# FastAPI + Docker + TailwindCSS

## Desarrollo con Carpeta Compartida

Carlos Rojas Sánchez

November 27, 2025

# Introducción

# Objetivo

- Crear una API con FastAPI.
- Ejecutarla dentro de Docker.
- Desarrollar desde fuera con volúmenes.
- Integrar TailwindCSS.
- Usar imagen ligera: Alpine.

# Estructura del Proyecto

# Estructura Inicial

```
fastapi-docker/
├── app/
│   └── main.py
├── requirements.txt
└── Dockerfile
```

# Código: FastAPI

# Archivo: app/main.py

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI(title="Ejemplo FastAPI en Docker")

class Item(BaseModel):
name: str
price: float
in_stock: bool = True

@app.get("/")
async def root():
return {"message": "Hola desde FastAPI en Docker!"}

@app.post("/items/")
async def create_item(item: Item):
return {"item_received": item}
```

# Docker: Ejecución con Carpeta Compartida

# Volumen Docker

- `-v $(pwd)/app:/app` → Carpeta compartida.
- FastAPI recarga automáticamente cambios.

# Ejecutar Contenedor

```
docker run -it --rm -p 8000:8000
-v $(pwd)/app:/app
fastapi-docker
```

# Integrar TailwindCSS

# Estructura Extendida

```
fastapi-docker/
├── app/
│   ├── main.py
│   ├── static/
│   │   └── output.css
│   └── templates/
│       └── index.html
├── tailwind/
│   ├── input.css
│   └── tailwind.config.js
├── requirements.txt
└── Dockerfile
```

# input.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

# ¿Para qué sirven estas líneas de TailwindCSS?

- `@tailwind base;`
  Carga los estilos base de Tailwind: normalización de CSS, reglas tipográficas y estilos iniciales comunes.
- `@tailwind components;`
  Incluye los componentes predefinidos de Tailwind, como botones, formularios, tarjetas y utilidades agrupadas.
- `@tailwind utilities;`
  Agrega todas las clases utilitarias (p. ej., `p-4`, `bg-blue-500`, `text-center`), que son el núcleo del enfoque utility-first de Tailwind.

# tailwind.config.js

```
module.exports = {
    content: ["../app/templates/**/*.html"],
    theme: {
        extend: {},
    },
    plugins: [],
}
```

# ¿Para qué sirve cada línea en `tailwind.config.js`?

```
module.exports = {
```

• Exporta un objeto para que Node.js y Tailwind puedan usar la configuración.

```
content: ["../app/templates/**/*.html"],
```

• Indica a Tailwind qué archivos debe escanear para detectar qué clases CSS se usan. • Con esto genera solo las clases necesarias (evita CSS pesado). • '**/*.html' significa "todos los HTML en subcarpetas".

```
theme: {
    extend: {},
},
```

• Permite extender el tema por defecto de Tailwind. • Aquí puedes agregar: - Colores personalizados - Tipografías - Espaciados - Breakpoints • 'extend' evita sobreescribir completamente el tema base.

```
plugins: [],
```

• Lista de plugins adicionales (formularios, tipografías, etc.). • Vacío = sin plugins.

```
}
```

# index.html usando TailwindCSS

```
<!DOCTYPE html>
<html>
<head>
  <link href="/static/output.css" rel="stylesheet">
</head>
<body class="bg-gray-100 text-center p-10">
  <h1 class="text-4xl font-bold text-blue-600">
    Hola desde FastAPI + TailwindCSS
  </h1>
</body>
</html>
```

# Dockerfile: Versión Alpine

# Dockerfile Completo (python:3.11-alpine)

```
FROM python:3.11-alpine
RUN apk update && apk add --no-cache
nodejs
npm
gcc
musl-dev
python3-dev
RUN npm install -g tailwindcss
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD tailwindcss -i tailwind/input.css
-o app/static/output.css --watch &
uvicorn app.main:app --host 0.0.0.0 --reload
```

# Ejecutar Proyecto con Tailwind

# Comando Final

```
docker run -it --rm -p 8000:8000
-v $(pwd)/app:/app
-v $(pwd)/tailwind:/tailwind
fastapi-docker
```

# Resultado

- FastAPI corriendo dentro de Docker.
- TailwindCSS se recompila en tiempo real.
- Desarrollo externo con volúmenes.
- Imagen ligera basada en Alpine.