**Documentación para el Despliegue y Mantenimiento de la Red Social Basada en Hobbies**

**1. Introducción**

Esta documentación describe los procedimientos y pasos necesarios para realizar el **despliegue** de la plataforma de la red social basada en hobbies en el entorno de producción, así como las prácticas de **mantenimiento** continuo necesarias para garantizar su operatividad y mejoras constantes.

**2. Requisitos Previos para el Despliegue**

**2.1. Requisitos de Hardware**

* Servidor físico o virtual con las siguientes especificaciones mínimas:
  + CPU: 4 núcleos.
  + RAM: 8 GB.
  + Espacio en disco: 100 GB.
  + Conexión de red: 1 Gbps.

**2.2. Requisitos de Software**

* **Sistema Operativo**: Ubuntu 22.04 LTS o superior.
* **Base de Datos**: PostgreSQL 13 o superior.
* **Servidor Web**: Nginx (para manejo de tráfico web).
* **Backend**: Node.js (version 18+), NestJS.
* **Frontend**: Next.js (versión 15+).
* **Contenedores**: Docker y Docker Compose (si se utiliza contenedores).
* **Certificado SSL**: Para encriptación HTTPS (por ejemplo, Let's Encrypt).
* **Herramientas de CI/CD**: GitHub Actions (o cualquier herramienta preferida).

**3. Procedimiento de Despliegue**

**3.1. Despliegue del Backend (NestJS)**

1. **Configuración del servidor**
   * Actualizar el sistema operativo:

bash

Copiar código

sudo apt update && sudo apt upgrade -y

sudo apt install -y curl git build-essential

* + Instalar Node.js y npm:

bash

Copiar código

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt install -y nodejs

1. **Instalar dependencias y preparar el entorno**
   * Clonar el repositorio del backend desde GitHub:

bash

Copiar código

git clone https://github.com/tuusuario/backend-repo.git

cd backend-repo

* + Instalar las dependencias del proyecto:

bash

Copiar código

npm install

1. **Configuración de la base de datos**
   * Crear y configurar la base de datos en PostgreSQL:

bash

Copiar código

sudo -u postgres psql

CREATE DATABASE hobbies;

CREATE USER hobbies\_user WITH PASSWORD 'tu\_contraseña';

GRANT ALL PRIVILEGES ON DATABASE hobbies TO hobbies\_user;

* + Configurar la conexión de la base de datos en el archivo .env de NestJS.

1. **Compilación y despliegue**
   * Compilar el proyecto y ejecutar el backend:

bash

Copiar código

npm run build

npm run start:prod

**3.2. Despliegue del Frontend (Next.js)**

1. **Instalación de dependencias**
   * Clonar el repositorio del frontend:

bash

Copiar código

git clone https://github.com/tuusuario/frontend-repo.git

cd frontend-repo

* + Instalar dependencias:

bash

Copiar código

npm install

1. **Configuración de variables de entorno**
   * Configurar las variables de entorno necesarias en el archivo .env (por ejemplo, claves API, URLs del backend, etc.).
2. **Compilación y despliegue**
   * Construir la versión de producción:

bash

Copiar código

npm run build

* + Iniciar el servidor:

bash

Copiar código

npm run start

**3.3. Configuración de Nginx y SSL**

1. **Instalar Nginx**:

bash

Copiar código

sudo apt install nginx

1. **Configurar Nginx para el backend y el frontend**
   * Crear un archivo de configuración de Nginx para el backend y el frontend.
   * Configurar el proxy inverso para redirigir las solicitudes a las aplicaciones backend y frontend.

Ejemplo de configuración:

nginx

Copiar código

server {

listen 80;

server\_name midominio.com www.midominio.com;

location / {

proxy\_pass http://localhost:3000; # Para el frontend (Next.js)

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

location /api/ {

proxy\_pass http://localhost:4000; # Para el backend (NestJS)

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

1. **Instalar certificado SSL (Let’s Encrypt)**
   * Usar Certbot para instalar un certificado SSL gratuito:

bash

Copiar código

sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d midominio.com -d www.midominio.com

**3.4. CI/CD con GitHub Actions**

1. **Crear un flujo de trabajo de CI/CD**:
   * Definir un archivo de flujo de trabajo (.github/workflows/deploy.yml) para automatizar el despliegue.

Ejemplo básico de configuración:

yaml

Copiar código

name: Deploy to Production

on:

push:

branches:

- main

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '18'

- name: Install Dependencies

run: npm install

- name: Build and Deploy

run: npm run build && npm run deploy

**4. Procedimiento de Mantenimiento**

**4.1. Actualización del Sistema**

* **Actualización de dependencias**: Se debe verificar y actualizar periódicamente las dependencias del backend y frontend para mantener la seguridad.

bash

Copiar código

npm update

* **Actualización de la base de datos**: Si se requieren actualizaciones de esquema, crear migraciones usando Prisma o TypeORM según sea necesario.

**4.2. Monitoreo del Sistema**

* **Monitoreo de servidor**: Utilizar herramientas como **Uptime Robot** o **Prometheus** para monitorear el tiempo de actividad y el rendimiento del servidor.
* **Monitoreo de aplicaciones**: Usar **Sentry** o **LogRocket** para capturar errores y hacer un seguimiento de los problemas de la aplicación.

**4.3. Copias de Seguridad**

* Realizar copias de seguridad regulares de la base de datos utilizando herramientas como **pg\_dump** para PostgreSQL.
* Configurar **backups automáticos** para los servidores de archivos y bases de datos.

**4.4. Gestión de Errores y Soporte**

* Utilizar sistemas de tickets (como **Jira** o **Zendesk**) para registrar errores y gestionar el soporte de los usuarios.
* Realizar revisiones periódicas de errores y optimizaciones de la plataforma.

**4.5. Actualización de Funcionalidades**

* Realizar actualizaciones de funcionalidades según la retroalimentación de los usuarios, pruebas de calidad y necesidades comerciales.
* Utilizar el sistema de **CI/CD** para desplegar nuevas versiones de manera controlada y eficiente.

**4.6. Seguridad Continuada**

* Realizar auditorías de seguridad periódicas.
* Implementar **actualizaciones de seguridad** de manera continua para evitar vulnerabilidades.