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