Entrega 4 - Tolerância a Falhas e Segurança

Disciplina: Sistemas Distribuídos
Data de Entrega: 05/12/2025
Alunos: Danilo Carvalho De Oliveira, Guilherme Faria da Silva, Vinicius Henrique Domingues

---

1. Visão Geral da Entrega

Esta etapa foca em robustecer a arquitetura de microsserviços implementada anteriormente, garantindo que o sistema seja resiliente a falhas parciais e seguro contra interceptações. O objetivo foi implementar mecanismos de **Tolerância a Falhas** (detecção e recuperação automática), **Segurança** (criptografia de tráfego e proteção de cabeçalhos) e **Monitoramento** (observabilidade).

---

2. Tolerância a Falhas

Para atender aos requisitos de detecção e recuperação de falhas, foram implementadas estratégias tanto na camada de aplicação quanto na infraestrutura.

Recuperação Automática (Retry Pattern)

Problema: Na arquitetura distribuída, a operação de "Deletar Lista" no `listas-service` exige uma comunicação síncrona com o `itens-service` para remover os itens associados. Se o `itens-service` estiver temporariamente indisponível (ex: reiniciando), a operação falharia, deixando "itens órfãos" no banco de dados (inconsistência).

Solução: Implementação do padrão de projeto **Retry com Exponential Backoff**.

Implementação: No serviço de listas, a chamada HTTP para o serviço de itens foi encapsulada em um laço de repetição (`while`). O sistema realiza até 3 tentativas de conexão, aguardando 1 segundo (`sleep`) entre elas, antes de declarar falha definitiva. Isso garante que falhas transientes de rede não quebrem a funcionalidade para o usuário.

Detecção de Falhas (Health Checks)

Solução: Adoção do padrão **Health Check API**.

Implementação: Cada microsserviço (`user`, `listas`, `itens`) agora expõe um endpoint dedicado (`/health`) que verifica internamente a conectividade com o banco de dados (MongoDB) e o tempo de atividade (`uptime`).

Resposta de Sucesso (HTTP 200):

```
{
  "status": "UP",
  "database": "connected"
}
```

Isso permite que orquestradores (como Docker ou Kubernetes) e ferramentas de monitoramento detectem automaticamente se um contêiner travou e precisa ser reiniciado.

---

3. Segurança

A segurança foi aplicada em profundidade, cobrindo transporte de dados e proteção da aplicação.

Criptografia de Comunicação (HTTPS/TLS)

Problema: O tráfego HTTP padrão permite que dados sensíveis (senhas, tokens JWT) sejam interceptados em texto puro (sniffing).

Solução: Configuração de criptografia ponta a ponta utilizando **TLS 1.2/1.3**.

Implementação:

- O Nginx foi configurado como **Proxy Reverso com terminação SSL**
- Foram gerados certificados X.509 autoassinados (Self-Signed) via OpenSSL
- O servidor força o redirecionamento de todo tráfego da porta 80 (HTTP) para a 443 (HTTPS)
- Apesar do alerta de "Certificado Autoassinado" no navegador (esperado em ambiente de desenvolvimento sem domínio público), o túnel criptografado está ativo e funcional

Proteção de Aplicação

Implementação: Além da autenticação via JWT (implementada na Entrega 2), foi adicionada a biblioteca **Helmet** nos serviços backend. Isso protege a aplicação contra vulnerabilidades web conhecidas, configurando cabeçalhos HTTP de segurança adequados (como proteção XSS e HSTS).

---

4. Monitoramento e Observabilidade

Para garantir a rastreabilidade do sistema distribuído.

Logs Estruturados

Solução: Substituição de logs simples (`console.log`) por logs estruturados via biblioteca **Winston**.

Implementação: Os logs agora são gerados em formato JSON padronizado, contendo `timestamp`, nível de severidade e metadados da requisição. Isso facilita a ingestão futura por ferramentas de análise de logs (como ELK Stack ou CloudWatch) e permite depuração em tempo real via terminal do Docker.

---

5. Requisitos Atendidos na Entrega

- ✅ **Tolerância a Falhas:** Implementado (Retry Pattern na comunicação síncrona e Health Checks para detecção)
- ✅ **Segurança:** Implementado (Criptografia TLS/HTTPS no Nginx e proteção de headers com Helmet)
- ✅ **Monitoramento:** Implementado (Logs estruturados JSON e endpoints de saúde)