Proposta de Arquitetura de Microsserviços do *PharmaSys*

A arquitetura proposta para o *PharmaSys* adota o paradigma de **microsserviços orientados a eventos (Event-Driven Architecture)**, com comunicação híbrida entre **APIs REST** e **mensageria assíncrona**. Essa abordagem garante escalabilidade, isolamento de falhas, rastreabilidade e aderência às Boas Práticas de Fabricação (GMP) exigidas pela ANVISA.

5.1 Estrutura Geral

O sistema é composto por seis microsserviços principais, cada um responsável por um domínio específico do negócio farmacêutico:

Microsserviço	Função Principal
Auth Service	Gerenciamento de autenticação (OAuth2, JWT) e autorização baseada em papéis (RBAC).
Produção Service	Controle de Ordens de Produção (OP), registro de etapas e criação de lotes.
CQ Service	Registro de amostras e laudos de Controle de Qualidade, bloqueio e liberação de lotes.
Estoque Service	Movimentação de insumos e produtos, aplicação da regra FEFO, expedição e devoluções.
Relatórios Service	Consolidação de dados e geração de relatórios de rastreabilidade.
Gateway API	Ponto único de acesso, roteamento de requisições e validação de tokens de segurança.

Cada microsserviço possui seu **banco de dados independente (PostgreSQL)**, garantindo isolamento e autonomia das operações. A comunicação assíncrona é realizada por meio do **RabbitMQ**, enquanto operações síncronas são expostas via **FastAPI (REST)**.

5.2 Comunicação e Integração

A integração entre serviços ocorre através de **eventos de domínio, que nada mais são do que** ações reais do processo produtivo:

Evento	Origem → Destino	Descrição
`lote.criado`	Produção → CQ	Indica a criação de um novo lote e aciona a coleta de amostra.
`laudo.emitido`	CQ → Estoque	Comunica a aprovação ou reprovação do lote.
`desvio.aberto`	CQ → Produção	Sinaliza um desvio de processo, bloqueando etapas futuras da OP.
`expedicao.confirmada`	Estoque → Relatórios	Confirma o envio de um lote aprovado ao destino final.

Esses eventos são publicados e consumidos de forma assíncrona, assegurando **consistência eventual** e **baixo acoplamento** entre módulos.

Cada serviço expõe endpoints REST versionados e documentados via **OpenAPI**, com suporte a *Idempotency-Key* e respostas padronizadas.

5.3 Regras de Negócio e Fluxos Operacionais

A arquitetura reflete as principais regras de negócio do setor farmacêutico:

1. **Quarentena obrigatória –** todo lote nasce em quarentena e só é liberado após laudo aprovado.
2. **FEFO (First Expired, First Out) –** o sistema prioriza a utilização de produtos com validade mais próxima.
3. **Controle de desvios –** ao detectar um desvio, o *CQ Service* bloqueia automaticamente a OP correspondente.
4. **Laudo obrigatório para expedição –** o *Estoque Service* não permite expedições sem laudo aprovado.

Fluxo resumido:

```
[Produção] → cria lote → evento (lote.criado)
[CQ] → realiza análise → evento (laudo.emitido)
[Estoque] → libera expedição → evento (expedicao.confirmada)
[Relatórios] → consolida rastreabilidade
```

Esse fluxo garante rastreabilidade ponta a ponta, desde a fabricação até a entrega ao cliente final.

5.4 Segurança e Conformidade

O *PharmaSys* utiliza o **Keycloak** como provedor de identidade, implementando autenticação **OAuth2 com tokens JWT** e autorização baseada em papéis. Os principais papéis definidos são:

- *Administrador*: acesso total ao sistema;
- *Gestor de Produção*: criação de ordens e lotes;
- *Analista CQ*: emissão de laudos e controle de desvios;
- *Estoquista*: movimentação e expedição de produtos;
- *Auditor*: acesso somente-leitura para fins de rastreabilidade e conformidade.

Cada operação crítica é registrada em um **log de auditoria imutável**, armazenado no serviço de **Monitoramento e Auditoria** (stack ELK – Elasticsearch, Logstash e Kibana), conforme exigido pelas normas da ANVISA e GMP.

5.5 Tecnologias e Ferramentas

Camada	Tecnologias Propostas
Backend	Python (FastAPI), SQLAlchemy, Alembic
Banco de Dados	PostgreSQL (transacional), Redis (cache), Elasticsearch (logs)
Mensageria	RabbitMQ (fila de eventos), com *dead-letter queues*
Autenticação	Keycloak (OAuth2 / JWT)
Frontend	React ou Next.js (opcional para interface)
DevOps	Docker, Docker Compose e Kubernetes (deploy produtivo)
CI/CD	GitHub Actions (lint, testes, build, push de imagens)
Monitoramento	Prometheus, Grafana, OpenTelemetry

5.6 Benefícios da Arquitetura

A adoção de microsserviços traz benefícios diretos para o *PharmaSys*:

- **Escalabilidade independente**: cada módulo cresce conforme a demanda.
- **Alta disponibilidade**: falhas isoladas não comprometem o sistema como um todo.
- **Manutenibilidade**: atualizações e correções são aplicadas por serviço.
- **Conformidade regulatória**: logs imutáveis e rastreabilidade integral de eventos.

- **Sustentabilidade**: a automação e o controle de perdas contribuem para os ODS 3, 9 e 12.

5.7 Estrutura de Implementação (Repositório)

Organização sugerida do projeto no GitHub:

```
/pharmasys
  /gateway
  /services
    /auth
    /producao
    /cq
    /estoque
    /relatorios
  /deploy
    /docker-compose.yml
    /helm
  /docs
    /apis (OpenAPI)
    /events (schemas JSON)
```

Cada pasta de serviço contém seu próprio *Dockerfile*, migrações (Alembic), testes automatizados e documentação OpenAPI.

5.8 Considerações Finais

A arquitetura de microsserviços proposta proporciona uma base sólida para o desenvolvimento de um sistema farmacêutico moderno, seguro e sustentável.

O *PharmaSys*, ao adotar comunicação orientada a eventos e mecanismos robustos de rastreabilidade, se torna capaz de atender às exigências regulatórias e operacionais da indústria farmacêutica, ao mesmo tempo em que promove inovação e eficiência de processos.