Checkpoint 1 – Desenvolver uma API SpringBoot de Pedidos

Disciplina: Arquitetura SOA e Web Services

Professor: Jefferson Junior

Grupo: Até 3 membros

Objetivo da Atividade

Este checkpoint tem como objetivo aplicar os conceitos iniciais de arquitetura orientada a serviços (SOA) e desenvolvimento de API Spring Boot, criando um serviço para cadastrar e gerenciar pedidos de clientes.

1. Criando o Projeto Spring Boot (1 ponto)

Criar um projeto Maven com as seguintes configurações:

- Group: br.com.fiap
- Artifact: checkpoint1
- Name: checkpoint1
- Description: Checkpoint 1
- Package Name: br.com.fiap.checkpoint1
- Packaging: Jar
- Java Version: 17
- Spring Boot Version: 3.1.*
- Dependências: Spring Web, Spring Boot DevTools, Lombok, Spring Data JPA, H2 Database

Dica: Utilize o Spring Initializr (start.spring.io) para gerar o projeto.

2. Estruturando a API (1 ponto)

O projeto deve seguir a estrutura abaixo no diretório src/main/java/br/com/fiap/checkpoint1:

├── controller // Responsável pelos endpoints

├── model // Representação dos dados (Entidades)

├── ervisse // Interface para interação com o banco de dados

├── ervisse // Contém as regras de negócio

Crie uma entidade chamada Pedido com os seguintes atributos:

- id (Long, chave primária, gerada automaticamente)
- clienteNome (String, não pode ser nulo ou vazio)
- dataPedido (LocalDate, deve ser preenchida automaticamente com a data atual)
- valorTotal (double, não pode ser negativo)

Regras de validação:

- O nome do cliente deve ser obrigatório.
- O valor total do pedido não pode ser negativo.

3. Criando o Repositório (2 pontos)

Criar a interface PedidoRepository no pacote repository, estendendo JpaRepository<Pedido, Long> para facilitar as operações de banco de dados.

4. Criando o Service (2 pontos)

Criar a classe PedidoService no pacote service, responsável por implementar as regras de negócio, utilizando a interface do repositório.

5. Criando o Controller (2 pontos)

Criar a classe PedidoController no pacote controller, responsável por expor os endpoints REST:

- GET /pedidos → Retorna a lista de pedidos cadastrados.
- GET /pedidos/{id} → Retorna um pedido pelo ID.
- POST /pedidos → Cria um novo pedido.
- PUT /pedidos/{id} → Atualiza um pedido existente.
- DELETE /pedidos/{id} → Remove um pedido pelo ID.

Regras de implementação:

- Utilize Spring Data JPA para interagir com o banco de dados.
- Utilize H2 Database como banco de dados em memória.

6. Configurando o Banco de Dados (1 ponto)

No arquivo application.properties, configure o H2 Database para armazenamento dos pedidos. O console do H2 deve estar habilitado para que os alunos possam visualizar os dados armazenados.

7. Testando a API (1 ponto)

Utilize o Postman ou outra ferramenta similar para testar todos os endpoints da API.

Testes obrigatórios:

1. Criar um novo pedido.
2. Buscar todos os pedidos.
3. Buscar um pedido pelo ID.
4. Atualizar um pedido.
5. Deletar um pedido.

Documentação:

- Incluir um README no repositório explicando os endpoints e exemplos de requisições/respostas.

Entregáveis (cada aluno deverá entregar individualmente)

1) Repositório no GitHub

- Nome: fiap-checkpoint1
- O repositório deve conter:
- Código-fonte do projeto
- Arquivos de configuração do banco de dados
- README com instruções de uso e exemplos de requisições (print das telas indicando os 5 testes realizados).

2) Relatório de Testes

Cada aluno deve enviar para o email do professor um arquivo .txt contendo nome completo do aluno e o link do repositório no GitHub.

Exemplo (checkpoint1.txt):

Aluno(a)	Github
Maria	https://github.com/maria/fiap-checkpoint1
João	https://github.com/joao/fiap-checkpoint1
Ana	https://github.com/ana/fiap-checkpoint1