To-Do App – Microservicios (Prácticas 1, 2 y 3)

Este proyecto implementa una arquitectura de microservicios utilizando Docker Compose, Kafka, balanceo de carga y autenticación servicio a servicio mediante API Key. Todo el sistema se levanta con un solo comando.

1. Arquitectura general

El sistema está compuesto por:

- **tasks-service**: gestión de tareas. Publica eventos en Kafka y valida usuarios vía HTTP.
- **users-service**: gestión de usuarios y estadísticas. Consume eventos de Kafka.
- **PostgreSQL** independiente para cada servicio.
- **Dos réplicas de tasks-service** detrás de un **Load Balancer NGINX**.
- **Kafka + Zookeeper** con creación automática del topic.
- **Frontend NGINX** para gestionar tareas.
- **Adminer** para visualizar datos.

Toda la infraestructura está declarada en `docker-compose.yml` sin pasos manuales.

2. Práctica 1 – División en microservicios

La aplicación se separó en dos microservicios independientes (tasks y users), cada uno con su propia base de datos, imagen Docker y lógica por dominio.
Se implementó un Load Balancer frontal a dos réplicas de `tasks-service`.
La comunicación síncrona utiliza HTTP con retry y circuit breaker básico.

3. Práctica 2 – Kafka y eventos

Se configuró un clúster Kafka con creación automática del topic `tasks-events`.
`tasks-service` publica eventos al crear/editar/eliminar tareas.
`users-service` consume los eventos y actualiza la tabla `tasks_per_user`, eliminando dependencias directas entre servicios.

4. Práctica 3 – Autenticación servicio a servicio

Se implementó autenticación mediante **API Key interna** incluida en `docker-compose.yml`.

- Las llamadas de tasks-service hacia users-service incluyen:

```
X-Internal-Api-Key: supersecret-internal-key
```

- users-service verifica la clave en todos los endpoints internos y rechaza accesos no autorizados.

5. Ejecución del proyecto

Requisito: Docker y Docker Compose.

Para iniciar todo el sistema:

```
docker compose up
```

Sin configuraciones adicionales.

Endpoints de referencia

- Frontend: http://localhost:8090
- Load Balancer (tasks): http://localhost:6000/api/tasks
- Users-service: http://localhost:7000/api/users
- Adminer (DB): http://localhost:8083

6. Organización del repositorio

```
tasks-service/
users-service/
frontend/
docker-compose.yml
README.md
```

7. Resumen

El proyecto cumple:

- División en microservicios
- Réplicas con Load Balancer
- Kafka y arquitectura basada en eventos
- Autenticación servicio a servicio mediante API Key
- Infraestructura completamente declarada como código
- Ejecución directa con `docker compose up`