UNIVERSIDADE PAULISTA

ENZZO CANSANI MORAIS
FAUSTO NERIS SILVA
GUILHERME DE CARVALHO RIBEIRO
IGOR SOUSA DA SILVA
RAFAEL LIMA FRANCA

E-COMMERCE INVERTIDO: VALORIZANDO MICROEMPREENDEDORES

Exploração de um modelo inovador de comércio online para ampliar a visibilidade e marketing de microempreendedores

SANTANA DE PARNAÍBA - SP
2024

ENZZO CANSANI MORAIS – G2332E0

FAUSTO NERIS SILVA – G2463B8

GUILHERME DE CARVALHO RIBEIRO – T0139G1

IGOR SOUSA DA SILVA – T055822

RAFAEL LIMA FRANCA – N732AE1

E-COMMERCE INVERTIDO: VALORIZANDO MICROEMPREENDEDORES

Exploração de um modelo inovador de comércio online para ampliar a visibilidade e marketing de microempreendedores

Trabalho de conclusão de curso para a obtenção do título de graduação em Ciência da Computação apresentado à Universidade Paulista – UNIP.

Orientador Prof. Me. Edy Carlos Hidemi Hayashida

SANTANA DE PARNAÍBA – SP

2024

E-COMMERCE INVERTIDO: VALORIZANDO MICROEMPREENDEDORES / Enzzo Cansani; Fausto Neris; Guilherme Carvalho; Igor Sousa; Rafael Lima ...[et al.]. - 2024.
85 f. : il. color

Trabalho de Conclusão de Curso (Graduação) apresentado ao Instituto de Ciência Exatas e Tecnologia da Universidade Paulista, Santana de Parnaíba, 2024.
Área de Concentração: Tecnologia.
Orientador: Prof. Me. Edy Carlos Hidemi Hayashida .

1. E-commerce invertido. 2. E-commerce. 3. Microempreendedor. 4. Tecnologia. 5. Leilão. I. , Enzzo Cansani; Fausto Neris; Guilherme Carvalho; Igor Sousa; Rafael Lima. II. , Edy Carlos Hidemi Hayashida (orientador).

ENZZO CANSANI – G2332E0

FAUSTO NERIS SILVA – G2463B8

GUILHERME DE CARVALHO RIBEIRO – T0139G1

IGOR SOUSA DA SILVA – T055822

RAFAEL LIMA FRANCA – N732AE1

E-COMMERCE INVERTIDO: VALORIZANDO MICROEMPREENDEDORES

Exploração de um modelo inovador de comércio online para ampliar a visibilidade e marketing de microempreendedores

Trabalho de conclusão de curso para a obtenção do título de graduação em Ciência da Computação apresentado à Universidade Paulista – UNIP.

Aprovado em: 27/11/2024

BANCA EXAMINADORA

_____/___/_____

Prof. Edy Hayashida

Universidade Paulista – UNIP

_____/___/_____

Prof. Adriano Domingues

Universidade Paulista – UNIP

_____/___/_____

Prof. Marcelo Castro

Universidade Paulista – UNIP

RESUMO

Este trabalho apresenta o desenvolvimento de um e-commerce invertido, que vem a ser uma plataforma inovadora, no qual o seu foco é a ampliação da visibilidade dos microempreendedores e criação de um ambiente de compra mais personalizado e humanizado, permitindo uma maior interação entre quem compra e quem fornece. Em um modelo tradicional de e-commerce, o cliente adquire diretamente o produto, já nesse sistema, o diferencial é que o usuário tem a possiblidade de oferecer um lance por ele, ficando a critério do fornecedor o aceite ou recusa da oferta. Esse tipo de negociação direta é uma maneira de valorização ao pequeno empresário. Com objetivo de vencer os desafios enfrentados pelas micro e pequenas empresas (como restrições de investimento em marketing digital e acesso limitado a tecnologias), o sistema busca fornecer uma visibilidade maior e uma interface de uso facilitado, algo que favorece tanto o consumidor quanto o empreendedor. Funcionalidades técnicas, como avaliações dos produtos e um canal de comunicação direto com o fornecedor, foram integradas para garantir uma experiência interativa, segura e que fortaleça os laços entre as partes envolvidas. A arquitetura do sistema utiliza tecnologias como Angular, AWS, NoSQL, Java, entre outros, tudo visando manter um desempenho, escalabilidade e fácil manutenção. Além disso, a segurança das informações que são trafegadas, estão de acordo e alinhadas com as normas da LGPD. A plataforma não só amplia o alcance dos microempreendedores ao mercado digital, como também fortalece a economia local, disponibilizando oportunidades de crescimento. Com um grande potencial para futuras melhorias e atualizações, o sistema poderá agregar inteligência artificial para uma melhor experiência do usuário e relatórios personalizados aos fornecedores. De maneira geral, o e-commerce invertido é uma alternativa viável e inclusiva do comércio eletrônico para benefício de pequenos empresários quanto consumidores conscientes.

Palavra-chave: E-commerce invertido, e-commerce, microempreendedores, economia.

ABSTRACT

This work presents the development of a reverse e-commerce platform, an innovative platform aimed at increasing visibility for micro-entrepreneurs and creating a more personalized and humanized shopping environment, allowing for greater interaction between buyers and suppliers. In a traditional e-commerce model, the customer directly purchases the product; however, in this system, the unique feature is that the user has the option to place a bid on it, with the supplier having the discretion to accept or decline the offer. This type of direct negotiation is a way to support small businesses. Aiming to address challenges faced by micro and small businesses (such as limited investment in digital marketing and restricted access to technology), the system seeks to provide greater visibility and an easy-to-use interface that benefits both consumers and entrepreneurs. Technical features like product reviews, and a direct communication channel with suppliers were integrated to ensure an interactive, secure experience that strengthens the connections between the involved parties. The system architecture utilizes technologies such as Angular, AWS, NoSQL, Java, among others, all aimed at maintaining performance, scalability, and easy maintenance. Additionally, information security complies with and aligns with LGPD regulations. The platform not only expands micro-entrepreneurs' access to the digital market but also strengthens the local economy by providing growth opportunities. With significant potential for future improvements and updates, the system may incorporate artificial intelligence for an enhanced user experience and personalized reports for suppliers. Overall, the reverse e-commerce model is a viable and inclusive alternative in e-commerce, benefiting both small business owners and conscious consumers.

KEYWORD: Reverse e-commerce, e-commerce, micro-entrepreneurs, economy.

LISTA DE ILUSTRAÇÕES

Figura 1 - Modelo Entidade Relacionamento - MER 47
Figura 2 - Caso de Uso 48
Figura 3 - Diagrama de Classe - Tela de login 50
Figura 4 - Diagrama de Classe - Tela de Produto 51
Figura 5 - Diagrama de Classe - Listagem de Produtos 52
Figura 6 - Diagrama de Classe – Feedback 53

LISTA DE ABREVIATURAS e SIGLAS

API – Application Programming Interface

AWS – Amazon Web Services

BERT - Bidirectional Encoder Representations for Transformers

BFF – Back-end for Front-end

B2B – Business to business

B2C – Business to consumer

B2E – Business to Employee

CRUD – Create Read Update Delete

CSS - Cascading Style Sheets

DevOps – Development Operations

DevSecOps – Development Security Operations

DL – Deep Learning

EDI – Electronic Data Interchange

EFT – Electronic Funds Transfer

ERP – Enterprise Resource Planning

GCP – Google Cloud Platform

GCS – Gestão da Cadeia de Suprimentos

GPT - Generative Pre-trained Transformer

GQ – Gestão de Qualidade

GQCS – Gestão da Qualidade na Cadeia de Suprimentos

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IA - Inteligência Artificial

JDK – Java Development Kit

JRE – Java Runtime Environment

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

LGPD – Lei Geral de Proteção de Dados

MEI – Microempreendedor Individual

ML – Machine Learning

MMOG - Massively Multiplayer Online Games

MPE – Micro e Pequenas Empresas

MVC – Model View Control

NER - Named Entity Recognition

NLP – Natural Language Processing

NLU – Natural Language Understanding

POO – Programação Orientada a Objetos

P2P – Peer-to-Peer (Ponto a ponto)

SEBRAE – Serviço Brasileiro de Apoio às Micro e Pequenas Empresas

SES – Simple E-mail Service

SOA - Service Oriented Architecture (Arquitetura Orientada a Serviços)

SQL – Structured Query Language

SUMÁRIO

1 INTRODUÇÃO 15
1.1 Problematização / Motivação 16
1.2 Objetivo Geral 17
1.3 Objetivos Específicos 17
1.4 Metodologia 18
1.5 Organização do Texto 19
2 REVISÃO DA LITERATURA 20
2.1 Tecnologia 20
2.2 Benefícios da tecnologia 21
2.3 Benefícios socioeconômicos que a tecnologia fornece 22
2.4 Lei Geral de Proteção de Dados (LGPD) 23
2.5 Relação Cliente X Fornecedor 23
2.6 Microempreendedores 24
2.7 *E-commerces* 25
2.8 *Business-to-Business* (B2B) 26
2.9 *Business-to-Consumer* (B2C) 26
2.10 *Business-to-Employee* (B2E) 27
2.11 Controle de qualidade nos serviços entregues 28
2.12 Sistemas Distribuídos 28
2.12.1 Modelo Cliente-Servidor 29
2.12.2 Modelo *Peer-to-Peer* (P2P) 29
2.12.3 Modelo de Arquitetura Orientada a Serviços (SOA) 30
2.12.4 Modelo de Computação em Nuvem 30
2.13 Algoritmos 30
2.14 Inteligência Artificial (IA) 31
2.15 *Machine Learning* (ML) 32
2.16 *Deep Learning* (DL) 33
2.17 *Chatbot* 34
2.18 *Natural Language Processing* (NLP) 35
2.19 *Natural Language Understanding* (NLU) 36
3 MATERIAIS E MÉTODOS 37
3.1 Linguagens utilizadas 37

3.1.1 Ferramentas e suas vantagens....38
3.1.1.1 Angular....38
3.1.1.2 Amazon Web Services (AWS)....39
3.1.1.3 CSS....39
3.1.1.4 DevOps....40
3.1.1.5 HTML....40
3.1.1.6 Java....41
3.1.1.7 Node....42
3.1.1.8 NoSQL....42
3.1.1.9 SQL....43
3.1.1.10 Typescript....44
3.2 Sistemas Distribuídos....44
3.3 Requisitos do sistema....45
3.3.1 Requisitos funcionais....45
3.3.2 Requisitos não funcionais....46
3.4 Entidades....46
3.5 Casos de uso....48
3.6 Diagrama de classe....49
3.6.1 Login....49
3.6.2 Produto (Visão fornecedor)....51
3.6.3 Produto (Visão usuário)....52
3.6.4 Feedback....53
3.7 Algoritmos....54
3.7.1 Login....54
3.7.1.1 *Controllers – Login*....55
3.7.1.2 *Modules – Login*....55
3.7.1.2.1 *user-types.enum.ts*....55
3.7.1.2.2 *user-authentication.model.ts*....56
3.7.1.3 *user.model.ts*....56
3.7.1.4 *Repositories – Login*....57
3.7.1.5 *Services – Login*....58
3.7.2 Registro de Produto....59
3.7.2.1 *product-register.component.ts*....59
3.7.2.2 *suppliers.controller.ts*....61

3.7.2.3 *supplier.repository.ts* 62
3.7.2.4 suppliers.service.ts 64
3.7.2.5 *SupplierProductController.java* 64
3.7.2.6 SupplierProductRepository.java 65
3.7.2.7 *SupplierProductService.java* 66
3.7.3 Listagem de Produtos 67
3.7.3.1 *list-products.component.ts* 67
3.7.3.2 *customers.controller.ts* 70
3.7.3.3 *customers.repository.ts* 71
3.7.3.4 *customers.service.ts* 73
3.7.3.5 *ProductController.java* 74
3.7.3.6 *ProductRepository.java* 75
3.7.3.7 *ProductService.java* 76
3.7.4 Link GitHub 77
4 RESULTADOS 78
5 CONCLUSÃO 79
REFERÊNCIAS BIBLIOGRÁFICAS 81

1 INTRODUÇÃO

De acordo com Kenski (2012), o termo tecnologia vai muito além do que geralmente é conhecido, se referindo apenas as máquinas. Segundo ela, essa expressão envolve todo o tipo de criação que o ser humano fez no decorrer de todos os séculos, suas maneiras de utilização e aplicação.

Em vista disso, é de conhecimento geral que a tecnologia se tornou algo crucial atualmente, estando presente em basicamente tudo que fazemos, seja uma transferência bancária, aparelhos de realidade virtual, os próprios celulares, notebook, e muitos outros exemplos. Consequentemente, gera uma certa proximidade e um aumento no relacionamento entre pessoas, via redes sociais (KENSKI, 2012). Através delas, os usuários estão diariamente expostos a diversos anúncios persuasivos, estimulando a comprar tal produto, mostrando inúmeras vantagens em adquiri-lo. Dessa maneira, levanta em questão um modelo de negócio que é popular globalmente, que são os *e-commerces*, no qual o usuário pesquisa um produto, verifica vários do mesmo, com preços diferentes, fornecedores diferentes e fecha uma compra com o que melhor te agrada. (PAIVA, 2021)

Dado o contexto acima, surgiu uma ideia inovadora, de realizar um "*e-commerce* invertido", em que visa dar uma dinâmica ao comércio *online*, dando foco principal aos microempreendedores, e deixando um pouco de lado as grandes marcas.

Ao realizar uma pesquisa no site, o usuário terá acesso a diversas informações relacionadas ao produto ou serviço, avaliações realizadas pelos clientes, e o diferencial, detalhamento sobre o fornecedor/prestador do serviço, como seu contato, um pouco sobre o seu trabalho. Como a plataforma visa focar no microempreendedor, deixá-lo em evidência no mercado, ele poderá fazer seu marketing, convencer o usuário do porquê deve comprar seu produto ou aderir seus serviços.

O usuário é o principal alvo a ser atingido com o desenvolvimento do projeto, pois ele que fará o sistema caminhar. Dito isso, um fator que busca beneficiá-lo com a utilização da nossa plataforma, é que ao realizar uma busca de algo do seu interesse, o sistema lhe apresentará um sistema de lance, no qual cada fornecedor informa o menor preço a ofertar pelo seu serviço ou produto, além de pontos fidelidade, entre outras recompensas que o cliente poderá adquirir ao comprar com determinado microempreendedor.

O desenvolvimento desse projeto busca explorar, além de vários aspectos técnicos, mas também o lado socioeconômico, já que a visibilidade ampliada que será dada aos pequenos empresários, acarretará um leque maior de diversidade de produtos no mercado *online*, tanto quanto oportunidades econômicas em comércios e comunidades locais.

1.1 Problematização / Motivação

De acordo com dados divulgados do Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE, 2024), o Brasil é ranqueado como o terceiro país com maior número de pequenos negócios ativos, sendo que as micro e pequenas empresas (MPEs) respondem por 52% dos empregos com carteira assinada no setor privado.

Apesar de responderem por 99% dos estabelecimentos do Brasil, essas micro e pequenas empresas enfrentam grandes desafios. Um deles é a falta de investimento financeiro, gerando uma remuneração menos atrativa. Além desse, um dos que mais se destaca, é a falta de financiamento para investir em soluções mais produtivas, fazendo com que muitas dessas empresas busquem soluções mais baratas, negligenciando algumas áreas.

É de grande importância e cuidado, desde o começo de um negócio, cuidar da imagem de uma empresa e discutir as maneiras usadas para gerar atratividade. Pela falta de capital, uma área bem comum que acaba sendo negligenciada é a área do marketing, o que dificulta a aquisição de clientes. Ter um bom portfólio de clientes, ter uma boa equipe de *marketing*, investir em tráfego pago e orgânico, ou ter um *software* que disponibiliza visibilidade e atratividade para uma empresa são algumas maneiras de alavancar a receita de um negócio. (SEBRAE, 2024)

Conforme o problema citado acima, que se refere a problemas de marketing, visibilidade e captação de clientes, torna-se possível criar algo acessível para aquelas micro e pequenas empresas (MPEs) que desejam divulgar seus produtos ou serviços, e terem a oportunidade de serem vistas por todas as pessoas. Além disso, o *software* também facilitará o consumidor, já que estará disponível as possíveis soluções para determinado problema, as diferenças de preço e as informações do fornecedor.

1.2 Objetivo Geral

Usando como base nossa motivação, pode-se afirmar que o objetivo geral é unir de fato o fornecedor ao cliente, provendo serviços ou produtos para o usuário da plataforma, visando resolver os problemas dele.

Hoje, sendo um microempreendedor, o maior desafio é ter seu empreendimento com boa visibilidade, com intuito de propagar e desenvolver seu negócio, criar relacionamentos e fidelidade com o cliente. Nossa plataforma fornecerá todo material necessário para que esse vínculo seja possível, com a nossa base de dados, podemos fazer com que um negócio com pouca visibilidade escale e consiga vender mais do que usando métodos convencionais já existentes na *internet*.

> "A *Internet* eliminou barreiras geográficas, permitindo que empresas de contabilidade alcancem clientes em qualquer canto do mundo. Uma presença digital otimizada pode atrair e engajar clientes de diferentes regiões, oferecendo soluções personalizadas para suas necessidades específicas". (DUARTE, 2024)

De forma geral, o projeto baseia-se em dar destaque aos microempreendedores, proporcionando a eles um sistema de fácil utilização e com ótima exposição da sua marca, produtos e serviços. Aos clientes, a plataforma terá o intuito de ser um tipo de compra centralizada apenas no que ele busca, de maneira interativa, objetiva e informativa. (DUARTE, 2024)

A partir do citado, a visão é solucionar esse problema do microempreendedor, fazendo com que ele consiga destacar e demonstrar o seu trabalho, e com isso, de forma rápida e tecnológica dar um passo a mais no seu negócio.

1.3 Objetivos Específicos

Há alguns objetivos que serão cumpridos durante o desenvolvimento do projeto. Como o foco é ajudar a vida do microempreendedor, será adicionada diversas funcionalidades para auxiliar a vida dele dentro da plataforma. São elas:

- Envio de avaliação por parte do usuário
- Canal direto com o fornecedor

É de suma importância garantir que o consumidor possa estabelecer contato direto com o fornecedor, proporcionando uma experiência de compra mais segura e com propriedade. Para facilitar essa comunicação, terá a implementação de uma função que direciona o usuário para um Chat exclusivo com o microempreendedor, oferecendo uma plataforma ágil para esclarecimento de dúvidas e suporte. Além disso, os usuários terão a opção de se comunicar via *e-mail*. Tudo será voltado para proporcionar uma melhor interação e facilitar o dia a dia tanto do fornecedor quanto do cliente final.

Além das melhorias que serão implementadas no sistema, abaixo estão listados os objetivos específicos do projeto:

- Analisar o comércio *online* e identificar oportunidades para os microempreendedores
- Integrar funcionalidades que além da exposição do produto/serviço, também faça um marketing sobre o microempreendedor, contando sua história e formas de contato.
- Impacto socioeconômico, visto que aumentará a visibilidade de comércios locais e pequenos empresários.
- Trabalhar em questões éticas e de segurança digital quanto as informações, envolvendo LGPD.
- Realização de testes iniciais com microempreendedores selecionados, coletando feedback e adquirindo experiências para um processo de melhoria contínua do sistema.

1.4 Metodologia

Nossa plataforma visará a melhoria da performance, eficiência do produto e serviço entregue para o cliente. Aumentando assim a confiança no fornecedor e a qualidade do seu serviço.

A pesquisa exploratória prosseguirá a partir de avaliações e análises aprofundadas na aplicação desse modelo de negócio, levando em consideração seja quaisquer obstáculos tecnológicos, ética e principalmente os cuidados relacionados à segurança digital das informações para com as pessoas envolvidas. Acrescentando a

isso, será analisado os potenciais benefícios para os usuários, que além de ter o que procura em suas mãos, poderá também ter um conhecimento maior e promover uma valorização do comércio local.

A falta de mão de obra qualificada, também é um desafio considerado na área do empreendedorismo, sendo difícil manejar tudo sozinho. Por razões de mudanças de mercado de trabalho, progresso tecnológico, mudanças na economia que resultam em demandas por novos profissionais etc. (BORELLI, 2023)

1.5 Organização do Texto

O capítulo 1 introduz o tema abordado, contextualizando a problematização e motivação que levaram à realização deste trabalho. São detalhados os objetivos gerais e específicos, que guiam o desenvolvimento do estudo, juntamente da metodologia utilizada, explicando as abordagens adotadas para alcançar os resultados esperados. Por fim, é exibido uma visão geral sobre a organização do texto.

No capítulo 2 será apresentado a revisão da literatura, no qual foram lidos artigos científicos, livros, resumos e trabalhos que fornecem contextualizações sobre os temas abordados e embasamento teórico, ajudando numa compreensão mais aprofundada sobre o tema.

O capítulo 3 visa demonstrar o detalhamento dos materiais e métodos durante a construção do estudo e do desenvolvimento do trabalho. São apresentadas as tecnologias presentes no projeto, destacando seus conceitos e as suas respectivas vantagens. Aborda também os requisitos funcionais e não funcionais que direcionaram o trabalho, juntamente das entidades necessárias. Por fim, diagrama de classe e caso de uso para demonstrar as relações presentes no sistema.

Para finalizar, temos um capítulo reservado para as referências bibliográficas consultadas durante o desenvolvimento do projeto e da pesquisa.

.

2 REVISÃO DA LITERATURA

A abordagem desse tópico tem objetivo de contextualizar e demonstrar a importância de diversos temas citados durante a realização do trabalho. Serão fundamentados a partir de citações de artigos científicos e livros lidos.

2.1 Tecnologia

Como Tavares (2023) pontua, a tecnologia é algo que vem desde os tempos antigos, e ao decorrer dos anos foi evoluindo cada vez mais, melhorando padrões de vida da sociedade e se tornando cada vez mais presente na vida do ser humano. Olhando para a própria casa, geladeira, micro-ondas, celular, televisão, tudo tem tecnologia envolvida.

No envolvimento entre a humanidade, ciência e tecnologia, isso permite abrir novos caminhos para um futuro promissor. Esse entrelaçamento permite o surgimento de novas tecnologias, pois os conhecimentos são diversificados em várias áreas como meio de resolução de problemas complexos, causando impactos globais. Exemplos de novas tecnologias que estão surgindo, já são e serão ainda mais fundamentais futuramente são inteligência artificial, biotecnologia, energias renováveis. (TAVARES, 2023)

> “A *internet* é a maior rede de comunicação já criada e está presente em praticamente todos os aspectos da nossa vida. A *internet* permitiu que as pessoas se comunicassem e partilhassem informação de forma mais rápida, permitiu que as pessoas se conectassem com pessoas de todo o mundo e que acessassem uma enorme quantidade de informação”. (TAVARES, 2023)

Conforme citado, a *internet* está presente em tudo na vida do ser humano, e nos tempos atuais vem tendo um aumento significativo nos comércios eletrônicos, muito em vista do cenário pandêmico que vivenciou o mundo todo entre 2020 e 2023. De acordo com Tomé (2021), o comércio eletrônico cresceu 40% em números de vendas no Brasil em comparação ao ano anterior, e registrou valor recorde de vendas. Com isso, em meio a um cenário que não era possível a saída de casa por motivos de saúde global, o *e-commerce* se tornou a principal alternativa para aquisição de itens de todos os tipos, para satisfazer a vontade do usuário.

Como aponta Tomé (2021), o *e-commerce* não vem ao mundo com objetivo de substituir comércios físicos, mas sim como uma ótima forma de impulsionar o *marketing* de microempreendedores, fornecendo a eles meios para a criação das suas lojas virtuais.

2.2 Benefícios da tecnologia

A tecnologia vem tendo sua importância desde a criação da roda até a revolução digital vivenciada no mundo atualmente. É de se destacar as diversas áreas que ela está presente e desempenhando papel fundamental no desenvolvimento da sociedade como um todo. Martins (2023) reforça o quanto a tecnologia afetou principalmente as conexões pessoais, aproximando as pessoas mesmo que distantes fisicamente.

Áreas como medicina, educação, trabalho e comunicação foram as principais afetadas com essa nova era, e que vem crescendo potencialmente a cada ano que passa.

Na área de comunicação, geograficamente as distâncias foram diminuídas e a velocidade que as informações são espalhadas globalmente se tornaram muito mais rápidas. É de um ponto positivo essa disseminação das informações, para debates sociais/políticos por exemplo, mas ao mesmo tempo é necessária uma cautela com as informações passadas, já que causam impactos gigantes no mundo todo.

Para a qualidade de vida, teve uma melhora significativa na área da saúde, proporcionando equipamentos avançados, uma precisão mais assertiva em diagnósticos, e descoberta de novos tratamentos e curas para algumas doenças.

No conceito da educação, um grande exemplo de como foi influenciado pela tecnologia, foi com o agravamento da COVID-19, no qual as escolas precisaram aderir ao modelo *home office* para aprendizado. Além disso, muitos cursos *online* estão se tornando cada vez mais frequente como forma de estudo. (MARTINS, 2023)

De acordo com Martins (2023), as formas como a tecnologia impacta na qualidade de vidas das pessoas são na comunicação (fortalecimento das conexões sociais), entretenimento (*streaming* de filmes, jogos *online*, música), mobilidade urbana (aplicativos de transporte e mapas *online* em tempo real), compras *online* (aquisição de produtos no conforto de casa), assistência à saúde (aplicativos de

telemedicina que permitem consultas *online*), gestão financeira (aplicativos de bancos e de gestão financeira pessoal).

2.3 Benefícios socioeconômicos que a tecnologia fornece

Nos tempos atuais, principalmente quando se trata de MEI (Microempreendedor Individual), é de tremenda importância a utilização de ferramentas e/ou serviços digitais a fim de otimizar tarefas, processos, e visar a diminuição do tempo gasto para realizar determinada atividade ou determinado problema.

O principal desafio enfrentado atualmente pelo MEI (Microempreendedor Individual) é a falta de investimento inicial, provocando a negligência de determinadas áreas que são importantes para uma empresa, fazendo com que ferramentas digitais suprem essas necessidades iniciais.

Whatsapp, *Instagram, Facebook, Youtube, X*, e *Google Analytics* são algumas das plataformas utilizadas por microempreendedores a fim de solucionarem algumas faltas que possuem inicialmente, como comunicação direta com o cliente, proposta de marketing e divulgação, produção de conteúdo, pesquisas relacionadas ao público-alvo e até mesmo a venda de determinado produto ou serviço. (SOUSA, 2019)

> “Quanto às ferramentas digitais, pode-se afirmar que estas trazem inúmeros benefícios indispensáveis ao MEI principalmente no que tange ao marketing, Hoje o âmbito *online* tornou-se um mundo de alternativas, oportunidades, perspectivas promissoras e a escolha pelo um canal digital adequado e acertado poderá trazer viabilidades de negócios almejados de maneira sustentável. Ou seja, estas ferramentas são essenciais para um melhoramento no trabalho do MEI, principalmente se o empreendedor individual pretende ser um MEI de sucesso.” (SOUSA, 2019)

Sem a utilização de ferramentas digitais disponíveis pela tecnologia, tornava-se necessário existir pessoas e/ou equipes que realizavam determinada ação. Atualmente, microempreendedores que buscam essas alternativas, conseguem uma alta redução de custo e um grande aumento de eficiência.

2.4 Lei Geral de Proteção de Dados (LGPD)

O surgimento dessa lei tem o objetivo de estabelecer diretrizes sobre como as empresas devem garantir a proteção e privacidade dos dados de cada pessoa, assegurando a segurança das informações recebidas e enviadas. Seja a pessoa física ou jurídica, a lei trabalha em cima dos dados físicos e digitais. (GOV 1, 2024)

Como dito por Rapôso (2019), esse tipo de proteção está se tornando cada vez mais importante nos mundos atuais, dado o aumento de vazamento de dados das empresas ou até do próprio estado, realizados por hackers, comprometendo assim, a integridade das organizações.

Alguns dos princípios sobre a LGPD estão descritos abaixo com suas respectivas descrições:

- Consentimento: para utilização dos dados do indivíduo, requer a autorização explícita dele, exceto em algumas exceções apontadas na lei.
- Finalidade: o uso dos dados deve ser para propósito real e informado ao usuário, proibindo assim, o uso para qualquer finalidade diferente da determinada.
- Necessidade: se limita a usar apenas os dados necessários para tal operação, sem uso excessivo de dados que não serão úteis.
- Segurança: Medidas adotadas para proteção dos dados contra ataques, acessos não autorizados, perda, alteração ou destruição deles.
- Transparência: deixar o usuário ciente da utilização e tratamento dos seus dados.
- Não discriminatório: os dados do usuário não podem ser usados para fins de abusivos ou discriminatórios. (GOV 2, 2024)

2.5 Relação Cliente X Fornecedor

Com o avanço da tecnologia, as trocas entre cliente e fornecedor se tornam mais simples e mais rápidas, tornando as relações comerciais cada vez mais competitivas, fazendo com que empresas necessitem de uma gestão de qualidade cada vez melhor e estejam aptas a criarem bons relacionamentos.

De acordo com Christopher (2016), a GCS (Gestão da Cadeia de Suprimentos) é a gestão da relação de uma organização que envolve tanto fornecedores quanto clientes, que tem como objetivo final entregar valor ao cliente com o menor custo possível. Por apresentar diversas áreas em comum com a GQ (Gestão de Qualidade), denomina-se GQCS (Gestão da Qualidade na Cadeia de Suprimentos) a ligação entre ambas. (FOSTER, 2011)

Dentro da GQCS (Gestão da Qualidade na Cadeia de Suprimentos), destaca-se a área de gestão de relacionamento com fornecedores. (MONCZKA, 2009)

> "A área de gerenciamento do relacionamento com o fornecedor é importante porque trata da base de fornecimento da empresa, devendo esta área ser melhor desenvolvida ou mantida para o sucesso no longo prazo, pois a participação do fornecedor está correlacionada diretamente a melhora do desempenho organizacional". (ZENG, 2013)

Uma boa gestão entre o relacionamento cliente fornecedor trás diversas características positivas para uma empresa, destaca-se duas delas: qualidade nos produtos ou serviços ofertados, já que essas duas empresas ou pessoas dedicam-se para resolver o mesmo problema ou situação, e uma alta produtividade, visto que produzem mais utilizando menos tempo e recurso. (SIMCHI-LEVI, 2003)

Contudo, um bom comportamento colaborativo, interessado na construção de vínculos de longo prazo e na criação de valor como parceiros, torna-se necessário para pessoas ou empresas que queiram alavancar seus negócios, já que este bom relacionamento impacta diretamente na preservação de um crescimento contínuo.

2.6 Microempreendedores

Com o advento da globalização e da divisão mundial do trabalho é notório que os países do sul global estão em processo de subjugação perante os países do norte global, ou seja, estão operando de forma a favorecer os países dominantes com tecnologia e mão de obra barata enquanto não conseguem alavancar a sua situação econômica local ou desenvolver outras atividades laborais. (OLIVEIRA, 2014)

Nos últimos anos, a população economicamente ativa vem sofrendo muitas dificuldades para alavancar sua carreira profissional e se manter no mercado de trabalho. Com o aumento da concorrência e dos requisitos para conseguir um

emprego, pessoas buscam maneiras para alavancar a sua renda, como produção familiar, trabalho por conta própria, venda de produtos, ou até mesmo fazendo a abertura de pequenas empresas prestadoras de serviços. (FORTE, 2014)

Microempreendedor individual (MEI), é a pessoa que por motivos citados acima define por ocupar uma função de trabalho desassociada de empresas alheias, ou seja, decide abrir um negócio próprio, ainda que pequeno, com diversas dificuldades e sem as formalidades de uma empresa formada assumindo o risco do que está por vir.

> “Como é evidenciado extensamente na literatura, esse tipo de empreendedorismo tende a ser frequente em situações de baixo desenvolvimento econômico.” (MORAIS, 2022)

Pessoas que assumem o risco de estarem mais propicias a informalidade tem mais chances de ascenderem socialmente, pois o Microempreendedor Individual (MEI), apesar de ainda estarem na informalidade estão no caminho do desenvolvimento do seu negócio e tendem a ter maior possibilidade de adquirirem ascensão social.

2.7 *E-commerces*

Conforme Mendonça (2016) O *E-commerce* foi iniciado na década de 70 onde aconteceram as primeiras transações eletrônicas utilizando as formas do EDI (*Electronic Data Interchange*) que tem como base transferências eletrônicas de documentos e EFT (*Electronic Funds Transfer*) que sua principal função são as transferências eletrônicas de fundos, ambas são utilizadas pelo setor bancário.

Com a chegada da internet, o *E-commerce* se fortaleceu, facilitando todo tipo de compra e venda. O *e-commerce* permite que os consumidores realizem compras de qualquer lugar e horário, o crescimento desse mecanismo aumenta exponencialmente desde o seu início, podendo até mesmo ultrapassar a venda convencional que temos hoje em dia.

O mais incrível presente nos *e-commerces* é que pequenos comerciantes conseguem atender uma demanda alta de clientes, mesmo sem um estabelecimento compatível, fazendo com que assim, mais pessoas se motivem para criar seu negócio e ir atrás de atender a expectativa do cliente.

Há diversos tipos de *e-commerces*, voltados para os diversos setores, sendo algum deles:

- B2B (*Business to Business*): É a relação entre empresas, onde as transações ocorrem de forma privada e partilhada entre elas.
- B2C (*Business to Consumer*): Esse tipo de *e-commerce* é o mais conhecido, onde envolve o fabricante e as distribuidoras para o consumidor final.
- B2E (*Business to Employee*): Esse *e-commerce* visa a empresa criar plataformas para oferecer produtos para o funcionário com menores preços. (MENDONÇA, 2016)

2.8 *Business-to-Business* (B2B)

Em concordância com Marques (2024), o comércio eletrônico B2B, ou comércio eletrônico *business-to-business*, é uma plataforma *online* para transações comerciais entre empresas. Diferentemente do comércio eletrônico B2C (*business-to-consumer*), onde as transações ocorrem entre uma empresa e o consumidor final, o B2B tem foco em atender às necessidades de uma empresa, fornecendo produtos ou serviços para outras empresas.

Neste caso, o cliente normalmente é uma empresa, revendedor, distribuidor ou instituição. O comércio eletrônico B2B facilita a compra e venda de produtos ou serviços em grandes quantidades, muitas vezes em quantidades maiores do que as transações B2C. As transações podem incluir a compra de matérias-primas, componentes, equipamentos, *software*, serviços comerciais, etc.

As plataformas de *e-commerce* B2B costumam oferecer funcionalidades específicas para atender às necessidades desse mercado, como catálogos personalizados, precificação no atacado, cotações em massa, integração com sistemas de gestão empresarial (ERP), ferramentas de automação de vendas, etc. O principal objetivo é simplificar e agilizar o processo de compra e venda entre empresas, proporcionando eficiência e comodidade para ambas as partes (MARQUES, 2024).

2.9 *Business-to-Consumer* (B2C)

Como dito por Felipin (2024), o comércio eletrônico B2C, ou comércio eletrônico *business-to-consume*r, é um modelo de negócios *online* que realiza transações

comerciais entre empresas e consumidores finais. Nesse caso, cliente é a pessoa física que adquire um produto ou serviço para uso ou consumo próprio.

No comércio eletrônico B2C, as empresas normalmente operam lojas virtuais onde os consumidores podem navegar, pesquisar produtos, comparar preços, comprar e receber os produtos diretamente em sua porta. Essas lojas *online* oferecem uma ampla gama de produtos, desde eletrônicos, roupas e produtos de beleza até serviços como assinaturas de streaming e reservas de viagens.

Os volumes de transações no comércio eletrônico B2C são normalmente menores do que os do B2B, e os preços são frequentemente definidos com base nas preferências individuais dos consumidores e nas capacidades financeiras. As estratégias de marketing digital desempenham um papel vital no *e-commerce* B2C, com foco na atração de clientes, na melhoria da experiência do usuário e no incentivo à fidelidade à marca.

As plataformas de comércio eletrônico B2C fornecem carrinhos de compras, sistemas de pagamento seguros, análises de produtos, recomendações personalizadas e suporte ao cliente para garantir que os consumidores tenham uma experiência de compra *online* satisfatória e conveniente. O objetivo é proporcionar uma jornada de compra eficiente e agradável, impulsionar as vendas e construir relacionamentos duradouros com os clientes. (FELIPIN, 2022)

2.10 *Business-to-Employee* (B2E)

A classe de negócios B2E, *business-to-employee* trata das interações dentro da organização, a empresa, seus funcionários e entre eles. Com a tecnologia *web*, os especialistas podem acessar práticas e conhecimentos propagados por meio da internet e, consequentemente, novos conceitos de produtos e serviços podem se mover na empresa. A *Internet* proporciona a maior facilidade para os profissionais obterem informações relevantes e a informação necessária para sua ação. Este modelo afeta significativamente técnicas de aprendizado, na manutenção do conhecimento empresarial, aumentando a eficiência da equipe de tomada de decisão e vendas. A classe B2E pode incluir, por exemplo, *e-mail*, colaboração *on-line*, videoconferência e instrução sobre distância e portais da empresa.

O B2E é uma estratégia de negócios inventiva com o objetivo de aumentar a eficiência operacional e a satisfação dos funcionários. As empresas podem aumentar

a lealdade, a moral e até mesmo as vendas ao fornecer produtos e serviços diretamente aos funcionários. Assim, B2E é uma estratégia valiosa que todas as empresas devem considerar. (BERALDI, 2024)

2.11 Controle de qualidade nos serviços entregues

A qualidade de um serviço entregue pode ser medida em diversos conceitos da área, como por exemplo atender as necessidades do usuário (sejam elas presentes ou futuras), se adequar ao ponto de vista esperado pelo cliente, ou como também a junção total de *marketing*, engenharia, fabricação e manutenção de um produto que deverá atender às expectativas do cliente.

No conceito geral, controle de qualidade representa um produto ou serviço que atende perfeitamente de forma confiável, acessível, segura e no tempo certo às necessidades dos clientes. Mas, é importante para a empresa que quando construído e ofertado, seja na primeira vez de maneira certa, sem erro, sem atraso e sem perdas. (FAGGIÃO, 2024)

2.12 Sistemas Distribuídos

De acordo com COULOURIS (2007), um sistema distribuído consiste em componentes localizados em diferentes computadores conectados em rede que se comunicam e coordenam suas atividades exclusivamente por meio da troca de mensagens.

As conexões de rede entre computadores por si só não caracterizam sistemas distribuídos, quando falamos em hardware, estamos falando de máquinas autônomas, logo quando falamos em software, estamos falando de programas que os usuários pensam que estão instalados no dispositivo que estão usando. (LOPER, 2019)

A principal razão para criar e utilizar sistemas distribuídos é a necessidade de compartilhar recursos. O conceito de “recurso” é bastante amplo e inclui qualquer coisa que possa ser efetivamente compartilhada entre computadores em rede. Isso varia de hardware, como discos rígidos e impressoras, a elementos de software, como arquivos, bancos de dados e diferentes tipos de dados. Por exemplo, um fluxo de vídeo de uma câmera digital ou áudio de uma chamada de celular. (LOPER, 2019)

Um grande exemplo disso é a própria internet, que é um grande sistema distribuído. Definido como uma grande rede de computadores, roteadores e outros componentes que podem transferir dados de um local para outro. Ela fornece seus usuários serviços como WWW, e-mail e FTP, que podem ser usados de qualquer lugar e quando quiser.

Num exemplo citado por COULOURIS (2007), os jogos online *multiplayer*, conhecidos como MMOGs, criam uma experiência envolvente onde milhares de jogadores interagem em um mundo virtual contínuo pela Internet. Esses mundos estão cada vez mais sofisticados, com arenas de jogo bastante complexas. A criação desses jogos é um grande desafio para a tecnologia de sistemas distribuídos, principalmente pela necessidade de respostas rápidas para garantir uma boa experiência aos jogadores.

Além disso, manter uma visão consistente do mundo compartilhado e transmitir eventos em tempo real para tantos usuários são grandes desafios. Isso ilustra bem os problemas que os desenvolvedores de sistemas distribuídos enfrentam hoje.

2.12.1 Modelo Cliente-Servidor

No modelo cliente-servidor, há uma divisão clara de responsabilidades: o cliente faz as requisições, e o servidor as processa. Esse é um dos modelos mais comuns, especialmente em aplicações web. Contudo, como apontado por Tanenbaum (2007), pode ser difícil, em alguns casos, definir quem é o cliente e quem é o servidor, já que ambos podem desempenhar funções diferentes dependendo do contexto. Por exemplo, em sistemas de arquitetura em camadas, como os que acessam bancos de dados, geralmente temos três níveis. O primeiro é a interface do usuário, que permite a interação com o sistema. O segundo é o processamento, onde o aplicativo propriamente dito opera. E, por fim, temos a camada de dados, responsável por gerenciar as informações e operações realizadas.

2.12.2 Modelo *Peer-to-Peer* (P2P)

Diferente do modelo cliente-servidor, o modelo ponto a ponto (P2P) não tem uma hierarquia fixa. Aqui, todos os nós da rede, chamados de peers, funcionam tanto como clientes quanto como servidores. Em vez de depender de uma única entidade

central, eles compartilham e consomem recursos entre si. Um exemplo conhecido dessa arquitetura é o BitTorrent, onde os usuários compartilham partes de arquivos enquanto baixam outras de diferentes nós da rede. Apesar de permitir maior descentralização e escalabilidade, o modelo P2P enfrenta desafios como segurança, consistência dos dados e confiança entre os nós. (COULOURIS, 2007)

2.12.3 Modelo de Arquitetura Orientada a Serviços (SOA)

A arquitetura orientada a serviços (SOA) organiza a funcionalidade de um sistema em serviços independentes que podem ser acessados e utilizados por outras partes do sistema. Isso facilita a integração de diferentes aplicações, já que cada serviço é modular e pode ser reutilizado em diferentes contextos. A SOA é amplamente adotada em sistemas corporativos, principalmente em cenários que exigem interoperabilidade entre diversas plataformas. No entanto, à medida que o número de serviços cresce, surgem questões relacionadas à coordenação, governança e desempenho. (COULOURIS, 2007)

2.12.4 Modelo de Computação em Nuvem

A computação em nuvem é um modelo de sistemas distribuídos que permite que recursos como armazenamento e processamento sejam fornecidos como serviços através da internet. Esse modelo elimina a necessidade de uma infraestrutura física própria, já que os recursos podem ser alocados conforme a demanda. Empresas como AWS, Google Cloud e Microsoft Azure popularizaram a computação em nuvem, oferecendo soluções escaláveis e flexíveis, onde os usuários pagam apenas pelo que utilizam. Apesar de suas vantagens, a computação em nuvem apresenta desafios como a garantia de segurança dos dados, a minimização da latência e a alta disponibilidade dos serviços. (HENNESSY, 2018)

2.13 Algoritmos

De acordo com Gillespie (2018), algoritmos estão presentes em todo lugar, se você usa internet, com certeza já usufrui de algum algoritmo específico. Algoritmo nada mais é do que uma sequência de passos para entregar a solução de algum

problema específico. Um dos tipos predominantes de algoritmos no dia a dia são os implementados em mecanismos de busca, plataforma de mídia social, sistema de recomendações e base de dados, no qual ele tem como principal característica selecionar qual informação é a mais adequada para apresentar para os usuários.

Os algoritmos não necessariamente são um *software* completo, no conceito mais amplo de algoritmo, eles podem ser classificados como procedimentos codificados que, com base em cálculos específicos, transformam dados em informação desejada. Os algoritmos denominados de relevância pública, estão equivalentes aos mesmos procedimentos matemáticos.

Abaixo estão alguns atributos sobre os algoritmos de relevância publica:

- Padrões de inclusão: Servem para selecionar o que vai ser apresentado para o usuário, fazendo a assertividade por index e a exclusão das demais informações desnecessárias.
- Ciclos de antecipação: Ocorre um teste para conseguir conhecer e prever completamente a fundo os seus usuários.
- Avaliação de relevância: São parâmetros propostos para o algoritmo para determinar quais informações são relevantes, ocultando-os dos usuários, retornando apenas informações de um conhecimento considerado apropriado e legitimo.
- Promessa da objetividade algorítmica: Promete a imparcialidade do algoritmo, garantido seu caráter técnico, mesmo que haja controvérsias.
- Entrelaçamento com a prática: Interagir com o usuário e como eles reconfiguram o algoritmo para se adequar a eles.
- A produção de públicos calculados: Molda uma noção para os algoritmos públicos, e seleciona quem está mais bem posicionado para receber e se beneficiar do conhecimento informado. (GILLESPIE, 2018)

2.14 Inteligência Artificial (IA)

A inteligência artificial é uma área da ciência da computação que tem o objetivo de criar sistemas onde as máquinas executam tarefas que antes eram realizadas por pessoas, substituindo a inteligência humana pela inteligência do sistema. De maneira geral, é criar máquinas que possam agir como seres humanos. (IBM 4, 2024)

> “A IA faz parte da nossa vida cotidiana. Acessamos sistemas inteligentes para programar o itinerário com o Waze, pesquisar no Google e receber da Netflix e do Spotify recomendações de filmes e músicas. A Amazon captura nossas preferências no fluxo de dados que coleta a partir das nossas interações com a plataforma. A Siri, da Apple, e a Alexa, da Amazon, são assistentes pessoais digitais inteligentes que nos ajudam a localizar informações úteis com acesso por meio de voz”. (KAUFMAN, 2022)

A organização das máquinas de inteligência artificial pode ser baseada em quatro etapas, sendo elas:

- Máquinas reativas: é um exemplo de máquina que não grava dados, portanto não pode aprender consigo mesma. Ela se baseia em respostas pré-programadas.
- Memória limitada: é a definição de sistemas que utilizam de sua memória para melhoria contínua, como por exemplo máquinas de aprendizado profundo (deep learning).
- Teoria da mente: algo ainda não existente, mas que tem o objetivo de emular uma mente humana, incluindo tomadas de decisões, emoções e reação a uma determinada situação.
- Autoconhecimento: é uma teoria onde a inteligência artificial tem conhecimento da sua própria existência e determina seus próprios sentimentos como um humano. (GOOGLE CLOUD 2, 2024)

2.15 *Machine Learning* (ML)

A *machine learning* é uma subárea da inteligência artificial e ela tem o objetivo de ensinar as máquinas a partir dos dados que são consumidos, melhorando gradativamente seu desempenho e sem necessitar de programação de tarefas para essa realização. Apesar da sua independência em aprendizado, eles precisam regularmente serem atualizados para que fiquem de acordo com determinada situação e coerente com os dados mais recentes. (GOOGLE CLOUD 1, 2024)

> ”O *machine learning* e a IA são frequentemente abordados juntos, e os termos às vezes são usados de forma intercambiável, mas não significam a mesma coisa. Uma distinção importante é que, embora todo machine learning seja IA, nem toda IA é machine learning.” (ORACLE 3, 2024)

Existem dois tipos de algoritmos para ML, que a diferença está justamente na maneira como cada um lida e aprende com seus dados. São eles:

- ML Supervisionado: o algoritmo tem uma saída pré-definida e consiste em um humano ensinar como essa máquina deve se comportar, alimentando-a com dados rotulados e determinando seus possíveis resultados.
- ML Não supervisionado: esse modelo consiste no aprendizado sem que um humano forneça qualquer tipo de direção e orientação. A máquina por si só aprende a identificar padrões de dados ocultos e suas saídas não são especificadas ou definidas. (MONARD, 2024)

Um outro modelo existente, mais comum em projetos de robótica e jogos, é o aprendizado por reforço, no qual o sistema aprende através da tratativa de tentativa e erro. O sistema apenas progride quando começa obter uma sequência positiva de resultados e assim adotando um padrão de respostas.

Os algoritmos das máquinas de aprendizado estão presentes em redes neurais (simulação de um cérebro humano através de processamentos), regressão linear (previsão de valores numéricos), regressão logística (previsão de respostas simples, como sim ou não), agrupamento (agrupamento de dados para análise de suas diferenças), árvores de decisão (decisão baseada nas ramificações) e florestas aleatórias (prevê um valor com base nas árvores de decisões). (IBM 2, 2024)

2.16 *Deep Learning* (DL)

O *deep learning* é um subconjunto do machine learning combinado com redes neurais, que são configuradas através de um treinamento com grande quantidade de dados de alta complexidade. Como a alimentação desse tipo de máquina é bem diversificado, vindo de inúmeras fontes diferentes, o aprendizado se faz de forma automática, seguindo os mesmos conceitos de aprendizado de máquina, onde se aprende sozinha sem interferência de humano. (REDHAT, 2024)

> “O *deep learning* impulsiona muitos aplicativos e serviços de inteligência artificial (IA) que melhoram a automação, realizando tarefas analíticas e físicas sem intervenção humana. A tecnologia de *deep learning* está por trás de produtos e serviços cotidianos (como

> assistentes digitais, controles remotos de TV ativados por voz e detecção de fraude de cartão de crédito), bem como tecnologias emergentes (como carros autônomos).” (IBM 3, 2024)

A escolha do *deep learning* se dá pela grande capacidade de processamento de dados de alta complexidade, como por exemplo imagens, textos e vídeos, descobrindo dados que não são mostrados através de métodos tradicionais. As redes neurais profundas automatizam a rotulagem dos dados, de forma que ele mantenha a alta precisão, eficiência e ainda economize recursos. Outra grande vantagem de seu uso é sua facilidade em se adaptar conforme outros tipos de dados são informados, sendo versátil para novas situações, o que ajuda bastante na segurança cibernética. Essa combinação de poderes da eficiência, precisão e adaptabilidade torna essa ferramenta extremamente útil em vários setores. (ORACLE 4, 2024)

2.17 *Chatbot*

O objetivo de um *chatbot* é espelhar interações humanas, sejam elas por voz ou texto, fazendo com que as pessoas se comuniquem de forma digital como se fosse uma interação humano-humano. As suas variações se dão em diversos modos, seja desde um pequeno *chatbot* de respostas curtas e esclarecer dúvidas até sistemas de aprendizado que vão evoluindo e se desenvolvendo cada vez mais através de informações fornecidas pelos usuários. Existem dois principais tipos de *chatbots*, no qual eles processam diversos tipos de dados para fornecer respostas de acordo com a solicitação do usuário.

- Chatbot orientado a tarefas (declarativos): São destinados para funções de suporte e serviço, algo que não envolva muita informação a ser processada, pois fornecem respostas automatizadas em formato de conversa para o usuário e seguem uma estrutura específica. Sua operação é com base de processamento de linguagem natural (NLP) e com aprendizado de máquina.
- Chatbot orientado por dados e preditivos (conversacionais): Esse modelo de *chatbot* é voltado para algo mais específico e que exige um maior processamento de dados, pois são mais interativos e personalizados. Eles utilizam de compreensão da linguagem natural (NLU), processamento de linguagem natural (NLP) e aprendizado de máquina (ML). Com essas funções,

> permite a ele se adaptar as comodidades do usuário ao longo do tempo, identificando suas preferências e necessidades. Exemplos desse tipo de *chatbot* é a própria Siri da Apple, Alexa da Amazon. (ORACLE 1, 2024)

2.18 *Natural Language Processing* (NLP)

O processamento de linguagem natural é uma área que foca na interação da linguagem de computadores (através de inteligência artificial) com a linguagem do ser humano, potencializando máquinas e capacitando-as para uma melhor interpretação e geração de linguagem natural. (VIEIRA, 2024)

A combinação dessas linguagens possibilita o computador compreender o que está sendo processado, como por exemplo conversão de dados em formato de texto para voz. Abaixo algumas tarefas que são incluídas a partir desse modelo:

- Reconhecimento de voz: converte o que foi dito para texto. Devido a individualidade de cada pessoa, seja pelo sotaque, maneira de falar palavras gramaticalmente erradas, entonação, torna o processo de entendimento mais complicado.
- Marcação de trecho em voz: determina o significado e sentido de uma palavra específica de acordo com seu contexto, como por exemplo, em uma situação uma palavra pode ser um verbo, mas em outra pode ser um substantivo.
- *Named Entity Recognition*: tem a capacidade de atribuir uma entidade única a certas coisas e diferenciar elas, como por exemplo “San Diego” é uma cidade e “João” é um nome.
- Análise de sentimento: identifica o sentimento que a pessoa demonstra através da fala, algum tipo de emoção, sarcasmo. Esse tipo de processo já é bem frequente e presente em muitas aplicações atuais.
- Tradução automática: utiliza-se da metodologia de entender o contexto da frase que está sendo redigida em um determinado idioma, para poder traduzir em um outro e passar a mesma ideia e emoção. Apenas a tradução livre de cada palavra individualmente, geraria um texto desconexo e sem sentimento do que está sendo dito.

- Detectar *spam*: de maneira geral, esse tópico é considerado como o um dos mais funcionais dentro do NLP. O bloqueio de e-mails é feito através da identificação das palavras contidas nele, se possui erros gramaticais, excesso de ofertas envolvendo temas financeiros, ameaças. Resultando em um e-mail de spam ou pishing.
- Agentes virtuais: é um tipo de assistente que vai aprendendo conforme o tempo, quanto mais convivência com o ser humano, vai se adaptando aos seus gostos e costumes. Reconhecem padrões de linguagens para poder responder com maior precisão possível ao usuário. Exemplos disso: Siri (Apple) e Alexa (Amazon).

Os seus principais componentes se baseiam na divisão do texto em frases menores, o que é chamado de tokenização. A análise do texto se dá em três partes, morfológica (classificação gramatical de cada palavra), sintática (estruturar as sentenças para entender a relação das palavras) e semântica (significado das palavras dado o contexto). Por fim, a modelagem dos tópicos, que seria juntar os textos prontos e com sentido em textos maiores. (IBM 1, 2024)

2.19 *Natural Language Understanding* (NLU)

A compreensão da linguagem natural é uma subárea do processamento de linguagem natural, e seu foco maior está no entendimento da linguagem humana por parte das máquinas, com objetivo de fazer com que as máquinas entendam os contextos da linguagem natural. (SANCHES, 2024)

> "Quanto uma pessoa vê ou ouve uma sentença, ela faz completo uso de seu conhecimento e inteligência para entender. Isso não inclui apenas gramática, mas também o seu conhecimento sobre palavras, contexto da sentença, e mais importante, sua compreensão do assunto. Para modelar esse processo de entendimento de linguagem em um computador, nós precisamos de um programa que combine gramática, semântica, e raciocínio em um jeito íntimo, concentrando em suas interações." (WINOGRAD, 1972)

As suas principais características são parecidas com a NLP, mas de uma forma mais eficaz e melhorada.

- Análise de sintática: estruturar as frases em componentes gramaticais.

- Análise de semântica: responsável pela interpretação do texto dado um determinado contexto.
- Reconhecimento de entidades nomeadas (NER): verificar se é uma pessoa, local ou organização.
- Análise de sentimento: através do texto, verifica a sensação que está sendo passada.
- Resposta a perguntas: se baseando no contexto da frase, ele obtém a informação necessária para responder uma pergunta específica feita pelo usuário.
- Reconhecimento da Intenção: entender a intenção real atrás de uma pergunta.

Algumas arquiteturas como BERT e GPT estão sendo essenciais para o uso da NLU e o aprendizado contínuo, com diversos dados sendo processados. Como trabalham com grande escala de dados, cada vez mais vai se tornando mais precisa na captura e identificação de diferenças entre as linguagens. (BENDER, 2024)

3 MATERIAIS E MÉTODOS

Para esse tópico será abordado todas os tipos de materiais, ferramentas e métodos utilizados para desenvolvimento do projeto.

3.1 Linguagens utilizadas

Na criação de um *e-commerce*, é necessário pensar em estratégias de escalabilidade visando o aumento gradual de usuários na plataforma. Tendo essa base, de fato é preciso pensar em tecnologias que suportam essa escalabilidade sem haver problemas. No nosso *e-commerce* invertido, usaremos 5 principais *stacks* para suprir essa demanda de usuários que pode haver em nossa plataforma, sendo elas:

- Angular: é um *framework* onde o core dele é o typescript (um *super-set* do *javascript* que contêm tipagem e outras funções). Nele será feito todo o desenvolvimento da visibilidade da plataforma, separando cada parte do processo em componentes reativos usando *Typescript*, HTML e CSS.

- AWS: Os serviços da Amazon serão utilizados para trazer performance para a aplicação, através de sistemas de filas para desenvolver processos assíncronos, sistemas para envio de *e-mails*, como por exemplo o SES para notificar o fornecedor/usuário.
- GCP - Google Cloud: será utilizado para armazenar todo o core do site, onde ele poderá ficar no ar 24/7 para ser acessado a qualquer momento.
- Java - *Springboot*: Java é uma linguagem fortemente tipada, onde se pode utilizar recursos de Orientação a Objetos (POO) para um código mais limpo e manutenível. *Springboot* é o *framework web* que será usado para construir as requisições do *front-end* para o *back-end*. Tendo uma segurança adequada usando *SpringSecurity*, validação de entidades e criptografia de dados.
- Node - Nestjs: Node é um conceito mais abstrato onde utiliza *JavaScript* no lado do servidor. Sua aplicação se dá constantemente para criação de micro serviços, *Streams*. O node combinando com o *framework* Nest, será onde organizaremos os projetos funções lambdas, usando *serverless*, de forma modularizada.

3.1.1 Ferramentas e suas vantagens

Neste tópico será abordado os principais conceitos e vantagens sobre cada linguagem e tecnologia utilizadas no desenvolvimento do projeto. Cada tópico fornecerá um breve resumo sobre cada um presente no *software.*

3.1.1.1 Angular

Desenvolvido pelo Google, e de código aberto, o Angular, é um *framework* desenvolvido para criação de aplicativos dinâmicos e interativos, possui uma abordagem orientada a componentes para a construção de interfaces, e pode utilizar *TypeScript*, HTML e CSS dentro de cada componente reativo.

> “AngularJS é um *framework JavaScript* que simplifica o desenvolvimento de aplicações *web* dinâmicas robustas, viabilizando a implementação do conceituado modelo MVC (*Model-View-Controller*). Ele apresenta características satisfatórias tais como: produtividade, desempenho, fácil customização, entre outras.” (PEREIRA, 2014)

Angular é um *framework javascript* que possibilita criar e manipular componentes reativos, conseguindo isolar a parte de HTML, CSS, e Regra de Negócio da própria tela.

3.1.1.2 Amazon Web Services (AWS)

AWS ou *Amazon Web Services* é uma plataforma de nuvem que alimenta infraestruturas e aplicações desenvolvido pela *Amazon*. A AWS fornece serviços tecnológicos sob demanda, os quais podem ser utilizados para criar e gerir virtualmente qualquer aplicação. Ela oferece diversos serviços, como: infraestrutura tecnológica, armazenamento e base de dados, machine learnings, entre outros. (AWS 1, 2024)

> "*Amazon Web Services* (AWS) é a principal oferta de arquitetura do tipo cloud *computing* da atualidade. Esta arquitetura permite às empresas o acesso a serviços de infraestrutura na forma *on demand*. A ideia é que esta forma de aquisição de serviços de infraestrutura de TI, conhecida como serviços de infraestrutura de nuvem, reduza custos, minimize os riscos do negócio e maximize as oportunidades. Do ponto de vista teórico, o serviço tradicional de *DATACENTER*, quando comparado ao serviço de nuvem, é pouco eficaz, considerando que o uso e a capacidade dos recursos não podem ser otimizados como no modelo baseado em nuvem." (VERAS, 2013)

Usando node na criação de micro serviços, conseguimos implementar alguns serviços externos nele, como o AWS. Dentro do AWS, existem várias ferramentas para usarmos e dar corpo para nossa aplicação, com o conceito de funções lambdas.

3.1.1.3 CSS

CSS, *Cascading Style Sheets* (Folhas de Estilo em Cascata), é uma linguagem de estilo utilizada para gerenciar a apresentação visual de documentos HTML (Linguagem de Marcação de Hipertexto). Ela determina a aparência dos elementos HTML em uma página da *web*, incluindo características como cores, tipografia, espaçamento, dimensões e posicionamento dos elementos.

O propósito do CSS é separar o conteúdo (HTML) da apresentação (estilo). Essa abordagem possibilita criar um documento HTML com a estrutura e o conteúdo da página e, posteriormente, aplicar um arquivo CSS para definir como esse conteúdo será estilizado e formatado. Essa divisão promove um desenvolvimento *web* mais eficiente, flexível e de fácil manutenção, pois permite ajustar a aparência de um site sem precisar modificar o conteúdo HTML original. (DIO, 2024)

3.1.1.4 DevOps

DevOps é uma forma de estruturar o desenvolvimento de soluções. Também pode ser definido como um conjunto de ferramentas e práticas adotadas para melhorar o desempenho de uma equipe e distribuir a responsabilidade, fazendo com que ela seja compartilhada. Em uma equipe de DevOps, as equipes de desenvolvimento e operações não ficam mais fragmentadas, trabalham em conjunto desde o desenvolvimento e testes até a implementação e operações. Quando necessário, equipes de segurança também podem adotar o conceito, e o nome passa a ser DevSecOps. (ATLASSIAN 1, 2024)

> “Um composto de Dev (desenvolvimento) e Ops (operações), o DevOps é a união de pessoas, processos e tecnologias para fornecer continuamente valor aos clientes.” (MICROSOFT, 2024)

O uso do DevOps será fundamental no processo de automatização de processos, além de gerar uma melhor integração entre a equipe, resultando em entregas rápidas e mais efetivas de soluções para o sistema.

3.1.1.5 HTML

HTML (Linguagem de Marcação de Hipertexto) é a base fundamental da *internet*, usado para definir a estrutura de um conteúdo *WEB*. Outras tecnologias auxiliares são empregadas para *design* visual e funcionalidades interativas em páginas da *web*.

O termo "hipertexto" se refere aos *links* que conectam páginas da *web*, tanto dentro de um mesmo site quanto entre diferentes sites. Esses *links* são vitais para a

web, permitindo que os usuários naveguem e descubram conteúdos de forma eficiente, ao compartilhar e vincular conteúdos na *internet*.

Um elemento HTML é distinguido de outros textos em um documento por meio de tags, que consistem no nome do elemento entre os caracteres "<" e ">". Esses elementos podem ser escritos em letras maiúsculas, minúsculas ou em uma combinação de ambas, pois a diferenciação entre maiúsculas e minúsculas não afeta o reconhecimento das *tags*. (MOZILLA, 2024)

Para nossa faixa da plataforma, usaremos os principais recursos para desenvolver ela. Com a marcação de texto (HTML) conseguimos arquitetar nossa estrutura do site e juntamente com as folhas de estilo (CSS) conseguimos personalizar fonte, cor ou qualquer estilização do nosso site.

3.1.1.6 Java

A história do Java remonta a 1995, quando foi criado nos laboratórios da *Sun Microsystems*, após extensa pesquisa científica e tecnológica. Esta plataforma oferece um ambiente completo para o desenvolvimento e execução de programas, composta por:

- Uma linguagem de programação de alto nível orientada a objetos;
- A Máquina Virtual Java (*Java Virtual Machine* ou JVM), que assegura a independência de plataforma, permitindo que o código seja executado na JVM e seja portado para diferentes plataformas, como Windows ou Linux;
- O *Java Runtime Environment* ou JRE, que inclui a máquina virtual e recursos essenciais para a execução de aplicações Java; e
- O *Java Development Kit* ou JDK, um conjunto de utilitários que suportam o desenvolvimento de aplicações.

No Java, os programas são escritos em arquivos com extensão .java, que posteriormente são compilados para arquivos “*.class*”. Esses arquivos contêm os *bytecodes*, que serão executados na máquina virtual.

A JVM está disponível em diversos sistemas operacionais, permitindo a execução da mesma aplicação Java em diferentes ambientes, como Windows, macOS, Linux e Solaris. Essa capacidade ilustra um dos conceitos fundamentais do Java: a portabilidade. (BESSA, 2023)

O Java é uma linguagem extremamente forte no quesito tipagem. Para criação de modelos de entidade, ele encaixa perfeitamente. Juntamente com o uso de *Springboot*, conseguimos criar serviços fáceis e manuteníveis durante o desenvolvimento.

3.1.1.7 Node

Sistemas para *web* desenvolvidos em outras plataformas usam uma arquitetura denominada bloqueante, a qual cria-se uma fila de requisições, onde cada requisição é processada uma por uma. Diante disso, existe um grande problema nesses sistemas, já que por enfileirar os processamentos, a frequência de gargalos é maior, e consequentemente é preciso aumentar a capacidade dos hardwares dos servidores.

Para resolver tal problema, Ryan Dahl e mais 14 colaboradores criam o Node.js, o qual possui uma arquitetura diferente, denominada de não bloqueante, o que permite fazer diversos processamentos ao mesmo tempo, diminuindo a frequência de gargalos, e fazendo com que não seja necessário, até certo ponto, fazer *upgrade* nos *hardwares* dos servidores. (PEREIRA, 2014)

> “Como um tempo de execução JavaScript assíncrono orientado a eventos, o Node.js foi projetado para construir aplicativos de rede escalonáveis.” NODE.JS (2024)

Node é o que chamamos de *javascript* no lado servidor, que por si só é uma linguagem muito volátil, possibilitando criar diversos sistemas através dele. Node se destaca pela sua facilidade em criar processos assíncronos e sistemas de *streaming* de dados.

3.1.1.8 NoSQL

O termo 'NoSQL' denota uma categoria de bancos de dados não relacionais, nos quais os dados são armazenados em formatos distintos dos tradicionais bancos de dados relacionais baseados em tabelas. No entanto, esses bancos de dados NoSQL são acessíveis através de APIs comuns, linguagens declarativas de consulta estruturada e exemplos de linguagens de consulta, razão pela qual são também conhecidos como bancos de dados "não apenas SQL".

Os desenvolvedores preferem os bancos de dados NoSQL devido à sua afinidade natural com o desenvolvimento ágil, sendo capazes de se adaptar rapidamente às mudanças nos requisitos. Eles permitem uma modelagem de dados mais intuitiva e próxima à forma como os dados são utilizados pelas aplicações, reduzindo a necessidade de transformações extensas ao armazenar ou recuperar dados utilizando APIs NoSQL. Além disso, os bancos de dados NoSQL podem explorar ao máximo a computação em nuvem, proporcionando tempo de inatividade mínimo. (ORACLE 2, 2024)

O NoSQL é um sistema de banco de dados não relacional, ou seja, ao invés de trabalharmos com tabelas, usamos documentos para armazenar nossos dados. A integração dele com o *javascript* é excelente, visto que ele foi criado em cima da linguagem.

3.1.1.9 SQL

SQL, ou Linguagem de Consulta Estruturada, é uma linguagem de programação projetada para manipular informações em um banco de dados relacional. Em um banco de dados relacional, os dados são organizados em tabelas, onde as linhas representam registros individuais e as colunas representam os diferentes atributos dos dados. Por meio de instruções SQL, é possível armazenar, atualizar, excluir, pesquisar e recuperar dados do banco de dados, além de otimizar sua performance.

Essa linguagem de consulta é amplamente adotada em diversas aplicações devido à sua popularidade. Analistas e desenvolvedores de dados recorrem à SQL por sua compatibilidade com várias linguagens de programação. (AWS 2, 2024)

O uso do SQL será fundamental no desenvolvimento de tabelas relacionais, como nas citadas posteriormente no tópico Entidades. Sua utilização se dará principalmente pela capacidade de realizar operações complexas, e lidar com grandes volumes de dados, além de uma versatilidade de uso em qualquer tipo de SGBD.

3.1.1.10 Typescript

Com a presença de erros causados pela tipagem dinâmica do *JavaScript*, a ideia de tipagem estática ajuda a desenvolver um código mais confiável e robusto, diminuindo possíveis erros futuros.

> “TypeScript é uma linguagem de programação fortemente tipada baseada em JavaScript, oferecendo melhores ferramentas em qualquer escala.” TYPESCRIPT (2024)

Typescript nada mais é que um *superset* para a linguagem Javascript. Javascript é uma linguagem de tipagem dinâmica, Typescript veio para suprir essa falta de tipagem dentro dos sistemas *javascript* que ajudam na validação de dados, criação de entidades, etc.

3.2 Sistemas Distribuídos

Para o sistema a ser desenvolvido e levando em consideração as tecnologias e ferramentas escolhidas, o modelo de sistema distribuído que melhor se encaixa é cliente-servidor (vide tópico 2.12.1). Sendo o cliente, a interface, o que é responsável pela interação do usuário, no qual faz requisições que são enviadas ao servidor. Enquanto o servidor é a camada *backend*, que fica responsável pelo processamento das solicitações feitas pelo cliente, acessando e manipulando os dados no banco de dados e devolvendo uma resposta.

No nosso sistema, a camada *frontend* será desenvolvida com o Angular e Typescript, já o *backend* com Node.js e Java, enquanto o banco de dados será o SQL e NoSQL. E por fim, a infraestrutura de nuvem pode ser gerenciada pela AWS.

Um exemplo disso na prática é que através das chamadas HTTP, o *frontend* se comunica com o servidor, com isso, o *backend* recebe os dados e processa a lógica de negócio interagindo com o banco de dados, armazenando, alterando e recuperando dados. Toda essa infraestrutura pode ser hospedada na AWS que oferece serviços que permitem que o sistema seja escalável, seguro e resiliente.

As vantagens de utilizar esse modelo são:

- **Escalabilidade:** possibilidade de adicionar mais instâncias/servidores para comportar mais usuários.
- **Descentralização:** uma separação do processamento e lógica de negócio (que são executados no servidor) da interface do usuário (que é processada no lado do cliente).
- **Segurança:** Existe possiblidade de configuração de sistemas de autenticação para garantia de segurança dos dados presentes no sistema.
- **Controle e flexibilidade:** A atuação de forma independente do cliente e servidor, sendo possível alterações em um sem que haja necessidade de alterações no outro. (AWS 3, 2024)

3.3 Requisitos do sistema

A determinação dos requisitos, sejam eles funcionais ou não funcionais, são de extrema importância para a modelagem do projeto, pois facilitam a identificar quais linguagens e tecnologias são mais adequadas para atender o que é esperado durante o desenvolvimento do sistema. Através deles é possível estabelecer uma conexão entre camadas de desenvolvimento e gerenciamento de banco de dados.

3.3.1 Requisitos funcionais

Os requisitos funcionais são as tarefas que o sistema deve ser capaz de realizar, os problemas e necessidades a serem atendidas com tal desenvolvimento. (VALENTE, 2020).

- Gerenciamento de usuários: o sistema deve permitir que tanto o cliente quanto o fornecedor possam se cadastrar, alterar seus dados e fazer login na plataforma.
- Catálogo de produtos: o papel dos fornecedores ao divulgar seus produtos, com máximo de detalhes possível, além de preço e imagens.
- Processo de compras simples: os clientes podem navegar no sistema em busca de um produto do seu agrado e realizar pagamento de forma segura.

- Gestão de estoque e pedidos: o sistema deve ter um controle do estoque disponível a cada compra realizada ou reposição pelo fornecedor.

3.3.2 Requisitos não funcionais

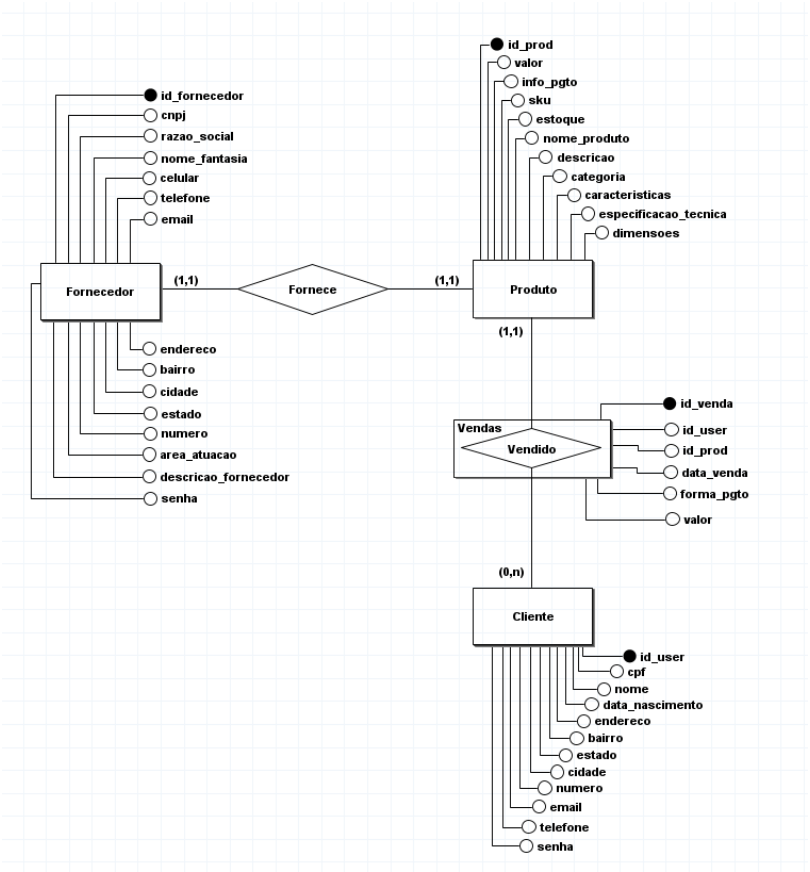
Esses requisitos não funcionais estão relacionados no modo como o sistema funcionará a partir dos requisitos funcionais impostos. Sua característica é garantir o funcionamento correto da plataforma. (VALENTE, 2020)

- Desempenho e escalabilidade: o sistema deve ser capaz de suportar uma grande quantidade de transações realizadas e usuários navegando pela plataforma.
- Segurança: deve conter uma forte segurança em todas as camadas do sistema, para garantir a proteção de todo tipo informação do usuário, como dados confidenciais e financeiros.
- Facilidade de manutenção: a escolha de tecnologias no sistema deve ser visando uma fácil e rápida manutenção, para novas atualizações ou correções de bugs da plataforma.

3.4 Entidades

De acordo com a Figura 1, será exibido um modelo conceitual que abrange como o sistema será desenvolvido no quesito banco de dados. O sistema consistirá em 4 tabelas: Cliente. Fornecedor, Vendas e Produto.

Figura 1 - Modelo Entidade Relacionamento - MER



Fonte: Elaboração própria

A relação entre elas se dá que o fornecedor fornece um produto, produto este que será vendido a um usuário.

Para a relação de fornecedor, será criado as propriedades de id_fornecedor (que será um identificador único para cada fornecedor), CNPJ, razao_social, nome_fantasia, celular, email, telefone, endereco, bairro, cidade, estado, numero, area_atuacao, descricao_fornecedor, senha (que será cadastrada de forma criptografada).

Na tabela de produtos será composta por id_prod, valor, info_pgto, sku, estoque, nome_produto, descrição, categoria, características, especificacao_tecnica, dimensões, id_forn (que será uma chave estrangeira para vincular a qual fornecedor esse produto pertence)

Em Cliente terá id_user, cpf, nome, data_nascimento, endereco, bairro, estado, cidade, numero, email, telefone, senha.

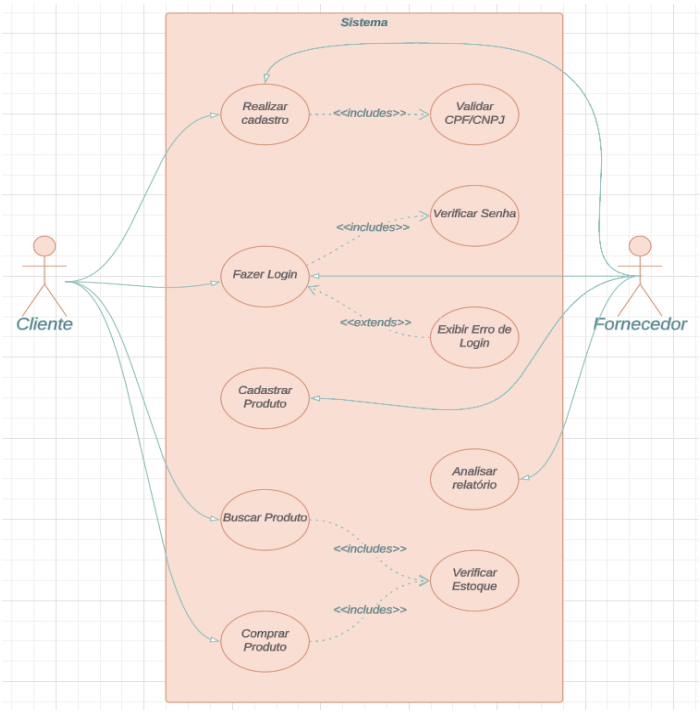
Na relação entre produto e fornecedor será criada a tabela de venda, que armazenará todas as informações das vendas realizadas, dentre elas o id_prod (que

através dele poderá buscar dados do fornecedor desse produto), id_venda, id_user, data_venda, forma_pgto, valor.

3.5 Casos de uso

Conforme dito por Cockburn (2007), o caso de uso é utilizado para dar uma visão geral sobre o sistema, mostrando de forma simples como funciona, e as respectivas ações que os usuários podem ter sobre ele. A interação entre os atores e o sistema se dá através de cenários como se registrar, fazer um login, buscar ou comprar um produto, verificar relatórios.

Figura 2 - Caso de Uso



Fonte: Elaboração própria

Com base na figura 2, ela representa o diagrama de caso de uso do sistema desenvolvido. Ele é constituído por atores, casos de uso e relacionamentos.

Os atores são algo ou alguém que utiliza o sistema afim de atingir determinado objetivo, portanto, para esse caso, são os clientes e os fornecedores os principais responsáveis pela utilização da plataforma.

Os clientes têm como ação realizar cadastro, fazer login, buscar e comprar um produto. Já o fornecedor, além de também fazer seu cadastro e login, ele poderá

cadastrar produto e analisar relatórios de vendas, sejam eles semanais, mensais ou anuais.

As funções incluídas no sistema são pequenas funcionalidades que podem acompanhar um caso de uso base, como por exemplo, ao se registrar, o sistema acionará o método para validar o CPF/CNPJ. O mesmo ocorre quando realizar o login, que irá verificar se a senha está correta, e ao buscar ou comprar um produto, que terá uma breve verificação se o produto procurado está em estoque.

Em contrapartida, uma função estendida é um comportamento opcional para uma determinada ação, pode ou não ocorrer. Para o sistema, um exemplo disso é a mensagem de alerta de senha inválida caso tenha uma tentativa falha de login.

3.6 Diagrama de classe

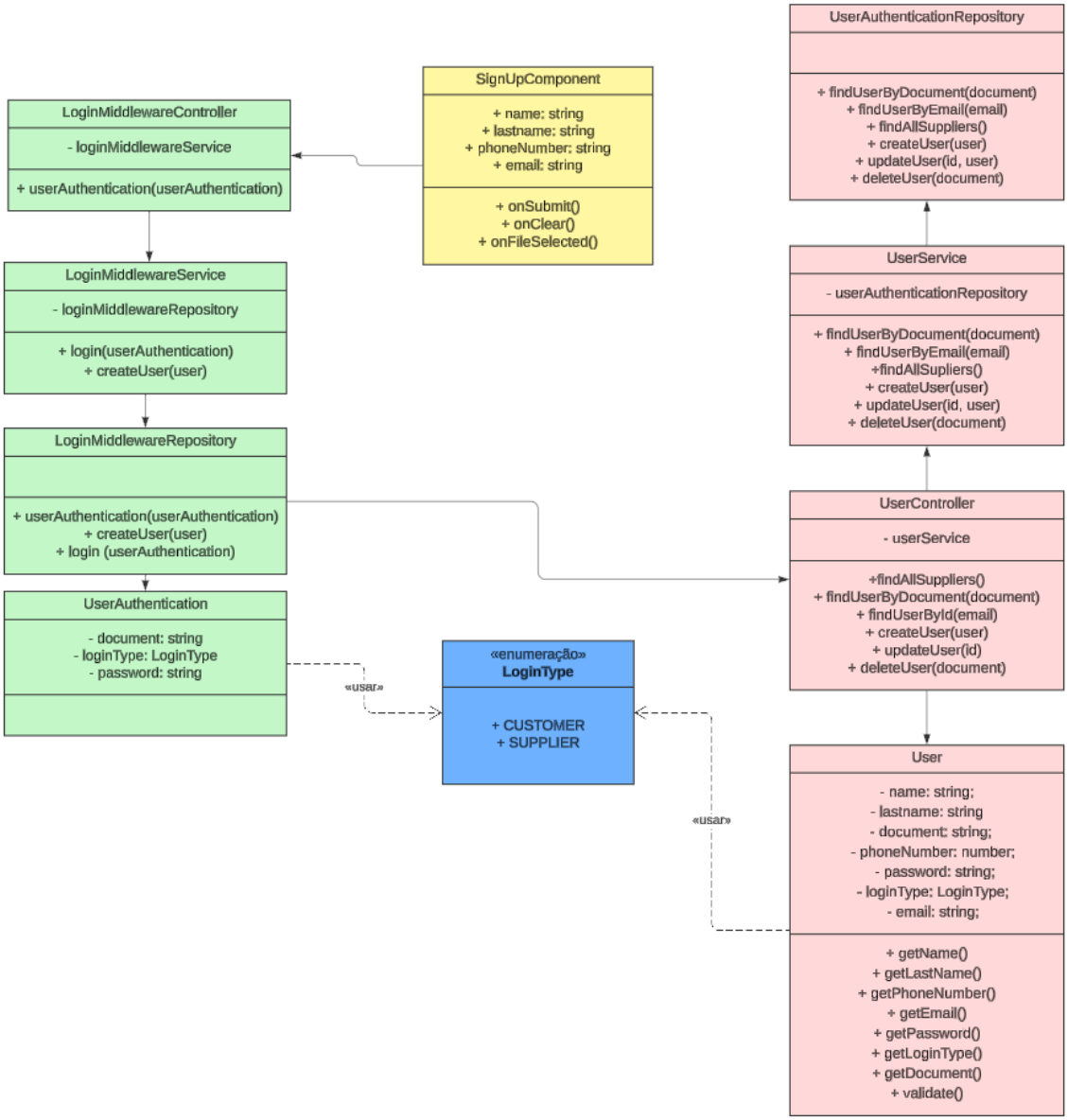
Para todo sistema desenvolvido utilizando o modelo de programação orientada a objetos, o diagrama de classes se torna algo essencial para visualização das classes pertencentes ao sistema, junto de seus métodos e propriedades. (DEVMEDIA, 2024).

Nos tópicos seguintes, serão abordados e explicados os diagramas de classes presentes no sistema desenvolvido.

3.6.1 Login

A figura 3 é uma representação do diagrama de classes da tela de login, que em uma tentativa de login, passa por algumas verificações internas antes de liberar o acesso ao usuário. Validações essas que são executadas fora da visão do usuário e executadas em questão de segundos.

Figura 3 - Diagrama de Classe - Tela de login



Fonte: Elaboração própria

O processo acontece em três etapas, a camada *front-end* (tela do usuário), onde ele irá fornecer as informações de login. Assim que enviadas, passará por um middleware que chamará a classe de decriptação da senha e enviará para outro middleware, onde de fato chamará a API para consulta dos dados do usuário e retornar se login foi bem-sucedido ou dados incorretos.

Para a tela de login, foi implementada um *Auth Guard,* que será responsável pela proteção das rotas, garantindo toda a segurança necessária ao sistema e que

apenas usuários autenticados possam acessá-lo. Além disso, ele guardará os métodos essenciais para a interação com o BFF (*Back-End for Front-End*).

No momento do login, é enviada uma solicitação ao BFF, que por sua vez se responsabilizará por toda a lógica de autenticação, incluindo também a validação dos dados dos usuários. Na sequência, é chamada o serviço “users” para uma confirmação da existência do usuário e consequentemente a geração de um token, sendo o mesmo retornado a tela de login.

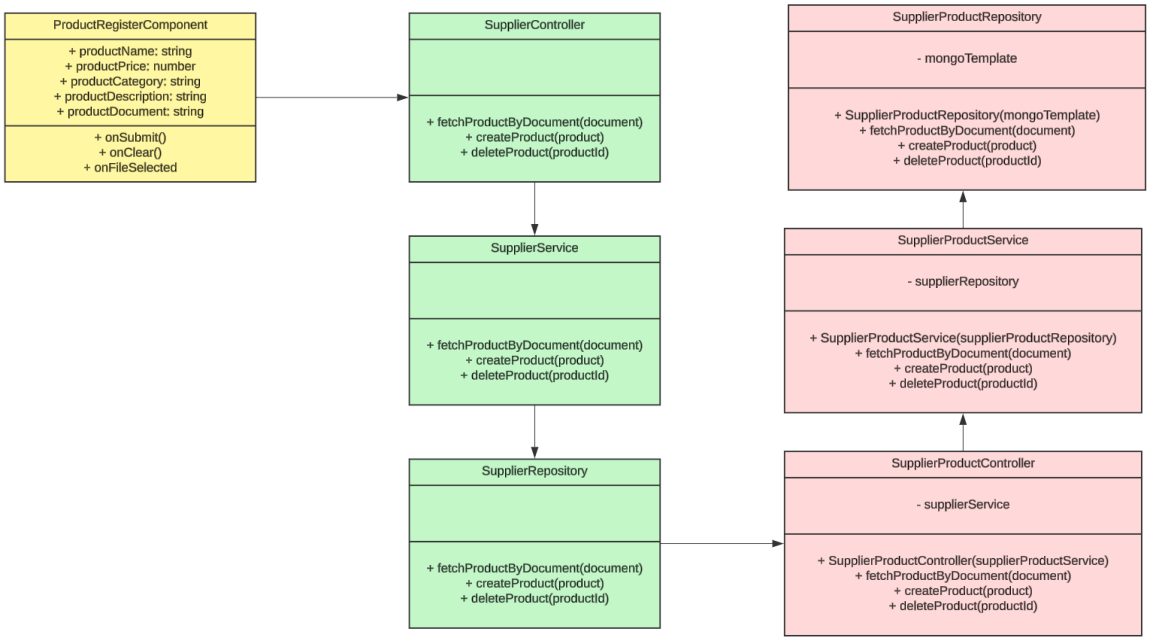
O *Auth Guard* consumirá o token recebido, para que a cada ação do usuário no sistema, seja verificado a validade desse token. Em caso de expiração, o usuário será deslogado automaticamente e precisará logar novamente.

Além do método de login, também foi implementado o de criação de usuário. Ele, por sua vez, também interage com o BFF, que ao receber a solicitação, chama o serviço *users* e realização o cadastro do novo login no sistema através dos modelos definidos.

3.6.2 Produto (Visão fornecedor)

O diagrama de classe apresentado na Figura 4 é uma representação do funcionamento da etapa de produtos na visão do fornecedor.

Figura 4 - Diagrama de Classe - Tela de Produto



Fonte: Elaboração própria

A tela de cadastro de produtos, que é uma visão exclusiva do fornecedor, ela ocorre em três etapas, sendo o front-end, BFF e o micro serviço de produto.

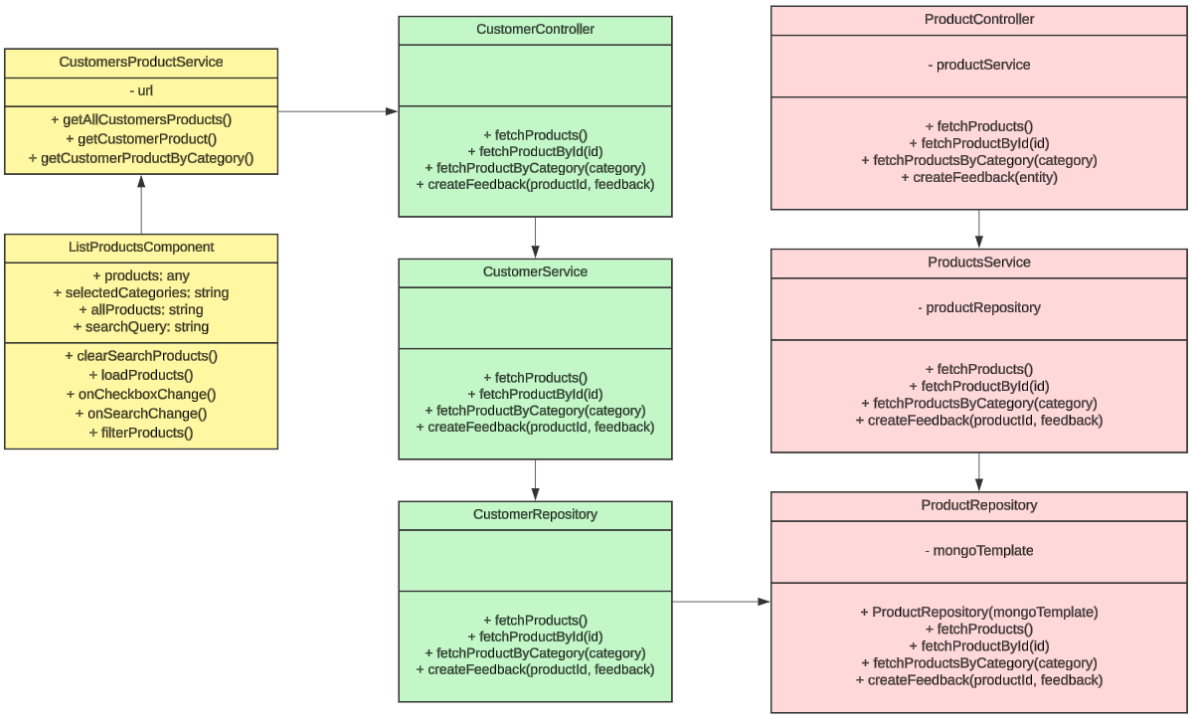
Na camada do front-end, a interação do fornecedor com a tela é para que ele possa cadastrar todos os produtos disponíveis para venda, preenchendo todos os detalhes necessários, para que, posteriormente, eles estejam visíveis ao usuário.

Em seguida, no BFF, são feitas todas as validações dos dados antes do envio ao micro serviço de produtos, que é o responsável pelas ações em cima dos itens cadastrados pelo fornecedor. Ações estas, que além das funcionalidades de criação, também é possível excluir ou realizar alguma filtragem dos dados com base em determinados critérios.

3.6.3 Produto (Visão usuário)

O diagrama de classe na Figura 5 representa o funcionamento do sistema no momento da listagem de produtos.

Figura 5 - Diagrama de Classe - Listagem de Produtos



Fonte: Elaboração própria

Em relação ao comportamento dos dados, ele é similar ao de cadastro de produtos, no qual a requisição é feita pelo front-end, passa pelo BFF do usuário e para finalizar vai ao micro serviço de listagem de produto.

Na etapa do front-end, a interação do usuário com a tela é para que ele possa realizar a listagem dos produtos da maneira que preferir, retornando em tela apenas os produtos desejados, de acordo com seu filtro.

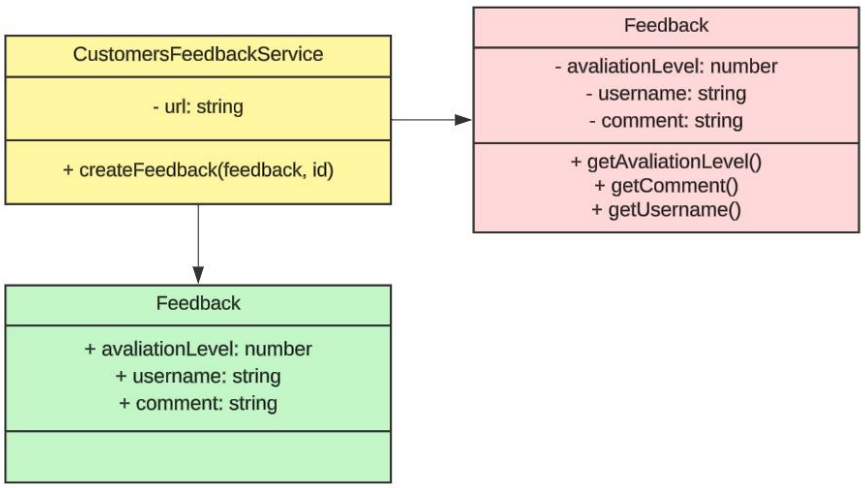
Em seguida, no BFF, são realizadas as tratativas do produto para uma filtragem, seja pela categoria ou pelo ID. Com sucesso nessa etapa, são enviados os parâmetros para o micro serviço de produto, para que ele retorne a tela do cliente apenas o que foi buscado.

Outra ação que esses serviços ficam responsáveis também é pela inserção do feedback do produto.

3.6.4 Feedback

A Figura 6 exibe o modelo do sistema para o cadastro de feedbacks por parte dos usuários, em relação aos produtos vendidos pelos fornecedores.

Figura 6 - Diagrama de Classe – Feedback



Fonte: Elaboração própria

Eles fornecem uma estrutura básica para o sistema de feedback, no qual o CustomerFeedbackService se comunica com o back-end para envio dos dados

informados pelo cliente. Com o método existente nele, é realizada uma requisição HTTP que serve para envio de um novo feedback ou atualização de um já feito.

Já as classes Feedback, servem apenas para manipulação dos objetos, no qual pode acessar as propriedades, ou os métodos presentes nela.

3.7 Algoritmos

Para esse módulo, será apresentado o código presente em cada parte do sistema e suas respectivas responsabilidades, que trabalhando em conjunto, fazem o sistema funcionar corretamente.

Cada tela foi desenvolvida usando o sistema de arquitetura em camadas (*modules, services, controllers* e *repositories)*, onde cada camada tem sua função específica.

- *Controllers* (controlador)*:* atua como uma interface entre o *front-end* e a lógica de negócio (*services).* Ele recebe as requisições e retorna as devidas respostas ao cliente.
- *Modules*: são responsáveis por agrupar os componentes relacionados as funcionalidades que foram lhe atribuídas, como por exemplo autenticação de usuários.
- *Services:* envolve toda a lógica de negócio da aplicação, sendo um mediador entre o *controllers* e *repositories.*
- *Repositories* (repositório)*:* responsável pela interação com o banco de dados, acessando e manipulando os dados. Além disso, fornece os meios necessários para uma comunicação dos services com os dados coletados.

Além de promover a separação das responsabilidades, ajuda na manutenção e escalabilidade do código, facilitando assim, em futuras modificações. (CASTIGLIONI, 2022)

3.7.1 Login

Na sequência, será demonstrado o código da tela de login baseado na arquitetura de camadas.

3.7.1.1 *Controllers – Login*

É realizada uma elaboração de rota para autenticação de usuários, enviando os dados no corpo da requisição para que o controlador delegue a lógica ao LoginMiddlewareService.

```
import { Body, Controller, Post } from "@nestjs/common";

import { LoginMiddlewareService } from "../services/login-middleware.service";
import { UserAuthentication } from "../models/user-authentication.model";
import { User } from "../models/user.model";

@Controller("authentication")
export class LoginMiddlewareController {
  constructor(
    private readonly loginMiddlewareService: LoginMiddlewareService,
  ) {}

  @Post()
  async login(@Body() userAuthentication: UserAuthentication): Promise<any> {
      return await this.loginMiddlewareService.login(userAuthentication);
  }
}
```

3.7.1.2 *Modules – Login*

O *module* do login é composto por três arquivos, o *user-types.enum.ts, user-authentication.model.ts, user.models.ts.*

3.7.1.2.1 *user-types.enum.ts*

Esse módulo disponibiliza um enum com duas opções: CUSTOMER e SUPPLIER, que serve para identificar o tipo de usuário a se logar no sistema, um cliente ou fornecedor.

```
export enum LoginType {
  CUSTOMER = 'CUSTOMER',
  SUPPLIER = 'SUPPLIER',
}
```

3.7.1.2.2 *user-authentication.model.ts*

Serve como modelo de dados para autenticação do usuário, com as propriedades *document, loginType e password.*

```
import { LoginType } from './enums/user-types.enum';

export type UserAuthentication = {
  document: string;
  loginType: LoginType;
  password: string;
}
```

3.7.1.3 *user.model.ts*

É um modelo de dados para criação de um novo usuário, com propriedades *name, lastname, password, loginType, phoneNumber, email,* e os métodos *getName(), getLastName(), getPassword(), getLoginType(), getPhoneNumber()* e *getEmail()* com função de acessar os valores.

```
import { LoginType } from './enums/user-types.enum';

export class User {
  private name: string;
  private lastname: string;
  private password: string;
  private loginType: LoginType;
  private phoneNumber: number;
  private email: string;

  getName(): string {
    return this.name;
  }

  getLastname(): string {
    return this.lastname;
  }

  getPhoneNumber(): number {
    return this.phoneNumber;
  }
```

```
  getEmail(): string {
    return this.email;
  }

  getPassword(): string {
    return this.password;
  }

  getLoginType(): LoginType {
    return this.loginType;
  }
}
```

3.7.1.4 *Repositories – Login*

Fica encarregado da comunicação com as APIs de login e criação dos usuários, sendo as requisições feitas através do axios.

```
import { Injectable } from "@nestjs/common";

import axios from "axios";

import { UserAuthentication } from "../models/user-authentication.model";
import { User } from "../models/user.model";
import { API_BASE_URL } from "../../../../envoriments/envoriments";

@Injectable()
export class LoginMiddlewareRepository {
  async userAuthentication(user: UserAuthentication): Promise<string> {
    try {
      const request = await axios.post(
        `${API_BASE_URL.USERS}/api/authentication`,
        user,
      );
      const { data } = request;
      return data;
    } catch (error) {
      console.log(
        `Middleware - Ocorreu um erro ao autenticar usuário: ${error}`,
      );
      throw error;
    }
  }
  async createUser(user: User): Promise<User> {
    try {
      const request = await axios.post<User>(
```

```
        `${API_BASE_URL.USERS}/login`,
        user,
      );
      const { data } = request;
      return data;
    } catch (error) {
      console.log(`Middleware - Ocorreu um erro ao criar o usuário:
${error}`);
      throw error;
    }
  }

  async login(userAuthentication: UserAuthentication): Promise<any>{
    try{
      const request = await axios.post<any>(
        `${API_BASE_URL.USERS}/api/authetication`,
        userAuthentication
      );
      return request.data;

    } catch (error) {
      console.log(
        `Erro na autenticação: ${error}`
      );
      throw error;
    }
  }
}
```

Ambos os métodos *userAuthentication* e *createUser* enviam dados para API e retornam resultados, mas enquanto o primeiro apenas faz uma validação da autenticidade retornando em tela se ok, o segundo faz requisição de criação de usuário mostrando no fim o *user* criado.

Por fim, o método *login* faz a requisição POST para autenticar o usuário, retornando a resposta da API. Caso retorne erro, ele é lançado no console.

3.7.1.5 *Services – Login*

Esse código é a ponte entre o *controller* e *repositor,* responsável pela implementação da lógica de negócios para criação de usuários e autenticação.

```
import { Injectable } from "@nestjs/common";

import { UserAuthentication } from "../models/user-authentication.model";
import { User } from "../models/user.model";
import { LoginMiddlewareRepository } from "../repositories/login-
middleware.repository";

@Injectable()
export class LoginMiddlewareService {
  constructor(
    private readonly loginMiddlewareRepository: LoginMiddlewareRepository,
  ) {}

  async createUser(user: User): Promise<User> {
    if (!user) throw new Error("Usuário inválido.");
    return await this.loginMiddlewareRepository.createUser(user);
  }

  async login(userAuthentication: UserAuthentication): Promise<any>{
    return await
this.loginMiddlewareRepository.userAuthentication(userAuthentication);
  }
}
```

O *createUser* faz uma validação das informações do novo usuário e delega a sua criação para o repositório. Já o *login* recebe o *userAuthentication* como parâmetro, passando suas credenciais e aguardando um retorno, para verificar a veracidade dos dados.

3.7.2 Registro de Produto

Com base na arquitetura em camadas, os tópicos de 3.7.2.1 a 3.7.2.7 apresentam os códigos relacionados ao cadastro de um produto no sistema.

3.7.2.1 *product-register.component.ts*

Esse componente permite que o usuário cadastre um produto ao preencher o formulário. Quando realizado o envio, é montada uma estrutura JSON contendo todos os dados e é passado como payload para o método de criação de produto. Ocorrendo tudo certo no cadastro, o formulário é limpo, permitindo um novo cadastro. Caso contrário, uma mensagem de erro é exibida

```
import { Component } from '@angular/core';
import { HeaderComponent } from '../header/header.component';
import { FooterComponent } from '../footer/footer.component';
import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from
'@angular/forms';
import { SuppliersProductService } from '../../services/suppliers-
product.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-product-register',
  standalone: true,
  imports: [HeaderComponent, FooterComponent, ReactiveFormsModule,
CommonModule],
  templateUrl: './product-register.component.html',
  styleUrl: './product-register.component.css'
})
export class ProductRegisterComponent {
  productForm: FormGroup

  constructor(
    private fb: FormBuilder,
    private suppliersProductService: SuppliersProductService
  ) {
    this.productForm = this.fb.group({
      productName: ['', Validators.required],
      productPrice: ['', Validators.required],
      productCategory: ['', Validators.required],
      productDescription: [''],
      productSpecification: [''],
      productDocument: [null]
    })
  }

  onSubmit() {
    if (this.productForm.valid) {
      const formData = this.productForm.value

      //payload teste (fix)
      const payload = {
        name: formData.productName,
        price: formData.productPrice,
        image: "~/images/",
        category: formData.productCategory,
        feedbacks: [],
        specifications: {
            model: formData.productName,
            color: null,
            description: formData.productDescription
```

```
        }
      }

      this.suppliersProductService.createProduct(payload).subscribe({
        next: (response) => {
          console.log('Produto cadastrado com sucesso:', response)
          this.onClear()
        },
        error: (error) => {
          console.error('Erro ao cadastrar o produto:', error)
        }
      })
    } else {
      console.log('Formulário inválido!');
    }
  }

  onClear() {
    this.productForm.reset();
  }

  onFileSelected(event: any) {
    const file = event.target.files[0];
    if (file) {
      this.productForm.patchValue({ productDocument: file });
    }
  }
}
```

O método *onFileSelected* nada mais é do que uma atualização do campo de arquivo do formulário.

3.7.2.2 *suppliers.controller.ts*

Esse controlador fornece as funções básicas de um CRUD relacionadas aos produtos dos fornecedores, como por exemplo criação, exclusão e busca por documento.

```
import { Body, Controller, Delete, Get, Param, Post } from "@nestjs/common";
import { Products } from "../../models/products.model";
import { SupplierService } from "../services/suppliers.service";
```

```
@Controller('supplier/product')
export class SupplierController{

    constructor(private readonly supplierService: SupplierService) {}

    @Get(':document')
    async fetchProductByDocument(@Param('document') document: string):
Promise<Products[]>{
        return await this.supplierService.fetchProductByDocument(document);
    }

    @Post()
    async createProduct(@Body() product: Products): Promise<Products>{
        return await this.supplierService.createProduct(product);
    }

    @Delete(':productId')
    async deleteProduct(@Param('productId') productId: string): Promise<void>{
        await this.supplierService.deleteProduct(productId);
    }
}
```

Os três *endpoints* presentes no *SupplierController* são o *GET* (*fetchProductByDocument*) para busca dos produtos pelo documento, o *POST* (*createProduct*) para criação de um novo produto, e por fim, o *DELETE* (*deleteProduct*) para exclusão do produto.

3.7.2.3 *supplier.repository.ts*

Ele fica responsável pelas requisições para uma API externa, sejam elas para busca, criação ou exclusão de produtos. Cada método presente possui suas devidas tratativas, tanto para o envio correto quanto para quando ocorre algum erro, para que fiquem registrados e gerenciados.

```
import { Injectable } from "@nestjs/common";
import { Products } from "../../models/products.model";
import axios from "axios";
import { API_BASE_URL } from "src/envoriments/envoriments";

@Injectable()
export class SupplierRepository{
    async fetchProductByDocument(document : string): Promise<Products[]>{
        try {
```

```
            const request = await axios.get(
                `${API_BASE_URL.PRODUCTS_SUPPLIER}/${document}`
            );
            return request.data;

        } catch (error) {
            console.log(
                `Erro na procura do fornecedor: ${error}`
            );
            throw error;
        }
    }

    async createProduct(product: Products): Promise<Products>{
        try {
            const request = await axios.post(
                `${API_BASE_URL.PRODUCTS_SUPPLIER}`,
                product
            );
            return request.data

        } catch (error) {
            console.log(
                `Erro ao criar produto: ${error}`
            );
            throw error;
        }

    }

    async deleteProduct(productId: string): Promise<void>{
        try {
            await axios.delete(
                `${API_BASE_URL.PRODUCTS_SUPPLIER}/${productId}`
            );
        } catch (error) {
            console.log(
                `Erro ao deletar produto pelo ID: ${error}`
            );
            throw error;

        }
    }
}
```

3.7.2.4 suppliers.service.ts

Sua responsabilidade é coordenar as operações de busca, criação e exclusão de produtos. Ele faz utilização do *SupplierRepository (supplier.repository.ts)* para realizar as chamadas necessárias das APIs e enviar os dados retornados para o controlador (*suppliers.controller.ts*), separando assim, logicamente as responsabilidades entre arquivos.

```
import { Injectable } from "@nestjs/common";
import { Products } from "../../models/products.model";
import { SupplierRepository } from "../repositories/supplier.repository";

@Injectable()
export class SupplierService{
    constructor(private readonly supplierRepository: SupplierRepository) {}

    async fetchProductByDocument(document: string): Promise<Products[]>{
        return await this.supplierRepository.fetchProductByDocument(document);
    }
    async createProduct(product: Products): Promise<Products>{
        return await this.supplierRepository.createProduct(product);
    }
    async deleteProduct(productId: string): Promise<void>{
        await this.supplierRepository.deleteProduct(productId);
    }
}
```

3.7.2.5 *SupplierProductController.java*

Esse controlador em JAVA atua como uma interface entre as requisições HTTP e a lógica de negócios que compõem o *SupplierProductService*. Através dele são definidos os *endpoints* para criação, exclusão e busca dos produtos dos fornecedores.

```
package com.fier.products.modules.suppliers.controllers;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.fier.products.modules.models.entity.Product;
import com.fier.products.modules.suppliers.services.SupplierProductService;

import java.util.List;
```

```
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@RestController
@RequestMapping("supplier/product")
public class SupplierProductController {

    private final SupplierProductService supplierService;
    public SupplierProductController(SupplierProductService
supplierProductService) {
        this.supplierService = supplierProductService;
    }

    @GetMapping("{document}")
    public List<Product> fetchProductByDocument(@PathVariable String document)
{
        return this.supplierService.fetchProductByDocument(document);
    }

    @PostMapping()
    public Product createProduct(@RequestBody Product product) {
        return this.supplierService.createProduct(product);
    }

    @DeleteMapping("{productId}")
    public void deleteProduct(@PathVariable String productId) {
        this.supplierService.deleteProduct(productId);
    }

}
```

3.7.2.6 SupplierProductRepository.java

É a camada de acesso aos dados, usando o MongoDB para armazenamento dos dados dos produtos. Com o encapsulamento das operações do banco de dados realizado por essa camada, permite que o serviço ou controlador acesse os métodos sem a preocupação dos detalhes de conexão e implementação com base de dados.

```java
package com.fier.products.modules.suppliers.repositories;

import java.util.List;

import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.stereotype.Repository;

import com.fier.products.modules.models.entity.Product;

@Repository
public class SupplierProductRepository {
    private final MongoTemplate mongoTemplate;

    public SupplierProductRepository(MongoTemplate mongoTemplate) {
        this.mongoTemplate = mongoTemplate;
    }

    public List<Product> fetchProductByDocument(String document) {
        var query = new Query()
            .addCriteria(Criteria.where("document").is(document));

        return this.mongoTemplate.find(query, Product.class);
    }

    public Product createProduct(Product product) {
        return this.mongoTemplate.save(product);
    }

    public void deleteProduct(String productId) {
        var query = new Query()
            .addCriteria(Criteria.where("_id").is(productId));
        this.mongoTemplate.remove(query, Product.class);
    }
}
```

3.7.2.7 *SupplierProductService.java*

O encapsulamento da lógica de negócio dos produtos dos fornecedores fica nessa camada, e através dela são fornecidos os métodos que o controlador pode chamar para realização das operações de busca, criação e exclusão.

```java
package com.fier.products.modules.suppliers.services;
import java.util.List;
```

```java
import org.springframework.stereotype.Service;
import com.fier.products.modules.models.entity.Product;
import
com.fier.products.modules.suppliers.repositories.SupplierProductRepository;

@Service
public class SupplierProductService {

    private final SupplierProductRepository supplierRepository;
    public SupplierProductService(SupplierProductRepository
supplierProductRepository) {
        this.supplierRepository = supplierProductRepository;
    }

    public List<Product> fetchProductByDocument(String document) {
        return this.supplierRepository.fetchProductByDocument(document);
    }

    public Product createProduct(Product product) {
        return this.supplierRepository.createProduct(product);
    }

    public void deleteProduct(String productId) {
        this.supplierRepository.deleteProduct(productId);
    }
}
```

3.7.3 Listagem de Produtos

Com base na arquitetura em camadas, os tópicos de 3.7.3.1 a 3.7.3.7 apresentam os códigos relacionados a listagem ou filtragem de um ou mais produtos no sistema.

3.7.3.1 *list-products.component.ts*

Esse componente em *typescript* é o responsável pela regra do front-end da tela de listagem de produtos.

Ele é composto pelos métodos *clearSearchProducts, loadProducts, onCheckboxChange, onSearchChange* e *filterProducts*.

```
import { CustomersProductService } from './../../services/customers-
product.service'
import { Component } from '@angular/core'
import { CommonModule } from '@angular/common'
import { HeaderComponent } from '../header/header.component'
import { FooterComponent } from '../footer/footer.component'
import { CardProductComponent } from '../card-product/card-product.component'
import { forkJoin } from 'rxjs'
import { catchError } from 'rxjs/operators'
import { of } from 'rxjs'

@Component({
  selector: 'app-list-products',
  standalone: true,
  imports: [HeaderComponent, FooterComponent, CardProductComponent,
CommonModule],
  templateUrl: './list-products.component.html',
  styleUrls: ['./list-products.component.css']
})
export class ListProductsComponent {
  products: any[] = [];
  selectedCategories: Set<string> = new Set();

  allProducts: any[] = []
  searchQuery: string = ''

  constructor(private customersProductService: CustomersProductService) {}

  ngOnInit(): void {
    this.loadProducts();
  }

  clearSearchProducts(): void {
    this.customersProductService.getAllCustomersProducts().subscribe(data => {
      if (data && data.length > 0) {
        this.products = data
      } else {
        this.products = []
      }
    })
  }

  loadProducts(): void {
    const categories = Array.from(this.selectedCategories);

    if (categories.length > 0) {
      const productRequests = categories.map(category => {
        return
this.customersProductService.getCustomerProductByCategory(category).pipe(
```

```
          catchError(err => {
            console.error(`Err ${category}:`, err)
            return of([])
          })
        )
      })

      forkJoin(productRequests).subscribe(results => {
        const allProducts = results.flat()

        if (allProducts.length > 0) {
          this.allProducts = allProducts
        } else {
          this.allProducts = []
        }

        this.filterProducts()
      })
    } else {
      this.customersProductService.getAllCustomersProducts().subscribe(data =>
{
        if (data && data.length > 0) {
          this.allProducts = data
        } else {
          this.allProducts = []
        }

        this.filterProducts()
      });
    }
  }

  onCheckboxChange(event: any, category: string): void {
    if (event.target.checked) {
      this.selectedCategories.add(category)
    } else {
      this.selectedCategories.delete(category)
    }

    this.loadProducts()
  }

  onSearchChange(searchQuery: string): void {
    this.searchQuery = searchQuery
    this.filterProducts()
  }

  filterProducts(): void {
    if (this.searchQuery) {
```

```
      this.products = this.allProducts.filter(product =>
        product.name.toLowerCase().includes(this.searchQuery.toLowerCase())
      )
    } else {
      this.products = this.allProducts
    }
  }
}
```

Uma breve explicação sobre o funcionamento dos métodos presentes nesse componente:

- *clearSearchProducts*: faz uma limpa nos filtros realizados anteriormente e exibe todos os produtos da API.
- *loadProducts:* verifica se possui categorias selecionadas, para que possa fazer uma busca específica para cada uma delas, se houver. Caso não exista categorias selecionadas, por padrão, irá trazer todos os produtos disponíveis na API.
- *onCheckboxChange:* função responsável pelo controle da inserção ou não de uma categoria ao filtro de busca.
- *onSearchChange:* atualiza o *searchQuery* para realizar a filtragem dos produtos.
- *filterProducts:* função principal que realiza o filtro dos produtos.

3.7.3.2 *customers.controller.ts*

Ele expõe quatro *endpoints* para esse serviço, sendo eles *GET* (*product, productId, productCategory*) e *PUT* (*feedback*).

As chamadas *GET* são responsáveis pela busca dos produtos, sejam eles por ID, por categoria ou retornar todos presente na base de dados. Já o *PUT* fica com a função de cadastrar um *feedback* para o produto.

```
import { Body, Controller, Get, Param, Put } from "@nestjs/common";
import { Products } from "../../models/products.model";
import { CustomerService } from "../services/customers.service";
import { Feedback } from "../../models/feedback.model";

@Controller('customer/products')
export class CustomerController{
```

```
    constructor(private readonly customerService: CustomerService) {}

    @Get()
    async fetchProducts(): Promise<Products[]> {
        return await this.customerService.fetchProducts();
    }

    @Get(':id')
    async fetchProductsById(@Param('id') id: string): Promise<Products>{
        return await this.customerService.fetchProductsById(id);
    }

    @Get('categories/:category')
    async fetchProductsByCategory(@Param('category') category: string):
Promise<Products[]>{
        return await this.customerService.fetchProductsByCategory(category);
    }

    @Put('feedback/:productId')
    async createFeedback(@Param('productId') productId: string, @Body()
feedback: Feedback): Promise<boolean>{
        return await this.customerService.createFeedback(productId, feedback);
    }
}
```

3.7.3.3 *customers.repository.ts*

Através desse componente é realiza douma interação direta com a API externa, fazendo chamadas para buscar e manipular os dados dos produtos.

```
import { Injectable } from "@nestjs/common";
import { Products } from "../../models/products.model";
import axios from "axios";
import { API_BASE_URL } from "src/envoriments/envoriments";
import { Feedback } from "../../models/feedback.model";

@Injectable()
export class CustomerRepository {
    async fetchProducts(): Promise<Products[]> {
        try {
            const request = await axios.get(
                `${API_BASE_URL.PRODUCTS_CUSTOMER}`
            );
            return request.data;
```

```
        } catch (error) {
            console.log(
                `Erro na requisição de produtos: ${error}`
            );
            throw error;

        }
    }

    async fetchProductsById(id: string): Promise<Products> {
        try {
            const request = await axios.get(
                `${API_BASE_URL.PRODUCTS_CUSTOMER}/${id}`
            );
            return request.data;

        } catch (error) {
            console.log(
                `Erro na procura por id do produto: ${error}`
            );
            throw error;

        }
    }

    async fetchProductsByCategory(category: string): Promise<Products[]>{
        try {
            const request = await axios.get(
                `${API_BASE_URL.PRODUCTS_CUSTOMER}/categories/${category}`
            );
            return request.data;

        } catch (error) {
            console.log(
                `Erro na procura por categoria de produto: ${error}`
            );
            throw error;
        }
    }

    async createFeedback(productId: string, feedback: Feedback):
Promise<boolean>{
        try {
            const request = await axios.put(
                `${API_BASE_URL.PRODUCTS_CUSTOMER}/feedback/${productId}`,
                feedback
            );
            return request.data;
```

```
        } catch (error) {
            console.log(
                `Erro no envio de feedback: ${error}`
            );
            throw error;
        }
    }
}
```

3.7.3.4 *customers.service.ts*

Essa é camada intermediadora entre o controlador e o repositório, ficando responsável pela delegação das operações do CRUD dos produtos e do *feedback*.

Todos os métodos presentes nela realizam chamadas para o repositório, porém cada um com suas responsabilidades individuais, que são elas:

- *fetchProducts:* obter todos os produtos.
- *fetchProductsById:* obter um produto pelo ID.
- *fetchProductsByCategory:* obter os produtos por categoria.
- *createFeedback:* envio do *feedback* para o produto.

```
import { Injectable } from "@nestjs/common";
import { Products } from "../../models/products.model";
import { CustomerRepository } from "../repositories/customers.repository";
import { Feedback } from "../../models/feedback.model";

@Injectable()
export class CustomerService{

    constructor(private readonly customerRepository: CustomerRepository) {}

    async fetchProducts(): Promise<Products[]>{
        return await this.customerRepository.fetchProducts();
    }

    async fetchProductsById(id: string): Promise<Products>{
        return await this.customerRepository.fetchProductsById(id);
    }
    async fetchProductsByCategory(category: string): Promise<Products[]>{
        return await
this.customerRepository.fetchProductsByCategory(category);
    }
```

```
    async createFeedback(productId: string, feedback: Feedback):
Promise<boolean>{
        return await this.customerRepository.createFeedback(productId,
feedback);
    }
}
```

3.7.3.5 *ProductController.java*

Esse controlador disponibiliza os *endpoints* para recuperação dos produtos por categoria ou ID, e permite também adicionar os *feedbacks* a um produto específico.

```
package com.fier.products.modules.customers.controllers;

import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestBody;

import com.fier.products.modules.customers.services.ProductsService;
import com.fier.products.modules.models.entity.Feedback;
import com.fier.products.modules.models.entity.Product;

@RestController
@RequestMapping("/customer/products")
public class ProductController {

    private final ProductsService productService;

    public ProductController(ProductsService productService) {
        this.productService = productService;
    }

    @GetMapping()
    public List<Product> fetchProducts() {
        return this.productService.fetchProducts();
    }

    @GetMapping("{id}")
    public Product fetchProductById(@PathVariable String id) {
        return this.productService.fetchProductById(id);
```

```
    }

    @GetMapping("categories/{category}")
    public List<Product> fetchProductsByCategory(@PathVariable String
category) {
        return this.productService.fetchProductsByCategory(category);
    }

    @PutMapping("feedback/{productId}")
    public boolean createFeedback(@PathVariable String productId, @RequestBody
Feedback entity) {
        return this.productService.createFeedbacks(productId, entity);
    }
}
```

3.7.3.6 *ProductRepository.java*

A responsabilidade dessa camada é estar com os dados fornecidos pelo banco de dados MongoDB. Através de consultas e atualizações, ele ajuda na recuperação das informações de um ou mais produtos, sejam eles filtrados pelo ID ou pela categoria. Adicionalmente, também faz inserção dos dados do *feedback*.

```
package com.fier.products.modules.customers.repositories;
import java.util.List;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.Repository;
import com.fier.products.modules.models.entity.Feedback;
import com.fier.products.modules.models.entity.Product;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.core.query.Update;

@Repository
public class ProductRepository {
    private final MongoTemplate mongoTemplate;
    public ProductRepository(MongoTemplate mongoTemplate) {
        this.mongoTemplate = mongoTemplate;
    }

    public List<Product> fetchProducts() {
        return this.mongoTemplate.findAll(Product.class);
    }

    public Product fetchProductById(String id) {
        var query = new Query()
```

```
            .addCriteria(Criteria.where("_id").is(id));

        return this.mongoTemplate.findOne(query, Product.class);
    }

    public List<Product> fetchProductsByCategory(String category) {
        var query = new Query()
            .addCriteria(Criteria.where("category").is(category));

        return this.mongoTemplate.find(query, Product.class);
    }

    public boolean createFeedback(String productId, Feedback feedback) {
        var query = new Query()
            .addCriteria(Criteria.where("_id").is(productId));
        var update = new Update()
            .push("feedbacks", feedback);
        var updated = this.mongoTemplate.updateFirst(query, update,
Product.class);

        return updated.getMatchedCount() > 0;
    }
}
```

3.7.3.7 *ProductService.java*

Atua como intermediário entre o controlador e o repositório, fornecendo os métodos necessários para encapsulamento da lógica de negócio da listagem dos produtos e cadastro de *feedback*. É possível fazer busca pelos produtos usando o ID (*fetchProductById*) ou categoria (*fetchProductsByCategory*), além da inserção de feedbacks no produto (*createFeedbacks*).

```
package com.fier.products.modules.customers.services;

import java.util.List;

import org.springframework.stereotype.Service;

import com.fier.products.modules.customers.repositories.ProductRepository;
import com.fier.products.modules.models.entity.Feedback;
import com.fier.products.modules.models.entity.Product;

@Service
public class ProductsService {
```

```
    private final ProductRepository productRepository;
    public ProductsService(ProductRepository productRepository) {
        this.productRepository = productRepository;
    }

    public List<Product> fetchProducts() {
        return this.productRepository.fetchProducts();
    }

    public Product fetchProductById(String id) {
        return this.productRepository.fetchProductById(id);
    }

    public List<Product> fetchProductsByCategory(String category) {
        return this.productRepository.fetchProductsByCategory(category);
    }

    public boolean createFeedbacks(String productId, Feedback feedback) {
        return this.productRepository.createFeedback(productId, feedback);
    }
}
```

3.7.4 Link GitHub

Abaixo estarão os links dos repositórios para consulta dos demais códigos utilizados no desenvolvimento do projeto.

- https://github.com/faustoneris/fier
- https://github.com/faustoneris/users
- https://github.com/faustoneris/fier-bff
- https://github.com/faustoneris/products
- https://github.com/faustoneris/emails

4 RESULTADOS

Os resultados obtidos com o desenvolvimento e testes da plataforma de *E-commerce* Invertido evidenciam seu potencial para atender às necessidades dos microempreendedores e para melhorar a experiência de compra dos consumidores.

O *feedback* obtido em testes iniciais com microempreendedores escolhidos foi um dos aspectos mais notáveis. Esta interação forneceu *feedback* crucial para a melhoria constante da plataforma, confirmando funcionalidades e permitindo modificações de acordo com as necessidades reais dos usuários, o que é crucial para a experiência prática e eficiente do sistema.

A plataforma possui várias funcionalidades para ajudar na promoção e divulgação dos microempreendedores, elementos fundamentais para empreendimentos de pequeno porte. Dentre os recursos disponibilizados, merecem destaque a possibilidade de envio de avaliações pelos usuários e um meio direto de comunicação entre cliente e fornecedor. Estes elementos fomentam maior clareza e criam uma ligação mais direta entre as partes envolvidas, reforçando a confiança e aprimorando a experiência de compra.

A plataforma foi projetada tecnicamente para lidar com um grande volume de transações e proporcionar segurança em todas as camadas do sistema. A estrutura assegura desempenho, escalabilidade e operação segura ao atender aos requisitos funcionais e não funcionais. Esses elementos são essenciais para preservar a confiança dos usuários, oferecendo uma experiência de navegação segura e consistente.

Em resumo, os achados sugerem que a plataforma de *E-commerce* Invertido atinge seu objetivo principal de destacar microempresários, proporcionando um ambiente sólido e seguro onde eles podem divulgar seus produtos e serviços de maneira mais independente e abrangente. Esses progressos representam um avanço importante para consolidar o comércio digital inclusivo, expandindo as chances de visibilidade e êxito para pequenas empresas no ambiente virtual.

5 CONCLUSÃO

Este projeto propôs o desenvolvimento de um modelo inovador de comércio eletrônico "*e-commerce* invertido" com o objetivo central de valorizar os microempreendedores e otimizar sua visibilidade e oportunidades de interação com os consumidores. A proposta visa atender à crescente demanda por soluções de comércio digital, inclusive entre as pequenas empresas, que enfrentam desafios de acesso ao mercado e tecnológicos que esses empreendedores frequentemente enfrentam. Diferente do modelo tradicional que prioriza grandes marcas e exerce forte influência nas escolhas dos consumidores, o e-commerce reverso tem como foco expor os microempreendedores e promover os clientes a obterem um ambiente de experiência de compra mais personalizado e consciente.

Ao longo do processo de desenvolvimento, foram explorados diversos recursos técnicos e métodos para apoiar a criação de um sistema escalável, seguro e fácil de usar. Tecnologias como Angular, AWS e NoSQL foram escolhidas pelo alto desempenho e segurança, enquanto a arquitetura cliente-servidor oferece flexibilidade para usuários finais e fornecedores. Por exemplo, a integração com a função de chat e a avaliação direta de fornecedores permitem que a plataforma, ao fomentar relações mais próximas e de confiança entre clientes e microempreendedores (importante fator de fidelização e confiança no comércio *online*), se torne num fator diferenciador.

Além dos aspectos técnicos, o projeto também se baseia em princípios éticos e legais, como o cumprimento da LGPD. Isto é especialmente importante no atual cenário digital, onde a proteção de dados é crucial para a confiabilidade das plataformas de comércio eletrônico.

No aspecto socioeconômico, o impacto esperado é significativo: ao divulgar os produtos e serviços de pequenos empresários, a plataforma auxilia no fortalecimento da economia local, estimulando a compra de produtos variados e criando chances de trabalho e desenvolvimento nas comunidades. O *E-commerce* invertido não só expande o acesso dos microempresários ao ambiente digital, mas também fomenta um padrão de consumo mais inclusivo e sustentável.

Em relação a futuros projetos, a plataforma possui capacidade para se desenvolver de diversas formas. A inclusão de recursos como um programa de recompensas para clientes regulares, criação de um chatbot para melhor atender o

cliente, incorporação de inteligência artificial para aprimorar ainda mais a experiência de compra e a implementação de relatórios sofisticados para microempresários são apenas alguns dos caminhos possíveis. Ademais, colaborações com outras ações de suporte ao microempreendedor, como cursos de capacitação em *marketing* digital, poderiam potencializar ainda mais o efeito da plataforma, oferecendo um apoio mais completo para os pequenos negócios.

Em suma, o *E-commerce* Invertido é um progresso notável na área de comércio eletrônico, proporcionando um ambiente equitativo e favorável ao desenvolvimento dos microempresários. Supõe-se que este modelo possa auxiliar na construção de uma economia mais balanceada e interligada, ao mesmo tempo que fomenta um mercado mais ativo e variado.

REFERÊNCIAS BIBLIOGRÁFICAS

ATLASSIAN 1. **O que é DevOps?** Disponível em: <https://www.atlassian.com/br/devops> Acesso em: 08 de maio de 2024.

ATLASSIAN 2. **O que é computação em nuvem? Uma visão geral da nuvem.** Disponível em: <https://www.atlassian.com/br/microservices/cloud-computing> Acesso em: 09 de maio de 2024.

AWS 1. **Computação em nuvem com AWS.** Disponível em: <https://aws.amazon.com/pt/what-is-aws/> Acesso em: 09 de maio de 2024.

AWS 2. **O que é SQL (linguagem de consulta estruturada).** Disponível em: <https://aws.amazon.com/pt/what-is/sql/#:~:text=A%20Linguagem%20de%20consulta%20estruturada%20(SQL)%20%C3%A9%20uma%20linguagem%20padr%C3%A3o,por%20atualiza%C3%A7%C3%B5es%20e%20melhorias%20cont%C3%ADnuas.> Acesso em: 10 de maio de 2024.

AWS 3. **O que é computação distribuída?** Disponível em: <https://aws.amazon.com/pt/what-is/distributed-computing/> Acesso em: 27 de setembro de 2024.

BENDER, E. M. **Climbing towards NLU: On Meaning Form, and Understanding in the Age of Data**. Disponível em: <https://aclanthology.org/2020.acl-main.463.pdf> Acesso em: 21 de maio de 2024.

BERALDI, L. C. **INTERNET E A PERFORMANCE ORGANIZACIONAL.** Disponível em: <https://img.fae.edu/galeria/getImage/1/16574999889705246.pdf> Acesso em: 08 de maio de 2024.

BESSA, A. **Java: o que é, linguagem e um Guia para iniciar na tecnologia**. Disponível em: <https://www.alura.com.br/artigos/java> Acesso em: 10 de maio de 2024.

BORELLI, F. **Busca por mão-de-obra é o maior desafio do empreendedor**. Disponível em: <https://ecoregional.com.br/destaque/busca-por-mao-de-obra-e-o-maior-desafio-do-empreendedor> Acesso em: 08 de março de 2024.

CASTIGLIONI, M. H. **Arquitetura em Camadas.** Disponível em: <https://imasters.com.br/arquitetura-da-informacao/arquitetura-em-camadas> Acesso em: 11 de setembro de 2024.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos: conceitos e projeto.** Bookman Companhia Ed., 2007.

DEVMEDIA. **Orientações básicas na elaboração de um diagrama de Classe.** Disponível em: <https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224> Acesso em: 10 de maio de 2024.

DIO. **Conheça a linguagem CSS e suas vantagens no front-end.** Disponível em: <https://www.dio.me/articles/conheca-a-linguagem-css-e-suas-vantagens-no-front-end> Acesso em:10 de maio de 2024.

DUARTE, R. D. **VISÃO GERAL DAS ESTRATÉGIAS DE MARKETING DAS GRANDES EMPRESAS CONTÁBEIS**. Disponível em: <https://www.robertodiasduarte.com.br/visao-geral-das-taticas-de-marketing-das-grandes-empresas-contabeis> Acesso em: 08 de março de 2024.

FAGGIÃO, D. F.; KESA, F. H.; GONZALEZ, I. V. D. P.; PELISSARI, A. S. **CONTROLE DE QUALIDADE E MARKETING DE SERVIÇOS: Estudo das ações de uma empresa especializada no ramo de retífica de motores junto a seus clientes**. Disponível em: <https://www.aedb.br/seget/arquivos/artigos12/47716570.pdf> Acesso em: 12 de abril de 2024.

FELIPIN, P.; CISLAGHI, T. P. **Comércio eletrônico B2C: estudo de caso em uma indústria vinícola da Serra Gaúcha.** Disponível em: <https://revistas.unilasalle.edu.br/index.php/desenvolve/article/view/10246> Acesso em: 01 de maio de 2024.

FOSTER, S. T.; WALLIN, C.; OGDEN, J. Towards a better understanding of supply chain quality management practices. **International Journal of Production Research,** v. 49, n. 8, p. 2285–2300, 2011.

GILLESPIE, T. **A Relevância dos algoritmos.** Disponível em: <https://revistaseletronicas.fiamfaam.br/index.php/recicofi/article/view/722/563> Acesso em: 11 de abril de 2024.

GOOGLE CLOUD 1. **O que é machine learning?** Disponível em: <https://cloud.google.com/learn/what-is-machine-learning?hl=pt-br> Acesso em: 21 de maio de 2024.

GOOGLE CLOUD 2. **O que é inteligência artificial (IA)?** Disponível em: <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=pt-br> Acesso em: 22 de maio de 2024.

GOV 1. **Lei Geral de Proteção de Dados Pessoais (LGPD).** Disponível em: <https://www.gov.br/esporte/pt-br/acesso-a-informacao/lgpd> Acesso em: 21 de maio de 2024.

GOV 2. **Princípios da LGPD.** Disponível em: <https://www.gov.br/esporte/pt-br/acesso-a-informacao/lgpd/principios-da-lgpd> Acesso em: 21 de maio de 2024.

HENNESSY, J. L.; PATTERSON, D. A. **Computer Architecture : A Quantitative Approach.** ELSEVIER INDIA, 2018.

IBM 1. **O que é processamento de linguagem natural (NLP)?** Disponível em: <https://www.ibm.com/br-pt/topics/natural-language-processing> Acesso em: 21 de maio de 2024.

IBM 2. **O que é aprendizado de máquina (ML)?** Disponível em: <https://www.ibm.com/br-pt/topics/machine-learning> Acesso em: 21 de maio de 2024.

IBM 3. **O que é deep learning?** Disponível em: <https://www.ibm.com/br-pt/topics/deep-learning> Acesso em: 21 de maio de 2024.

KAUFMAN, D. **Desmistificando a inteligência artificial.** Autêntica, 2022.

KENSKI, V. M. **Educação e tecnologias: Um novo ritmo da informação**. Papirus, 2012.

LOPER, A. A.; SILVA, N. S.; LOPES, G. M. G. **Projetos de redes e sistemas distribuídos.** Editora e Distribuidora Educacional, 2019.

MARQUES, I. S. **E-commerce no Mercado B2B: Soluções para o ponto de venda.** Disponível em: <https://ubibliorum.ubi.pt/bitstream/10400.6/10214/1/7161_15204.pdf> Acesso em: 01 de maio de 2024.

MARTINS, R. **Qual a importância da tecnologia e seus benefícios.** Disponível em: <https://www.grupoabl.com.br/post/qual-a-import%C3%A2ncia-da-tecnologia-e-seus-benef%C3%ADcios> Acesso em: 10 de abril de 2024.

MATA, A. S.; LIMA, E. C. S. **DESENVOLVIMENTO DE UMA APLICAÇÃO GAMIFICADA UTILIZANDO OS FRAMEWORKS ANGULAR E NEST.JS PARA AUXILIAR NO TRATAMENTO DE CRIANÇAS COM TDAH.** Disponível em: <https://periodicorease.pro.br/rease/article/view/12134/5597> Acesso em: 03 de maio de 2024.

MENDONÇA, H. G. **E-Commerce.** Disponível em: <https://periodicos.uninove.br/iptec/article/view/9361/pdf_49> Acesso em: 12 de abril de 2024.

MICROSOFT. **O que é DevOps?.** Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-devops> Acesso em: 08 de maio de 2024.

MONARD, M. C**. Conceitos sobre aprendizado de máquina.** Disponível em: <https://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf> Acesso em: 21 de maio de 2024.

MONCZKA;, R. M. et al. **Purchasing and Supply Chain Management.** 4. ed. United States of America: South-Western Cengage Learning, 2009.

MORAIS, M. C. A.; EMMENDOERFER, M. L.; VITÓRIA, J. R.; MENDES, W. A. **Determinantes socioeconômicos do microempreendedor individual (MEI)**. 2022. Disponível em: <https://regepe.org.br/regepe/article/view/2070/528> Acesso em: 12 de abril de 2024.

MOZILLA. **HTML: Linguagem de Marcação de Hipertexto.** Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML> Acesso em: 10 de maio de 2024.

OLIVEIRA, O. V.; FORTE, S. H. A. C. **Microempreendedor individual: Fatores da informalidade.** Disponível em: <https://repositorio.unp.br/index.php/connexio/article/view/800> Acesso em: 13 de abril de 2024.

ORACLE 1. **O que é um chatbot?** Disponível em: <https://www.oracle.com/br/chatbots/what-is-a-chatbot/> Acesso em: 12 de abril de 2024.

ORACLE 2. **O que é NoSQL?** Disponível em: <https://www.oracle.com/br/database/nosql/what-is-nosql/> Acesso em:10 de maio de 2024.

ORACLE 3. **O que é machine learning?** Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning/> Acesso em: 21 de maio de 2024.

ORACLE 4. **O que é deep learning?** Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-deep-learning/> Acesso em: 21 de maio de 2024.

PAIVA, D; VALENTE, E. S. O. **Um olhar semiótico para anúncios publicitários: estratégias persuasivas**. Disponível em: <https://www.fira.edu.br/repositorio/wp-content/uploads/tainacan-items/5/1487/DENISE-PAIVA-TCC-2SEM-FIRA-2021.pdf> Acesso em: 05 de abril de 2024.

PEREIRA, C. R. **Aplicações web real-time com Node.js**. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=Wm-CCwAAQBAJ&oi=fnd&pg=PT7&dq=node+js&ots=_fm-4rwqhL&sig=t17RiNtt_hnmPmWXyUFPsicnDeM&redir_esc=y#v=onepage&q=node%20js&f=false> Acesso em: 03 de maio de 2024.

PEREIRA, M. H. R. **AngularJS. – Uma abordagem prática e objetiva.** Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=MUelBQAAQBAJ&oi=fnd&pg=PA13&dq=angularjs&ots=qJ8H7ZCZzX&sig=lqFGSnEnTsC8y-ei8kgTbllyGpY&redir_esc=y#v=onepage&q=angularjs&f=false> Acesso em: 03 de maio de 2024.

SEBRAE. **Pequenos negócios: a base da economia do nosso país**. Disponível em: <https://sebrae.com.br/sites/PortalSebrae/artigos/pequenos-negocios-a-base-da-economia-do-nosso-pais,85e97325a3937810VgnVCM1000001b00320aRCRD> Acesso em: 07 de março de 2024.

RASHID, K.; ASLAN, H. Bussiness excellence througth total supply chain quality management. **Asian Journal on Quality**, v. 13, n. 3, p. 309–324, 2012.

REDHAT. **O que é deep learning?** Disponível em: <https://www.redhat.com/pt-br/topics/digital-transformation/what-is-deep-learning> Acesso em: 21 de maio de 2024.

SANCHES, M. F. **Processamento e entendimento de linguagem natural no gerenciamento de emergências para obtenção de consciência situacional.** Disponível em: <https://aberto.univem.edu.br/bitstream/handle/11077/1662/Matheus%20Ferraroni%20Sanches.pdf?sequence=1&isAllowed=y> Acesso em: 21 de maio de 2024.

SILVA, M. E. L. **Modelagem de ecossistemas de software das plataformas de computação em nuvem: Amazon Web Services e Google Cloud Platform.** Disponível em:

<https://repositorio.ufc.br/bitstream/riufc/70833/1/2022_tcc_meldasilva.pdf> Acesso em: 09 de maio de 2024.

SIMCHI-LEVI, D. **Cadeia de suprimentos: projeto e gestão.** Porto Alegre: Bookman, 2003.

SOUSA, L. S.; LOPES, P. L.; BARBOSA, M. V.; MOURA, R. G. **A tecnologia em prol do microempreendedor individual – MEI: Ferramentas digitais e suas funcionalidades.** Disponível em: <https://www.aedb.br/seget/arquivos/artigos19/8728328.pdf> Acesso em: 02 de maio de 2024.

TANENBAUM, A. S.; STEEN, M. V. **Systems Distributed: Principles and Paradigms.** Pearson Prentice Hall, 2006.

TAVARES, J. **A ciência e a tecnologia na sociedade.** Disponível em: <https://www.mindtek.com.br/2023/10/a-ciencia-e-tecnologia-na-sociedade/> Acesso em: 06 de abril de 2024.

TOMÉ, L. M. **Comércio eletrônico x Pandemia de coronavírus.** Disponível em: <https://www.bnb.gov.br/s482-dspace/bitstream/123456789/908/1/2021_CDS_178.pdf> Acesso em: 10 de abril de 2024.

VALENTE, M. T. **Engenharia de software moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.** Independente, 2020. Capítulo 3.

VERAS, M. **Arquitetura de Nuvem.** Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=RY7kp2dT0jwC&oi=fnd&pg=PA1&dq=aws+o+que+e&ots=jfW1vc8aXY&sig=r8fg4zBm_h6NGFQa8o_b0tK2ERs#v=onepage&q=aws%20o%20que%20e&f=false> Acesso em: 09 de maio de 2024.

VIEIRA, R. **Linguagens especializadas em corpora.** EDIPUCRS, 2010. Página 184 – 189.

WINOGRAD, T. **Understanding natural language.** Academic PR, 1972.

ZENG, J.; PHAN, C. A.; MATSUI, Y. Supply chain quality management practices and performance: An empirical study. **Operations Management Research**, v. 6, n. 1–2, p. 19–31, 2013.

ZORZENON, R. **Práticas de Gestão da Qualidade na Relação Cliente-Fornecedor em Produtos Eletrônicos.** Disponível em: <https://repositorio.ufscar.br/bitstream/handle/ufscar/11212/Dissertacao-DOS-20190228%20Final.pdf?sequence=2&isAllowed=y> Acesso em: 02 de maio de 2024.