Projeto Final: Plataforma de E-Commerce Distribuída "DistriCommerce"

O **DistriCommerce** será um sistema de e-commerce distribuído que aborda todos os tópicos do curso, utilizando **Java + Spring Boot** no backend e **React/Angular + Thymeleaf** no frontend. O sistema será escalável, tolerante a falhas e implantado em **Docker/Kubernetes**, com integração a **AWS/Azure** para computação em nuvem e borda.

IV **por ILDO VIEIRA**

Arquitetura do Sistema (Unidade I)

O projeto seguirá uma **arquitetura de microservices** com os seguintes componentes:

1. Frontend (Cliente)

- **Tecnologia**: React (ou JavaFX para desktop)
- **Comunicação**: REST (Spring Web) + WebSocket (notificações em tempo real)
- **Funcionalidades**: Catálogo de produtos, Carrinho de compras, Checkout e pagamento, Dashboard de monitoramento (Grafana embutido)

2. Backend (Serviços Distribuídos)

Microservice	Tecnologia	Descrição
API Gateway	Spring Cloud Gateway	Roteamento e balanceamento de carga
Serviço de Produtos	Spring Boot + MongoDB	CRUD de produtos, busca e catálogo
Serviço de Pedidos	Spring Boot + PostgreSQL	Processamento de pedidos e histórico
Serviço de Pagamentos	Spring Boot + RabbitMQ	Integração com gateways de pagamento (assíncrono)
Serviço de Usuários	Spring Boot + JWT	Autenticação e perfil do cliente
Serviço de Recomendações	gRPC + ML (Python)	Sistema de recomendação em tempo real
Serviço de Logística	Kafka + Spring Cloud Stream	Rastreamento de entregas (IoT)

3. Infraestrutura

- **Docker** para conteinerização
- **Kubernetes** (minikube para desenvolvimento, AWS EKS/AKS em produção)
- **Prometheus + Grafana** para monitoramento
- **ELK Stack** (Elasticsearch, Logstash, Kibana) para logs
- **RabbitMQ/Kafka** para mensageria assíncrona
- **Terraform** para provisionamento na nuvem (AWS/Azure)

Funcionalidades por Unidade do Curso - Parte 1

Unidade I - Fundamentos de Sistemas Distribuídos

Introdução a Sistemas Distribuídos

O e-commerce é um exemplo clássico de sistema distribuído (alta disponibilidade, escalabilidade).

Requisitos de Projeto

- Escalabilidade horizontal (Kubernetes)
- Tolerância a falhas (Circuit Breaker, Retry Policies)

Modelos de Arquitetura

- Microservices (Spring Boot)
- Cliente-Servidor (Frontend/Backend)
- Peer-to-Peer (Cache distribuído com Redis)

Modelos de Interação

- Síncrona (REST/gRPC)
- Assíncrona (RabbitMQ/Kafka)

Modelos de Falha

- Simulação de falhas (Chaos Engineering)
- Detecção via Health Checks (Spring Actuator)

Funcionalidades por Unidade do Curso - Parte 2

Unidade II - Padrões e Tecnologias

Middleware

- gRPC para comunicação interna entre serviços.
- REST para APIs públicas.

Padrões de Invocação

- RPC (gRPC para recomendação de produtos)
- Mensageria (Kafka para logística)

Resiliência

- Circuit Breaker (Resilience4J)
- Retry em falhas de pagamento.

Computação em Nuvem/Borda

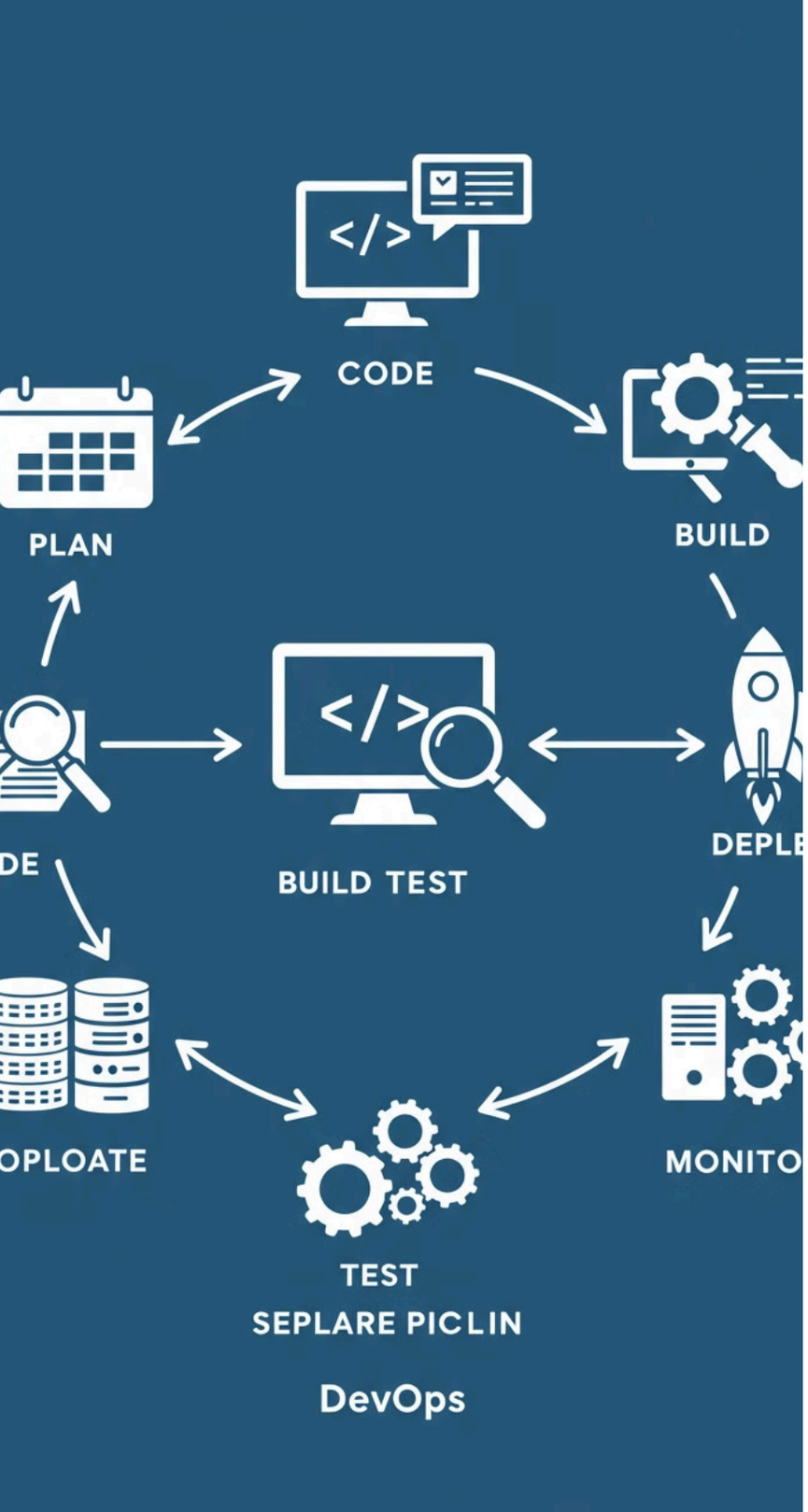
- Implantação na AWS/Azure (IaaS/PaaS)
- Edge Computing para cache regional (Cloudflare)

IoT

- Rastreamento de entregas via MQTT (sensores IoT em caminhões)

Virtualização

- Docker para containers
- Kubernetes para orquestração



Funcionalidades por Unidade do Curso - Parte 3

Unidade III - Desenvolvimento e Implantação

Configuração de Ambientes

Terraform para AWS/Azure

Ansible para configuração automatizada

CI/CD

GitHub Actions/GitLab CI

Deploy contínuo no Kubernetes

Avaliação

Testes de carga (JMeter)

Monitoramento (Prometheus)

Métricas de latência/throughput (Grafana)

Plano de Desenvolvimento (8 Semanas)

Semanas 1-2

- Definir requisitos
- Configurar Docker/K8s
- Criar serviços básicos (Produtos, Usuários)

Semanas 3-4

- Implementar API Gateway
- Integrar RabbitMQ/Kafka
- Criar frontend básico

Semanas 5-6

- Circuit Breaker e Resiliência
- Sistema de pagamento assíncrono
- Logística com IoT (MQTT)

Semana 7

- CI/CD com GitHub Actions
- Monitoramento (Prometheus/Grafana)
- Testes de carga

Semana 8

- Documentação
- Ajustes finais
- Vídeo de demonstração

Tecnologias Utilizadas

Categoria	Tecnologias
Backend	Java 17+, Spring Boot, Spring Cloud, gRPC
Frontend	React (ou JavaFX)
Banco de Dados	MongoDB, PostgreSQL, Redis
Mensageria	RabbitMQ, Apache Kafka
DevOps	Docker, Kubernetes, Terraform, GitHub Actions
Cloud	AWS (ECR, EKS, S3) ou Azure (AKS, Blob Storage)
Monitoramento	Prometheus, Grafana, ELK Stack

Como Executar o Projeto

Pré-requisitos

- Docker + Kubernetes (minikube)
- Java 17+
- Node.js (para frontend React)

Clonar repositório

```
git clone https://github.com/seu-usuario/districommerce.git
cd districommerce
```

Subir containers

```
docker-compose up -d
```

Acessar frontend

```
http://localhost:3000
```

Conclusão

O **DistriCommerce** é um projeto completo que abrange **todos os tópicos do curso**, desde fundamentos de sistemas distribuídos até implantação em nuvem com Kubernetes. Ele demonstra:

Escalabilidade

(K8s)

Tolerância a Falhas

(Circuit Breaker)

Comunicação Síncrona/Assíncrona

(REST, gRPC, Kafka)

IoT

(Rastreamento de entregas)

Monitoramento

(Prometheus/Grafana)

Tempo estimado: 2 meses (8 semanas) com uma equipe de 3-5 desenvolvedores.

Próximos passos: Implementar recomendações com IA (Python + gRPC) e expandir para mobile (Flutter).

Dúvidas? Posso detalhar qualquer parte do projeto! 🚀