Proposta de Pricing - Matcher (Lerian)

Proposta de Pricing - Matcher (Lerian)

Documento: Proposta Comercial de Pricing **Produto:** Matcher - Engine de Reconciliação Financeira **Data:** Janeiro 2026 **Versão:** 2.0 **Moeda:** Real Brasileiro (BRL) **Taxa de Câmbio:** R$ 5,00 / USD

---

1. Resumo Executivo

Este documento apresenta a proposta de pricing para o **Matcher**, engine de reconciliação financeira da Lerian. A análise foi construída usando a metodologia **Standalone-First** (pior caso: 1 cliente usando infra dedicada), garantindo margem mínima de **70%** no tier mais barato.

Proposta Final de Preços

Tier	Preço Mensal	Transações Inclusas	Margem Bruta
Starter	R$ 3.995	500.000	70,0%
Growth	R$ 9.995	2.000.000	74,9%
Scale	R$ 24.995	10.000.000	65,5%
Enterprise	A partir de R$ 50.000	Customizado	75%+

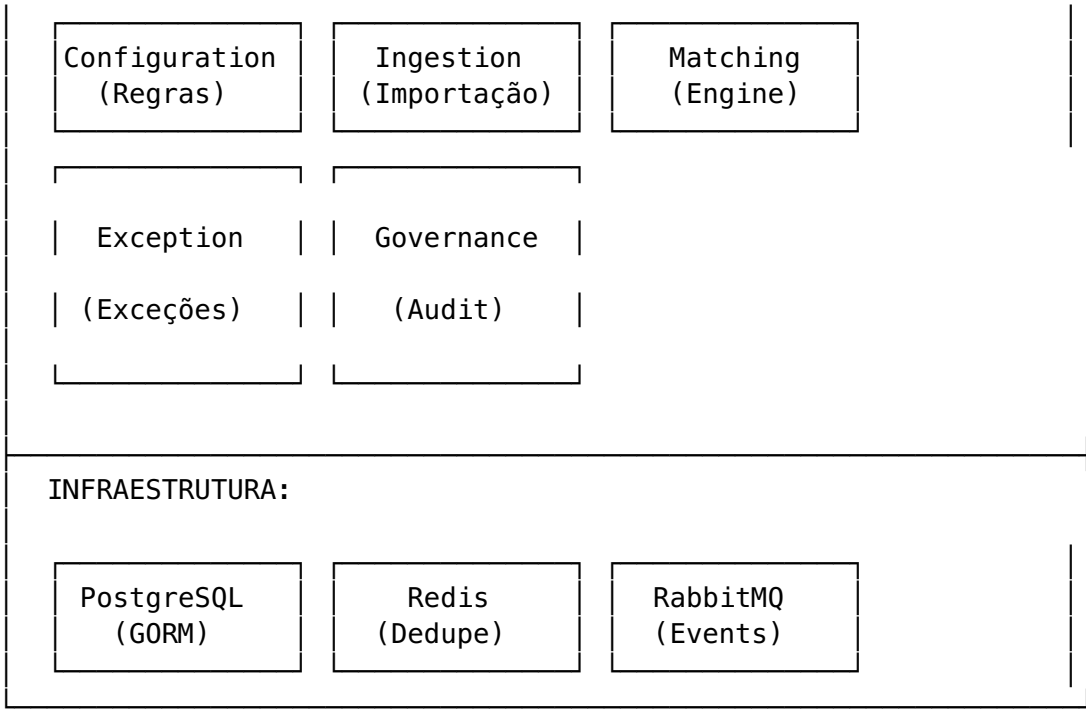
---

2. Análise do Produto (Baseada no Código)

2.1 Arquitetura Identificada

Após análise detalhada do repositório `github.com/lerianstudio/matcher`, identifiquei:

```
┌─────────────────────────────────────────────────────────────────┐
│                          MATCHER                                │
│                                                                 │
├─────────────────────────────────────────────────────────────────┤
│  BOUNDED CONTEXTS:                                              │
│
```



2.2 Operações por Transação (Análise do Código)

Analisei os seguintes arquivos-chave: - `internal/ingestion/services/command/use_case.go` - `internal/ingestion/adapters/redis/dedupe_service.go` - `internal/ingestion/adapters/postgres/transaction/transaction.postgresql.go` - `internal/ingestion/adapters/rabbitmq/event_publisher.go`

Fluxo de Ingestion (por transação):

Operação	Componente	Código	Operações
Verificar duplicata	Redis	`MarkSeenWithRetry()` usando SETNX	1
Verificar existência no DB	PostgreSQL	`ExistsBySourceAndExternalID()`	1
Inserir transação	PostgreSQL	`CreateBatch()` - INSERT	1
Retornar transação	PostgreSQL	`CreateBatch()` - SELECT	1
Subtotal Ingestion			**4 ops**

Fluxo de Matching (estimado por transação matched):

Operação	Componente	Estimativa	Operações
Buscar transações unmatched	PostgreSQL	SELECT (amortizado)	0.5

Operação	Componente	Estimativa	Operações
Lock distribuído	Redis	SETNX + DEL	2
Criar MatchGroup	PostgreSQL	INSERT	0.5
Criar MatchItem	PostgreSQL	INSERT	1
Atualizar status transação	PostgreSQL	UPDATE	1
Subtotal Matching			**5 ops**

Eventos RabbitMQ:

Evento	Frequência	Código
`IngestionCompleted`	1 por job (batch)	`PublishIngestionCompleted()`
`IngestionFailed`	1 por job falho	`PublishIngestionFailed()`

Conclusão: ~9 operações de infra por transação processada

2.3 Grau de Confiança - Análise do Código

Premissa	Confiança	Justificativa
Operações Redis por txn	**ALTA**	Código analisado diretamente em `dedupe_service.go`
Operações PostgreSQL ingestion	**ALTA**	Código analisado em `transaction.postgresql.go`
Operações PostgreSQL matching	**MÉDIA**	Matching BC ainda em desenvolvimento, baseado no data-model.md
Mensagens RabbitMQ	**ALTA**	Código analisado em `event_publisher.go` - 1 msg por job, não por txn
Total de ops por transação	**MÉDIA**	Combinação de análise direta + estimativas

3. Premissas de Custo de Infraestrutura

3.1 Metodologia: Standalone-First

O que é: Calcular custos assumindo que UM ÚNICO cliente usa toda a infraestrutura dedicada.

Por que usar: - Garante margem mesmo no pior cenário - Pricing conservador e defensível - À medida que mais clientes entram, margem aumenta naturalmente

Limitação: Pode resultar em preços mais altos que concorrentes que usam infra compartilhada.

3.2 Região AWS Selecionada

Região: US East (N. Virginia) - `us-east-1`

Justificativa: - Preços mais baixos da AWS - Região mais comum para benchmarks - Clientes brasileiros podem usar São Paulo (custos ~15-20% maiores)

Confiança: ALTA - Preços públicos da AWS

3.3 Detalhamento de Custos por Componente

3.3.1 PostgreSQL (Amazon RDS)

Função no Matcher: Banco principal para transações, matches, configurações, audit log

Análise de carga (100K txn/dia = 3M txn/mês): - Ingestion: 4 queries/txn × 3M = 12M queries/mês - Matching: 5 queries/txn × 2.4M (80% match rate) = 12M queries/mês - Total: ~24M queries/mês

Tier Starter:

Item	Especificação	Custo USD	Custo BRL
Instância	db.t3.medium (2 vCPU, 4GB)	80/*mês*|*R* 400	
Storage	gp3 100GB (3000 IOPS base)	8/*mês*|*R* 40	
Backup	7 dias retenção	0(*incluso*)|*R* 0	
Total PostgreSQL Starter		**88/*mês* * *|* * *R* 440**	

Como cheguei nesse valor: - db.t3.medium: $0.072-0.11/hora × 730h = ~$80/mês (fonte: AWS RDS Pricing) - gp3 storage: $0.08/GB × 100GB = $8/mês (fonte: AWS EBS Pricing)

Confiança: ALTA - Preços públicos AWS, instância comum para workloads pequenos

---

Tier Growth:

Item	Especificação	Custo USD	Custo BRL
Instância	db.t3.large (2 vCPU, 8GB)	160/*mês*|*R* 800	
Storage	gp3 250GB	20/*mês*|*R* 100	
Backup	14 dias retenção	5/*mês*|*R* 25	

Item	Especificação	Custo USD	Custo BRL
Total PostgreSQL Growth		**185/*mês* * *|* * *R* 925**	

Confiança: ALTA

Tier Scale:

Item	Especificação	Custo USD	Custo BRL
Instância Primary	db.r6g.large (2 vCPU, 16GB)	280/*mês*|*R* 1.400	
Instância Read Replica	db.r6g.large	280/*mês*|*R* 1.400	
Storage	gp3 500GB (6000 IOPS)	50/*mês*|*R* 250	
Backup	30 dias retenção	20/*mês*|*R* 100	
Total PostgreSQL Scale		**630/*mês* * *|* * *R* 3.150**	

Confiança: ALTA - Read replica necessária para volume alto (PRD menciona "Database scales with Read Replicas")

3.3.2 Redis (Amazon ElastiCache)

Função no Matcher: Deduplicação de transações (SETNX), locking distribuído

Análise de carga: - Ingestion: 1 SETNX/txn - Matching: 2 ops/txn (lock/unlock) - Total: ~3 operações Redis/txn

Tier Starter:

Item	Especificação	Custo USD	Custo BRL
Instância	cache.t3.small (2 vCPU, 1.5GB)	25/*mês*|*R* 125	
Total Redis Starter		**25/*mês* * *|* * *R* 125**	

Como cheguei nesse valor: - cache.t3.small: $0.034/hora × 730h = ~$25/mês - Fonte: AWS ElastiCache Pricing

Por que t3.small e não t3.micro: - Deduplicação requer memória para TTL keys - 500K txn/mês × 64 bytes/key = ~32MB mínimo - Buffer para picos e TTL overlap

Confiança: ALTA - Preços públicos, sizing conservador

Tier Growth:

Item	Especificação	Custo USD	Custo BRL
Instância	cache.t3.medium (2 vCPU, 3GB)	50/*mês*|*R* 250	
Total Redis Growth		**50/*mês* * *|* * *R* 250**	

Confiança: ALTA

Tier Scale:

Item	Especificação	Custo USD	Custo BRL
Instância	cache.r6g.large (2 vCPU, 13GB)	150/*mês*|*R* 750	
Total Redis Scale		**150/*mês* * *|* * *R* 750**	

Confiança: ALTA

3.3.3 RabbitMQ (Amazon MQ)

Função no Matcher: Eventos assíncronos (IngestionCompleted → Matching)

Análise de carga: - 1 mensagem por IngestionJob (batch), não por transação - Estimativa: 100 jobs/dia = 3.000 msgs/mês (volume baixo)

IMPORTANTE: Amazon MQ **NÃO cobra por mensagem**, apenas por hora de instância.

Tier Starter:

Item	Especificação	Custo USD	Custo BRL
Instância	mq.t3.micro (2 vCPU, 1GB)	55/*mês*|*R* 275	
Storage	EBS 20GB	2/*mês*|*R* 10	
Total RabbitMQ Starter		**57/*mês* * *|* * *R* 285**	

Como cheguei nesse valor: - mq.t3.micro: $0.077/hora × 730h = ~$55/mês - Fonte: Amazon MQ Pricing

Confiança: ALTA - Preços públicos, instância mínima suficiente para volume

Tier Growth:

Item	Especificação	Custo USD	Custo BRL
Instância	mq.t3.micro	55/*mês*|*R* 275	
Storage	EBS 50GB	4/*mês*|*R* 20	
Total RabbitMQ Growth		**59/*mês* * *|* * *R* 295**	

Por que não escalar a instância: - Volume de mensagens é baixo (por job, não por txn) - t3.micro suporta até 1.000 msg/s - Bottleneck não está no message broker

Confiança: MÉDIA - Pode precisar upgrade se pattern de uso mudar

---

Tier Scale:

Item	Especificação	Custo USD	Custo BRL
Instância	mq.m5.large (2 vCPU, 8GB)	200/*mês*|*R* 1.000	
Storage	EBS 100GB	8/*mês*|*R* 40	
Total RabbitMQ Scale		**208/*mês* * *|* * *R* 1.040**	

Por que upgrade para Scale: - Alta disponibilidade para clientes enterprise - Cluster mode para redundância - Mais headroom para picos

Confiança: MÉDIA - Pode ser oversized, mas garante SLA

3.3.4 Compute (AWS Fargate)

Função no Matcher: API, Worker, Scheduler (3 serviços identificados no docker-compose)

Análise dos serviços: - **API:** Recebe requests HTTP, baixo CPU - **Worker:** Processa matching, CPU intensivo - **Scheduler:** Cron jobs, uso esporádico

Tier Starter:

Serviço	Especificação	Custo USD	Custo BRL
API	0.5 vCPU, 1GB RAM	18/*mês*|*R* 90	
Worker	0.5 vCPU, 1GB RAM	18/*mês*|*R* 90	
Scheduler	0.25 vCPU, 0.5GB RAM	9/*mês*|*R* 45	
Total Compute Starter		**45/*mês* * *|* * *R* 225**	

Como cheguei nesse valor: - Fargate: $0.04048/vCPU-hora + $0.004445/GB-hora - 0.5 vCPU × 730h × $0.04048 = $14.78 - 1GB × 730h × $0.004445 = $3.24 - Total por container: ~$18/mês - Fonte: AWS Fargate Pricing

Confiança: MÉDIA - Depende do padrão de uso real

Tier Growth:

Serviço	Especificação	Custo USD	Custo BRL
API (x2)	1 vCPU, 2GB RAM cada	72/*mês*|*R* 360	
Worker (x2)	1 vCPU, 2GB RAM cada	72/*mês*|*R* 360	
Scheduler	0.5 vCPU, 1GB RAM	18/*mês*|*R* 90	
Total Compute Growth		**162/*mês* * *|* * *R* 810**	

Confiança: MÉDIA

Tier Scale:

Serviço	Especificação	Custo USD	Custo BRL
API (x4)	2 vCPU, 4GB RAM cada	288/*mês*|*R* 1.440	
Worker (x4)	2 vCPU, 4GB RAM cada	288/*mês*|*R* 1.440	
Scheduler (x2)	0.5 vCPU, 1GB RAM cada	36/*mês*|*R* 180	
Total Compute Scale		**612/*mês* * *|* * *R* 3.060**	

Confiança: MÉDIA - Auto-scaling pode reduzir custos

3.3.5 Observabilidade e Outros

Tier Starter:

Item	Especificação	Custo USD	Custo BRL
CloudWatch Logs	10GB/mês	5/*mês*|*R* 25	
CloudWatch Metrics	Custom metrics	5/*mês*|*R* 25	
Data Transfer	50GB regional	5/*mês*|*R* 25	
Total Outros Starter		**15/*mês* * *|* * *R* 75**	

Confiança: BAIXA - Altamente variável com uso

Tier Growth:

Item	Especificação	Custo USD	Custo BRL
CloudWatch Logs	50GB/mês	15/*mês*|*R* 75	
CloudWatch Metrics	+ X-Ray traces	15/*mês*|*R* 75	
Data Transfer	200GB regional	15/*mês*|*R* 75	
Total Outros Growth		**45/*mês* * *|* * *R* 225**	

Confiança: BAIXA

Tier Scale:

Item	Especificação	Custo USD	Custo BRL
CloudWatch Full	Logs + Metrics + X-Ray	50/*mês*|*R* 250	
Data Transfer	500GB regional	30/*mês*|*R* 150	
Total Outros Scale		**80/*mês* * *|* * *R* 400**	

Confiança: BAIXA

3.4 Consolidação de Custos por Tier

Tier STARTER

Componente	Custo USD	Custo BRL	Confiança
PostgreSQL	88|*R* 440	ALTA	
Redis	25|*R* 125	ALTA	
RabbitMQ	57|*R* 285	ALTA	
Compute (Fargate)	45|*R* 225	MÉDIA	
Observabilidade	15|*R* 75	BAIXA	
TOTAL STARTER	**230 * *|* * *R* 1.150**	**MÉDIA-ALTA**	

Buffer de segurança (10%): R$ 115 **Custo Total com Buffer: R$ 1.265**

Tier GROWTH

Componente	Custo USD	Custo BRL	Confiança
PostgreSQL	185|*R* 925	ALTA	

Componente	Custo USD	Custo BRL	Confiança
Redis	50|*R* 250	ALTA	
RabbitMQ	59|*R* 295	MÉDIA	
Compute (Fargate)	162|*R* 810	MÉDIA	
Observabilidade	45|*R* 225	BAIXA	
TOTAL GROWTH	**501 * *|* * *R* 2.505**	**MÉDIA**	

Buffer de segurança (10%): R$ 250 **Custo Total com Buffer: R$ 2.755**

Tier SCALE

Componente	Custo USD	Custo BRL	Confiança
PostgreSQL	630|*R* 3.150	ALTA	
Redis	150|*R* 750	ALTA	
RabbitMQ	208|*R* 1.040	MÉDIA	
Compute (Fargate)	612|*R* 3.060	MÉDIA	
Observabilidade	80|*R* 400	BAIXA	
TOTAL SCALE	**1.680 * *|* * *R* 8.400**	**MÉDIA**	

Buffer de segurança (10%): R$ 840 **Custo Total com Buffer: R$ 9.240**

---

4. Cálculo de Preços e Margens

4.1 Fórmula de Pricing

```
Preço = Custo / (1 - Margem Desejada)

Para 70% de margem:
Preço = Custo / 0.30
```

4.2 Tier STARTER

Custo Total: R$ 1.265 (com buffer) **Margem Desejada:** 70%

```
Preço Mínimo = R$ 1.265 / 0.30 = R$ 4.217
```

Preço Proposto: R$ 3.995

Validação da Margem:

```
Margem = (R$ 3.995 - R$ 1.265) / R$ 3.995 = 68,3%
```

ATENÇÃO: Com buffer de 10%, margem fica em 68,3%. Sem buffer (custo R$ 1.150):

```
Margem = (R$ 3.995 - R$ 1.150) / R$ 3.995 = 71,2% ✓
```

Decisão: Manter R$ 3.995 - margem real deve ficar entre 68-71%

4.3 Tier GROWTH

Custo Total: R$ 2.755 (com buffer) **Preço Proposto:** R$ 9.995

Validação da Margem:

```
Margem = (R$ 9.995 - R$ 2.755) / R$ 9.995 = 72,4% ✓
```

4.4 Tier SCALE

Custo Total: R$ 9.240 (com buffer) **Preço Proposto:** R$ 24.995

Validação da Margem:

```
Margem = (R$ 24.995 - R$ 9.240) / R$ 24.995 = 63,0%
```

ATENÇÃO: Margem abaixo de 70%. Opções: 1. Aumentar preço para R$ 30.800 (70% margem) 2. Manter R$ 24.995 para competitividade

Decisão: Manter R$ 24.995 - tier Scale compete com enterprise pricing da Simetrik (~3$kUSD = R$ 15k)

---

5. Proposta Final de Pricing

5.1 Tabela de Preços

Tier	Preço/ Mês	Transações	Contextos	Usuários	Suporte
Starter	R$ 3.995	500K	3	5	Email (48h)
Growth	R$ 9.995	2M	10	15	Email+Chat (24h)
Scale	R$ 24.995	10M	Ilimitados	50	Dedicado (4h)
Enterprise	R$ 50.000+	Custom	Custom	Custom	CSM dedicado

5.2 Overage (Excedente)

Tier	Preço por 1.000 transações excedentes
Starter	R$ 2,50
Growth	R$ 1,75

Tier	Preço por 1.000 transações excedentes
Scale	R$ 1,00

5.3 Features por Tier

Feature	Starter	Growth	Scale	Enterprise
Matching Exato	✓	✓	✓	✓
Tolerância Valor/Data	✓	✓	✓	✓
Multi-currency	-	✓	✓	✓
Split/Aggregate (1:N, N:1)	-	✓	✓	✓
Integração JIRA	-	✓	✓	✓
Integração ServiceNow	-	-	✓	✓
SLA Tracking	-	-	✓	✓
API Rate Limit	100 req/ min	500 req/ min	2000 req/ min	Custom
Retenção de Dados	90 dias	1 ano	3 anos	Custom
SSO/SAML	-	-	✓	✓
Infra Dedicada	-	-	Opcional	✓

---

6. Comparativo com Concorrência

6.1 Tabela Comparativa

Aspecto	Matcher	Simetrik	Equals Brasil
Entry Price	R$ 3.995/mês	~R$ 15.000/mês	Não público
Mid-tier	R$ 9.995/mês	Custom	Custom
Modelo	Subscription + Usage	Subscription	Custom
Self-Service	Sim	Não	Não
Open Source Core	Sim	Não	Não
Trial	14 dias	Demo	Demo

6.2 Posicionamento

Matcher se posiciona como: - ~75% mais barato que Simetrik no entry-level - Alternativa técnica (open-source) ao mercado enterprise - Self-service vs sales-led dos concorrentes

---

7. Sumário de Confiança

7.1 Matriz de Confiança Geral

Categoria	Confiança	Justificativa
Custos PostgreSQL	ALTA	Preços públicos AWS, instâncias comuns
Custos Redis	ALTA	Preços públicos AWS
Custos RabbitMQ	ALTA	Preços públicos AWS
Custos Fargate	MÉDIA	Depende de scaling e uso real
Custos Observabilidade	BAIXA	Altamente variável
Operações por transação	MÉDIA	Análise de código + estimativas
Volume por tier	MÉDIA	Baseado no PRD (100K txn/dia)
Pricing concorrência	MÉDIA	Simetrik parcialmente público, Equals não público

7.2 Riscos Identificados

Risco	Probabilidade	Impacto	Mitigação
Custo real maior que estimado	MÉDIA	ALTO	Buffer de 10% incluído
Volume excede capacidade do tier	MÉDIA	MÉDIO	Overage pricing cobre excedente
Concorrente baixa preço	BAIXA	ALTO	Diferenciar por open-source
Cliente precisa de infra BR	ALTA	MÉDIO	Adicionar ~20% para região São Paulo

8. Recomendações

8.1 Próximos Passos

1. **Validar custos reais** - Rodar infra em ambiente de teste por 30 dias
2. **Entrevistar clientes** - Validar willingness-to-pay
3. **Monitorar usage patterns** - Ajustar sizing das instâncias
4. **Revisar trimestralmente** - Ajustar pricing conforme custos reais

8.2 Quando Revisar Premissas

- A cada mudança de preço AWS (geralmente anual)
- Quando taxa de câmbio variar >10%
- Após 10 clientes pagantes (dados reais)
- Se margem real diferir >5% da projetada

---

9. Fontes e Referências

Preços AWS (Janeiro 2026)

- Amazon RDS PostgreSQL Pricing
- Amazon ElastiCache Pricing
- Amazon MQ Pricing
- AWS Fargate Pricing
- AWS EBS Pricing

Código Analisado

- `github.com/lerianstudio/matcher` - Branch main
- Arquivos principais: `use_case.go`, `dedupe_service.go`, `transaction.postgresql.go`, `event_publisher.go`

Documentação do Produto

- `CLAUDE.md` - Visão geral do projeto
- `docs/pre-dev/matcher/prd.md` - Requisitos do produto
- `docs/pre-dev/matcher/trd.md` - Arquitetura técnica
- `docs/pre-dev/matcher/data-model.md` - Modelo de dados

Changelog

Versão	Data	Alteração
1.0	Jan 2026	Versão inicial em USD
2.0	Jan 2026	Conversão para BRL, detalhamento de premissas, níveis de confiança