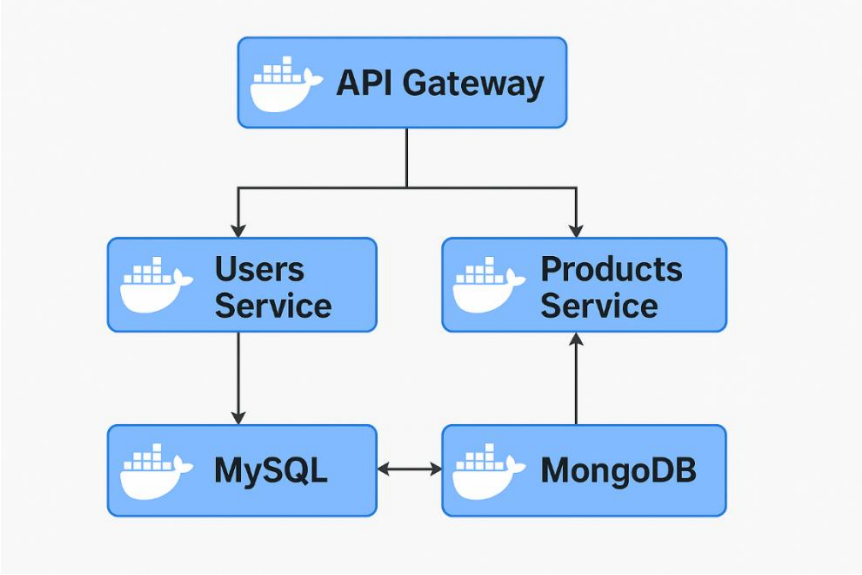
1. **Introducción**

   Esta guía tiene como propósito orientarlos en el proceso de dockerización de un proyecto basado en microservicios.
   El proyecto MICROSERVICIO-GATEWAY implementa una arquitectura modular con comunicación entre servicios y bases de datos.

2. **Arquitectura del proyecto**

- API Gateway está en la parte superior, actuando como punto de entrada para las solicitudes externas.
- Desde el Gateway se comunican dos microservicios:
  - Users Service (gestiona usuarios) → conectado a MySQL
  - Products Service (gestiona productos) → conectado a MongoDB
- Las bases de datos (MySQL y MongoDB) están en la capa inferior y son accesibles solo por sus respectivos servicios.

Todo esto ocurre dentro de la red interna creada por Docker Compose, lo que permite que los contenedores se comuniquen usando nombres de servicio en lugar de direcciones IP.



3. **Preparación del entorno**

   1. Clonar el repositorio: https://github.com/GuilloBurgos/microservicio_api_gateway.git
   2. Abrir el proyecto en Visual Studio Code.
   3. Instalar dependencias con:

      cd api-gateway && npm install
      cd ../products-service && npm install
      cd ../users-service && npm install

4. Creación de Dockerfile para los microservicios

Un Dockerfile define cómo se construirá la imagen de Docker para cada aplicación.

4.1 Dockerfile para api-gateway

Crea el archivo MICROSERVICIO-GATEWAY/api-gateway/Dockerfile:

Dockerfile

```
# Usar una imagen base de Node.js
FROM node:18-alpine

# Establecer el directorio de trabajo dentro del contenedor
WORKDIR /usr/src/app

# Copiar los archivos de definición de dependencias
COPY package*.json ./

# Instalar dependencias
RUN npm install

# Copiar el código fuente de la aplicación al directorio de trabajo
COPY . .

# El API Gateway se ejecuta en el puerto 3000
EXPOSE 3000

# Comando para iniciar la aplicación
CMD [ "node", "index.js" ]
```

4.2 Dockerfile para products-service

Crea el archivo MICROSERVICIO-GATEWAY/products-service/Dockerfile:

Dockerfile

```
# Usar una imagen base de Node.js
FROM node:18-alpine

# Establecer el directorio de trabajo dentro del contenedor
WORKDIR /usr/src/app

# Copiar los archivos de definición de dependencias
COPY package*.json ./

# Instalar dependencias
```

```
RUN npm install

# Copiar el código fuente
COPY . .

# El servicio de productos se ejecuta en el puerto 3002
EXPOSE 3002

# Comando para iniciar la aplicación
CMD [ "node", "index.js" ]
```

4.3 Dockerfile para users-service

Crea el archivo MICROSERVICIO-GATEWAY/users-service/Dockerfile:

Dockerfile

```
FROM node:18-alpine

# Directorio de trabajo
WORKDIR /app

# Copiar archivos de dependencias
COPY package*.json ./

# Instalar dependencias
RUN npm install

# Copiar el resto del código
COPY . .

# Exponer puerto
EXPOSE 3001

# Comando para esperar MySQL y luego iniciar el servicio
CMD sh -c "until nc -z mysql 3306; do echo 'Esperando MySQL...'; sleep 2;
done; node index.js"
```

A tener en cuenta:

Aunque tenemos `depends_on` en el Dockerfile de users-service, **no garantiza que MySQL esté listo para aceptar conexiones** cuando `users-service` arranca.

Por lo tanto fue necesario agregar un script para que `users-service` espere antes de ejecutar `sequelize.authenticate()`.

5. Creación del archivo docker-compose.yml

El archivo docker-compose.yml orquesta todos los servicios (las bases de datos y los microservicios) para que puedan ejecutarse como un solo sistema.

Crea este archivo en la **raíz del proyecto** (MICROSERVICIO-GATEWAY/docker-compose.yml).

5.1 Actualizar Variables de Entorno (.env)

Debes actualizar las variables de entorno de tus microservicios para que apunten a los nombres de host de los contenedores de la base de datos, no a localhost.

Servicio	Archivo	Clave Original	Nuevo Valor (para Docker)
products-service	.env	mongodb://localhost:27017/productos_db	mongodb://mongodb:27017/productos_db
users-service	.env	DB_HOST = localhost	DB_HOST = mysql

Nota: En un entorno de Docker Compose, el nombre del servicio (por ejemplo, mongodb o mysql) actúa como el nombre de host dentro de la red Docker.

5.2 Configuración del docker-compose.yml

YAML

```yaml
services:
  # Base de Datos para Products Service
  mongodb:
    image: mongo:latest
    container_name: mongodb
    ports:
      - "27017:27017" # Solo para acceso externo si es necesario, si no, es opcional
    volumes:
      - mongodb_data:/data/db # Persistencia de datos
    environment:
      MONGO_INITDB_DATABASE: productos_db # Nombre de la DB

  # Base de Datos para Users Service
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - "3307:3306" # Solo para acceso externo si es necesario, si no, es opcional
    environment:
```

```
      MYSQL_ROOT_PASSWORD: rootpassword # Contraseña de root, reemplázala por una más
segura
      MYSQL_DATABASE: node_mvc # Nombre de la DB
    volumes:
      - mysql_data:/var/lib/mysql # Persistencia de datos

  # Microservicio de Usuarios (users-service)
  users-service:
    build:
      context: ./users-service # Ruta al Dockerfile
      dockerfile: Dockerfile
    container_name: users-service
    ports:
      - "3001:3001"
    depends_on:
      - mysql # Asegura que MySQL inicie antes que este servicio
    env_file:
      - ./users-service/.env # Carga las variables de entorno

  # Microservicio de Productos (products-service)
  products-service:
    build:
      context: ./products-service # Ruta al Dockerfile
      dockerfile: Dockerfile
    container_name: products-service
    ports:
      - "3002:3002"
    depends_on:
      - mongodb # Asegura que MongoDB inicie antes que este servicio
    env_file:
      - ./products-service/.env # Carga las variables de entorno

  # API Gateway
  api-gateway:
    build:
      context: ./api-gateway # Ruta al Dockerfile
      dockerfile: Dockerfile
    container_name: api-gateway
    ports:
      - "3000:3000" # Puerto de acceso principal para el cliente
    depends_on:
      - users-service # Asegura que los microservicios estén activos
      - products-service

volumes:
  mongodb_data:
  mysql_data:
```

6. Ejecución del Proyecto con Docker Compose

Una vez que tengas todos los Dockerfile y el docker-compose.yml configurados, puedes levantar todo tu sistema con un solo comando.

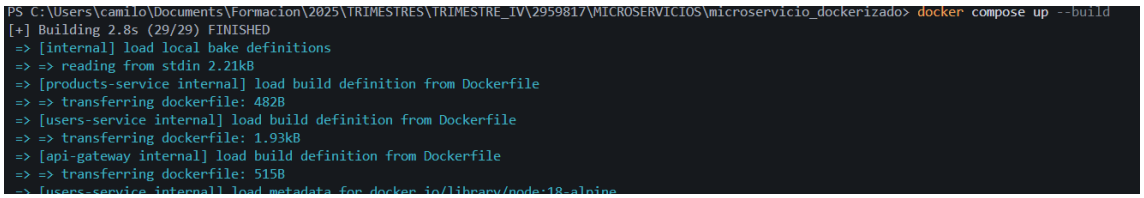
Asegúrate de estar en la carpeta raíz del proyecto (MICROSERVICIO-GATEWAY/).

Ejecuta el siguiente comando en tu terminal:

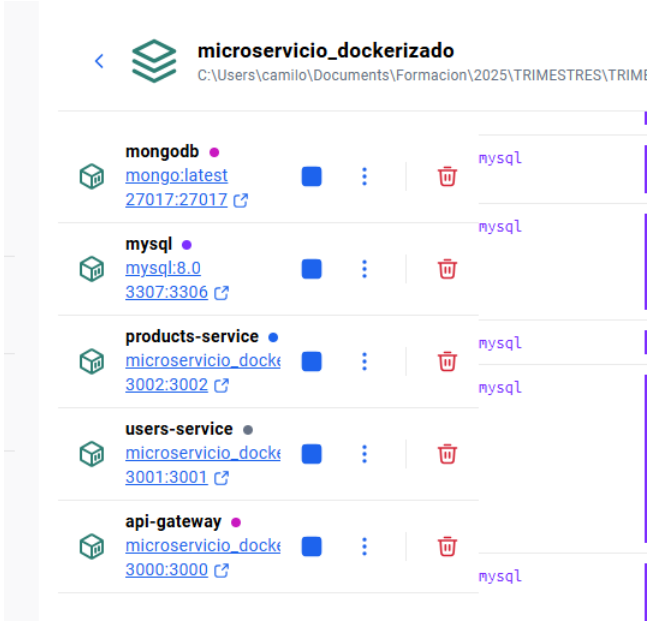
```
\MICROSERVICIOS\microservicio_dockerizado> docker compose up --build
```

- **up:** Crea y arranca los contenedores.
- **build:** Fuerza a Docker a reconstruir las imágenes de tus microservicios (usa esto la primera vez o si has cambiado el código).

Verificación: Una vez que todos los servicios estén arrancados, deberías ver los logs de cada servicio en tu terminal.



Contenedores en Docker Desktop



El **API Gateway** estará escuchando en http://localhost:3000.

Pruebas

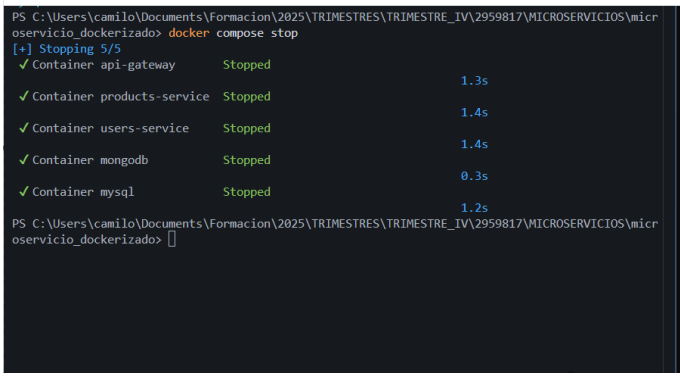
Ahora puedes acceder a tus microservicios a través del API Gateway:

Recurso	Método	URL de prueba
Usuarios	GET	http://localhost:3000/usuarios
Productos	GET	http://localhost:3000/productos
Crear Usuario	POST	http://localhost:3000/usuarios

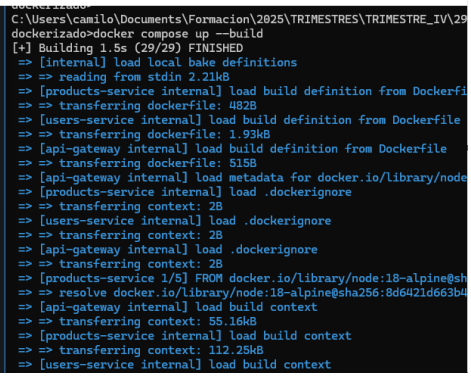
4. Comandos Utiles de Docker

Detener los servicios sin eliminar contenedores:

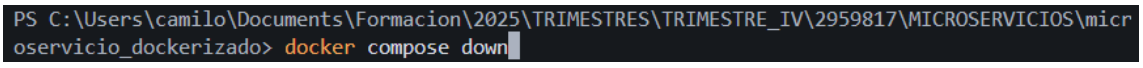
Proceso detenido pero sin eliminar el contenedor y las imágenes:
docker compose stop



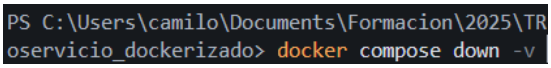
Proceso iniciado con el comando:
docker compose up --build



Detener los servicios y eliminar contenedores:

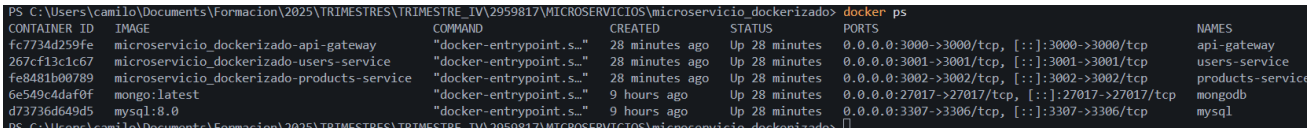


Detener, eliminar contenedores y volúmenes (datos de DB):



Ver el estado de todos tus contenedores:

docker ps



Ver logs de un contenedor específico

Los logs en Docker son los registros generados por un contenedor en ejecución que contienen información sobre su actividad, estado y errores.

```
docker logs api-gateway
docker logs products-service
docker logs mongodb
docker logs mysql
```

- **Para seguir los logs en tiempo real:**

```
docker logs -f api-gateway
```

- **Ver redes y comprobar que todos están en la misma**

```
docker network ls

docker network inspect <nombre_de_la_red>
```

Link del repositorio del proyecto Dockerizado

https://github.com/GuilloBurgos/microservicio_dockerizado.git