Guia Passo a Passo — Projeto SISPRADIA

Este documento serve como **roteiro prático e mental** para a construção do projeto **SISPRADIA (Sistema de Práticas Diárias)** usando **Spring Boot + JPA + REST**, seguindo boas práticas e separação de responsabilidades.

> Objetivo: **saber o que criar, quando criar e por quê**, sem depender de copiar código.

Visão Geral da Arquitetura

Fluxo padrão de uma requisição:

Cliente → Controller → Serviço de Aplicação → Serviço de Domínio → Repositório → Banco

Volta:

Banco → Repositório → Serviço de Domínio → Serviço de Aplicação → DTO → Controller → Cliente

Cada camada tem **uma responsabilidade clara**.

Etapa 1 — Entidades (✔️já feito)

O que são

Representam o **negócio** e o **estado persistido** no sistema.

Regras

- Não conhecem DTO
- Não conhecem Controller
- Não conhecem HTTP
- Representam o mundo real

Exemplos

- Usuario
- Pratica / Habito
- RegistroDiario
- Meta

Etapa 2 — Repositórios (✔️ já feito)

O que são

Interfaces de acesso ao banco de dados.

Responsabilidade

- Buscar
- Salvar
- Verificar existência

Importante

- Repositório **não valida regra de negócio**
- Apenas consulta dados

Etapa 3 — Serviço de Domínio (VOCÊ FAZ PRIMEIRO)

O que é

Camada que contém **as regras de negócio**.

Pergunta-chave

> "Quais regras precisam ser respeitadas para esta entidade existir?"

Exemplos de regras

- Não cadastrar usuário com email duplicado
- Não cadastrar prática sem usuário
- Definir data de criação automaticamente
- Ativar entidade ao cadastrar

Características

- Recebe e retorna **Entidades**
- Usa Repositórios
- Não conhece DTO
- Não conhece Controller

Ordem mental para criar

1. Liste as regras no papel
2. Transforme regras em métodos
3. Depois implemente

Etapa 4 — DTOs (Entrada e Saída)

O que são

Objetos de **comunicação externa**.

Tipos comuns

- DTO de entrada (ex: CadastrarUsuarioDto)
- DTO de saída (ex: ListarUsuarioDto)

Regras

- Contêm apenas dados
- Podem ter validações (@NotNull, @Email etc.)
- Não contêm lógica de negócio

Importante

DTO **não é entidade** e **não representa o domínio**.

Etapa 5 — Conversores

O que são

Tradutores entre **DTO ↔ Entidade**.

Responsabilidade

- DTO → Entidade
- Entidade → DTO

Regras

- Não acessam banco
- Não validam regra de negócio
- Apenas copiam dados

Pergunta mental

> "Como transformar o que veio de fora em algo que o domínio entenda?"

Etapa 6 — Serviço de Aplicação

O que é

Camada que **orquestra o fluxo**.

Responsabilidade

1. Receber DTO
2. Converter DTO → Entidade
3. Buscar entidades relacionadas (ex: Usuario)
4. Chamar Serviço de Domínio
5. Converter Entidade → DTO

Importante

- Não contém regra de negócio
- Não acessa banco diretamente (usa domínio)

Pense nele como

> O maestro que coordena, mas não toca o instrumento

Etapa 7 — Controller (REST)

O que é

Camada de **entrada HTTP**.

Responsabilidade

- Receber requisição
- Validar DTO (@Valid)
- Chamar Serviço de Aplicação
- Retornar DTO

Regras

- Não contém regra de negócio
- Não acessa repositório
- Não converte entidade

Etapa 8 — Tratamento de Exceções (opcional agora)

O que é

Centraliza erros de negócio e erros técnicos.

Exemplos

- RecursoNaoEncontradoExcecao
- RegraNegocioExcecao
- Handler global

Ordem Recomendada para Cada Nova Funcionalidade

Sempre siga esta sequência:

1. Entidade (se necessário)
2. Repositório
3. **Serviço de Domínio** (regras primeiro)
4. DTOs
5. Conversor
6. Serviço de Aplicação
7. Controller

Regra de Ouro

> **Se você não sabe onde colocar um código, pergunte:**
>
> - Isso é regra de negócio? → Domínio
> - Isso é comunicação? → DTO
> - Isso é fluxo? → Serviço de Aplicação
> - Isso é HTTP? → Controller

Objetivo Final

Ao seguir este roteiro, você: - Aprende arquitetura de verdade - Consegue criar sistemas sozinho - Para de depender de IA para pensar - Cria projetos profissionais no GitHub

Este documento é seu **mapa**. Volte nele sempre que se perder.