Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

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.

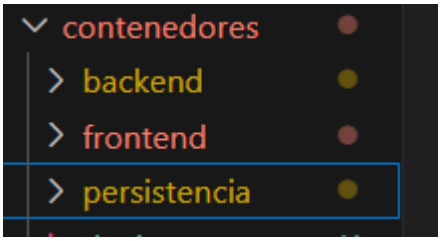

Fig. 1 Directorios de contenedores

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

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker network create unir-app-network
ebce3f7c61429e955c3399ff3258ffb284e04e432865f31f32c3af11c21d0a79
```

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker network inspect unir-app-network
[
    {
        "Name": "unir-app-network",
        "Id": "ebce3f7c61429e955c3399ff3258ffb284e04e432865f31f32c3af11c21d0a79",
        "Created": "2025-10-05T04:45:30.099298964Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.enable_ipv4": "true",
            "com.docker.network.enable_ipv6": "false"
        },
        "Labels": {}
    }
```

Fig. 2 Detalle de la red Docker creada.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

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

```
FROM postgres:latest (last pushed 3 days ago)

ENV POSTGRES_DB=unir
ENV POSTGRES_USER=unir
ENV POSTGRES_PASSWORD=unirAdmin

COPY init.sql /docker-entrypoint-initdb.d/

COPY 99-allow-remote-connections.sh /docker-entrypoint-initdb.d/
RUN chmod +x /docker-entrypoint-initdb.d/99-allow-remote-connections.sh

EXPOSE 5432
```

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.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

Archivos complementarios de la base de datos.

contenedores > persistencia > init.sql

```sql
-- Creación de la tabla mis_viajes
CREATE TABLE mis_viajes (
    id SERIAL PRIMARY KEY,
    destino VARCHAR(255) NOT NULL,
    fecha_viaje DATE,
    descripcion TEXT,
    creado_en TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

-- Insertar algunos datos de ejemplo
INSERT INTO mis_viajes (destino, fecha_viaje, descripcion) VALUES
('París', '2026-06-15', 'Viaje para conocer la Torre Eiffel.'),
('Roma', '2026-09-22', 'Visita al Coliseo y al Vaticano.'),
('Tokio', '2027-03-10', 'Exploración de la cultura y gastronomía japonesa.');
```

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

contenedores > persistencia > $ 99-allow-remote-connections.sh

```bash
#!/bin/bash
set -e

echo "listen_addresses = '*'" >> "$PGDATA/postgresql.conf"
echo "host all all all md5" >> "$PGDATA/pg_hba.conf"
```

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

Ejecución del contenedor.

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/persistencia$ docker build -t postgres-db .
[+] Building 2.8s (10/10) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 785B                                                                          0.0s
 => [internal] load metadata for docker.io/library/postgres:latest                                            1.4s
 => [auth] library/postgres:pull token for registry-1.docker.io                                               0.0s
 => [internal] load .dockerignore                                                                             0.1s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load build context                                                                             0.1s
 => => transferring context: 80B                                                                              0.0s
 => [1/4] FROM docker.io/library/postgres:latest@sha256:073e7c8b84e2197f94c8083634640ab37105effe1bc853ca4d5fbece3219b0e8  0.1s
 => => resolve docker.io/library/postgres:latest@sha256:073e7c8b84e2197f94c8083634640ab37105effe1bc853ca4d5fbece3219b0e8  0.1s
 => CACHED [2/4] COPY init.sql /docker-entrypoint-initdb.d/                                                   0.0s
 => CACHED [3/4] COPY 99-allow-remote-connections.sh /docker-entrypoint-initdb.d/                             0.0s
 => [4/4] RUN chmod +x /docker-entrypoint-initdb.d/99-allow-remote-connections.sh                             0.7s
 => exporting to image                                                                                        0.3s
 => => exporting layers                                                                                       0.1s
 => => exporting manifest sha256:dcb74d9927e9869baf215fa01a8e173abd653e2c0660d3ea8f7dc87cd8830660             0.0s
 => => exporting config sha256:d7ceacdf4be02a703cfab1367b9d66d6f91e8861b34b61d0dc4acd1123e5c10a               0.0s
 => => exporting attestation manifest sha256:65e2c658efac3656d8eb404436e163242b943e9518b04161a6dc28a9bc7a200b  0.0s
 => => exporting manifest list sha256:7641e78678947fb47dcf778a13102b88d1cf33602d88e79f067ce824326f1201        0.0s
 => => naming to docker.io/library/postgres-db:latest                                                         0.0s
 => => unpacking to docker.io/library/postgres-db:latest                                                      0.1s
```

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

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/persistencia$ docker run --name postgres-container --network unir-app-network -p 5232:5432 -e POSTGRES_PASSWORD=unirAdmin -d postgres-db
b16747ab654a40a668eb64caf4562f79747777bf3952b40739a085f9fae0f3d5
```

Fig. 7 Creación de contenedor: postgres-container.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

Despliegue del contenedor.

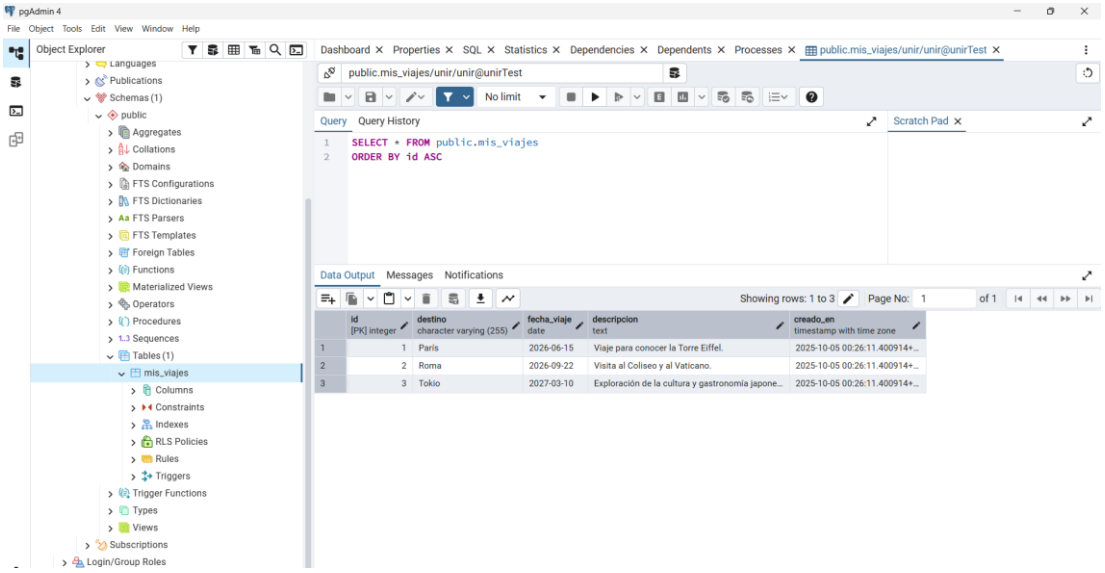

Fig. 8 Ingreso a la base de datos por medio de pgAdmin.

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

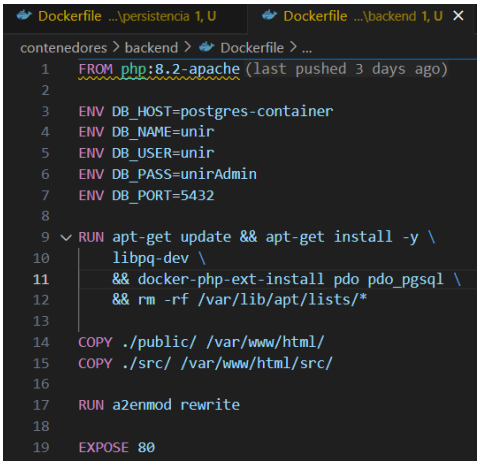

Fig. 9 Archivo Dockerfile con la imagen actualizada de php 8.2 con apache y API.

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.

contenedores > backend > src > Database.php > ...

```php
<?php
1 reference | 0 implementations
class Database {
    2 references
    private $host;
    2 references
    private $db_name;
    2 references
    private $username;
    2 references
    private $password;
    2 references
    private $port;
    4 references
    public $conn;

    1 reference | 0 overrides
    public function __construct() {
        $this->host = getenv(name: 'DB_HOST');
        $this->db_name = getenv(name: 'DB_NAME');
        $this->username = getenv(name: 'DB_USER');
        $this->password = getenv(name: 'DB_PASS');
        $this->port = getenv(name: 'DB_PORT') ?: '5432';
    }

    1 reference | 0 overrides
    public function getConnection(): PDO {
        $this->conn = null;
        $dsn = "pgsql:host=" . $this->host . ";port=" . $this->port . ";dbname=" . $this->db_name;

        try {
            $this->conn = new PDO(dsn: $dsn, username: $this->username, password: $this->password);
            $this->conn->setAttribute(attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $exception) {
            echo "Connection error: " . $exception->getMessage();
        }

        return $this->conn;
```

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

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

contenedores > backend > public > index.php

```php
<?php
header(header: "Access-Control-Allow-Origin: *");
header(header: "Content-Type: application/json; charset=UTF-8");

include_once '/var/www/html/src/Database.php';

$database = new Database();
$db = $database->getConnection();

if ($db) {
    try {
        $query = "SELECT id, , descripcion FROM mis_viajes ORDER BY id ASC";
        $stmt = $db->prepare(query: $query);
        $stmt->execute();

        $num = $stmt->rowCount();

        if ($num > 0) {
            $viajes_arr = array();
            $viajes_arr["records"] = array();

            while ($row = $stmt->fetch(mode: PDO::FETCH_ASSOC)) {
                extract(array: &$row);
                $viaje_item = array(
                    "id" => $id,
                    "destino" => $destino,
                    "fecha_viaje" => $fecha_viaje,
                    "descripcion" => $descripcion
                );
                array_push(array: &$viajes_arr["records"], values: $viaje_item);
            }

            http_response_code(response_code: 200);
            echo json_encode(value: $viajes_arr);
        } else {
            http_response_code(response_code: 404);
            echo json_encode(value: array("message" => "No se encontraron viajes."));
```

Double-click to insert

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

Ejecución del contenedor.

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker build -t mi-api .
[+] Building 22.6s (11/11) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                             0.0s
 => => transferring dockerfile: 1.32kB                                                                                           0.0s
 => [internal] load metadata for docker.io/library/php:8.2-apache                                                                1.5s
 => [auth] library/php:pull token for registry-1.docker.io                                                                       0.0s
 => [internal] load .dockerignore                                                                                                0.1s
 => => transferring context: 2B                                                                                                  0.0s
 => CACHED [1/5] FROM docker.io/library/php:8.2-apache@sha256:b3876890595b471c1eeebe0b073a81070f18100045c92761cb926eb80aca839c   0.0s
 => => resolve docker.io/library/php:8.2-apache@sha256:b3876890595b471c1eeebe0b073a81070f18100045c92761cb926eb80aca839c           0.0s
 => [internal] load build context                                                                                                0.0s
 => => transferring context: 123B                                                                                                0.0s
 => [2/5] RUN apt-get update && apt-get install -y     libpq-dev     && docker-php-ext-install pdo pdo_pgsql     && rm -rf /var/lib/apt/lists/*   18.5s
 => [3/5] COPY ./public/ /var/www/html/                                                                                          0.1s
 => [4/5] COPY ./src/ /var/www/html/src/                                                                                         0.1s
 => [5/5] RUN a2enmod rewrite                                                                                                    0.5s
 => exporting to image                                                                                                           1.7s
 => => exporting layers                                                                                                          1.2s
 => => exporting manifest sha256:dc46a3d21144599bc21a0e7510624768ce1388223a86863980130d65775f7fb9                                0.0s
 => => exporting config sha256:a3e83b9fe0675530f97f05744c259f6a841859a5129af42c0df7ae495324b389                                  0.0s
 => => exporting attestation manifest sha256:4e845a22a45c7854cbe41c89544ff15b16bfd3dd0d40aefdda3793b0e2e91ca1                    0.0s
 => => exporting manifest list sha256:1cc92684a6098fe1aa060e288ae9077a87e4ea7248167de714343d5ca423559f                           0.0s
 => => naming to docker.io/library/mi-api:latest                                                                                 0.0s
 => => unpacking to docker.io/library/mi-api:latest                                                                              0.3s
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$
```

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

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker run --name mi-api-container --network unir-app-network -p 8080:80 -d mi-api
eac72c6628d71dc72fe58f1173b1caa475c5d8e7c3bba9f1e6e2bdcfd80e5dfa
```

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

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

Despliegue del contenedor.

Fig. 14 consulta API - localhost:8080.

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

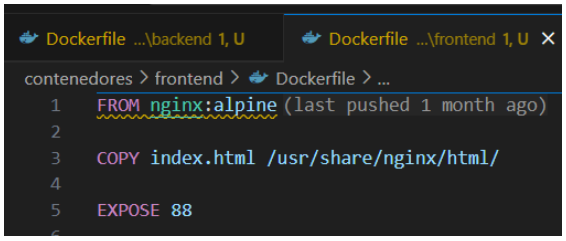

Fig. 15 Archivo Dockerfile con la imagen oficial de Nginx.

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.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

Archivos complementarios del front-end.

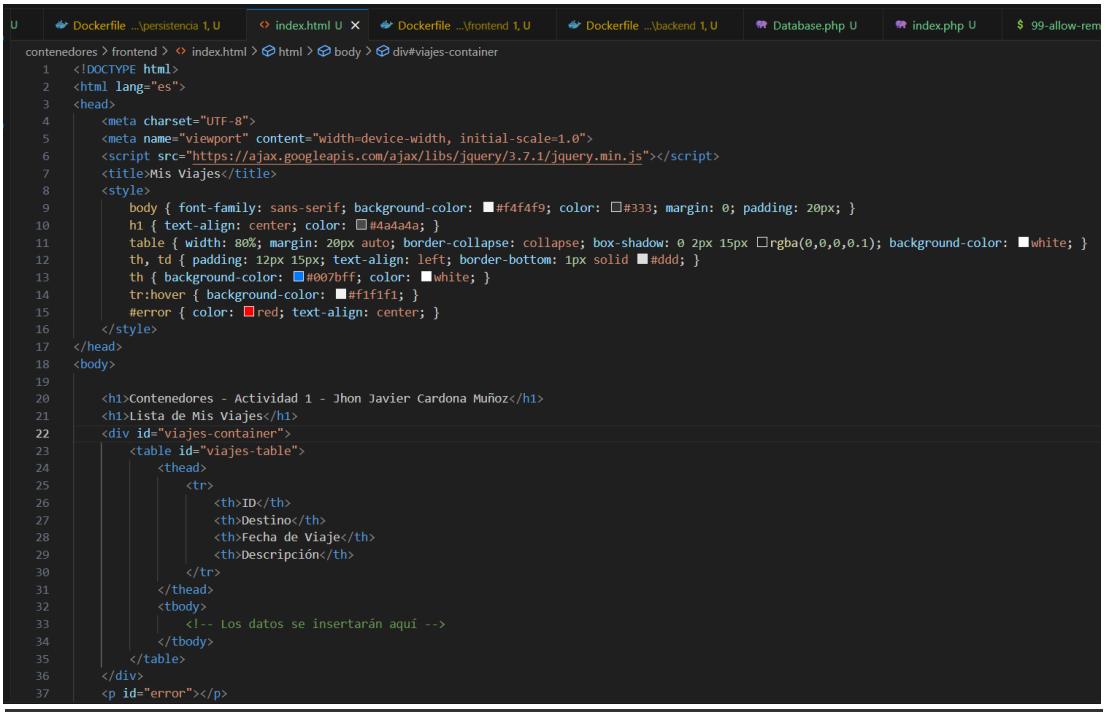

Fig. 16 index.html, témplate creado del lado del front-end para visualizar datos.

Ejecución del contenedor.

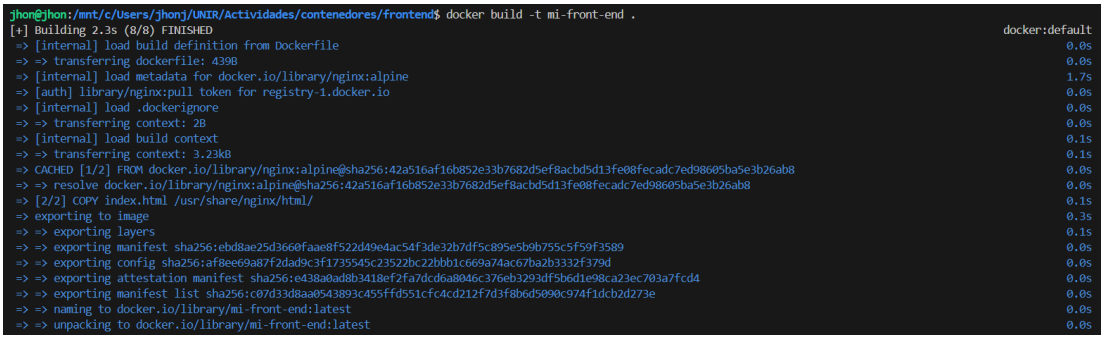

Fig. 17 Creación de imagen propia: mi-front-end.

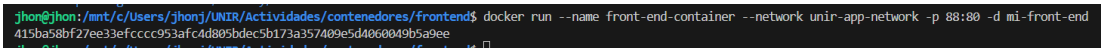

Fig. 18 Creación de contenedor: front-end-container.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

Despliegue del contenedor.

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.

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/persistencia$ docker tag postgres-db jjcarmu/postgres-db:1.0
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/persistencia$ docker push jjcarmu/postgres-db:1.0
The push refers to repository [docker.io/jjcarmu/postgres-db]
fe961117aa7f: Pushed
4f4fb700ef54: Pushed
203b16f56a7d: Pushed
a585c5f82f15: Pushed
1014e14b3351: Pushed
f5af7533693a: Pushed
2433c366ca00: Pushed
f0d70120d9e2: Pushed
0366f7384c6c: Pushed
f69a7c424b50: Pushed
8c7716127147: Pushed
9a68d6020eab: Pushed
edd90ab5059f: Pushed
f045741ea401: Pushed
dd6d7b9d8ba8: Pushed
eed0ac863490: Pushed
af60ce4418c9: Pushed
751039babae5: Pushed
1.0: digest: sha256:7641e78678947fb47dcf778a13102b88d1cf33602d88e79f067ce824326f1201 size: 856
```

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

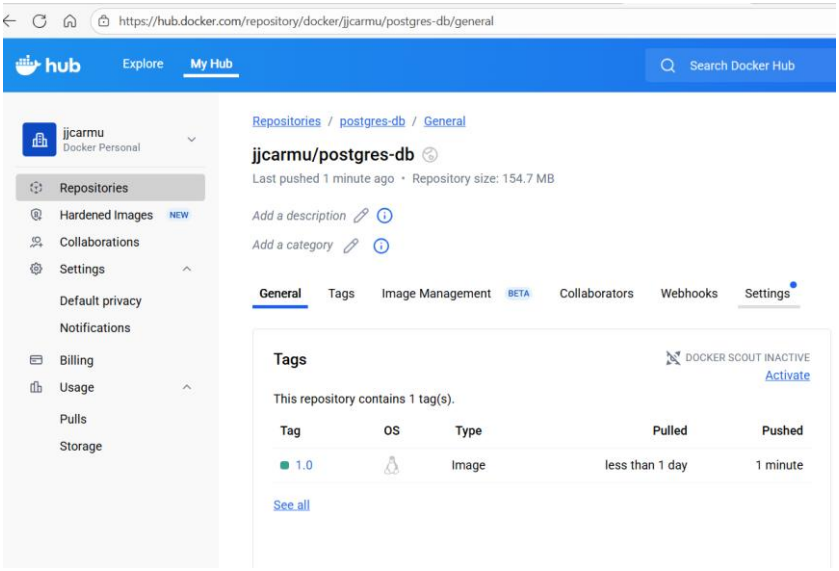

Fig. 21 Repositorio postgres-db (jjcarmu/postgres-db - Docker Image | Docker Hub).

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker tag mi-api jjcarmu/mi-api:1.0
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/backend$ docker push jjcarmu/mi-api:1.0
The push refers to repository [docker.io/jjcarmu/mi-api]
e2efb146ff2c: Waiting
b29d636b5d63: Waiting
8c7716127147: Waiting
e2efb146ff2c: Pushed
634ab520a54a: Pushed
ec1283305d46: Pushed
08a4a8d8574b: Pushed
4ad1ce053292: Pushed
f0ad566fa0d3: Pushed
2e39efed0f04: Pushed
4f4fb700ef54: Mounted from jjcarmu/postgres-db
cdb5bbef0e17: Pushed
fd3572251ab8: Pushed
3f814cc06e5a: Pushed
e9d7b3818d3e: Pushed
83a4c1b3ff65: Pushed
042f7bbd46e8: Pushed
349592d2c6d1: Pushed
50e2b6face72: Pushed
addd852546e4: Pushed
1.0: digest: sha256:1cc92684a6098fe1aa060e288ae9077a87e4ea7248167de714343d5ca423559f size: 856
```

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

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

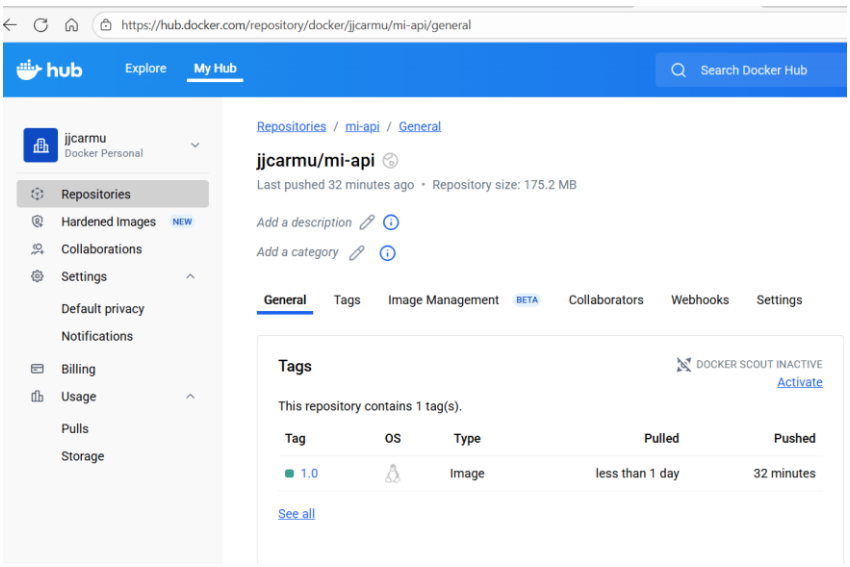

Fig. 23 Repositorio mi-api (jjcarmu/mi-api - Docker Image | Docker Hub).

```
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/frontend$ docker tag mi-front-end jjcarmu/mi-front-end:1.0
jhon@jhon:/mnt/c/Users/jhonj/UNIR/Actividades/contenedores/frontend$ docker push jjcarmu/mi-front-end:1.0
The push refers to repository [docker.io/jjcarmu/mi-front-end]
9adfbae99cb7: Pushed
b8c14131e456: Pushed
61fb610257c8: Pushed
c9ebe2ff2d2c: Pushed
9824c27679d3: Pushed
6bc572a340ec: Pushed
7a8a46741e18: Pushed
a992fbc61ecc: Pushed
403e3f251637: Pushed
cb1ff4086f82: Pushed
1.0: digest: sha256:c07d33d8aa0543893c455ffd551cfc4cd212f7d3f8b6d5090c974f1dcb2d273e size: 856
```

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

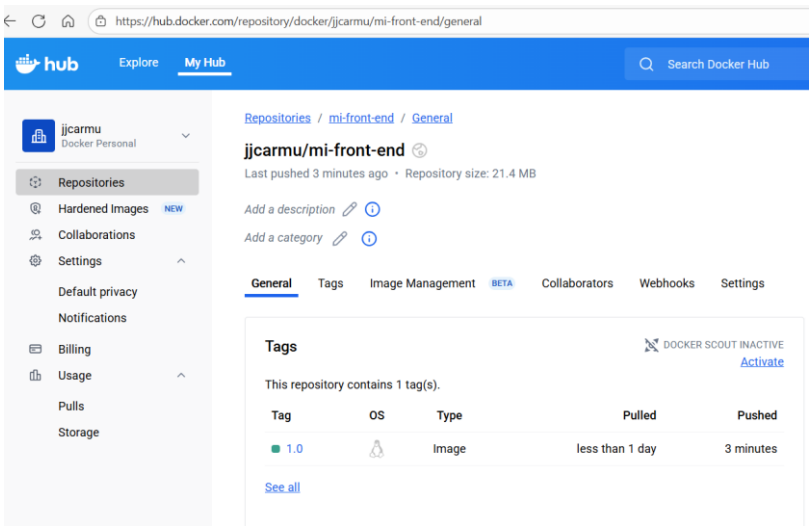

Fig. 25 Repositorio mi-front-end (jjcarmu/mi-front-end - Docker Image | Docker Hub).

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Cardona Muñoz	05/10/2025
	Nombre: Jhon Javier	

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.

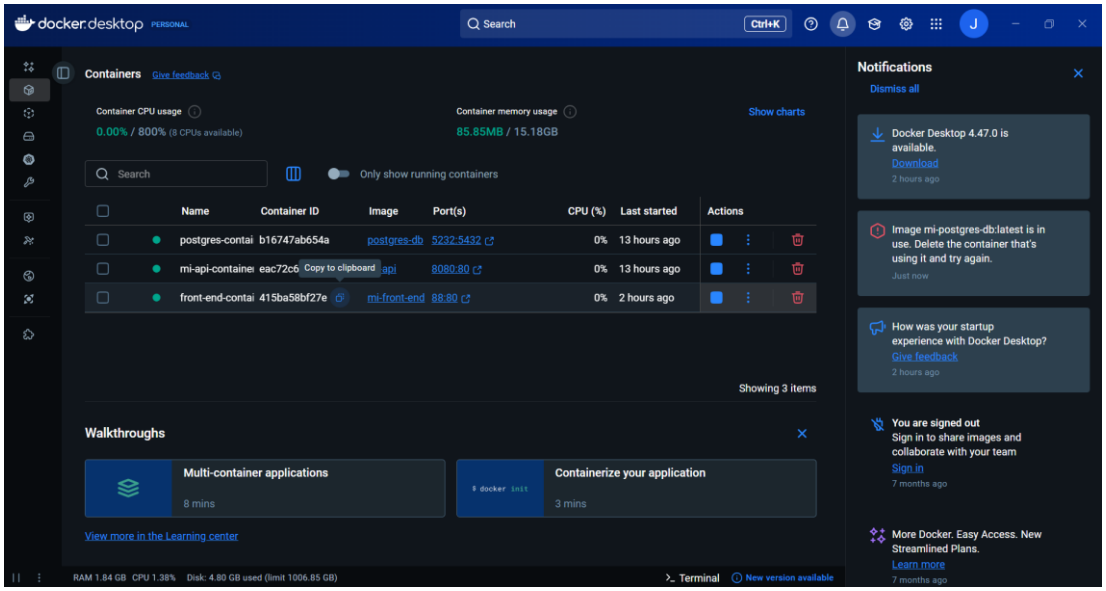

Fig. 26 Visualización de los contenedores desde Docker.desktop.