Docs

Imagen Docker (API)

Imagen Docker (API)

Anonymous · 27/08/2026

Table of Contents

Imagen Docker (API)

Imagen Docker (API)

Anonymous · 27/08/2026

Imagen Docker (API)

Last updated on 27/08/2026

Introducción

LibreDTE Core vía API en un contenedor

Anonymous · 27/08/2026

LibreDTE Core vía API en un contenedor

LibreDTE Core también se puede correr como una imagen Docker que expone únicamente los Servicios Web (API), sin el resto del sitio. Construida en el repositorio `libredte-lib-core-api`.

Importante

A diferencia de usar la Biblioteca PHP, el Dispatcher o el Bridge Python directamente, consumir esta imagen por HTTP no combina tu código con el de LibreDTE dentro de un mismo proceso: tu aplicación cliente no queda sujeta a los términos de la licencia AGPL por el solo hecho de llamarla. Solo el software de LibreDTE que corre dentro del contenedor debe mantenerse libre.

Inicio rápido

La forma más simple de probarlo, con la imagen ya publicada:

```
docker run -d --rm \
    --name libredte-lib-core-api \
    -p 8080:80 \
    -e APP_URL=http://localhost:8080 \
    ghcr.io/libredte/libredte-lib-core-api:latest
```

Nota

Revisa Uso de la imagen Docker para otras formas de correrlo (`docker compose`, cambiar el puerto, dejarlo como servicio persistente) y Servicios Web (API) para lo que expone una vez corriendo.

Uso de la imagen Docker

Uso completo de la imagen Docker

Anonymous · 27/08/2026

Uso de la imagen Docker

El contenedor siempre escucha internamente en el puerto **80** (fijo, no se configura). El puerto externo lo eliges tú al mapearlo con `-p`, sin ninguna relación con el interno.

`APP_URL` es aparte: es la URL que la propia API usa para armar enlaces absolutos en sus respuestas (HATEOAS, spec de OpenAPI, etc.), y debería apuntar al puerto externo que elegiste para que esos enlaces sean correctos. Si no la configuras, o no coincide con el puerto externo real, la API igual funciona — solo los enlaces que devuelve quedarán con la URL por defecto.

`docker run`, con la imagen ya publicada

Corriendo directo la imagen publicada en GHCR, en segundo plano:

```
docker run -d --rm \
    --name libredte-lib-core-api \
    -p 8080:80 \
    -e APP_URL=http://localhost:8080 \
    ghcr.io/libredte/libredte-lib-core-api:latest
```

`-d` lo deja corriendo en segundo plano (sin `-d` se queda en primer plano y `Ctrl+C` lo detiene). `--rm` hace que el contenedor se borre solo cuando lo detengas, en vez de quedar acumulado como "Exited". Para ver los logs o detenerlo:

```
docker logs -f libredte-lib-core-api
docker stop libredte-lib-core-api
```

Para usar otro puerto externo (ej. 9090), cambia ambos valores:

```
docker run -d --rm \
    --name libredte-lib-core-api \
    -p 9090:80 \
    -e APP_URL=http://localhost:9090 \
    ghcr.io/libredte/libredte-lib-core-api:latest
```

Los ejemplos anteriores usan `--rm`: cómodo para probar, pero **no** vuelve a levantarse solo si reinicias

el host (`--restart` y `--rm` son mutuamente excluyentes en Docker). Para un servicio persistente que sobreviva reinicios, saca `--rm` y usa `--restart unless-stopped`:

```
docker run -d --restart unless-stopped \
    --name libredte-lib-core-api \
    -p 8080:80 \
    -e APP_URL=http://localhost:8080 \
    ghcr.io/libredte/libredte-lib-core-api:latest
```

Nota

Esto asume que el propio Docker arranca al iniciar el host: en Docker Desktop necesitas tener activado “iniciar Docker Desktop al iniciar sesión”; en Linux con systemd, `dockerd` normalmente ya arranca solo.

`docker compose`

Clonando el repositorio, `docker compose` baja la imagen ya publicada (no la reconstruye) mientras no le pidas `--build`:

```
git clone git@github.com:LibreDTE/libredte-lib-core-api.git
cd libredte-lib-core-api
docker compose up -d
```

(sin `-d` también queda en primer plano; con `-d` corre en segundo plano y `docker compose logs -f` / `docker compose down` sirven para verlo o detenerlo).

`docker-compose.yml` expone dos variables de entorno independientes, cada una con su propio default (`8080` para ambas, pero no están ligadas entre sí):

- `PORT`: el puerto externo publicado en el host (el interno del contenedor siempre es 80).
- `APP_URL`: la URL que la API usa para armar sus enlaces.

Para usar otro puerto:

```
PORT=9090 APP_URL=http://localhost:9090 docker compose up -d
```

Construir la imagen localmente

En vez de usar la publicada en GHCR:

```
docker build -t libredte-lib-core-api .
```

Last updated on 27/08/2026

Arquitectura

Cómo está armada la imagen y desarrollo sin Docker

Anonymous · 27/08/2026

Arquitectura

La imagen se construye en 2 etapas:

1. Una etapa con `composer:2` que corre `composer install` (sin dev-dependencies) para generar `vendor/`.
2. La etapa final, sobre `dunglas/frankenphp`, que instala las extensiones de PHP que hacen falta (`gd`, `intl`, `soap`) y copia el código junto al `vendor/` generado en la etapa anterior.

FrankenPHP ya trae un Caddyfile por defecto que sirve el directorio `public/` (coincide con el de este proyecto) y escucha en la dirección que indique `SERVER_NAME`. No se necesita ningún Caddyfile propio: el puerto interno queda fijo en `SERVER_NAME=:80`.

Desarrollo sin Docker

Para modificar el código sin construir la imagen:

```
git clone git@github.com:LibreDTE/libredte-lib-core-api.git
cd libredte-lib-core-api
composer install
cp .env-dist .env
php -S localhost:8080 -t public
```

Last updated on 27/08/2026