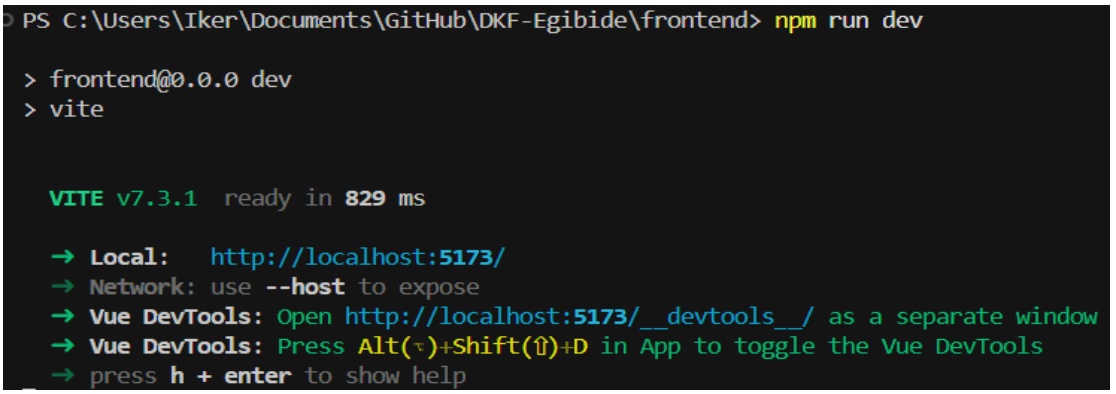
Documentación
Despliegue

1. LOCAL

a. Arranque del servidor de desarrollo del frontend (Vue)

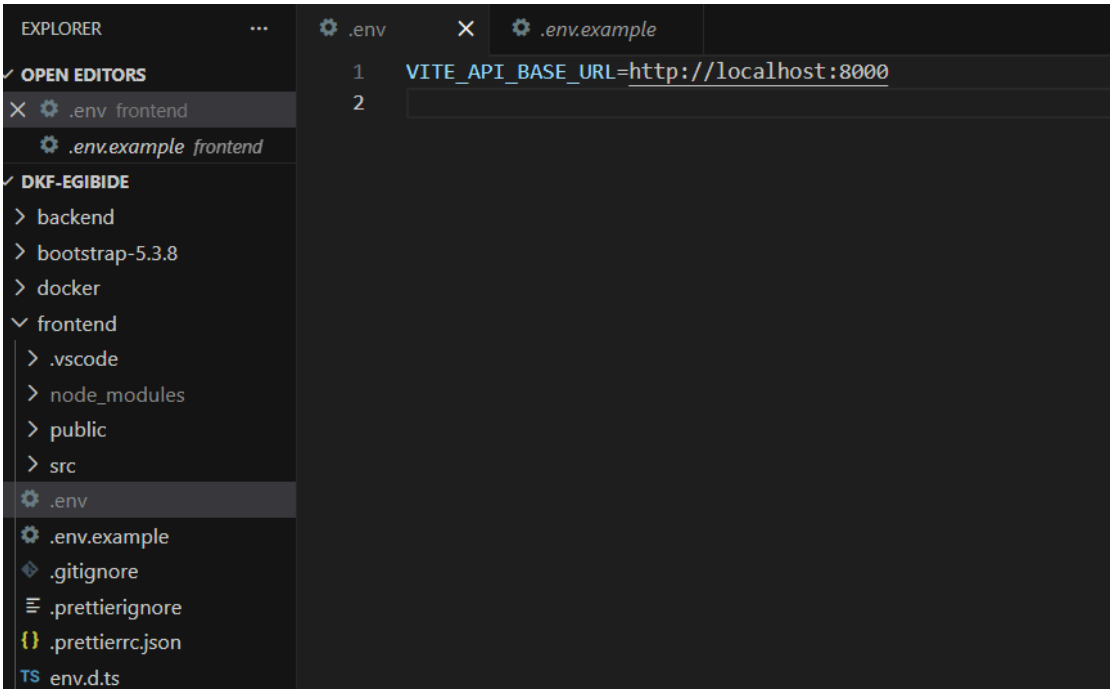
Una vez abierto el editor de código con nuestro proyecto finalizado, hemos abierto la terminal y arrancamos el servidor de desarrollo frontend de Vue mediante el comando **`npm run dev`**

Directorio de trabajo desde el que se ejecuta el comando:
C:\Users\Iker\Documents\GitHub\DKF-Egibide\frontend



b. Configuración del archivo de entorno del frontend (.env)

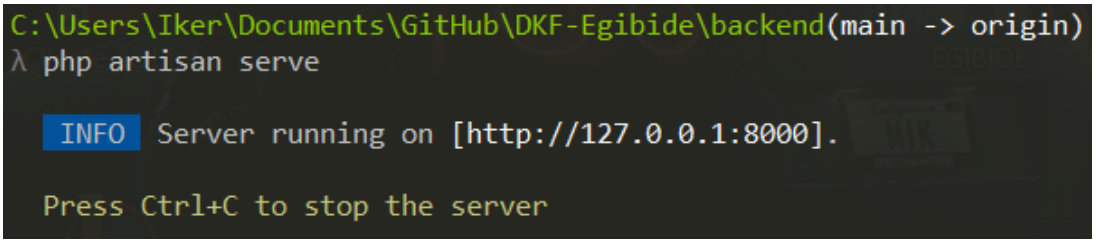
Si no disponemos del archivo **.env** en el directorio del frontend, debemos copiar el archivo **.env.example** en el mismo directorio y renombrarlo como **.env**, para poder ejecutar el proyecto en entorno local.



c. Arranque del servidor de desarrollo del backend (Laravel)

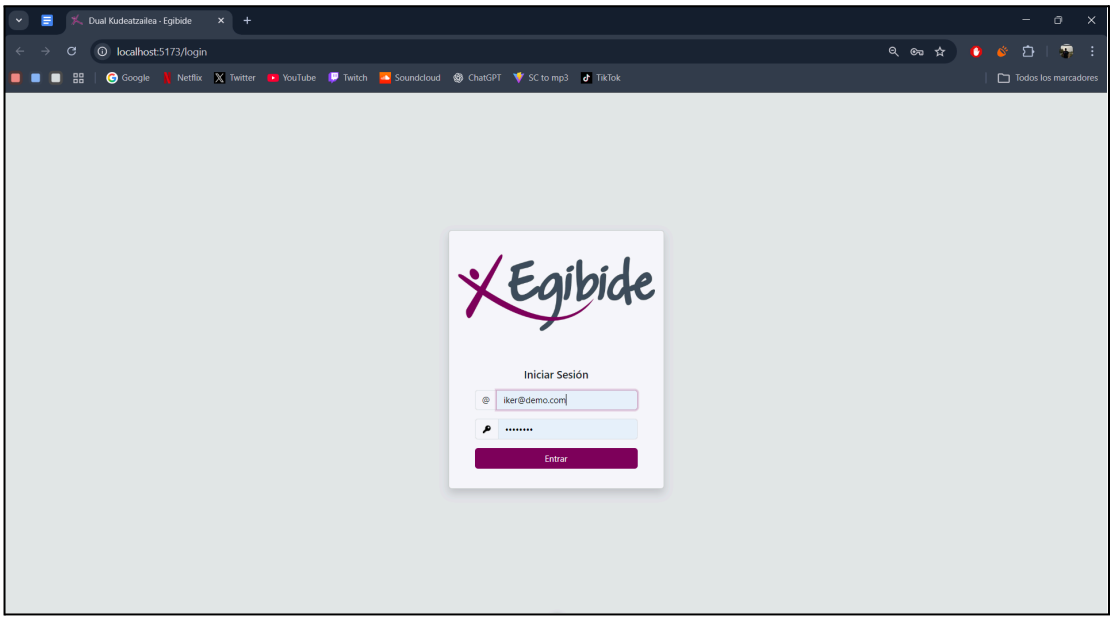
Después de arrancar el servidor de desarrollo de Vue, hemos abierto Laragon y, desde su terminal, ejecutamos el comando **`php artisan serve`** para iniciar el servidor de desarrollo de Laravel.

Directorio de trabajo desde el que se ejecuta el comando:
C:\Users\Iker\Documents\GitHub\DKF-Egibide\backend



d. Acceso a la aplicación en entorno local

Con ambos entornos de desarrollo en funcionamiento, accedemos a la aplicación desde el navegador mediante la ruta **`http://localhost:5173/login`**, la cual también podemos acceder escribiendo “o” en la terminal del editor de código.

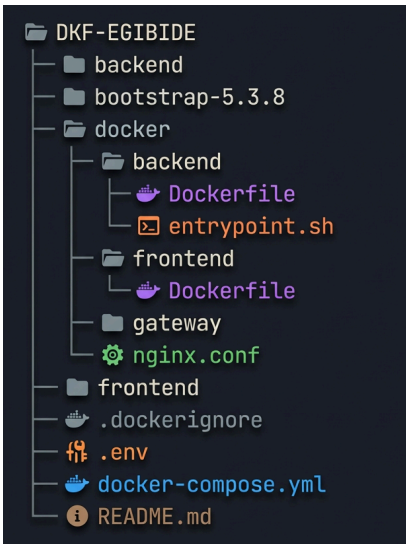


2. PRODUCCIÓN

a. Introducción:

En este documento se explica cómo desplegar la aplicación **Vue.js + Laravel** en un servidor **Debian** (OpenNebula) utilizando **Docker** y **Docker Compose**, con un **gateway Nginx** que unifica frontend y backend, y acceso externo seguro mediante **Ngrok**.

b. Estructura del proyecto:



Descripción de carpetas y archivos:

- **frontend/:** código del frontend (SPA) y .dockerignore.
- **backend/:** código de Laravel (incluye Bootstrap 5.3.8).
- **docker/backend/:** Dockerfile y entrypoint del backend.
- **docker/frontend/:** Dockerfile del frontend.
- **.env:** variables de entorno.
- **docker-compose.yml:** orquestación de contenedores.

c. Configuración Docker y Docker Compose:

El archivo docker-compose.yml en la raíz del proyecto define todos los servicios y cómo interactúan.

```yaml
version: '3.8'

services:
  # Gateway - Nginx que unifica frontend y backend
  gateway:
    image: nginx:alpine
```

```
    container_name: dkf-gateway
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./docker/nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - frontend
      - backend
    restart: always
    networks:
      - dkf-network

  # Frontend - Vue.js compilado
  frontend:
    build:
      context: ./frontend
      dockerfile: ../docker/frontend/Dockerfile
    expose:
      - "80"
    container_name: dkf-frontend
    restart: always
    networks:
      - dkf-network

  # Backend - Laravel con PHP-FPM
  backend:
    build:
      context: .
      dockerfile: ./docker/backend/Dockerfile
    expose:
      - "9000"
    container_name: dkf-backend
    environment:
      DB_CONNECTION: mysql
      DB_HOST: database
      DB_PORT: 3306
      DB_DATABASE: ${DB_DATABASE}
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
    volumes:
```

```
      - storage_data:/var/www/html/storage
    depends_on:
      - database
    restart: always
    networks:
      - dkf-network

  # Base de Datos - MySQL
  database:
    image: mysql:8.0
    container_name: dkf-database
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      MYSQL_DATABASE: ${DB_DATABASE}
      MYSQL_USER: ${DB_USERNAME}
      MYSQL_PASSWORD: ${DB_PASSWORD}
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    networks:
      - dkf-network
    ports:
      - "3306:3306"

volumes:
  db_data:
  storage_data:

networks:
  dkf-network:
    driver: bridge
```

Define 4 servicios:

- **database:** MySQL 8.0 con persistencia de datos.
- **backend:** Laravel ejecutándose con PHP-FPM.
- **frontend:** aplicación Vue compilada y servida como SPA.
- **gateway:** Nginx como reverse proxy unificando frontend y backend.

Volúmenes:
db_data: datos persistentes de MySQL.

storage_data: directorio storage/ de Laravel.

Red:
Todos los contenedores se comunican a través de la red interna dkf-network creada por Docker Compose.

d. Configuración del Backend (Laravel):

Dockerfile (docker/backend/Dockerfile)

- PHP 8.4 con PHP-FPM.
- Extensiones PHP necesarias para Laravel instaladas.
- Composer instalado desde la imagen oficial.
- Código del backend copiado a /var/www/html.
- Instalación de las dependencias de Laravel (optimizadas para producción).
- Permisos configurados para storage/ y bootstrap/cache.
- entrypoint.sh ejecuta las tareas iniciales (migraciones y seeders).
- Puerto 9000 expuesto para PHP-FPM.

```
# Laravel con PHP-FPM

FROM php:8.4-fpm

WORKDIR /var/www/html

# Instalar dependencias del sistema
RUN apt-get update && apt-get install -y \
    git \
    curl \
    zip \
    unzip \
    libpng-dev \
    libonig-dev \
    libxml2-dev \
    libzip-dev \
    && rm -rf /var/lib/apt/lists/*

# Instalar extensiones PHP
RUN docker-php-ext-install \
    pdo \
    pdo_mysql \
    mbstring \
```

```
    exif \
    pcntl \
    bcmath \
    gd \
    zip

# Instalar Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Copiar código Laravel
COPY ./backend /var/www/html

# Instalar dependencias de Laravel
RUN composer install --no-dev --optimize-autoloader --no-interaction

# Permisos
RUN chown -R www-data:www-data /var/www/html \
    && chmod -R 775 /var/www/html/storage /var/www/html/bootstrap/cache

# Script inicial que ejecuta las migraciones y los seeders.
COPY ./docker/backend/entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

ENTRYPOINT ["/entrypoint.sh"]

EXPOSE 9000
```

entrypoint.sh

Este script se ejecuta cada vez que se arranca el contenedor del backend y se encarga de dejar Laravel listo para su uso en producción.

- Espera a que la base de datos MySQL esté disponible.
- Copia el archivo .env desde .env.example si no existe.
- Genera la APP_KEY de Laravel.
- Ejecuta las migraciones y los seeders en modo forzado.
- Limpia y genera las cachés de configuración y rutas.
- Arranca PHP-FPM en primer plano.

```
#!/bin/sh

set -e

# Esperar a que la base de datos esté lista
echo "Esperando a MySQL..."
until php -r "new PDO('mysql:host=' . getenv('DB_HOST') . ';port=' .
getenv('DB_PORT'), getenv('DB_USERNAME'), getenv('DB_PASSWORD'));"
2>/dev/null; do
  echo -n "."
  sleep 1
done

echo "MySQL listo"

# Generar APP_KEY si no existe
if [ ! -f /var/www/html/.env ]; then
    cp /var/www/html/.env.example /var/www/html/.env
fi

php artisan key:generate --force
php artisan migrate --force --seed
php artisan config:clear
php artisan config:cache
php artisan route:cache

exec php-fpm
```

CORS (backend/config/cors.php)
Configura los permisos de Cross-Origin Resource Sharing (CORS) para el backend Laravel.

- Permite solicitudes desde el frontend (FRONTEND_URL) y la IP local (192.168.50.39).
- Permite solicitudes desde Ngrok (*.ngrok-free.app).
- Aplica a las rutas api/* y sanctum/csrf-cookie.
- Métodos y cabeceras permitidos: todos (*).
- Soporta credenciales (cookies y headers de autenticación).
- Caché de preflight desactivada (max_age = 0).

```php
<?php

return [

    'paths' => ['api/*', 'sanctum/csrf-cookie'],

    'allowed_methods' => ['*'],

    'allowed_origins' => ['http://192.168.50.39', 'http://192.168.50.39:80'],

    'allowed_origins_patterns' => [],

    'allowed_headers' => ['*'],

    'exposed_headers' => [],

    'max_age' => 0,

    'supports_credentials' => true,

];
```

Variables de entorno (.env)

- DB_ROOT_PASSWORD=root
- DB_DATABASE=dkf_egibide
- DB_USERNAME=dkf_user
- DB_PASSWORD=dkf_pass

e. Configuración del Frontend (Vue):

Dockerfile (docker/frontend/Dockerfile)
Build de múltiples etapas para la aplicación Vue.js.

Etapa 1 – Compilación:

- Imagen base: node:20-alpine.
- Instala dependencias usando npm ci.
- Copia el código fuente y compila la aplicación con npm run build.
- Variable de entorno VITE_API_BASE_URL configurable para la API.

Etapa 2 – Servidor:

- Imagen base: nginx:alpine.
- Copia la carpeta dist/ generada por Vue al directorio de Nginx /usr/share/nginx/html.
- Expone el puerto 80.
- Comando por defecto: nginx -g 'daemon off;' para mantener el contenedor en ejecución.

```
# Build de múltiples etapas para Vue.js

# Etapa 1: Compilar Vue
FROM node:20-alpine AS build

WORKDIR /app

# Copiar package.json del frontend
COPY package*.json ./

# Instalar dependencias
RUN npm ci

# Copiar código fuente
COPY . ./

# Variable de entorno para la api
ENV VITE_API_BASE_URL=""

# Build para producción
RUN npm run build

# Etapa 2: Servidor Nginx
FROM nginx:alpine

# Copiar archivos compilados
COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

f. Configuración del Nginx (Apache)

Este archivo configura el gateway Nginx que unifica el acceso al frontend y al backend.

- Define un upstream para el frontend (frontend:80).
- Redirige las solicitudes a /api hacia el backend Laravel usando PHP-FPM.
- Redirige todas las demás solicitudes hacia el frontend Vue compilado.
- Configura cache de 1 año para assets estáticos (JS, CSS, imágenes, fuentes).
- Registra accesos y errores en /var/log/nginx/.

```
# Gateway que unifica frontend y backend

upstream frontend {
    server frontend:80;
}

server {
    listen 80;
    server_name localhost;

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    # ========================================
    # BACKEND API
    # ========================================
    location /api {
        include fastcgi_params;
        fastcgi_pass backend:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME /var/www/html/public/index.php;
        fastcgi_param SCRIPT_NAME /index.php;
    }

    # ========================================
    # FRONTEND
    # ========================================
    location / {
        proxy_pass http://frontend;
        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;
    }

    # Assets estáticos con cache
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        proxy_pass http://frontend;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
```

g. Configuración de Ngrok (túnel HTTPS)

Ngrok permite exponer tu aplicación local al exterior a través de una URL pública segura. En este proyecto, el túnel apunta al gateway Nginx que unifica frontend y backend.

Instalación:

- **Descargar el archivo comprimido:**
  wget https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-linux-amd64.tgz

- **Descomprimirlo:**
  tar -xvzf ngrok-v3-stable-linux-amd64.tgz

- **Mover el ejecutable a una ruta del sistema:**
  sudo mv ngrok /usr/local/bin/ngrok

- **Comprobar la instalación:**
  ngrok version

Si muestra la versión, Ngrok ya está listo para usar.

Autenticación con token:
Para evitar limitaciones, añade tu token de autenticación de Ngrok:
ngrok config add-authtoken "TU_TOKEN_DE_NGROK"

Publicar la aplicación:
Como el gateway Nginx escucha en el puerto 80, crea un túnel hacia ese puerto:
ngrok http 80

Ngrok se queda ejecutándose y muestra una URL pública HTTPS. Esta URL permite acceder a toda la aplicación (frontend VUE SPA y API Laravel a través del gateway)

h. Conclusión

En producción, la aplicación funciona mediante un **gateway Nginx** que unifica el acceso al **frontend Vue** y al **backend Laravel**, mientras la base de datos **MySQL** mantiene la persistencia de datos. Además **Ngrok** permite un acceso externo seguro mediante HTTPS.

El frontend se sirve como **SPA**, con rutas gestionadas por Nginx, y el backend Laravel gestiona la **API optimizada para producción**, incluyendo migraciones y seeders. Los contenedores se comunican entre sí mediante la **red interna de Docker Compose**, asegurando un entorno aislado pero totalmente funcional.

Esta configuración garantiza que la aplicación pueda ser desplegada de manera **reproducible y segura**, funcionando correctamente tanto localmente como a través de un túnel externo seguro proporcionado por Ngrok.

3. REPARTO

Tarea / Fase	Responsable(s)
Primeros cambios	Eneko
Intento de solución	Danel + Iker
1ª Documentación	Danel + Iker
Solución final	Eneko
2ª Documentación	Danel + Iker