

Conteúdo Programático

Especialista Microsserviços

Nível 1 - Arquitetura de Software e Microsserviços

1. Boas-vindas e arquitetura de software
1.1. Boas-vindas à formação
1.2. Alinhando expectativas sobre as aulas e projetos
1.3. Por que arquitetura é importante?
1.4. O que é arquitetura de software?
1.5. As leis da arquitetura de software
1.6. Responsabilidades e perfil do arquiteto de software
1.7. O que é arquitetura de solução?
1.8. O que é design de software (software design)?
1.9. O que é design de código (code design)?
1.10. Estilos arquiteturais, padrões arquiteturais e padrões de projeto
1.11. Diagramas arquiteturais formais (UML, C4 Model) e ad hoc
1.12. Documentações arquiteturais: Design Docs e ADRs
1.13. O que é System Design e System Design Interview?
1.14. Desafio: Refletindo sobre características arquiteturais

2. Arquitetura monolítica e microsserviços
2.1. Conhecendo a arquitetura monolítica e os monólitos clássicos
2.2. Organização de equipes e a Lei de Conway
2.3. Arquitetura monolítica não é legado!
2.4. Pesadelo dos monólitos: quando tudo sai do controle
2.5. Conhecendo a arquitetura de microsserviços
2.6. Microsserviços possuem seus próprios estados
2.7. Microsserviços se comunicam via rede em uma arquitetura distribuída
2.8. Microsserviços são autônomos e independentes
2.9. Microsserviços são componentes pequenos e coesos
2.10. Microsserviços possuem flexibilidade tecnológica
2.11. Microsserviços são implantados de forma independente
2.12. Microsserviços são resilientes e tolerantes a falhas
2.13. Microsserviços são escaláveis de forma independente

3. Práticas e tecnologias facilitadoras para microsserviços
3.1. Práticas e tecnologias facilitadoras
3.2. Times autônomos e estrutura descentralizada
3.3. Manobra inversa da Lei de Conway
3.4. Times pequenos e o conceito de Two Pizza Team
3.5. Cultura de ownership e a mentalidade de produtos (e não projetos)
3.6. Team Topologies e os times em organizações distribuídas
3.7. Cultura DevOps
3.8. Testes automatizados e Pirâmide de Testes
3.9. Containers e orquestração
3.10. Cloud Computing
3.11. Microservices Patterns: não reinvente a roda
3.12. Exercício: Microservices Patterns
3.13. Microservice Chassis com Spring Boot e Spring Cloud

3.14. Ecossistema de projetos da CNCF

4. Desafios da arquitetura de microsserviços
4.1. Conhecendo os principais desafios e desvantagens
4.2. Complexidade na modelagem de serviços e o risco do monólito distribuído
4.3. Dificuldade na refatoração de fronteiras dos serviços
4.4. Complexidade da computação distribuída
4.5. Teorema CAP e consistência eventual
4.6. Desafio: Teorema CAP
4.7. Aumento de custos de infraestrutura e operação
4.8. Riscos de segurança
4.9. Conformidade com regulamentações de privacidade (LGPD)
4.10. Hype Driven Development e Technology Sprawl

5. Transição para arquitetura de microsserviços
5.1. Fuja de microsserviços: quando não vale a pena a adoção
5.2. Abrace os microsserviços: quando realmente faz sentido
5.3. Monólito Modular: alternativa aos microsserviços
5.4. Estratégias Greenfield: Microservices First e Monolith First
5.5. Estratégias Brownfield: Big Bang, Modularização Incremental e Strangler Fig Pattern
5.6. Anti-patterns organizacionais: Scattershot Adoption e Red Flag Law
5.7. Desafio: Escolhendo a arquitetura de um sistema

6. Padrões de comunicação
6.1. Tipos de comunicação entre microsserviços
6.2. Introdução ao Request-response Pattern
6.3. Comunicação bloqueante e não bloqueante
6.4. O que são contratos?
6.5. Serialização
6.6. Acoplamento e microsserviços
6.7. Messaging Patterns e mensageria
6.8. Producer-consumer Pattern
6.9. Publisher-subscriber Pattern

7. Modelagem de microsserviços
7.1. Introdução à modelagem de microsserviços
7.2. Baixo acoplamento: Reduza as dependências entre serviços
7.3. Acoplamento por domínio
7.4. Acoplamento por repasse de dados
7.5. Acoplamento por recurso comum
7.6. Acoplamento por estrutura interna e Information hiding
7.7. Alta coesão, Single Responsibility Principle e Common Closure Principle
7.8. Lei de Constantine: Equilibrando entre acoplamento e coesão
7.9. Decompondo por capacidades de negócio (Business Capabilities)
7.10. Refinando fronteiras por subcapacidades e requisitos não funcionais
7.11. Reconhecendo bad smells na modelagem de microsserviços
7.12. Projeto AlgaSensors: Entendendo o negócio da empresa
7.13. Projeto AlgaSensors: Modelando um monólito
7.14. Projeto AlgaSensors: Modelando microsserviços por capacidades de negócio
7.15. Projeto AlgaSensors: Refinando a modelagem

7.16. Desafio: Modelagem de microsserviços
7.17. Decomposição por subdomínios: como DDD pode ajudar?

8. Setup do desenvolvedor
8.1. O que você precisa saber antes de colocar a mão na massa
8.2. Preparando o ambiente básico do desenvolvedor Java
8.3. Maven ou Gradle?
8.4. Git Repository: Monorepo vs multirepo
8.5. Criando Git Repository dos serviços
8.6. Configurando Git submodules
8.7. Criando microsserviços com Spring Boot e Gradle
8.8. Multiprojetos com a IntelliJ IDEA
8.9. Gradle: Estrutura do projeto
8.10. Instalando o plugin do Lombok na IntelliJ IDEA
8.11. Usando Git Bash na IntelliJ IDEA
8.12. Adicionando plugins do Gradle
8.13. Adicionando novas dependências no Gradle
8.14. Gerenciando serviços na IntelliJ IDEA
8.15. O que é SonarQube?
8.16. Instalando o SonarQube plugin na IntelliJ IDEA

9. Identificadores em sistemas distribuídos
9.1. O problema dos identificadores nos sistemas distribuídos
9.2. Entendendo o UUID v4 e v7
9.3. Entendendo o TSID
9.4. Usando o UUID v4 e v7 com Java
9.5. Usando o TSID com Java
9.6. Outros tipos e abordagens: Snowflake ID, Ticket Server e Dual Identifier

10. Implementação de microsserviços
10.1. O que vamos implementar?
10.2. Implementando um microsserviço com Spring
10.3. Implementando um serializador customizado para TSID com Jackson
10.4. Implementando a persistência com Jakarta Persistence e Spring Data JPA
10.5. Ajustando a representação do recurso REST com classe Model
10.6. Implementando a consulta de sensores
10.7. Implementando paginação de resultados da consulta
10.8. YAML ou Properties: qual usar?
10.9. Desafio: atualização e remoção de Sensor
10.10. Desafio: inativação dos sensores
10.11. Implementando microsserviço de Temperature Processing
10.12. Setup do microsserviço de Temperature Monitoring
10.13. Implementando endpoints do Sensor Monitoring
10.14. Implementando a consulta do log de temperaturas
10.15. Desafio: configuração de alertas para os sensores

11. Integração de REST API com RestClient do Spring
11.1. Revendo a comunicação dos microsserviços
11.2. Integração de REST API com RestClient do Spring
11.3. HTTP Status Codes para erros na integração

11.4. Lidando com erros de timeout
11.5. Simplificando a implementação com Factory Pattern
11.6. Implementando consulta de dados usando GET
11.7. Implementando client declarativo com HTTP Interface

12. Comunicação com mensageria e RabbitMQ
12.1. Comunicação assíncrona no projeto AlgaSensors
12.2. O message broker RabbitMQ
12.3. Componentes do RabbitMQ
12.4. Nomenclaturas e versionamento
12.5. Simulando RabbitMQ
12.6. Configurando RabbitMQ com Docker
12.7. Criando Exchanges e Queues via GUI do Admin no RabbitMQ
12.8. O Spring AMQP
12.9. Configurando o Spring AMQP no projeto
12.10. Criando uma Exchange com Spring
12.11. Criando Queue e Binding com Spring
12.12. Publicando mensagens
12.13. Adicionando cabeçalhos na mensagem
12.14. Consumindo mensagens
12.15. Persistindo dados de temperatura
12.16. Competing Consumers Pattern
12.17. Múltiplas Queues para um consumidor
12.18. Implementando service para os alertas
12.19. Direct Exchange
12.20. Topic Exchange
12.21. Headers Exchange

13. Resiliência com RabbitMQ
13.1. Retry Pattern
13.2. Lidando com erros em um consumer usando Retry Pattern
13.3. Dead Letter Channel Pattern e Dead Letter Queue Pattern
13.4. Utilizando uma Dead Letter Queue (DLQ)
13.5. O que fazer com as mensagens da DLQ
13.6. Mensageria não acaba por aqui

14. Conclusão e próximos passos
14.1. Conclusão do Nível 1

Nível 2 - Domain-Driven Design

1. Domain-Driven Design e design estratégico
1.1. O que é um Domínio de Negócio?
1.2. Atacando a complexidade do domínio com DDD
1.3. Como DDD ajuda na arquitetura de microsserviços
1.4. Espaço do problema e espaço da solução
1.5. Começando pelo Design Estratégico
1.6. Descobrindo o domínio através dos Domain Experts
1.7. Entendendo o papel dos subdomínios
1.8. Identificando e decompondo subdomínios

1.9. Os 3 tipos de subdomínios: Core, Supporting e Generic Subdomain
1.10. Desafio: tipos de subdomínios
1.11. Destilando subdomínios
1.12. Domain Model: indo para o espaço da solução
1.13. Encarando de frente o Big Ball of Mud (BBoM)
1.14. Contexto é rei: quando o significado depende
1.15. Bounded Contexts e Linguagem Ubíqua
1.16. Heurísticas para Bounded Contexts
1.17. Quando faz sentido usar DDD?

2. Integração dos Bounded Contexts
2.1. Explorando os relacionamentos entre equipes
2.2. Conhecendo os padrões de Context Mapping
2.3. Evoluindo juntos com Partnership pattern
2.4. Compartilhando parte do modelo com Shared Kernel pattern
2.5. Negociando integrações com Customer-supplier Development pattern
2.6. Consistência imposta com Conformist pattern
2.7. Protegendo integrações com Anti-corruption Layer pattern
2.8. Expondo interfaces descontaminadas com Open Host Service e Published Language
2.9. Escolhendo a independência com Separate Ways pattern
2.10. Mapa de navegação: a bússola do DDD estratégico
2.11. Desafio: integração de Bounded Contexts

3. Projeto AlgaShop: design estratégico e arquitetura de microsserviços
3.1. Planejando a descoberta de domínio
3.2. Descobrindo e classificando subdomínios
3.3. Explorando subdomínios de Vendas e Catálogo de Produtos
3.4. Explorando subdomínios de Cobranças e Processamento de Pagamentos
3.5. Explorando subdomínios de Entregas e Identidade
3.6. Visualizando as integrações com Context Mapping
3.7. Modelando a arquitetura de microsserviços

4. Introdução ao design tático do DDD
4.1. O que é o design tático
4.2. Explorando os padrões táticos
4.3. Começando a modelagem do Domain Model com UML
4.4. Associações relacionando os objetos de negócio

5. Entities, Value Objects e Factories
5.1. Entendendo o Domain Entity Pattern
5.2. Anemic Domain Model e Rich Domain Model
5.3. Database-centric domain model
5.4. Domain Entity e Jakarta Persistence Entity
5.5. Escolhendo identificadores para Domain Entities
5.6. Command Query Separation (CQS) e o DDD
5.7. Exercício: Configurando repositórios Git do projeto
5.8. Criando projeto do microsserviço de Ordering
5.9. Refinando Domain Model
5.10. Entendendo o JavaBean Pattern
5.11. O problema do JavaBean Pattern no Domain Model

5.12. Refinando em direção ao Rich Domain Model
5.13. Refatorando a Customer Entity em direção a um Rich model
5.14. Dando adeus aos getters do JavaBean
5.15. Implementando a geração de UUID v7
5.16. Adicionando validações na entidade Customer
5.17. Testando validações com testes unitários
5.18. Implementando e validando regras de negócio com testes unitários
5.19. Exceções para regras de negócio
5.20. Implementando a funcionalidade de pontos de lealdade
5.21. Conceitos errados comuns sobre Domain Entities
5.22. O que é um Value Object?
5.23. Enriquecendo o Domain Model com Value Objects
5.24. Implementando Value Objects
5.25. Refatorando as entidades para usar Value Objects
5.26. Desafio: implementação de Value Objects para Customer
5.27. Refinando Domain Model: Adicionando Value Object de endereço
5.28. Factories: criando objetos complexos com Creational Patterns do GoF
5.29. Implementando Value Object de Address
5.30. Simplificando a criação de Customer com Static Factory Method
5.31. Usando Builder e Factory Method juntos
5.32. Implementando Test Data Builders

6. Aggregates e mais sobre Factories
6.1. Entendendo o Domain Aggregate Pattern e a Aggregate Root
6.2. As características de um Aggregate
6.3. Regras de negócio e invariantes
6.4. Consistência e transações
6.5. Relacionamentos e Aggregates
6.6. Tamanho ideal para um Aggregate
6.7. Analisando os detalhes de Order
6.8. Refinando os detalhes da Order Entity
6.9. Analisando Aggregates
6.10. Modelagem de Aggregates
6.11. Refinando Customer e Product
6.12. Desafio: Implemente o Value Object de Money e Quantity
6.13. Implementando novos identificadores
6.14. Implementando Aggregate de Order
6.15. Implementando Factory Method e Builder em Order e OrderItem
6.16. Adicionando OrderItem em um Order
6.17. Protegendo Collections
6.18. Propriedades calculadas
6.19. Controlando alteração de status
6.20. Usando regras para o controle de alteração de status
6.21. Implementando métodos para o preenchimento de uma Order
6.22. Implementando regras de negócio para garantir invariantes
6.23. Implementando o padrão TestDataBuilder em Order
6.24. Aprimorando Exceptions com Factory Method
6.25. Alterando quantidade de um item
6.26. Refinando Order Aggregate
6.27. Implementando Value Object de Product

6.28. Value Objects e regras de negócio
6.29. Refinando a linguagem onipresente da implementação
6.30. Evoluindo modelo de BillingInfo
6.31. Gerando Aggregates inteiros com Factories
6.32. Desafio: Bloqueando edição de um Order
6.33. Desafio: Remoção de itens de um Order
6.34. Desafio: Implemente o método para marcar um Order como ready
6.35. Desafio: Cancelamento de um Order
6.36. Detalhando Shopping Cart Aggregate
6.37. Desafio: Implemente o Shopping Cart Aggregate

7. Isolando o Domain Model com estilos arquiteturais
7.1. A importância de isolar o Domain Model
7.2. Estilos arquiteturais, princípios e sua relação com DDD
7.3. Layered Architecture
7.4. Isolando o Domain Model com Layered Architecture
7.5. Utilizando a Layered Architecture na prática

8. Repositories e infraestrutura de persistência
8.1. Repository Pattern
8.2. Persistence Model e Domain Model
8.3. Definindo um Repository no Domain Model
8.4. Soluções para persistência: Jakarta Persistence e Spring Data JPA
8.5. Preparando dependências do projeto
8.6. Implementando Persistence Model
8.7. Como testar a persistência?
8.8. Isolando execução dos testes de integração
8.9. Testando modelo de persistência com testes de integração
8.10. Otimizando testes de persistência com @DataJpaTest
8.11. Persistindo um Aggregate
8.12. Assembler: Conversor de Domain Entity para Jakarta Persistence Entity
8.13. Disassembler: Conversor de Jakarta Persistence Entity para Domain Entity
8.14. Reconstituição: Entendendo a recuperação do estado de um Aggregate
8.15. Implementando propriedades únicas para o modelo de persistência
8.16. Atualizando o estado de um Aggregate
8.17. O problema da concorrência
8.18. Optimistic Locking
8.19. Implementando Optimistic Lock
8.20. Protegendo alterações indevidas de Version
8.21. Definindo propriedades compostas usando @Embbeded
8.22. Integrando Embeddable e Entity
8.23. Exercício: Copiando dados dos Value Objects para Embeddables
8.24. Exercício: Reconstituindo Value Objects compostos
8.25. Persistindo Entities em cascata
8.26. Mesclando dados dos itens de um pedido
8.27. Exercício: Continue a implementação do Disassembler
8.28. Consultas para informações resumidas
8.29. Problema do Lazy Loading
8.30. Desafio: Implemente persistência para Customer
8.31. Relacionando entidades de persistência

8.32. Consultas com filtros
8.33. Consultas para listagens
8.34. Criando consultas com JPQL
8.35. Consultas para somas e contagens com filtros
8.36. Consultas ligadas a verificações
8.37. Desafio: Implemente persistência para Shopping Cart

9. Domain Services
9.1. Introdução aos Services
9.2. Entendendo o Domain Service Pattern
9.3. Desenvolvendo um Domain Service
9.4. Desenvolvendo o Domain Service para registro de Customer
9.5. Usando Mockito para testar um Domain Service
9.6. Domain Service para consulta de Product
9.7. Como lidar com atualizações em massa
9.8. Usar Annotations do Spring ou não
9.9. Desafio - Implemente o Domain Service para Checkout
9.10. Desafio - Implemente o Domain Service de BuyNow
9.11. Desafio - Implemente o Domain Service para ShoppingCart

10. Integração entre Bounded Contexts e Anti-corruption Layer (ACL)
10.1. ACL e integração entre contextos
10.2. Domain Service e ACL para calcular frete
10.3. Domain Service para buscar endereço AlgaShop
10.4. API da Rapi-Dex com WireMock
10.5. Integração com Spring RestClient

11. Modularização dirigida pelo domínio
11.1. Introdução aos Modules
11.2. Organizando módulos do domínio
11.3. Modules aplicados na camada de infrastructure

12. Application Services e a camada de Application
12.1. Application Services Pattern
12.2. Application Service para casos de uso de Customer
12.3. Consulta de dados
12.4. Modularização na camada de Application
12.5. Mapeadores automáticos com ModelMapper
12.6. Customizando mapeamento de propriedades
12.7. Implementando caso de uso de atualização de Customer
12.8. Implementando caso de uso compra instantânea
12.9. Desafio - Implemente o arquivamento de Customer
12.10. Desafio - Implemente alteração de email para Customer
12.11. Desafio - Application Service para adição de Loyalty Points
12.12. Desafio - Application Service para gestão de Shopping Cart
12.13. Desafio - Application Service para gestão de Order
12.14. Desafio - Application Service para Checkout

13. Domain Events
13.1. Domain Event Pattern

13.2. Modelando eventos de Customer
13.3. Implementando evento de Customer Registered
13.4. Implementando o evento de Customer Archived ₣1
13.5. Implementando infraestrutura para publicação de eventos
13.6. Escutando eventos com EventListener
13.7. Garantindo escuta de eventos com Mockito Spy
13.8. Reagindo a eventos
13.9. Enriquecendo eventos
13.10. Modelando eventos de Order
13.11. Desafio: Implemente os eventos de Order
13.12. Implementando o processamento de Loyalty Points
13.13. Modelando eventos de Shopping Cart
13.14. Desafio: Implemento os eventos de Shopping Cart
13.15. Corrigindo testes de integração

14. Specifications
14.1. Domain Specification Pattern
14.2. Implementando regra para frete grátis
14.3. Implementando Specification parametrizável
14.4. Implementando Specification composta
14.5. Utilizando Specification no Checkout Service
14.6. Exercício: Compartilhe sua opinião sobre o DDD tático em sua abordagem purista

15. Command Query Responsibility Segregation (CQRS)
15.1. O que é CQRS?
15.2. Separando componentes de consultas e comandos em Customer
15.3. Otimizando consulta de Customer com JPQL
15.4. Implementação simplificada com Entity Manager
15.5. Modelo de leitura detalhado para Order
15.6. Implementando consulta detalhada para Order
15.7. Consulta paginada para Order com Criteria API
15.8. Definindo filtros e ordenação
15.9. Filtrando Order por Customer
15.10. Adicionado mais filtros para Order
15.11. Implementando ordenação
15.12. Desafio: Consulta de clientes
15.13. Desafio: Consulta de carrinhos

16. Abordagem pragmática para o DDD
16.1. DDD pragmático
16.2. Construindo o microsserviço de Billing
16.3. Domain Model de Billing
16.4. Implementando o Domain Model
16.5. Utilizando o JavaBean Pattern de maneira segura
16.6. Factories e Builders
16.7. Comportamento rico nas Entities
16.8. Validações nas Entities
16.9. Validações nos Value Objects
16.10. Desafio: Implemente testes unitarios para Invoice
16.11. Adicionando persistência nas Entities

16.12. Adicionando persistência nos Value Objects
16.13. Aprimorando geração de colunas com Hibernate
16.14. Repository orientado a persistência
16.15. Domain Services e ACL
16.16. Application Service com CQRS para geração de Invoice
16.17. Application Service para pagamento de Invoice
16.18. Consultas com QueryService
16.19. Controle de concorrência e auditoria
16.20. Emitindo e escutando eventos de Invoice
16.21. Exercício: Compartilhe sua opinião, DDD puro ou pragmático?

17. Conclusão e próximos passos
17.1. Conclusão Nível 2

Nível 3 - Design e Engenharia de Microsserviços com REST

1. Camada de Presentation com REST APIs
1.1. O que esperar desse nível
1.2. Entendendo o Presentation Layer
1.3. Como arquitetar e modelar REST APIs
1.4. Dicas para modelagem de recursos
1.5. Como testar REST APIs?

2. Documentação de contratos e Design First com OpenAPI
2.1. Trabalhando com contratos nas REST APIs
2.2. OpenAPI Specification (OAS) e Swagger
2.3. Ferramentas OpenAPI
2.4. Criando o documento de especificação
2.5. Definindo recursos
2.6. Detalhando o modelos do recursos
2.7. Definindo requisições e respostas dos recursos
2.8. Detalhando validações nos modelos
2.9. Descrevendo modelo de erro padrão RFC 7807
2.10. Descrevendo modelo de paginação
2.11. Descrevendo query param
2.12. Simplificando modelo de Customer na paginação
2.13. Múltiplos formatos de requests em um endpoint
2.14. Modelando Enums
2.15. Desafio: Documente os recursos de consulta de Order
2.16. Desafio: Documente os recursos de ShoppingCart

3. Implementação de REST API com testes de contrato
3.1. Testes de contrato
3.2. Testando endpoint de criação de cliente
3.3. Usando Mock no teste de contrato
3.4. Testes para erros de requisição
3.5. Testes com paginação
3.6. Avaliando Location Header
3.7. Testes com Path dinâmico
3.8. Aprimorando tratamento de erros para Exceptions

3.9. Desafio: Desenvolva e teste o contrato do recurso de clientes

4. Contract-Driven Development com Spring Cloud Contract
4.1. Contract-driven Development
4.2. Spring Cloud Contract
4.3. Domínio do catálogo de produtos
4.4. Criando projeto Product Catalog com Spring Cloud Contract
4.5. Criando definição de contrato testável
4.6. Implementando recurso orientado ao contrato
4.7. Evoluindo contratos e recursos
4.8. Gerando e executando Stub
4.9. Contratos e Stub dinâmicos e com validação
4.10. Testando respostas de erro padrão RFC 7807
4.11. Contrato e recurso de listagem
4.12. Implementando Spring Cloud Contract no Ordering
4.13. Teste de contrato com Mocks
4.14. Simulando erros em Ordering com Mocks
4.15. Refatorando testes de contrato de Product
4.16. Simulações de erros com Mocks em Product
4.17. Detalhando validações contrato do contrato
4.18. Desafio: Contrato e implementação de recursos para Products
4.19. Desafio: Contrato e implementação de recursos para Category
4.20. Desafio: Contrato e implementação de recursos para Order
4.21. Desafio: Contrato e implementação de recursos para ShoppingCart
4.22. Gerando documentação da REST API com Spring REST Docs

5. Testes para REST APIs com REST Assured e Postman
5.1. Testes de integração ou teste de API?
5.2. Teste de integração com REST Assured
5.3. Corrigindo respostas de erro da REST API
5.4. Avaliando resposta de forma de detalhada
5.5. Configurando Mock de Product Catalog com WireMock e Docker
5.6. Integrando com Product Catalog API
5.7. Utilizando WireMock diretamente no Java
5.8. Testando indisponibilidade na integração
5.9. JSON vs DTOs: qual utilizar nos testes de integração?
5.10. Testes usando Stub gerado pelo Spring Cloud Contract
5.11. Testes manuais com Postman
5.12. Aprimorando respostas da API de Customer
5.13. Aprimorando respostas da API de ShoppingCart
5.14. Evitando conflitos entre testes de integração
5.15. Desafio: Testes de integração para Recursos de Customer
5.16. Desafio: Testes de integração para Recursos ShoppingCart
5.17. Desafio: Testes de integração para criação de Order
5.18. Desafio: Adicione o WireMock em ShippingCostServiceIT

6. Versionamento e migração banco de dados com Flyway e PostgreSQL
6.1. Versionamento de banco de dados com Flyway
6.2. O banco de dados PostgreSQL
6.3. Adicionando novos componentes a arquitetura

6.4. Configurando PostgreSQL com Docker
6.5. Dependências e configurações para Flyway e PostgreSQL
6.6. Usando plugin do Flyway para Gradle
6.7. Executando Flyway Migrations via Spring Boot
6.8. Implementando schema completo do banco do dados de Ordering
6.9. Configurando dados de testes com Flyway
6.10. Configurando um banco de dados real para testes de integração
6.11. Limpando dados de teste com @Sql
6.12. Usando Flyway para limpar o banco de testes
6.13. Configurando testes com @DataJpaTest para usar o Postgres
6.14. Corrigindo problemas de vazamento conexão nos testes
6.15. Evitando o uso do @DirtiesContext
6.16. Alimentando testes de integração com dados via SQL
6.17. Adicionando unique key para Shopping Cart
6.18. Desafio: Implemente a integração com Flyway e o Postgres no microsserviço de Billing

7. Integração com REST API de pagamentos Fastpay
7.1. Integração com REST API de pagamentos
7.2. Trabalhando com cartões de crédito de forma segura
7.3. Conhecendo a API da Fastpay
7.4. Preparando estrutura básica para integração
7.5. Implementando API Client para gestão de cartões de crédito
7.6. Validando integração usando testes automatizados
7.7. API Client para gestão de pagamentos
7.8. Domain Service de pagamentos como ACL
7.9. Testes de integração API de pagamentos
7.10. Camada de Application para Credit Card
7.11. Camada de Presentation em Billing
7.12. Recebendo Webhooks
7.13. Criando Mock da API de pagamentos com WireMock
7.14. Adicionando WireMock diretamente no Java
7.15. Desafio: Aprimorando tratamento de erros e validações
7.16. Recebendo dados de cartão no microsserviço de Ordering
7.17. Corrigindo persistência de Billing Email

8. Docker e Microservice Template
8.1. Dockerização, Profiles e Service Template
8.2. Dockerizando um microsserviço
8.3. Utilizando Profile Group
8.4. Desafio: Dockerize Billing
8.5. Múltiplos Docker Compose
8.6. Criando Microservice Template

9. Distributed Scheduling
9.1. Task Scheduling para sistemas distribuídos
9.2. Adicionando um novo microsserviço na arquitetura
9.3. Criando microsserviço de Billing Scheduler
9.4. Implementando expiração de uma Invoice
9.5. Implementando Scheduling com Spring
9.6. Processando dados com JdbcTemplate

9.7. Evitando concorrência com Lock e transações
9.8. Batch update
9.9. Transformando aplicação em um Short-lived Microservice
9.10. Integração com Fastpay

10. Testcontainers
10.1. A biblioteca Testcontainers
10.2. Utilizando Postgres com Testcontainers
10.3. Automatizando ciclo de vida do container e configurações de conexão
10.4. Abstraindo configuração de testes
10.5. Adicionando Testcontainer aos testes de Presentation
10.6. Configurando um Testcontainer como um Bean
10.7. Desafio: Migre os testes para utilizar o Testcontainers em Ordering
10.8. Desafio: Migre os testes para utilizar o Testcontainers em Billing

11. Hexagonal Architecture (Ports & Adapters)
11.1. Entendendo a Arquitetura Hexagonal (Hexagonal Architecture)
11.2. Organizando pacotes
11.3. Definindo um Input Port para gestão de Shopping Cart
11.4. Definindo Ports para consulta de Shopping Carts
11.5. Definindo um Input Adapter para gestão de Shopping Cart via REST
11.6. Organizando Output Adapters de Shopping Cart
11.7. Definindo Ports e Adapters para Customers
11.8. Desafio: Implemente Ports e Adapters para Orders
11.9. Organizando componentes genéricos
11.10. Modelo puro ou pragmático? Entendendo o papel de um Configurator
11.11. Desafio: Refatorando testes para seguir a Hexagonal Architecture

12. Conclusão e próximos passos
12.1. Conclusão do Nível 3

Nível 4 - Resiliência e Escalabilidade em Microsserviços

1. Resiliência, Escalabilidade e NoSQL
1.1. Introdução ao nível
1.2. Bancos de dados relacionais e os desafios enfrentados
1.3. Bancos de dados não relacionais (NoSQL)
1.4. As famílias de bancos de dados não relacionais
1.5. Como os diferentes bancos consultam os dados?
1.6. Estrutura, dados e esquema
1.7. ACID e BASE

2. Implementação de microsserviço com MongoDB
2.1. Estudo de caso E-commerces e NoSQL
2.2. Revisando documentação do Product Catalog
2.3. Configurando MongoDB e MongoDB Compass
2.4. Atualizando e adicionando novas dependências no projeto
2.5. Mapeando Domain Entity como Document
2.6. Testando persistência de um Document
2.7. Configurando codificação de UUID

2.8. Implementando conversor de datas
2.9. Implementando versionamento e auditoria
2.10. Mapeando Product com Document
2.11. Cadastro de Product com Application Service
2.12. Enriquecendo o Aggregate de Product
2.13. Aprimorando tratamento de Exceptions
2.14. Implementando atualização de Product
2.15. Relacionamentos e Documents
2.16. Gerando propriedades dinâmicas
2.17. Geração de Slug e descrição simplificado

3. Filtros e projeções com MongoDB
3.1. Consultas usando MQL
3.2. Operadores na MQL
3.3. Consultas e projeções via MongoDB Shell
3.4. Filtros e projeções com Repository
3.5. Conectando camadas para consulta de Products
3.6. Estrutura para filtros dinâmicos
3.7. Filtros com Criteria e MongoOperations (MongoTemplate)
3.8. Filtrando valores de datas
3.9. Filtrando valores decimais
3.10. Comparando dois campos no mesmo documento
3.11. Filtros In, Is e Gt com Criteria
3.12. Filtros com Or e Regex
3.13. Importando dados via Java

4. Índices no MongoDB
4.1. Como funcionam os índices no MongoDB?
4.2. Índices individuais
4.3. Índices compostos e ordenação
4.4. Criando índices simples via Spring
4.5. Criando índices compostos via Spring
4.6. Criando índice parcial via Spring
4.7. Índice para busca textual via Spring
4.8. Deixando o DataLoader compatível com criação de índices
4.9. Desafio: Filtros para Category

5. Aggregation Framework do MongoDB
5.1. Aggregation Framework
5.2. Pipeline e operadores do Aggregation Framework
5.3. Agrupando resultados com $group
5.4. Joins com $lookup
5.5. Aggregation Framework no Spring
5.6. Adicionando contagem total
5.7. Valores dinâmicos em projeções

6. Modelagem, desnormalização e consistência eventual
6.1. Modelagem e desnormalização
6.2. Garantindo consistência eventual na aplicação
6.3. Alterações de schema e controle de versão

6.4. Desnormalizando o modelo
6.5. Migração de dados
6.6. Adaptando consultas para o novo schema
6.7. Eventos de aplicação
6.8. Consistência eventual via eventos
6.9. Recriando índices
6.10. Eventos de domínio para Product
6.11. Desafio: Mais eventos de domínio para Product
6.12. Opinião: Desnormalização

7. Controle de concorrência e atomicidade
7.1. Controle de concorrência e atomicidade
7.2. Controle de concorrência usando $inc
7.3. Retornando produto alterado de Update
7.4. Domain Service para alteração de estoque
7.5. Interligando camadas para funcionalidade de estoque

8. Cluster e transações no MongoDB
8. Cluster e transações no MongoDB
8.1. Transações no MongoDB
8.2. Como funciona um cluster do MongoDB
8.3. Configurando cluster
8.4. Conectando ao cluster usando hostname
8.5. Implementando histórico de movimentação de estoque
8.6. Transações com Spring e MongoDB
8.7. Eventos e transações
8.8. Testes e transações
8.9. Apêndice Product Catalog
8.10. Apêndice Ordering
8.11. Apêndice Billing
8.12. Apêndice Billing Scheduller
8.13. Apêndice Template

9. Caching Patterns e Redis
9.1. Cache e sistemas distribuídos 1
9.2. Client-side Cache e Server-Side Cache
9.3. Centralized cache e Distributed cache
9.4. HTTP Caching: Padrão para cache na Web (RFC9111)
9.5. HTTP Caching: Headers e funcionamento
9.6. Implementações de Cache no Spring Framework
9.7. HTTP Caching com Spring Framework
9.8. Padrões de escrita: Cache-Aside e Write-Through
9.9. Estratégias de invalidação de cache
9.10. Redis: serviço de cache ou banco de dados?
9.11. Configurando Redis como servidor de Cache (Centralized Cache) no Docker
9.12. Gestão e comandos básicos Redis
9.13. Implementando Server-side cache com Cache-Aside
9.14. Invalidando cache via TTL
9.15. Client-side cache e Cache-Aside
9.16. Escrita de cache com Write-Through

9.17. Invalidação de cache via gatilho com CacheEvict
9.18. Resiliência contra indisponibilidade no servidor de cache
9.19. Mudanças de schema e erros de desserialização
9.20. Server-Side Cache em uma listagem com filtros
9.21. Cache: consistência eventual ou limpeza programática?
9.22. Cache HTTP para uma listagem com filtros

10. Padrões de resiliência em microsserviços
10.1. Padrões de resiliência
10.2. Timeout Pattern
10.3. Utilizando Timeout Pattern na integração com fretes
10.4. Utilizando Timeout Pattern na integração com Produtos
10.5. Retry Pattern
10.6. Utilizando o Retry Pattern com @Retryable
10.7. Bulkhead Pattern
10.8. Utilizando Bulkhead Pattern com @ConcurrencyLimit
10.9. Cache e Retry qual a ordem ideal?
10.10. Ignorando HTTP Client Errors no Retry
10.11. Circuit Breaker Pattern
10.12. Resiliência com Spring Cloud Circuit Breaker
10.13. Visualizando estado do Circuit Breaker
10.14. Desafio: Resiliência para Ordering na integração com Rapidex
10.15. Fallback Pattern
10.16. Utilizando Fallback Pattern
10.17. Endpoint para consulta de frete
10.18. Desafio: Resiliência para Billing na integração com Fastpay

11. Health Check e Spring Boot Actuator
11.1. Health Check Pattern
11.2. Spring Boot Actuator
11.3. Implementando Health Check com Spring Boot Actuator
11.4. Customizando regras de Readiness
11.5. Customizando Health Check
11.6. Desafio: Implemente Health check em Ordering e Billing
11.7. Indicadores de Circuit Breaker no Actuator
11.8. Desafio: Implemente o Health check para o Circuit Breaker para Billing
11.9. Desafio: Dockerizando Product Catalog

12. Performance testing e Virtual Threads
12.1. Análise de performance e escalabilidade
12.2. Instalando o K6
12.3. Escrevendo primeiro Smoke test com K6
12.4. Testando microsserviço de Product Catalog com K6
12.5. Implementando um Load testing (Teste de carga)
12.6. Thresholds com K6
12.7. Utilizando variáveis de ambiente nos testes
12.8. Explorando Scenarios e Executors no K6
12.9. Testando escritas com Volume Test
12.10. O que são Virtual Threads?
12.11. Utilizando Virtual Threads no Spring

13. Armazenamento de arquivos com AWS S3 e LocalStack
13.1. Introdução a AWS S3, Localstack e Spring Cloud AWS
13.2. Configurando container Docker para Localstack
13.3. Criando Bucket S3 no LocalStack
13.4. Adicionando imagens ao Bucket da Localstack
13.5. Implementando estrutura de gestão de upload de arquivos
13.6. Implementando propriedade de Images na Collection de Product
13.7. Implementando Application Service de Image
13.8. Implementando Query Service para Image
13.9. Gerando URL de acesso para imagens
13.10. Configurando Spring Cloud AWS para o S3
13.11. Geração de Pre Signed URL para upload com AWS S3
13.12. Implementando a remoção de arquivo com AWS SDK
13.13. Adicionando referência das imagens ao banco de produtos
14. Conclusão e próximos passos
14.1. Conclusão do Nível 4

Nível 5 - Segurança, Autenticação e Autorização

1. Fundamentos de Segurança em Arquiteturas Distribuídas
1.1. O que é segurança?
1.2. Desafios e benefícios de segurança em arquiteturas distribuídas
1.3. Identidade, autenticação e autorização
1.4. Gestão de Identidade e Acesso
1.5. Certificados vs. Tokens vs. Senhas
1.6. Princípios essenciais de segurança
1.7. Vulnerabilidades comuns e análise estática de código
1.8. Necessidades da AlgaShop

2. Introdução ao OAuth2 e Authorization Server
2.1. Fundamentos do OAuth2
2.2. Criando estrutura inicial do Spring Authorization Server
2.3. Componentes do OAuth2
2.4. Como definir Clients e Scopes?
2.5. Configurando Clients e Scopes da AlgaShop
2.6. Entendendo o fluxo de autorização Client Credentials
2.7. Endpoints OAuth2 e geração de token com Client Credentials
2.8. Endpoint de descoberta de configuração (OAuth2 metadata)

3. Tokens e JWT
3.1. Tokens opacos e transparentes
3.2. JSON Web Tokens (JWT)
3.3. Explorando o token JWT e entendendo os Claims
3.4. Endpoints: Introspecção e JSON Web Key Set
3.5. Configurando tempo de expiração de um token
3.6. Explorando os tokens de acesso opacos (By Reference Tokens)

4. Resource Server e filtros de segurança
4.1. Transformando microsserviço em um Resource Server
4.2. Configurando descoberta de chave pública
4.3. Fixando Issuer no Authorization Server
4.4. Identificando Scopes necessários
4.5. Configurando filtros globais de segurança para proteger os endpoints
4.6. Utilizando filtros por método para proteger os endpoints
4.7. Annotations customizadas para filtros em métodos
4.8. Desafio: Finalize a implementação dos filtros de segurança em Product Catalog
4.9. Desafio: Transforme Billing em um Resource Server
4.10. Desafio: Transforme Ordering em um Resource Serve
4.11. Criando Mock para JwtDecoder
4.12. Refatorando testes de integração para o utilizar tokens
4.13. Testando fluxos de acesso proibido (403)
4.14. Ajustando testes e Billing

5. Integração entre microsserviços com OAuth2
5.1. Transformando Ordering em um OAuth2 Client
5.2. Implementando gestão de Access Token para o RestClient
5.3. Otimizando geração de token pelo Client
5.4. Diminuindo acoplamento entre Client e Authorization Server

6. Authorization Code e Refresh Token
6.1. Entendendo o fluxo do Grant Type Authorization Code
6.2. Analisando necessidades do AlgaShop E-commerce Web App
6.3. Configurando Clients para AlgaShop E-commerce Web App
6.4. Testando fluxo do Authorization Code passo a passo
6.5. Testando fluxo do Authorization Code automático via Postman
6.6. Entendendo o Grant Type Refresh Token
6.7. Habilitando Grant Type Refresh Token
6.8. Preparando banco de dados para o Authorization Server
6.9. Armazenamento de autorizações
6.10. Armazenamento de consentimento do usuário

7. OpenID Connect
7.1. Camada de identidade com OpenID Connect (OIDC)
7.2. Implementando estrutura de persistência para usuários
7.3. Integrando login com banco de dados
7.4. Habilitando e customizando o ID Token
7.5. Customizando Access Token
7.6. Explorando endpoints do OpenID Connect
7.7. Implementando SecurityFilterChains
7.8. Customizando o OIDC UserInfo endpoint
7.9. Explorando OIDC Logout
7.10. Invalidando refresh tokens e autorizações após logout
7.11. Gerenciamento de sessão com Spring Session

8. Gestão e REST API de usuários
8.1. Iniciando implementação da REST API de usuários
8.2. Implementando funcionalidade de cadastro de usuário

8.3. Identificando o usuário logado
8.4. Aprimorando auditoria
8.5. Filtros de segurança para REST API de usuários
8.6. Desafio: Implemente endpoints na REST API de usuários Resultado
8.7. Desafio: Aprimore a capacidade de auditoria

9. Integrando Clients públicos ou seguros
9.1. Demonstrando um Client OAuth2 seguro: Spring Web App
9.2. Clients públicos: desvantagens e opções de segurança
9.3. Grant Type Authorization Code com PKCE
9.4. Adicionando Client Admin no Authorization Server
9.5. Testando fluxo Authorization Code com PKCE
9.6. Demonstrando um Client OAuth2 publico Angular
9.7. Preparando Authorization Server para permitir Silent Refresh
9.8. Demonstrando Silent Refresh
9.9. Dockerizando Authorization Server

10. Permissões e controle de acesso
10.1. RBAC - Role-Based Access Control
10.2. Implementação Roles no Authorization Server
10.3. Transformando Roles em Authorities
10.4. Regras de segurança para o registro de usuários
10.5. Regras de segurança para edição de usuário
10.6. Limitando ações por Role com PreAuthorize
10.7. Limitando consulta de pedidos
10.8. Adicionando regras a criação de pedidos
10.9. Limitando Scopes por Roles
10.10. Limitar acesso aos Clients por Role
10.11. Validando Scopes por Role durante a geração de Authorization Code

11. Customizando Authorization Server
11.1. Customizando tela login
11.2. Customizando tela logout
11.3. Customizando tela consentimento
11.4. Configurando redirecionamento padrão após login

12. Técnicas para testes com Spring Security
12.1. Testes envolvendo segurança com Mocks
12.2. Injetando autenticação JWT
12.3. Testes para garantir regras de segurança
12.4. Testes envolvendo API Client autenticadas

13. Validação de e-mail e alteração de senha
13.1. Token para alteração de senha e verificação de email
13.2. Alteração de senha via token
13.3. Camada de apresentação para alteração de senha
13.4. Envio de email para alteração de senha
13.5. Funcionalidade “esqueci minha senha”
13.6. Envio de email assíncrono

14. Identity-Bound REST Resources e restrições de acesso
14.1. Desafios de integração e segurança da AlgaShop
14.2. Expandindo Identity-Bound REST Resources
14.3. Implementando recursos "me" de Customer
14.4. Implementando recursos "me" de Ordering
14.5. Desafio - Refatoração dos testes de integração e contrato
14.6. Desafio - Implementando recursos "me" de Shopping Cart
14.7. Desafio - Implementando recursos "me" de Users
14.8. Desafio - Implementando recursos "me" de Credit Carts
14.9. Desafio - Revisando permissões Product Catalog

15. Armazenamento seguro de configurações e segredos
15.1. Centralized e Externalized Configuration Patterns
15.2. Armazenando segredos no AWS Secret Manager
15.3. Armazenando configurações e segredos no Parameter Store
15.4. Gerando par de chaves RSA
15.5. Preparando Authorization Server para integração com a AWS
15.6. Carregando segredos do AWS Secret Manager
15.7. Importando múltiplos parâmetros do AWS Parameter Store
15.8. Importando parâmetros compartilhados
15.9. Armazenando e carregando chave RSA no AWS Secret Manager
15.10. Gerando JWKS a partir de chave privada RSA
15.11. Carregando parâmetros de listas
15.12. Otimizando importação de parâmetros para LocalStack
15.13. Desafio: Analise e centralize os parâmetros de configuração dos microsserviços

16. Conclusão nível 5
16.1. Revisando arquitetura e atualizando diagrama do sistema
16.2. Conclusão nível 5

Nível 6 - API Gateway, BFF e Service Discovery

1. Service Discovery Patterns com Eureka
1.1. Service Discovery Patterns
1.2. Criando Service Registry Eureka com Spring Cloud Netflix
1.3. Implementando configuração básica do Eureka
1.4. Self-Registration Pattern em ação: Microsserviço como Client do Eureka
1.5. Customizando informações de registro do Client
1.6. Configurando Eureka Server para considerar o Health Check
1.7. Desafio: Implemente o Eureka Client nos microsserviços
1.8. Entendendo o Load Balance Pattern
1.9. Client-Side Service Discovery e Client-Side Load Balacing com RestClient e Eureka
1.10. Dockerizando o Service Registry e configurando AWS Parameter
1.11. Health Check e indisponibilidade

2. API Gateway com Spring Cloud Gateway
2.1. API Gateway Pattern
2.2. Criando projeto com Spring Cloud Gateway

2.3. Implementando roteamento no Gateway
2.4. Reescrevendo Path do request
2.5. Manipulando headers da response
2.6. Roteamento com Server-side service discovery e Load Balance
2.7. Gerenciando CORS no Gateway
2.8. Removendo Headers duplicados com filtros globais
2.9. Configurando Actuator e Healthcheck
2.10. Authorization Server atrás do API Gateway
2.11. Validando tokens JWT no Gateway
2.12. Desafio: Definição de rotas
2.13. Dockerizando o API Gateway
2.14. Ajustando CORS

3. Padrões de resiliência e performance no API Gateway
3.1. Resiliência aplicada ao Gateway
3.2. Aplicando o Timeout Pattern
3.3. Aplicando o Retry Pattern
3.4. Implementando Circuit Breaker Pattern com Resilience4j
3.5. Implementando filtro de Local Response Cache com Caffeine
3.6. Implementando Rate Limit Pattern com Redis

4. Back-end for front-end (BFF) e integração com Apps AlgaShop
4.1. Introdução ao Back-end for front-end e BFF Patterns
4.2. API Composition no BFF para o App do E-commerce
4.3. Desafio: Crie a estrutura inicial do BFF para o Admin
4.4. Adicionando BFF e Apps a arquitetura
4.5. Integração com Front-end do E-commerce
4.6. Integração com Front-end do Admin
4.7. Dois Clients duas necessidades de Cache diferentes
4.8. Otimizando resposta com JSON Filter
4.9. Bloqueando acesso a usuários de clientes
4.10. Ajustando configurações não utilizadas pelo E-commerce
4.11. Flexibilizando remoção de cartão de crédito
4.12. Configurando Docker Compose para Client Apps
4.13. Desafio: Externalizando configurações dos BFFs
5. Conclusão e próximos passos
5.1. Conclusão nível 6

Nível 7 - Event-Driven Architecture com Kafka

1. Introdução a Event-Driven Architecture
1.1. Introdução ao nível
1.2. O desafio da comunicação no fluxo de pedidos
1.3. Eventos e eventos de integração
1.4. Event-Driven Architecture (EDA)
1.5. Padrões de comunicação assíncrona e mensageria
1.6. Componentes Event-Driven Architecture

1.7. Workflows - Orquestração e coreografia

2. Kafka Arquitetura e funcionamento
2.1. Kafka Event Streaming Platform
2.2. Records
2.3. Ecossistema e APIs
2.4. Topics e partitions
2.5. Consumer groups
2.6. Pull vs Push
2.7. Cluster

3. Streaming de eventos com Spring Kafka
3.1. Criando cluster Kafka com Docker
3.2. GUI para Kafka Broker com Docker
3.3. Criando tópicos no Producer com Spring Kafka
3.4. Enviando eventos com Kafka Template
3.5. Garantindo ordem dos eventos nas partições
3.6. Implementando o consumidor
3.7. Lidando com erros de desserialização
3.8. Customizando o TypeId para serialização
3.9. Desserialização com TypeId customizado
3.10. Processando eventos com TypeId desconhecido
3.11. Customizando identificação do TypeId
3.12. Padronizando envio de eventos

4. Padrões e modelagem de eventos
4.1. Introdução aos padrões e modelagem de eventos
4.2. Descobrindo eventos e reações
4.3. Armadilhas na modelagem de eventos
4.4. Eventos e reações da AlgaShop
4.5. Utilizando Event Notification Pattern
4.6. Reagindo aos eventos
4.7. Versionamento e alteração de contrato
4.8. Utilizando Event-Carried State Transfer Pattern
4.9. Validando consumo de eventos com Bean Validation
4.10. Validando produção de eventos com Bean Validation

5. Documentação de contratos de eventos com AsyncAPI
5.1. Contratos para APIs assíncronas
5.2. Design-first e Code-first
5.3. Introdução ao AsyncAPI
5.4. Criando a base do documento AsyncAPI
5.5. Descrevendo messages
5.6. Reaproveitando schemas
5.7. Descrevendo operations
5.8. Versões e depreciação de eventos
5.9. Documentando o consumer e reaproveitando mensagens

6. Técnicas de resiliência
6.1. Resiliência com Kafka
6.2. Retry e Timeout no producer
6.3. Consistência no banco quando o evento falha
6.4. Corrigindo o cluster Kafka para simular falhas
6.5. Confirmações de recebimento (Acknowledgements)
6.6. Tratamento de erros no producer
6.7. Timeout no consumer
6.8. Retry e tratamento de erros no consumer
6.9. Consumidores paralelos via threads
6.10. Idempotência no produtor
6.11. Técnicas para idempotência no consumidor
6.12. Implementando idempotência no consumidor
6.13. Limpando de forma programática a tabela de mensagens
6.14. Entendendo o Dead Letter Channel Pattern
6.15. Implementando um Dead Letter Topic no consumer
6.16. Simplificando a identificação do TypeId em um producer-consumer
6.17. O problema do Dual Write
6.18. Publicando eventos em Ordering
6.19. Command Message Pattern
6.20. Listen to yourself Pattern
6.21. Listen to yourself Pattern com eventos
6.22. Listener de eventos para geração de Order
6.23. Listen to yourself Pattern com comandos
6.24. Comandos desconhecidos e idempotência
6.25. Outbox Pattern
6.26. A tabela de outbox
6.27. A entidade e o repositório do outbox
6.28. Gravando o evento dentro da transação
6.29. O KafkaTemplate e o sender da outbox
6.30. Implementando Polling Publisher para o outbox
6.31. Retry e backoff quando o envio falha
6.32. As guardas do ciclo
6.33. Mantendo a ordem por Aggregate
6.35. Publicando eventos depois do Commit
6.36. Desafio - Opinião sobre técnicas de resiliência

7. Workflow com coreografia e eventos
7.1. Coreografia e Event Collaboration
7.2. Workflow da AlgaShop
7.3. Apresentando contratos AsyncAPI
7.4. Publicando evento de pedido realizado
7.5. Desafio - Integre o Billing com Kafka
7.6. Gerando fatura para o pedido
7.7. Desafio - Publique evento de fatura emitida
7.8. Processando fatura emitida e webhook de pagamentos
7.9. Desafio - Prepare Ordering para consumir eventos do Billing
7.10. Reagindo à fatura paga
7.11. Apêndice - Corrigindo testes

8. Transações distribuídas com Saga e coreografia
8.1. Entendendo transações distribuídas com Saga
8.2. Apresentando a Saga da AlgaShop
8.3. Cancelando o pedido pela fatura cancelada
8.4. Modelando a reserva de estoque
8.5. Retirando itens reservados do estoque
8.6. Reservando estoque para o pedido
8.7. Desafio - Torne Product Catalog um consumidor dos eventos de Ordering
8.8. Desafio - Reserve o estoque do pedido pago
8.9. Desafio - Publique o resultado da reserva
8.10. Desafio - Marque o pedido como pronto para envio
8.11. Estornando o pagamento como compensação
8.12. Desafio - Publique eventos de Produto e os escute
8.13. Desafio - Opine sobre a Saga com coreografia

9. Saga e orquestração
9.1. Saga com orquestração
9.2. O papel do Command, Reply e CorrelationId na Saga
9.3. Fluxo da AlgaShop com Saga e orquestração
9.4. Desafio - Remova o checkout assíncrono (Listen to yourself) e refatore o Snapshot
9.5. Desafio - Remova a Saga com coreografia para dar lugar à Saga com orquestração
9.6. Modelando o estado da Saga
9.7. Implementando infraestrutura de persistência para Saga
9.8. Iniciando a Saga quando o pedido é realizado
9.9. Separando mensagens de entrada e de saída
9.10. Apresentando os contratos AsyncAPI da Saga
9.11. Preparando o Outbox para transportar comandos
9.12. Enviando o comando de emissão de fatura
9.13. Gerando a fatura a partir do comando
9.14. O problema do reply assíncrono - guardando o reply pendente
9.15. Enviando o reply manualmente ao coordenador
9.16. Recebendo replies no canal da Saga
9.17. Avançando a Saga com a fatura paga
9.18. Reservando o estoque e reply automático (@SendTo)
9.19. Conclua a Saga com a reserva confirmada
9.20. Compensando o pedido quando o pagamento é recusado
9.21. Compensando a fatura quando falta estoque
9.22. Processando cancelamento da fatura
9.23. Validando quantidade antes da compra
9.24. Desafio - Envie os comandos da Saga sem Outbox
9.25. Desafio - Ative os validadores de payload no Kafka
9.26. Desafio - Opine sobre a Saga com orquestração

10. Notificações
10.1. Notificações, comandos e eventos
10.2. Apresentando o esqueleto do serviço de notificações
10.3. Apresentando os contratos OpenAPI e AsyncAPI
10.4. Validando sintaxe HTML

10.5. Renderizando templates com Thymeleaf
10.6. Enviando e-mail via SMTP
10.7. Desafio - Implemente a API de cadastro e gerenciamento de templates de e-mail
10.8. Desafio - Implemente a API de envio de e-mails
10.9. Desafio - Proteja a API com OAuth2 Resource Server
10.10. Desafio - Registre o Notifications no Eureka e exponha-o pelo gateway-admin
10.11. Desafio - Receba comandos de e-mail pelo Kafka
10.12. Desafio - Envie o comando de e-mail a partir do Ordering
11. Conclusão e próximos passos
11.1. Conclusão e próximos passos

Nível 8 - Integração e Streaming de Dados com Microsserviços (em desenvolvimento)

1. Relatórios e dados em microsserviços
1.1. Introdução ao nível
1.2. Relatórios e microsserviços
1.3. Relatórios operacional e tático no Ordering
1.4. Desafio - Implemente o endpoint de métricas de clientes e carrinhos ativos
1.5. Desafio - Implemente o endpoint de métricas de transações de pagamento por mês
1.6. O problema do relatório estratégico
1.7. Reporting Database e ETL
1.8. O que é CDC e quando usá-lo
2. Kafka Connect e CDC com Debezium
2.1. Kafka Connect - workers, conectores, tasks, converters e transforms
2.2. Configurando Kafka Connect com Docker
2.3. Insights - o microsserviço de relatórios da AlgaShop
2.4. Preparando bancos de dados
2.5. Extraindo pedidos com Debezium
2.6. Simplificando a mensagem com transforms
2.7. Renomeando campos com transforms
2.8. Gravando pedidos no Insights com o sink JDBC
2.9. Automatizando a criação dos conectores
2.10. Deletes, tombstones e tipos de dados
2.11. Lidando com falhas nos conectores
2.12. Desafio - Replique os clientes
2.13. Desafio - Replique os itens dos pedidos
2.14. Extraindo produtos do MongoDB
2.15. Gravando produtos no Insights
2.16. Convertendo o UUID do MongoDB com um SMT customizado
2.17. Reprocessando dados e evoluindo o schema

Nível 9 - Práticas Avançadas de DevOps (em desenvolvimento)

1. Fundamentos de DevOps
1.1. Introdução ao curso
1.2. O que é DevOps na prática

1.3. O que é Continuous Integration, Continuous Deployment e Continuous Delivery
1.4. Conhecendo o The-Twelve Factor App
1.5. Convertendo o Delivery Tracking para Gradle
1.6. Desafio: Conversão para Gradle do projeto Courier
1.7. Alterando application.yml para usar ENVs
1.8. Configurando o Flyway
1.9. Conhecendo o Spring Boot Actuator
1.10. Criando o Dockerfile do projeto
1.11. Configurando o Docker Compose do projeto Delivery
1.12. Desafio: Dockerfile e Docker Compose do projeto Courier
2. Fundamentos do GitLab CI/CD
2.1. Introdução ao GitLab
2.2. Primeiros passos com GitLab CI/CD
2.3. Before/After script, retry e artifacts
2.4. Controlando execução de jobs
2.5. Criando a primeira pipeline do projeto
2.6. Utilizando services em um Job
2.7. Entendendo o Docker DinD
2.8. Construindo pipeline com build e push para Docker Hub
2.9. Desafio: Pipeline no Courier-Management
3. Semantic Release
3.1. Git Flow
3.2. Semantic Version (SemVer)
3.3. Convetional Commits
3.4. Configurando o Semantic Release no projeto
3.5. Configurando o Semantic Release na pipeline
3.6. Utilizando âncora na pipeline
3.7. Criando uma imagem para o Semantic Release
3.8. Desafio: Semantic Release no Courier-Management
4. Garantia de qualidade na entrega
4.1. Quebrando mitos sobre Qualidade de Software
4.2. Utilizando Testcontainers na pipeline
4.3. Executando a pipeline com Testcontainers
4.4. Qualidade de código na prática com SonarQube
4.5. Configurando cobertura de teste com JaCoCo
4.6. Configurando verificação de cobertura de teste com JaCoCo
4.7. Configurando SonarLint no Visual Studio Code
4.8. Automatizando análise do SonarQube na pipeline
4.9. Gerando status report no Merge Request
4.10. Sincronizando codebase do GitLab com o SonarQube
4.11. Desafio: Replique o pipeline de qualidade no Courier-Management
5. Segurança no ciclo de desenvolvimento
5.1. Revisando os 12 fatores no projeto
5.2. Shift Left Security: Prevenindo CVEs no desenvolvimento
5.3. Scan de CVEs no projeto com Trivy
5.4. Scan de CVEs na pipeline
5.5. Assinando imagens de container com Cosign
5.6. Assinando imagens na pipeline
5.7. Verificando assinaturas antes do deploy
5.8. Desafio: Replique o pipeline segurança no Courier-Management

5.9. Conclusão e próximos passos
6. Apêndice: Pipeline Algashop (conteúdo em desenvolvimento)

Nível 10 - Observabilidade de Microsserviços (em desenvolvimento)

Nível 11 - Microsserviços em Produção na AWS (em desenvolvimento)

1. Introdução a Cloud Computing
1.1. Introdução ao nível 11
1.2. O que e Cloud Computing: definição e evolução
1.3. Cloud vs On-Premise: quando usar cada abordagem
1.4. Fundamentos de Cloud Computing (IaaS, PaaS, SaaS)
1.5. Características essenciais: elasticidade, escalabilidade e self-service
1.6. Cloud Providers e Modelo de precificação
1.7. Shared Responsibility Model
2. Fundamentos de Redes na AWS
2.1. Visão geral da rede do AlgaDelivery na AWS
2.2. Criando sua conta na Amazon AWS
2.3. Criando budgets para controle de custo
2.4. Regiões e AZs (Zonas de Disponibilidades)
2.5. VPC: Fundamentos da Rede Virtual na AWS
2.6. CIDR Blocks e Planejamento de Endereçamento IP
2.7. Criando uma VPC na AWS
2.8. Subnets: Segmentação da VPC
2.9. Criando Subnets na AWS
2.10. Route Tables e Decisão de Trafego
2.11. Criando Route tables na AWS
2.12. Internet Gateway: Conectando a VPC a Internet
2.13. Criando Internet Gateway na AWS
2.14. NAT Gateway: Saida Segura para Subnets Privadas
2.15. Criando NAT Gateway na AWS
2.16. Security Groups: Firewall de recursos
2.17. Provisionando uma EC2 Pública na VPC
2.18. Criando um Elastic IP em uma EC2
2.19. Provisionando uma EC2 Privada
3. Infraestrutura como Código (IaC) na AWS
3.1. Introdução ao IaC
3.2. Explorando o Terraform e OpenTofu
3.3. Setup do ambiente (OpenTofu + AWS CLI)
3.4. Configurando um Provider
3.5. Criando o primeiro recurso com OpenTofu
3.6. Fluxo de trabalho com OpenTofu
3.7. Trabalhando com variáveis
3.8. Gerando Outputs
3.9. Explorando o arquivo tfstate

3.10. Backend remoto com S3
3.11. Criando uma VPC com OpenTofu
3.12. Criando subnets publicas com OpenTofu
3.13. Utilizando expressions e meta-arguments
3.14. Criando Internet Gateway com OpenTofu
3.15. Criando subnets privadas e Nat Gateway com OpenTofu
3.16. Configurando Route tables com OpenTofu
3.17. Criando Security Groups (Bastion + Private)
3.18. Utilizando Data Sources e Functions
3.19. Criando EC2 com OpenTofu
3.20. Testando conectividade da rede
3.21. Destruindo os recursos com OpenTofu
4. IaC Modular e Serviços Gerenciados na AWS
4.1. Introdução ao módulo
4.2. Utilizando dynamics blocks
4.3. Criando modulo VPC
4.4. Tipos de coleções e for expressions
4.5. Refatorando o módulo VPC
4.6. Criando modulo EC2
4.7. Explorando multi-env no OpenTofu
4.8. Lifecycle e boas praticas no OpenTofu
4.9. Criando um RDS PostgreSQL via Console
4.10. Provisionando RDS PostgreSQL via OpenTofu
4.11. Gerenciando senhas com segurança no RDS
4.12. Criando modulo RDS
4.13. Configurando backup e janela de manutenção no RDS
4.14. Criando um ElastiCache Valkey via Console
4.15. Provisionando ElastiCache Valkey via OpenTofu
4.16. Desafio: Criando Modulo ElastiCache
4.17. Provisionando Amazon MSK (Kafka) via Console
4.18. Provisionando Amazon MSK (Kafka) via OpenTofu
4.19. Desafio: Modulo MSK
4.20. Testando conectividade do banco RDS
4.21. Destruindo os recursos

5. Kubernetes: Fundamentos e Workloads
5.1. Containers vs VMs - isolamento, portabilidade e trade-offs
5.2. Introdução ao Kubernetes
5.3. Kubernetes- arquitetura, Control Plane e Worker Nodes
5.4. Instalando kubectl, Kind e criando cluster local
5.5. Pods- ciclo de vida
5.6. Pods multicontainer
5.7. Namespaces, labels e annotations
5.8. Replicaset - failover e disponibilidade de pods
5.9. Deployments
5.10. Deployments- revisões e rollback
5.11. Volumes - emptydir e hostpath
5.12. ConfigMaps
5.13. Secrets
5.14. Persistent Volumes, PVC e Storage Classes

5.15. StatefulSets
5.16. Services
5.17. Services - Nodeport e Headless
5.18. Self Healing com probes
5.19. Resource request e limits
5.20. Ingress Controller e Ingress
5.21. Setup do AlgaDelivery no K8S
5.22. HPA - autoscaling de pods
5.23. Desafio: Realize a configuração dos manifestos do Courier Management

** Alguns dos níveis do EMS estão em desenvolvimento. Solicite a brochura para conhecer o conteúdo.*