Atividade DDD

Objetivo

Executar, do começo ao fim, os **passos de Domain-Driven Design (DDD)** para propor o **primeiro modelo** e **primeira API** de um módulo de busca. O foco é **aprender a pensar e comunicar o domínio**, não implementar código.

Cenário

Sua universidade mantém uma **Biblioteca Digital** usada por estudantes e pesquisadores para descobrir publicações (artigos, teses, livros, relatórios técnicos). Várias ferramentas internas precisam de busca:

- um portal público (estudantes buscando trabalhos)
- uma ferramenta interna de curadoria (bibliotecários ajustando metadados)
- rotinas de ingestão (importação de novos registros e deduplicação)

Historicamente, cada ferramenta implementou suas próprias regras de busca. Com o tempo, isso gerou comportamento inconsistente, correções duplicadas e integrações frágeis. A equipe decidiu construir um **módulo de busca reutilizável** (um componente/biblioteca, não um serviço) para que múltiplos clientes reaproveitem a mesma semântica de busca.

Conceitos centrais do domínio (como o negócio fala)

Uma **Publicação** é um registro que representa uma obra. Cada publicação possui:

- **PublicationId** (identificador estável, nunca reutilizado)
- **Título**
- **Resumo** (opcional)
- **Autores** (lista ordenada)
- **Ano** (opcional)
- **Veículo** (periódico/conferência/série de livros, opcional)
- **Palavras-chave/Tags** (opcional)

- **Texto completo** (opcional; pode não estar disponível por direitos autorais)

Necessidades dos usuários

Usuários normalmente querem:

- buscar por tema: “gestão de dívida técnica”, “microservices”, “kubernetes”
- filtrar resultados por intervalo de anos (ex.: 2020–2023)
- filtrar por autor (“Jane Doe”) ou por veículo (“ICSE”)
- preferir correspondências no **título** em relação ao resumo (comportamento de ranqueamento)
- ordenar por **relevância** por padrão, com opção de ordenar por **ano (mais recentes primeiro)**
- ver uma breve explicação: “por que isso apareceu?” (trecho/snippet ou campos correspondentes)

Requisitos funcionais

Seu módulo deve suportar dois grupos de operações.

A) Indexação

- Adicionar ou atualizar uma publicação no índice.
- Remover uma publicação.
- Reconstruir o índice (reindexação) a partir de uma lista de publicações.
- Preferir atualizações idempotentes (rodar a mesma atualização duas vezes não deve corromper o estado).

B) Busca

- Consulta por palavras-chave em campos relevantes do domínio (título/resumo/autores/palavras-chave).
- Filtros:
    - intervalo de anos (de/até)

    - veículo igual
    - autor contém
- Ordenação:
    - padrão por relevância
    - opcional por ano (decrescente)
- Resultados devem incluir:
    - `PublicationId`
    - metadados suficientes para exibição (no mínimo título + ano)
    - um artefato de explicação (trecho/snippet **ou** lista de campos que casaram)

Entregáveis (o que o grupo deve entregar)

1) Linguagem Ubíqua (Glossário)

Um documento curto com:

- lista de termos do domínio (ex.: “Publicação”, “Autor”, “Consulta”, “Filtro”, “Relevância”…)
- definição clara (1–2 frases)
- exemplo de uso (1 frase ou 1 caso)

Critério: termos consistentes e sem “termos técnicos de ferramenta”.

---

2) Subdomínios e Bounded Context

Um diagrama simples (pode ser desenho/PlantUML) indicando:

- quais partes são **core** / **supporting** / **generic** (justificativa curta)
- qual é o **Bounded Context de Busca** (Search)

- o que está **dentro** e **fora** do contexto de Busca

Critério: fronteiras claras e decisões justificadas (mesmo que sejam “primeira versão”).

---

3) Mapa de Contexto (Context Map)

Um desenho mostrando:

- quem consome o módulo de busca (portal, backoffice, etc.)
- de onde vêm os dados (catálogo/publicações)
- como a busca se integra (ex.: eventos, sincronização, chamadas diretas)

Critério: visão “de sistema”, não apenas classes.

---

4) Modelo Tático (Tactical DDD)

Um diagrama (ou lista estruturada) contendo:

- **Entidades** (com identidade)
- **Value Objects** (imutáveis, sem identidade)
- **Serviços de Domínio** (se necessário)
- **Regras/invariantes** importantes
- candidatos a **Agregados** (se fizer sentido)

Critério: coerência (por que algo é entidade vs VO), e alinhamento com a linguagem ubíqua.

---

5) Esboço da API pública (em linguagem do domínio)

Um documento com:

- nomes de operações (ex.: `indexarPublicacao`, `buscarPublicacoes`, `removerProduto`, etc.)

- tipos de entrada e saída **em termos do domínio** (não em termos de tecnologia)
- 2–3 exemplos de uso (pseudo-código)

Critério: a API “soa” como negócio, não como implementação.