Feature 1: configuração essencial e gestão de entidade primária

A primeira feature do projeto será a base para todo o desenvolvimento da disciplina. Ela visa consolidar os conhecimentos iniciais de configuração de um projeto Spring Boot, modelagem de uma entidade central e a implementação das camadas fundamentais para sua gestão em memória, expondo-a através de uma API REST.

Lembrem-se que o contexto do projeto (ex: sistema de biblioteca, gestão de estoque, catálogo de filmes) será **escolhido por vocês e precisará ser aprovado pelo professor da disciplina** antes do início do desenvolvimento. As regras e implementações descritas abaixo deverão ser aplicadas sobre o domínio que vocês selecionarem.

Objetivo

Configurar a aplicação Spring Boot, definir a entidade de domínio principal do seu projeto e implementar as camadas de `Controller` e `Service` para gerenciar essa entidade em memória, utilizando os conceitos introduzidos nas aulas 01 e 02.

Conceitos e requisitos técnicos a serem aplicados

1. Configuração do projeto

- Utilizem o **Spring Initializr** para criar um novo projeto Spring Boot.
- Assegurem-se de incluir a dependência `Spring Web` e configurar a versão do Java conforme o que foi discutido em aula (Java 17 ou superior).
- Configurem seu ambiente de desenvolvimento (IDE - Eclipse ou similar) para importar e executar o projeto corretamente, utilizando o Maven Build (`clean install`).

2. Modelagem da Entidade de Domínio

- Identifiquem a **entidade principal** do seu projeto (ex: `Livro`, `Produto`, `Aluno`, `Cliente`).
- Crie uma classe Java para representar essa entidade.
- Implementem atributos relevantes para a entidade, utilizando diferentes tipos de dados (String, int, double, boolean etc.).
- Assegurem a **encapsulamento** dos atributos (com `getters` e `setters`, ou utilizando `@Data` do Lombok).
- Sobrescrevam o método `toString()` da entidade para uma apresentação clara e completa de seus dados.
- A entidade deve possuir um atributo `id` (do tipo `Integer` ou `Long`).

3. Camada de Serviço (Service Layer)

- Crie uma interface genérica (`CrudService<T, ID>`) com os métodos CRUD básicos (`salvar`, `buscarPorId`, `excluir`, `listarTodos`).
- Implementem uma classe de serviço específica para a sua entidade principal (ex: `LivroService`) que implemente essa interface.
- Essa classe de serviço deve ser anotada com `@Service` para ser gerenciada pelo Spring.

- **Persistência em Memória:** Utilize um `Map` (preferencialmente `ConcurrentHashMap` para simular um ambiente robusto) dentro da sua classe de serviço para armazenar as instâncias da sua entidade.
- **Geração de ID:** Implemente a lógica para gerar IDs únicos e sequenciais para a sua entidade utilizando `AtomicInteger`.

4. População Inicial de Dados com Loader

- Crie uma classe `Loader` em seu projeto.
- Esta classe deve ser configurada para ser executada automaticamente na inicialização da aplicação Spring, utilizando a anotação `@Component` e implementando a interface `ApplicationRunner`.
- **Leitura de arquivo texto:** prepare um arquivo texto simples com dados para popular a sua entidade (ex: `livros.txt`, onde cada linha representa uma entidade e os dados são separados por vírgula ou outro delimitador).
- **Injeção do serviço:** injete a sua classe de serviço (ex: `LivroService`) no `Loader` para que ele possa utilizar o método `salvar` para persistir as entidades lidas do arquivo no `Map` em memória.
- **Confirmação da população:** após a leitura e salvamento de todos os dados do arquivo, utilize o método `listarTodos` do seu serviço para recuperar as entidades recém-cadastradas e imprima-as no console (`System.out.println`) para confirmar que os dados foram carregados corretamente.

5. Camada de Controle (Controller Layer - API REST)

- Crie uma classe `Controller` para a sua entidade principal (ex: `LivroController`).
- Anote-a com `@RestController` para construir serviços RESTful.
- Implemente um endpoint (`@GetMapping`) para listar todas as entidades salvas em memória. O retorno desse endpoint deve ser uma coleção de objetos completos da sua entidade (evitando retornar apenas atributos individuais).
- Garanta que o nome do endpoint siga a boa prática de pluralidade (ex: `/livros`).
- **Injeção de Dependência:** Injete a sua classe de serviço (ex: `LivroService`) no `Controller` utilizando o padrão de **injeção por construtor** (a forma mais recomendada no Spring Boot).

6. Execução e Teste

- A aplicação deve ser capaz de iniciar sem erros.
- Vocês devem ser capazes de acessar o endpoint via navegador (ou ferramenta como Postman/Insomnia) e visualizar a lista de entidades que foram pré-populadas ou adicionadas em memória.

Pontos Bônus (Opcional para a Feature 1 - Sugestão para Feature 2)

Se o seu domínio permitir, e vocês se sentirem à vontade, já podem começar a aplicar conceitos de **Herança** (como a classe `Pessoa` abstrata e suas subclasses `Vendedor`/`Comprador`) ou **Associação** (como `Vendedor` tendo um `Endereco`). No entanto, esses conceitos serão aprofundados e mandatórios em features posteriores.

Entrega

- O código-fonte do projeto no GitHub (apresentar o link do repositório).
- Uma breve demonstração do funcionamento da API via navegador/Postman, mostrando a listagem das entidades.

Esta primeira feature é crucial para solidificarmos os alicerces do desenvolvimento com Spring Boot. Contem comigo para quaisquer dúvidas ao longo do processo!