Proyecto TCP Cliente/Servidor

Python — Arquitectura Escalable & Docker-Ready

Este proyecto implementa una arquitectura cliente-servidor TCP modular, escalable y profesional en Python.
Está diseñado para servir como base sólida para aplicaciones que requieran comunicación por sockets, con soporte para pruebas, despliegue con Docker y ejecución local.

🧩 Descripción General

El servidor escucha conexiones TCP y responde mensajes en **mayúsculas**, mientras el cliente interactúa desde consola.

✅ Características

- Estructura limpia y escalable (tipo clean architecture light)
- Logging estructurado y configurable
- Lógica de negocio desacoplada del socket
- Soporte completo para Docker y Makefile
- Pruebas unitarias con `pytest`

⚙️ Tecnologías Usadas

- Python 3.10+
- `socket`, `threading`, `logging` (stdlib)
- Docker & Docker Compose (opcional)
- Makefile (automatización de comandos)
- Pytest (para pruebas)

📦 Estructura del Proyecto

```
tcp_app/
├── app/
│   ├── config/             # Configuración (host, puerto, buffer)
│   ├── core/               # Logger y protocolos
│   ├── services/           # Lógica de negocio (procesamiento de
mensajes)
│   ├── handlers/           # Manejadores de conexión
│   ├── client_app/         # Ejecutor del cliente
│   └── server_app/         # Ejecutor del servidor
├── docker/                 # Dockerfile y docker-compose
├── tests/                  # Pruebas unitarias
```

```
├── Makefile                # Comandos automatizados
├── requirements.txt
├── .dockerignore
└── README.md               # Este archivo
```

🚀 Formas de Ejecutar el Proyecto

🔷 Opción 1: Ejecución Local (sin Docker)

> Requiere tener Python instalado

```
# 1. Crear entorno virtual
python -m venv venv
source venv/bin/activate  # en Windows: venv\Scripts\activate

# 2. Instalar dependencias
pip install -r requirements.txt
```

🟢 **Iniciar servidor:**

```
python -m app.server_app.run_server
```

🔵 **Iniciar cliente (otra terminal):**

```
python -m app.client_app.run_client
```

🔷 Opción 2: Usando Docker

> Requiere tener Docker y Docker Compose

🛠️ **Construir contenedores:**

```
docker-compose -f docker/docker-compose.yml build
```

🟢 **Levantar servidor:**

```
docker-compose -f docker/docker-compose.yml up servidor -d
```

🔵 **Ejecutar cliente (interactivo):**

```
docker-compose -f docker/docker-compose.yml run cliente
```

🔷 Opción 3: Usando Makefile (más cómodo)

> Requiere `make` y Docker instalado

```
make build      # Construye imágenes Docker
make up         # Levanta servidor en background
make client     # Ejecuta cliente en consola interactiva
make test       # Corre pruebas unitarias (pytest)
make down       # Apaga los contenedores
make clean      # Elimina imágenes, volúmenes y contenedores
```

🧪 Pruebas Manuales

✅ Mensaje Normal

```
👉 hola mundo
📄 Respuesta: HOLA MUNDO
```

✅ Desconexión

```
👉 DESCONEXION
⛔ Cliente y servidor cierran sesión correctamente
```

🧪 Pruebas Unitarias

> Requiere tener `pytest` instalado (`pip install pytest`)

```
pytest tests/
```

✨ Extensiones Futuras Sugeridas

- Comandos personalizados (PING, STATS, LOGIN, etc.)
- Persistencia de logs o historial de mensajes
- Control de múltiples clientes simultáneos (broadcast, chat, etc.)
- Panel web usando FastAPI (admin o monitoreo)

👨‍💻 Autor

Desarrollado por Franklin Giovanni Aranda Rodríguez.

📫 Contacto

¿Preguntas o mejoras? ¡Estoy disponible para feedback y contribuciones!
