2025inf329Grupo04

Backlog (2 tasks)

Decisões Técnicas e Casos Extremos Blue

📚 Documentação do Projeto

1. Visão Geral e Objetivo

O objetivo é produzir o núcleo (core) de um Sistema de Recomendação em **Java**, utilizando a biblioteca **Apache Mahout**, para o e-commerce de livros **Bookmarket**. O sistema deve ser inicialmente **in-memory** mas com arquitetura preparada para futura persistência (via Repositórios).

Conformidade de Segurança: Todas as soluções devem aderir estritamente a: **SECURITY/COMPLIANCE** (foco em Pseudonimização e Minimização de Dados).

4.1. Stack Tecnológico e Arquitetura

Componente	Detalhe
Linguagem	Java 21+
Build Tool	Maven
ML Core	Apache Mahout (PearsonCorrelationSimilarity, GenericUserBasedRecommender)
Arquitetura	Domain-Driven Design (DDD) simplificado, com Repositórios In-Memory.
Modelagem	Utiliza IDs numéricos (`long`) para Usuário/Livro (Mahout IDs e conformidade LGPD/GDPR).

4.2. Tratamento de Casos Extremos (Fallback)

Cenário	Decisão (Ajuste do Product Manager)
Cliente Sem Preferências (Novo)	O Mahout não conseguirá gerar recomendações. O sistema deve usar o **Fallback B** e **Retornar os Top N Bestsellers** (US1) como recomendações genéricas.
Escala de Dados	A população de dados de teste será em **Escala Micro** (10 usuários, 20 livros, 50 avaliações).
Regra de Preço Promocional	O **PricingEngine** será a única fonte de verdade para o menor preço vigente (Promoção ou Padrão).

Time estimate: oh **Time spent:** oh

Casos de Teste (Ampla Cobertura) Blue

A documentação completa exige que **todos** os cenários (positivos, negativos, limites) sejam cobertos por testes unitários e BDD.

User Story	ID	Cenário	Tipo de Cobertura	Resultado Esperado Chave
US1	P01	Retornar 5 bestsellers	Positivo	Lista com 5 livros, ordenada por contagem de vendas.
US1	N02	Limite Inválido (N=0)	Negativo	Lançar exceção de `IllegalArgumentException`.
US2	N01	Nota Inválida (Acima de 5.0)	Negativo	Lançar exceção de `IllegalArgumentException`.
US2	P03	Atualização de Avaliação	Positivo	Nota anterior é sobrescrita no DataModel.
US3/US4	P01	Recomendação Padrão	Positivo	Retornar 5 livros não avaliados, com preço correto (Média/Menor).
US3/US4	N01	Cliente Novo (Fallback)	Negativo	Retornar os Top 5 Bestsellers (Fallback B).
US4	P02	Cálculo Menor Valor (Promoção)	Positivo	Preço final deve ser o preço promocional mais baixo.

Time estimate: oh **Time spent:** oh

Planned (7 tasks)

US1: Listagem dos Bestsellers

Green

Bestsellers do Bookmarket

Como um Usuário (Visitante ou Assinante) identificado
EU QUERO solicitar a lista dos N livros mais vendidos no Bookmarket
PARA QUE eu possa descobrir rapidamente quais títulos estão em alta.

Regra de Negócio:

- Bestsellers são computados pela **contagem total de unidades vendidas**.
- Limite `N` deve ser **1 <= N <= 100**.

Cenário Principal: Listar a quantidade N solicitada

CENÁRIO: O sistema deve retornar exatamente N bestsellers válidos
DADO QUE o Bookmarket possui um histórico de vendas com mais de 100 livros únicos
E a venda é definida pela contagem de unidades vendidas (não pelo valor da receita)
QUANDO o usuário solicita a lista dos N bestsellers, onde 1 <= N <= 100
ENTÃO o sistema DEVE retornar uma lista contendo exatamente N objetos "Livro"
E a lista DEVE estar ordenada em ordem decrescente pela quantidade de unidades vendidas
E cada "Livro" retornado DEVE conter o Título, o Autor e a Contagem de Vendas.

Critérios de Aceite (Critérios de Qualidade e Funcionamento)

- A lista de bestsellers deve ser computada com base na contagem total de unidades vendidas por título.
- A consulta deve permitir um parâmetro de entrada inteiro N (o limite da lista).
- O sistema deve garantir que 1 ≤ N ≤ 100.
- A ordenação deve ser estritamente decrescente pela contagem de vendas. Em caso de empate na contagem de vendas, a ordenação secundária será pelo Título do Livro (alfabética crescente).
- A resposta deve ser rápida o suficiente para não demandar interação ou polling do usuário, indicando uma implementação eficiente (em Java, com Mahout ou lógica própria para este caso).

Casos de Teste (Ampla Cobertura: Positivo e Negativo)

ID	Descrição	Passos de Execução	Resultado Esperado	Tipo de Teste
P01	Retornar 10 bestsellers (Valor comum)	1. Executar a consulta com N=10.	1. Retornar uma lista com 10 livros, ordenados corretamente.	Positivo
P02	Retornar o valor máximo (N=100)	1. Executar a consulta com N=100.	1. Retornar uma lista com 100 livros, ordenados corretamente.	Positivo (Limite)
P03	Retornar o valor mínimo (N=1)	1. Executar a consulta com N=1.	1. Retornar uma lista com 1 livro (o mais vendido), ordenado corretamente.	Positivo (Limite)
P04	Histórico com livros empatados	1. Executar a consulta com N=5 (onde 2 livros estão empatados).	1. Os livros empatados devem aparecer sequencialmente, ordenados secundariamente pelo **Título**.	Positivo (Edge Case)
N01	Limite Inválido: N maior que 100	1. Executar a consulta com N=101.	1. O sistema deve retornar um **erro de validação** (ex: `IllegalArgumentException` ou tratamento de erro de entrada).	Negativo
N02	Limite Inválido: N igual a zero	1. Executar a consulta com N=0.	1. O sistema deve retornar um **erro de validação**.	Negativo
N03	Histórico de Vendas Vazio	1. Executar a consulta com N=10 em um ambiente sem histórico de vendas.	1. O sistema deve retornar uma **lista vazia**.	Negativo (Ausência de Dados)

| N04 | Histórico de Vendas Insuficiente | 1. Executar a consulta com N=50 em um ambiente com apenas 20 livros únicos vendidos. | 1. O sistema deve retornar uma lista com os **20 livros disponíveis**, ordenados corretamente. | Negativo (Dados Insuficientes) |

Time estimate: oh **Time spent:** oh

US2: Avaliação e Registro de Preferência de Livros

Green

FEATURE: Avaliação e Registro de Preferência de Livros

Como um Cliente (Visitante ou Assinante) identificado
EU QUERO fornecer uma nota de 0 a 5 para um livro
PARA QUE o Bookmarket possa registrar minha preferência e usá-la em futuras recomendações.

Regra de Negócio:

- Nota (`rating`) deve estar no intervalo **[0.0, 5.0]**.
- O registro deve ser inserido ou atualizado no **DataModel do Mahout**.

Cenário Principal: Registro de uma avaliação válida

CENÁRIO: O sistema deve registrar uma avaliação válida e ser integrado ao DataModel do Mahout
DADO QUE eu estou identificado com um User ID numérico (Opção A)
E o livro a ser avaliado possui um ID numérico válido
E eu desejo avaliar o livro com uma nota entre 0 e 5 (inclusive)
QUANDO eu submeto a nota para o livro
ENTÃO o sistema DEVE registrar essa avaliação de forma persistente (in-memory, no escopo atual)
E o registro DEVE ser inserido ou atualizado no objeto DataModel do Mahout (FastByIDMap, GenericDataModel)
E o sistema DEVE retornar um status de sucesso (sem exigir uma interação de tela, a menos que seja um erro).

Critérios de Aceite

- A função de avaliação deve aceitar três parâmetros de entrada: user_id (inteiro), book_id (inteiro) e rating (float/double, 0.0 a 5.0).
- O sistema deve utilizar as classes de Mahout (Preference, GenericPreference, GenericUserPreferenceArray) para armazenar os dados de preferência.
- O rating deve ser validado para estar no intervalo [0.0, 5.0].
  O sistema deve ser capaz de atualizar uma avaliação existente para o mesmo par (user_id, book_id).
- A implementação deve garantir que o método populateEvaluation(Random rand) seja o responsável por gerar os dados iniciais de teste (ratings) e popular a estrutura DataModel.

Casos de Teste

ID	Descrição	Passos de Execução	Resultado Esperado	Tipo de Teste
P01	Avaliação Válida (Nota 5.0)	1. Submeter avaliação 5.0 para user_id=1, book_id=10.	1. Avaliação registrada com sucesso no DataModel.	Positivo
P02	Avaliação Válida (Nota 0.0)	1. Submeter avaliação 0.0 para user_id=2, book_id=20.	1. Avaliação registrada com sucesso no DataModel.	Positivo (Limite)
P03	Atualização de Avaliação	1. Submeter nota 3.0 para (user_id=3, book_id=30). 2. Submeter nota 4.5 para o mesmo par.	1. Sucesso. 2. A nota no DataModel deve ser 4.5.	Positivo (Update)
N01	Nota Inválida: Acima do Limite	1. Submeter avaliação 5.1.	1. Retornar erro de validação (ex: InvalidRatingException). Nenhuma alteração no DataModel.	Negativo
N02	Nota Inválida: Abaixo do Limite	1. Submeter avaliação -0.1.	1. Retornar erro de validação. Nenhuma alteração no DataModel.	Negativo
N03	User ID ou Book ID Inválido (Ex: null/não numérico)	1. Tentar submeter avaliação com

IDs inválidos (Ex: user_id=null, book_id="abc"). | 1. Retornar erro de validação (Ex: IllegalArgumentException). Nenhuma alteração no DataModel. | Negativo (Invalidez de Entrada) |

Time estimate: 0h **Time spent:** 0h

US3: Sugestão de Livros para Clientes Regulares (Valor Médio)

Green

FEATURE: Sugestão de Livros para Clientes Regulares (Valor Médio)

Como um Cliente Regular (Visitante ou Assinante)
EU QUERO receber 5 sugestões de livros baseadas no meu perfil de consumo
PARA QUE eu possa encontrar novos títulos relevantes, exibindo o seu Valor Médio de Venda.

Regra de Negócio (Preço):

- O preço exibido é o **Valor Médio de Venda Histórica**, calculado a partir da média dos preços de todas as unidades vendidas.

Cenário Principal: Retorno de 5 recomendações com o Valor Médio correto

CENÁRIO: O sistema deve retornar 5 livros recomendados com o preço calculado corretamente
DADO QUE o DataModel do Mahout está populado com as preferências dos clientes
E o algoritmo de similaridade definido é o PearsonCorrelationSimilarity
E o mecanismo de recomendação é o GenericUserBasedRecommender com NearestNUserNeighborhood
QUANDO eu (como Cliente) solicito 5 recomendações
ENTÃO o sistema DEVE invocar a função getRecommendationByUsers(c_id)
E o sistema DEVE retornar uma lista de até 5 objetos "Livro"
E a lista DEVE conter livros que o Cliente não avaliou anteriormente
E cada "Livro" retornado DEVE exibir o seu Valor Médio, calculado como a **Média dos preços de todas as unidades vendidas** historicamente.

Critérios de Aceite

- A implementação deve utilizar a estrutura e as classes fornecidas:
  -- Similaridade: PearsonCorrelationSimilarity
  -- Vizinhança: NearestNUserNeighborhood
  -- Recomendador: GenericUserBasedRecommender
- A função getRecommendationByUsers(int c_id) deve ser implementada no núcleo (Bookmarket).
- O sistema deve calcular e anexar o campo Valor Médio ao objeto Book, conforme a regra: Média dos preços de todas as unidades vendidas (isto implica na necessidade de uma estrutura de dados de "Histórico de Vendas" que armazene pares (book_id, preço_venda)).
- O número de recomendações deve ser 5 (ou o máximo possível se menos de 5 estiverem disponíveis).
- O algoritmo Mahout deve ser configurado para ignorar itens já avaliados pelo cliente.

Casos de Teste

ID	Descrição	Passos de Execução	Resultado Esperado	Tipo de Teste
P01	Recomendação Padrão	1. Cliente solicita 5 sugestões.	1. Retornar 5 livros não avaliados, com Valor Médio correto.	Positivo
P02	Recomendações Insuficientes	1. Cliente solicita 5 sugestões, mas apenas 2 são possíveis.	1. Retornar 2 livros não avaliados, com Valor Médio correto.	Positivo (Edge Case)
P03	Cálculo do Valor Médio	1. Verificar o Valor Médio para o livro A (vendido a R$10 e R$20).	1. Valor Médio de R$15.00 deve ser exibido.	Positivo (Cálculo)
N01	Cliente Sem Preferências (novo)	1. Cliente novo (sem avaliações) solicita 5 sugestões.	1. Retornar Lista Vazia (ou tratamento específico).	Negativo (Ausência de Dados)

| N02 | Livro Já Avaliado Recomendado | 1. Verificar se um livro com nota 5.0 está na lista de recomendações. | 1. O livro não deve estar na lista. | Negativo (Filtro) |

Time estimate: oh **Time spent:** oh

US4: Sugestão de Livros para Assinantes (Menor Valor)

Green

FEATURE: Sugestão de Livros para Assinantes (Menor Valor)

Como um Assinante
EU QUERO receber 5 sugestões de livros baseadas no meu perfil de consumo
PARA QUE eu possa encontrar novos títulos relevantes, exibindo o seu Menor Valor Promocional.

Regra de Negócio (Preço):

- O preço exibido é o **Menor Valor Disponível**, que é o **preço promocional do livro naquele momento**. Se não houver promoção, será o preço de varejo padrão.

Cenário Principal: Retorno de 5 recomendações com o Menor Valor correto

CENÁRIO: O sistema deve retornar 5 livros recomendados com o preço promocional
DADO QUE o DataModel do Mahout está populado com as preferências dos assinantes
E eu estou identificado como um Assinante
QUANDO eu (como Assinante) solicito 5 recomendações
ENTÃO o sistema DEVE retornar uma lista de até 5 objetos "Livro"
E a lista DEVE conter livros que o Assinante não avaliou anteriormente
E cada "Livro" retornado DEVE exibir o seu Menor Valor, que é o **preço promocional do livro naquele momento**.

Critérios de Aceite

- A lógica de recomendação do Mahout (PearsonCorrelationSimilarity, GenericUserBasedRecommender, NearestNUserNeighborhood) deve ser a mesma da US3.
- O sistema deve ser capaz de distinguir entre Cliente (US3) e Assinante (US4) para aplicar a regra de preço correta.
- O sistema deve calcular e anexar o campo Menor Valor ao objeto Book, conforme a regra: Preço promocional do livro naquele momento (isto implica na necessidade de uma estrutura de dados de "Estoque/Ofertas Atuais" que armazene o book_id e o preço promocional vigente, se houver).
- Se o livro não tiver preço promocional, o menor valor deve ser o preço de tabela padrão.

Casos de Teste

ID	Descrição	Passos de Execução	Resultado Esperado	Tipo de Teste
P01	Recomendação Padrão	1. Assinante solicita 5 sugestões.	1. Retornar 5 livros não avaliados, com Menor Valor Promocional correto.	Positivo
P02	Cálculo do Menor Valor (Com Promoção)	1. Verificar o Menor Valor para o livro B (Preço Tabela: R$50; Promoção: R$40).	1. Menor Valor de R$40.00 deve ser exibido.	Positivo (Cálculo)
P03	Cálculo do Menor Valor (Sem Promoção)	1. Verificar o Menor Valor para o livro C (Preço Tabela: R$35; Sem Promoção).	1. Menor Valor de R$35.00 deve ser exibido.	Positivo (Cálculo)
N01	Teste de Classificação	1. Cliente Padrão executa a US4 (Recursos de Assinante).	1. O sistema deve retornar um erro de permissão ou uma lista vazia/diferenciada (não deve retornar o recurso Premium).	Negativo (Autorização)

Time estimate: oh **Time spent:** oh

Épico 2: Estrutura inicial [13 pontos] Yellow

Time estimate: oh **Time spent:** oh

- [ ] T104 Criar InMemoryBookRepository (Estrutura de Maps) [3 pontos]
- [ ] T105 Implementar DataPopulator e lógica de geração de dados Micro Scale [5 pontos]
- [ ] T106 Setup do Mahout (instância de DataModel e RecommenderService) [5 pontos]

Épico 3: Cadastros principais [6 pontos] Yellow

Time estimate: oh **Time spent:** oh

- [ ] T201 US1: Implementar getBestsellers(n) (Lógica de contagem e ordenação) [3 pontos]
- [ ] T202 US2: Implementar updateRating() (Validação e registro de preferência) [3 pontos]

Épico 4: Motor de recomendação [16 pontos] Yellow

Time estimate: oh **Time spent:** oh

- [ ] T203 US3/US4: Implementar lógica de Preços (getAverageSalePrice, getLowestAvailablePrice) [5 pontos]
- [ ] T204 US3/US4: Implementar getRecommendationByUsers(userId, isSubscriber) (Mahout Core Logic) [8 pontos]
- [ ] T205 US3/US4: Implementar Lógica de Fallback (Clientev Novo -> Bestsellers) [3 pontos]

In execution (1 task)

Épico 1: Setup [8 pontos] Yellow

Time estimate: oh **Time spent:** oh

- [ ] T101 Configurar projeto Java/Maven e dependências (pom.xml) [1 ponto]
- [ ] T102 Criar Entidades de Domínio (Book.java, User.java) [2 pontos]
- [ ] T103 Criar Interfaces do Repositório (Contratos para Repositórios) [2 pontos]
- [ ] T104 Criar InMemoryBookRepository (Estrutura de Maps) [3 pontos]

Done (0 tasks)