### **Instructivo para Desplegar el Frontend de PetFlow con Docker**

Hola equipo,

Esta guía explica cómo empaquetar la aplicación de frontend (React + Vite) en una imagen de Docker y ejecutarla como un contenedor. Esto nos permitirá desplegar la interfaz de usuario de forma aislada y consistente.

#### **Paso 1: Crear el archivo Dockerfile**

En la raíz del proyecto del frontend, crea un archivo llamado Dockerfile (sin extensión) y pega el siguiente contenido. Este archivo contiene las instrucciones para construir nuestra aplicación.

**Archivo: Dockerfile**

codeDockerfile

# --------------------------------------------------------------------  
# Etapa 1: Construcción (Build Stage)  
# Aquí compilamos la aplicación de React para obtener archivos estáticos.  
# --------------------------------------------------------------------  
FROM node:20-alpine AS builder  
  
# Establecemos el directorio de trabajo dentro del contenedor  
WORKDIR /app  
  
# Declaramos un argumento que se pasará durante la construcción de la imagen.  
# Esto es crucial para la clave de la API de Gemini.  
ARG API\_KEY  
  
# Hacemos que el argumento esté disponible como una variable de entorno DENTRO de esta etapa.  
ENV API\_KEY=$API\_KEY  
  
# Copiamos el package.json para instalar las dependencias  
COPY package.json ./  
  
# Instalamos las dependencias  
RUN npm install  
  
# Copiamos todo el código fuente del proyecto  
COPY . .  
  
# Ejecutamos el script de build de Vite.  
# Vite reemplazará `process.env.API\_KEY` con el valor real que pasamos.  
RUN npm run build  
  
# --------------------------------------------------------------------  
# Etapa 2: Producción (Production Stage)  
# Aquí servimos los archivos estáticos generados con un servidor web ligero.  
# --------------------------------------------------------------------  
FROM nginx:alpine  
  
# Copiamos los archivos generados en la etapa anterior a la carpeta web de Nginx  
COPY --from=builder /app/dist /usr/share/nginx/html  
  
# Exponemos el puerto 80, que es donde Nginx escucha por defecto  
EXPOSE 80  
  
# El comando por defecto de la imagen de Nginx se encarga de iniciar el servidor.

#### **Paso 2: Crear el archivo .dockerignore**

Para que la imagen de Docker sea ligera y segura, necesitamos evitar que se copien archivos innecesarios. Crea un archivo llamado .dockerignore en la misma carpeta raíz y pega esto:

**Archivo: .dockerignore**

codeCode

# Dependencias de Node  
node\_modules  
  
# Carpeta de salida del build  
dist  
.vite  
  
# Logs y archivos de depuración  
npm-debug.log\*  
yarn-debug.log\*  
yarn-error.log\*  
  
# Archivos de entorno locales (¡nunca deben ir en la imagen!)  
.env  
.env.local  
.env.\*.local  
  
# Archivos del sistema operativo  
.DS\_Store  
Thumbs.db

#### **Paso 3: Construir y Ejecutar el Contenedor**

Abre una terminal en la raíz del proyecto y ejecuta los siguientes comandos.

**1. Construir la Imagen de Docker:**

Este comando lee el Dockerfile y construye la imagen. **Es muy importante** que reemplaces "tu\_clave\_de\_api\_aqui" con la API\_KEY real de Google Gemini.

codeBash

docker build --build-arg API\_KEY="tu\_clave\_de\_api\_aqui" -t petflow-frontend .

* --build-arg API\_KEY="...": Pasa la clave de la API al proceso de construcción.
* -t petflow-frontend: Le da un nombre y una etiqueta a nuestra imagen (petflow-frontend:latest).
* .: Indica que el contexto de construcción es el directorio actual.

**2. Ejecutar el Contenedor desde la Imagen:**

Una vez construida la imagen, podemos crear y ejecutar un contenedor a partir de ella.

codeBash

docker run -d -p 8080:80 --name petflow-app petflow-frontend

* -d: Ejecuta el contenedor en segundo plano (detached mode).
* -p 8080:80: Mapea el puerto 8080 de tu máquina al puerto 80 del contenedor. Esto significa que podrás acceder a la aplicación desde tu navegador en <http://localhost:8080>.
* --name petflow-app: Le da un nombre fácil de recordar a nuestro contenedor.
* petflow-frontend: El nombre de la imagen que queremos usar.

¡Y eso es todo! Con estos pasos, la aplicación frontend estará funcionando dentro de un contenedor de Docker, lista para ser desplegada en cualquier lugar que soporte Docker.

### **Instructivo para Ejecutar PetFlow con Docker Compose**

Aquí están las instrucciones paso a paso para levantar la aplicación de frontend de PetFlow usando el archivo docker-compose.yml. Este método es la forma recomendada para ejecutar el proyecto, ya que automatiza todo el proceso.

#### **Requisitos Previos**

1. **Tener Docker instalado:** Asegúrate de que Docker Desktop (o Docker Engine en Linux) esté instalado y en ejecución en tu máquina.

#### **Paso 1: Configurar tu Clave de API**

Antes de iniciar, debes proporcionar tu clave secreta de la API de Google Gemini.

1. Busca el archivo llamado .env en la raíz del proyecto.
2. Ábrelo y verás esta línea:

codeCode

API\_KEY=tu\_clave\_de\_api\_aqui

1. **Reemplaza** "tu\_clave\_de\_api\_aqui" con tu clave de API real. Guarda el archivo.

*Este paso es crucial. Sin una clave válida, el Planificador de Comidas con IA no funcionará.*

#### **Paso 2: Construir y Levantar la Aplicación**

Ahora, con un solo comando, podemos construir la imagen de Docker y ejecutar el contenedor.

1. Abre una terminal o línea de comandos.
2. Navega hasta la carpeta raíz del proyecto (donde se encuentra el archivo docker-compose.yml).
3. Ejecuta el siguiente comando:

codeBash

docker-compose up --build -d

* **¿Qué hace este comando?**
  + docker-compose up: Lee el docker-compose.yml y crea/inicia los servicios definidos (en este caso, nuestro frontend).
  + --build: Le dice a Docker que construya la imagen desde cero usando el Dockerfile la primera vez (o si has hecho cambios en el código).
  + -d: Ejecuta el contenedor en "modo detached" (en segundo plano), para que no ocupe tu terminal.

El proceso puede tardar un par de minutos la primera vez mientras descarga las dependencias y construye la imagen.

#### **Paso 3: Acceder a la Aplicación**

Una vez que el comando anterior termine, ¡la aplicación estará en vivo!

1. Abre tu navegador web.
2. Ve a la siguiente dirección: [**http://localhost:8080**](http://localhost:8080)

Deberías ver la pantalla de login de PetFlow.

#### **Paso 4: Detener la Aplicación**

Cuando hayas terminado de trabajar y quieras detener el contenedor, sigue estos pasos:

1. Vuelve a la misma terminal y al mismo directorio del proyecto.
2. Ejecuta el comando:

codeBash

docker-compose down

* **¿Qué hace este comando?**
  + Detiene y elimina de forma segura el contenedor que estaba en ejecución, liberando el puerto 8080.

Este es el flujo de trabajo completo. Usar docker-compose asegura que todos en el equipo ejecuten la aplicación exactamente de la misma manera.