Plataforma Educacional com Pipeline CI/CD, Docker e Kubernetes

Título do Projeto

Plataforma Educacional com Pipeline CI/CD, Docker e Kubernetes

Objetivo

Evoluir a **Plataforma Educacional Distribuída** desenvolvida no módulo anterior, transformando-a em um **ecossistema DevOps completo**, com automação de build, testes, integração, entrega e orquestração em ambiente Kubernetes.

O trabalho tem como finalidade aplicar, de forma prática, os conceitos de **Git/GitHub**, **Docker**, **GitHub Actions**, **Kubernetes** e **Cultura DevOps**, preparando o sistema para rodar em ambientes reais com escalabilidade, resiliência e processos automatizados de entrega contínua.

Descrição Geral

- **Controle de versão estruturado** em GitHub, aplicando branching model adequado e uso de Pull Requests.
- **Containerização** de todos os microsserviços, cada um com seu próprio Dockerfile e práticas recomendadas (multi-stage builds, camadas otimizadas, variáveis de ambiente, etc.).
- **Orquestração em Kubernetes**, com cada API rodando como um Deployment isolado, configurada via YAML (Deployments, Services, ConfigMaps, Secrets).
- **Pipeline CI/CD com GitHub Actions**, incluindo:
    - Build e testes automatizados.
    - Lint e análise estática de código.
    - Deploy automático em das imagens no Docker Hub
- **Observabilidade e resiliência**, com restart policies, logs e métricas.
- **Automação de provisionamento local** via docker-compose, permitindo que qualquer desenvolvedor rode o ecossistema rapidamente em sua máquina.

Componentes do Sistema

Os mesmos **Bounded Contexts** e APIs definidos no módulo anterior:

- **Auth API**: autenticação, cadastro de usuários e emissão de JWT.
- **Conteúdo API**: cadastro/consulta de cursos e aulas.
- **Alunos API**: matrícula, progresso, certificados.
- **Pagamentos API**: processamento e eventos de pagamento.
- **BFF (Backend for Frontend)**: orquestração de chamadas para simplificar o consumo pelo front-end.

Evolução agora exigida: cada serviço deve estar conteinerizado, versionado, integrado ao pipeline e implantado em cluster Kubernetes.

Requisitos Técnicos

- Linguagem: C# com .NET 8 ou superior.
- Banco de Dados: SQL Server (prod), SQLite com seed para testes.
- Containerização: Docker, com Dockerfiles individuais por serviço.
- Orquestração: Kubernetes (pode ser Kind/Minikube local).
- Pipeline: GitHub Actions, com workflows para build, testes, lint, deploy.
- Resiliência: Retry, Circuit Breaker (Polly), health checks nativos do Kubernetes.
- Documentação: Swagger ativo em cada API + README.md consolidando o ambiente DevOps.

Critérios de Sucesso

- Cada serviço roda de forma **isolada em containers** e sob **Kubernetes**.
- **Pipeline CI/CD** executa automaticamente build, testes e deploy da imagem.
- **Fluxos de negócio** (login, matrícula, pagamento, certificado) funcionam integralmente no cluster.
- **Resiliência comprovada**: falhas em um serviço (ex: pagamento) não derrubam o ecossistema.
- **Observabilidade**: health checks acessíveis.
- **Documentação clara** (README + Swagger) permitindo reprodutibilidade do ambiente.

Prazos e Tipo de Entrega

- **Tipo de Entrega:** Projeto para desenvolvimento **INDIVIDUAL**
- Início: 03/11/2025
- Primeira Entrega: 24/11/2025
- Entrega Final: 22/12/2025

Instruções Importantes

- Todos os serviços devem ser **containerizados** e configurados para rodar em **Kubernetes**.
- O pipeline CI/CD deve ser **reprodutível e validado** realizando Deploy no Docker Hub.
- Os **manifests YAML** devem contemplar: Deployment, Service, ConfigMap, Secret e liveness/readiness probes.
- O **README.md** deve explicar como rodar o ambiente localmente via Docker Compose e como subir o cluster Kubernetes.
- O **FEEDBACK.md** deve consolidar interações e melhorias, como no módulo anterior.

Entrega

Repositório no GitHub:

- O código deve ser versionado e entregue através de um repositório público no Github e dentro dos padrões especificados em: https://github.com/desenvolvedor-io/template-repositorio-mba

Documentação:

- O README.md deve seguir as diretivas e padrões informados na documentação do projeto referência.

- Incluir um arquivo FEEDBACK.md no repositório onde os feedbacks serão consolidados, o instrutor fará um PR no repositório atualizando este arquivo.

Matriz de avaliação

Os projetos serão avaliados e receberão uma nota de 0 até 10 considerando os critérios a seguir:

Critério	Peso	Comentários
Funcionalidades DevOps	30%	Pipeline CI/CD funcionando, deploy em Docker Hub
Qualidade do Código	30%	Dockerfiles bem estruturados, YAMLs claros, código limpo e versionado
Integração Kubernetes	10%	Manifests completos e funcionamento real dos serviços no cluster.
Observabilidade	10%	Health checks, logs, métricas básicas implementadas.
Documentação	10%	README.md e Swagger completos e atualizados.
Resolução de Feedbacks	10%	Considere como o aluno ou grupo abordou os feedbacks da revisão de código.