Creación de la aplicación para gestión financiera en contenedores en la plataforma Docker

Para el desarrollo de la actividad-1, se toma una aplicación en PHP de 3 capas, (base de datos, back-end y front-end).

A continuación de describe todos los pasos realizados en la actividad.

1 – creación de la estructura de carpetas que contiene los archivos de cada capa.

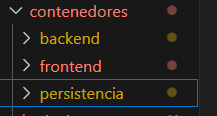


Fig. 1 Directorios de contenedores

2 – Creación de red de Docker. para la comunicación entre los contenedores que contiene cada capa.



Texto

El contenido generado por IA puede ser incorrecto.

Fig. 2 Detalle de la red Docker creada.

3 - Construcción de la imagen de la capa de base de datos

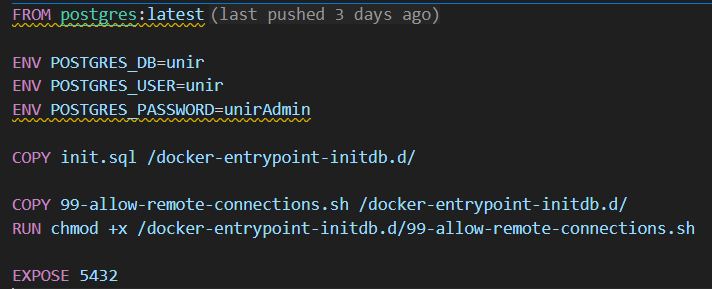


Fig. 3 Archivo Dockerfile con la imagen actualizada de postgres.

Descripción del Dockerfile:

**FROM postgres:latest**; Se descarga la imagen oficial de PostgreSQL más actualizada como base.

**ENV POSTGRES\_DB=unir, ENV POSTGRES\_USER=unir, ENV POSTGRES\_PASSWORD=unirAdmin**; Variables de entorno para la configuración inicial de la base de datos.

**COPY init.sql /docker-entrypoint-initdb.d/**; se copia el Script SQL init.sql, que se encarga de crear una tabla e insertar registros de pruebas, en el directorio de inicialización de base de datos del contenedor, ejecutándose de forma automática la primera ves que se inicie el contenedor.

**COPY 99-allow-remote-connections.sh /docker-entrypoint-initdb.d/**; Se copia Script en bash, que se configura los permisos de acceso a la base de datos desde cualquier IP.

**RUN chmod +x /docker-entrypoint-initdb.d/99-allow-remote-connections.sh**; Se da permisos de ejecución en el contenedor, después de que inicie la base de datos

**EXPOSE 5432**; Se expone el puerto 5432, el cual es por defecto de PostgreSQL en el contenedor creado.

Archivos complementarios de la base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 4 Archivo Script SQL, que crea tabla y datos de pruebas.

Una captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Fig. 5 Archivo Script bash, que cambia la configuración de PostgreSQL, para permitir conexión de cualquier IP.

Ejecución del contenedor.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 6 Creación de imagen propia: postgres-db.

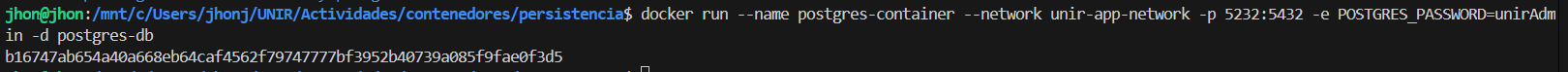


Fig. 7 Creación de contenedor: postgres-container.

Despliegue del contenedor.

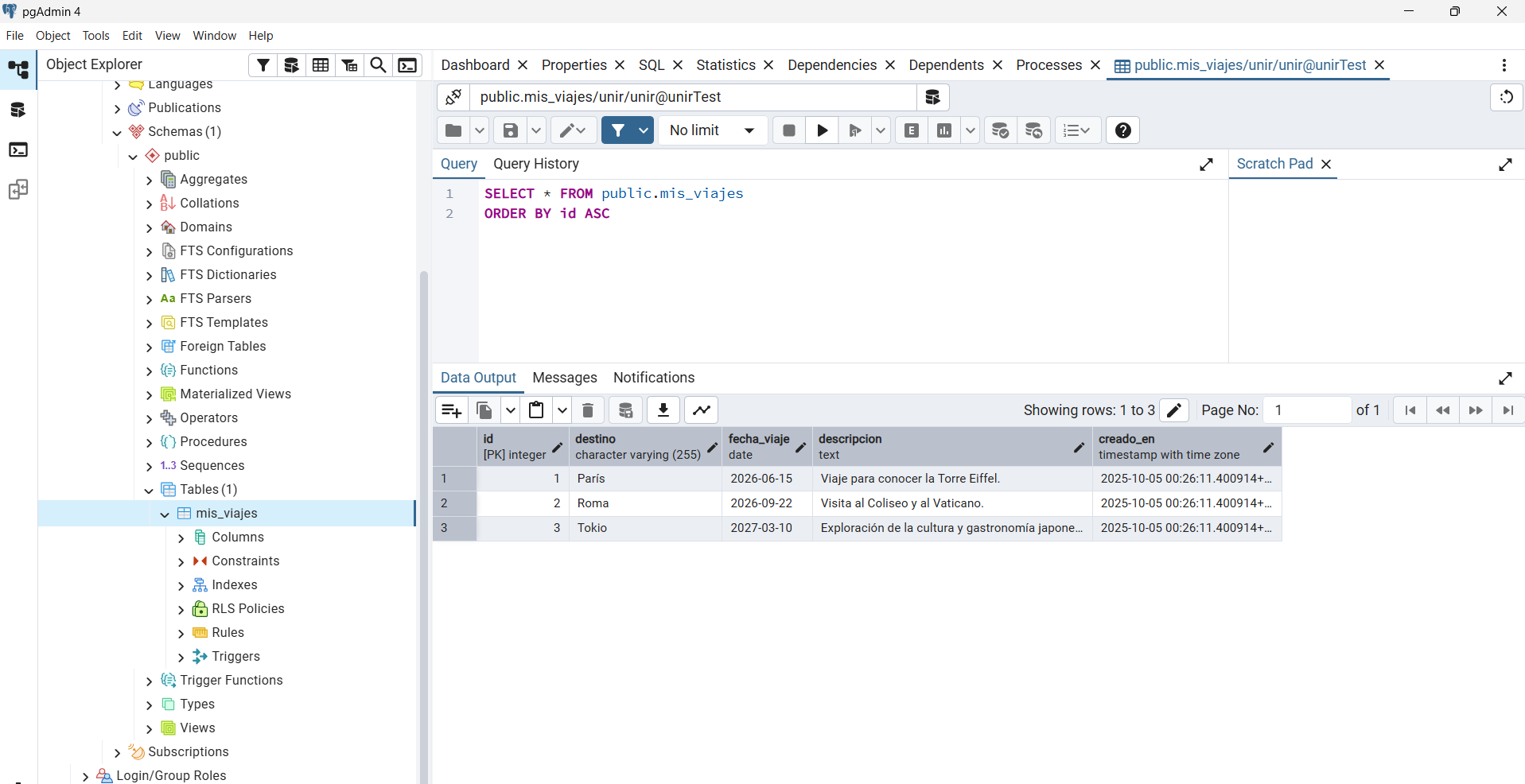


Fig. 8 Ingreso a la base de datos por medio de pgAdmin.

4 - Construcción de la imagen del API (back-end).

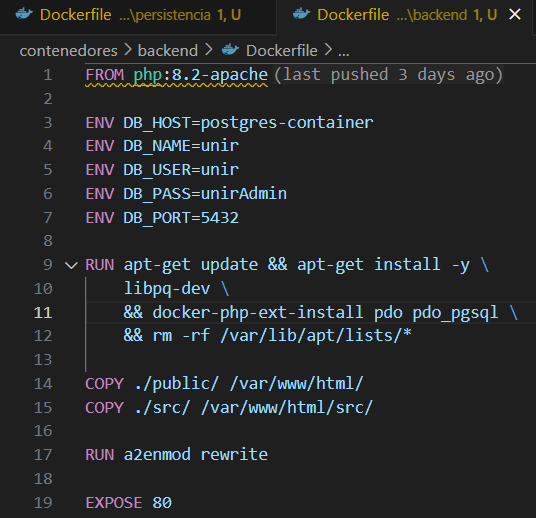


Fig. 9 Archivo Dockerfile con la imagen actualizada de php 8.2 con apache y API.

Descripción del Dockerfile:

**FROM php:8.2-apache**; Se descarga la imagen oficial de PHP con el servidor Apache

**ENV DB\_HOST=postgres-container, ENV DB\_NAME=unir, ENV DB\_USER=unir, ENV, DB\_PASS=unirAdmin, ENV DB\_PORT=5432**; Variables de entorno para la conexión de la base de datos.

**RUN apt-get update && apt-get install -y \**

**libpq-dev \**

**&& docker-php-ext-install pdo pdo\_pgsql \**

**&& rm -rf /var/lib/apt/lists/\*;** Instala las dependencias del sistema para la extensión de PostgreSQL, luego las extensiones de PHP y finalmente limpiar la caché de apt para reducir el tamaño de la imagen.

**COPY ./public/ /var/www/html/**; Se copia los archivos de la aplicación al directorio web del servidor en el contenedor

**COPY ./src/ /var/www/html/src**; Se copia los archivos do configuración de la aplicación al directorio src del servidor en el contenedor

**RUN a2enmod rewrite**; Se habilita el módulo de reescritura de Apache para URLs amigables

**EXPOSE 80**; Se expone el puerto 80, el cual es por defecto del servidor apache en el contenedor creado.

Archivos complementarios del back-end.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 10 Clase Database.php con la configuración para conectarse a la base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 11 index.php, clase php que responde json la consulta solicitada.

Ejecución del contenedor.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 12 Creación de imagen propia: mi-api.



Fig. 13 Creación de contenedor: mi-api-container.

Despliegue del contenedor.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Fig. 14 consulta API - localhost:8080.

5 - Construcción de la imagen del Vista (front-end).

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Fig. 15 Archivo Dockerfile con la imagen oficial de Nginx.

Descripción del Dockerfile:

**FROM nginx:alpine**; Se descarga como base imagen ligera de Nginx.

**COPY index.html /usr/share/nginx/html/**; Se copia el archivo HTML del frontend al directorio web raíz de Nginx.

**EXPOSE 88**; Se expone el puerto 88, el cual es por defecto del servidor Nginx en el contenedor creado.

Archivos complementarios del front-end.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 16 index.html, témplate creado del lado del front-end para visualizar datos.

Ejecución del contenedor.

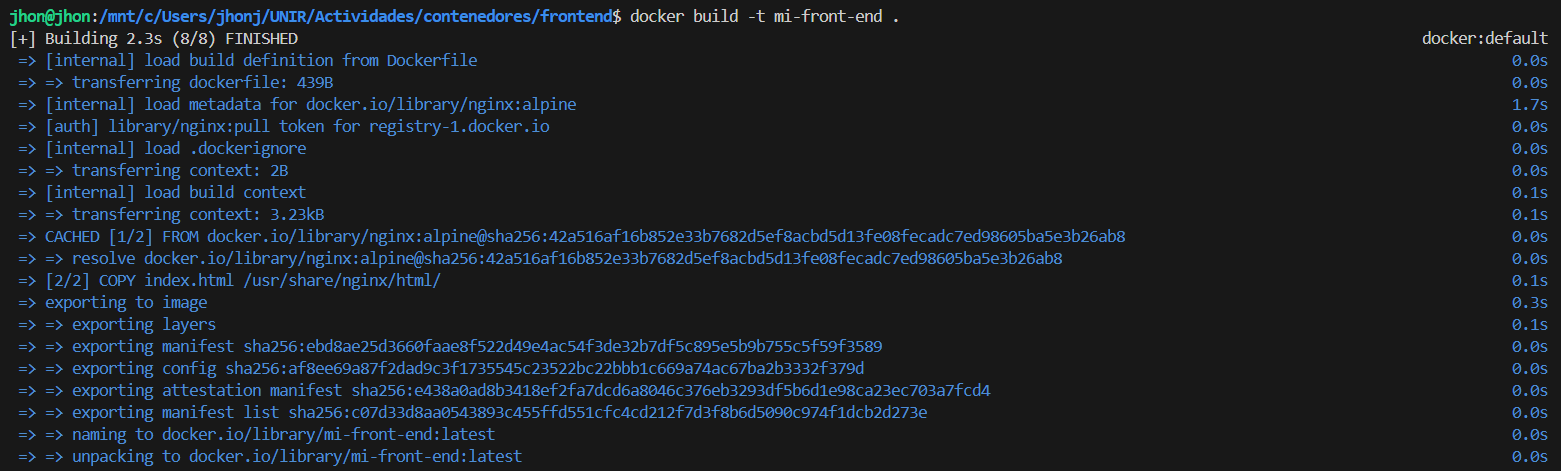


Fig. 17 Creación de imagen propia: mi-front-end.



Fig. 18 Creación de contenedor: front-end-container.

Despliegue del contenedor.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Fig. 19 Vista data – localhost:88.

6 - Subir imágenes a Docker Hub.

Se procede a crear una cuenta en Docker Hub, en donde se van a subir las imágenes de la actividad, sobre los repositorios postgres-db, mi-api y mi-front-end.

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 20 Creación y subida de la imagen en el repositorio jjcarmu/postgres-db

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Fig. 21 Repositorio postgres-db ([jjcarmu/postgres-db - Docker Image | Docker Hub](https://hub.docker.com/r/jjcarmu/postgres-db)).

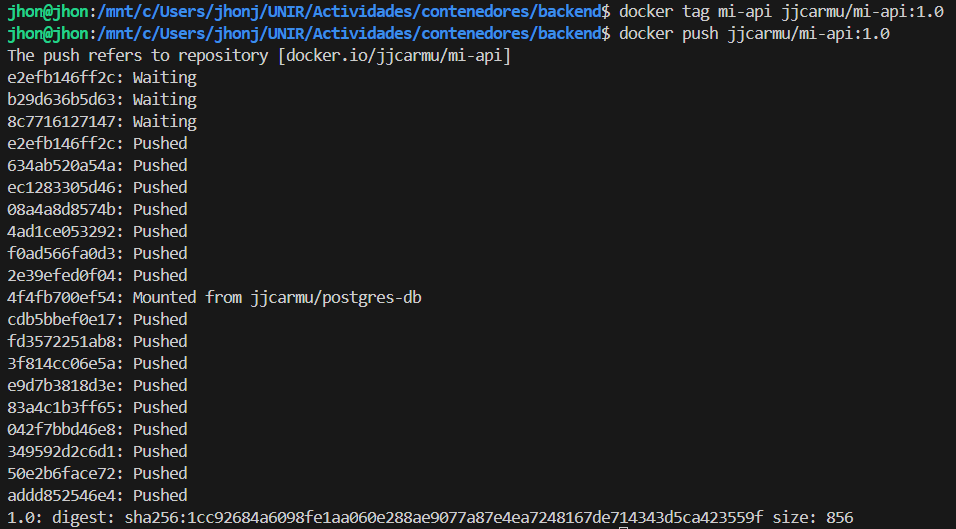


Fig. 22 Creación y subida de la imagen en el repositorio jjcarmu/mi-api

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Fig. 23 Repositorio mi-api ([jjcarmu/mi-api - Docker Image | Docker Hub](https://hub.docker.com/r/jjcarmu/mi-api)).

Texto

El contenido generado por IA puede ser incorrecto.

Fig. 24 Creación y subida de la imagen en el repositorio jjcarmu/mi-front-end

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Fig. 25 Repositorio mi-front-end ([jjcarmu/mi-front-end - Docker Image | Docker Hub](https://hub.docker.com/r/jjcarmu/mi-front-end)).

**Conclusiones:** Trabajar con contenedores ayuda agilizar el despliegue y la creación rápida de ambientes, ya se un entorno de desarrollo, pruebas o producción.

Existen una gran variedad de herramientas que ayudan a manejar los contenedores, como supervisarlos, entre estos esta Docker.desktop o los plugin de los diversos IDES de desarrollo, que facilitan la manipulación de los contenedores.

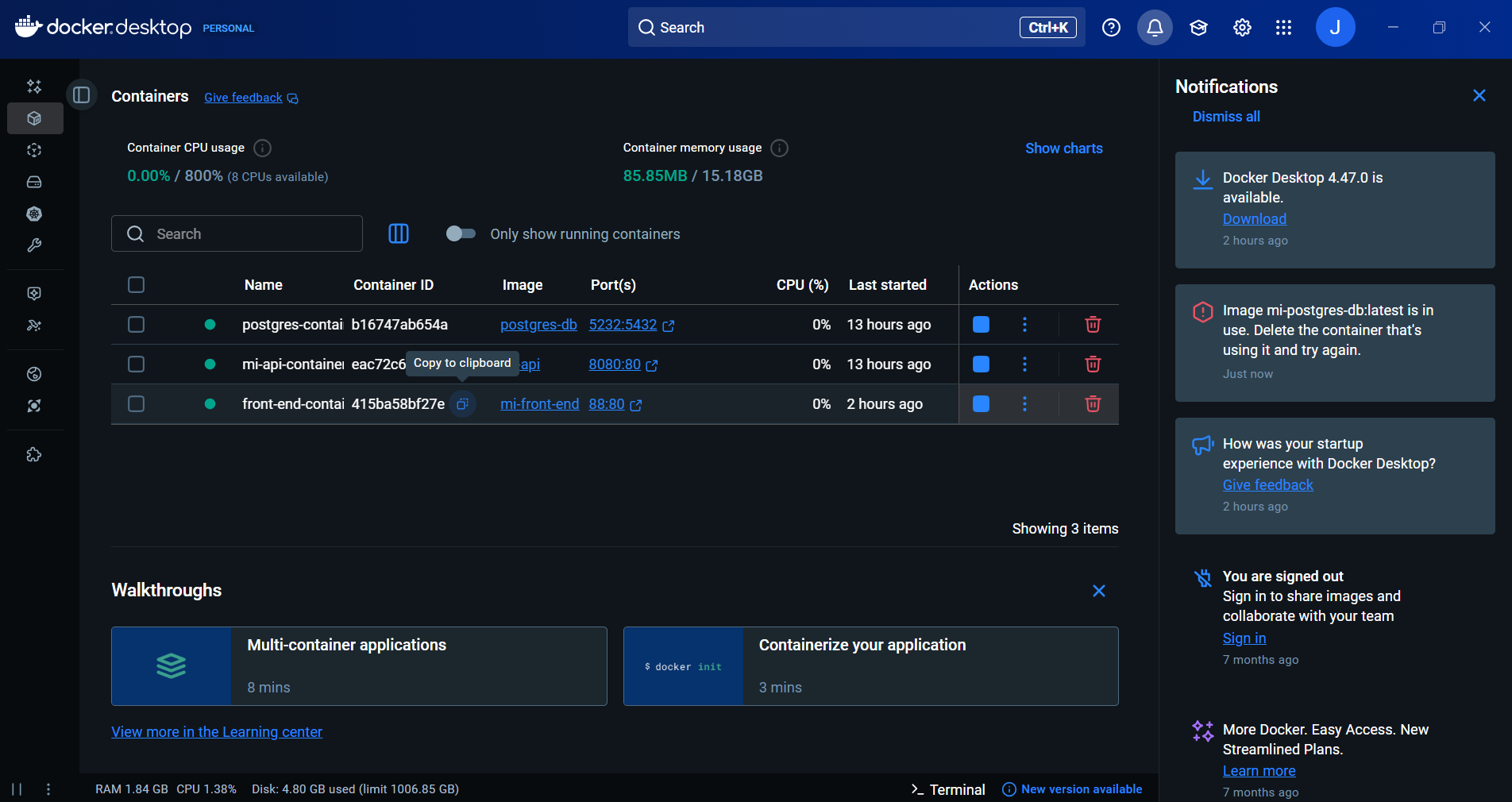


Fig. 26 Visualización de los contenedores desde Docker.desktop.