GCC129 – SISTEMAS DISTRIBUÍDOS

DOCENTE: BRUNO DA SILVA MACÊDO

RELATÓRIO FINAL

CARLOS EDUARDO **BORGES DE SOUSA**

Sistema Distribuído com Múltiplos Agentes de IA

Lavras – MG
2025

Sumário

1. Introdução **3**
2. Desenvolvimento **4**
2.1 Arquitetura do Sistema 4
2.2 Agentes Inteligentes 4
2.3 Comunicação e Microsserviços 4
2.4 Testes e Exemplo de Uso 5
2.5 Visões Arquitetônicas 5
2.6 RIPD e STRIDE 6
2.6.1 Identificação e Avaliação dos Riscos 6
2.6.2 Medidas de Mitigação Adotadas 7
3. Considerações Finais **9**
4. Referências **10**

1. Introdução

Sistemas distribuídos são fundamentais para a construção de aplicações escaláveis, resilientes e de alto desempenho. Com o avanço da Inteligência Artificial (IA) e da modularização de sistemas via microsserviços, tornou-se viável a criação de arquiteturas compostas por múltiplos agentes autônomos, que se comunicam entre si para realizar tarefas especializadas.

Este projeto teve como objetivo a construção de um sistema distribuído composto por múltiplos agentes de IA, cada um com uma responsabilidade específica. O sistema utiliza comunicação entre microsserviços, containers Docker para isolamento e gerenciamento, e um frontend intuitivo. As funcionalidades oferecidas incluem análise de texto, cálculos estatísticos e análise financeira personalizada, representando um exemplo prático da integração entre IA e arquiteturas distribuídas modernas.

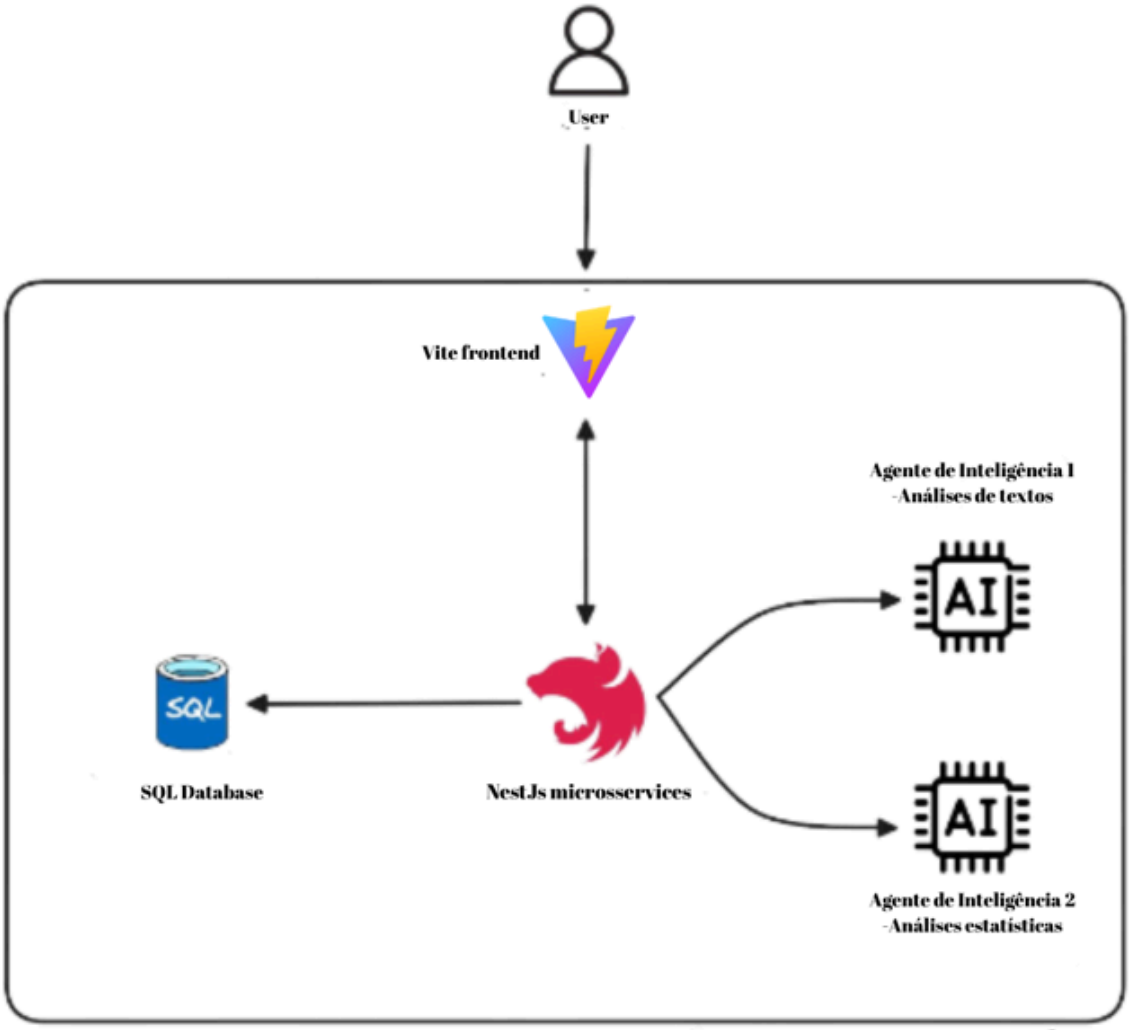
Além da implementação técnica, o projeto também considerou aspectos de segurança, documentação arquitetônica e validação do problema abordado.

2. Desenvolvimento

2.1 Arquitetura do Sistema

O sistema é composto por três agentes principais de IA, todos implementados como microserviços e expostos via APIs REST. A comunicação entre eles segue o padrão Agent-to-Agent (A2A), em que os serviços interagem diretamente por meio de chamadas HTTP, cada um executando funções específicas:

- Agent1-IA (porta 5000): realiza análise de textos, extraindo métricas estatísticas e interpretações semânticas.
- Agent2-IA (porta 5001): executa cálculos estatísticos como média, desvio padrão e mediana.
- Finance Agent (porta 5002): realiza análises financeiras personalizadas com base em dados numéricos e descrições textuais.



O backend, desenvolvido com NestJS, orquestra a comunicação entre os agentes e funciona como ponte para o frontend, desenvolvido em React com TypeScript, que oferece uma interface responsiva, moderna e de fácil utilização.

2.2 Agentes Inteligentes

Os três agentes de IA implementados possuem funcionalidades distintas e foram desenvolvidos com foco em modularidade. Cada agente é executado de forma independente em sua respectiva porta e exposto como microsserviço. O Finance Agent foi containerizado usando Docker, promovendo isolamento e escalabilidade.

Os agentes colaboram entre si: o Agent1-IA pode acionar o Agent2-IA para complementar análises com cálculos estatísticos, demonstrando a flexibilidade e integração da arquitetura baseada em microserviços.

2.3 Comunicação e Microsserviços

A comunicação entre os agentes segue o padrão Agent-to-Agent (A2A), com requisições HTTP internas configuradas por variáveis de ambiente. Cada agente funciona como microsserviço autônomo, iniciado por scripts específicos. O sistema expõe uma API RESTful implementada em NestJS, que atua como intermediário entre os agentes e o frontend.

2.4 Testes e Exemplo de Uso

Um exemplo de input para o agente financeiro:

```
{
  "monthly_income": 5000,
  "monthly_expenses": 3500,
  "total_savings": 8000,
  "total_debt": 12000
}
```

Saída esperada:
- Taxa de Poupança: 30%
- Ratio de Despesas: 70%
- Ratio de Dívidas: 240%
- Meses de Fundo de Emergência: 2.3

Dicas geradas incluem: consolidar dívidas, controlar gastos com necessidades básicas e automatizar economias.

Além disso, o projeto inclui testes automatizados com um script específico (`test_finance_agent.py`) que valida os endpoints do agente financeiro.

2.5 Visões Arquitetônicas

A visão arquitetônica inicial do sistema foi fundamentada na separação de responsabilidades entre os agentes de IA, o backend central e o frontend. Cada agente realiza tarefas específicas e se comunica por meio de APIs REST, promovendo modularidade, reusabilidade e escalabilidade.

Após a fase de desenvolvimento, foram adotadas práticas adicionais de segurança e organização dos serviços. Utilizou-se o isolamento de contêineres por meio do Docker, variáveis de ambiente para configuração dos agentes, e scripts de inicialização para automatizar o processo. Essa estrutura contribui para a manutenção e evolução futura do sistema.

2.6 RIPD e STRIDE

Com o objetivo de garantir a segurança da aplicação distribuída desenvolvida, foi realizada uma análise de riscos baseada no modelo STRIDE, aliado aos princípios da avaliação de impacto à proteção de dados. Essa abordagem permite identificar possíveis ameaças em sistemas distribuídos, categorizando-as conforme seus efeitos sobre a integridade, confidencialidade, disponibilidade e autenticidade do sistema.
A metodologia STRIDE contempla seis classes de ameaças principais: falsificação de identidade (Spoofing), manipulação de dados (Tampering), repúdio de ações (Repudiation), exposição de informações (Information Disclosure), negação de serviço (Denial of Service) e elevação de privilégios (Elevation of Privilege). Cada uma dessas categorias foi analisada dentro do escopo da arquitetura implementada, considerando os agentes de IA, o backend orquestrador e a interface frontend.

2.6.1 Identificação e Avaliação dos Riscos

Spoofing (Falsificação de Identidade)

- **Risco:** Um serviço externo tenta simular um dos agentes legítimos.
- **Impacto:** Pode resultar na execução de análises com dados maliciosos ou falsificados.
- **Probabilidade:** Média
- **Severidade:** Alta

Tampering (Manipulação de Dados)

- **Risco:** Interceptação e modificação de dados enviados entre os serviços.
- **Impacto:** Geração de resultados incorretos ou manipulados nas análises dos agentes.
- **Probabilidade:** Média
- **Severidade**: Alta

Repudiation (Repúdio de Ações)

- **Risco:** Dificuldade em rastrear ações realizadas nos agentes e backend.
- **Impacto:** Impossibilidade de verificar a origem de resultados ou identificar falhas.
- **Probabilidade:** Média
- **Severidade:** Média

Information Disclosure (Exposição de Informações)

- **Risco:** Acesso não autorizado a dados sensíveis fornecidos pelo usuário, como renda e dívidas.
- **Impacto:** Comprometimento da privacidade e possível uso indevido das informações.
- **Probabilidade:** Média
- **Severidade:** Alta

Denial of Service (Negação de Serviço)

- **Risco:** Envio de múltiplas requisições seguidas para sobrecarregar os agentes ou backend.
- **Impacto:** Instabilidade ou interrupção temporária do sistema.
- **Probabilidade:** Média
- **Severidade:** Alta

Elevation of Privilege (Elevação de Privilégios)

- **Risco:** Tentativas de um componente acessar funções que não lhe são permitidas.
- **Impacto:** Acesso indevido a outras partes do sistema.
- **Aplicabilidade:** Não aplicável, pois o sistema não possui níveis hierárquicos ou controle de usuários.

2.6.2 Medidas de Mitigação Adotadas

Spoofing

- Cada agente é executado em porta exclusiva e configurado por meio de variáveis de ambiente, dificultando acessos externos não autorizados.

Tampering

- A comunicação entre backend e agentes é feita localmente em ambiente controlado. Os serviços são isolados em containers Docker, reduzindo a superfície de ataque.

Repudiation

- O backend mantém registros internos (logs) com dados básicos de requisições, permitindo identificar erros ou comportamentos inesperados durante a execução.

Information Disclosure

- Nenhum dado sensível é armazenado. As informações são utilizadas apenas em memória, durante o processamento. A aplicação é local, reduzindo o risco de vazamentos.

Denial of Service

- O backend trata falhas dos agentes com mensagens amigáveis, evitando interrupções em cadeia. Há validações de entrada para minimizar o processamento desnecessário.

Elevation of Privilege

- Como o sistema não possui autenticação ou diferentes perfis de usuários, esse risco não se aplica ao contexto atual da aplicação.

3. Considerações Finais

O projeto atendeu aos objetivos propostos, implementando uma arquitetura moderna baseada em microserviços e múltiplos agentes de IA. O sistema é funcional, bem documentado e preparado para expansão.

Foram contempladas práticas importantes como container, fallback, testes automatizados, uso de API RESTful e segurança baseada no modelo STRIDE. O frontend proporciona uma experiência clara e eficaz para o usuário, com formulários intuitivos e visualização organizada dos resultados.

Trata-se de uma solução com potencial de aplicação prática e relevância social, especialmente no auxílio à educação e planejamento financeiro.

4. Referências

TANENBAUM, A. S.; VAN STEEN, M. Distributed Systems: Principles and Paradigms. Pearson Education, 2007.

CHACON, S.; STRAUB, B. Pro Git. Apress, 2014. Disponível em: https://git-scm.com/book/en/v2.

OWASP Foundation. OWASP Top Ten Web Application Security Risks. 2023. Disponível em: https://owasp.org/www-project-top-ten/.