Universidad Intercontinental de la Empresa

Práctica 2.2

Unidad II
Fecha de entrega: 14/11/25 14:59 (hora límite)
Entregables: Documento de texto con enlace al repositorio de GitHub en el que se pueda ver toda la evolución y desarrollo del proyecto y donde esté la contenerización completa.

Objetivo

Contenerizar la práctica 1.3 (implementación básica de clases Python con un main que prueba los métodos a través de TiendaService) construyendo un Dockerfile profesional, ejecutando la app en un contenedor y publicando la imagen en un registry. Practicarás buenas prácticas de Docker, .dockerignore, etiquetado de imágenes, y un flujo build → run → test → push.

Tarea

Utilizando el repositorio creado en la práctica 2.1 con nombre <tunombre>-devops-practica (debe ser público).

1. Crea en la raíz del repositorio un fichero .dockerignore para acelerar la construcción de los builds y evitar contenido innecesario en la imagen con el siguiente contenido.

```
__pycache__/
*.pyc
.git
.gitignore
.venv
venv/
.pytest_cache/
.DS_Store
tests/
```

2. Crea el dockerfile.
    a. Utiliza la imagen python:3.12-slim
    b. Copia las dependencias (requirements.txt) para que se instalen todas las librerías que necesitas. Si no tienes un requirements.txt, crea uno en la carpeta raíz con el comando pip freeze > requirements.txt
    c. Asegúrate de añadir el comando de pip install para que se instalen éstas dependencias.
    d. Copia el código fuente con un comando COPY
    e. Cambia los permisos para que tu usuario pueda ejecutar el código
    f. Añade el comando de inicio con el CMD
3. Construye la imagen y pruébala localmente utilizando docker build y docker run

4. Actualiza el README.md del repositorio con la información relacionada a docker:
    a. Cómo construir la imagen.
    b. Cómo ejecutarla (incluye ejemplos de docker run).
    c. Variables de entorno soportadas (si aplica).
    d. Salida esperada o pasos de prueba (qué debería imprimirse o qué endpoints probar).
5. Sube los cambios al repositorio remoto (*git push origin feature/docker*).
6. Una vez finalizado el trabajo en la rama feature/docker, abre un **Pull Request** hacia la rama main, revisa el código y fusiona los cambios una vez aprobados.
7. Verifica que en la rama main está el dockerfile, .dockerignore, requirements.txt, el código fuente funcional y el README con instrucciones.

Consejos

- Commits pequeños y descriptivos (ej.: feat(docker): primera versión del Dockerfile / docs: añade guía de ejecución).
- Una rama por tarea/feature, vida corta.
- Mantén versiones fijadas en requirements.txt (p. ej., requests==2.32.3).
- Usa variables de entorno (no subas secretos).
- Protege main (PRs, revisiones, checks).
- Considera tamaños de imagen: usa slim, limpia caches, evita copiar .git.
- Si la app crece, evalúa multi-stage build (opcional).
-

Entregable

Un repositorio organizado con:

- dockerfile y .dockerignore.
- requirements.txt con versiones fijadas.
- Código de la práctica 1.3 en /src con main ejecutable y TiendaService.
- README.md completo (build, run, prueba, pull de la imagen).
- Historial de commits claro.
- Ramas feature/docker y main.
- Pull Request registrado y mergeado.