Encerramento da Sprint 6

(Cerimônia de Retrospectiva da Sprint Encerrada (Gráfico de burndown da sprint consolidada (sprints 1,2,3,4,5,6)) está na página 07 e Cerimônia de Encerramento da Sprint:está na página 08)

Integrantes: Gabriel Reinhardt Dos Reis, Leonardo Bampi Rech, Kauana Braz, Johann Manfro

Cerimônia Revisão da Sprint Encerrada:

- O que foi entregue para o cliente?

Concluiu-se a ferramenta de visualização de localização de mercados parceiros, onde indica-se o mercado para usuário optar, e o início do desenvolvimento da ferramenta de indicação de recompra de produto, sem que esta estivesse funcional.

- Qual foi a avaliação do cliente?

O cliente observou o funcionamento da ferramenta de visualização de localização de mercados parceiros, não apenas observando o funcionamento da ferramenta, como também acompanhando os testes e o próprio cliente fazendo uso da ferramenta. Não foi relevante ao cliente que a ferramenta de indicação de recompra de produto não estivesse funcional nem passível de testes.

- Faça um documento do código que foi desenvolvido. Neste documento, demonstre o trabalho com imagens das telas, do projeto, do código.

```
===== MENU ADMINISTRADOR =====
1. Criar administrador
2. Cadastrar mercado
3. Alterar mercado
4. Excluir mercado
5. Listar usuários
6. Logout
Escolha uma opção: 3

--- Mercados Cadastrados ---
1. andreazza
Escolha o número do mercado para alterar: 1
O que deseja alterar?
1. Nome
2. CEP
3. Complemento
4. Localizações
Opção: 4

--- Localizações de andreazza ---
1. avenida julho de castilhos, Nº: 123 (lat: -12385746,000000, long: 908821312,000000)
Escolha a localização para alterar: 1
Novo endereço: avenida sao joao
Novo número: 12345
Nova latitude: -123128321
Nova longitude: -9478342
Localização atualizada.
```

```java
private void alterarMercado() {
    if (mercados.isEmpty()) {
        System.out.println("Nenhum mercado cadastrado.\n");
        return;
    }
    System.out.println("\n--- Mercados Cadastrados ---");
    for (int i = 0; i < mercados.size(); i++) {
        System.out.printf("%d. %s%n", i + 1, mercados.get(i).getNome());
    }
    System.out.print("Escolha o número do mercado para alterar: ");
    int idx = Integer.parseInt(scanner.nextLine().trim()) - 1;
    if (idx < 0 || idx >= mercados.size()) {
        System.out.println("Índice inválido.\n");
        return;
    }
    Mercado m = mercados.get(idx);

    System.out.println("O que deseja alterar?");
    System.out.println("1. Nome");
    System.out.println("2. CEP");
    System.out.println("3. Complemento");
    System.out.println("4. Localizações");
    System.out.print("Opção: ");
    String op = scanner.nextLine().trim();

    switch (op) {
        case "1":
            System.out.print("Novo nome: ");
            m.setNome(scanner.nextLine().trim());
            System.out.println("Nome atualizado.\n");
            break;
        case "2":
            System.out.print("Novo CEP: ");
            m.setCep(scanner.nextLine().trim());
            System.out.println("CEP atualizado.\n");
            break;
        case "3":
            System.out.print("Novo complemento: ");
            m.setComplemento(scanner.nextLine().trim());
            System.out.println("Complemento atualizado.\n");
            break;
        case "4":
            if (m.getLocalizacoes().isEmpty()) {
                System.out.println("Não há localizações cadastradas.\n");
                return;
            }
            System.out.println("\n--- Localizações de " + m.getNome() + " ---");
            for (int i = 0; i < m.getLocalizacoes().size(); i++) {
                Mercado.Localizacao loc = m.getLocalizacoes().get(i);
                System.out.printf("%d. %s, Nº: %s (lat: %.6f, long: %.6f)%n",
                    i + 1, loc.getEndereco(), loc.getNumero(), loc.getLatitude(), loc.getLongitude());
            }
            System.out.print("Escolha a localização para alterar: ");
            int locIdx = Integer.parseInt(scanner.nextLine().trim()) - 1;
            if (locIdx < 0 || locIdx >= m.getLocalizacoes().size()) {
                System.out.println("Índice inválido.\n");
                return;
            }
            System.out.print("Novo endereço: ");
            String novoEnd = scanner.nextLine().trim();
            System.out.print("Novo número: ");
            String novoNum = scanner.nextLine().trim();
            System.out.print("Nova latitude: ");
            double novaLat = Double.parseDouble(scanner.nextLine().trim());
            System.out.print("Nova longitude: ");
            double novaLon = Double.parseDouble(scanner.nextLine().trim());
            m.updateLocalizacao(locIdx, novoEnd, novoNum, novaLat, novaLon);
            System.out.println("Localização atualizada.\n");
            break;
        default:
            System.out.println("Opção inválida.\n");
    }
```

```java
public void addLocalizacao(String endereco, String numero, double latitude, double longitude) {
    this.localizacoes.add(new Localizacao(endereco, numero, latitude, longitude));
}

public void updateLocalizacao(int index, String endereco, String numero, double latitude, double longitude) {
    Localizacao loc = this.localizacoes.get(index);
    loc.setEndereco(endereco);
    loc.setNumero(numero);
    loc.setLatitude(latitude);
    loc.setLongitude(longitude);
}

public void removeLocalizacao(int index) {
    this.localizacoes.remove(index);
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Mercado: ").append(nome).append("\n");
    for (Localizacao loc : localizacoes) {
        sb.append("Endereço: ").append(loc.getEndereco())
          .append(" | Nº: ").append(loc.getNumero())
          .append(" | Latitude: ").append(loc.getLatitude())
          .append(" | Longitude: ").append(loc.getLongitude())
          .append("\n");
    }
    return sb.toString();
}
```

```java
package TopLista;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class Item {
    private String nome;
    private int quantidade;
    private double preco;
    private String descricao;
    private LocalDate validade;

    private static List<Item> historicoCompras = new ArrayList<>();

    public Item(String nome, int quantidade, double preco, String descricao, LocalDate validade) {
        this.nome = nome;
        this.quantidade = quantidade;
        this.preco = preco;
        this.descricao = descricao;
        this.validade = validade;

        historicoCompras.add(this);
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public double getPreco() {
```

```java
    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public LocalDate getValidade() {
        return validade;
    }

    public void setValidade(LocalDate validade) {
        this.validade = validade;
    }

    public static List<Item> recomendarItens() {
        List<Item> recomendados = new ArrayList<>();
        for (Item item : historicoCompras) {
            boolean jaExiste = recomendados.stream()
                    .anyMatch(i -> i.getNome().equalsIgnoreCase(item.getNome()));
            if (!jaExiste) {
                recomendados.add(item);
            }
        }
        return recomendados;
    }

    @Override
    public String toString() {
        return String.format("Item: %s | Quantidade: %d | Preço: R$%.2f | Validade: %s | Descrição: %s",
                nome, quantidade, preco, validade, descricao);
    }
}
```

```
Login bem-sucedido! Bem-vindo, admin.

===== MENU ADMINISTRADOR =====
1. Criar administrador
2. Cadastrar mercado
3. Alterar mercado
4. Excluir mercado
5. Listar usuários
6. Cadastrar item
7. Alterar item
8. Listar itens
9. Logout
Escolha uma opção: 6
Nome do item: margarina
Quantidade: 100
Preço: 6.90
Descrição: marca qually
Validade (AAAA-MM-DD): 2025-07-19
Item cadastrado com sucesso!

===== MENU ADMINISTRADOR =====
1. Criar administrador
2. Cadastrar mercado
3. Alterar mercado
4. Excluir mercado
5. Listar usuários
6. Cadastrar item
7. Alterar item
8. Listar itens
9. Logout
Escolha uma opção: 8

--- Itens Cadastrados ---
1. Item: margarina | Quantidade: 100 | Preço: R$6,90 | Validade: 2025-07-19 | Descrição: marca qually
------------------------
```

```java
private void cadastrarItem() {
    System.out.print("Nome do item: ");
    String nome = scanner.nextLine().trim();
    System.out.print("Quantidade: ");
    int quantidade = Integer.parseInt(scanner.nextLine().trim());
    System.out.print("Preço: ");
    double preco = Double.parseDouble(scanner.nextLine().trim());
    System.out.print("Descrição: ");
    String descricao = scanner.nextLine().trim();
    System.out.print("Validade (AAAA-MM-DD): ");
    LocalDate validade = LocalDate.parse(scanner.nextLine().trim());

    Item novoItem = new Item(nome, quantidade, preco, descricao, validade);
    itens.add(novoItem);

    System.out.println("Item cadastrado com sucesso!\n");
}

private void alterarItem() {
    if (itens.isEmpty()) {
        System.out.println("Nenhum item cadastrado.\n");
        return;
    }

    System.out.println("--- Itens Cadastrados ---");
    for (int i = 0; i < itens.size(); i++) {
        System.out.printf("%d. %s%n", i + 1, itens.get(i).getNome());
    }

    System.out.print("Escolha o número do item para alterar: ");
    int idx = Integer.parseInt(scanner.nextLine().trim()) - 1;

    if (idx < 0 || idx >= itens.size()) {
        System.out.println("Índice inválido.\n");
        return;
    }

    Item item = itens.get(idx);

    System.out.println("1. Nome\n2. Quantidade\n3. Preço\n4. Descrição\n5. Validade");
    System.out.print("Escolha o campo para alterar: ");
    String op = scanner.nextLine().trim();

    switch (op) {
        case "1": System.out.print("Novo nome: "); item.setNome(scanner.nextLine().trim()); break;
```

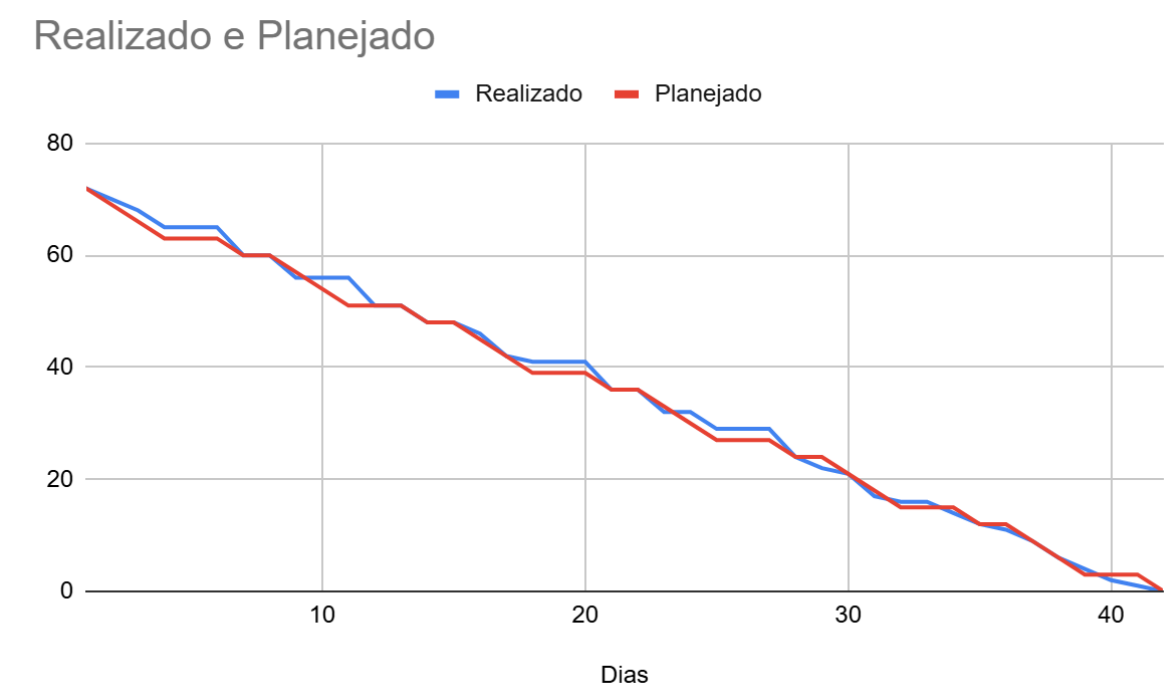
```
        switch (op) {
            case "1": System.out.print("Novo nome: "); item.setNome(scanner.nextLine().trim()); break;
            case "2": System.out.print("Nova quantidade: "); item.setQuantidade(Integer.parseInt(scanner.nextLine().t
            case "3": System.out.print("Novo preço: "); item.setPreco(Double.parseDouble(scanner.nextLine().trim()));
            case "4": System.out.print("Nova descrição: "); item.setDescricao(scanner.nextLine().trim()); break;
            case "5": System.out.print("Nova validade (AAAA-MM-DD): "); item.setValidade(LocalDate.parse(scanner.nextL
            default: System.out.println("Opção inválida."); return;
        }

        System.out.println("Item alterado com sucesso.\n");
    }

    private void listarItens() {
        if (itens.isEmpty()) {
            System.out.println("Nenhum item cadastrado.\n");
            return;
        }
        System.out.println("\n--- Itens Cadastrados ---");
        for (int i = 0; i < itens.size(); i++) {
            System.out.printf("%d. %s\n", i + 1, itens.get(i));
        }
        System.out.println("-----------------------\n");
    }

    private void visualizarItens() {
        if (itens.isEmpty()) {
            System.out.println("Nenhum item disponível.\n");
            return;
        }

        System.out.println("\n--- Itens Disponíveis ---");
        for (Item item : itens) {
            System.out.println(item);
        }
        System.out.println();
    }

    private void verRecomendacoes() {
        List<Item> recomendados = Item.recomendarItens();
        if (recomendados.isEmpty()) {
            System.out.println("Nenhuma recomendação disponível.\n");
        } else {
            System.out.println("\n📌 Recomendações com base nas últimas compras:");
            for (Item item : recomendados) {
```

```
private void verRecomendacoes() {
    List<Item> recomendados = Item.recomendarItens();
    if (recomendados.isEmpty()) {
        System.out.println("Nenhuma recomendação disponível.\n");
    } else {
        System.out.println("\n📌 Recomendações com base nas últimas compras:");
        for (Item item : recomendados) {
            System.out.println("- " + item.getNome());
        }
    }
    System.out.println();
}
```

```
===== MENU USUÁRIO =====
1. Visualizar mercados
2. Ver itens disponíveis
3. Registrar item comprado
4. Ver recomendações
5. Logout
Escolha uma opção: 3

--- Itens Disponíveis para Comprar ---
1. margarina
2. macarrao
Escolha o número do item comprado: 2
Informe a quantidade comprada: 4
Item registrado no histórico de compras!

===== MENU USUÁRIO =====
1. Visualizar mercados
2. Ver itens disponíveis
3. Registrar item comprado
4. Ver recomendações
5. Logout
Escolha uma opção: 4

📌 Recomendações com base em suas compras:
- macarrao | Quantidade recomendada: 4,0 (baseado em 1 compras)

===== MENU USUÁRIO =====
1. Visualizar mercados
2. Ver itens disponíveis
3. Registrar item comprado
4. Ver recomendações
5. Logout
Escolha uma opção:
```

Cerimônia de Retrospectiva da Sprint Encerrada:

Gráfico de burndown da sprint consolidada (sprints 1,2,3,4,5,6) :



Cerimônia de Encerramento da Sprint:

Nome do membro da equipe	Qual foi minha contribuição na sprint ?	Quais impedimentos aconteceram e não foram removidos?	O que deu certo e devemos manter?	O que pode ser melhorado na próxima sprint?
Gabriel	Modelagem dos dados de consumo usados na funcionalidade de sugestão de recompra.	Dificuldade em identificar padrões de consumo sem uma base histórica de dados adequada.	A estruturação clara das histórias facilitou o desenvolvimento e testes.	Iniciar a sprint com todos os acessos e integrações externas validadas.
Kauana	Testes da funcionalidade de sugestão de mercados parceiros no mapa.	Falta de acesso à API de localização dos mercados parceiros, o que atrasou os testes.	O time conseguiu manter um bom ritmo de entrega, completando as histórias previstas.	Reservar tempo no planejamento para testes com usuários e validação real.
Johann	Criação das classes principais para a sugestão de recompra e cumprimento dos requisitos	Falta de documentação técnica dos serviços externos utilizados.	A priorização de funcionalidades com base no valor para o usuário funcionou bem.	Antecipar testes de integração com APIs externas.
Leonardo	Definição e priorização da história de usuário sobre a visualização dos mercados no mapa.	Comunicação limitada entre o time (Product Owner x Desenvolvedores) sobre regras da funcionalidade de sugestão.	A colaboração entre membros durante os refinamentos foi eficiente.	Melhorar a comunicação entre P.O. e desenvolvedores quanto aos critérios de aceitação.