Docs

Imagen Docker (Consola)

Imagen Docker (Consola)

Anonymous · 27/08/2026

Table of Contents

Imagen Docker (Consola)

Imagen Docker (Consola)

Anonymous · 27/08/2026

Imagen Docker (Consola)

Last updated on 27/08/2026

Introducción

LibreDTE Core vía consola en un contenedor

Anonymous · 27/08/2026

LibreDTE Core vía consola en un contenedor

LibreDTE Core también se puede correr como una imagen Docker que expone cada operación de `libredte/libredte-lib-core` como su propio comando de Symfony Console (`bin/console`), invocable desde cualquier lenguaje vía `docker exec` — sin instalar PHP en tu máquina. Construida en el repositorio `libredte-lib-core-console`.

Importante

A diferencia de usar la Biblioteca PHP, el Dispatcher o el Bridge Python directamente, consumir esta imagen por `docker exec` 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 invocarla. Solo el software de LibreDTE que corre dentro del contenedor debe mantenerse libre.

Inicio rápido

La imagen no sirve nada por HTTP ni tiene ningún puerto que mapear: el proceso principal solo mantiene el contenedor vivo para poder ejecutarle comandos puntuales con `docker exec`.

```
docker run -d --restart unless-stopped \
    --name libredte-lib-core-console \
    ghcr.io/libredte/libredte-lib-core-console:latest

docker exec libredte-lib-core-console bin/console list
```

Nota

Revisa Uso de la imagen Docker para invocar una operación real, `docker compose`, y la forma de usarla una sola vez sin dejar nada corriendo.

Uso de la imagen Docker

Uso completo de la imagen Docker

Anonymous · 27/08/2026

Uso de la imagen Docker

La imagen no expone ningún puerto: no hay nada que mapear con `-p`. El proceso principal del contenedor solo lo mantiene vivo para poder ejecutarle comandos puntuales con `docker exec`.

`docker run`, con la imagen ya publicada

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

```
docker run -d --restart unless-stopped \
    --name libredte-lib-core-console \
    ghcr.io/libredte/libredte-lib-core-console:latest
```

```
docker exec libredte-lib-core-console bin/console list
docker exec libredte-lib-core-console bin/console help
billing:identifier:caf_loader:load

echo '{"parameters": {"xml": "<caf>...</caf>"}}' \
    | docker exec -i libredte-lib-core-console bin/console
billing:identifier:caf_loader:load
```

`-i` en el `exec` es necesario solo cuando la entrada va por STDIN (como en el `echo` de arriba); para un archivo como argumento no hace falta:

```
docker cp archivo.json libredte-lib-core-console:/tmp/archivo.json
docker exec libredte-lib-core-console bin/console billing:identifier:caf_loader:load
/tmp/archivo.json
```

Para ver los logs o detenerlo:

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

Nota

Revisa Dispatcher y el README de `libredte-lib-core-console` para el detalle de la entrada

(JSON/YAML/XML bajo `"parameters"`), la respuesta (`{"meta": ..., "data": ...}`) y los exit codes.

Uso único, sin dejar nada corriendo

Sobreescribiendo el `CMD` de la imagen se evita levantar un contenedor persistente cuando solo se necesita ejecutar un comando una vez:

```
echo '{"parameters": {"xml": "<caf>...</caf>"}}' \
    | docker run --rm -i ghcr.io/libredte/libredte-lib-core-console:latest \
        bin/console billing:identifier:caf_loader:load
```

`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-console.git
cd libredte-lib-core-console
docker compose up -d

docker compose exec libredte-lib-core-console bin/console list
docker compose down
```

Construir la imagen localmente

En vez de usar la publicada en GHCR:

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

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 `php:8.5-cli` (sin FrankenPHP: esta imagen no sirve nada por HTTP), que instala las extensiones de PHP que hacen falta y copia el código junto al `vendor/` generado en la etapa anterior.

El proceso principal del contenedor (`tail -f /dev/null`) no ejecuta nada por sí mismo — solo mantiene el contenedor vivo para que `docker exec` (o un `docker run` que sobreescriba ese `CMD`) pueda invocar `bin/console`.

Extensiones de PHP

`libredte/libredte-lib-core` y sus dependencias (`derafu/certificate`, `derafu/signature`, `mpdf/mpdf`, `tecnickcom/tc-lib-barcode`, etc.) solo necesitan `gd`/`intl`/`soap` además de lo que `php:8.5-cli` ya trae compilado (`curl`/`mbstring`/`openssl`/`xml`/entre otras). La imagen igual instala el mismo set más amplio que usa `docker-php8.5-caddy-server` (la imagen de referencia con la que se desarrollan/prueban estas mismas librerías): `bcmath`, `exif`, `gd`, `intl`, `pdo_mysql`, `pdo_pgsql`, `soap`, `sockets`, `zip` — como margen de seguridad ante una futura dependencia, sin arrastrar nada de lo específico a esa imagen (Caddy, SSH, Supervisor, Node, Python, Redis, Xdebug), que no aplica acá.

Desarrollo sin Docker

Para modificar el código sin construir la imagen:

```
git clone git@github.com:LibreDTE/libredte-lib-core-console.git
cd libredte-lib-core-console
composer install
bin/console list
```

Last updated on 27/08/2026