot
  datasource:
    url: jdbc:mysql://localhost:3306/vacantes_BBDD_2025_RETO?serverTimezone=UTC
    username: root
    password: mysql
    driver-class-name: com.mysql.cj.jdbc.Driver
  profiles:
    active: dev
  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect

springdoc:
  api-docs:
    path: /v3/api-docs
  swagger-ui:
    path: /swagger-ui.html

jwt:
  secret: TuClaveSecretaDebeSerLoSuficientementeLargaParaSerSegura
  expiration: 86400000
```

- Crea un archivo `application-prod.yml` en la misma carpeta para producción:

```
spring:
  application:
    name: SGVE-SpringBoot
  datasource:
    url: jdbc:mysql://mysql:3306/vacantes_BBDD_2025_RETO
    username: vhpdev
    password: password@vhpdev
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
    show-sql: false

springdoc:
  api-docs:
    path: /v3/api-docs
  swagger-ui:
    path: /swagger-ui.html

jwt:
  secret: TuClaveSecretaDebeSerLoSuficientementeLargaParaSerSegura
  expiration: 86400000
```

- Si tienes un archivo `application.properties`, puedes eliminarlo o renombrarlo.

El archivo de producción será usado automáticamente en el servidor gracias al parámetro en el Dockerfile, mientras que tu configuración local seguirá funcionando para desarrollo.

6.4 Compilar y transferir la aplicación Spring Boot

En tu máquina de desarrollo:

Compilar con Maven (nuestro caso), Run as → Maven build

```
./mvnw clean package
```

O con Gradle

```
./gradlew clean build
```

Transferir al servidor:

Nos conectamos a nuestra ruta donde se ha generado el JAR de Spring y lanzamos un scp hacia la ruta destino del servidor:

```
scp Reto4DAW-1.jar ubuntu@algarciasi.com:/home/ubuntu
```

6.5 Iniciar los servicios Docker en el servidor

Una vez transferido el JAR al servidor, inicia los contenedores:

Comprobar que todo está en su lugar

```
ssh tuusuario@tu-servidor "ls -la ~/app-deployment/backend/target/"
```

Iniciar los contenedores

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose up -d"
```

Verificar que los contenedores están funcionando

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose ps"
```

Solución de problemas comunes:

Ver logs para detectar errores

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose logs backend"
```

Si hay errores de versión de Java

Edita el Dockerfile para usar la versión correcta de Java

```
ssh tuusuario@tu-servidor "nano ~/app-deployment/backend/Dockerfile"
```

Si hay errores de 'ContainerConfig' o similares

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose down -v"
```

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose rm -f backend"
```

```
ssh tuusuario@tu-servidor "docker rmi app-deployment_backend"
```

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose up -d"
```

6.6 Verificar la aplicación

Una vez que los contenedores estén funcionando, verifica que puedes acceder a:

- * Frontend (Angular): <http://tudominio.com/>
- * Backend (Spring Boot): <http://tudominio.com/api/>
- * Administrador de BD (Adminer): <http://tudominio.com/adminer/>

Para Adminer, usa estos datos para conectar:

- * Sistema: MySQL
- * Servidor: mysql (el nombre del servicio en docker-compose)
- * Usuario: vhpdev (o el usuario que configuraste)
- * Contraseña: password@vhpdev (o la contraseña que configuraste)
- * Base de datos: vacantes_BBDD_2025_RETO

6.7 Configurar rutas correctas para Angular moderno y Spring Boot

Para aplicaciones modernas, es importante configurar correctamente las rutas:

Verificar la estructura de archivos Angular

```
ls -la /var/www/reto/frontend/
```

Para Angular moderno (versión 18+), la estructura de compilación incluye subdirectorios como "browser":

Modificar la configuración de Nginx para apuntar al subdirectorio correcto

```
sudo nano /etc/nginx/sites-available/tuaplicacion
```

Actualiza la configuración de Nginx con las rutas correctas:

```
server {  
    server_name tudominio.es;  
  
    # Frontend (Angular moderno)  
    location / {  
        root /var/www/reto/frontend/sgve-angular/browser;  
        index index.html;  
        try_files $uri $uri/ /index.html;  
    }  
  
    # Backend API (Spring Boot con prefijo /api)  
    location /api/ {  
        proxy_pass http://localhost:8080; # Sin barra final para no duplicar /api  
        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; # Importante para Spring Security  
        proxy_set_header X-Forwarded-Port 443;  
    }  
}
```

```
    proxy_set_header X-Forwarded-Host $host;
}

# Adminer (administración de BD)
location /adminer/ {
    proxy_pass http://localhost: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;
}

listen 443 ssl; # managed by Certbot
# Configuración SSL añadida por Certbot
}

server {
    if ($host = tudominio.es) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    server_name tudominio.es;
    return 404; # managed by Certbot
}
```

Verificar la configuración y reiniciar Nginx

```
sudo nginx -t
```

```
sudo systemctl restart nginx
```

6.8 Verificar la correcta comunicación entre componentes

Para comprobar que todo funciona correctamente:

Verificar que el backend responde desde la maquina local con un curl:

```
curl -v https://tudominio.es/api/vacantes
```

Revisar logs de backend si hay problemas

```
cd ~/app-deployment && docker-compose logs backend
```

Revisar logs de Nginx

```
sudo tail -n 50 /var/log/nginx/error.log
```

Si al intentar registrarte o usar otras funcionalidades encuentras problemas, verifica:

1. Errores CORS en la consola del navegador (F12 > Consola)
2. Problemas de autenticación en los logs del backend
3. Que las URLs en el frontend apunten al dominio correcto (no a localhost)

7. Configuración de SSL/HTTPS

Un sitio web profesional debe contar con HTTPS para proteger la comunicación entre usuarios y servidor.

7.1 Instalar Certbot

Certbot es una herramienta que permite obtener certificados SSL gratuitos de Let's Encrypt.

Instalar Certbot y el plugin para Nginx

```
sudo apt install -y certbot python3-certbot-nginx
```

7.2 Obtener certificado SSL

Reemplaza tudominio.com con tu dominio real

```
sudo certbot --nginx -d tudominio.com
```

Durante el proceso, te solicitará:

- Un correo electrónico para notificaciones importantes
- Aceptar los términos de servicio
- Si quieres redirigir automáticamente HTTP a HTTPS (recomendado)

Certbot modificará automáticamente tu configuración de Nginx para usar HTTPS.

7.3 Verificar configuración

Comprobar que la renovación de certificados está configurada

```
sudo systemctl status certbot.timer
```

Probar la renovación (sin hacer cambios reales)

```
sudo certbot renew --dry-run
```

8. Mantenimiento y operaciones comunes

8.1 Actualizar la aplicación

Para actualizar el frontend (Angular):

En tu máquina local

```
ng build
```

Transferir archivos al servidor

```
scp Reto4DAW-1.jar ubuntu@algarciasi.com:/home/ubuntu
```



```
sudo cp -r /home/ubuntu/reto-angular/browser/* /var/www/reto4/frontend/reto-angular/browser/
```

En el servidor

```
cd ~/app-deployment && docker-compose restart backend
```

8.2 Comandos Docker útiles

Ver estado de los contenedores

```
docker-compose ps
```

Ver logs

```
docker-compose logs -f backend
```

Reiniciar un servicio específico

```
docker-compose restart backend
```

Detener todos los servicios

```
docker-compose down
```

Iniciar todos los servicios

```
docker-compose up -d
```

8.4 Configuración de entornos de Angular para producción

Para aplicaciones Angular, es crucial configurar correctamente los entornos:

- **Configura los archivos de entorno**:**

- En `src/environments/environment.ts` (desarrollo):

```
export const environment = {
```

```
production: false,  
apiUrl: 'http://localhost:8080'  
};
```

- En `src/environments/environment.prod.ts` (producción):

```
export const environment = {  
  production: true,  
  apiUrl: 'https://tudominio.es' // URL de producción  
};
```

- **Asegúrate de que angular.json tenga la configuración correcta**:**

```
"configurations": {  
  "production": {  
    "budgets": [...],  
    "fileReplacements": [  
      {  
        "replace": "src/environments/environment.ts",  
        "with": "src/environments/environment.prod.ts"  
      }  
    ],  
    "outputHashing": "all"  
  },  
  "development": {...}  
}
```

- **Compila específicamente para producción**:**

```
ng build --configuration production
```

- **Verifica que no quedan referencias a localhost**:**

```
grep -r "localhost" dist/
```

- **Transfiere los archivos al servidor**:**

```
scp -r dist/* tuusuario@tu-servidor:/var/www/reto/frontend/sgve-angular/browser/
```

- **Limpia la caché del navegador**** después de actualizar la aplicación

8.5 Solución de problemas de conexión frontend-backend

Si el frontend no puede conectarse correctamente al backend:

- **Verifica las peticiones en el navegador**:**

- Abre las DevTools (F12)
- Ve a la pestaña Network
- Intenta ejecutar la operación problemática
- Revisa a qué URL se están enviando las peticiones

- **Configura CORS en Spring Boot**:**

```
@Override
```

```
public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
        .allowedOrigins("http://localhost:4200", "https://tudominio.es")
        .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
        .allowedHeaders("*")
        .allowCredentials(true);
}
```

- **Configura cabeceras HTTPS en Nginx**:**

```
location /api/{  
    proxy_pass http://localhost:8080;  
    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; # Importante para HTTPS  
    proxy_set_header X-Forwarded-Port 443;  
}
```

- **Reinicia los servicios después de cambios**:**

Para el backend

```
ssh tuusuario@tu-servidor "cd ~/app-deployment && docker-compose restart backend"
```

Para Nginx (solo si cambias su configuración)

```
ssh tuusuario@tu-servidor "sudo systemctl restart nginx"
```

En tu máquina de desarrollo:

Compilar la aplicación Angular

`ng build --prod` # o simplemente: *ng build* (en Angular más reciente)

Luego, transferir los archivos al servidor:

Desde tu máquina local, transferir archivos compilados

```
scp -r dist/* tuusuario@tu-servidor:/var/www/tuaplicacion/frontend/
```

9. SCP

9.1 Angular

Enviar ficheros de maquina local a servidor

```
scp -r dist/reto-angular ubuntu@algarciasi.com:/home/ubuntu/
```

Configurar permisos superusuario

```
sudo rm -rf /var/www/reto4/frontend/reto-angular/browser/*
```

```
sudo cp -r /home/ubuntu/reto-angular/browser/* /var/www/reto4/frontend/reto-  
angular/browser/
```

```
sudo chown -R www-data:www-data /var/www/reto4/frontend/reto-angular/browser
```

```
sudo find /var/www/reto4/frontend/reto-angular/browser -type d -exec chmod 755 {} \;
```

```
sudo find /var/www/reto4/frontend/reto-angular/browser -type f -exec chmod 644 {} \;
```

```
sudo systemctl reload nginx
```

9.2 Spring

Enviar ficheros de maquina local a servidor

```
PS D:\Formacion - Estudio\FP - CICLO SUPERIOR\DAW UNIR\2º\Entorno
```

```
Servidor\WS_TOOLS_CLASE\reto4daw_backend-main\target> scp Reto4DAW-1.jar
```

```
ubuntu@algarciasi.com:/home/ubuntu
```

```
ubuntu@algarciasi.com's password:
```

```
Reto4DAW-
```

```
1.jar
```

```
100% 64MB 1
```

```
1.1MB/s 00:05
```

Mover jar a la ruta y ejecutarlo

```
root@vps-7591ae7c:/var/www/reto4/backend# sudo cp -r /home/ubuntu/Reto4DAW-1.jar  
/var/www/reto4/backend/
```

```
java -jar Reto4DAW-1.jar
```

10. Bind9

Se instala bind9 para gestionar los subdominios desde el propio servidor y no depender de IONOS.

```
root@vps-7591ae7c:/usr/share/doc/bind9# cat /etc/bind/named.conf.local

// (ZONA DIRECTA) Archivo para búsquedas directas

zone "algarciasi.com" {
    type master;
    file "/etc/bind/db.algarciasi.com";
    allow-transfer {213.251.188.141;};
};

// (ZONA INVERSA) Archivo para búsquedas inversas

zone "155.80.151.in-addr.arpa" {
    type master;
    file "/etc/bind/db.155.80.151";
    allow-transfer {213.251.188.141;};
};
```

```
root@vps-7591ae7c:/usr/share/doc/bind9# cat /etc/bind/db.algarciasi.com

$TTL 86400
@ IN SOA ns1.algarciasi.com. root.algarciasi.com. (
        2025042102; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        86400 ) ; Negative Cache TTL
;
@ IN NS ns1.algarciasi.com.
ns1 IN A 151.80.155.91
```

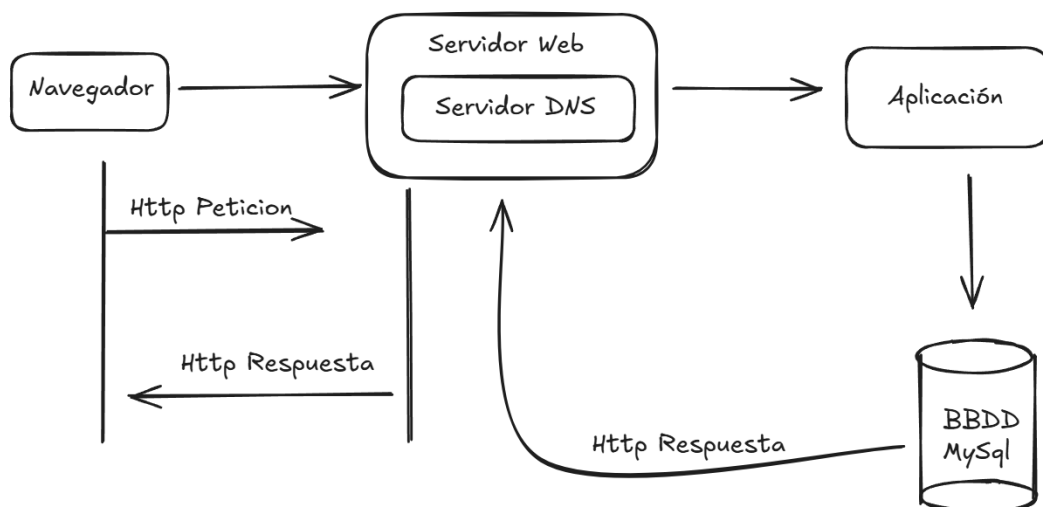
```

@      IN  A    151.80.155.91
www     IN  A    151.80.155.91
tienda  IN  A    151.80.155.91
blog    IN  A    151.80.155.91
ownercheck IN TXT 6702ed13
mail    IN  A    151.80.155.91
@      IN  MX   10  mail.algarcias.com.
quepasa IN  A    151.80.155.91
@      IN  TXT  "estrella galicia"
warbasico IN A    151.80.155.91
examen  IN  A    151.80.155.91
tomcatrealmadrid IN A 151.80.155.91
@      IN  TXT  "Derbi Champions... ¿revancha del minuto 93?"
ftpatletico IN A 151.80.155.91



















```

11. Flujo Web

Esquema Web



Respuestas del servidor web

| | | |
|---|---|---|
| 200 OK Request succeeded  | 201 Created Resource created  | 202 Accepted Accepted request  |
| 200 No Content Accepted request  | 301 Moved Permanently Accepted request  | 302 Found Temporarily moved  |
| 304 Not Modified Not modified  | 400 Bad Request Bad request error  | 401 Unauthorized Needs authentication  |
| 403 Forbidden Access Forbidden  | 404 Not Found Resource Not Found  | 405 Not Allowed Method not allowed  |
| 408 Req Timeout Request timed out  | 500 Internal Server Error Server error  | 501 Not Implemented Not implemented  |
| 502 Bad Gateway Bad gateway error  | 502 Service Unavailable Service unavailable  | 504 Gateway Timeout Gateway Timeout  |