UNIVERSIDADE FEDERAL DO PAMPA

Valmir Thume Junior

Desenvolvimento de um Sistema para Comércio Varejista Usando Personal Scrum

Alegrete
2024

Valmir Thume Junior

Desenvolvimento de um Sistema para Comércio Varejista Usando Personal Scrum

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Schepke

Alegrete
2024

VALMIR THUME JUNIOR

DESENVOLVIMENTO DE UM SISTEMA PARA COMÉRCIO VAREJISTA USANDO PERSONAL SCRUM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação

Dissertação defendida e aprovada em: 10 de dezembro de 2024.

Banca examinadora:

_____

Prof. Dr. Claudio Schepke

Orientador

UNIPAMPA

_____

Profa. Dra. Aline Vieira de Mello

UNIPAMPA

_____

Me. Deise Dall'Agnol Severo

UNIPAMPA

Assinado eletronicamente por **ALINE VIEIRA DE MELLO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 10/12/2024, às 19:13, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 10/12/2024, às 19:21, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Deise Dall'agnol Severo**, **Usuário Externo**, em 10/12/2024, às 22:40, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1598130** e o código CRC **9BB0A676**.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer ao meu orientador, o Professor Cláudio Schepke, que sempre se dispôs a me ajudar no que fosse preciso, sem a sua ajuda não seria possível concluir essa importante etapa da minha vida.

Porém não posso me esquecer das pessoas que tornaram possível chegar onde estou hoje, sempre apoiando e dando-me forças para continuar. Quero agradecer imensamente a minha família, meu pai Valmir, minha mãe Elaine, minha mãe do coração Márcia, minha namorada Giovana, meus irmãos Bianca, Nayane e Dyonatan, e meus avós, Ivo e Elourdes, Querino e Valéria e Paulo e Ana. Sem o apoio destas pessoas, nada disso seria possível.

Gostaria também de agradecer os meus grandes amigos e irmãos de longa data, Leonardo e Rodrigo, que estiveram comigo durante toda a graduação e sempre me auxiliaram no que fosse preciso. Agradecendo também ao meu grande amigo Gabriel, que mesmo com a distância sempre manteve a nossa parceria firme e forte.

Quero estender os agradecimentos a pessoas que já não estão mais entre nós, mas que foram extremamente fundamentais na minha formação como ser humano. Agradeço a minha bisavó Maria Suzana, minha avó Valéria e minha vó do coração, Thereza. Vocês jamais serão esquecidas!

RESUMO

A crescente introdução da tecnologia no ambiente corporativo trouxe à tona necessidades que até a alguns anos atrás não existiam. O setor varejista foi um dos principais setores afetados. Sistemas ERP (*Enterprise Resource Planning*) chegaram trazendo facilidade no gerenciamento de uma grande quantidade de processos que, até então, eram realizados manualmente. A empresa alvo deste trabalho está localizada no noroeste do Estado do Rio Grande do Sul. A mesma enfrenta um problema recorrente nos últimos anos por não possuir integração entre as distribuições de seu software de gestão empresarial, dificultando assim o trabalho de cadastramento de mercadorias, como também de outras tarefas, sendo seu principal efeito colateral a perda excessiva de tempo para realizá-la nas demais filiais. Para isso, este trabalho é motivado para solucionar este problema de forma a facilitar a tarefa realizada através do uso de um novo software capaz de compreender as lacunas deixadas pelo software legado da empresa, trazendo, assim, agilidade no exercício dos trabalhos através de uma solução de baixo custo para esta organização. Este trabalho buscou apresentar algumas das opções disponíveis no mercado de maneira a entender as principais funcionalidades oferecidas e valores envolvidos na contratação destes serviços. Em posse disso, foram analisadas as principais funcionalidades já existentes com as necessidades específicas da empresa em estudo, de maneira com que a solução desenvolvida trouxesse funcionalidades efetivamente necessárias para que as necessidades da empresa fossem atendidas, trazendo uma resolução eficiente para o principal problema enfrentado, a dificuldade de gestão da matriz e suas filiais de maneira centralizada. Neste trabalho de conclusão foi implementada parte de uma solução ERP web para o setor de comércio varejista utilizando a metodologia *Personal Scrum*, onde foram pontuados os benefícios do uso desta abordagem no processo de desenvolvimento do sistema em um período de 4 *sprints*. Ao final do desenvolvimento do trabalho, concluímos os módulos de Produtos, Funcionários, como também o módulo de Autenticação, onde estes foram desenvolvidos tanto no *Back-End* quanto no *Front-End*. Também foi possível notar o quão importante se mostrou o Personal Scrum, trazendo melhorias significativas no que diz respeito a foco, comprometimento e, principalmente, na organização das atividades desenvolvidas.

Palavras-chave: Enterprise Resource Planning. Personal Scrum. Comércio Varejista.

ABSTRACT

The increasing introduction of technology into the corporate environment has brought about a range of needs that did not exist until a few years ago. The retail sector was one of the first sectors affected. ERP (Enterprise Resource Planning) systems manage many processes previously performed manually. The location of the company targeted by this work is in the northwest of the state of Rio Grande do Sul. The company has faced a recurring problem in recent years due to the lack of integration between the distributions of its business management software, thus making it hard to register goods, as well as other tasks, with the main side effect being the excessive loss of time in performing them in different branches. To this end, this work is motivated to solve this problem to facilitate the task performed through the use of new software capable of understanding the gaps left by the company's legacy software, thus bringing agility to the performance of work through a low-cost solution for this organization. This paper sought to present some of the options available on the market to understand the main functionalities offered and the values involved in contracting these services. With this in mind, the main functionalities already in existence were highlighted with the specific needs of the company under study so that the solution developed would bring useful functionalities to meet the company's needs, providing an efficient solution to the main problem faced, the difficulty of managing the head office and its branches in a centralized manner. This final paper innovatively developed part of a web ERP solution for the retail sector using the methodology offered by *Personal Scrum*, where we highlight the benefits of using this approach in the system development process over 4 *sprints*.

Key-words: Enterprise Resource Planning. Personal Scrum. Retail Business.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na era digital, o comércio varejista experimentou uma transformação profunda, impulsionada pela integração de sistemas avançados. As lojas físicas tradicionais expandiram suas operações para o ambiente online, criando uma presença digital sólida e permitindo transações e interações com os clientes pela internet. A dependência de sistemas informatizados e plataformas de *e-commerce* tornou-se essencial para gerenciar inventários, processar pedidos, rastrear vendas e oferecer uma experiência personalizada ao cliente. A eficiência operacional, as análises de dados precisas e a capacidade de adaptar-se rapidamente às demandas do mercado são cada vez mais dependentes da tecnologia digital, estabelecendo uma interconexão vital entre o comércio varejista e os sistemas digitais (LIE, 2021).

De acordo com Alday e Pinochet (2017), a presença virtual de comércios varejistas traz inúmeras vantagens. Tudo acaba por ser facilitado com a automação eletrônica de vendas, dado que há uma redução no tempo demandado para realizar transações. Além disso, a expansão da imagem da marca da empresa faz com que o alcance da área de clientes atingidos por suas campanhas de marketing seja elevado exponencialmente, gerando assim um novo canal de captação de clientes.

Beraldi e Filho (2000) também levantam pontos positivos para a informatização na gestão de pequenas empresas. Os autores apontam o ganho significativo de eficácia e eficiência, garantindo competitividade e o aumento no faturamento dessa forma. Ademais, citam também a melhora na tomada de decisão e no controle interno das operações, além de permitir a automatização de tarefas rotineiras.

O desenvolvimento de sistemas de gestão no comércio varejista representa um desafio crucial e contemporâneo no ambiente empresarial atual, dada a crescente dependência de tecnologias digitais para operações eficientes e experiências aprimoradas do cliente. O gerenciamento eficaz dos diversos serviços que uma empresa envolve, como gestão de vendas, controle de estoque, processamento de pagamentos, análise de dados e interfaces de cliente, são fundamentais para otimizar operações, fornecer *insights* estratégicos e garantir a satisfação do cliente.

Utilizar sistemas de gestão traz uma quantidade considerável de benefícios para quem a realiza em seu estabelecimento comercial. Pricefy (2024) apresenta algumas vantagens, como o aumento no controle de estoque, onde é notável a facilidade de acesso à informações sobre determinado produto, sua quantidade e suas características. Isso faz com que indiretamente surja outra grande vantagem que é o planejamento eficaz, onde as informações provindas do sistema podem ser levadas em conta nas decisões da empresa, tornando-se assim um fator determinante para ações mais assertivas. Outra vantagem que merece atenção é que, com a obtenção de dados confiáveis sobre o estoque, podem ser estabelecidas estratégias para precificação de produtos, que determinam o custo ideal para as vendas, sem que ocorra a situação em que o produto está muito caro ou muito

barato.

Ao observar a realidade de comércios distantes dos grandes centros, percebe-se a grande dificuldade para integrar sistemas entre as filiais e a matriz por parte dos proprietários de comércio. A principal justificativa para esse problema são os altos custos envolvidos na aquisição de planos que cobrem a integração. Além disso, sistemas de fornecedores diferentes nem sempre possibilitam a interoperabilidade entre si, sendo necessário adquirir planos com valores elevados para atender a demanda de integração entre os sistemas. Outro ponto a ser abordado é o fato de que não existem planos personalizados para as necessidades das empresas, o que significa que estas muitas vezes arcam com custos de funcionalidades que não são cruciais e necessárias para a organização. Ainda também, o desenvolvimento de software de forma individual apresenta diversos desafios de gerenciamento. Para isso é necessário fazer uso de alguma abordagem para conduzir o processo de criação de software.

No entanto, a implementação bem-sucedida e a manutenção de um sistemas de gestão no contexto varejista enfrenta inúmeros obstáculos, incluindo complexidade tecnológica, custos, segurança da informação e resistência organizacional. Este trabalho de conclusão de curso propõe, como estudo de caso, o desenvolvimento de um sistema de gerenciamento empresarial para uma loja do comércio varejista fazendo uso do *Personal Scrum*, analisando seus impactos e possíveis soluções, visando fornecer *insights* valiosos e recomendações práticas para aprimorar a eficiência operacional e a competitividade das empresas varejistas no cenário digital em constante evolução.

Este trabalho tem três grandes contribuições:

- Resolver o problema de integração de software de uma empresa específica;
- Generalizar essa solução a fim de que a mesma possa ser adotada em outras empresas com problemas semelhantes;
- Apresentar como o *Personal Scrum* pode apoiar o desenvolvimento em projetos individuais.

O restante da monografia está dividido em 4 capítulos. O Capítulo 2 apresenta aspectos conceituais relacionados com o desenvolvimento deste trabalho. O Capítulo 3 traz a metodologia adotada para a criação do sistema. No Capítulo 4 são descritas as implementações realizadas. O Capítulo 5 apresenta os resultados obtidos com a implementação de algumas partes do sistema. O Capítulo 6 discute os resultados.

2 SOFTWARES RELACIONADOS

Atualmente, os sistemas Web são amplamente utilizados para a realização de manipulações simples, além de facilitar o acesso e fornecer disponibilidade aos dados armazenados. Os sistemas Web permitem a automatização para os mais diferentes tipos de processos que uma organização possui, fazendo com que sejam versáteis. Isso assegura uma comunicação instantânea e digital que traz melhorias significativas para a manipulação de dados, agilizando, dessa forma, a gestão como um todo (MATUTE; AVILA-PESANTEZ; AVILA, 2020).

A Tabela 1 traz informações obtidas através de uma busca exploratória que visa identificar as soluções disponíveis no mercado em relação a sistemas ERP. A tabela pode ser utilizada como base para uma análise de recursos fornecidos em sistemas para gestão empresarial, evidenciando os componentes de seus planos e os custos envolvidos na contratação.

Os softwares investigados foram alcançados por meio de uma pesquisa, buscando identificar empresas que fornecem um produto semelhante ao idealizado como solução para a empresa em estudo. O método usado para a pesquisa foi uma busca simples na Web, utilizando palavras como *Software ERP* e *Software de gestão empresarial*. A limitação dos produtos relacionados deu-se selecionando provedores do serviço pela sua colocação no mercado e considerando os *feedbacks* recebidos pelas corporações sobre o seu serviço.

2.1 VendaSimples

VendaSimples é um sistema ERP que tem como funcionalidades apenas a gestão de vendas e a emissão de Nota Fiscal Eletrônica (NF-e) como parte do pacote oferecido pela empresa em seu plano de entrada, custando um total de R$ 69,90 mensais[1].

Dentre as funcionalidades examinadas, o software apresenta algumas deficiências funcionais que são cruciais para uma empresa. Entre os pontos a serem destacados está a ausência de gestão de estoque e gestão financeira, que representam juntas uma grande perda de valor para o sistema. O sistema também não disponibiliza relatórios ou *dashboards*, não permite integração com plataformas *e-commerce* e gestão multi empresarial, como também não incorpora operações *offline*.

2.2 Senior

O *software* oferecido pela empresa *Senior* traz uma variedade considerada satisfatória para o seu plano base. Porém não apresenta valores para consulta, o que dificulta a análise do seu custo-benefício.

Dentre as funcionalidades oferecidas, destacam-se a gestão de estoque, gestão financeira, gestão de vendas e a emissão de *NF-e*. Já a falta de disponibilidade de relatórios

[1] (VENDASIMPLES, 2024)

Tabela 1 – Recursos de gestão empresarial oferecidos por empresas do setor varejista

Recursos oferecidos no plano base de cada sistema ERP										
		Empresas								
		VendaSimples	Senior	UpGestão	TOTVS ERP Protheus	Com2 Business	Gestão Click	Linx ERP	Cake ERP	Presente Trabalho
Recursos	Gestão de Estoque	✗	✓	✓	✓	✓	✓	✓	✓	✓
	Gestão Financeira	✗	✓	✓	✓	✗	✓	✓	✓	✓
	Gestão de Vendas	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Emissão de NF-e	✓	✓	✓	✓	✓	✗	✓	✓	✓
	Relatórios	✗	✗	✓	✗	✓	✗	✓	✓	✓
	Integração e-commerce	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Operações Offline	✗	✗	✗	✗	✗	✗	✓	✗	✓
	Multiempresa	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Valor mensal	R$ 69,90	ND	R$ 198	ND	R$ 80	R$ 69,90	ND	R$ 139	ND

Fonte: O Autor

reflete em uma grande perda para a parte interessada em seus serviços. O *software* também carece de integração *e-commerce*, operações *offline* e gerenciamento de mais de uma unidade de uma empresa[2].

2.3 Upgestão

UpGestão[3] oferece como plano mais em conta por R$ 198,00, o que representa um custo mensal elevado. Porém, o valor tem total relação com as funções que o sistema disponibiliza. No plano estão inclusas a gestão de vendas, gestão de finanças, gestão de estoque, emissão de *NF-e* e relatórios. Este é um bom conjunto de funcionalidades.

O sistema deixa a desejar em 3 funcionalidades: integração com o *e-commerce*, operações offline e gerenciamento multi-empresarial. Mesmo assim, levando em consideração as funções disponibilizadas, é um sistema que pode ter um bom custo-benefício.

[2] (EMPRESARIAL, 2024)
[3] (TECH, 2024)

Além disso, a maneira como a *UpGestão* apresenta os dados em relatórios, o quão intuitivo é o seu sistema e como suas funcionalidades atendem as necessidades do contratante, acredito que esses aspectos podem contribuir ainda mais em seu favor quando falamos de custo-benefício sobre o que o software entrega.

2.4 *Protheus* - TOTVS

O sistema ERP da *TOTVS*, conhecido como *Protheus*, é o software que traz mais funcionalidades interessantes para aqueles que necessitam cobrir apenas as necessidades básicas para seu negócio. O custo para adquirir seus serviços não é divulgado no site oficial da empresa, tornando difícil a ponderação sobre benefícios × custos.

Além de não disponibilizar relatórios ao contratante, o software também não oferece integração a *e-commerce*, gerenciamento de mais de uma unidade/filial da empresa e não implementa tolerância a ações realizadas de maneira *offline*[4].

2.5 Com2 Business

Esta empresa possui no mercado um *software* que, assim como as outras, oferece um bom conjunto de funcionalidades, que engloba gestão de estoque, gestão de vendas, emissão de *NF-e* e relatórios de apresentação de métricas e resultados. Todas essas funcionalidades são oferecidas em um plano com custo mensal de R$ 80,00, valor que, pelas funcionalidades oferecidas, torna-se um diferencial para este software.

O principal ponto negativo a ser destacado é a falta de cobertura sobre a gestão financeira, que é uma funcionalidade amplamente usada atualmente, tendo em vista que esse controle é dificultoso até mesmo com o auxílio de planilhas, sendo acentuado quando feito de maneira manuscrita. Além disso, assim como os outros *softwares* mencionados até então, o mesmo não fornece integração com plataforma *e-commerce*, permite somente a gestão de uma unidade dentro de uma organização e não cobre operações *offline*[5].

2.6 GestãoClick

Dentre as funcionalidades analisadas em GestãoClick, este *software* fornece para empresas a gestão financeira, de vendas e de estoque por um total de R$ 69,90 ao mês. Assim como os demais *softwares*, também não dispõe de integração com algum sistema *e-commerce*, não permite operações offline e gestão multi-empresarial.

Nesta ERP destacam-se duas ausências importantes. A falta de relatórios para a apresentação de resultados periódicos pode ser um fator importante no momento da decisão por parte de um gestor empresarial. Entre as empresas que fornecem sistemas para

[4] (TOTVS, 2024)
[5] (COM2BUSINESS, 2024)

a gestão empresarial, esta ferramenta destaca-se negativamente pela carência na emissão da Nota Fiscal Eletrônica (NF-e). Esta lacuna em aberto tende a distanciar empresas e diminuir a aderência por este software, tendo em vista que ao deixar de emitir *NF-e*, segundo o artigo 3º da lei nº 8.846 de 21 de Janeiro de 1994[6], se não comprovada a sua emissão, é aplicada uma multa de 300% sobre o valor dos bens envolvidos na operação[7].

2.7 Linx

O *software* disponibilizado pela empresa *Linx* traz o mais completo sistema dentre os pesquisados. O seu diferencial em relação aos outros *softwares* é o tratamento de operações *offline*, permitindo ao usuário a realização de vendas, cadastros e transações sem que haja a necessidade de uma conexão estável com a *internet*, porém não é disponibilizada a precificação destes serviços.

A ERP fornece funcionalidades de gestão de estoque, de vendas e financeira, além de emitir *NF-e* e disponibilizar relatórios para análise de desempenho. Dentre os recursos faltantes, estão somente a integração *e-commerce* e gerenciamento multi-empresa[8].

2.8 Cake

O sistema ERP desenvolvido pela *Cake* implementa recursos semelhantes ao sistema pertencente a *Com2 Business*. Ele cobre funcionalidades de gestão de vendas, estoque e finanças. O sistema também traz consigo relatórios e emite *NF-e*.

O *software*, porém, não possui a capacidade de gerenciamento de mais de uma distribuição de uma mesma empresa, gerando limitações significativas para as mesmas. Ele também não permite a realização de operações sem conexão com a internet e não possibilita a integração do seu sistema com qualquer plataforma *e-commerce*[9].

2.9 Considerações Finais

A pesquisa exploratória abordada nas seções anteriores traz uma visão abrangente dos *softwares* disponíveis no mercado, assim como também as suas funcionalidades diante de condições de aquisição de baixo custo. Com isso, é possível estabelecer um paralelo com as reais necessidades das empresas do segmento do comércio varejista com as funcionalidades que este trabalho trará no sistema proposto. O levantamento de *softwares* destacou também que existem diversas ofertas no mercado por *software* de gestão empresarial. Porém alguns sistemas não apresentam de maneira clara e transparente os valores envolvidos, não permitindo assim uma análise justa em relação aos demais *softwares*.

[6] (Governo Federal, 1994)
[7] (GESTAOCLICK, 2024)
[8] (LINX, 2024)
[9] (ERP, 2024)

O propósito deste trabalho é buscar preencher estas colunas encontradas. Quer-se focar em manter o equilíbrio entre gastos e recursos funcionais, de maneira a oferecer um leque de opções para as empresas que buscam cobrir suas necessidades com um *software* de qualidade, sem a necessidade de arcar com altos custos, sejam eles com aquisição de recursos computacionais próprios ou de licenças de *software*.

3 METODOLOGIA

Como estudo de caso, considerou-se como problema base a situação de uma determinada empresa de comércio varejista. Atualmente a empresa possui um *software* sem a possibilidade de atualização para realizar todo o controle de vendas e estoque. O software é mantido localmente nos computadores da empresa, havendo uma cópia instalada localmente em cada loja. O sistema armazena os dados em um banco de dados relacional, permitindo assim a realização de consultas sobre produtos, clientes cadastrados e outros dados relacionados a vendas e movimentações financeiras.

O sistema oferece uma ampla variedade de funcionalidades atrativas. No entanto, sua interface pouco intuitiva e menus confusos podem causar desconforto durante o uso. Os *backups* devem ser realizados manualmente, havendo a necessidade de indicar um caminho para alguma pasta no computador que armazenará todos os *backups*. Ainda nesta opção, juntamente com a funcionalidade de gerar *backup*, existe a possibilidade de recuperar o estado da aplicação, fornecendo como entrada um *backup* anteriormente gerado. Além disso, a autenticação inicialmente é configurada com apenas um perfil (supervisor) e uma senha padrão. Devido a desafios na adição de perfis para vendedores, isso resulta em conceder acesso irrestrito a todos os dados e funcionalidades do sistema, incluindo informações que, em sua maioria, deveriam ser restritas à visualização exclusiva do proprietário ou do departamento de finanças.

Para resolver os desafios da empresa, este trabalho propõem o desenvolvimento de uma aplicação web destinada ao uso geral da empresa, que envolve tanto a matriz quanto suas filiais. Essa aplicação tem como objetivo substituir o *software* legado atualmente em uso. As próximas seções apresentam a delimitação das tecnologias utilizadas.

3.1 Integrated Development Environment (IDE)

Como ferramenta para auxiliar o desenvolvimento foi utilizado o *Visual Studio Code*, também conhecido como *VS Code*. A escolha por essa *IDE* é baseada na sua grande capacidade de integração com programas que garantem agilidade no processo de desenvolvimento de *software*. A acoplação desta *IDE* com sistemas de versionamento de código traz uma enorme facilidade para trabalhar com sistemas como o *Git*, para a criação de versões locais, e uma maneira simplificada para destinar a versão atual para o repositório *GitHub*, que dispensa a utilização de um terminal com linhas de comandos para realizar o versionamento.

O *IntelliSense*, presente na *IDE*, fornece preenchimento automático de código, seja na declaração de variáveis tipadas ou na criação de funções e módulos. Além disso, o recurso disponibiliza extensões que trazem funcionalidades que auxiliam no desenvolvimento de acordo com a linguagem/*framework*/biblioteca utilizada. Um ponto interessante é que as extensões são executadas em processos diferentes da *IDE*. Isso garante que o editor não fique lento. Outro ponto importante a destacar é a integração com ferramentas de

depuração de código, que permitem uma visualização do estado da aplicação em tempo de execução, sendo possível verificar variáveis em momentos específicos durante a execução do código através de pontos de interrupção (MICROSOFT, 2024b).

3.2 Controle de Versão

Para realizar o versionamento do código produzido durante o processo de desenvolvimento foram utilizados o *Git* e o *GitHub*. O *Git* é uma ferramenta gratuita que garante a criação de várias versões do software em produção, permitindo comparar versões, consultar e regressar para alguma versão anterior, entre outros benefícios. Além disso, o sistema de ramificações com as chamadas *branchs* garante que se possa desviar o fluxo de trabalho da *main*(ramificação principal), criando novas *branchs*, com o intuito de modificar-se uma parte específica do código, mantendo na *main* somente o código em sua versão estável (GIT, 2024).

GitHub é o local onde o código do *software* foi disponibilizado, permitindo que seja acessível de qualquer lugar. Este sistema incorpora os recursos de versionamento provindo de *Git*, para que seja proporcionado e utilizado de maneira colaborativa (KINSTA, 2024).

É possível encontrar os repositórios do trabalho em:

- Front-End - *https://github.com/JuniorThume/entERPrise-Front-TCC.git*
- Back-End - *https://github.com/JuniorThume/entERPrise-Back-TCC.git*

3.3 Scrum

As metodologias ágeis são abordagens para o gerenciamento e desenvolvimento de projetos que priorizam a adaptabilidade, a colaboração e a entrega contínua de valor. Elas surgiram como uma alternativa aos métodos tradicionais, como o modelo em cascata, que frequentemente sofrem com a rigidez diante de mudanças de requisitos.

A escolha de *Scrum* como método ágil está relacionada com a eficiência e flexibilidade disposta no *framework*. Sua estrutura ágil facilita a adaptação a mudanças nos requisitos do projeto, permitindo entregas incrementais e rápidas. O ciclo regular de *sprints*, com revisões e retrospectivas, promove uma abordagem iterativa e a melhoria contínua do processo. A gestão eficaz do *backlog* do produto e da *sprint*, juntamente com a flexibilidade para ajustar prioridades, contribuem para a entrega de valor ao cliente de maneira rápida e alinhada com as necessidades em constante evolução do projeto. Esses elementos combinados tornam *Scrum* um ótimo aliado no processo de desenvolvimento do projeto (Scrum Guide, 2020).

O autor Costa (2016) traz uma nova abordagem sobre o desenvolvimento usando Scrum, o *Personal Scrum*, onde apenas uma pessoa é responsável pelo processo de evolução do projeto. O *Personal Scrum* é uma adaptação do *framework* Scrum tradicional,

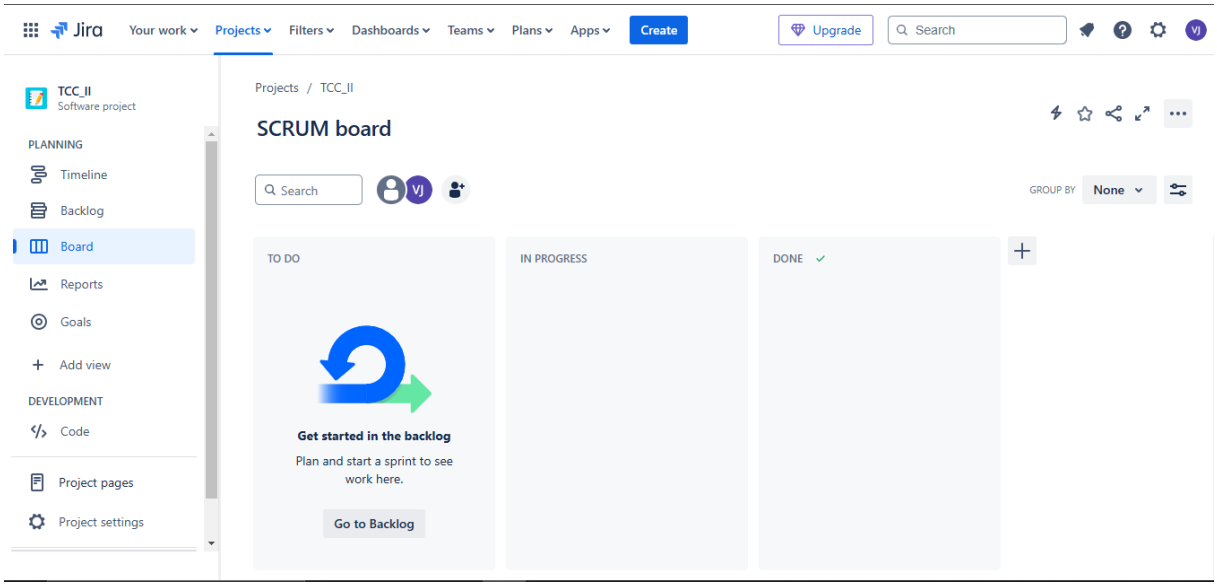

Figura 1 – Jira

voltado para o desenvolvimento de projetos individuais, como no caso de desenvolvedores independentes de jogos. Nesse contexto, o desenvolvedor assume todos os papéis do *Scrum*, incluindo *Product Owner*, *Scrum Master* e equipe de desenvolvimento. Ele é responsável por planejar, executar e avaliar todas as etapas do processo de desenvolvimento, garantindo a aplicação dos princípios ágeis de forma personalizada e eficiente.

Para o controle do quadro *Kanban*, *product backlog* e *sprint backlog* foi utilizado *Jira*, facilitando a definição e priorização de tarefas e duração de *sprints*, oferecendo uma visão mais clara do andamento do projeto e dos prazos relacionados.

A Figura 1 apresenta a página no site que exibe as tarefas da *sprint* atualmente em vigor, apresentando as colunas *To Do* - Fazer, *In Progress* - Em Andamento e *Done* - Pronto. Cada tarefa criada no Jira necessariamente terá que passar por cada uma das 3 colunas, de modo que assim se mantêm mapeado o que será feito, o que está em andamento e o que já foi concluído dentro de uma *sprint*. Com a ajuda do software Jira, foi definida uma duração de *sprints* de 2 semanas, buscando equilibrar a necessidade de agilidade e a redução de riscos envolvidos no desenvolvimento.

3.4 Model-View-Controller (MVC)

A arquitetura escolhida para o projeto de software foi o padrão MVC. Esta escolha deve-se pela abordagem que esta arquitetura apresenta, oferecendo uma clara separação de responsabilidades, promovendo a manutenção e a escalabilidade do código. O *Model*(Modelo) concentra-se nas operações de dados e regras de negócios, garantindo uma gestão eficiente dos dados da aplicação. A *View* (Visão), por sua vez, proporciona uma interface de usuário intuitiva e responsiva, desacoplada da lógica subjacente. O *Controller* (Controlador) atua como um mediador eficaz, gerenciando as interações do usuário,

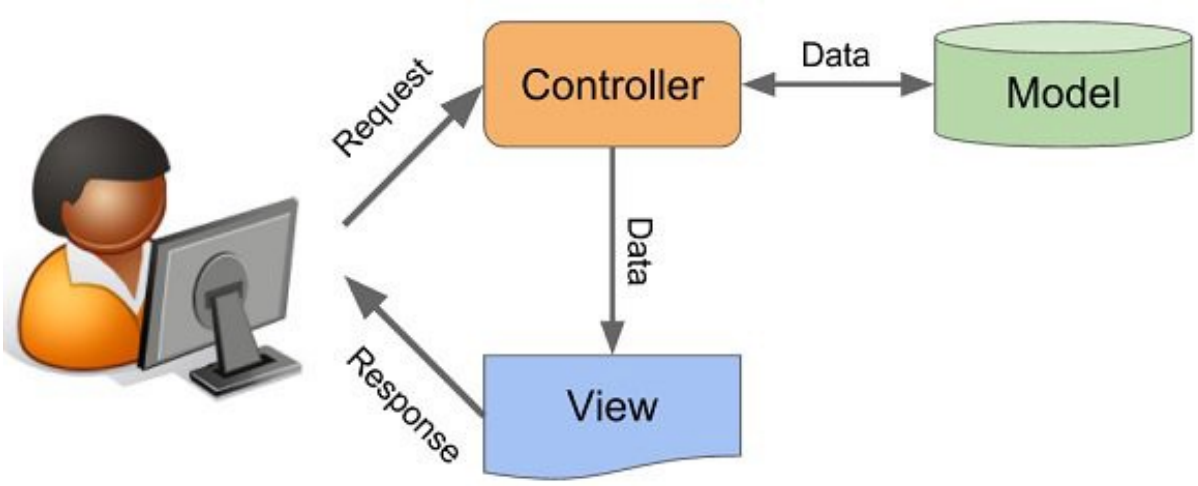

Figura 2 – Padrão Arquitetônico MVC

Fonte: (SILVA, 2024)

processando as entradas e atualizando tanto o *Model* quanto a *View*, conforme necessário.

Essa arquitetura modular facilita a adição de novos recursos ou alterações nos requisitos, pois as alterações em uma camada não afetam diretamente as outras. O padrão MVC não apenas aumenta a legibilidade e compreensão do código, mas também oferece uma base sólida para futuras expansões e melhorias na aplicação (MICROSOFT, 2024a). A Figura 2 apresenta de maneira gráfica as colocações feitas acima acerca do padrão MVC.

Quanto às responsabilidades, *Model* é responsável por gerenciar os dados, *View* é responsável por exibir os dados e o *Controller* é responsável por lidar com a entrada do usuário e atualizar *Model* e *View* quando necessário. Os três componentes interagem entre si enviando e recebendo dados e mensagens. *Controller* responde a eventos, recebe a entrada do usuário e envia comandos para *Model* e *View* para atualizar seus estados. *Model* atualiza seu estado com base nos comandos recebidos de *Controller* e notifica *View* de quaisquer alterações. *View*, então, atualiza sua exibição com base no estado atualizado do modelo.

3.5 React

Para o desenvolvimento das telas do sistemas foi utilizado *React*. *React* é uma biblioteca de *JavaScript*. A decisão de utilizá-la no sistema está diretamente ligada a necessidade envolvida na criação da interface que o usuário interage. Um ponto que realça a escolha pela *React* é sua orientação a componentes, que torna o desenvolvimento organizado, modular e reutilizável, características estas que facilitam a manutenção (REACT, 2024a).

Outro fator determinante para o uso de *React* é a implementação do *Virtual DOM*,

que permite que apenas partes específicas da interface sejam alteradas e renderizadas novamente, sem que haja a necessidade de realizar uma atualização total da página (REACT, 2024b). Ainda, *React* possui compatibilidade com diversas bibliotecas e ferramentas, além de ter integração com *Redux*, que é responsável pelo gerenciamento de estados nos componentes. *React Router* auxilia no direcionamento para as páginas que compõem a interface do sistema. Dessa forma, ele permite uma maior separação de responsabilidades dentro da aplicação.

Por fim, com *Jest* é possível testar o *software* de maneira que se possa verificar a corretividade dos componentes e funções em *React* (JEST, 2024). A comunidade de usuários de *React* é extremamente ativa, trazendo frequentemente discussões que enriquecem e aprimoram o ecossistema do *framework*, além de fornecer todo o suporte necessário em cenários problemáticos durante o desenvolvimento.

Também foi utilizado *TypeScript*, que é um superconjunto de *JavaScript*. Ele adiciona recursos que não são nativos de *JavaScript*, como por exemplo tipagem estática (que é definida explicitamente na codificação) (MELO, 2020).

3.6 Node

A plataforma *Node.js* tem por característica permitir com que códigos escritos em *JavaScript* sejam executados fora do ambiente de navegadores (Google Chrome, Mozilla Firefox, Microsoft Edge, entre outros) (BESSA, 2024). Uma das razões pela escolha de *Node* é sua capacidade de lidar muito bem com operações de entrada e saída não bloqueantes, pois trabalha com operações de maneira assíncrona, fazendo com que o servidor possa processar outras requisições enquanto as demais operações não foram finalizadas. Além disso, pela sua vasta quantidade de módulos e bibliotecas, a plataforma facilita a interação com os mais diversos banco de dados (MySQL, MongoDB e PostgreSQL) e com *APIs RESTful* (Awari Atividades de Ensino LTDA, 2024).

4 DESENVOLVIMENTO

A etapa de implementação representa um passo importante na criação de software. Porém, isso não é tudo no processo de construção de um novo software. É importante destacar as fases anteriores a esta, fases estas que juntas compõem o Ciclo de Vida de Desenvolvimento de *Sofware* (CVDS), conhecido no inglês como *Software Development Life Cycle* (SDLC).

De acordo com (DENNIS; WIXOM; ROTH, 2014), o Ciclo de Vida de Desenvolvimento de Software pode ser dividido em 4 etapas: Planejamento, Análise, Projeto e, por fim, a etapa de implementação. É uma estrutura que define as etapas necessárias para criar, desenvolver e manter um software, colaborando também com equipes no que diz respeito a condução de planejar, organizar e executar projetos.

Com isso em mente, este projeto buscou ao máximo incluir processos anteriormente mencionados no processo de criação do software, utilizando os métodos do CVDS para produzir algo de valor, tanto para a empresa alvo, oferecendo um produto confiável e eficiente, quanto para atender quesitos de qualidade de software. O modelo de execução do Ciclo de Vida é encaixado como modelo Ágil. Isso se deve a utilização do *Personal Scrum* na construção do software. Significa também que se trata de um modelo de desenvolvimento cíclico, trazendo uma maior ênfase em ser um processo iterativo e incremental.

4.1 Planejamento

A etapa de Planejamento, descrita no Ciclo de Vida de Desenvolvimento de Software, representa a fase inicial do projeto, sendo o ponto de partida de uma ideação anteriormente realizada. Esta etapa é marcada pela definição do porquê o software deve ser construído. Devido ao formato empregado neste trabalho, não foi necessário realizar qualquer análise de viabilidade que visasse a ponderação sobre a continuação ou não do projeto, pois é entendido que a necessidade da empresa alvo deve sim ser levada em conta, porém a proposta de analisar o desenvolvimento através do uso do *Personal Scrum* prevalece sobre qualquer condição.

No contexto do trabalho, esta fase do *CVDS* serviu para esclarecer questionamentos acerca do problema a ser tratado e para entender de que forma *Personal Scrum* se acoplaria no processo de criação de software. Também foi um período de exploração do software legado da empresa, visando compreender como ocorrem os processos em fases de cadastramento de produtos e clientes, lançamentos de vendas, consultas de inadimplentes, entre outros pontos.

4.2 Análise

Essa fase consiste em entender alguns aspectos mais profundos do software a ser desenvolvido. É nesta etapa que o sistema recebe a atribuição das características sobre *quem* irá operá-lo, funcionalidades e especificações de operação de *como* e *onde* irá operar. Através da observação do software legado utilizado pela empresa, foi possível elencar requisitos quanto a usabilidade do sistema, como também de algumas condições de funcionamento para o sistema. Passada essa etapa, foi possível concluí-la com o desenvolvimento dos requisitos do sistema, como também diagramas de casos de uso.

Tendo em mente as partes que iriam compor o sistema, foram elicitados os requisitos funcionais e não funcionais de forma que isso fosse extremamente úteis no momento da criação do *product backlog*. Após a observação do atual sistema da empresa, foi possível elaborar um documento que carrega os requisitos funcionais e não funcionais, apresentados na Figura 3. A etapa de levantamento de requisitos do software visa dar mais clareza das funcionalidades que o software precisa abranger.

Dos requisitos mencionados acima, foi possível concluir a parte referente ao gerenciamento de produtos, no que diz respeito aos requisitos funcionais. Já da parte dos não funcionais, foram abrangidos os quesitos operacionais do sistema, como também a parte que recai sobre definições de segurança, trazendo o controle de acesso de acordo com o privilégio e também a parte responsável pela autenticação no sistema.

4.3 Projeto

A fase de Projeto, ou também chamada de fase de Design, foca na documentação do sistema, envolvendo quesitos de hardware, software, como também fatores envoltos na estrutura de rede necessária (DENNIS; WIXOM; ROTH, 2014). Existe uma grande contribuição desta etapa no processo de documentação do software, principalmente devido as representações sobre como o sistema estará organizado e também de como se apresentará para o usuário do sistema.

4.3.1 Modelagem do Banco de Dados

Com o auxílio da ferramenta *BR Modelo*, foi definido o modelo conceitual e lógico do banco de dados, onde a modelagem visa dar clareza sobre as tabelas e a forma com que se relacionam. Como apresentado na Figura 4, o modelo conceitual é a primeira representação abstrata do banco de dados. Ele visa capturar os requisitos de dados de maneira ampla e independente de qualquer tecnologia específica de banco de dados ou restrições de implementação.

Já o modelo lógico é apresentado na Figura 5. O modelo lógico é uma representação mais detalhada e técnica dos dados, baseado no modelo conceitual, mas com maior

Requisitos Funcionais

1. *Gerenciamento de Produtos, Clientes e Vendas*
   - 1.1. O sistema permitirá o CRUD de produtos, estoque, clientes, fornecedores e filiais.
   - 1.2. O sistema permitirá o CRUD de vendas, contas e compras.

2. *Relatórios e Alertas*
   - 2.1. O sistema fornecerá relatórios gerais de cadastros.
   - 2.2. O sistema alertará sobre produtos com baixo estoque.
   - 2.3. O sistema alertará sobre contas a vencer por parte dos clientes.
   - 2.4. O sistema permitirá a visualização do histórico de compras dos clientes.
   - 2.5. O sistema permitirá que os funcionários registrem recebimentos, vinculando-os a clientes e vendas.

Requisitos Não Funcionais

1. *Operacional*
   - 1.1. O sistema deve ser acessível por meio de navegadores modernos, como Chrome, Firefox e Edge.
   - 1.2. O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela, incluindo dispositivos móveis e desktops.

2. *Desempenho*
   - 2.1. O sistema deve garantir que as informações estejam sempre atualizadas, refletindo as ações realizadas pelos usuários em tempo real.
   - 2.2. O sistema deve garantir tempos de resposta que não excedam 5 segundos para consultas mais complexas, como geração de relatórios detalhados.

3. *Segurança*
   - 3.1. O sistema deve ser seguro, utilizando autenticação para garantir que apenas usuários autorizados tenham acesso às funcionalidades.
   - 3.2. Apenas a proprietária e usuários com privilégios apropriados poderão acessar funcionalidades específicas.
   - 3.3. O sistema realizará backups automáticos dos dados periodicamente.
   - 3.4. O sistema oferecerá diferentes níveis de acesso de acordo com o privilégio(administrador, gerente e vendedores).

Figura 3 – Requisitos funcionais e não funcionais do sistema

profundidade para refletir como os dados serão estruturados, incluindo chaves estrangeiras onde existem relações diretas entre tabelas.

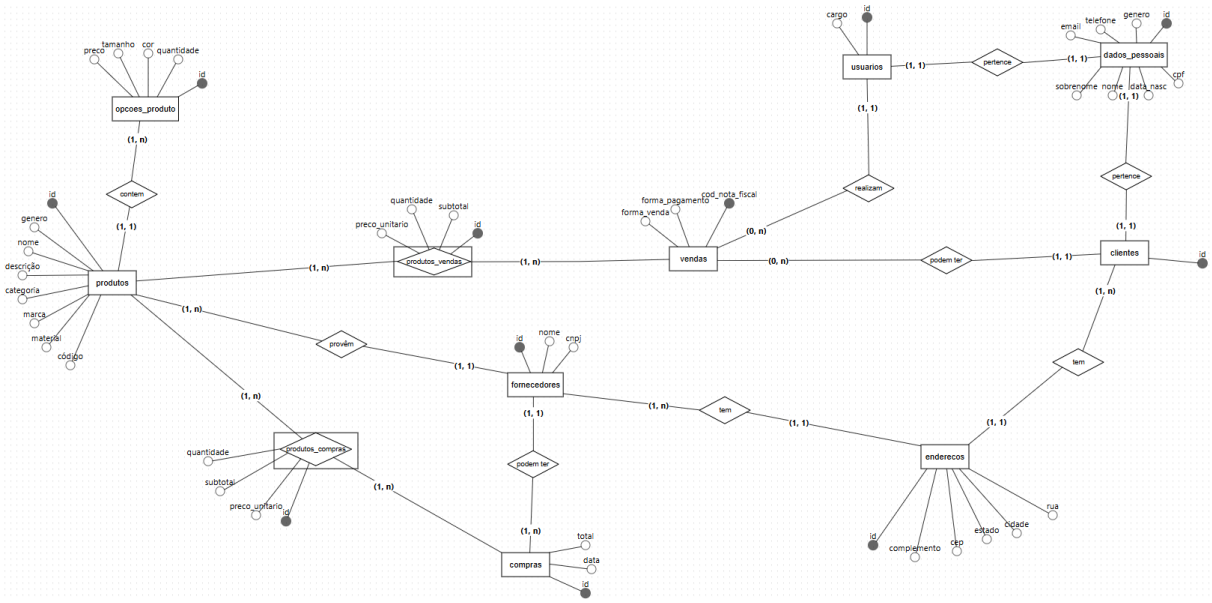

Figura 4 – Modelo Conceitual

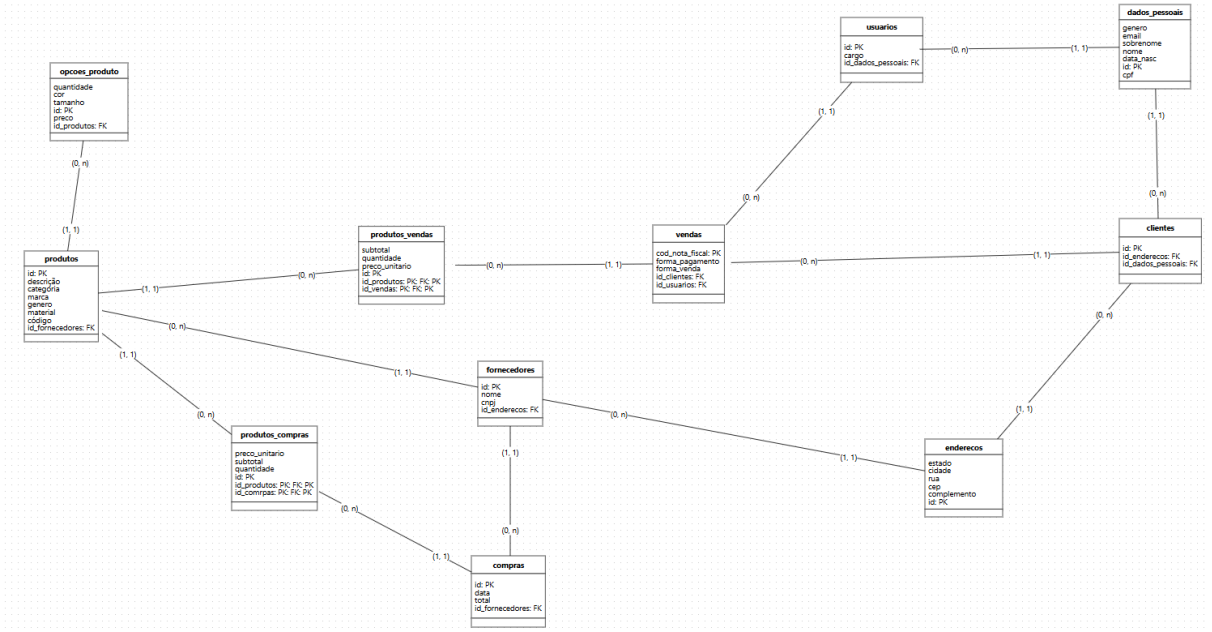

Figura 5 – Modelo Lógico

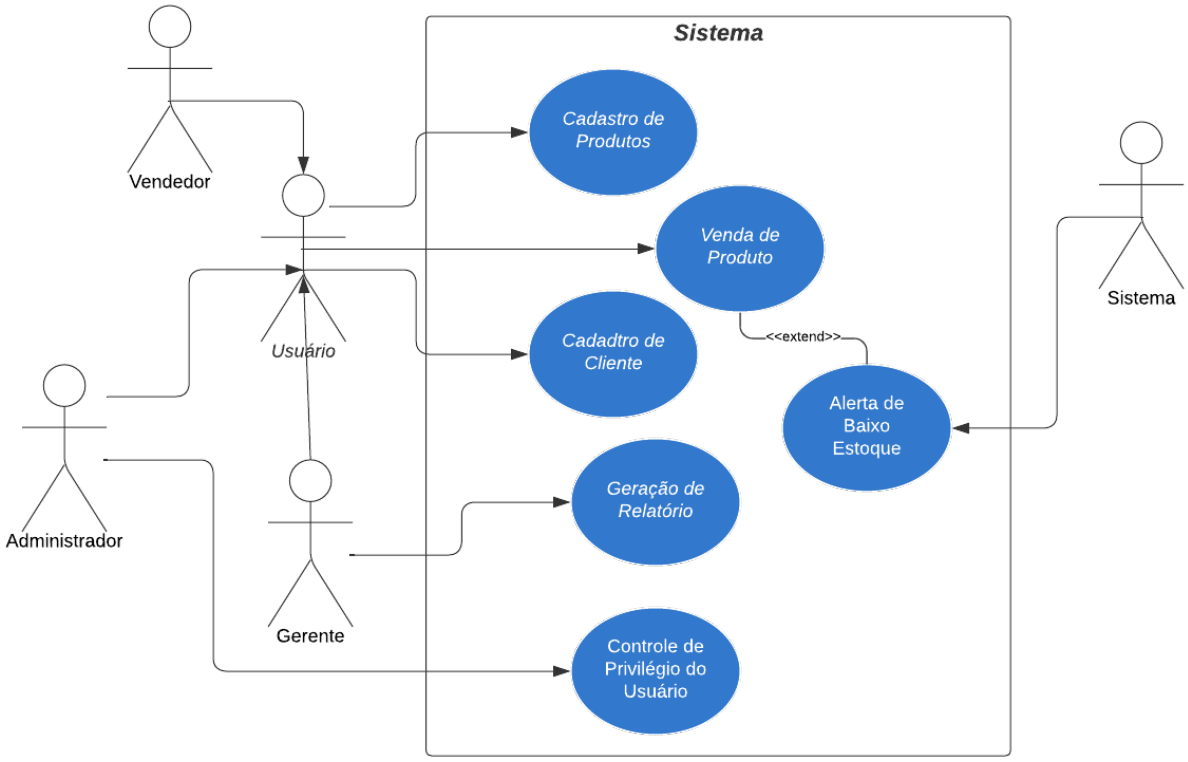

Figura 6 – Diagrama de Casos de Uso

4.3.2 Casos de uso

A criação dos casos de uso do software foi realizada com base nos requisitos levantados durante a etapa inicial da análise. Esse processo teve como objetivo principal traduzir os requisitos em funcionalidades claras e detalhadas, definindo como o sistema deve se comportar diante de determinadas situações de operação.

A Figura 6 apresenta os casos de usos desenvolvidos neste trabalho, trazendo as relações de atores com seus respectivos casos de uso. No presente trabalho, houve o desenvolvimento do caso de uso *Cadastro de Produtos*, deixando pendente de realização os demais.

Cada caso de uso foi estruturado a partir de elementos fundamentais. Primeiramente foram identificados os atores que interagiriam com o sistema, como usuários, que são representados pelo administrador do sistema, gerentes ou vendedores. Em seguida, foi descrito o fluxo principal de cada funcionalidade, representando o caminho padrão para atingir os objetivos propostos. Também foram mapeados fluxos alternativos, contemplando variações ou situações de erro, além de pré e pós-condições que definiam os estados inicial e final de cada interação. Os casos de uso estão disponíveis na seção de anexos, na Figura 16 e Figura 17, contidas no item C.

4.3.3 Prototipação de telas

Para a prototipação das telas, utilizou-se Figma[1]. Os protótipos das telas podem ser encontrados na Seção A dos Anexos, onde são descritas e apresentadas as funcionalidades contidas em cada página.

A Figura 7 representa a tela que o usuário acessa inicialmente ao clicar na seção de Produtos, nela temos as descrições sobre os componentes integrantes da tela, onde há produtos listados que permitem ações como remover, atualizar e visualizar o produto com mais detalhes. Além disso, há o campo de busca que permite encontrar produtos com palavras-chave contidas em seu nome, e, ao lado, os filtros que podem ser aplicados para encontrar produtos com determinadas características(como categoria, material do produto, gênero e marca). Ainda há o botão *Adicionar* que direciona o usuário para a tela de cadastro de produto(essa tela pode ser encontrada no Anexo A), como também, na parte inferior da imagem, o menu de paginação que estará presente sempre que o número de produtos for maior que 5 produtos na consulta ao banco de dados.

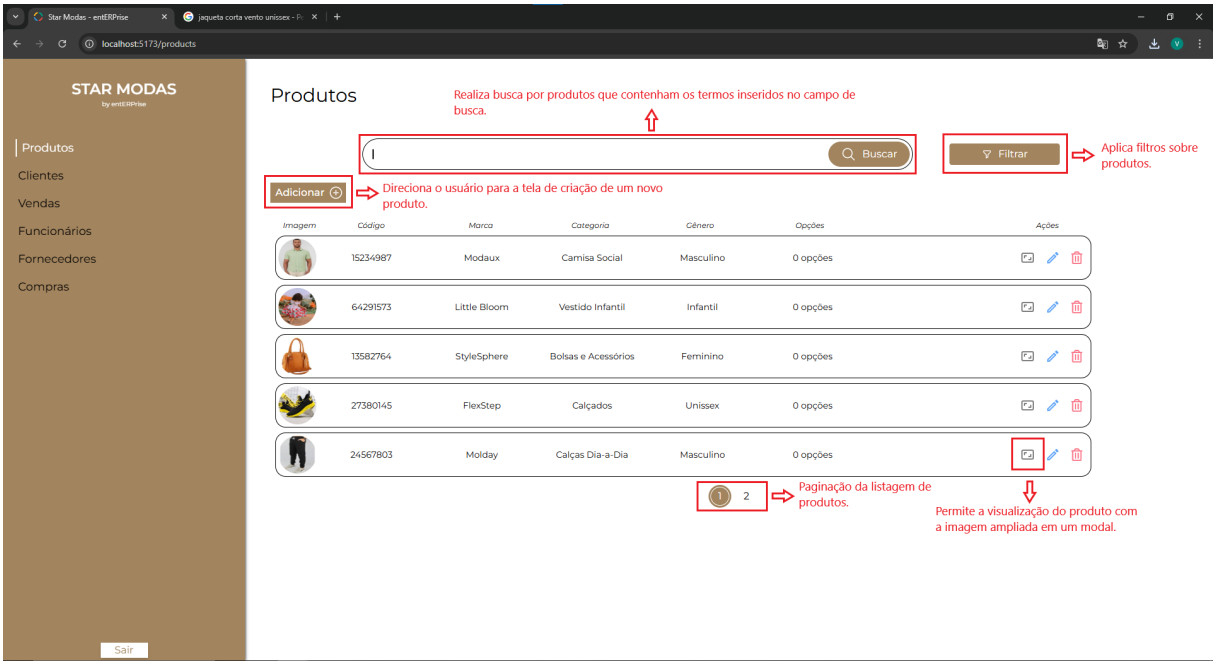

Figura 7 – Tela de listagem de produtos

A utilização de uma ferramenta para apoiar o processo de prototipação foi de suma importância. Mesmo que não existisse uma ideia muito clara de como seriam as telas, quaisquer alterações demandavam menos trabalho do que realizar as alterações diretamente no código, especialmente nos casos onde a ideia era apenas vislumbrar como determinada alteração ornaria com os demais componentes de uma página.

[1] <https://www.figma.com/>

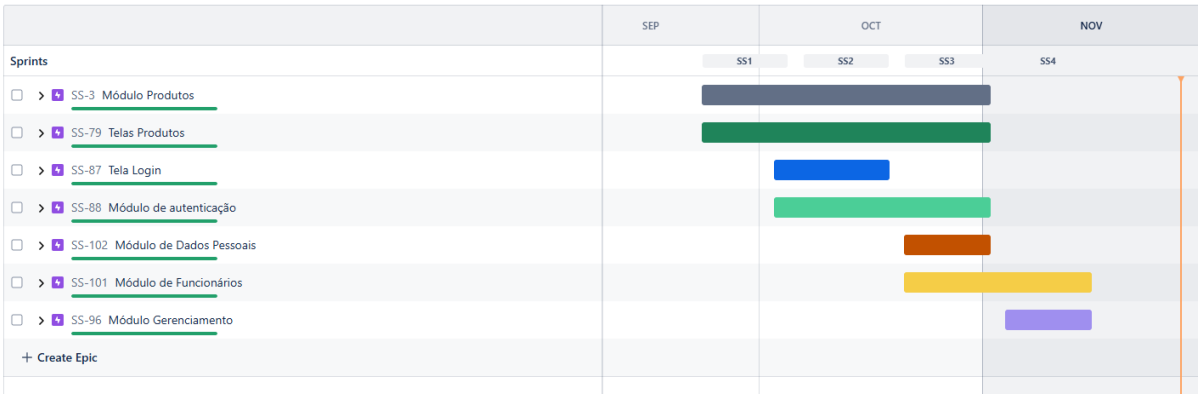

Figura 8 – Painel da linha do tempo do Jira

4.3.4 Jira

A configuração do Jira[2] foi realizada totalmemnte no ambiente web, iniciando pela nomeação do projeto, definindo o *product backlog*, duração de *sprints*, entre outras configurações. O seu uso permitiu com que houvesse um maior controle sobre as atividades a serem realizadas, possibilitando o rastreamento dos itens já concluídos, entre outras vantagens.

A utilização da ferramenta apresentou algumas dificuldades, principalmente no período de configuração e nas primeiras utilizações. Porém o uso quase que diário permitiu que duvidas fossem sanadas no decorrer das *sprints*, mostrando seu verdadeiro poder de gestão, mesmo em sua versão gratuita.

A Figura 8 apresenta como é o painel da linha do tempo no Jira, permitindo a visualização do andamento do projeto ao longo do tempo, facilitando a visualização do trabalho realizado dentro do *timebox* das *sprints*.

4.3.5 Sprints

O *timebox* de uma *sprint* foi estabelecido em 2 semanas, tendo em vista que fornece menor risco sobre os entregáveis em comparação com *sprints* com durações maiores. Além disso, é um ciclo curto o suficiente para que se mantenha focado nas metas, evitando mudanças de escopo que poderiam ocorrer em durações maiores.

Ao inicio de cada *sprint*, foram realizadas "reuniões de uma pessoa", onde a ideia era definir as tarefas e as metas da *sprint* para as duas semanas seguintes. Nesta etapa, foram elencadas as tarefas com maior prioridade. Porém existiram situações em que tarefas de menor prioridade impedirem a execução de tarefas com prioridade maiores, como em casos de tarefas de configuração em relação a tarefas de criação de serviços.

Já ao final de cada uma das *sprints*, foram realizadas as *sprint review* com a finalidade de validar os entregáveis e entender em que pontos poderia ser melhorada a

[2] <https://www.atlassian.com/br/software/jira>

Figura 9 – Product Backlog no Jira

condução da próxima *sprint*, passando por uma espécie de autocrítica.

4.3.6 Product Backlog

Antes de iniciar a primeira *sprint*, foi necessário criar o *product backlog* do produto. O mesmo foi feito baseado no documento de requisitos funcionais e não funcionais levantados anteriormente, a Figura **9** apresenta o estágio inicial do *product backlog*, que naturalmente sofreu alterações durante o processo de desenvolvimento.

Com isso, em um primeiro momento, foram criadas tarefas amplas, genéricas o suficiente para estas pudessem ser quebrada em sub-tarefas.Ao longo do desenvolvimento foram sendo adicionadas novas tarefas com as mais diferentes prioridades, por vezes alterando significativamente o número de entregáveis ao final da *sprint*.

4.4 Implementação

É neste ponto onde todos os artefatos viram código, que posteriormente viram funcionalidades para suprir a necessidade de quem irá utilizar o *software*, ou seja, é a etapa em que reunimos toda a engenharia anteriormente feita, baseando-se nela para conduzir o desenvolvimento do projeto.

Esta etapa representou um desafio muito grande, principalmente pelo ponto de vista da organização entre o desenvolvimento *Back-End* e *Front-End* de forma concomitante.

Além disso, foi a etapa que naturalmente demandou mais tempo para ser desenvolvida, e consequentemente, a que mais trouxe resultados, tanto para perceber os efeitos da aplicação do *Personal Scrum* neste contexto, como também no quesito de produção de software propriamente dito.

4.4.1 *Back-End*

Inicialmente, o desenvolvimento iniciou-se pelo *back-end*, parte responsável por receber as requisições e trabalhar em conjunto com o banco de dados para fornecer a gerência necessária das informações. A construção do projeto no *back-end* seguiu determinados padrões que a caracterizam como uma *API RESTful*.

4.4.1.1 API RESTful

Para que uma API receba essa nomenclatura, é necessário que a mesma siga um conjunto de 6 principais premissas (FIELDING, 2000). Para construir uma API RESTful, as seguintes definições devem ser seguidas:

- Cliente-Servidor - Define o estilo de comunicação entre o cliente (solicitante) e o servidor (fornecedor de informações), separando as responsabilidades e permitindo maior escalabilidade do sistema.
- Sem estado - A API não deve armazenar informações de estado sobre requisições. Cada requisição deve conter todas as informações necessárias para que o servidor a processe, o que simplifica a escalabilidade e a recuperação de falhas.
- Cache - Respostas de uma API RESTful devem ser definidas como cacheáveis ou não-cacheáveis, para melhorar a eficiência e reduzir a latência. Esse princípio permite que respostas sejam armazenadas em cache, evitando solicitações redundantes ao servidor.
- Sistema em Camadas - A arquitetura deve permitir a organização do sistema em camadas, o que significa que o cliente não sabe necessariamente se está comunicando-se diretamente com o servidor final ou com um intermediário. Isso contribui para a flexibilidade e segurança da arquitetura.
- Código sob Demanda (opcional) - Esse é o único princípio opcional. Refere-se à capacidade da API de enviar código executável, como scripts, para o cliente, permitindo que funcionalidades adicionais sejam carregadas conforme necessário.

4.4.1.2 Codificação

Como ponto de partida, foi iniciado um projeto em *Node* com a agregação do *framework Express* para fornecer um servidor web baseado na comunicação através do protocolo HTTP (**H**yper**t**ext **T**ransport **P**rotocol). Este protocolo disponibiliza métodos que dizem o que será feito com os recursos indicados na requisição. O sistema fez uso dos verbos POST, PUT, GET e DELETE.

As requisições do tipo POST são interpretadas como ações de criação de novos recursos. Já requisições do tipo GET referem-se a recuperação de informações pelo lado do

requerente. Requisições do tipo PUT possuem o viés de atualizar um recurso e requisições DELETE, de forma autoexplicativa, removem os recursos indicados na requisição.

Ademais, foram utilizadas bibliotecas com o intuito de estipular o que o consumidor destes dados pode ou não fazer. Dentre as principais bibliotecas utilizadas, estão:

- *Celebrate* - Usada para validar os campos no corpo de uma requisição, além de validar parâmetros de rotas, termos em buscas de cada rota exposta.
- *CORS* - Necessária para permitir que requisições possam ser feitas de origens diferentes da URL do site no navegador.
- *JsonWebToken* - Necessária para gerar os *tokens* de autenticação.

4.4.1.3 TypeORM

Com a ajuda do *TypeORM*, foram definidas as entidades, as mesmas identificadas no modelo conceitual (Figura 4), de forma com que o TypeORM dispõe de uma interface para a manipulação dos dados das tabelas. Em outras palavras, o *TypeORM* faz o mapeamento das entidades definidas em código e fornece uma série de funcionalidades para manipular os dados referentes a cada entidade modelada. Com isso, foi reduzida a complexidade de consultas e o tempo de desenvolvimento para funcionalidades que envolviam consultas foi drasticamente reduzido. A utilização dessa biblioteca tornou a configuração do PostgreSQL mais simples, sendo necessário apenas a correta definição de modelos e seus tipos nas migrações realizadas pelo *TypeORM*.

4.4.1.4 Autenticação

Para ter acesso as funcionalidades do sistema, foi utilizado o JWT (*JSON Web Token*) como padrão para realizar a autenticação no sistema. Dessa forma não é necessário estabelecer e gerir sessões de acesso no lado do servidor, reduzindo a complexidade e o processamento que esse gerenciamento acarretaria.

No contexto deste sistema foi definido que a validade de um *token* de autenticação não excede o prazo de 30 minutos e, após isso, ele torna-se inválido. Como consequência, o usuário automaticamente perde o direito de interagir com as rotas protegidas pela autenticação, sendo necessária a geração de um novo *token*. Foi definido na lógica de autenticação que todo *token* gerado no *back-end* é armazenado nos *cookies* do site no navegador para garantir a persistência segura do *token*, que é necessário estar contido nas requisições para produzir efeitos sobre o sistema.

Existem duas formas de receber um *token* válido do sistema. A primeira delas é realizando o login no sistema, isso fará com que o *token* seja armazenado e permita realizar requisições enquanto estiver válido. A segunda é quando o *token* anteriormente gerado passa a ser inválido e torna-se necessária a geração de um novo *token*. Para isso, o

sistema verifica a validade do *token* em cada requisição, caso seja válido, o sistema retorna o sucesso ou a falha da operação. Caso seja inválido, o *website* realiza uma requisição para a rota que revalida o usuário. Nesta rota é verificado se o *token* foi gerado pelo sistema. Em caso positivo para essa avaliação, ele retorna um novo token que será guardado no *front-end* para ser usado nas requisições posteriores até uma nova expiração.

4.4.2 Documentação

A documentação de uma *API* é uma parte fundamental no processo de desenvolvimento, principalmente quando existe a divisão laboral entre desenvolvedores de sistemas provedores de dados no lado do servidor(back-end), e desenvolvedores de sistemas consumidores de dados, no lado do cliente(front-end). Para isso, é importante fornecer uma fonte de consulta para desenvolvedores que desejam consumir e manipular dados através de *APIs*.

Para realizar a documentação a *API* criada neste trabalho foi utilizado *Swagger*[3], com a *OpenAPI* na versão 3.0.0. *Swagger* é uma ferramenta amplamente utilizada para documentar, desenvolver e testar *APIs RESTful*, tornando-as mais acessíveis tanto para desenvolvedores quanto para usuários finais. Ele fornece uma interface gráfica que facilita a visualização e a interação com os *endpoints* da *API*, permitindo executar requisições diretamente na documentação para testar as funcionalidades.

Tendo em vista as diversas rotas disponibilizadas pela aplicação, tornou-se necessário criar a documentação das mesmas. Para isso foi utilizado o *Swagger*, na versão 3.0.0 da *OpenAPI*. OpenAPI fornece uma forma padronizada de descrever os serviços de uma *API*, permitindo que desenvolvedores, ferramentas e sistemas compreendam a *API* sem a necessidade de acessar seu código-fonte. Na seção B dos anexos, se encontra a documentação desenvolvida neste trabalho, junto com o link para consulta da mesma.

4.4.3 Front-End

Como segunda etapa na codificação, foi desenvolvido o *front-end* da aplicação, que é o componente que o usuário tem contato direto e que "conversa" com o *back-end*. Para isso foi iniciado um projeto *React* para desenvolver as telas de busca, criação, edição e remoção de produtos, além de criar tela para gerenciamento de funcionários para usuários com função de administrador ou gerente. Ainda também, foram desenvolvidas as telas de autenticação, criação de usuários e gerenciamento de usuários do sistema.

O projeto foi criado através da biblioteca Vite em sua versão 5.4.10, sendo responsável por instanciar uma estrutura básica com arquivos de configurações e arquivos necessários para a execução correta do React. Algumas bibliotecas foram utilizadas para agilizar o processo de desenvolvimento, entre elas estão:

[3] <https://app.swaggerhub.com/home>

- *React Hook Form* e *Zod* - Utilizada para realizar a validação de formulários. Seu uso é destinado para garantir que os dados inseridos estejam no formato e padrão adequados para o envio dos dados a API.
- *Tailwind CSS* - Responsável por toda a estilização das telas.
- *axios* - Centralizar a realização de requisições e o recebimento de respostas e erros.
- *React Router Dom* - Gerencia as rotas, permitindo o controle e configuração de rotas protegidas por autenticação e exposição de rotas públicas.

4.4.3.1 Autenticação e Autorização

Quanto a autenticação para ter acesso aos recursos do *back-end*, era necessário, inicialmente, realizar o *login*. Tendo as credenciais válidas, como resposta da autenticação, o navegador recebe o *token* de acesso gerado no servidor e armazena como *cookie*. Armazenar dados em forma de *cookies* permite com que informações sejam guardadas em um espaço de memória exclusivo para o site sendo acessado. Isso traz uma maior segurança quando esta-se trabalhando com persistência volátil no navegador.

Como critério para se manter autenticado no sistema, o usuário deve ter efetuado o *login* e não fechar o navegador. Caso haja o fechamento do navegador, o *token* de acesso é removido dos *cookies* do site e na próxima vez será solicitada uma nova autenticação quando o sistema for acessado.

No que envolve a autenticação, foi utilizando o mesmo conceito que no back-end. Cada funcionário com credenciais no sistema somente terá acesso aos recursos atribuídos a sua função dentro da empresa, sem permitir com que usuários não autorizados façam modificações que são cabíveis somente para cargos de gestão, como administradores e gerentes.

5 RESULTADOS

Com a finalização deste trabalho, foi deixado como material de documentação do sistema alguns artefatos que dão mais clareza sobre a direção que o desenvolvimento tomou, sendo uma forma de compreender como o desenvolvimento do software decorreu ao longo da sua criação. Dentre os artefatos que surgiram, é possível citar os requisitos do sistema, que fornecem um visão geral sobre funcionalidades e características que o software irá conter. É importante falar também dos casos de uso, que compreendem os envolvidos na realização de alguma ação por parte do sistema, detalhando condições de uso e até mesmo condições que devem ser satisfeitas assim que esta tarefa for concluída, contendo também a importância dada a determinada ação.

Os artefatos acima mencionados encontram-se na Seção 6.1, no capítulo de anexos. A organização das tarefas deu-se através da definição de um escopo maior, mas que internamente possui sub-tarefas para chegar a conclusão da tarefa maior. Este formato foi adotado para manter um maior controle sobre as atividades de um dado escopo, auxiliando no entendimento do que era necessário para se chegar a conclusão de um módulo de desenvolvimento.

Ao todo, foram entregues 40 itens ao longo das 4 *sprints* que foram realizadas, apresentando uma média de 10 entregáveis por *sprint*. É claro que deve ser considerado que as tarefas do período inicial tendem a ter um volume maior de entregas por conter tarefas de configuração de ambiente. Porém ainda assim a quantidade total entregue dentro do período de desenvolvimento deve ser destacado devido a fatores como carência de algumas habilidades no desenvolvimento e a concorrência do tempo para desenvolver com outras atividades da graduação.

Foram aproximadamente 2 meses de desenvolvimento que, ao findaram, possibilitam com que os usuários do sistema tivessem a capacidade de gerenciar os produtos em estoque, seja cadastrando novos, atualizando existentes ou removendo produto que saíram de produção. Além disso, o sistema permite que o administrador do sistema adicione novos funcionários, possibilitando a visualização simplificado do corpo de funcionários da loja. Ainda também, permite a adição de credenciais ligadas a funcionários, que passam a ter acesso ao sistema e a possibilidade de colaborar em tarefas que condizem com o nível de acesso aos dados, que é baseado em seu cargo organizacional. Esses pontos mostram um pouco da dimensão da eficácia da abordagem quando aplicada em projetos em que o trabalho é atribuído a uma única pessoa.

Quanto ao resultados em relação ao sistema desenvolvido, foram concluídas a codificação dos módulos de Produtos e Funcionários, trazendo um bom número de funcionalidades para o sistema. Dos módulos acima mencionados, apenas o módulo de Produtos apresenta testes unitários sobre as funcionalidades desenvolvidas, permitindo a experimentação do comportamento e da real completude do módulo.

6 CONSIDERAÇÕES FINAIS

Neste trabalho, foi realizado o desenvolvimento de dois módulos de um sistema ERP para uma loja do comércio varejista. Foi feito uso do *Personal Scrum*, que é uma abordagem que faz uso dos moldes do Scrum corporativo e adapta sua usabilidade de forma com que uma pessoa só seja capaz de utilizar e se beneficiar das vantagens que uma metodologia conduzida por em equipes de grandes empresas pode fornecer. Com este trabalho, é possível entender como a utilização de uma metodologia pode impactar significativamente no todo de um projeto, desde a sua criação até a etapa de entrega de versão.

6.0.1 Conclusão

A aplicação do *Personal Scrum* no desenvolvimento deste trabalho trouxe diversas vantagens. Porém o que mais se destaca é a organização alcançada através desta abordagem, além da contribuição para a manutenção do foco no período de cada *sprint*. Este foco reflete diretamente na qualidade e na quantidade dos entregáveis desenvolvidos.

É importante destacar também a capacidade de desenvolvimento do senso de comprometimento que é adquirido quando se participa de todos os processos da organização do projeto, desempenhando os 3 papéis fundamentais do *Scrum* (*Scrum Master, Product Owner e Develop Team*) simultaneamente, papéis esses que são apresentados em sua essência original. A utilização do *Personal Scrum* mostrou-se extremamente positiva, onde, mesmo sendo aplicado em um software de complexidade de média para alta, foi possível conduzir a sua aplicação até mesmo em um cenário onde não havia uma vasta experiência por parte do aplicante da metodologia. Isso destaca ainda mais o poder de condução de projeto que esta abordagem traz, tendo uma grande aplicabilidade, tornando o cenário da aplicação apenas um contextualizador da sua utilização, não condicionando as suas vantagens obtidas ao se utilizar da mesma, tornando-as sempre mensuráveis.

6.1 Trabalhos Futuros

Visando uma melhoria no software, pretende-se realizar a criação dos outros requisitos que não foram concluídos durante o desenvolvimento deste trabalho, principalmente os componentes de gerenciamento de vendas e clientes.

Existe também a ideia de realizar a implantação do sistema em ambiente de nuvem pública, visando a redução de custos com equipamentos locais, além de oferecer uma maior segurança sobre dados armazenados, como também elevar o tempo de disponibilidade do sistema. Em um momento posterior, quer-se realizar a refatoração do sistema, saindo de uma arquitetura cliente-servidor para a arquitetura de micro-serviços. Isso objetiva desacoplar serviços, fazendo com que a inatividade de um serviço não interfira na execução dos demais.

Seria ideal realizar a aplicação da metodologia de forma comparativa com outra metodologia, com finalidade de comparação com outra abordagem, diferentemente da comparação feita no artigo do *Personal Scrum* que visou mensurar os ganhos ao utilizar esta abordagem em relação a condução do desenvolvimento de forma totalmente desordenada e independente. Essa ideia visa trazer uma visão mais abrangente de como uma abordagem conduz um projeto e qual poderia ser a mais apropriada de acordo com o contexto que fosse aplicado.

Anexos

ANEXO A – TELAS DO SISTEMA

As telas da aplicação estão dispostas na Figura 10, Figura 11, Figura 12, Figura 7, Figura 13 e Figura 14. As imagens descrevem respectivamente as funcionalidades incluídas para cada referida tela: *Login*, Início do Sistema, Gerenciamento de Funcionários, Listagem de Produtos, Criação de Produtos e Criação de Opções para um Produto.

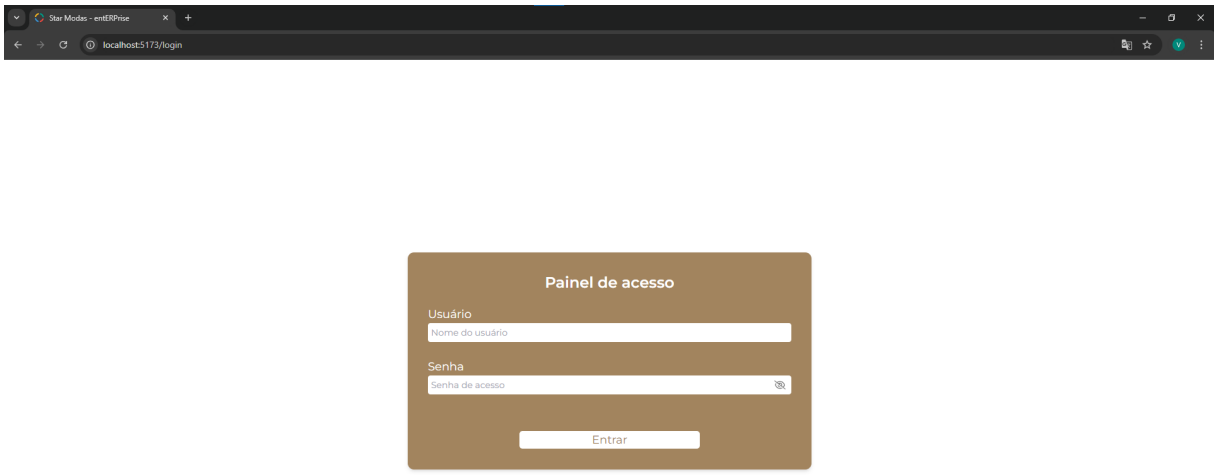

Figura 10 – Tela de login

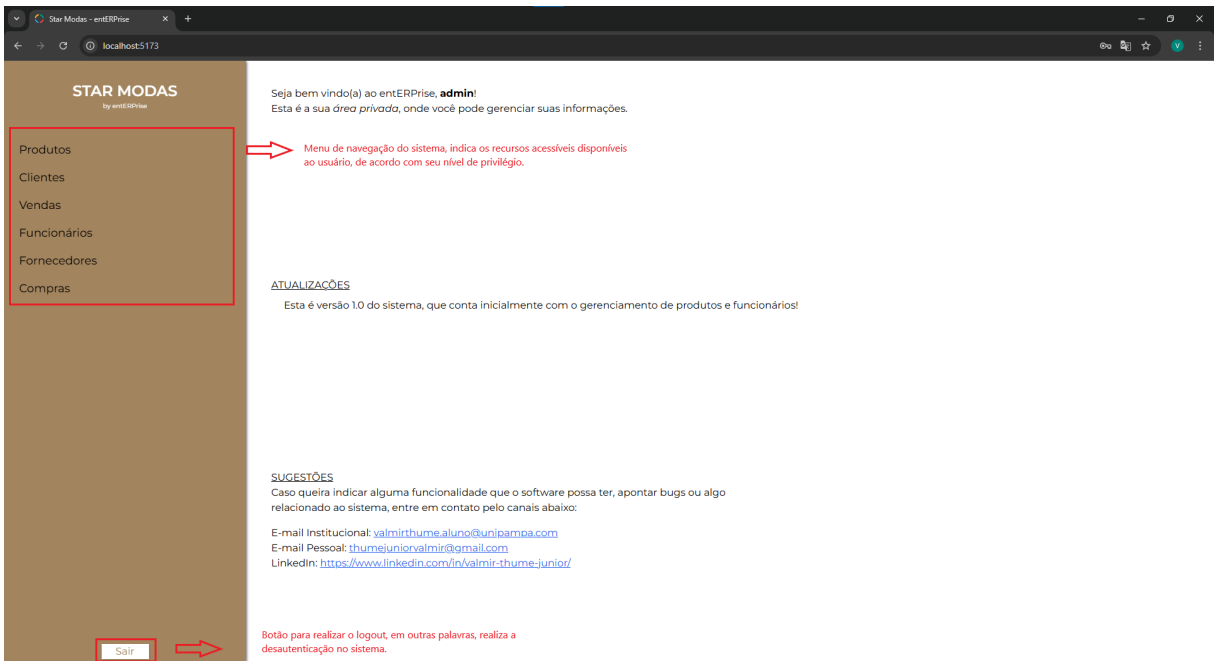

Figura 11 – Tela inicial do sistema

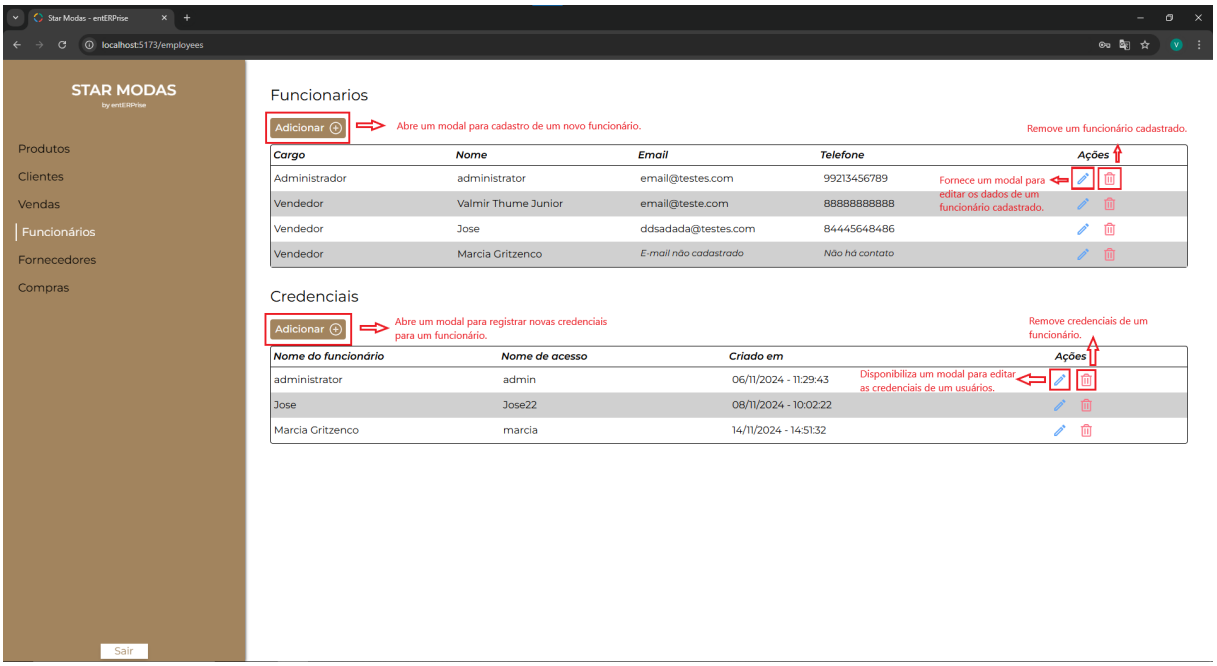

Figura 12 – Tela de gerenciamento de funcionários

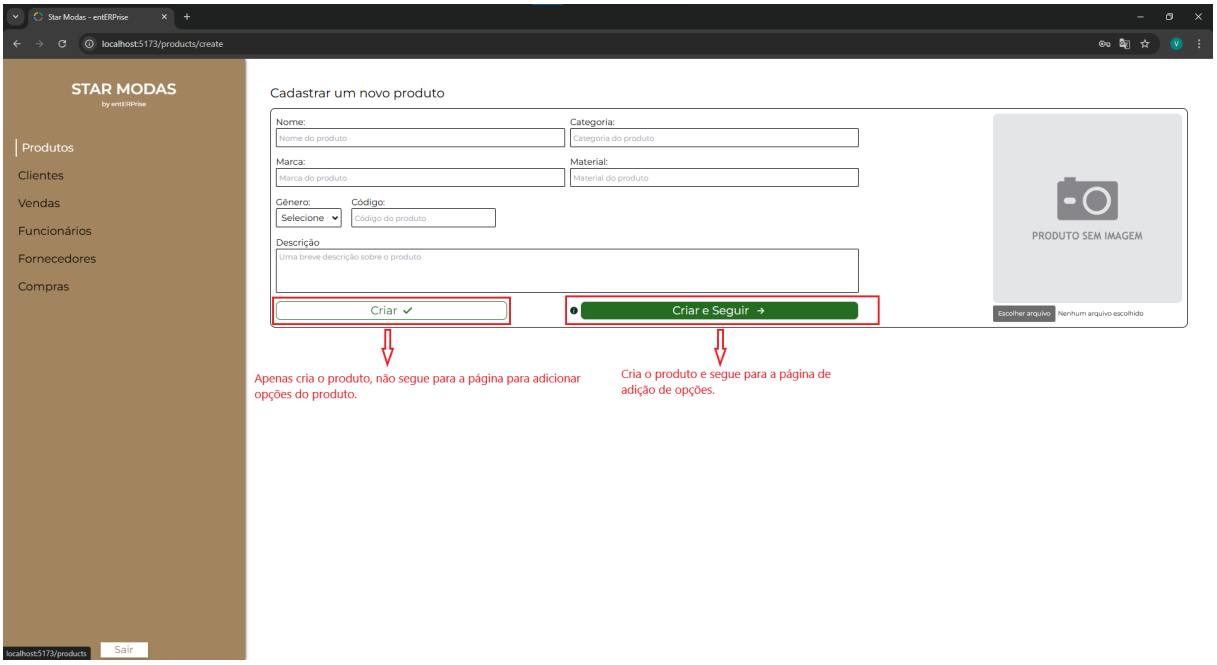

Figura 13 – Tela de criação de produtos

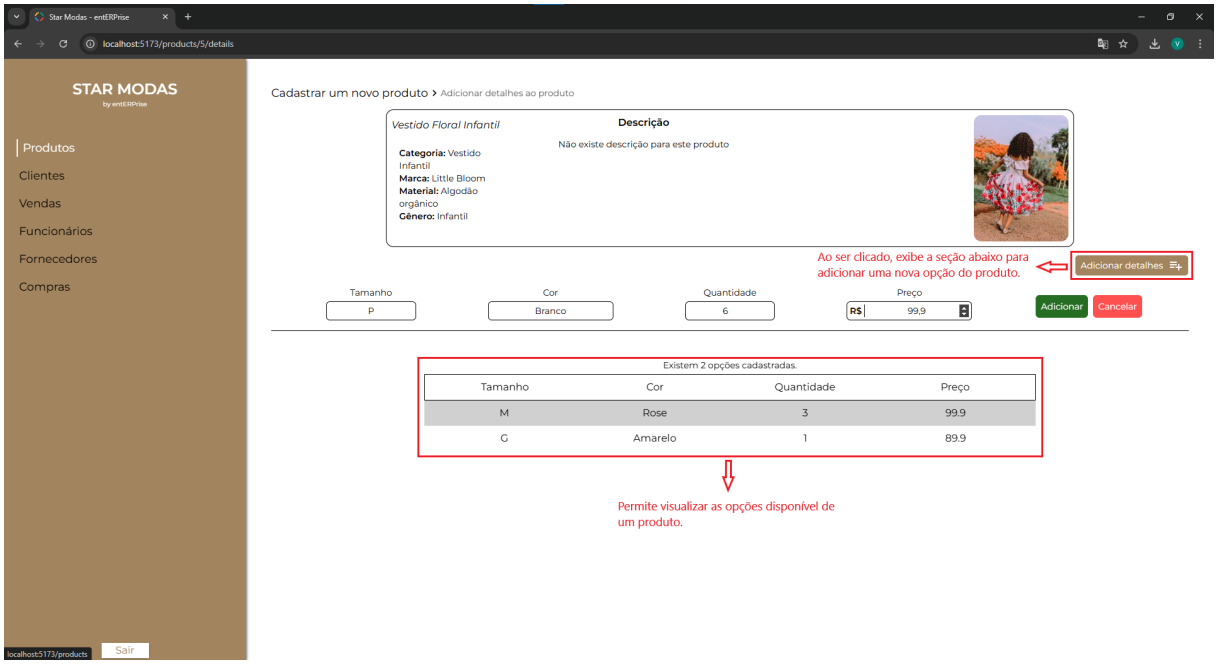

Figura 14 – Tela de criação de opções para um produto

ANEXO B – DOCUMENTAÇÃO

A Figura 15 apresenta as rotas que disponíveis na API. Exceto a rota de login, todas as demais são rotas protegidas que necessitam de autenticação e autorização. A documentação que permite explorar cada rota, seus parâmetros, corpos de requisição e respostas está disponível no link:

https://app.swaggerhub.com/apis/THUMEJUNIORVALMIR/entERPrise/1.0.0

Figura 15 – Documentação da API desenvolvida

ANEXO C – CASOS DE USO

Os casos de uso construídos foram baseados nos requisitos captados na etapa de Análise e são apresentados na Figura 16 e Figura 17. A primeira figura apresenta os casos de uso de 1 a 3, enquanto que a segunda figura exibe os casos de uso de 4 a 6.

Nome do Caso de Uso: Cadastro de Produtos	**ID:** UC01	**Prioridade:** Alta
Ator: Administrador e Gerente		
Descrição: Permite cadastrar novos produtos no sistema, incluindo informações como nome, código, preço e quantidade.		
Gatilho: O ator decide cadastrar um novo produto.		
Tipo: Externa		
Pré-condições: - O ator deve estar autenticado no sistema com permissões de acesso apropriadas.		
Curso Normal: - O ator acessa a seção de produtos. - O ator clica em "Adicionar Produto". - O ator insere as informações do produto (nome, tags, preço, cod. da loja e cod. do fabricante). - O ator confirma o cadastro. - O sistema salva o novo produto e exibe uma mensagem de sucesso.		
Pós-condições: - O produto é adicionado e fica disponível para consulta, edição e inclusão no estoque.		
Exceções: **-** Se informações obrigatórias estiverem ausentes, o sistema exibe uma mensagem de erro e solicita correção. **-** Se existir produto semelhante, alertar a similaridade entre os produtos.		

Nome do Caso de Uso: Venda de Produto	**ID:** UC02	**Prioridade:** Alta
Ator: Administrador e Gerente		
Descrição: Registrar uma venda de um ou mais produtos no sistema.		
Gatilho: Um cliente realiza uma compra.		
Tipo: Externa		
Pré-condições: - O ator deve estar autenticado no sistema com permissões para realizar vendas.		
Curso Normal: - O ator acessa a seção de vendas. - O ator seleciona os produtos que o cliente está comprando. - O ator insere a quantidade e confirma a venda. - O sistema registra a venda, atualiza o estoque e gera um recibo.		
Pós-condições: - O estoque é atualizado e a venda é registrada.		
Exceções: **-** Se não houver quantidade suficiente no estoque, o sistema exibe uma mensagem de erro e a venda não é concluída.		

Nome do Caso de Uso: Alerta de Baixo Estoque	**ID:** UC03	**Prioridade:** Alta
Ator: Sistema		
Descrição: Notifica a proprietária e funcionários sobre produtos com baixo estoque.		
Gatilho: Um produto atinge a quantidade mínima configurada.		
Tipo: Temporal		
Pré-condições: - Produtos devem estar cadastrados no sistema e os níveis de alerta devem estar configurados.		
Curso Normal: - O sistema monitora o estoque. - Quando a quantidade de um produto atinge o nível mínimo, o sistema envia um alerta aos usuários apropriados.		
Pós-condições: - A proprietária e funcionários são notificados e podem tomar as ações necessárias.		
Exceções: - Se o sistema não conseguir enviar a notificação (ex.: erro de comunicação), o alerta será exibido na próxima tentativa de acesso ao sistema.		

Figura 16 – Casos de uso do 1 ao 3

Nome do Caso de Uso: Controle de Privilégios de Usuário	**ID:** UC04	**Prioridade:** Alta
Ator: Administrador		
Descrição: Permite definir os níveis de acesso dos funcionários no sistema.		
Gatilho: A proprietária deseja atribuir ou alterar privilégios de acesso de um funcionário.		
Tipo: Externa		
Pré-condições: - A proprietária deve estar autenticada no sistema.		
Curso Normal: - A proprietária acessa a seção de usuários. - A proprietária seleciona o funcionário desejado. - A proprietária define ou altera os privilégios de acesso. - O sistema salva as alterações.		
Pós-condições: - O funcionário passa a ter o nível de acesso atualizado.		
Exceções: - Se houver erro ao salvar as permissões, o sistema exibe uma mensagem de erro e solicita uma nova tentativa.		

Nome do Caso de Uso: Geração de Relatórios	**ID:** UC05	**Prioridade:** Média
Ator: Administrador		
Descrição: Gera relatórios de vendas e estoque.		
Gatilho: A proprietária deseja analisar o desempenho do negócio.		
Tipo: Externa		
Pré-condições: - O ator deve estar autenticado no sistema com permissões de acesso apropriadas.		
Curso Normal: - A proprietária acessa a seção de relatórios. - A proprietária seleciona o tipo de relatório (vendas ou estoque). - A proprietária seleciona o período desejado. - O sistema gera o relatório e exibe as informações.		
Pós-condições: - O relatório é exibido e pode ser exportado.		
Exceções: - Se não houver dados para o período selecionado, o sistema exibe uma mensagem informativa.		

Nome do Caso de Uso: Cadastro de Cliente	**ID:** UC06	**Prioridade:** Média
Ator: Administrador e Gerente		
Descrição: Permite cadastrar um cliente, incluindo informações como nome, endereço, telefone e e-mail.		
Gatilho: O ator deseja cadastrar um novo cliente.		
Tipo: Externa		
Pré-condições: - O ator deve estar autenticado no sistema com permissões de acesso apropriadas.		
Curso Normal: - O ator acessa a seção de clientes. - O ator clica em "Adicionar Cliente". - O ator insere as informações do cliente (nome, endereço, telefone, e-mail). - O ator confirma o cadastro. - O sistema salva as informações do cliente e exibe uma mensagem de sucesso.		
Pós-condições: - O cliente é adicionado ao sistema e pode ser vinculado a vendas.		
Exceções: - Se informações obrigatórias estiverem ausentes, o sistema exibe uma mensagem de erro e solicita correção.		

Figura 17 – Casos de uso do 4 ao 6

REFERÊNCIAS

ALDAY, H. E. C.; PINOCHET, L. H. C. A tecnologia e-commerce como estratégia determinante no setor supermercadista. **Revista da FAE**, v. 5, n. 3, jan. 2017. Disponível em: <https://revistafae.fae.edu/revistafae/article/view/482>. Citado na página 17.

Awari Atividades de Ensino LTDA. **Aprenda Como Utilizar o Node.Js para Desenvolvimento de Backend**. 2024. <https://awari.com.br/aprenda-como-utilizar-o-node-js-para-desenvolvimento-de-backend>. [Acessado em 11-11-2024]. Citado na página 29.

BERALDI, L. C.; FILHO, E. E. Impacto da tecnologia de informação na gestão de pequenas empresas. **Ciência da Informação**, SciELO Brasil, v. 29, p. 46–50, 2000. Citado na página 17.

BESSA, A. **O que é Node.JS? Como funciona e um Guia para iniciar**. 2024. <https://www.alura.com.br/artigos/node-js>. [Acessado em 08-11-2024]. Citado na página 29.

COM2BUSINESS. **Com2 Business**. 2024. <https://com2business.com.br>. [Acessado em 06-11-2024]. Citado na página 21.

COSTA, K. R. N. **Personal scrum: Uma Alternativa ágil Para Desenvolvimento de Indie Games**. Universidade Federal do Pampa, 2016. Disponível em: <https://repositorio.unipampa.edu.br/jspui/handle/riu/1853>. Citado na página 26.

DENNIS, A.; WIXOM, B. H.; ROTH, R. M. **Análise e projeto de sistemas**. 5. ed. Rio de Janeiro: LTC, 2014. Citado 2 vezes nas páginas 31 e 32.

EMPRESARIAL, S. S. T. para G. **ERP Gestão Empresarial | Senior Sistemas**. 2024. <https://www.senior.com.br/solucoes/sistema-erp-gestao-empresarial>. [Acessado em 06-11-2024]. Citado na página 20.

ERP, C. **Cake ERP**. 2024. Https://cakeerp.com. [Acessado em 06-11-2024]. Citado na página 22.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. Tese (Publication) — University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado na página 39.

GESTAOCLICK. **ERP para Varejo**. 2024. <https://gestaoclick.com.br/erp-para-varejo>. [Acessado em 06-11-2024]. Citado na página 22.

GIT. **About - Git**. 2024. <https://git-scm.com/about/branching-and-merging>. [Acessado em 22-11-2024]. Citado na página 26.

Governo Federal. **Lei No. 8.846**. Brasília, 1994. Citado na página 22.

JEST. **Delightful JavaScript Testing**. 2024. <https://jestjs.io/pt-BR>. [Acessado em 08-11-2024]. Citado na página 29.

KINSTA. **Git vs GitHub: Qual é a Diferencca e Como Começar com Ambos**. 2024. <https://kinsta.com/pt/base-de-conhecimento/git-vs-github/#uma-introduo-ao-git-e-ao-controle-de-verses>. [Acessado em 22-11-2024]. Citado na página 26.

LIE, V. **O impacto da tecnologia no varejo: 4 exemplos práticos**. 2021. <https://www.ecommercebrasil.com.br/artigos/o-impacto-da-tecnologia-no-varejo-4-exemplos-praticos>. [Acessado em 07-11-2024]. Citado na página 17.

LINX. **Linx Sistema de Gestão ERP e PDV**. 2024. <https://www.linx.com.br/sistema-gestao-erp>. [Acessado em 06-11-2024]. Citado na página 22.

MATUTE, S. A.; AVILA-PESANTEZ, D.; AVILA, L. M. Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso. **Revista peruana de computación y sistemas**, v. 3, n. 1, 2020. Citado na página 19.

MELO, D. **O que é TypeScript? [Guia para iniciantes]**. 2020. <https://tecnoblog.net/responde/o-que-e-typescript-guia-para-iniciantes>. [Acessado em 13-11-2024]. Citado na página 29.

MICROSOFT. **Introdução ao ASP.NET Core MVC**. 2024. <https://learn.microsoft.com/pt-br/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website&tabs=visual-studio>. [Acessado em 25-11-2024]. Citado na página 28.

MICROSOFT. **Visual Studio Code - Code Editing. Redefined**. 2024. <https://code.visualstudio.com>. [Acessado em 21-11-2024]. Citado na página 26.

PRICEFY. **Como a integração de sistemas pode contribuir na gestão do varejo**. 2024. <https://www.pricefy.com.br/blog/como-a-integracao-de-sistemas-pode-contribuir-na-gestao-do-varejo>. Citado na página 17.

REACT. **React**. 2024. <https://react.dev>. [Acessado em 08-11-2024]. Citado na página 28.

REACT. **Virtual DOM e Objetos Internos**. 2024. Https://pt-br.legacy.reactjs.org/docs/faq-internals.html. [Acessado em 08-11-2024]. Citado na página 29.

Scrum Guide. **scrumguides.org**. 2020. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>. [Acessado em 26-11-2024]. Citado na página 26.

SILVA, G. **O que é arquitetura MVC?** 2024. <https://coodesh.com/blog/dicionario/o-que-e-arquitetura-mvc>. [Acessado em 27-11-2024]. Citado na página 28.

TECH, V. **Sistema de Gestão Empresarial Online | UpGestão**. 2024. <https://upgestao.com.br>. [Acessado em 06-11-2024]. Citado na página 20.

TOTVS. **Sistema de Gestão ERP**. 2024. <https://www.totvs.com/moda/?utm_campaign[0]=s-varejo>. [Acessado em 06-11-2024]. Citado na página 21.

VENDASIMPLES. **Sistema de Gestão Online com Emissão de NFe | VendaSimples**. 2024. <https://vendasimples.com.br/index.html>. [Acessado em 06-11-2024]. Citado na página 19.