

**PROJETO INTERDISCIPLINAR**

**Gestão de estoque**

**De supermercado**

Programa para gestão comercial

**Alunos:**

|  |  |
| --- | --- |
| **RGM** | **Nome** |
| 29278791 | Jonas da Silva Luiz |
| 29408547 | Rodolpho Lucciano Galerani |
| 29308119 | Davyd Brito |
| 30009456 | Milton andrade eufrasio junior |
| 30009332 | Yvan Marques de Michelli Sallai |
| 30344905 | Daniel alvaro da Silva |
| 32067232 | Felipe Ferreira Fonseca |

São Paulo

2023

**UNIVERSIDADE CRUZEIRO DO SUL**

**PROJETO INTERDISCIPLINAR**

**Gestão de estoque**

**De supermercado**

Programa para gestão comercial

## Trabalho apresentado como parte do requisito para aprovação na Disciplina de Projeto Interdisciplinar do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Cruzeiro do Sul.

**Orientadores:** Prof. Artur Ubaldo Marques, Prof. Edidio Rubens Dantas Lima e Prof. Paulo Vilela de Melo

São Paulo

2023**Sumário**

[1. Apresentação: 4](#_Toc129875560)

[1.1 Justificativa e Motivação 4](#_Toc129875561)

[2 Requisitos de análise e projeto de sistemas i 4](#_Toc129875562)

[3 Requisitos de engenharia de software 4](#_Toc129875563)

[4 Consideração finais 4](#_Toc129875564)

[5 BIBLIOGRAFIA 4](#_Toc129875565)

[APÊNDICE 1 – Cronograma de entrega de atividades. 5](#_Toc129875566)

# 1. Apresentação:

## 1.1 Justificativa e Motivação

A relevância deste trabalho e de extrema importância para nos estudantes do curso de ADS, pondo a prova nossos conhecimentos das matérias de Engenharia de Software e Análise e Projeto de Sistemas, a magnitude em questões preparatórias e crucial para nos sentirmos preparados para o mercado de trabalho.

A gestão de supermercados é um setor que exige muita eficiência e agilidade na tomada de decisões, sendo que a tecnologia pode ser uma grande aliada nesse processo. Por isso, desenvolver um software de gestão de supermercados pode trazer muitos benefícios para os negócios desse ramo, como a otimização dos processos de vendas, controle de estoque, gestão de compras, entre outras atividades.

Além disso, a criação de um software de gestão de supermercados pode ser uma excelente oportunidade para estudantes e profissionais de tecnologia desenvolverem habilidades em programação, análise de requisitos e design de software. Trata-se de um projeto que envolve muitas áreas do conhecimento, como a administração, a tecnologia da informação, a engenharia de software, a matemática, entre outras, o que torna essa uma excelente oportunidade para o aprimoramento profissional.

Além disso, a criação de um software de gestão de supermercados também pode trazer benefícios para o mercado, uma vez que a adoção de tecnologias de gestão mais avançadas pode trazer mais eficiência e competitividade para as empresas desse setor. Assim, esse projeto pode ser uma oportunidade para criar soluções inovadoras e impactar positivamente a sociedade.

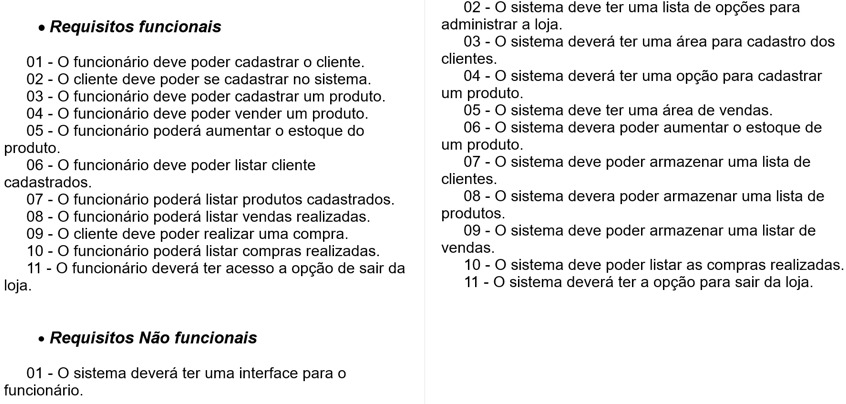
Por todas essas razões, a escolha do tema de gestão de supermercados para o desenvolvimento de um software pode ser uma excelente oportunidade para aprimorar a formação profissional e desenvolver habilidades práticas em diversas áreas, além de contribuir para o desenvolvimento do mercado e da sociedade em geral.

# 2 Requisitos de análise e projeto de sistemas i

Durante a análise, os requisitos funcionais e não funcionais são revisados em detalhes para garantir que sejam compreendidos corretamente. Isso inclui a identificação de casos de uso, diagrama de classes, a definição de interfaces e a análise de requisitos não funcionais, como desempenho, segurança e usabilidade.

Uma vez que os requisitos foram analisados e compreendidos, a etapa de projeto inicia-se. O projeto de sistema envolve a definição da arquitetura, a especificação dos componentes do sistema, o design das interfaces, a estruturação dos dados, a definição de algoritmos e a definição da estratégia de implantação, durante o processo de projeto, é necessário garantir que cada requisito identificado seja atendido adequadamente.

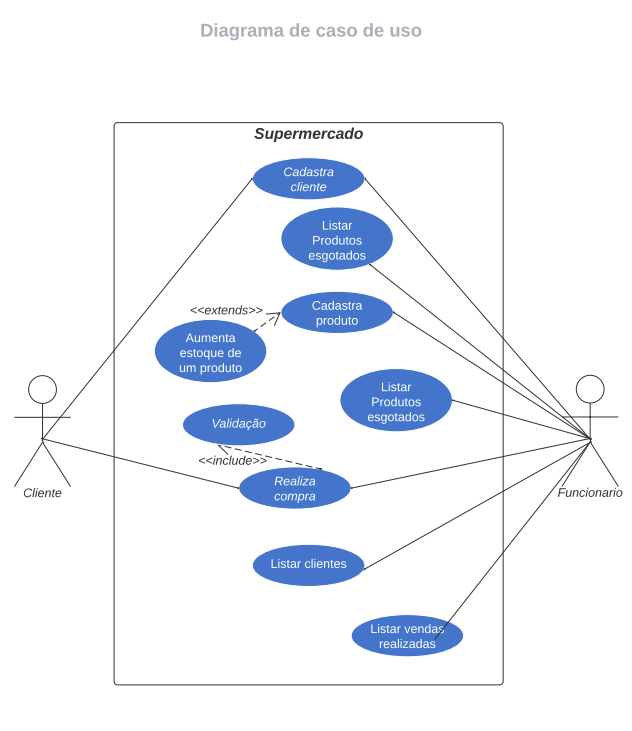
Um dos requisitos adicionados ao projeto foi a tabela de requisitos funcionais e não funcionais, as tabelas de requisitos funcionais e não funcionais são ferramentas de documentação que trazem informações com estrutura para selecionar os quatro principais requisitos solicitados e que serão de ótima agregação no projeto.



Obtivemos também o uso do diagrama de caso de uso que também se tornou uma forma de modelar e utilizar no desenvolvimento do software para descrever interações entre atores (usuários, sistemas externos) e o sistema em questão.

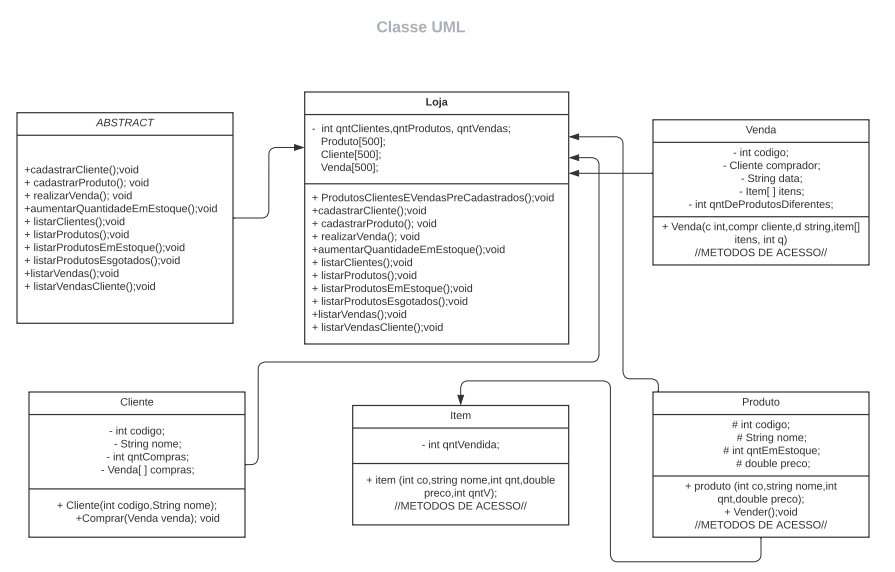
Os casos de uso trouxe como objetivo principal descrever o comportamento do sistema do ponto de vista do usuário, focando então nas funcionalidades que o sistema deve oferecer para atender às necessidades dos usuários.

Ele ajuda-nos na captura dos requisitos funcionais do sistema, descrevendo os diversos cenários de uso em que o sistema será empregado, assim formando uma narrativa que descreve uma sequência de passos que um ator realiza ao interagir com o sistema.



Tendo em vista a utilização também do diagrama de classe, no qual proporcionou uma representação visual das classes, seus atributos, métodos e relacionamentos, sendo uma das partes fundamental do projeto de software, trazendo uma visualização da estrutura do sistema, especificação de métodos e comportamentos e facilitando a comunicação e colaboração entre os membros do grupo, o diagrama de classe proporciona uma visão estruturada e visualmente expressiva da arquitetura e do design de um projeto.

Ele ajuda a identificar as classes e seus atributos, especificar os relacionamentos, definir os comportamentos das classes e fornecer uma base para a implementação do software.

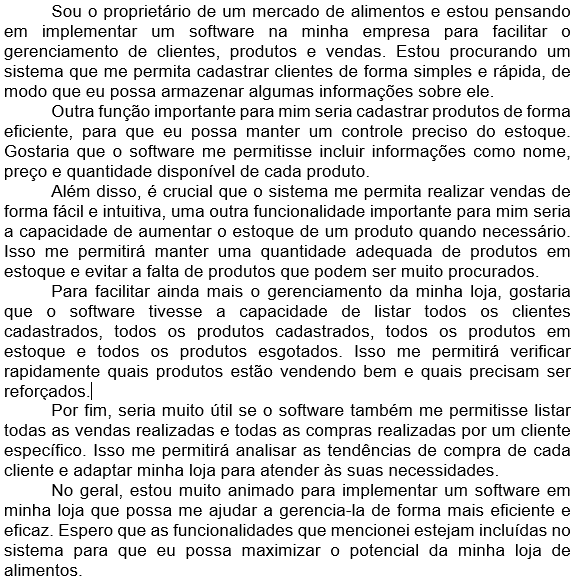


# 3 Requisitos de engenharia de software

O cumprimento dos requisitos do projeto em Engenharia de Software envolveu principalmente, garantir que o software atenda às necessidades e expectativas dos usuários, conforme definido na etapa de levantamento de requisitos.

Entretanto Decidimos começar pelo Desenvolvimento da História de Usuário: As histórias de usuário são uma técnica para capturar requisitos funcionais de forma ágil. O processo incluiu a identificação de usuários e personas, definição do formato da história, criação das histórias, priorização, detalhamento com critérios de aceitação, estimativa de esforço e alinhamento com as necessidades dos usuários para um desenvolvimento eficaz.

Optamos por usar o formato "Como um [tipo de usuário], eu quero [realizar uma ação] para [atingir um objetivo ou obter um benefício]". Esse formato nos ajuda a descrever claramente o que o usuário deseja realizar como e mostrado no corte a seguir.



Uma etapa muito importante para nosso projeto foi o desenvolvimento dos cards de usuário, os cards de usuário são uma ferramenta essencial para a metodologia ágil e nos ajudam a definir, priorizar e acompanhar as funcionalidades do nosso software. Foram desenvolvidos desta forma!



1. Levantamento de requisitos:

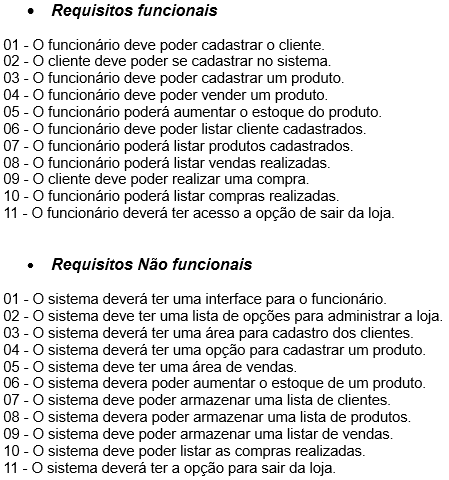
🔘 No início do projeto, realizamos sessões de levantamento de requisitos com os stakeholders, como usuários finais e especialistas no domínio do sistema.

🔘 Durante essas sessões, identificamos as necessidades e expectativas dos usuários, suas principais tarefas e os requisitos funcionais e não funcionais do software.

1. Identificação de funcionalidades:

🔘 Com base nos requisitos levantados, identificamos as funcionalidades-chave do sistema.

🔘 Cada funcionalidade é uma unidade de valor para o usuário e é representada por um card.



1. Definição dos atributos do card:

🔘 Cada card de usuário é composto por vários atributos, incluindo:

🔘 Título: Nome descritivo da funcionalidade.

🔘 Descrição: Detalhes sobre a funcionalidade e o valor que ela agrega.

🔘 Critérios de aceitação: Condições específicas que devem ser cumpridas para considerar a funcionalidade concluída.

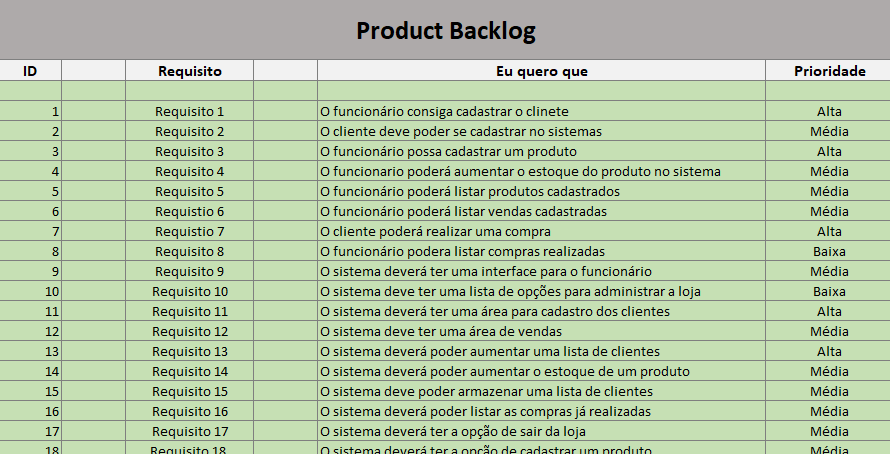
🔘 Pontuação de esforço: Estimativa de esforço necessário para implementar a funcionalidade (por exemplo, usando a técnica de pontos de história).

1. Priorização dos cards:

🔘 Após a definição dos cards de usuário, realizamos uma sessão de priorização com a equipe.

🔘 Durante essa sessão, avaliamos a importância de cada funcionalidade e atribuímos uma prioridade relativa a cada card (Que está na lista de backlog).

🔘 Isso nos ajuda a definir a ordem de implementação e garantir que as funcionalidades mais valiosas sejam desenvolvidas primeiro.

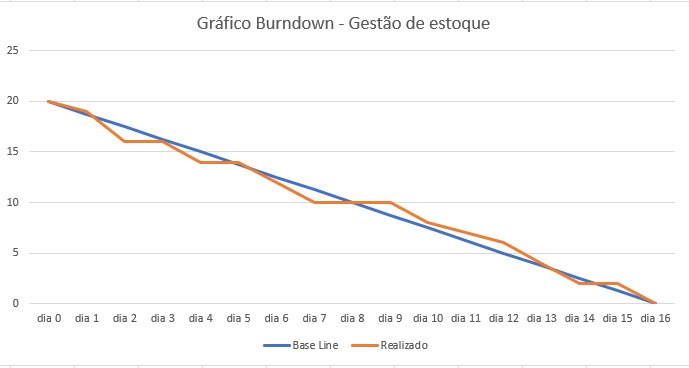


1. Acompanhamento e atualização dos cards:

🔘 Durante o desenvolvimento do projeto, mantemos um Gráfico de burndown para acompanhar o progresso dos cards de usuário.

🔘 À medida que as funcionalidades são implementadas, movemos os pontos do gráfico frequentemente.

🔘 À medida que novas informações ou requisitos surgem, atualizamos os cards para refletir as alterações necessárias.



Esse foi o processo de desenvolvimento dos cards de usuário,lista de backlog, gráfico de burndown, e requisitos funcionais e não funcionais para o nosso projeto interdisciplinar. Essa abordagem nos permite ter uma visão clara das funcionalidades e prioridades, além de acompanhar o progresso do desenvolvimento. Todos foram ferramentas fundamentais e poderosas para garantir que o software atenda às necessidades dos usuários e seja entregue de forma iterativa e incremental.

Chegando na parte de desenvolvimento do nosso sistema de gestão de estoque de supermercado em Java, levando em consideração os princípios da programação orientada a objetos e da Engenharia de Software.

**Análise de requisitos:**

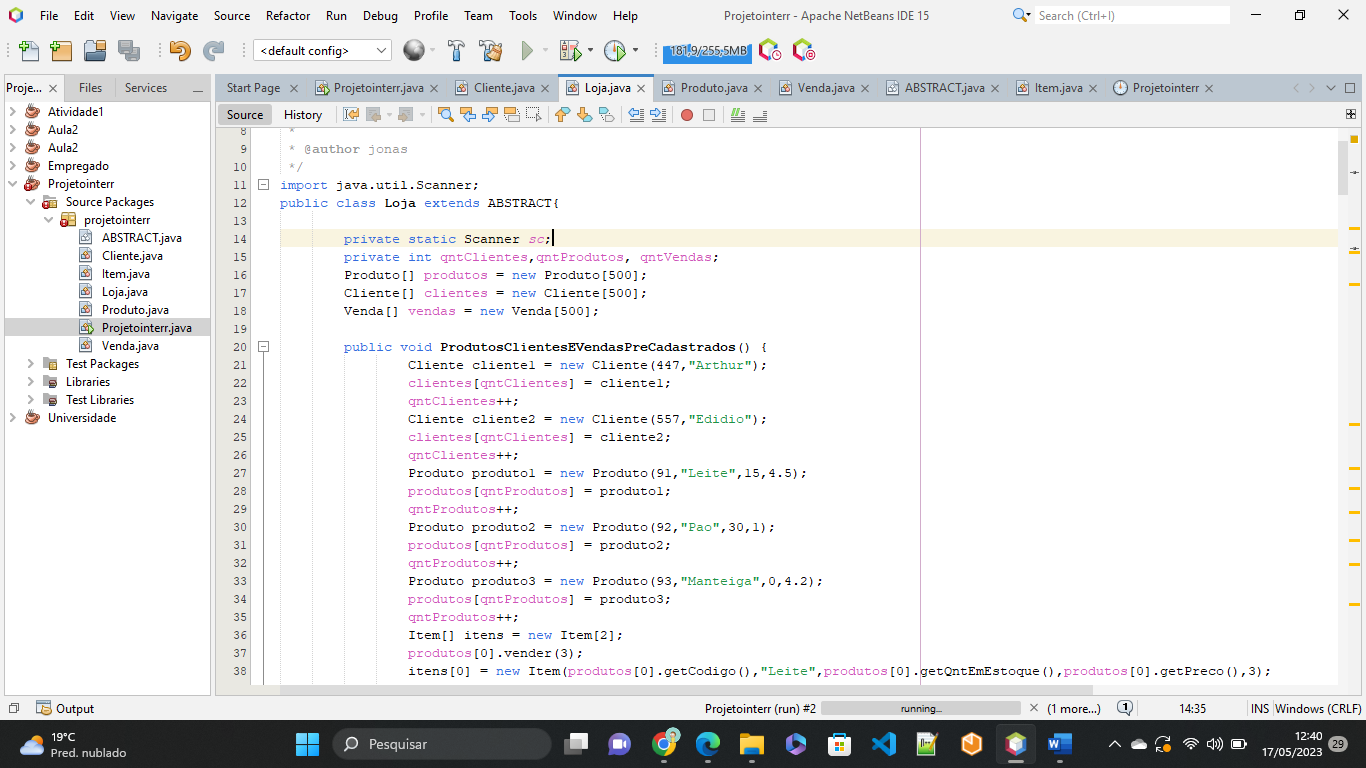
No início do projeto, realizamos uma análise detalhada dos requisitos do sistema de gestão de estoque. Identificamos as principais entidades envolvidas, como produtos, fornecedores, funcionários e clientes, e suas respectivas interações. Com base nessa análise, definimos as funcionalidades-chave do sistema, como registro de produtos, controle de estoque, compras, vendas e relatórios.

**Modelagem de classes:**

Utilizamos os conceitos da programação orientada a objetos para modelar as classes do sistema. Cada entidade identificada na análise de requisitos se tornou uma classe, com seus atributos e métodos correspondentes. Também aplicamos princípios como encapsulamento, herança e polimorfismo para obter um design flexível e extensível.

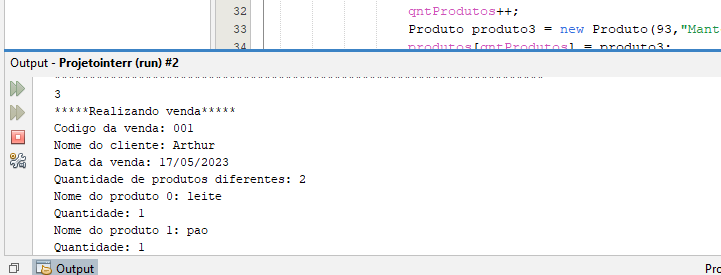
**Implementação em Java:**

Com a modelagem de classes pronta, iniciamos a implementação do sistema em Java. Utilizamos a linguagem Java por sua ampla adoção, suporte à programação orientada a objetos e vasta biblioteca padrão. Implementamos as classes, seus atributos e métodos, garantindo a consistência dos dados e a funcionalidade adequada do sistema.



**Testes unitários:**

Durante o desenvolvimento, aplicamos testes unitários para verificar a corretude do código. Criamos casos de teste para as diferentes funcionalidades do sistema, garantindo que as classes e métodos estejam produzindo os resultados esperados. Isso nos ajudou a identificar e corrigir problemas antes da integração das diferentes partes do sistema.



**Integração e testes de sistema:**

Após a conclusão da implementação das classes e suas funcionalidades, integramos as diferentes partes do sistema. Realizamos testes de sistema para garantir que todas as funcionalidades estejam funcionando corretamente em conjunto. Esses testes envolveram a interação entre as classes, a persistência dos dados e a correta atualização do estoque.

**Documentação e manutenção:**

Durante todo o processo de desenvolvimento, documentamos o sistema para facilitar o entendimento e a manutenção futura. Criamos documentação de código, manuais de uso e especificações técnicas para auxiliar os desenvolvedores e os usuários do sistema.

Esse foi o processo de desenvolvimento do nosso sistema de gestão de estoque de supermercado em Java, utilizando os princípios da programação orientada a objetos e da Engenharia de Software. Com uma análise adequada dos requisitos, uma modelagem eficiente, implementação cuidadosa, testes rigorosos e documentação adequada, conseguimos obter um sistema funcional, flexível e de fácil manutenção. Essas práticas nos permitiram atender às necessidades do supermercado e fornecer uma solução eficaz para a gestão de estoque.

Outro ponto muito importante de se explicar e de como foi o desenvolvimento da Sprint Backlog, as reuniões de revisão de sprint e uma retrospectiva para o nosso projeto interdisciplinar. Essas práticas são fundamentais para o sucesso de uma metodologia ágil, como o Scrum.

Desenvolvimento da Sprint Backlog:

🔘 Após o planejamento da sprint, a equipe se reuniu para desenvolver a Sprint Backlog.

🔘 A partir da lista de Backlog, que contém os requisitos priorizados, selecionamos os requisitos que seriam trabalhadas na sprint e as dividimos em tarefas menores.

🔘 Cada tarefa foi adicionada à Sprint Backlog, junto com uma estimativa de esforço e a pessoa responsável por sua execução.

Reuniões de Revisão de Sprint:

🔘 Ao final de cada sprint, realizamos a Reunião de Revisão de Sprint.

🔘Durante essa reunião, a equipe apresentou os requisitos concluídos e demonstrou as funcionalidades implementadas para todos os integrantes do grupo.

🔘Também foi uma oportunidade para receber feedback e fazer ajustes na lista de Backlog com base nas necessidades identificadas.

Retrospectiva da Sprint:

🔘 Logo após a Reunião de Revisão de Sprint, conduzimos uma retrospectiva para avaliar como foi a sprint e identificar melhorias para o próximo ciclo.

🔘Todos os membros da equipe foram convidados a compartilhar suas percepções e opiniões sobre o processo de desenvolvimento (em reuniões de intervalo de aula quase sempre), colaboração, comunicação e quaisquer problemas ou obstáculos enfrentados.

🔘 Com base nas discussões, identificamos pontos fortes e oportunidades de melhoria e definimos ações para implementar nas próximas sprints.

Essa foi a forma como desenvolvemos a Sprint Backlog, conduzimos as Reuniões de Revisão de Sprint e realizamos a retrospectiva para o nosso projeto interdisciplinar. Essas práticas ágeis nos ajudaram a planejar e acompanhar as atividades da sprint, receber feedback dos stakeholders e identificar oportunidades de melhoria contínua. Com um processo bem estruturado e a participação ativa da equipe, podemos otimizar nossas entregas e alcançar um desenvolvimento eficiente e alinhado às necessidades dos usuários.

# 4 Consideração finais

Durante a elaboração do projeto interdisciplinar envolvendo as disciplinas de Engenharia de Software e Análise de Projetos e Sistemas, encontramos tanto facilidades quanto dificuldades. Nesta seção final, destacaremos esses aspectos e forneceremos comentários adicionais pertinentes.

**Facilidades:**

Conhecimento adquirido: Ao longo do projeto, tivemos a oportunidade de aplicar os conhecimentos adquiridos durante as aulas de Engenharia de Software e Análise de Projetos e Sistemas. Isso inclui conceitos de desenvolvimento de software, metodologias ágeis, modelagem de sistemas, entre outros. Essa base de conhecimento nos ajudou a tomar decisões informadas durante o desenvolvimento do projeto.

Colaboração em equipe: Um dos pontos fortes do projeto foi a colaboração em equipe. Trabalhar em conjunto permitiu que aproveitássemos as habilidades individuais e compartilhássemos conhecimentos. A comunicação aberta e eficiente entre os membros da equipe ajudou a superar desafios e avançar de forma produtiva.

Metodologias ágeis: A adoção de metodologias ágeis, como o Scrum, trouxe benefícios significativos para o projeto. A flexibilidade e a capacidade de adaptação dessas metodologias nos permitiram responder rapidamente às mudanças de requisitos, realizar entregas incrementais e manter um fluxo contínuo de trabalho.

**Dificuldades:**

Gestão do tempo: Um dos desafios encontrados foi a gestão do tempo. Equilibrar as demandas do projeto com outros compromissos acadêmicos e pessoais exigiu uma boa organização e habilidades de gerenciamento do tempo. Encontramos alguns momentos em que o cronograma precisou ser ajustado para garantir a conclusão das tarefas dentro dos prazos estabelecidos.

Complexidade dos requisitos: Os requisitos do projeto apresentaram complexidades técnicas e funcionais, exigindo uma análise detalhada e uma compreensão profunda do domínio do problema. Essa complexidade adicionou desafios extras ao processo de desenvolvimento e exigiu um planejamento cuidadoso para garantir a implementação adequada das funcionalidades.

Coordenação e comunicação: Às vezes, a coordenação entre os membros da equipe e a comunicação efetiva foram desafios. Lidar com diferenças de opiniões, garantir a sincronização das atividades e manter todos os membros informados exigiu esforço adicional. No entanto, superamos essas dificuldades por meio de uma comunicação regular, reuniões de equipe e uso de ferramentas de colaboração.

Durante o projeto, aprendemos que a Engenharia de Software e a análise de projetos e sistemas são disciplinas complexas, que envolvem não apenas a aplicação de conhecimentos técnicos, mas também habilidades de gerenciamento, comunicação e colaboração. Além disso, compreendemos a importância de um planejamento adequado, da definição clara de requisitos e da realização de testes rigorosos para garantir a qualidade do sistema desenvolvido.

Também foi evidente a necessidade de um trabalho em equipe eficiente, com uma distribuição adequada de tarefas, compartilhamento de responsabilidades e apoio mútuo. Aprendemos que a diversidade de habilidades e experiências em uma equipe pode ser uma vantagem significativa na resolução de problemas complexos.

Por fim, reconhecemos a importância de uma abordagem iterativa e incremental no desenvolvimento de software, permitindo ajustes contínuos e melhorias ao longo do projeto. Essa abordagem nos ajudou a responder às mudanças de requisitos e a entregar um sistema funcional e de qualidade.

No geral, o projeto de Engenharia de Software e Análise de Projetos e Sistemas foi uma experiência enriquecedora, que nos permitiu aplicar os conhecimentos teóricos em um contexto prático. As facilidades encontradas e as dificuldades superadas contribuíram para nosso crescimento profissional e nos prepararam para enfrentar desafios futuros na área de desenvolvimento de software e análise de projetos e sistemas.

# 5 BIBLIOGRAFIA

MASCHIETTO, L. G. et al. Desenvolvimento de software com metodologias ágeis. Porto Alegre: SAGAH, 2021. (ebook).

PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021. (e-book).

SOMMERVILLE, I. Engenharia de software. 10. ed. São Paulo: Pearson, 2019. (e-book).

FOWLER, M. UML essencial: um breve guia para linguagem padrão. 3. ed. Porto Alegre: Bookman, 2011. (e-book)

KERZNER, H. Gestão de projetos: as melhores práticas. 4. ed. Porto Alegre: Bookman, 2020.

LEDUR, C. L. Análise e projeto de sistemas. Porto Alegre: Sagah, 2018.

SCHACH, S. R. Engenharia de software: os paradigmas clássico e orientado a objetos. 7. ed. Porto Alegre: AMGH, 2010 (e-book).

VETORAZZO, Adriana de Souza. Engenharia de software. Porto Alegre: Sagah Educação, 2018. (e-book).

•BEZERRA, E. Princípios de análise e projeto de sistemas com UML. 3. ed. Rio de Janeiro: Elsevier, 2015. (e-book).

•DENNIS, A.; WIXOM, B. H.; ROTH, R. M. Análise e projeto de sistemas. 5.ed. Rio de Janeiro: LTC, 2014. (e-book).

•FLYNN, M. J. Projeto de sistemas de computador: system-on-chip. Rio de Janeiro: LTC, 2014. (e-book)

•FOGGETTI, C. (Org). Gestão ágil de projetos. São Paulo: Pearson, 2015.

•PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 9.ed. Porto Alegre: AMGH, 2021. (e-book).

•SOMMERVILLE, I. Engenharia de software. 10. ed. São Paulo: Pearson,2019. (e-book).

•VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. Análise de pontos

# APÊNDICE 1 – Cronograma de entrega de atividades.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Descrição | Data | | Prazo do cronograma em semanas | | | | | | | | | | | | | | | | | |
| Início | Término |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | Definição do tema e planejamento inicial | 04/26 | 05/07 | P | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Análise dos requisitos  Do software | 05/05 | 05/10 | P |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Projeto da arquitetura  E detalhamento do Software | 05/05 | 05/18 | P |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Construção do  Software | 07/05 | 05/15 | P |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| R | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Testes de Software | 05/15 | 05/17 | P | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Entrega do projeto final e apresentação | 05/18 | 05/18 | P | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**OBS:**

1. **P =** previsto; **R =** realizado