

EX-JUD: Uma Plataforma Blockchain para Automação, Auditoria e Rastreabilidade de Serviços Extrajudiciais no Tribunal de Justiça de Mato Grosso do Sul

Renato Gil Arruda Vieira¹, Dr. Marcelo Augusto Santos Turine²

¹Tribunal de Justiça de Mato Grosso do Sul (TJMS)
Campo Grande – MS – Brasil

²Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

renato.vieira@ufms.br, marcelo.turine@ufms.br

Abstract. *EX-JUD is a four-layer microservices architecture over permissioned Hyperledger Fabric for certifying and auditing extrajudicial acts, combining a Transactional Outbox pattern, client-side gateway load balancing, smart contracts with ABAC and double-spend prevention, and Byzantine consensus ($n \geq 3f + 1$). Via continuous telemetry over 33 containers, we show no resource exhaustion.*

Resumo. *A EX-JUD é uma arquitetura de microsserviços em quatro camadas sobre blockchain permissionada (Hyperledger Fabric) para auditoria de atos extrajudiciais, combinando Transactional Outbox, balanceamento client-side de gateways, contratos com ABAC e prevenção de gasto duplo, e consenso bizantino ($n \geq 3f + 1$). Por telemetria contínua nos 33 contêineres, não observamos esgotamento de recursos.*

1. Introdução

A transformação digital do Judiciário brasileiro [CNJ 2024a, CNJ 2024b] ainda não alcançou os serviços extrajudiciais, fiscalizados pelas Corregedorias-Gerais de Justiça [Brasil 1994, Benacchio 2022] e apoiados em sistemas centralizados como o SIG-EX do TJMS [STI/TJMS 2023]. Redes *blockchain* permissionadas oferecem consenso replicado e rastreabilidade auditável, mas protocolos BFT [Castro and Liskov 1999, Lamport et al. 1982] introduzem custos de endosso e comunicação – o *trade-off* que motiva este trabalho. Propusemos a EX-JUD [Vieira 2026], consórcio em *Hyperledger Fabric* [Androulaki et al. 2018] que integra CGJ e serventias, apresentando: (i) arquitetura em quatro camadas; (ii) balanceamento de *gateways*; e (iii) avaliação de desempenho.

2. Arquitetura em Camadas e Nós da Rede

A EX-JUD organiza seus microsserviços em quatro camadas (Figura 1). A *Camada de Interfaces* localiza, via *Service Discovery*, o nó validador de menor latência; a *Camada de Identidade* concentra a *API Proxy* e duas Autoridades Certificadoras (CA) segregadas por domínio (*ca.tjms*, *ca.cartorios*), evitando falsificação de certificados por chave de borda comprometida. A *Camada de Processamento* reúne a *API Core* e os *orderers* SmartBFT. A *Camada de Imutabilidade* hospeda *peers*, *chaincode* e *ledger*.

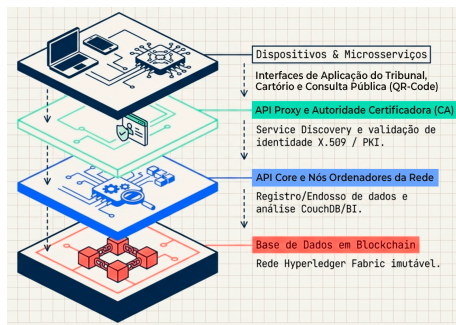


Figure 1. Fluxo EX-JUD.

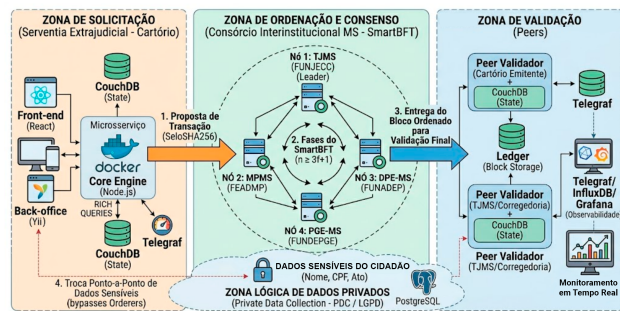


Figure 2. Zonas Transacionais.

A Figura 2 divide o sistema em três zonas: solicitação, ordenação/consenso (SmartBFT [Gao et al. 2019] entre TJMS, MPMS, DPE-MS e PGE-MS) e validação, com dados sigilosos em PDC (*Private Data Collection*). A arquitetura separa tarefas: a *API Core* (PHP/Yii/FrankenPHP) gere regras síncronas no PostgreSQL, enquanto a *API Proxy* (Node.js) conduz o fluxo no Couchdb. Ambas expõem 36 rotas via OpenAPI sobre gRPC; o proxy assina os dados e grava apenas o hash SHA-256 no ledger. A integração usa Transactional Outbox (ACID via worker). Os smart contracts (MintSelo, ConsumirSelo, RevogarSelo) rodam em Go via Chaincode-as-a-Service [Foundation 2025], garantindo o direito ao esquecimento com dados segregados.

3. Escalabilidade, Chaincode e Balanceamento de Gateways

A escalabilidade apoia-se em *peers* concorrentes (*goroutines*) e em *Channels* do Fabric por consórcio. A resiliência é reforçada por balanceamento *client-side*: o *DApp* do cartório reconfigura o *Gateway* da *API Proxy* (Figura 3) para nós parceiros se a infraestrutura central falhar, eliminando ponto único de falha.

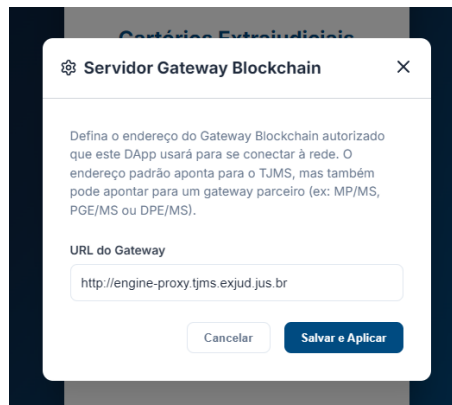


Figure 3. Alternância do Gateway no Portal do Cartório.

No *chaincode*, ConsumirSelo implementa ABAC e prevenção de gasto duplo (Código 1): valida a identidade MSP, verifica reuso do selo e trata dados sensíveis como *transient payloads*, distribuindo receita entre quatro fundos.

```
if clientMSPID == "TJMS-OrgMSP" || clientMSPID == "CNJ-OrgMSP" { return fmt.Errorf("ACL restrito: sem jurisdicao") }
if selo.Status == "CONSUMIDO" { return fmt.Errorf("selo ja utilizado") } // anti gasto duplo
if dadosPrivJSON, ok := transientMap["selo_priv_data"]; ok { _ = ctx.GetStub().PutPrivateData(CollPriv, SeloID, dadosPrivJSON) }
```

Listing 1. Trecho do chaincode ConsumirSelo: ACL, gasto duplo e PDC.

4. Avaliação de Desempenho e Monitoramento

Para suportar picos de 50 a 250 TPS sem bloqueios, adotou-se monitoramento contínuo via sidecars Telegraf e Grafana em 33 contêineres. Conforme a Tabela 1, o consumo de RAM e CPU manteve-se estável por nó, sem vazamento ou exaustão de E/S. A altura da blockchain evoluiu de 70 a 100 blocos, coerente com o custo do SmartBFT ante o Raft.

Table 1. Indicadores de desempenho observados em operação.

Métrica	Valor observado
RAM por nó Fabric (<i>peer/orderer</i>)	32–96 MiB
CPU por instância	0,01–0,015 núcleos
Virtualização (8 vCPUs, 33 contêineres)	≈24% CPU / 2,6–7,58 GB
Tráfego de rede entre <i>peers</i>	picos de até 3 kB/s
Descritores de arquivo abertos	50–125 (estável)

5. Conclusão e Trabalhos Futuros

A EX-JUD apresentou uma arquitetura *blockchain* com balanceamento de *gateways* e prevenção de gasto duplo, destacando-se de plataformas correlatas por adotar consenso bizantino. A solução garante não-repúdio e ancoragem sem exaurir recursos. Trabalhos futuros preveem testes geográficos de TPS, comparação SmartBFT/Raft e integração com sistemas de automação da justiça (eProc, SAJ e PJe).

Referências

Androulaki, E., Barger, A., Bortnikov, V., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. EuroSys '18, New York, NY, USA.

Benacchio, M. (2022). *Direito Notarial e Registral Contemporâneo: Governança, Compliance e Fiscalização*. Revista dos Tribunais, São Paulo, 2 edition.

Brasil (1994). Lei nº 8.935, de 18 de novembro de 1994. regulamenta o art. 236 da constituição federal, dispondo sobre serviços notariais e de registro. (lei dos cartórios).

Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, New Orleans, USA. USENIX Association.

CNJ (2024a). Plataforma digital do poder judiciário brasileiro (pdpj-br). Technical report, Brasília, Brasil.

CNJ (2024b). Programa justiça 4.0. Technical report, CNJ, Brasília, Brasil.

Foundation, H. (2025). Hyperledger fabric-x: The future of hyperledger fabric architecture.

Gao, S., Yu, T., Zhu, J., and Cai, W. (2019). T-pbft: An eigentrust-based practical byzantine fault tolerance consensus algorithm. *China Communications*, 16(12):111–123.

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401.

STI/TJMS (2023). *Manual do Sistema de Gerenciamento das Serventias Extrajudiciais (SIG-EX)*. Secretaria de Tecnologia da Informação, Campo Grande, MS.

Vieira, R. G. A. (2026). EX-JUD: Plataforma Blockchain para certificação e rastreabilidade de atos cartorários extrajudiciais no estado de mato grosso do sul. Dissertação de mestrado em computação aplicada, Faculdade de Computação, UFMS.