A PERFORMANCE COMPARISON OF AUTHENTICATION AND AUTHORIZATION PATTERNS FOR MICROSERVICES APPLICATIONS

Rafael Freitas Cardoso

Fevereiro de 2024

Orientador: Prof. Dr. Jéferson Campos Nobre

AGENDA

1. Introdução
2. Trabalhos Relacionados
3. Metodologia
4. Resultados
5. Conclusões

"An approach to developing software applications as a suite of small, independent services that communicate with each other through lightweight mechanisms, often using an HTTP resource API"

(Fowler, 2014)

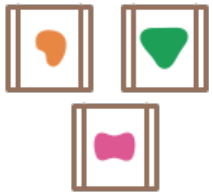
ARQUITETURA DE MICROSSERVIÇOS

- Estilo arquitetural emergente.
- Aplicação é decomposta em pequenos serviços independentes.
- Cada serviço possui uma funcionalidade bem definida.
- Comunicação entre serviços através de APIs.
- Encapsulamento da aplicação via gateway de roteamento.
- Geralmente acompanham contêinerização (Docker) e orquestração (Kubernetes).

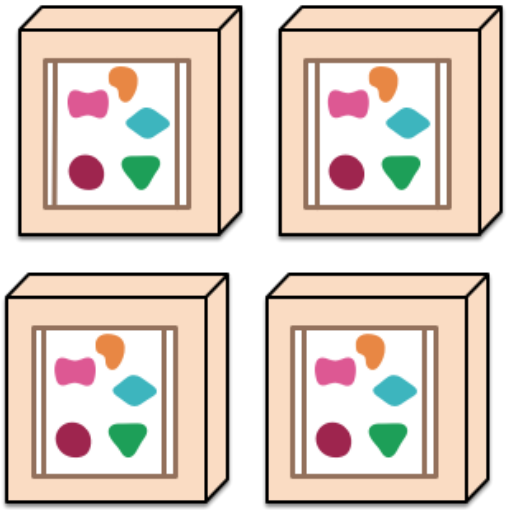
ARQUITETURA DE MICROSSERVIÇOS

A monolithic application puts all its functionality into a single process...

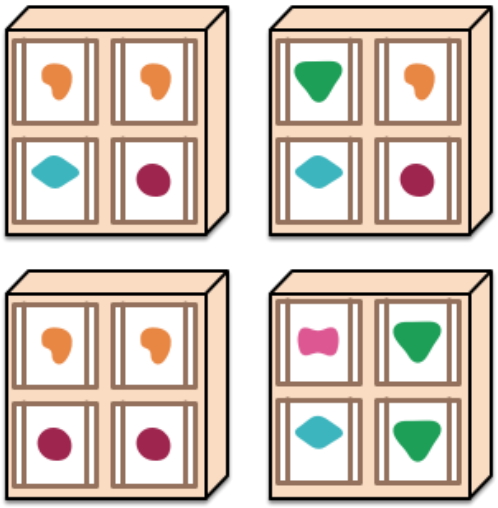
A microservices architecture puts each element of functionality into a separate service...



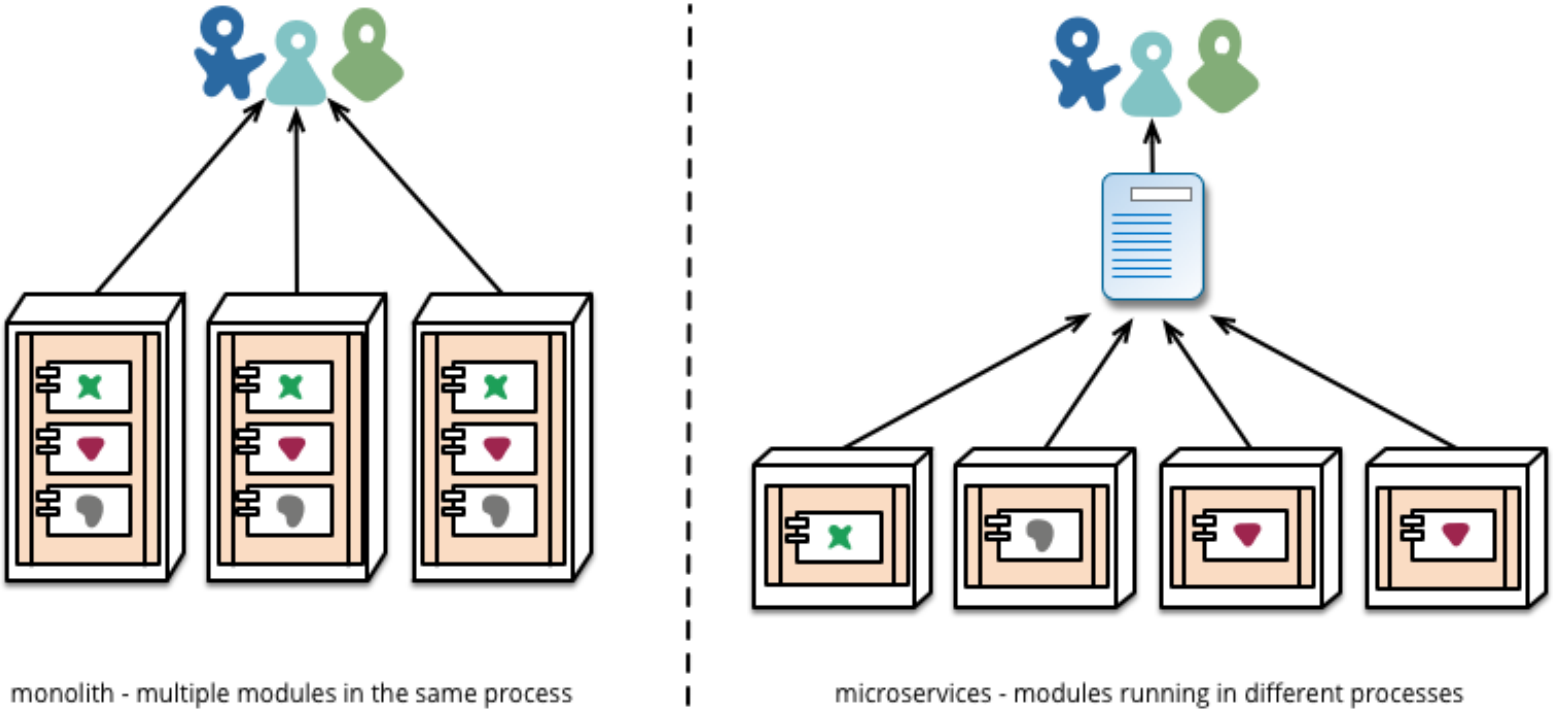
... and scales by replicating the monolith on multiple servers



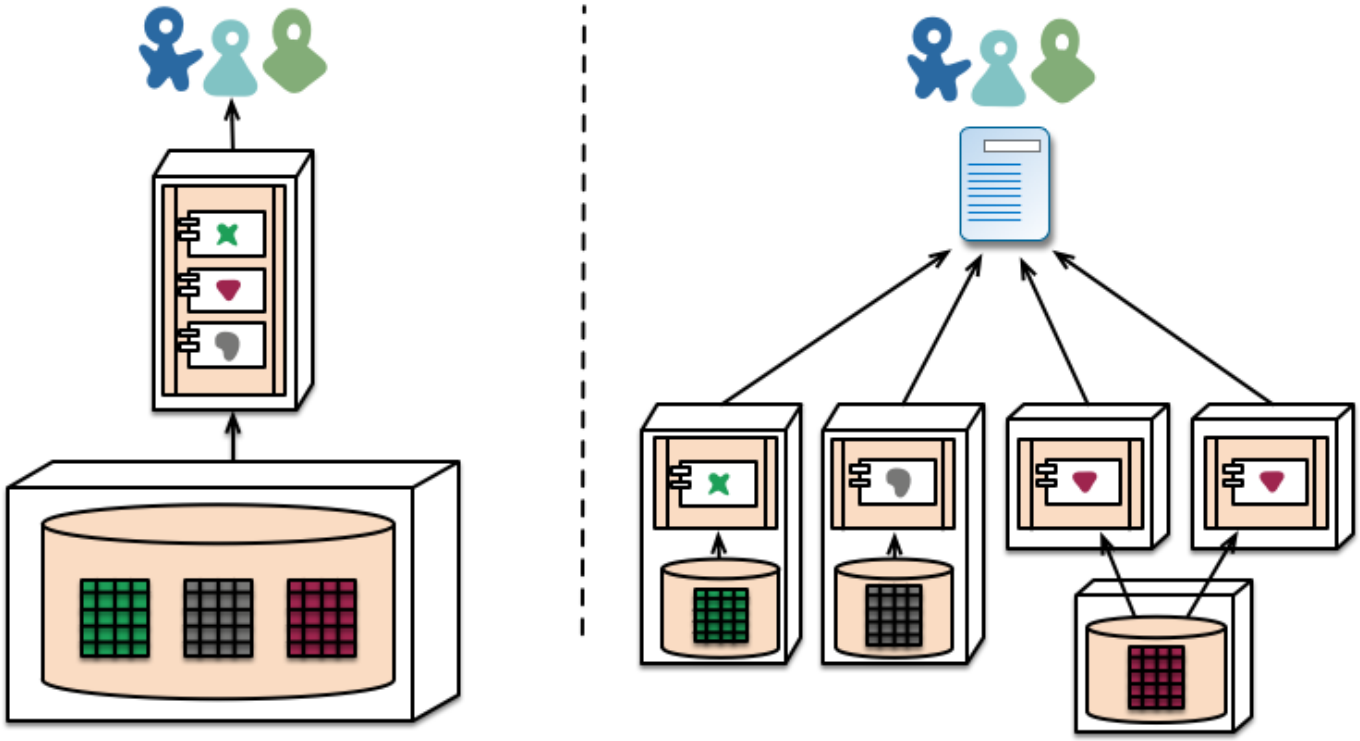
... and scales by distributing these services across servers, replicating as needed.



(Fowler, 2014)


monolith - multiple modules in the same process
microservices - modules running in different processes

(Fowler, 2014)


monolith - single database
microservices - application databases

(Fowler, 2014)

DESAFIOS DE SEGURANÇA

- Aumento da superfície de ataque resultante da proliferação de APIs e *endpoints*.
- Comunicação de dados sobre redes potencialmente inseguras.
- **Autenticação e autorização em ambientes distribuídos e dinâmicos.**

Edge-Level Authentication and Authorization (OWASP, 2017)

- A lógica de autenticação e autorização é transferida para a borda da aplicação, tipicamente no gateway ou *ingress controller*.
- Pode introduzir dependência da infraestrutura de rede.
- Não suporta a implementação de mecanismos de defesa em profundidade.

Centralized Service-Level Authentication and Authorization (OWASP, 2017)

- Centralização da lógica de autenticação e autorização dentro de um serviço dedicado ou provedor de identidade.
- Oferece controle e gerenciamento centralizados de identidades e permissões de usuários, simplificando a administração e a aplicação de políticas de controle de acesso.
- Pode introduzir pontos únicos de falha e gargalos de escalabilidade.

Decentralized Service-Level Authentication and Authorization (OWASP, 2017)

- Distribuição da lógica de autenticação e autorização entre serviços individuais.
- Gerenciamento de forma independente das identidades e permissões dos usuários.
- Maior autonomia e flexibilidade na definição de regras de negócio.
- Podem resultar na aplicação inconsistente das políticas de controle de acesso e no aumento da complexidade na sua gestão.

PERGUNTAS DE PESQUISA

- Qual é o impacto geral dos padrões de autenticação e autorização no desempenho de aplicações de microsserviços?
- Qual é o impacto geral desses padrões no consumo de recursos das aplicações?
- Como os impactos desses padrões se comparam entre si? Qual minimiza overhead e consumo de recursos?

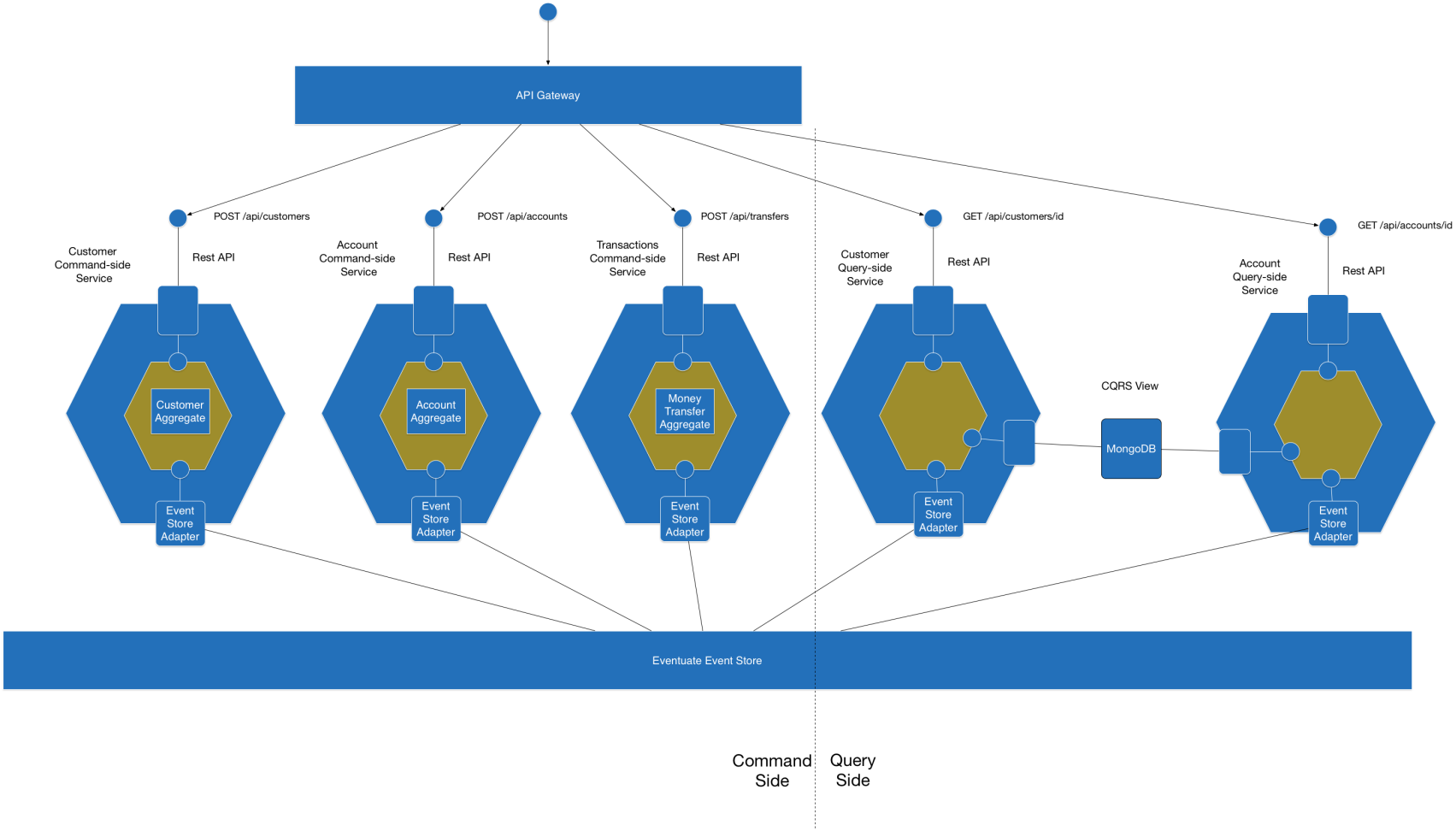
- (NASAB et al., 2023) Estudo empírico para identificar e validar práticas de segurança para sistemas de microsserviços. Análise manual de 861 pontos (GitHub e Stack Overflow). Pesquisa com 74 profissionais para avaliar a utilidade das práticas identificadas.
- (PEREIRA-VALE et al., 2019) Estudo de mapeamento visando identificar mecanismos de segurança usados em sistemas baseados em microsserviços descritos na literatura. Seleção de 26 artigos, extração, classificação e organização.
- Aspectos de segurança que são considerados mais importantes: **Autenticação**, **Autorização** e **Gerência de Credenciais**.

- (Triartono et al. 2019) Proposta de implementação de Role-Based Access Control (RBAC) através do uso de OAuth 2.0. OAuth é um framework centralizado, define um ponto unificado de controle.
- (Yarygina and Bagge 2018) Proposta do uso de Mutual Transport Layer Security (MTLS) em conjunto com tokens para realização de autenticação local (cada serviço é responsável por sua própria autenticação).

- (Sedghpour and Townend 2022]) Proposta de uso do eBPF (extended Berkeley Packet Filter) como um meio eficiente de se medir a performance de microsserviços. Tecnologia de monitoramento de eventos dentro do Kernel Linux.
- (Miano et al. 2021) Pesquisa acerca do uso do eBPF para funções de rede em microsserviços. Demonstração de programas para monitorar tráfego, além de medir latência e consumo de recursos.

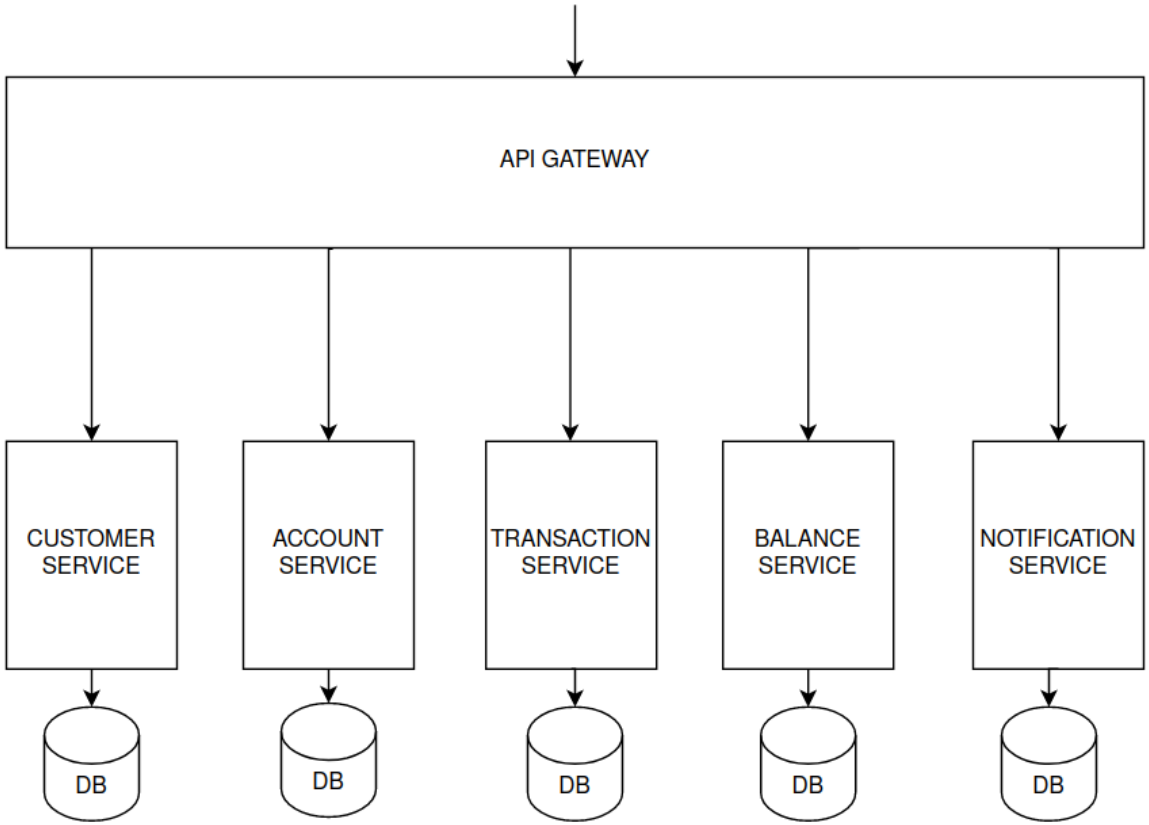
- (Costa et al. 2022) Avaliação e comparação do impacto em performance dos padrões de Tolerância a Falhas **Retry** e **Circuit Breaker** em aplicações de microsserviços.
- (Sedghpour et al. 2021) Estudo empírico para avaliar o impacto de alguns padrões de resiliência em cenários diversos de uma arquitetura de serviços. Proposta de otimização em um dos padrões.

- Desenvolver uma aplicação representativa da arquitetura de microsserviços.
- Gerar diferentes versões da aplicação base, incluindo mecanismos de autenticação e autorização para implementar cada um dos três padrões mencionados anteriormente.
- Conceber uma arquitetura de deploy e instrumentação da aplicação, suportando métricas relacionadas à performance e uso de recursos.
- Coletar métricas de cada versão da aplicação sob testes direcionados de carga.
- Comparar e analisar as métricas obtidas de forma a indicar o padrão de melhor desempenho geral.



(Retirado de https://github.com/cer/event-sourcing-examples)

- Escrita em Go
- Comunicação entre serviços via HTTP



ENDPOINTS DOS SERVIÇOS

Serviço	Endpoint
Customer Service	createCustomer
Customer Service	deleteCustomer
Customer Service	getCustomer
Account Service	createAccount
Account Service	deleteAccount
Account Service	deleteAccountsByCustomer
Account Service	addToBalance
Account Service	subtractFromBalance
Account Service	getAccount
Account Service	getAccountsByCustomer
Transaction Service	transferAmount
Transaction Service	transferAmountAndNotify
Transaction Service	getTransaction
Notification Service	notify
Notification Service	getNotification
Balance Service	getBalanceByCustomer
Balance Service	getBalanceHistory

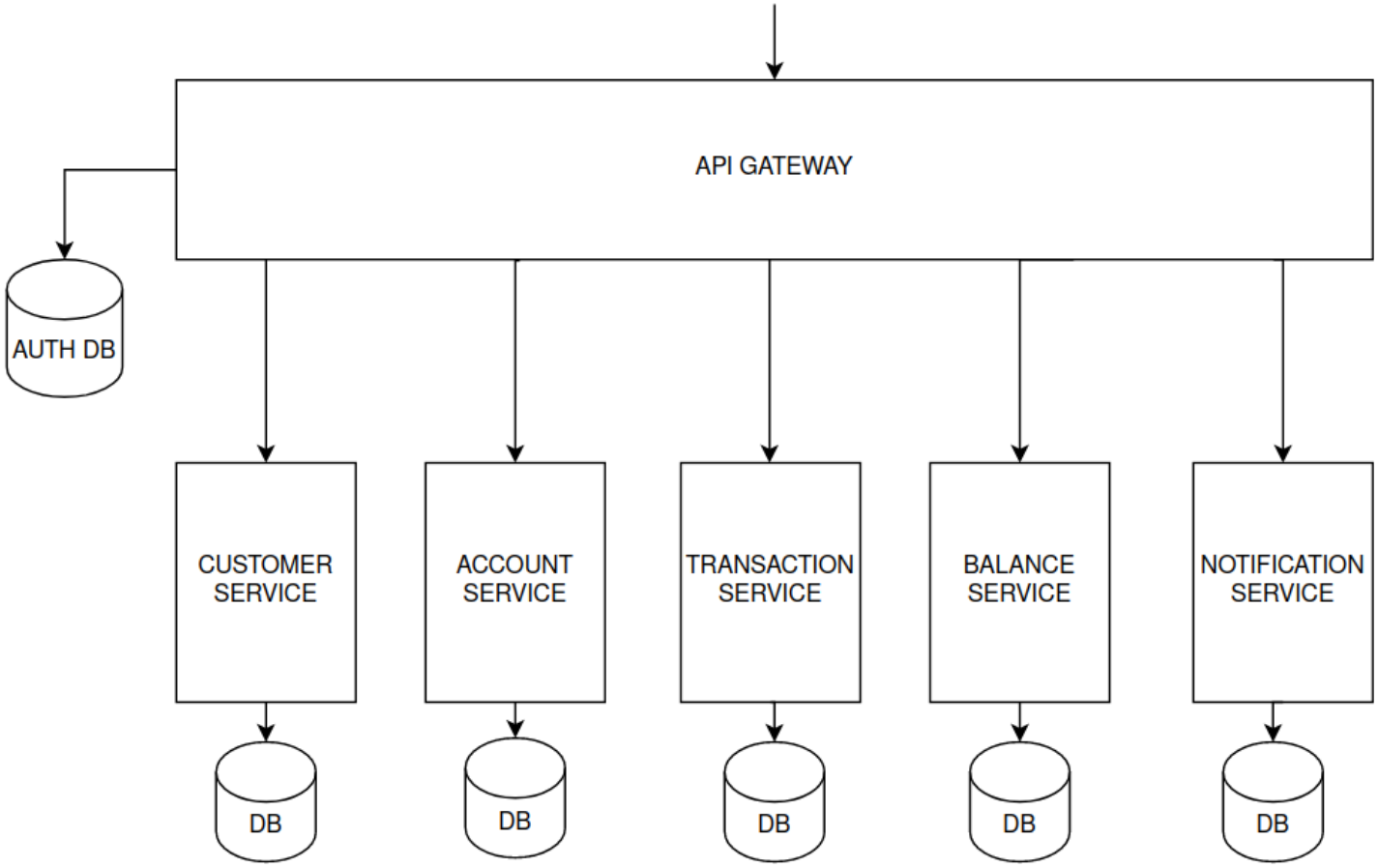
PROFUNDIDADE DOS ENDPOINTS

Endpoint	Serviços Envolvidos	Profundidade
createCustomer	Customer, Account	1
deleteCustomer	Account	0
getCustomer	Account	0
createAccount	Account	0
deleteAccount	Account	0
deleteAccountsByCustomer	Account	0
addToBalance	Account	0
subtractFromBalance	Account	0
getAccount	Account	0
getAccountsByCustomer	Account	0
transferAmount	Transaction, Account	2
transferAmountAndNotify	Transaction, Account, Notification, Customer	5
getTransaction	Transaction	0
notify	Notification, Account, Customer	2
getNotification	Notification	0
getBalanceByCustomer	Balance, Account	1
getBalanceHistory	Balance	0

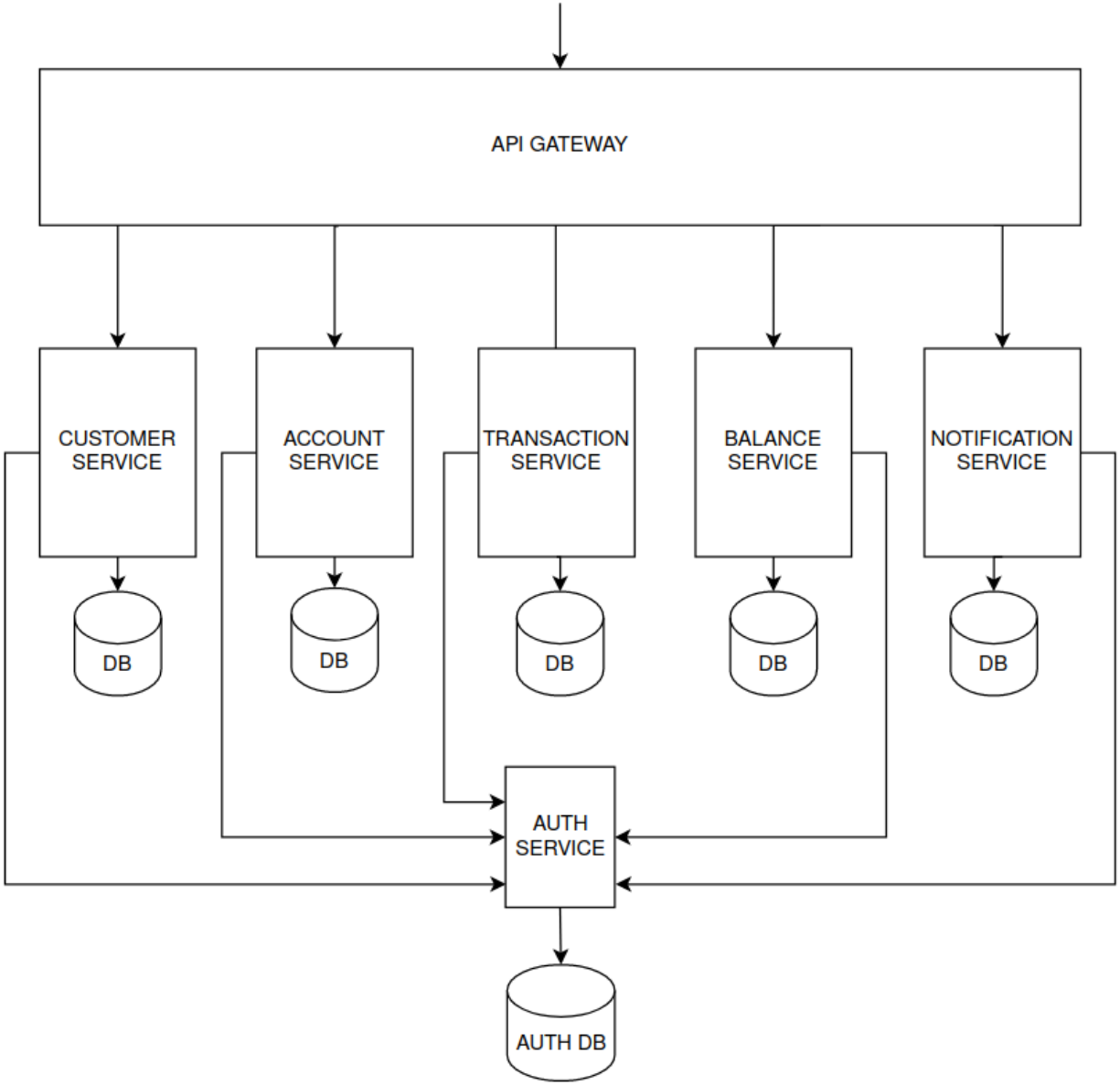
- Baseados em uma tabela de usuários.
- **Autenticação:** Combinação ID do usuário + senha passada na requisição para a API corresponde à uma entrada no banco de dados.
- **Autorização:** Usuário possui a permissão associada ao endpoint.

Campo	Tipo de Dado	Restrições
user_id	VARCHAR(255)	Primary Key
user_password	VARCHAR(255)	Not Null
can_read	BOOLEAN	Not Null
can_write	BOOLEAN	Not Null
can_delete	BOOLEAN	Not Null

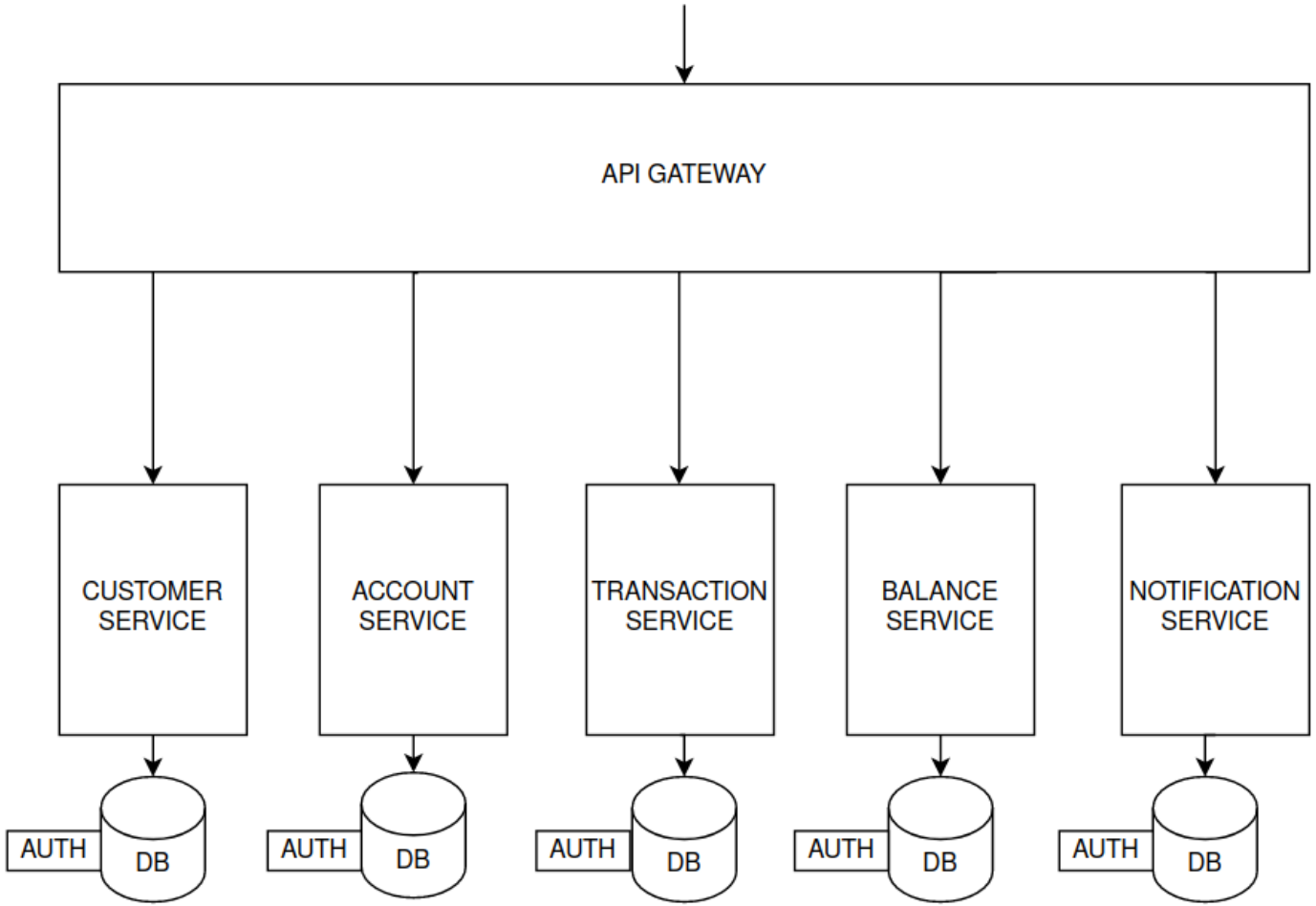
APLICAÇÃO DESENVOLVIDA - VERSÃO EDGE



APLICAÇÃO DESENVOLVIDA - VERSÃO CENTRALIZADA



APLICAÇÃO DESENVOLVIDA - VERSÃO DESCENTRALIZADA



- Serviços são encapsulados por contêineres Docker.
- Bancos de dados são implementados por instâncias PostgreSQL, também encapsulados por contêineres.
- Deploy sequencial das versões da aplicação no Cluster mantido pelo Grupo de Redes do INF, por meio de manifestos Kubernetes, dentro de um *namespace* exclusivo.
- Cada contêiner é individualmente executado em um pod, incluindo o gateway.
- Todos os pods recebem 1 CPU dedicada e 1 GB de memória, estando limitados a esses valores.

A coleta de métricas da aplicação se deu em dois níveis.

- No nível **externo**, uma ferramenta em Python foi desenvolvida para enviar requisições para o gateway da aplicação. A ferramenta simula interações de usuário com o sistema, coletando o tempo de resposta dos endpoints.
- No nível **interno**, o *namespace* recebeu instâncias do **Prometheus** e do **Cillium Hubble**. Essas ferramentas foram utilizadas para coletar dados relacionados ao consumo de recursos e uso de rede por parte da aplicação.

EXPERIMENTOS REALIZADOS

- Para cada um dos 17 endpoints, foram executadas 20.000 requisições sequenciais.
- O processo foi repetido 10 vezes, totalizando 200.000 requisições.
- Ao longo do período, a arquitetura previamente descrita coletou métricas em tempo real.

TEMPO DE RESPOSTA MÉDIO

Endpoint - Versão da Aplicação	**Tempo de Resposta (ms)**
createAccount - Baseline	4.26
createAccount - Edge	5.30 (124.41%)
createAccount - Centralized	5.86 (137.55%)
createAccount - Decentralized	4.90 (115.02%)
createCustomer - Baseline	5.85
createCustomer - Edge	6.98 (119.31%)
createCustomer - Centralized	8.45 (144.44%)
createCustomer - Decentralized	7.39 (126.32%)
transferAmount - Baseline	7.18
transferAmount - Edge	8.58 (119.49%)
transferAmount - Centralized	10.74 (149.58%)
transferAmount - Decentralized	9.07 (126.32%)
transferAmountAndNotify - Baseline	10.39
transferAmountAndNotify - Edge	11.97 (115.20%)
transferAmountAndNotify - Centralized	16.78 (161.50%)
transferAmountAndNotify - Decentralized	14.06 (135.32%)

MÉDIA DE REQUISIÇÕES PROCESSADAS POR SEGUNDO

Endpoint - Versão da Aplicação	**Requisições por Segundo**
createAccount - Baseline	228.78
createAccount - Edge	184.07 (80.45%)
createAccount - Centralized	167.31 (73.13%)
createAccount - Decentralized	199.61 (87.24%)
createCustomer - Baseline	169.48
createCustomer - Edge	141.98 (83.77%)
createCustomer - Centralized	117.51 (69.33%)
createCustomer - Decentralized	134.29 (79.23%)
transferAmount - Baseline	138.51
transferAmount - Edge	115.89 (83.66%)
transferAmount - Centralized	92.63 (66.87%)
transferAmount - Decentralized	109.69 (79.19%)
transferAmountAndNotify - Baseline	95.83
transferAmountAndNotify - Edge	83.14 (86.75%)
transferAmountAndNotify - Centralized	59.38 (61.96%)
transferAmountAndNotify - Decentralized	70.85 (73.93%)

USO TOTAL MÉDIO DE CPU

Endpoint - Versão da Aplicação	**Uso de CPU Total (%)**
createAccount - Baseline	46.54
createAccount - Edge	54.46 (117.01%)
createAccount - Centralized	61.15 (131.39%)
createAccount - Decentralized	53.51 (114.97%)
createCustomer - Baseline	48.41
createCustomer - Edge	53.25 (109.99%)
createCustomer - Centralized	63.66 (131.50%)
createCustomer - Decentralized	55.31 (114.25%)
transferAmount - Baseline	49.92
transferAmount - Edge	55.23 (110.63%)
transferAmount - Centralized	63.79 (127.78%)
transferAmount - Decentralized	56.01 (126.32%)
transferAmountAndNotify - Baseline	52.83
transferAmountAndNotify - Edge	58.63 (110.97%)
transferAmountAndNotify - Centralized	67.36 (127.50%)
transferAmountAndNotify - Decentralized	59.17 (112%)

Endpoint - Versão da Aplicação	**Uso de Memória Total (MiB)**
createAccount - Baseline	171.74
createAccount - Edge	205.67 (119.75%)
createAccount - Centralized	214.82 (125.08%)
createAccount - Decentralized	210.41 (122.51%)
createCustomer - Baseline	173.84
createCustomer - Edge	206.68 (118.89%)
createCustomer - Centralized	216.98 (124.81%)
createCustomer - Decentralized	211.29 (121.54%)
transferAmount - Baseline	175.69
transferAmount - Edge	209.37 (119.17%)
transferAmount - Centralized	218.49 (124.36%)
transferAmount - Decentralized	213.68 (121.62%)
transferAmountAndNotify - Baseline	177.41
transferAmountAndNotify - Edge	210.96 (118.91%)
transferAmountAndNotify - Centralized	219.99 (124%)
transferAmountAndNotify - Decentralized	215.21 (121.30%)

USO TOTAL MÉDIO DE REDE

Endpoint - Versão da Aplicação	**Total de Dados Transmitidos (MB/s)**
createAccount - Baseline	1.12
createAccount - Edge	1.26 (112.5%)
createAccount - Centralized	1.46 (130.35%)
createAccount - Decentralized	1.37 (122.32%)
createCustomer - Baseline	1.26
createCustomer - Edge	1.44 (114.28%)
createCustomer - Centralized	1.58 (125.39%)
createCustomer - Decentralized	1.55 (123.01%)
transferAmount - Baseline	1.30
transferAmount - Edge	1.52 (116.92%)
transferAmount - Centralized	1.69 (130%)
transferAmount - Decentralized	1.62 (124.61%)
transferAmountAndNotify - Baseline	1.44
transferAmountAndNotify - Edge	1.64 (113.88%)
transferAmountAndNotify - Centralized	1.92 (133.33%)
/transferAmountAndNotify - Decentralized	1.79 (124.30%)

USO TOTAL DE DISCO

Versão da Aplicação	Uso Total de Disco (MB)
Baseline	242.9
Edge	322 (132.56%)
Centralized	322 (132.56%)
Decentralized	394 (162.20%)

- Os dados coletados mostraram que o padrão de autenticação e autorização empregado possui impacto significativo tanto no desempenho quanto no perfil de consumo de recursos e uso de rede da aplicação.
- O padrão **centralizado** exibiu consistentemente o **maior** tempo de resposta e a **maior** utilização de recursos em todos os endpoints, indicando ineficiência em relação aos outros padrões.

- O padrão **de borda** exibiu consistentemente o **menor** tempo de resposta e a **menor** utilização de recursos em todos os endpoints, seguido de perto pelo padrão **descentralizado**, demonstrando serem padrões mais eficientes.
- É importante ressaltar que esses dois padrões **não** são equivalentes e **não** oferecem o mesmo nível de segurança.
- Em termos de custo-benefício, o padrão **descentralizado** emerge como o mais eficiente para a arquitetura de microsserviços.

- A contra indicação do padrão **descentralizado** corresponde a ambientes com limitações fortes em termos de espaço de armazenamento.
- A necessidade de manutenção de múltiplas cópias dos dados de usuário gera um uso aumentado de disco proporcional à quantidade de serviços.

CONTRIBUIÇÕES

Neste trabalhos nós:

- Desenvolvemos uma aplicação open-source representativa da arquitetura de microsserviços.
- Implementamos uma arquitetura de metrificação que pode ser replicada em outros projetos.
- Coletamos dados de desempenho acerca dos 3 padrões mais empregados de autenticação e autorização nesse contexto.
- Fizemos uma análise dos dados que gerou *insights* importantes a respeito do impacto desses padrões na aplicação.
- Concluímos que os resultados indicam que o padrão **descentralizado** é o mais eficiente.
- Avançamos o estado da arte da compreensão dos *tradeoffs* entre segurança e desempenho em microsserviços.

- Analisar o impacto de otimizações nos padrões de autenticação e autorização.
  - No padrão **centralizado**, mecanismos de *cache* que diminuem o tempo de resposta do serviço de autenticação.
  - No padrão **descentralizado**, mecanismos de compartilhamento dos dados de usuário, de forma a evitar a replicação.

REFERÊNCIAS

- COSTA, T. et al. Avaliação de desempenho de dois padrões de resiliência para microsserviços: Retry e circuit breaker. In: Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Porto Alegre, RS, Brasil: SBC, 2022. p. 517–530. ISSN 2177-9384. Available from Internet: <https://sol.sbc.org.br/index.php/sbrc/article/view/21194>.
- FOWLER, M. Microservices: a definition of this new architectural term. 2014. Available from Internet: <https://martinfowler.com/articles/microservices.html>
- MIANO, S. et al. A framework for ebpf-based network functions in an era of microservices. IEEE Transactions on Network and Service Management, v. 18, n. 1, p. 133–151, 2021.
- NASAB, A. R. et al. An empirical study of security practices for microservices systems. Journal of Systems and Software, Elsevier, v. 198, p. 111563, 2023.
- OWASP. Microservices Security Cheat Sheet. 2017. Available from Internet: <https://cheatsheetseries.owasp.org/cheatsheets/Microservices_Security_Cheat_Sheet.html>
- PEREIRA-VALE, A. et al. Security mechanisms used in microservices-based systems: A systematic mapping. In: 2019 XLV Latin American Computing Conference (CLEI). [S.l.: s.n.], 2019. p. 01–10.
- RICHARDSON, C. Event Sourcing Examples. 2024. Available from Internet: <https://github.com/cer/event-sourcing-examples>
- TRIARTONO, Z.; NEGARA, R. M.; SUSSI. Implementation of role-based access control on oauth 2.0 as authentication and authorization system. In: 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). [S.l.: s.n.], 2019. p. 259–263.
- YARYGINA, T.; BAGGE, A. H. Overcoming security challenges in microservice architectures. In: IEEE. 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). [S.l.], 2018. p. 11–20.

OBRIGADO!

Rafael Freitas Cardoso
Instituto de Informática — UFRGS
rfcardoso@inf.ufrgs.br