



BreShow: Um Microsistema de ERP de Código Aberto para Pequenas e Médias Empresas do Setor de Brechó

Venâncio Pessoa Igrejas Lopes Neto

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Claudio Miceli de Farias

Rio de Janeiro

Maio de 2022

BreShow: Um Microsistema de ERP de Código Aberto para
Pequenas e Médias Empresas do Setor de Brechó

Venâncio Pessoa Igrejas Lopes Neto

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO
DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA
DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGEN-
HEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Venâncio Pessoa Igrejas Lopes Neto

Orientador:

Prof. Claudio Miceli de Farias, D. Sc.

Examinador:

Prof. Flávio Luis de Mello, D. Sc.

Examinador:

Prof. Toacy Cavalcante de Oliveira, D. E.

Rio de Janeiro

Maio de 2022

Declaração de Autoria e de Direitos

Eu, *Venâncio Pessoa Igrejas Lopes Neto* CPF 145.689.997-07, autor da monografia *BreShow: Um Microsistema de ERP de Código Aberto para Pequenas e Médias Empresas do Setor de Brechó*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetua-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.

Venâncio Pessoa Igrejas Lopes Neto

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica - Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

Dedico este trabalho aos meus pais, ao meu orientador e toda população brasileira. Aos meus pais por toda a educação e ensinamentos impecáveis que me deram e dão até hoje, a população brasileira por me proporcionar a possibilidade de fazer um curso de nível superior de alta qualidade e ter investido em mim e nos meus esforços na minha vida acadêmica. Por ultimo e não menos importante, ao meu orientador que sem ele não teria conseguido concluir esta difícil tarefa.

AGRADECIMENTOS

Por mais que este trabalho tenha sido feito por uma pessoa, não teria conseguido concluir sem a ajuda de inúmeras pessoas que me deram apoio ao longo de toda a minha vida. Tentarei mencionar todos nessa sessão sabendo que irei falhar, pois são muitos. Dessa forma, já agradeço de antemão a quem não mencionei aqui por tudo que fez por mim. Saibam que se eu cheguei aqui, é porque vocês tiveram um papel fundamental.

Primeiramente, gostaria de agradecer aos meus pais Jussara e Adalberto, minhas irmãs Júlia e Luísa e minha namorada Larissa por tudo que fizeram por mim todos esses anos e pelo apoio de vocês na minha vida, que tem sido de extrema importância para ter chegado onde cheguei.

Aos meus professores por todo carinho, dedicação e paciência que tiveram ao me ensinar. Em especial, ao professor José Guilherme que me ensinou a maneira correta de se estudar no ensino médio, a professora de francês Ana Sílvia pelo aprendizado e conselhos, aos professores do Colégio Ponto de Ensino e do preparatório militar que me deram uma ótima base para a faculdade e aos professores universitários da UFRJ pelo conhecimento transmitido.

Agradeço as pessoas que influenciaram diretamente, ou indiretamente, nas minhas iniciações científicas tanto no laboratório do Grupo de Telecomunicações e Automação (GTA), como no Laboratório de Processamento de Sinais (LPS). Primeiramente, agradeço ao professor Otto Carlos Muniz Bandeira Duarte, que infelizmente nos deixou recentemente, e ao professor José Manoel de Seixas, os quais tive o imenso prazer de tê-los como orientadores durante o período de iniciação científica. Agradeço ao Lucas Airam e Lucas Santiago pelo apoio e por tudo que passamos no GTA e também ao Nathanael Moura Júnior, Gustavo Augusto Mascarenhas Goltz, Pedro Lisboa e Vinícius Mello pela força e ajuda nos conhecimentos práticos e teóricos de aprendizagem de máquina no LPS.

Ao meu orientador, Cláudio Miceli de Farias que foi presente e solícito na redação deste trabalho, assim como nas nossas reuniões sobre o mesmo.

Por ultimo mas não menos importante, gostaria de agradecer aos meus amigos e colegas de curso, sem eles não estaria me formando agora e não teria superado os momentos mais difíceis e complicados da vida acadêmica. Em especial, Lucas Fontes, Yuri Pereira, Daniel Porto, Camyla Romão, Letícia Ecard, Andrews Monzato, Brenno Rodrigues, Leonardo Barreto e Lucas Lessa pelo apoio, estudos em grupo e inúmeros trabalhos que fizemos ao longo dos períodos.

RESUMO

Com a popularização da internet e serviços a partir dela, o mercado de ERPs se modernizou em poucos anos e se tornou mais acessível, entretanto, custo e complexidade do sistema ainda podem ser um obstáculo para algumas categorias de empreendedores. Dessa forma, esse trabalho tem por objetivo o desenvolvimento de um ERP voltado para a análise de vendas onde, inicialmente, é focado para o setor de brechós com o intuito de auxiliar o empreendedor nas suas tomadas de decisões. O projeto é de código aberto e de fácil customização, onde em uma tela é feita o cadastro das vendas pelo usuário e da outra, é gerado a análise macro das informações das vendas filtradas por um período de data. A sua arquitetura e sistemas utilizados na integração são populares no mercado e também de código aberto, tornando-o de baixo custo e assim, viável para pequenos e médios empreendedores.

Palavras-Chave: ERP, Brechó, Aplicação Web, Arquitetura de Software, Docker, Grafana, Kecloak, NestJs, Angular

ABSTRACT

With the popularization of the internet and services from it, the market is becoming more and more competitive. Thus, this work aims to develop a web system aimed at sales analysis where, initially, it is aimed at the thrift store sector in order to assist the entrepreneur in its decision-making. The project is open source and easy to customize, where the user registers sales on one screen, and on the other, it is a macro for analyzing sales information filtered by a period of data. Its architecture and systems used in the integration are popular and also open source, making it low cost and thus viable for small and enterprising entrepreneurs.

Keywords: ERP, Thrift shop, Web application, Software architecture, Docker, Grafana, Kecloak, NestJs, Angular

Índice

Índice de Figuras	xi
1 Introdução	1
1.1 Motivação	2
1.2 Problema	3
1.3 Contribuição	5
1.4 Organização do Texto	5
2 Conceitos Básicos	7
2.1 Princípios SOLID	7
2.2 Arquitetura MVC	8
2.3 Clean Architecture	9
2.4 Folder-by-Feature	11
2.5 ERP	12
3 Trabalhos Relacionados	13
4 Proposta	17
4.1 Arquitetura de software	17
4.2 Módulos	18
4.2.1 Aplicação Web	19
4.2.2 Dashboard	21
4.2.3 Gerenciamento de Usuários	22
5 Implementação	26
5.1 Arquitetura	26
5.1.1 Orquestrador	28

5.1.2	Banco de dados	29
5.2	Aplicações	31
5.2.1	Gerenciador de usuários	31
5.2.2	Aplicação Web	34
5.2.3	Análise dos dados	41
5.3	Desafios	44
6	Prova de Conceito	47
6.1	Implantação	47
6.1.1	Docker	47
6.1.2	Github	48
6.1.3	Npm	48
6.1.4	Projeto Local	48
6.2	Usuário	49
6.2.1	Criação de Conta	49
6.2.2	Tela principal	49
6.2.3	Tela de Categoria	50
6.2.4	Tela de Fornecedor	53
6.2.5	Tela de Produtos	54
6.2.6	Tela de Dashboard	57
6.3	Administrador	59
6.3.1	Tela de Gerenciamento de Usuários	59
6.3.2	Tela de Dashboard	62
7	Conclusões e Trabalhos Futuros	64
7.1	Conclusão	64
7.2	Trabalhos Futuros	65
	Bibliography	66

Índice de Figuras

1.1	Exemplo do marketing da empresa TOTVS sobre o seu serviço de ERP	3
1.2	Exemplo dos valores do plano da empresa Bling sobre a sua plataforma	4
1.3	Exemplo dos valores dos micro serviços proporcionado pela Odoo . . .	4
2.1	modelo Clean Architecture desenvolvido por Uncle Bob.[1]	10
3.1	Quadro comparativo entre o Odoo(OpenERP) e Xtuple.[2]	14
3.2	Quadro comparativo entre o Odoo, MyCollab, OrangeScrum e Open Project.[3]	16
4.1	Diagrama de Venn dos Módulos do projeto	19
4.2	Fluxograma da tela de produtos	20
4.3	Fluxograma da tela de fornecedor	21
4.4	Fluxograma da tela de categorias	22
4.5	Caso de uso do módulo de dashboard	23
4.6	Lógica do login e aquisição dos tokens para a aplicação	24
5.1	Arquitetura macro do projeto separada por contêiner	27
5.2	Diagrama entidade relacionamento do banco de dados da aplicação .	30
5.3	Tela do cliets do realm BreshowAD do Keycloak	33
5.4	Estrutura de pastas do frontend da aplicação web	36
5.5	Estrutura da pasta “Core” do frontend da aplicação web	37
5.6	Arquivos relacionados as configurações do keycloak no backend	38
5.7	Estrutura da entidade de categoria do servidor	40
5.8	Código da regra de negócio de atualização de uma entidade de venda que está no arquivo “product.service.ts”	40
5.9	Painel de valores do dashboard de um usuário no Grafana	42

5.10	Painel de valores em barra do dashboard de um usuário no Grafana .	43
5.11	Painel de valores em estatística do dashboard de um usuário no Grafana	44
5.12	Painel de valores por fornecedor do dashboard de um usuário no Grafana	45
5.13	Dashboard completo das vendas de um usuário no Grafana	46
5.14	Aba de configuração de um dashboard no Grafana	46
6.1	Página para entrar na aplicação web.	50
6.2	Página de cadastro da aplicação web.	51
6.3	Página principal da aplicação web.	52
6.4	Página da categoria da aplicação web.	52
6.5	Janela flutuante da categoria.	53
6.6	Tabela da categoria listando a Blusa após ter sido criada.	53
6.7	Página dos fornecedores na aplicação web.	53
6.8	Janela flutuante da tela de fornecedores.	54
6.9	Tabela de fornecedores listando o novo fornecedor após ter sido criado.	55
6.10	Página dos produtos na aplicação web.	55
6.11	Janela flutuante da tela de produtos.	56
6.12	Tabela de produtos listando o novo produto após ter sido criado. . . .	56
6.13	tela principal listando todas as informações adicionadas do usuário em tabelas distintas.	57
6.14	Tela de login do módulo de análise.	58
6.15	Tela principal do usuário.	58
6.16	Painel de listas gerais expandido.	59
6.17	Painel de fornecedores e categorias expandido.	59
6.18	Painel de fornecedores e categorias expandido.	60
6.19	Página Inicial do Keycloak.	60
6.20	Página de login da área de administração.	61
6.21	Página Principal do Keycloak.	61
6.22	Tela de gerenciamento do usuário Venâncio.	62
6.23	Página de credenciais do usuário Venâncio.	62
6.24	Área de gerenciamento de usuários do módulo de dashboard.	63
6.25	Área de configuração	63

Capítulo 1

Introdução

O Planejamento de Recursos Empresariais, conhecido pela sigla em inglês ERP, tem sido implementado a anos em empresas de grande e médio porte melhorando a integração dos sistemas de informação, além de outros benefícios. Entretanto, a maioria dos médios e pequenos empreendedores consideram o custo como principal fator na implantação de um sistema ERP e se mostram relutantes em investir em melhorias após certo tempo de uso. Além disso, alguns desses sistemas não atendem à essas categorias de empreendedores devido as suas complexidades e funcionalidades, que acabam não sendo utilizadas.

Dessa forma, o seguinte trabalho tem como intuito a criação de um sistema ERP a partir de um projeto web de análise de vendas para auxiliar o microempreendedor ou empreendedor informal a compreender melhor suas vendas e que possa ajudá-lo a tomar melhores decisões baseadas em estatísticas e gráficos apresentados pelo sistema. Nele, consta micro serviços para gerenciamento de usuários, painéis dos dados e telas de cadastros das informações referente ao empreendimento do consumidor. Como o projeto é de código aberto, o individuo não precisará investir em mensalidades ou ter apenas alguns dias para teste antes de precisar investir capital no produto, como são feitos em sistemas semelhantes no mercado atual. Basta baixar o código fonte, seguir o tutorial de instalação e rodar o sistema.

Ao longo do texto será explicitado sobre as funcionalidades, arquiteturas e teorias por trás da aplicação. Assim como, motivações, problemas e trabalhos futuros.

1.1 Motivação

Com o avanço da tecnologia, produtos e serviços que um dia apenas empresas de grande porte tinham acesso, hoje em dia estão presentes no cotidiano. Banda larga, internet via Wi-fi, smartphones com poder de processamento maiores que computadores básicos de 15 anos atrás e inteligência artificial(IA) em produção são alguns dos exemplos de tecnologias presentes que facilitam a forma de viver e trabalhar.

Todo esse avanço obrigou as empresas a se tornarem mais complexas e robustas de ponta a ponta, afinal os sistemas de informação se tornaram mais eficientes, reduziram os custos e houveram uma melhora da logística, dessa forma impactando diretamente na competitividade do mercado. Um exemplo de sistema de informação que pode ajudar as companhias a aprimorarem a eficiência de suas performances seria o ERP, afinal, Esse sistema é pode integrar informações da companhia de vários aspectos, como financeiro, time, tempo, materiais, capacidade e outros [4].

Os ERPs são ferramentas estratégicas que sincroniza e integra sistemas de uma empresa para gerenciar de forma automática recursos, estoques, financeiro e informações de vendas, assim agilizando na resposta e entrega de seus serviços aos clientes e procurando minimizar qualquer barreira entre os setores da companhia. Com a melhora na eficiência da internet e a popularização da computação em nuvem, empresas desenvolveram Erps como Plataforma Como Serviço (PaaS)[5] para disponibilizar-los como Software Como Serviço (SaaS)[6] para seus clientes e com isso, não havendo a necessidade de instalar hardware físicos na companhia do cliente e nem em suas próprias localidades, dado que em sua grande maioria, os servidores são hospedados na nuvem e assim, podendo tornar o preço mais competitivo no mercado. Na Figura 1.1, podemos observar um exemplo de marketing da empresa TOTVS vendendo um Saas de ERP.

Devido aos fatores apontados acima, torna-se importante o uso desses sistemas para todo tipo de empreendedor ou companhia por conta da visibilidade que os gestores terão da empresa como um todo para tomada de decisões e um controle maior delas.



Figura 1.1: Exemplo do marketing da empresa TOTVS sobre o seu serviço de ERP

1.2 Problema

O ERP se tornou uma ferramenta importante e fundamental no mercado, contudo empresas que disponibilizam esse serviço para seus clientes cobram valores pelo serviço que, em determinadas situações, microempreendedores não estão dispostos ou não tem condições de contratar o serviço. O valor de cobrança e a forma varia de empresa para empresa, por exemplo, a TOTVS tem um central de vendas e a empresa entra em contado com o cliente, a Senior envia uma proposta através do email e a Bling, empresa conhecida no ramo e com clientes grandes, apresenta planos que começam com o valor de 25 reais e vão até 100 reais por mês de acordo com a necessidade do cliente, lembrando que eles diferem nos recursos oferecidos pela Bling e existe um período gratuito para testar a plataforma conforme podemos observar na figura 1.2.

Outro ponto a ser levantado é o fato do cliente pagar o valor fixo de todo o pacote de benefícios e não apenas o que ele irá utilizar, existem poucos serviços de ERPs que disponibilizam o usuário a escolher quais micro serviços ele quer para a sua empresa e pagar por eles separadamente, um exemplo é a empresa Odoo que faz esse tipo de transação conforme explicitado na figura 1.3, além de cobrar por usuário e por ser em dólar. A cobrança pelo pacote inteiro se torna ineficiente para o cliente pois parte do pré-suposto que será utilizada todas as funcionalidades e sistemas do software

	Plano 1	Plano 2	Plano 3	Plano 4
Finanças	✓	✓	✓	✓
Boleto Registrado	✓	✓	✓	✓
Conta Digital	✓	✓	✓	✓
Indústria	✓	✓	✗	✗
Recursos	✓	✓	✓	✓
Suporte	✓	✓	✓	✓
Arquivo auxiliar do SPED	✓	✗	✗	✗
Espaço	180MB	120MB	60MB	20MB
Usuários	15 usuários	10 usuários	5 usuários	2 usuários
Preço	R\$ 100,00 Por mês	R\$ 75,00 Por mês	R\$ 50,00 Por mês	R\$ 25,00 Por mês
Ação	teste grátis	teste grátis	teste grátis	teste grátis

Figura 1.2: Exemplo dos valores do plano da empresa Bling sobre a sua plataforma

vendido. Por outro lado, a cobrança por microserviço pode encarecer o produto final dependendo do tamanho da empresa e do numero de funcionário. A solução seria a avaliação de varias plataformas no mercado para encontrar uma que seja mais adequada. Contudo, continuamos com o problema de precisarmos investir um valor na plataforma e termos avaliação gratuita temporária pelo uso da plataforma.

Escolha o número de Usuários		Escolha seus Aplicativos		Resumo	
1	Usuários	CRM	Faturamento	1 Usuários	\$8.00 USD
		Website	Comércio Eletrônico	Desconto de usuário ⁽¹⁾	-\$2.00 USD
		Contabilidade	Projeto	0 Aplicativos	\$0.00 USD
		Fabricação	Compras	Total / mês ⁽²⁾	\$6.00 USD
		Marketing por email	Despesas	⁽¹⁾ Cobrado anualmente: \$72.00 USD	
			Eventos	EXPERIMENTE AGORA 15 dias teste grátis	
				COMPRE AGORA	
				⁽²⁾ Novos clientes obtém um desconto no número inicial de usuários adquiridos. (\$6.00 USD em vez de \$8.00 USD).	

Figura 1.3: Exemplo dos valores dos micro serviços proporcionado pela Odoo

Ainda na atualidade, existem microempreendedores e empreendedores informais que utilizam ferramentas básicas para gerenciar seus negócios, como por exemplo, o Excel, ferramenta de tabelas da Microsoft, ou cadernos usados para estudo ou de colégio. Essas ferramentas ajudam no dia a dia, porém para fazer uma análises a

longo prazo das estatísticas do negocio ou quais decisões tomarem, elas não são tão úteis e requer um esforço grande para analisar todos os dados que constam neles.

1.3 Contribuição

A contribuição deste trabalho consiste no desenvolvimento de um projeto de um modelo de ERP que seja de Código Aberto, disponível gratuitamente, modularizável e escalável. Por conta disso, utilizamos linguagens e sistema de repositório aberto, tais como, Keycloak, Composição de Docker, NestJs, Angular, Grafana e Postgres. Além disso, contamos com o uso de teorias de boas práticas de desenvolvimento de software, como os princípios SOLID, Clean Architecture, arquitetura MVC e estrutura Folder-by-Feature para facilitar a sua manutenção, dentre outras vantagens.

O projeto consiste apenas no módulo de vendas do erp, o que significa que contém um micro serviço para cadastro dos produtos vendidos e informações adicionais sobre ele, além de um micro serviço de painel de controle de série temporal para análise do macro das vendas. Além disso, pelo fato de ser escalável e modularizável, pode ser ajustado para outro seguimento com apenas algumas alterações no código fonte. O nome do projeto foi dado de BreShow e está disponível publicamente no seguinte link do GitHub para clonar e instalar: <https://github.com/VenancioIgrejas/BreShow>

Na pagina do projeto no Github contem informações de instalação e requisições do hardware, basta rodar a composição de Docker para levantar os micro serviços e após terminarem as configurações automaticamente, estarão disponíveis no navegador nos links informados no guia do projeto.

1.4 Organização do Texto

O seguinte trabalho foi separado em sete capítulos. O primeiro consiste na introdução, onde discorreremos sobre a motivação e contribuição do projeto desenvolvido e quais são os problemas que nos impulsionaram a criar esse sistema. O

segundo capítulo é voltado para os conhecimentos básicos, nele iremos apresentar conceitos de arquiteturas e boas práticas de programação utilizados no trabalho.

O terceiro capítulo aborda sobre trabalhos relacionados na literatura e o quarto consiste na proposta do projeto, onde fala-se sobre a estrutura do sistema e as divisões dos seus módulos. O quinto é sobre a sua implementação, tais como, quais frameworks usamos em seus módulos, como o código está organizado e as informações de sua estrutura. O sexto diz respeito a prova de conceito e é nele que mostramos como funciona o sistema, tanto pelo lado do usuário quanto pro administrador. Por ultimo, no sétimo capítulo, temos a conclusão, onde descrevemos como foi o processo do desenvolvimento desse trabalho, tanto na parte estrutural e conhecimento quanto na administração do tempo e problemas. Para fechar, discorreremos sobre trabalhos futuros e melhorias do projeto.

Capítulo 2

Conceitos Básicos

Neste capítulo serão discutidos, apresentados e conceituados teorias utilizadas neste trabalho com foco em aplicar técnicas de arquiteturas de software para tornar o código robusto, simplificação da manutenção a media que o torna mais complexo, independência de ferramentas periféricas e outras vantagens.

2.1 Princípios SOLID

O termo SOLID representa um acrônimo para os cinco primeiros princípios de designer de orientação a objeto desenvolvido pelo Robert C. Martin, conhecido como Uncle Bob.[7] Esses princípios estabelecem praticas que ajudam o programador a ter códigos mais transparentes, organizados, facilitam a manutenção ou evolução a medida que cresce o programa e a sua leitura para outros desenvolvedores.[8]

A letra S do SOLID vem de Principio da Responsabilidade Única (*Single Responsibility Principle*). Esse principio consiste que toda classe ou estrutura similar deve haver apenas um proposito, não deve ser uma canivete suíço onde ,em apenas um lugar, temos várias funções. Em outras palavras, uma classe deve ter métodos ou variáveis relacionadas com o seu propósito apenas e não uma única classe para todos os métodos do projeto, por exemplo.

A letra O vem de Princípio Aberto-Fechado (*Open-Close Principal*) e ele está relacionado aos objetos ou entidades estarem abertos para extensões, mas fechados para modificações. Esse principio significa que as classes bases ou já existentes não

devem ser mudadas quando a regra de negocio muda, mas sim, estendidas em classes novas.

Princípio da substituição de Liskov (*Liskov Substitution Principal*) é representado pela letra L no SOLID e sua definição formal introduzida por Barbara Liskov diz que: “Se para cada objeto o1 do tipo S há um objeto o2 do tipo T de forma que, para todos os programas P definidos em termos de T, o comportamento de P é inalterado quando o1 é substituído por o2 então S é um subtipo de T”. Em outras palavras, qualquer classe derivada de uma classe pai deve ter os mesmos comportamentos que ela sem a necessidade de alterá-la. Se há a necessidade, é porque a classe filha não deve ser herdada daquela classe pai.

A letra I representa o Princípio da Segregação da interface (*Interface Segregation Principle*) onde diz que as classes não deveriam ser forçadas a implementar interfaces que não serão utilizadas. Esse princípio nos diz que para não criar uma interface genérica para todas as classes que iremos criar, mas sim, para aplicarmos os métodos de herança quando necessário nas interfaces para segregarmos o máximo possível.

Por ultimo e não menos importante, o Princípio da Inversão de Dependência (*Dependency Inversion Principle*) representado pela letra D no SOLID diz que o código deve depender das suas abstrações e não das implementações, ou seja, deve-se, por exemplo, ser possível alterar o sistema do banco de dados sem precisar alterar a estrutura do código do projeto e isso é feito utilizando abstrações como as interfaces.

2.2 Arquitetura MVC

A sigla MVC significa *Model-View-Controller* que em português seria Modelo-Visualização-Controle. Essas 3 palavras significam as camadas onde o projeto será dividido e cada uma tem um significado ou papel importante e específico dentro do código.[11]

O padrão de arquitetura MVC é utilizado em muitos projetos por possuir benefícios como o isolamento das camadas, onde uma não interfere na outra. Ou seja, por ex-

emplo, o código da regra de negocio que está presente na camada do modelo não interfere no código da interface com o usuário que está na de visualização. Esse isolamento proporciona flexibilidade e reuso de classes, além de permitir que mais de um desenvolvedor trabalhe ao mesmo tempo no projeto e facilitando na manutenção ou adição de recursos.

A camada de controle serve para interpretar algum evento disparado pelo usuário na tela como algum dispositivo de entrada e assim, as mapeia em comandos que são enviados para a camada de modelo. Após o processamento dos dados, por esse mesmo fluxo, o controle envia a informação para o usuário que é informado na tela pela camada de visualização.

por sua vez, a camada de visualização é responsável por apresentar as informações oriundas das outras camadas na tela do usuário na forma de texto, gráfico, tabelas ou listas. Essa camada é apenas uma carcaça que ajeita os dados entregues a ela da melhor forma para o usuário as compreendê-las.

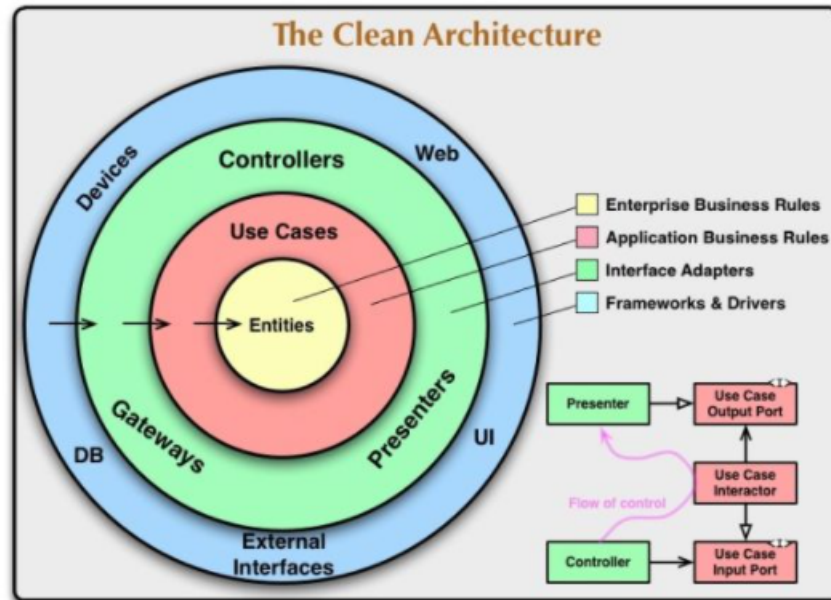
Por ultimo e não menos importante, o modelo gerencia um ou mais elementos de dados, é nele que são processados os dados tanto oriundos da camada de controle quanto do banco de dados. Dizemos que é nessa camada que as regras de negócio do projeto são desenvolvidas e por isso é considerada o coração da arquitetura, afinal é ela quem modela o problema a ser resolvido.

2.3 Clean Architecture

A Clean Architecture foi desenvolvida pelo Uncle Bob no ano de 2012 com a proposta de unificar as teorias de outras arquiteturas famosas da época, uma delas é a Onion Architecture desenvolvida por Jeffrey Palermo. Essa arquitetura tem por finalidade tornar o projeto mais independente dos frameworks, das interfaces dos usuário, banco de dados e de qualquer agentes externos.[14]

Na figura 2.1, temos um esquemático de como a arquitetura funciona na teoria. Os anéis representam diferentes áreas de softwares e as setas, as dependências entre

as camadas. Ou seja, por exemplo, a camada de “frameworks e drives” apenas enxerga a camada de “adaptadores de interface” e assim não tendo o conhecimento de camadas mais internas da arquitetura. De acordo com Uncle Bob, quanto mais próximo a camada estiver do centro maior será a sua abstração em relação ao projeto.



Fonte: <https://bit.ly/2a1px8t>

Figura 2.1: modelo Clean Architecture desenvolvido por Uncle Bob.[1]

A principal regra da Clean Architecture é o que podemos chamar de Princípio da Dependência, onde as camadas de dentro pra fora não devem apresentar nenhuma dependência externa, nem de forma indireta. Ficando assim, mais isoladas quanto maior for a sua abstração.

A camada “Entities” abrange as entidades do sistema e algumas regras de negócio da aplicação, é nela que são presentes as regras que terão menor probabilidade de mudança e são mais próximas das regras o qual o projeto fora feito. Podemos associa-la a interfaces e classes que apresentam uma representação do núcleo do negócio e eles são provavelmente a ultima coisa a ser alterada após uma modificação na parte externa da aplicação.

A camada “Use Cases” apresenta as regras de negócio mais específicas do sistema e é onde implementam todos os casos de usos da aplicação e o processamento dos

dados. Espera-se que apenas mudanças de requisitos do sistema afetem essa camada, ou seja, mudanças externas não influencie ela, como banco de dados, interface do usuário e frameworks.

A “Interface Adapters” são softwares que servem para serem intermediadores entre as camadas mais internas, como Entities e Use Case, e as externas. Nela são apresentados adaptadores que convertem os dados externos para se adequarem ao formato de dados das camadas mais internas e vice-versa. Um exemplo disso seria um AutoMapper, framework de ORM e Controles.

A camada mais externa conhecida pelo nome de “Frameworks” consiste em ferramentas como banco de dados, dispositivos de consultas ao projeto, interfaces externas e as telas. Normalmente, ocorre pouca alteração no código deles, apenas para interligar as suas camadas com a próxima mais interna.

A Clean Architecture tem os seus lados positivos e negativos, contudo sua ideia de arquitetura torna o código mais desacoplado e menos preso a determinadas tecnologias, afinal elas estão sempre em constantes mudanças e atualizações.

2.4 Folder-by-Feature

A estrutura “Folder-by-Feature” consiste em agrupar os arquivos de um projeto MVC em pastas cujo conteúdo apresenta as mesmas características. Em outras palavras, por exemplo, uma entidade chamada “Usuário” terá o seu arquivo de controle, de negocio, de transferência de objeto e sua entidade em si dentro de uma pasta chamada “Usuário”. [16]

Esse padrão é contrário ao utilizado normalmente por aplicações onde são separados as pastas pelo tipo de responsabilidade. Ou seja, por exemplo, a pasta “controller” de uma aplicação terão todos os controles das entidade do projeto. A priori, essa estrutura parece fazer mais sentido pra organizar e juntar arquivos que estão divididos pelas camadas MVC. Entretanto com o aumento do projeto e o número de arquivos e entidades, o desenvolvedor se vê tendo que procurar dentro

de poucas pastas, poucos arquivos relacionados aquela entidade que provavelmente estejam dentro de pastas com inúmeros outros arquivos que não são de interesse do indivíduo no momento.

O fato de utilizar esse padrão facilita a manutenção e evolução do código, pois todas as informações e características de uma entidade se encontram dentro de uma mesma pasta. Além disso, torna viável o desenvolvimento com múltiplos desenvolvedores trabalhando ao mesmo tempo, afinal cada um pode ser responsável por uma entidade e não por uma camada. Vale ressaltar que essa estrutura tem os seus lados negativos e nesse caso, ela é recomendada para projetos que possam escalar muito rápido de tamanho ou projetos que já são grandes desde o início.

2.5 ERP

A sigla ERP significa Enterprise Resource Planning que traduzido para o português se torna “Planejamento dos Recursos Empresariais”. Dessa forma, esse sistema consiste de um software empresarial que serve para o gerenciamento automático dos recursos, estoques, financeiro, informações de vendas e outros setores dentro da empresa. Logo, o ERP passa a ter um papel fundamental funcionando como um centralizador de fluxo de trabalho, afinal cada departamento, normalmente, necessita de uma ferramenta própria para poder exercer as tarefas e gerar relatórios.[17]

O ERP concentra as informações de forma inteligente e sem redundâncias, o que dá a empresas a autonomia na execução de tarefas repetitivas e interligando a comunicação entre as áreas. Assim, evitando que cada setor tenha que possuir um software isolado e com dados redundantes de outros sistemas, podendo ocorrer a inconsistência de informações dentro das empresas. Por conta disso, A implementação de uma ferramenta integrada é garantia de informações assertivas e sólidas para a empresa.

Capítulo 3

Trabalhos Relacionados

Por conta do vasto interesse em ERPs, artigos relacionados a estudos de caso sobre funcionamento, implementação e análises vem sendo publicados extensivamente. Contudo, poucos artigos acadêmicos são voltados para um sistema de código aberto de ERPs e em contra partida, há um numero maior de publicações no meio não acadêmico sobre o assunto. Esse fato nos revela uma lacuna entre a preocupação da academia e a demanda empresarial a respeito sobre esse assunto.

Trabalhos que visão micro, pequenas e médias empresas (MPME) exploram o uso de sistemas de erps que são de código aberto por conta do seu baixo custo e customização comparado a sistemas robustos e solidificados no mercado como SAP, Oracle e dentre outros [18]. Além disso, a implementação de um sistema desse traz benefícios para a empresa como redução de custos, aumento nas colaborações, análises da saúde da empresa de forma mais precisa e melhorada, aumento da produtividade, satisfação do cliente, eficiências nos negócios e assim, permite ampliação do seu mercado.

Existem muitos ERPs de código aberto(CA) no mercado, tais como, webERP [20], Compiere [21], Jfire [22], ERP5 [23], Odoo [24], Front Accounting [25] e etc. Esses ERPs apresentam funcionalidades similares como setor de compra, vendas, controle de estoque, gestão de produção, gestão de relacionamento com o cliente(CMR) e outros, mas diferem em pequenos pontos que são levados em conta na hora de escolher qual sistema se enquadra nas necessidades do empreendimento ou empresa como, por exemplo, a OFBIz [26] apresenta a funcionalidade de e-commerce que

a ADEMPIRE ERP não apresenta, O JPIRE [27] já suporta o volume de dados de empresas de todos os portes dado a sua flexibilidade e abrangência, algo que a XTUPLE [28] não abrange pois ela é projetada apenas para empresas de médio e pequeno porte, em contra partida, apresenta adaptação para a moeda brasileira em suas funcionalidades.

Trabalhos mais recentes veem utilizando o sistema Odoo nas implementações e casos de estudo por apresentarem resultados positivos a mais que outros concorrentes. Por exemplo, os sistemas Odoo e Xtuple apresentam programas traduzidos no idioma português diferentemente de seus concorrentes, mas o Odoo tem mais funcionalidades, como adaptação para impostos e alíquotas brasileiras e possibilidade de emissão de nota fiscal eletrônica, que o Xtuple não contém (Figura 3.1).

Outro caso está presente na Figura 3.2, onde há a comparação entre 4 sistemas de CA ERPs e apenas o Odoo apresenta todos os requisitos impostos e necessários no trabalho.

REQUISITOS	SOFTWARE	
	OpenERP	XTuple
Acesso para <i>download</i> pela internet	✓	✓
Ter a possibilidade de rodar o sistema em interface Linux	✓	✓
Possuir banco de dados gratuito	✓	✓
Possuir tutoriais de instalação disponibilizados na internet	✓	✓
Possuir tutoriais de implantação e operação disponíveis na internet	✓	✓
Possuir comunidades de discussão sobre o sistema	✓	✓
Possuir adaptação para moeda brasileira	✓	✓
Possuir adaptação para impostos e alíquotas brasileiras	✓	
Possuir possibilidade de emissão de nota fiscal eletrônica	✓	

Figura 3.1: Quadro comparativo entre o Odoo(OpenERP) e Xtuple.[2]

O sistema Odoo foi desenvolvido em 2005 com o nome de TinyERP e após três anos, mudou para OpenErp por ter sido incorporado e reconhecido por grandes empresas. Logo em seguida, após a adição dos módulos CRM, Website e E-commerce, a empresa adotou o nome de Odoo para o seu sistema e em 2015, entrou para a lista das companhias que mais crescem na Europa. A aplicação é capaz de automatizar quase toda uma empresa de forma performática, cobrindo quase toda as suas necessidades e integrando processo empresarial. Ele é dividido em módulos e aplicativos

que vão desde faturamento, manufatura até gerenciamento de projeto e gestão de armazenamento. A sua versão gratuita está disponível na Odoo Community tanto para baixar através de Download quanto versão de Docker, nele apresenta uma gama grande de módulos de ERP, assim como, o uso de um banco de dados de CA gratuito também. Existe a possibilidade de migrar para a versão paga, que dá direto a suporte e todos os módulos de ERPs existentes na aplicação.

Dessa forma, os sistemas de ERPs atuais apresentam um modo gratuito, porém alguns são limitados a acessibilidades da plataforma, como o caso do Oddo, e outros estão desatualizados em relação ao designer, serviços ou tecnologia. Assim, o seguinte trabalho tem por contribuição o desenvolvimento de um ERP totalmente gratuito e de código aberto voltado primeiramente para o segmento de brechós como prova de conceito, no qual apresenta o módulo de gerenciamento de usuário, de vendas e de análise, pois esses módulos apresentam uma importância primária para esse setor.

	Ferramentas de Gestão de Projetos	Odoo	MyCollab	OrangeScrum	Open Project
	Requisitos presentes na literatura				
1	Fiabilidade	X	X		
2	Performance	X			
3	Usabilidade	X			
4	Extensibilidade	X			
5	Comunidade – existência de suporte técnico	X	X	X	X
6	Conhecimentos da equipa informática	X			
7	Capacidades de manutenção a longo prazo	X			
8	Atualização funcional futura	X			
9	Tipo de licença – licença OS	X	X	X	X
	Requisitos apresentados pelos colaboradores				
10	Acesso a informação detalhada do projeto	X	X	X	
11	Definição datas de entrega e prioridades	X	X	X	X
12	Definição e atribuição de tarefas a executar em cada projeto	X	X	X	X
13	Alocação de gestor a cada projeto	X	X	X	
14	Criação base de dados de clientes	X			
15	Gestão de agendamento de reuniões	X		X	X
16	Diferentes níveis de acesso à informação	X	X		
17	Notificação de alterações	X	X	X	X
18	Notificação de início de tarefas	X	X	X	X
19	Avaliação do estado do projeto	X	X	X	X
20	Anexar documentos	X	X	X	X
21	Visualização de histórico de alterações	X	X		
22	Integração com <i>Microsoft Outlook</i>	X	X	X	X
23	Facilidade de uso	X		X	
24	Flexibilidade	X			
25	Mensagens instantâneas	X	X		

Figura 3.2: Quadro comparativo entre o Odoo, MyCollab, OrangeScrum e Open Project.[3]

Capítulo 4

Proposta

A proposta do trabalho consiste no desenvolvimento de um sistema de ERP de código aberto e gratuito disponibilizado na plataforma github voltado para o seguimento de brechós como prova de conceito. O motivo se dá pelo fato de, atualmente, alguns microempreendedores e trabalhadores autônomos (MTA) ainda utilizarem blocos de notas ou editores de planilhas digitais, como o Excel da Microsoft por exemplo, para terem a análise e o controle de suas vendas acarretando numa possível erro de calculo, além de trabalhos manuais repetitivos de forma desnecessária. Entretanto, sistemas ERPs presentes no mercado, em sua grande maioria, consistem de sistemas complexos com algumas ferramentas desnecessárias para os MTA que impactam na plataforma e os bloqueiam, além da sua instalação confusa para leigos.

O capítulo está dividido nos subcapítulos de arquitetura de software e módulos, o primeiro consiste numa proposta de aplicar os conceitos e teorias para tornar o nosso código mais robusto a atualizações e otimizações e a segunda, diz respeito aos módulos existentes no projeto e suas devidas importâncias para estudo de caso presente no trabalho.

4.1 Arquitetura de software

Ao longo das últimas duas décadas, os softwares passaram a ter uma importante influencia sobre nossas vidas, afinal eles estão mais complexos e robustos. Os sistemas estão cada vez mais agilizando, organizando e nos informando, eles passaram

a ter funções chave em operações extremamente complexas e, para isso, também foram necessários aprimoramentos e melhorias em seus códigos e é aí que vem a importância da arquitetura de software.

Os benefícios de se ter um software bem arquitetado vão além da performance do sistema. Além do aumento do desempenho, temos uma garantia de escalabilidade, um termo que as empresas preservam e almejam para seus sistemas pois isso refletirá no tamanho do orçamento da manutenção devido a refatoração do sistema legado para se adequar a nova atualização, caso não seja escalável. Por último e não menos importante, a personalização do sistema onde se torna um trabalho menos arduo de ser feito caso haja uma evolutiva e apresentar uma boa arquitetura.

Dessa forma, no presente trabalho, iremos focar em teorias e embasamentos para que o projeto apresente uma boa arquitetura de software e possa ser performático, escalável e personalizável. Para isso, utilizaremos ferramentas de código aberto que estão presentes no mercado e que estão consolidados, além de aplicar os conceitos de Clean Architecture e princípios Solid.

4.2 Módulos

Os módulos do projeto consistem em três: Aplicação Web, Dashboard e Gerenciamento de Usuários. O primeiro e o último são obrigatórios no projeto pois consistem em, respectivamente, alimentar o sistema com informações das vendas, assim como, informar e o gerenciamento dos usuários que controlar o acesso de cada indivíduo ao sistema.

Os módulos atuais são separados em 3, contudo o usuário terá acesso a apenas dois deles conforme informado na figura 4.1. A aplicação Web para adicionar as informações dos produtos vendidos e , dependendo do nível de autorização pelo administrador, terá acesso a uma parte da aplicação de dashboard para consultar e acompanhar as informações globais das vendas adicionadas na aplicação. O administrador, por outro lado, terá acesso ao Dashboard e Gerenciador de Usuários, onde irá gerenciar quais usuários terão acesso a que painéis de visualizações e no caso do

Gerenciamento, terá controle dos usuários tais como informações pessoais, reenvio de senha e alterações de informações.

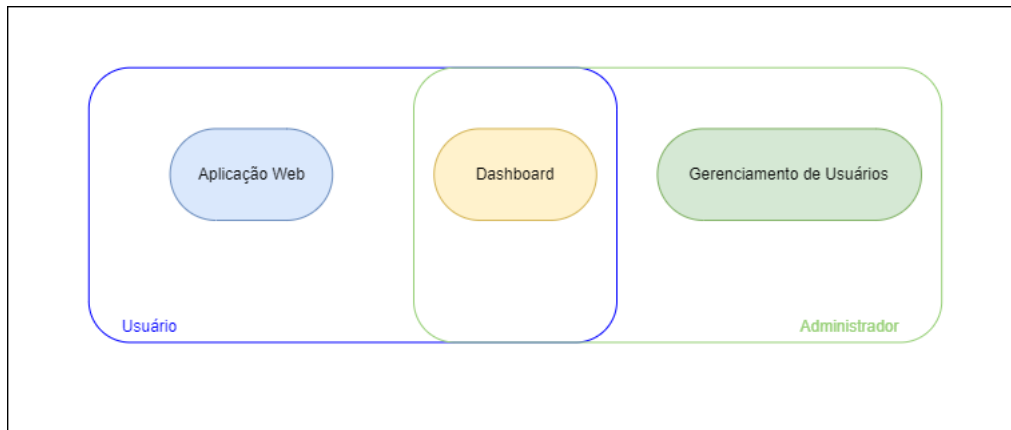


Figura 4.1: Diagrama de Venn dos Módulos do projeto

4.2.1 Aplicação Web

A aplicação consiste numa tela web onde o usuário consegue fazer a adição e edição de uma venda, fornecedor e tipo do produto. Ao adicionar uma venda, o individuo preencher um formulário com as informações do valor da venda, quantidade, fornecedor do produto, e o tipo do produto, além de outras informações menos relevantes para identificá-lo de forma mais organizada. Na tela principal de cada aba, há uma lista do que foi adicionado naquela aba, por exemplo, na tela principal da categoria, há uma lista com as categorias já criadas por aquele usuário.

Esse micro-serviço apresenta 4 abas dentro de um mesmo domínio do site: Produtos, Categorias, Fornecedor e Principal. As figuras 4.2, 4.3, 4.4 representam o fluxograma do caso de uso da aba de produtos, fornecedor e categorias, respectivamente. Na aba produtos, temos a lista de todos os itens cadastrados com os seus respectivos valores, categorias e fornecedores, sendo esses campos obrigatórios na hora do cadastro, além disso, temos a criação, edição e exclusão de itens nessa mesma pagina. No caso da aba de fornecedores, o formato se assemelha com a de produtos, contudo com campos referente a cadastro de pessoa como, nome, número de telefone, porcentagem do lucro e observação. Categoria idem, onde o usuário cria a categoria que achar melhor pois é de conhecimento que nem sempre as catego-

rias padrões existentes nas aplicações web do mercado se encaixam na estrutura de produtos da empresa.

Por ultimo, há a tela principal que lista as informações das três outras abas. Nesse caso, é apenas para leitura e com filtros para uma rápida pesquisa, essa função serve apenas para uma rápida consulta com o intuito de mostrar uma informação específica. Para aquisição de dados processados sobre uma visão macro da aplicação, recomenda-se acessar o serviço de dashboard.

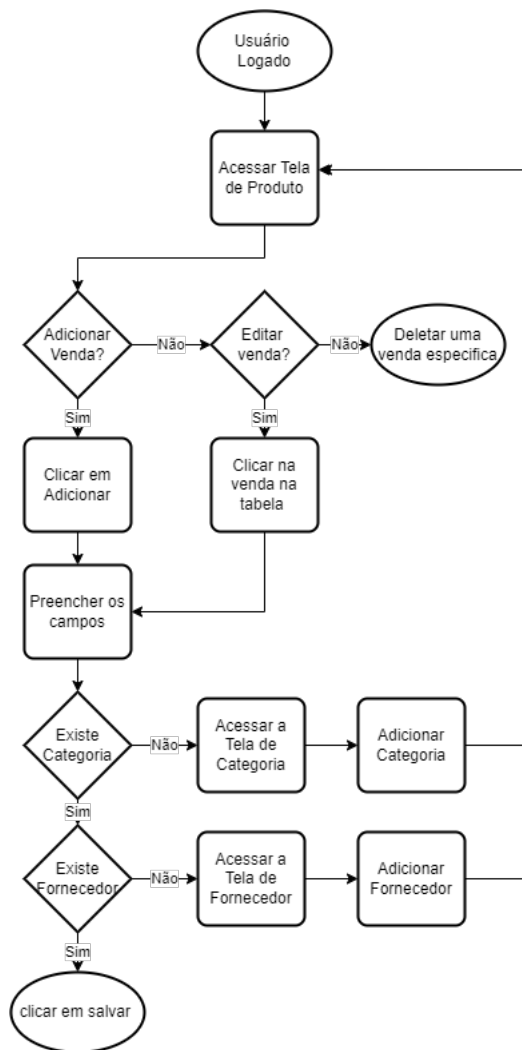


Figura 4.2: Fluxograma da tela de produtos

Dessa forma, a aplicação web funcionaria como uma adição de informações ao sistema, assim como o empreendedor que utiliza planilhas e blocos de nota adicionaria alguma venda nele. Além disso, Para esse módulo, foram aplicadas embasamentos

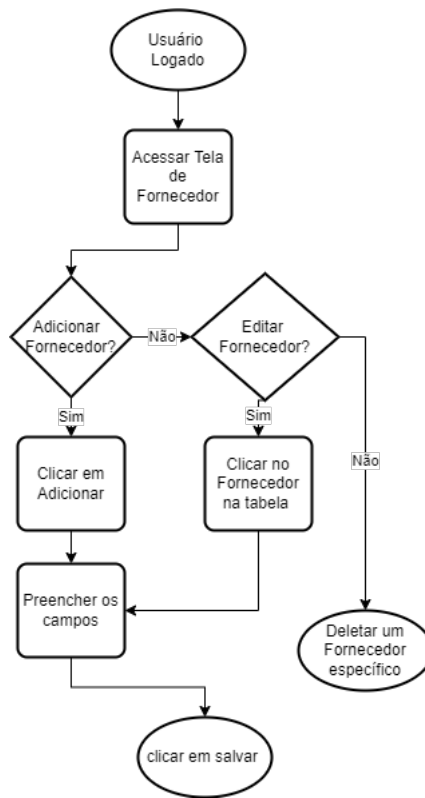


Figura 4.3: Fluxograma da tela de fornecedor

teóricos de arquitetura de software, como Clean Architecture para deixar o código e o fluxo desacoplados e permitindo uma personalização ou mudança com um mínimo de alteração nos arquivos e foram utilizados os princípios SOLID para termos um código organizado, de fácil entendimento e manutenção.

4.2.2 Dashboard

No serviço de dashboard, poderá ser visto informações macro da saúde de vendas do comercio. Nele constará gráficos de porcentagem das vendas por fornecedores, por categorias, além de séries temporais do lucro e do total de vendas diárias. Vale ressaltar que o filtro de período é aplicado para todos os gráficos simultaneamente, facilitando a compreensão tanto do total quanto de um período, seja do dia, de uma semana, de um mês, de um ano ou até do início da contabilização.

Nesse serviço existiram apenas dois tipos de contas, a de usuários e de administradores. O primeiro só poderá ter acesso de leitura dos gráficos e ajustes dos filtros para selecionar o período de análise, já o segundo, poderá alterar os os tipos

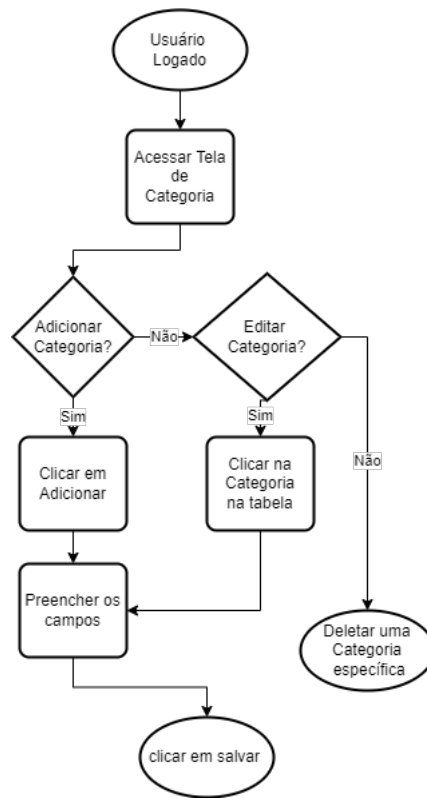


Figura 4.4: Fluxograma da tela de categorias

de gráficos, posições e até o que deve e quais dados podem ser mostrado. Além disso, o segundo terá controle dos usuários que poderão ler as tabelas e dados. A figura 4.5 retrata o caso de uso do dashboard na visão tanto do usuário quanto do administrador.

A ideia de termos a parte do dashboard separada da aplicação se consiste no fato de deixarmos o projeto como um todo mais fragmentado, com a ideia de micro serviços, afinal isso torna tanto a manutenção quanto o controle de qualidade do serviço mais alta pois apresenta um mínimo de impacto no resto de outros serviços e do projeto como um todo. Além do mais, facilita na hora de analisar alguma melhoria pro serviço e também permite a divisão de uma equipe para a sua manutenção.

4.2.3 Gerenciamento de Usuários

Esse serviço é essencial para funcionar com a aplicação web, pois ele faz parte do gerenciamento de usuários dela. Uma ideia inicial da aplicação web é que cada usuário tem os seus próprios dados e que um usuário não tem acesso a dados de outros

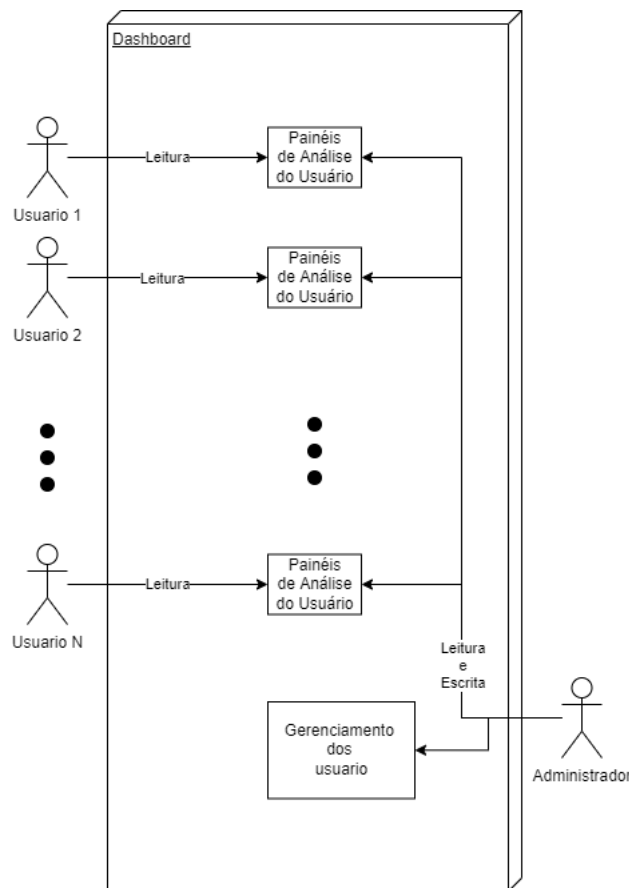


Figura 4.5: Caso de uso do módulo de dashboard

usuários e vice-versa. Dessa forma, para que isso possa ocorrer, há a necessidade do uso de um gerenciador de usuários.

Com o crescente aumento e avanço dos sistemas de software e tecnologias, governos e países estão adotando legislações mais rígidas em relação a armazenamento de dados dos usuários e ao uso deles, como por exemplo, a Lei Geral de Proteção de Dados Pessoais (LGPD) [30]. Dessa forma e pensando na segurança dos dados dos usuários, o projeto conta com um gerenciamento de usuários que armazena os dados em um ambiente separado da aplicação web, ou seja, tanto a tela de login quando a parte de segurança de navegação é feita por este micro serviço.

Dessa forma, visando a segurança, a pagina web trabalha com tokens que expiram ao invés do formato de acesso padrão, além da aplicação reconhecer os usuários não pelo nome ou outra informação pessoal, mas sim, por um id de 12 caracteres que contém letras e números. Isso aplica uma camada de segurança a aplicação,

dificultando no roubo de dados dos clientes. afinal, os dados sensíveis estarão dentro do banco do micro serviço de gerenciamento de usuários e não na da aplicação web.

Para esse módulo, será utilizado código de terceiros e Aberto, evitando esforço desnecessário e redundância de aplicação dado que se aplica ao nosso contexto e apresenta versões estáveis com poucos problemas, além de inúmeras extensões que permitem a personalização de layout e contas.

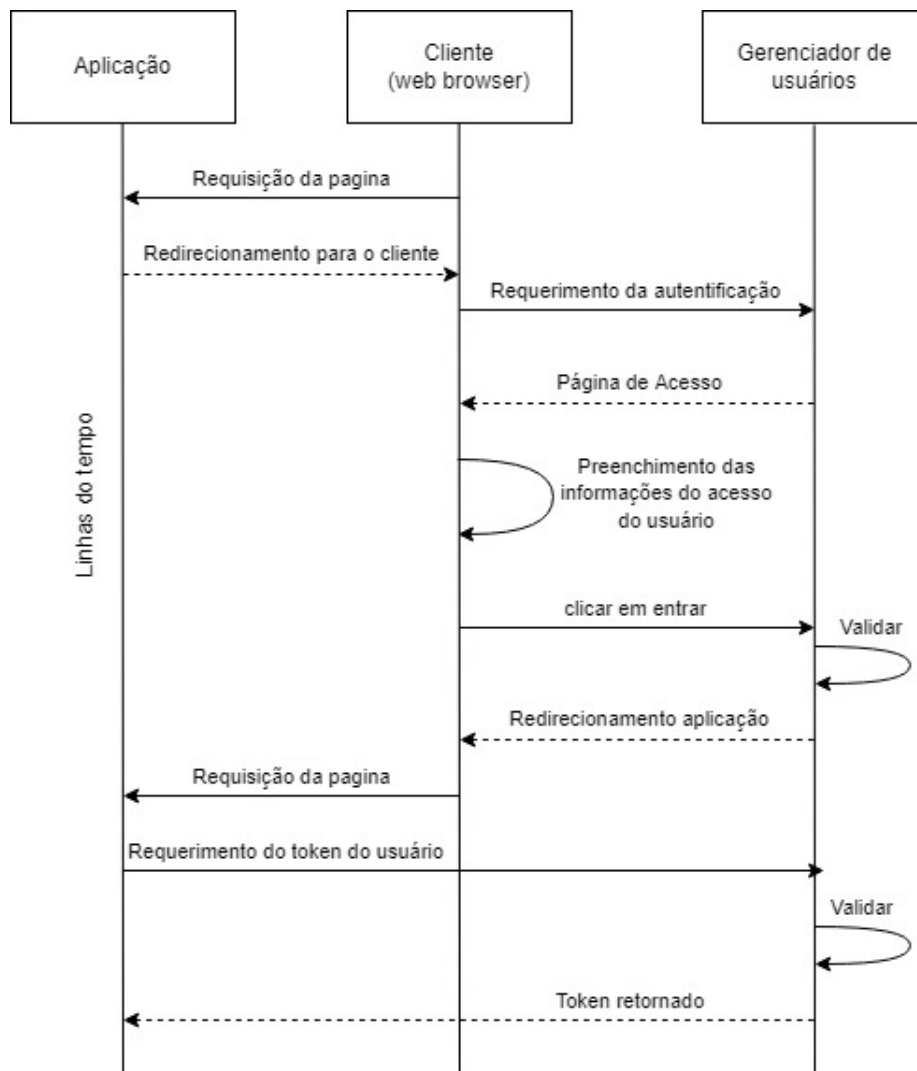


Figura 4.6: Lógica do login e aquisição dos tokens para a aplicação

A figura 4.1 representa a lógica por trás da tela de login e aquisição do token pela aplicação. Nesse fluxo, temos 3 objetos principais: A aplicação web, o cliente (usuário acessando a pagina pelo navegador de rede) e o gerenciador. O usuário faz uma primeira requisição para a aplicação e ela o redireciona para o gerenciador.

Caso o usuário já tenha se logado e o token de acesso não expirou, o gerenciador o redireciona direto para a aplicação, caso contrário o redireciona para a tela de login dele. Ao acessar a tela de login, o cliente preenche as informações para login ou cria um usuário através da tela de cadastro. Uma vez que clicou em entra, o gerenciador verifica se as informações batem com as informações em seu banco e caso esteja correta, o redireciona para a aplicação permitindo ele acessar as páginas internas dela. Dentro da aplicação, qualquer informação que deseja ser salva ou acesso de páginas, a aplicação requer o token do usuário para dar continuidade. caso tenha expirado ou está incorreto, o cliente é redirecionado para tela de login ou a ação não é permitida.

Conforme mostrado na figura 4.1, podemos ver que as únicas coisas que a aplicação sabe do usuário é se ele pode logar nas camadas mais internas dela ou alterar e visualizar certos dados dela através das informações que vem do gerenciador. Por outro lado, os dados sensíveis do usuário, como usuário e senha, são passados apenas para o gerenciador e ele que determina se o usuário pode ou não logar na aplicação. Em contra partida, o gerenciador não conhece os dados internos da aplicação, como por exemplo, o valor total de vendas ou quem são os fornecedores daquele usuário. Essa separação dos dois serviços tornam a aplicação mais segura e robusta contra exposição de dados sensíveis a indivíduos com segundas intenções.

Assim como foi explicitado no módulo de Dashboard, no caso do gerenciador de usuários da aplicação, também utilizaremos código de terceiros e aberto pois a sua implementação é completa e robusta, além de apresentar muito mais camadas de segurança comparado a um código desenvolvido de caráter próprio.

Capítulo 5

Implementação

Neste capítulo iremos explicitar como foi feita a implementação do projeto, assim como as escolhas das arquiteturas e conhecimentos teóricos aplicados na prática. Mostraremos os detalhes a nível de código e diagramas para que tudo seja o mais transparente possível para o leitor caso deseje replica-lo. Relembrando que o código está disponível no link <https://github.com/VenancioIgrejas/BreShow>.

O capítulo está dividido em duas seções, a primeira consiste na arquitetura do projeto e os sistemas periféricos internos para o funcionamento dela. Já a segunda, seria a implementação das aplicações, ou módulos, do projeto que, nesse caso, são microserviço dentro do sistema.

5.1 Arquitetura

A arquitetura do projeto foi pensada de forma a facilitar a sua instalação e configuração, assim escolheu-se containerização dos aplicativos e serviços. Afinal, basta o usuário rodar um comando para que quase tudo seja instalado e configurado. Entretanto, mesmo com a containerização, era necessário um meio de unir tudo e fazer os serviços conversarem entre si. Dessa forma, pensou-se na utilização de uma composição de contêiner para orquestrar todos os serviços de maneira simples e unificado.

Cada aplicação fica dentro de um contêiner e ao todo são 4 contêineres dentro da composição e cada uma é isolada da outra, mas compartilham da mesma rede. Temos

a de dashboard, que consiste numa aplicação de visualização de dados tratados do tipo interface gráfica para a análise de indicadores relacionados as informações de vendas que vieram do banco. A segunda aplicação é a de gerenciador de identidade e aplicação (GIA) que serve para gerenciar, validar e autenticar usuários para poderem acessar a aplicação web. Já a terceira é a aplicação de única pagina ou aplicação web (SPA), onde são fornecidos interfaces para o usuário poder adicionar informações referentes a vendas, fornecedores e categorias que, futuramente, serão utilizados pela aplicação de dashboard. Por último, a aplicação de banco de dados, o qual se conecta com todas os outros contêineres para prover ou armazenar dados ligados tanto ao fluxo de negocio quanto metadados das aplicações (figura 5.1).

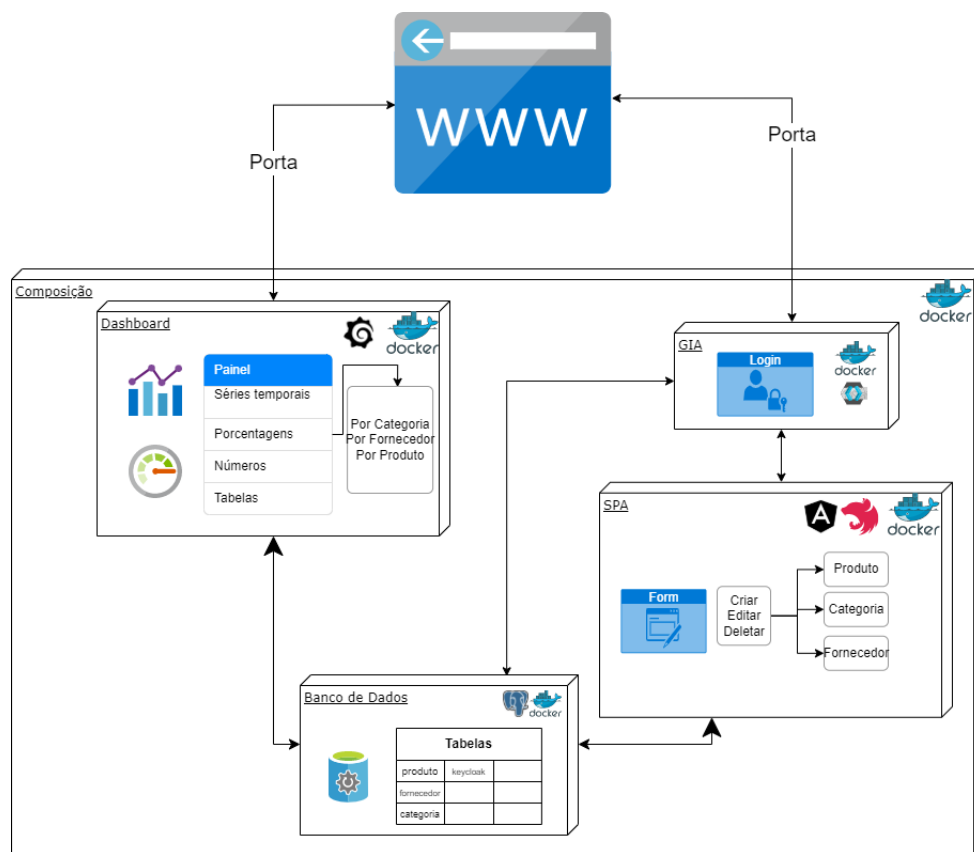


Figura 5.1: Arquitetura macro do projeto separada por contêiner

O fluxo consiste de duas urls que o usuário pode acessar através de portas, uma da aplicação web que é intermediado pelo GIA e a outra, pela aplicação de dashboard. Por sua vez, apenas eles e a SPA tem acesso ao banco de dados, onde são armazenados informações cruciais para o seu funcionamento e dos usuários. Dentro

da composição, os contêineres se comunicam através de portas e chaves, por exemplo, o dashboard, GIA e SPA utilizam a porta 5432 e autenticação por usuário para acessarem os dados no contêiner de banco de dados, já a comunicação entre o GIA e o SPA vem da porta 28080. Essas informações estão explícitas no arquivo `docker-compose.yml` presente na pasta raiz do projeto.

5.1.1 Orquestrador

O orquestrador que foi escolhido para o projeto foi o Docker, pois se trata de uma plataforma como serviço (PaaS) que já está consolidado no mercado e consegue isolar o software nele instalado de todo o sistema operacional do computador de onde fora instalado. além de ser possível escrever comandos em um arquivo denominado `Dockerfile` para ordenar uma sequência de configurações em tempo de execução de instalação do contêiner.

Como cada contêiner necessitaria do seu próprio `Dockerfile`, isso torna a configuração e instalação muito demorada caso precise fazer isso em um número elevado de aplicações em seus próprios contêineres. Por conta disso, utilizamos a ideia de composição, algo que já existe no Docker, e o arquivo de configuração da composição se encontra no caminho:

```
{raiz do projeto}/docker-compose.yml
```

Esse arquivo é uma junção de vários `Dockerfile` para instalar e configurar de forma paralela inúmeros contêineres. Para instalar os pacotes apropriados já com uma configuração padrão, foram utilizados imagens pré-configuradas das ferramentas disponibilizadas pela respectiva empresa que contem os direitos. Dessa forma, as imagens utilizadas dentro de `dockfiles` e `docker-compose` foram a `node:alpine` com `nestjs` para o backend do contêiner da SPA, `node:14` com `angular 12` para o frontend da SPA, `grafana/grafana:7.3.6` para o dashboard, `postgres-dev:1.0.0` para o banco de dados e por ultimo, `jboss/keycloak` para o GIA. Uma vez os arquivos configurados, basta rodar o comando “`docker-compose up`” no terminal na raiz do projeto para iniciar a composição.

5.1.2 Banco de dados

O banco de dados escolhido para o projeto fora o Postgresql, por se tratar de um banco de dados relacional de código aberto, documentação completa, uma comunidade ativa e ter uma imagem no DockerHub. A configuração do contêiner dele está no arquivo docker-compose com as seguintes linhas:

```
postgres:
  container_name: postgres_containerbreshow
  image: postgres-dev:1.0.0
  build:
    context: .
    target: development
    dockerfile: ./data/Dockerfile
  environment:
    POSTGRES_USER: ${POSTGRES_USER}
    POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
  volumes:
    - ./data/postgres-data:/var/lib/postgresql/data
  ports:
    - "5432:5432"
  networks:
    - postgres
```

Após a ativação do contêiner dele, não há a necessidade de mexer pois a criação das tabelas informadas na figura 5.1 são feitas de forma automática pelas outras aplicações. Além disso, visando a segurança, as informações que advém do campo “enviroment” no docker-compose.yaml são extraídas do arquivo de ambiente (.env) do projeto.

5.1.2.1 DER do Banco de Dados

A modelagem do projeto consiste na existência de 3 tabelas importantes dentro do banco, são elas product, category e provider, o qual são representadas pelas entidades Produto, Categoria e Fornecedor respectivamente. a tabela product apresenta

duas chaves estrangeiras que são da tabela category e provider, afinal toda venda deve necessariamente ser categorizada e deve pertencer a um fornecedor específico. Todas as tabelas apresentam uma coluna chamada “idUser” que representa o ID do usuário da aplicação o qual adicionou aquele dado, além de algumas colunas em comum que são metadados oriundos do pacote TypeORM. O Diagrama Entidade Relacionamento (DER), que foi gerado por um gerenciador de PostgreSQL, está representado na figura 5.2 e vale ressaltar que a tabela “migrations.typeorm” serve para o controle de versionamento das migrações feitas pelo TypeORM e é uma tabela de metadado.

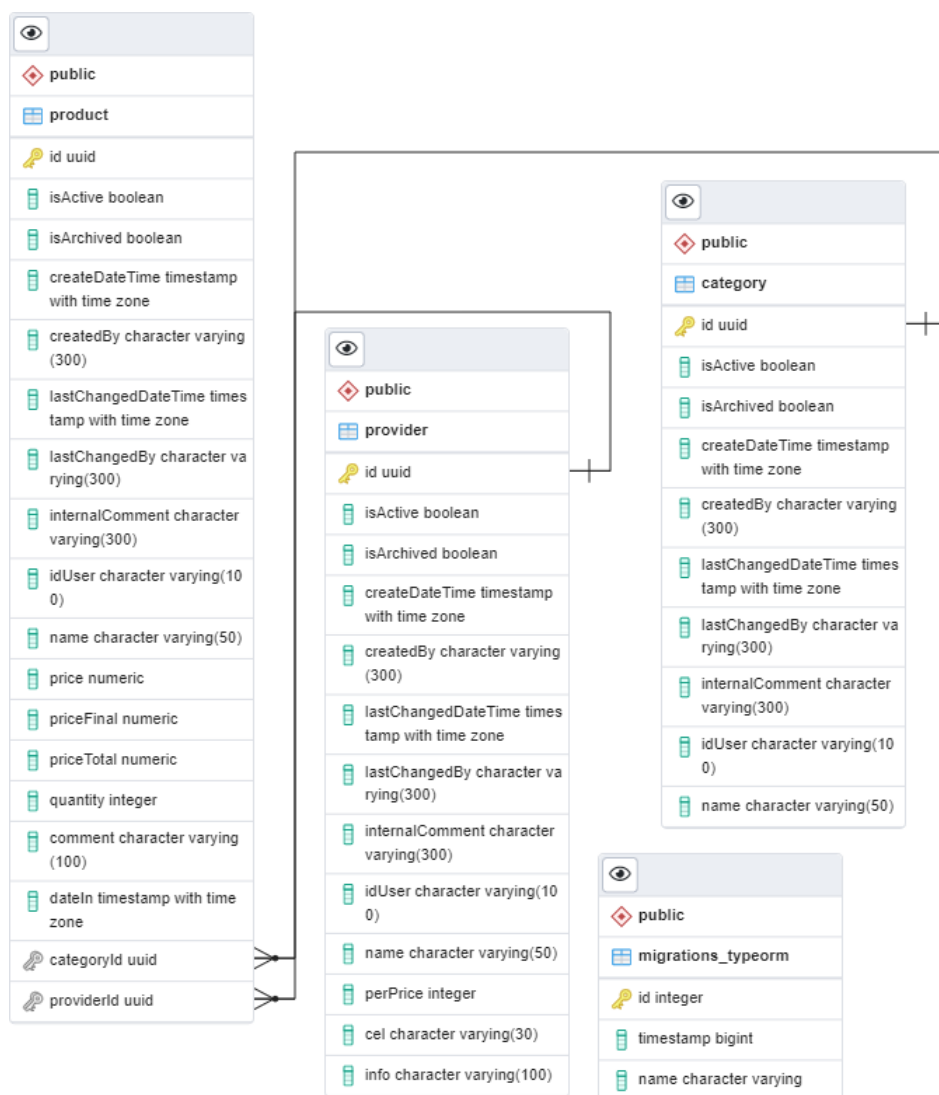


Figura 5.2: Diagrama entidade relacionamento do banco de dados da aplicação

5.2 Aplicações

Atualmente, o projeto conta com três aplicações em que dois deles trocam informações entre si da parte de segurança, que é o caso do gerenciados de usuário e a aplicação web. Já o de dashboard, é o único que funciona de forma isolada das outras aplicações.

Após análise da estrutura da aplicação web, foi estipulado que a parte de identificação, segurança e usuários seria delegado a uma terceira aplicação já existente e consolidada no mercado. Essa escolha fora feita para aumentar a segurança e confiabilidade do nosso projeto pelo usuário.

5.2.1 Gerenciador de usuários

O software de gerenciador de usuários escolhido foi o Keycloak por se tratar de um código aberto, recursos gratuitos completos e utilizado por muitas empresas de renome. Foi feita uma analise de outros software de GIA, como Firebase Authentication e Auth0, porém ambos apresentam limitações na versão gratuita o que vai contra a proposta do trabalho de ser um projeto totalmente gratuito.

Essa aplicação é instanciado quando é levantando os contêineres usando o comando docker-compose, a parte responsável do keycloak no arquivo docker-compose.yml é representado por:

```
keycloak:
  container_name: keycloak_containerbreshow
  image: jboss/keycloak
  volumes:
    - ./keycloakImports:/opt/jboss/keycloak/imports
  environment:
    DB_VENDOR: postgres
    DB_ADDR: postgres
    DB_DATABASE: ${POSTGRESQLDB}
    DB_USER: ${POSTGRES_USER}
    DB_PASSWORD: ${POSTGRES_PASSWORD}
```

```
DB_PORT: 5432
KEYCLOAK_USER: admin
KEYCLOAK_PASSWORD: password
JDBC_PARAMS: "ssl=false"
KEYCLOAK_IMPORT: /opt/jboss/keycloak/imports/realm-export.json
-Dkeycloak.profile.feature.upload-scripts=enabled
ports:
  - "28080:8080"
networks:
  - postgres
depends_on:
  - postgres
```

Lembrando que a imagem do Docker do keycloak é original da própria empresa e as variáveis de ambiente devem ser alteradas para uma maior segurança da aplicação. Utilizamos os volumes nesse caso para manter as informações salvas de configurações do realm e clients, o que são essenciais para o funcionamento do projeto.

Ao acessar o link <http://localhost:28080/> e logar com o usuário e senha presentes no código acima do docker-compose, deve-se colocar no realm BreshowAD pois é a que o projeto fora configurado. Ao clicar na aba “client” na esquerda aparecerá os clients tanto do próprio sistema quanto os criados manualmente apareceram no centro da tela e o resultado final deverá ser parecido com o da figura 5.3. Os clients “angular-front” e “nestjs-back” foram criados manualmente com o intuito de serem entidades de dentro da aplicação web para fazerem as requisições necessárias no Keycloak para se autenticarem, consultar informações do usuário ou se eles tem permissões de acesso e visualização de dados.

O client angular-front é referente ao front-end da aplicação web, ou seja, esse client serve como a porta de entrada da nossa aplicação e é ele que redireciona o usuário a tela de login do keycloak (da nossa aplicação web) caso tente acessar alguma url interna do sistema web não estando logado. Uma vez logado, o client permite que o usuário acesse as paginas internas da aplicação.

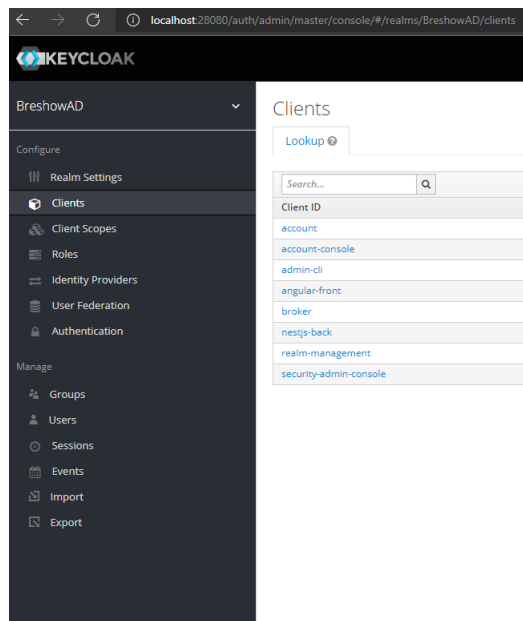


Figura 5.3: Tela do cliets do realm BreshowAD do Keycloak

O método de login mencionado no paragrafo anterior é possível devido ao fato do tipo de acesso do client ser publico, o que é uma opção selecionável na tela de configurações do próprio client. Ele ser desse tipo faz com que a aplicação que o utilize necessite de uma tela de login para validar o acesso do usuário a aplicação. Vale ressaltar também que é necessário configurar a rota raiz para o da aplicação web (<http://localhost:4200>) pois será onde o keycloak redirecionará o usuário após o login na plataforma, além disso, é possível configurar quais rotas são validas para o client acessar, tornando a aplicação mais segura a possíveis tentativas de roubo de dados.

O client nestjs-back é referente ao back-end da aplicação web e ele foi configurado pra ser do tipo “bearer-only”. Essa configuração obriga que ao fazer uma requisição para o servidor, seja necessário que passe no cabeçalho da requisição um JSON Web Token (JWT) comprovando que o individuo tem permissão para acessar ou atualizar informações ao servidor.

No nosso caso, O JWT pode ser requisitado pela aplicação web após o usuário ter se logado na tela de login para utiliza-lo para fazer qualquer operação no backend, isso tudo por baixo dos panos da aplicação web. Em outras palavras, o JWT é um intermédio de segurança que o front-end e o back-end da aplicação usam para troca

de informações entre eles e assim, não permitindo que requisições maliciosas tente roubar informações do backend. Pelo lado do keycloak, nada mais é que o client angular-front fornece um JWT do usuário logado para o client do nestjs-back para fazer o controle do acesso.

Na sessão sobre a aplicação web, iremos mostrar e explicar mais a fundo sobre as configurações desses clientes nas arquiteturas dela, como faz a instalação e que frameworks utilizamos para implantar o Keycloak neles. O que podemos observar é que o uso de dois clientes para a mesma aplicação se tornou útil pois transformou o backend em um microserviço que pode ser consumido por outros sistema, afinal basta ter o cabeçalho correto e o corpo da requisição no mesmo padrão para isso.

Nessa etapa da implementação, ela foi bem complicada e tivemos muitos problemas pois os erros que apareciam na aplicação referente ao Keycloak eram genéricos e foi necessário utilizar o método de tentativa e erro para conseguir configurá-lo da forma como está atualmente os dois clients. Por conta da documentação ser complexa para pessoas leigas no assunto de cibersegurança e a comunidade ainda estar em expansão, algumas configurações de seguranças adicionais não foram aplicadas por acabarem aparecendo bugs nos sistemas com logs confusos.

5.2.2 Aplicação Web

A aplicação web fora a única feita do zero e construída baseada em boas práticas de programação, utilizando princípios como o SOLID e Clean Architecture. O seu intuito é , através do navegador, permitir que os usuários de forma simplificada e fácil insiram informações sobre as suas vendas para alimentar o banco de dados com os dados refinados para uso posterior.

De acordo com a regra de negócio estipulado no projeto, as vendas que são adicionadas precisam estar relacionadas a uma categoria e a um fornecedor e dessa forma, precisou-se criar três tabelas referentes a cada entidade. Elas foram denominadas “product”, “category” e “provider”, sendo referenciadas as entidades de vendas, categorias e fornecedores respectivamente.

Essa aplicação foi separada em dois blocos, o primeiro bloco é referente ao código que está rodando no navegador e é responsável pela parte visual da aplicação conhecido como frontend. Já o segundo bloco, é onde o processamento dos dados e a aplicação da regra de negócio é feita, chamada-se de backend. Essa metodologia segue o padrão de projeto MVC e a código fora implementado seguindo os seus princípios.

5.2.2.1 FrontEnd

Inicialmente, escolheu-se React para essa parte da aplicação, porém tivemos alguns problemas com bibliotecas do Keycloak para validação dos usuários. Dessa forma, por conta desse problema e o fato do NestJs ter sido escolhido como o framework para o backend, que é escrito em Typescript e baseado na estrutura do angular, escolheu-se o Angular 12 para ser o motor do frontend. Além disso, para customização de componentes mais rebuscados, foram utilizados componentes desenvolvidos pelo PrimeNG.

O código dessa parte encontra-se dentro da pasta “client” que está na raiz do projeto e o seu esquema pode ser vista na figura 5.4. A organização do código consiste na criação de pastas voltadas para a fácil manutenção, compreensão e expansão do código ao expandir a regra de negócio. A parte principal do frontend está na pasta client/src/app, pois é onde as telas são montadas. A pasta “component” é onde armazena componentes utilizados em mais de uma tela como, por exemplo, a estrutura padrão. Já a pasta “module” é onde estão os mapeamentos das entidades que utilizamos na aplicação, como a categoria, fornecedor e produtos(vendas) que são da regra de negócio e jwt-token e router que são utilizados internamente no angular.

A pasta “service” está atrelado a montagem das requisições que são feitas ao usuário disparar uma ação na tela, como um botão de salvar, deletar ou criar, vale ressaltar que cada entidade tem o seu serviço separado, respeitando os princípios SOLID. Na pasta guard e init estão arquivos voltados para as configurações do Keycloak para o client “angular-front” do realm BreshowAD, a classe AuthGuard do arquivo guard/auth.guard.ts serve para validar e bloquear possíveis usuários que queiram acessar urls internas da aplicação sem estarem autorizados pelo Keycloak.

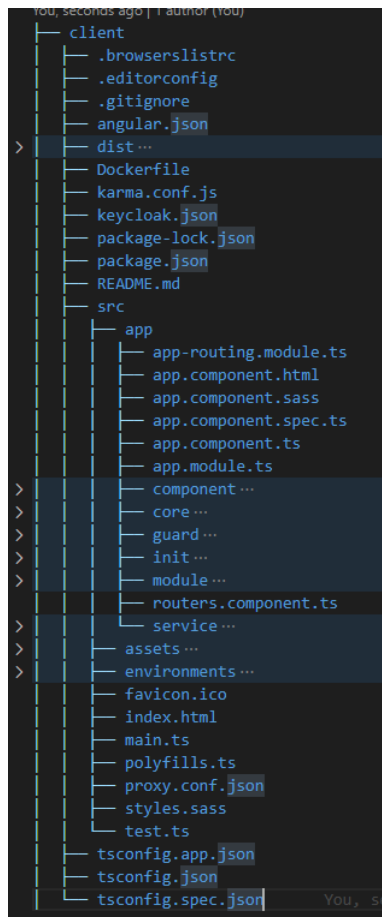


Figura 5.4: Estrutura de pastas do frontend da aplicação web

Além disso, o arquivo `keycloak-init.ts` tem o intuito de forçar a inicialização primeiro da configuração do serviço do keycloak e só após o retorno do serviço é que o angular inicia a aplicação.

A pasta “core” é onde estão os códigos referente as abas da aplicação web. Na figura 5.5, podemos observar que cada entidade apresenta sua própria pasta e suas diferenças. Esse rearranjo facilita na hora da manutenção de cada aba, pois elas estão separadas, contando também com a aba de home. A estrutura padrão do angular nos ajudou a criar um modelo onde dentro de cada pasta onde continha a sua pagina html, o arquivo de estilo (sass) e o arquivo de componente, que seria a “maquina de estados” por traz da tela. Alias, Decidimos separar o código do componente de popup de cada entidade (entidade-page) em um novo componente chamado entidade-form para deixar o código ainda mais organizado.

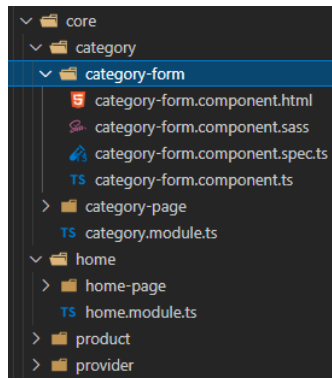


Figura 5.5: Estrutura da pasta “Core” do frontend da aplicação web

Outros arquivos que não foram mencionados no texto que estão presentes na figura 5.4 são relacionados a configurações do Kecoak, pacotes de bibliotecas utilizados, compilações, ajustes de rotas e configurações do Typescript e do Angular.

O desenvolvimento do frontend não foi tão complicado, o que demorou mais foi a confecção da primeira aba que foi a de categorias. As outras, que tinham o estilo parecido umas com as outras, foram desenvolvidas bem rápido. Por conta da arquitetura robusta da aplicação web, a parte estética das telas ficaram para trabalhos futuros.

5.2.2.2 Backend

Para o servidor, foi escolhido a ferramenta Nestjs por se tratar de um framework em Node e apresentar uma aceitação grande pela comunidade, além de uma documentação simples e completa. Outro ponto positivo desse framework é o fato de existir um pacote pronto e simples para conectar o client do keycloak no servidor. O backend da aplicação consiste na aplicação que está dentro da pasta “api” na raiz do projeto.

A configuração do keycloak no backend consiste 2 arquivos conforme mostrado na figura 5.6, um para adição dos módulos dele que está no arquivo app.module.ts no caminho raiz do projeto/api/src e o outro é o arquivo keycloak.json, onde é configurado as informações do client na aplicação do keycloak. No arquivo app.modules.ts, temos que o pacote ‘nest-keycloak-connect’ se encarrega de todo o trabalho árduo de configuração e troca de informações entre as plataformas do nestjs e keycloak. O que

é passado para ele são as informações que estão contidas no arquivo de configuração como, a url usada para autenticação, realm e o client que são usados, a palavra secreta do client e o tipo de acesso que, nesse caso, é apenas utilizando requisições que utilize a autenticação do tipo “bearer”.

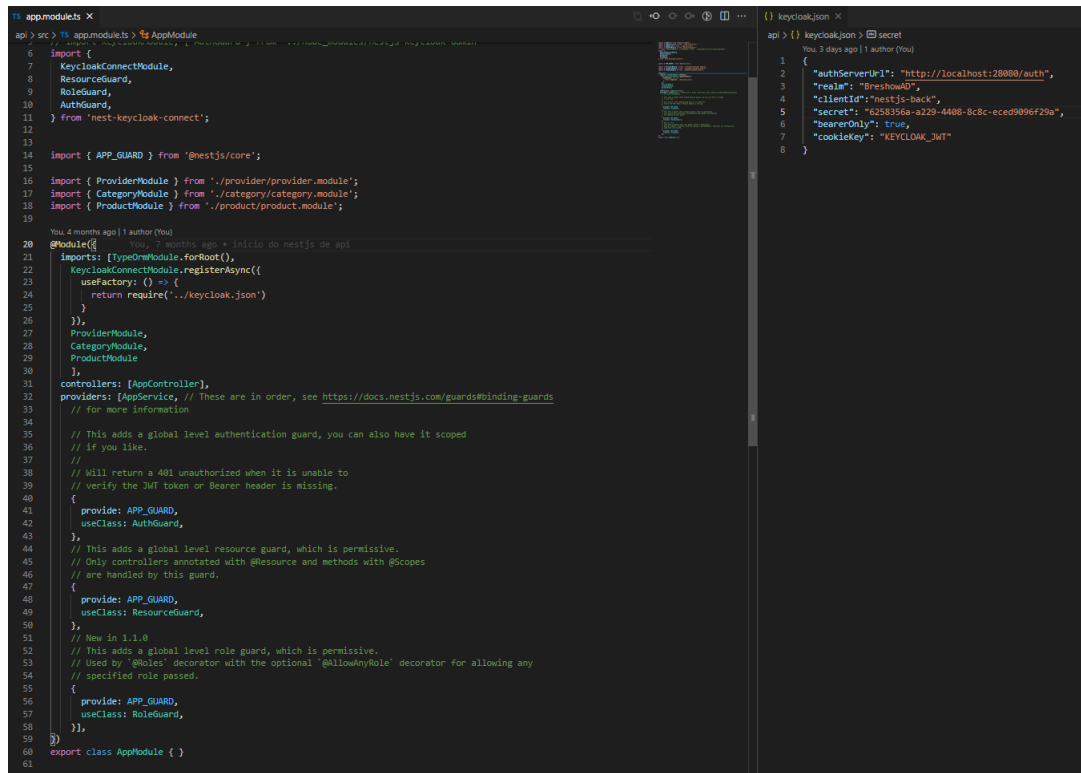


Figura 5.6: Arquivos relacionados as configurações do keycloak no backend

A segunda parte seria a configuração dos guardiões do nestjs, onde são classes que se encarregam de bloquear uma requisição caso suas informações não estejam de acordo com um padrão estipulado pela aplicação do Keycloak. No caso do pacote do Keycloak, temos 3 guardiões que bloqueiam as requisições de acordo com as configurações do client e do usuário e eles são instanciados em uma lista do objeto “providers” no arquivo app.module.ts. O que utilizaremos mais é a classe AuthGuard, onde ele bloqueia a requisição caso o bearer ou o jwt esteja incorreto, retornando o erro http 401 que significa não autorizado. Vale ressaltar que, uma vez adicionado esses guardiões, todos os endpoints do servidor terão por padrão eles ativados e apenas serão desativados se for adicionado um decorador nele.

Outro ponto importante de se comentar é a aplicação de teorias e padrões de software utilizados na arquitetura do servidor. A estrutura montada no servidor seguiu a proposta da Clean Architecture e pra isso, utilizou-se uma técnica de mapeamento de objeto relacional, do inglês Object Relational Mapper (ORM), para desacoplar a dependência da aplicação a algum determinado tipo de banco de dados. A nível de código, utilizamos a biblioteca TypeORM e Migração para fazemos o mapeamento das entidade e a criação ou destruição de tabelas apenas iniciando um script, respectivamente. A própria biblioteca se encarrega de traduzir o código para a língua nativa do banco de dados escolhido que, no nosso caso, fora o Postgresql. Além disso, introduzimos a ideia de objeto de transferência de dados, em inglês Data Transfer Object (DTO), para transferir os dados que vem das requisições do frontend para as entidades no servidor.

Além de aplicarmos o ORM e o DTO, que estão relacionados a camada de adaptadores das interfaces na abordagem da Clean Arctecture, o código foi separado por regra de negócio, ou seja, tanto a entidade de vendas quanto a categoria e o fornecedor têm a sua própria camada de negócio e modelos bem definida. A nível de código, por exemplo, a pasta da categoria (figura 5.7) apresenta o arquivo de “controller”, onde são feitas as requisições que estão diretamente ligados ao arquivo de “service” que por sua vez, necessita da “entity” para fazer as suas alterações ou regras de negócio da aplicação. O fluxo acima apresentado mostra um nível maior de dependência que a aplicação tem pras entidades do que para periféricos e adaptadores, dessa forma podemos observar que o código e sua arquitetura estão alinhados com os princípios da Clean Architecture.

Além desse padrão de software, mesclamos ela com a estrutura Folder-by-Feacture, afinal esse é um dos padrões usados pelo próprio Nestjs na criação de suas pastas, além de facilitar a manutenção, pois tudo que for relacionado aquela entidade, está dentro do seu próprio repositório. Isso pode ser visto na figura 5.7, em que tudo o que é relacionado a categoria está dentro do seu repositório.

Para os endpoints, foram criados 5 para cada entidade, são elas: Criar entidade, listar todas as entidade, trazer informação de uma específica, editar e deletar. Ape-

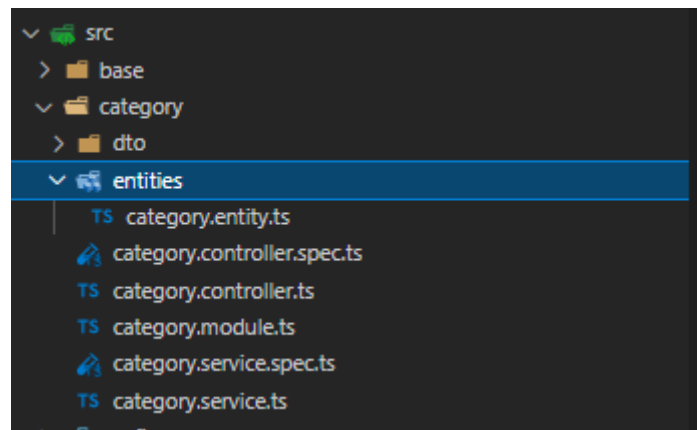


Figura 5.7: Estrutura da entidade de categoria do servidor

nas a entidade de vendas que apresenta uma regra de negocio há mais devido a cálculos voltados para valor total final da venda por conta do fornecedor conforme mostrado na figura 5.8.

```
async update(id: string, updateProductDto: UpdateProductDto) {
  const provider = await this.providerRepository.findOne(updateProductDto.providerId);
  const priceFinal = (1 - provider.perPrice/100.0) * updateProductDto.quantity * updateProductDto.price;

  return this.productRepository.save(<Product>{
    id: id,
    name: updateProductDto.name,
    price: updateProductDto.price,
    quantity: updateProductDto.quantity,
    priceTotal: updateProductDto.price * updateProductDto.quantity,
    priceFinal: priceFinal,
    comment: updateProductDto.comment,
    dateIn: updateProductDto.dateIn,
    provider: <Provider>[
      {
        id: updateProductDto.providerId,
        category: <Category>{
          id: updateProductDto.categoryId
        }
      }
    ]
  });
}
```

Figura 5.8: Código da regra de negócio de atualização de uma entidade de venda que está no arquivo “product.service.ts”

O código do servidor da aplicação não foi tão complexo para ser feito, apenas ocorreram alguns problemas com bugs na escrita do código do arquivo de Migração, pois alguns comandos não funcionavam na hora da tradução interna de código pra linguagem de banco. Por ultimo, ocorreu um problema de rota e SSL na hora de colocá-lo em produção dentro do contêiner, nos obrigando a deixar a aplicação web funcionando fora do Docker.

5.2.3 Análise dos dados

Inicialmente a aplicação de análise de dados seria uma parte da aplicação web, onde haveria uma aba mostrando gráficos com uma visão macro das vendas daquele usuário. Porém, decidimos utilizar alguma ferramenta já conhecida no mercado para esse propósito e após uma breve pesquisa, foi escolhido o Grafana para esse propósito. A vantagem de ser código aberto, ter uma gama extensa de extensões gratuitas e ser utilizado por inúmeras empresas grandes tais como, a Siemens e Paypal, nos fez escolhê-la. Outro ponto vantajoso é que existe uma imagem pronta dela no DockerHub, o que nos permitiu agregar a sua imagem a nossa composição de contêineres.

No arquivo de composição do docker, que é o `docker-compose.yml` na raiz do projeto, a parte responsável pelo grafanar é o seguinte código:

```
grafana:
  image: grafana/grafana:7.3.6
  ports:
    - 8084:3000
  volumes:
    - ./grafana:/var/lib/grafana
  networks:
    - postgres
```

O docker instancia uma imagem do Grafana, onde salva todas as informações importantes na pasta grafana do projeto e inicia a aplicação na porta 8084 do localhost. A aplicação monta os gráficos a partir dos dados vindo do banco de dados diretamente, ou seja, o código por trás dos dados que aparece em um painel são linguagens de banco de dados, como SQL. O banco é configurado na aba de “Datasource”, que no nosso caso foi o postgresql e só tivemos que colocar informações como host, usuário e senha.

Cada gráfico fica dentro de um painel, onde pode ser customizado tanto o tipo de gráfico, quanto os layouts e eixos. Por exemplo, a figura 5.9 nos mostra o gráfico

de vendas totais e lucros por dia e nele, podemos customizar o título, a cor de cada linha ou área no gráfico, se deve mostrar os números nos eixos e que tipo de números são esses. No painel tem até o símbolo da moeda brasileira caso algum eixo seja representado em moeda. Além disso, é no próprio painel que se escreve o código para trazer os dados que deseja aparecer no gráfico e existem palavras reservadas para poder aplicar os filtros temporais, além de necessariamente precisar de pelo menos um campo no formato de data.

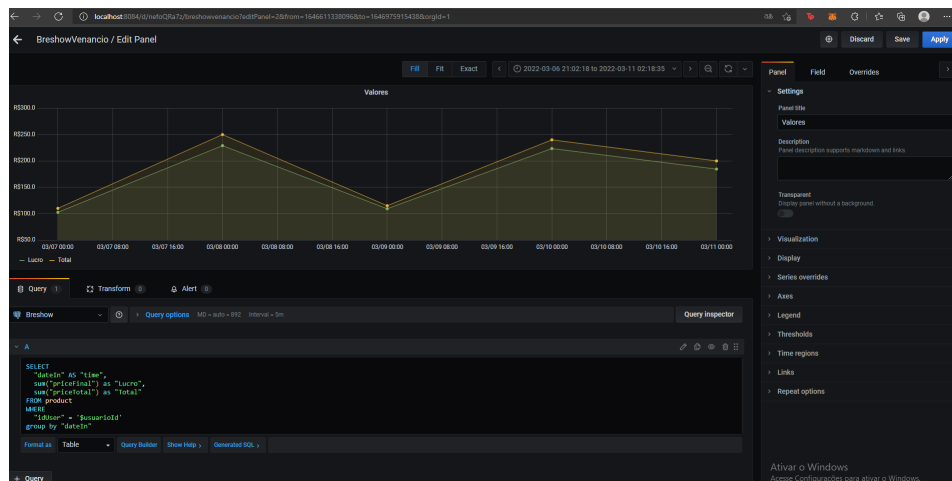


Figura 5.9: Painel de valores do dashboard de um usuário no Grafana

Além do gráfico de vendas que é uma série temporal, temos também de barra que representa a quantidade de produtos vendidos por dia, número estatístico que representa o número total de vendas naquele período no filtro e o gráfico em pizza que mostra a porcentagem de vendas por fornecedor além de ter texto nele. Eles estão exemplificados e seus códigos a mostra, respectivamente, nas figuras 5.10, 5.11, 5.12

Após a configuração e criação de cada painel, nós podemos organiza-los dentro do dashboard do usuário e ,no exemplo que utilizamos de teste, ficamos com a seguinte visão geral mostrado na figura 5.13 das vendas em um período de 08/03/2022 até 08/03/2022. Onde todos os dados vieram do banco de dados que é integrado com a aplicação web do projeto, tornando tanto o aplicativo de análise de dados quanto a da aplicação web isolados e sem interferência um do outro, apenas compartilhando a mesma base.

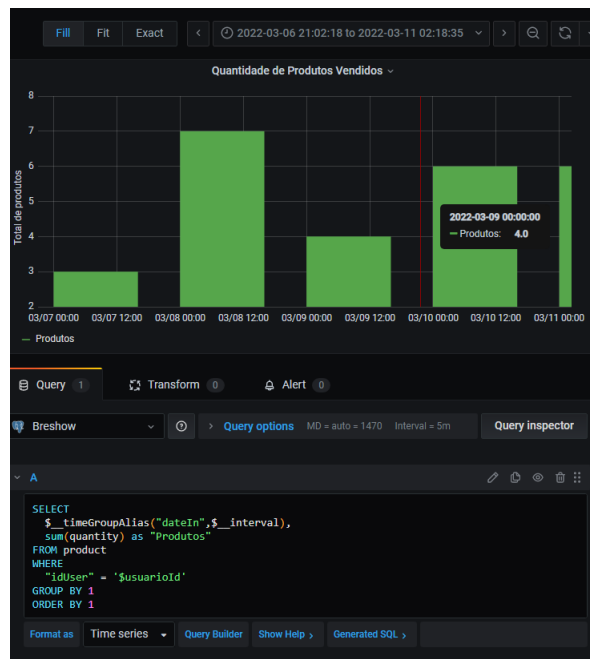


Figura 5.10: Painel de valores em barra do dashboard de um usuário no Grafana

Infelizmente, caso um novo usuário seja criado na aplicação web, o mesmo deve ser feito no Grafana na parte de gerenciamento de usuários, pois o Keycloak não é compartilhado com o Grafana. No caso do dashboard, recomenda-se apenas duplicar algum dashboard existente e alterar a variável “usuarios” no campo de configuração do dashboard duplicado para o nome do novo usuário (figura 5.14).

É de suma importância que o administrador do Grafana crie usuários específicos para cada um da aplicação web e os coloquem com apenas a visualização de seus respectivos dashboards, mas isso fica a sua escolha. alias, a versão atual que está no github já tem pelo menos um dashboard padrão pronto, apenas precisando alterar a variável “usuarios” para que ele traga os dados corretos nos gráficos.

Nessa parte do desenvolvimento não tive muitas complicações pois a documentação do Grafana é simples e intuitiva, além da imagem dele no Docker ser bem estável na versão em que está. Montar os gráficos e as consultas a banco foram trabalhosas, mas mesmo assim, foram bem mais fáceis que outras partes do projeto. Além do que, pelo fato de ter uma interface gráfica, facilitou a configuração com o banco de dados, sendo ele a única integração necessária com a aplicação.

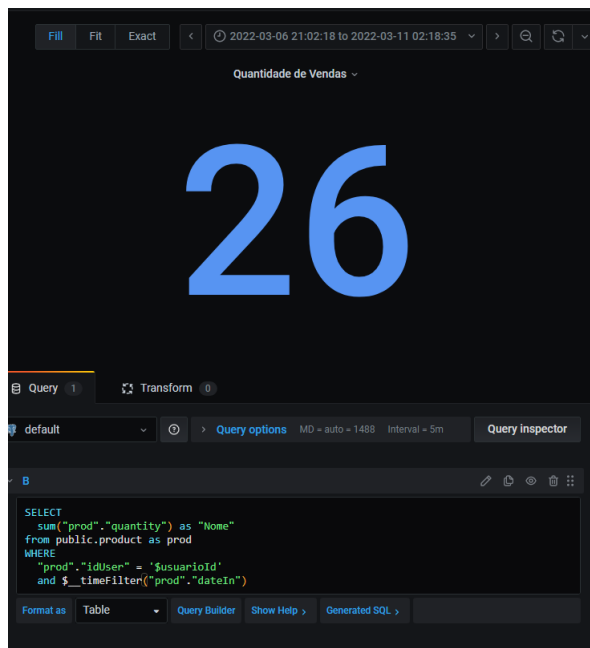


Figura 5.11: Painel de valores em estatística do dashboard de um usuário no Grafana

5.3 Desafios

Ao longo do desenvolvimento, foram alteradas algumas ferramentas por conta de problemas de compatibilidade entre elas, além da curva de aprendizagem serem bem inclinadas ou longas demais. Foi o caso da escolha do frontend, inicialmente utilizou React, contudo mudou-se para Angular 12 pelo fato da estrutura do backend, que é o Nestjs, ser baseado no angular e assim, facilitando o desenvolvimento da aplicação e a biblioteca de integração do Keycloak para angular 12 ser mais completa e documentada do que a de React. A escolha do gerenciamento de usuários seria o sistema Firebase da Google, porém ela apresenta alguns recursos que só são possíveis utilizar na versão paga, por conta disso, foi escolhido um sistema com os recursos totalmente gratuito, como o Keycloak.

O desenvolvimento da aplicação como um todo foi relativamente rápida pois a maioria das ferramentas que utilizamos apresentavam uma documentação clara, chegando até a ter vídeos com exemplos práticos, como, por exemplo, o nestjs, docker e angular 12. Contudo no caso do Keycloak, a documentação não era tão clara para alguém com pouco conhecimento na área de segurança de rede e por conta disso, a aplicação acabou passando do prazo estipulado do trabalho, afinal houveram

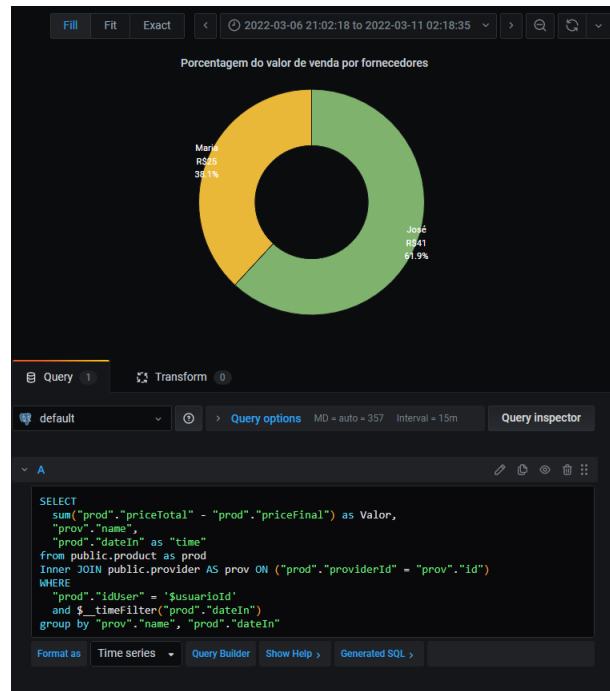


Figura 5.12: Painel de valores por fornecedor do dashboard de um usuário no Grafana

problemas na configuração de segurança entre o keycloak e as outras aplicações, o que acarretou em algumas semanas a mais de desenvolvimento.

Outro empecilho fora o sistema rodar em produção, ou seja, no seu estado final. Para que ele funcione da forma como foi almejado, era preciso que todos os módulos estivessem dentro de contêineres do Docker, mas para isso, era preciso configurar as urls dos micro serviços da aplicação web para poderem conversar entre eles e por conta de protocolos de segurança da rede, foi descoberto que precisaria acrescentar uma camada a aplicação. O que custaria mais tempo de desenvolvimento, o que estava fora do planejamento. Dessa forma, o projeto no estado atual roda apenas localmente na máquina onde fora instalado e o módulo da aplicação web precisou ficar fora do contêiner, dificultando um pouco a instalação do projeto como um todo.



Figura 5.13: Dashboard completo das vendas de um usuário no Grafana

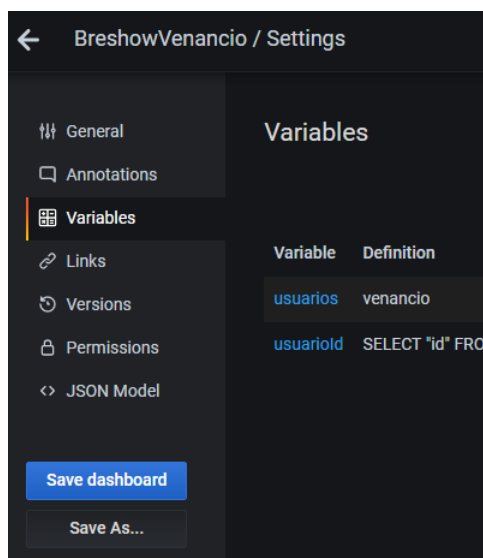


Figura 5.14: Aba de configuração de um dashboard no Grafana

Capítulo 6

Prova de Conceito

Neste capítulo, iremos mostrar a implantação do projeto que está presente no Github, assim como o seu funcionamento na pratica tanto na visão do usuário quanto na do Administrador.

6.1 Implantação

Na fase de implantação dos módulos, é necessário um conhecimento prévio sobre Docker, Git, GitHub, NodeJs e comandos para bash, que é a linha de comando para o sistema Linux. A implantação consiste no funcionamento de toda o ecossistema do sistema e tem o intuito de ser simples para o individuo que irá faze-la. Atualmente, o sistema funciona apenas de forma local, além de toda a sua documentação estar no próprio repositório do projeto caso seja de interesse do implantador.

6.1.1 Docker

Inicialmente precisamos ter o Docker instalado na máquina, ele pode ser encontrado e baixado no link <https://www.docker.com/products/docker-desktop>. Ele é necessário pois todos os módulos e configurações das aplicações já estão pré-configuradas nele e dessa forma, torna a instalação menos complicada e permite que as aplicações estejam desacopladas do sistema operacional do computador.

6.1.2 Github

Próximo passo é a clonagem do repositório do projeto no link <https://github.com/VenancioIgrejas/BreShow> ou rodar a seguinte linha de comando `“git clone https://github.com/VenancioIgrejas/BreShow.git”` no prompt de comando(cmd) ou bash dependendo do sistema operacional. Caso não tenha o *Git* instalado, basta acessar o site <https://git-scm.com/downloads> e seguir o passo a passo de instalação se atendo apenas para qual sistema operacional utiliza.

6.1.3 Npm

Por conta da aplicação web rodar em Node.js, é preciso que o seu gerenciador de pacote esteja instalado também. Dessa forma, basta entrar no link <https://nodejs.org/en/download/> e baixar a versão mais recente. Uma vez instalado, basta rodar o comando `“node -version”` e `“npm -version”` em um *prompt de comando*, se em ambos os casos apareçam a letra V e uns números, significa que a instalação do npm foi um sucesso.

6.1.4 Projeto Local

Na fase atual do projeto, ele não foi feito para rodar em produção pois ainda há algumas alterações de segurança e de rota a serem feitas, contudo ele funciona perfeitamente para rodar localmente.

6.1.4.1 Composição do docker

Uma vez que o Docker esteja instalado, deve-se rodar o comando `“docker-compose up”` na raiz da solução para que os serviços sejam criados e configurados. Esse processo pode demorar um pouco pois serão instanciados 4 serviços ao mesmo tempo que são eles: a imagem do Keycloak, do Grafana, do Angular12 e do NestJS.

6.1.4.2 Rodando a aplicação web

Primeiramente, precisa-se esperar a composição do docker terminar as configurações dos serviços, o que demora entorno de 3 minutos. Apenas após o seu término, é possível rodar a aplicação web e para isso, escreva o comando na linha de comando da

raiz do projeto “./installApp.sh”. Esse comando irá instalar todas as dependências e os módulos a serem utilizados pelo cliente e servidor da aplicação web, assim como configurações de compilação e tabelas de banco, tudo automatizado.

Após a instalação da aplicação, basta rodar o comando na linha de comando da raiz do projeto “./startDevApp.sh” que irá iniciar os dois micro-serviços da aplicação web e estarão prontos para o uso. Caso não tenha alterado algum arquivo de ambiente, a aplicação estará rodando no caminho <http://localhost:4200/>

6.2 Usuário

Nesse ponto, consideramos usuário o indivíduo que irá colocar as informações sobre fornecedores, Categoria dos produtos e as vendas de cada produto, lembrando que é obrigatório a informação sobre qual categoria aquele produto se encontra e qual o seu fornecedor. Além do mais, cada usuário tem acesso apenas as informações dos seus produtos. Já na parte de dashboard, esse controle é feito pelo administrador.

Quando o usuário acessar o link <http://localhost:4200/>, ele será redirecionado para a tela de login do sistema, afinal o mesmo ainda não está logado. Conforme podemos observar na figura 6.1

6.2.1 Criação de Conta

Inicialmente, o usuário precisa-rá criar uma conta. Para isso basta clicar no texto em azul escrito “*Register*” conforme aparece na figura 6.1. Ao clicar, será direcionado para uma tela de cadastro e nela, precisará adicionar informações pessoais tais como nome, e-mail, usuário, senha e dentre outros. Um exemplo a ser representado está na figura 6.2

6.2.2 Tela principal

Na figura 6.3 mostra a tela principal da aplicação web do projeto do trabalho apos a criação do usuário de exemplo, nele é possível visualizar no canto superior esquerdo as abas por onde podemos navegar, que está marcado de vermelho. No

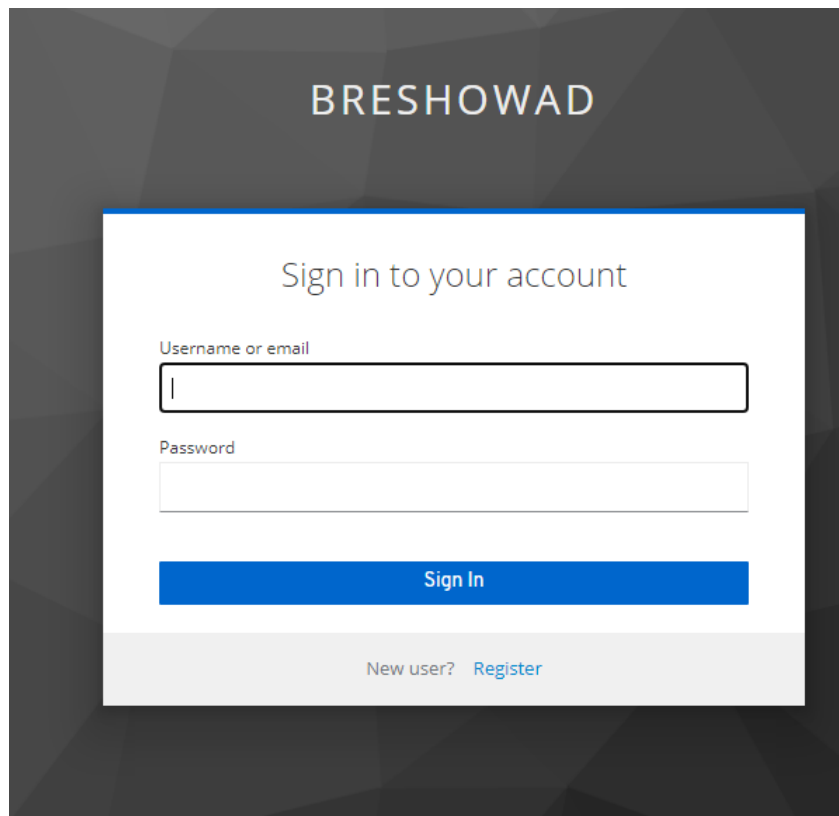


Figura 6.1: Página para entrar na aplicação web.

canto superior direito circulado de azul está as informações do nome do usuário e um botão para se deslogar da aplicação. Por fim, no centro circulado de laranja está um resumo sobre os produtos, as categorias e os fornecedores já cadastrados pelo usuário e seu intuito é de uma rápida analisada no âmbito micro nas tabelas caso precise.

Para acessar as paginas relacionadas aos produtos vendidos tais como, Produto, Categoria e Fornecedor, basta clicar na botão desejado na parte superior esquerda indicado com circulo vermelho.

6.2.3 Tela de Categoria

A tela de categoria apresenta uma tabela com a lista de todas as categorias adicionadas pelo usuário e um botão de adicionar logo a cima da lista para criar um registro novo de categoria, conforme mostrado na figura 6.4.

BRESHOWAD

Register


First name
Venâncio

Last name
Igrejas

Email
venancio@teste.br

Username
venancio

Password
.....

Confirm password
..... 

[« Back to Login](#)

Register

Figura 6.2: Página de cadastro da aplicação web.

O intuito da categoria é ser um texto livre onde o usuário possa criar a categoria que quiser e não se prender a categorias pré-configuradas ou criadas de forma padrão, como, por exemplo, gênero, objeto e cor.

6.2.3.1 Criando uma categoria

Ao clicar no botão de adicionar, irá abrir uma janela flutuante onde o usuário deverá preencher com as informações relacionadas aos itens de vendas que futuramente serão adicionados em outra tela. A categoria serve para agrupar os produtos na hora de adicionar eles no sistema. A janela flutuante apresenta apenas um campo chamado “Nome”, esse campo significa qual o nome da categoria a ser criado. Na

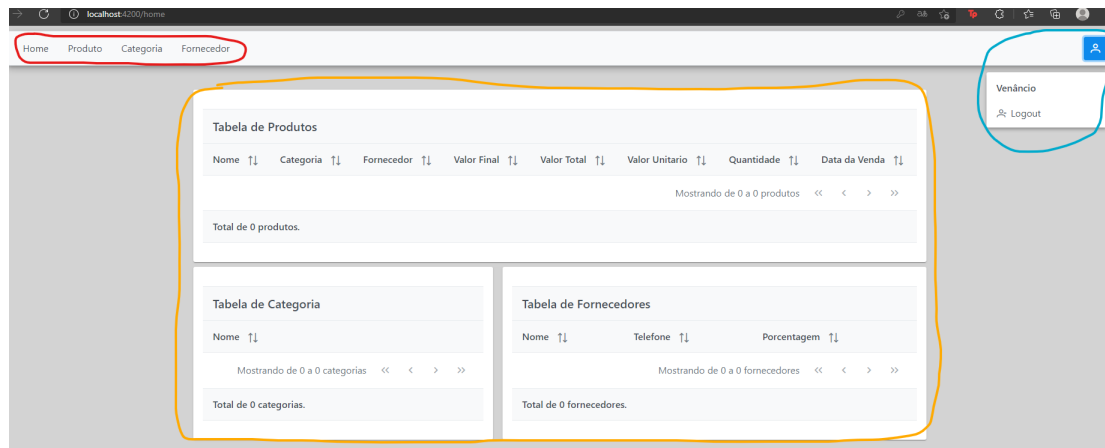


Figura 6.3: Página principal da aplicação web.

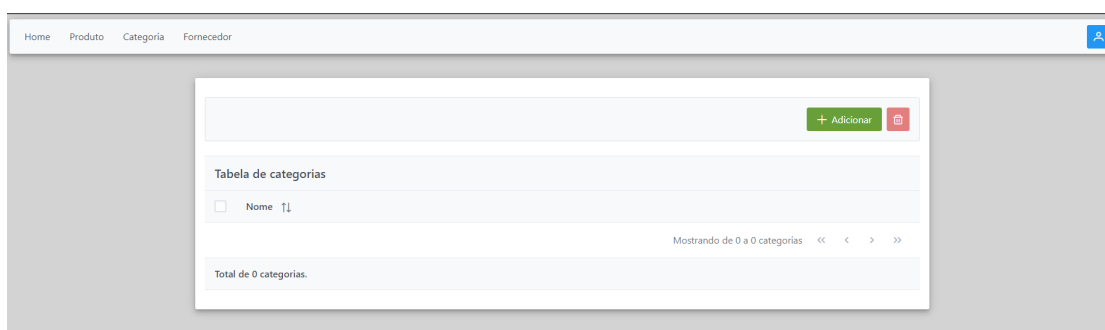


Figura 6.4: Página da categoria da aplicação web.

figura 6.5, estamos criando uma categoria chamada de blusa, após clicar em salvar, aparece a categoria criada na lista da tabela (figura 6.6)

6.2.3.2 Editando ou deletando uma categoria

Para alterar o nome de uma categoria, basta clicar no botão redondo verde com um simbolo de um lápis que está na mesma horizontal que o nome da categoria que deseja editar (figura 6.6). Ao clicar, irá abrir a mesma janela flutuante que a de adição com o campo nome já preenchido. Basta alterá-lo e salvar, após isso, a tela irá recarregar e a alteração completa.

Para deletar a categoria, basta clicar no botão redondo laranja que está no mesmo nível que o nome da categoria desejada (figura 6.6). Ao clicar, a categoria e *todos os registros relacionados a ela* serão deletados do sistema e a tela será recarregada com a nova alteração.

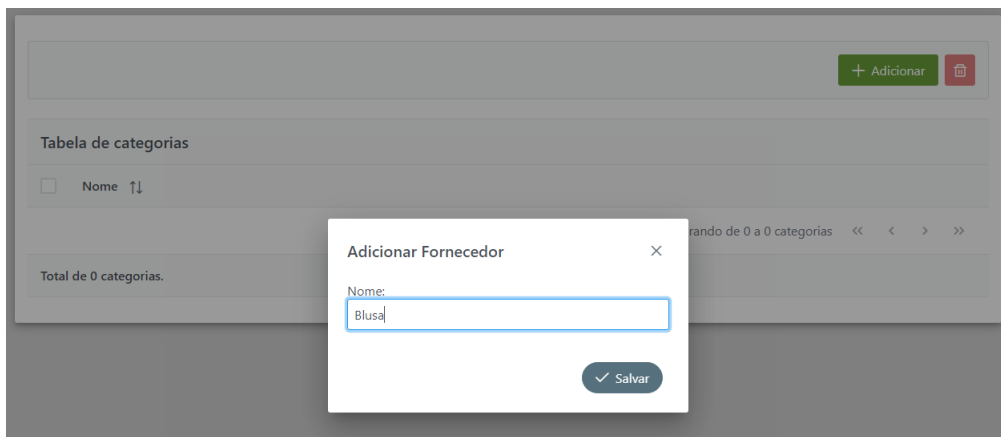


Figura 6.5: Janela flutuante da categoria.

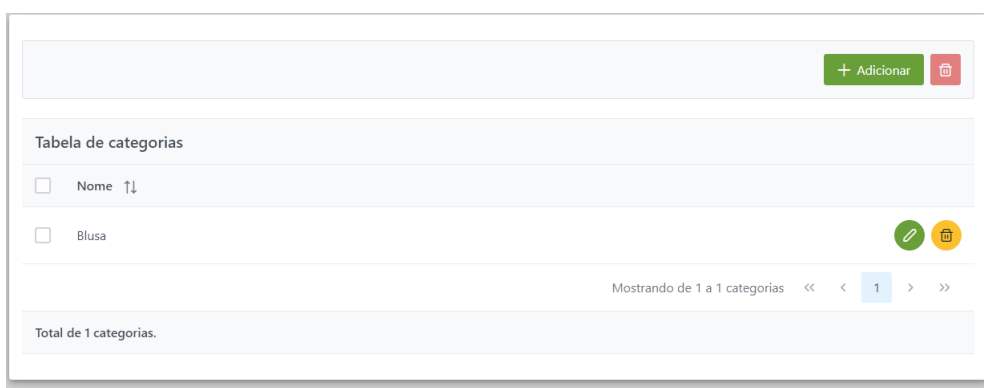


Figura 6.6: Tabela da categoria listando a Blusa após ter sido criada.

6.2.4 Tela de Fornecedor

A tela de fornecedor se assemelha com da categoria na estrutura como pode ser visto na figura 6.7, o que muda são os campos na janela flutuante ao adicionar ou editar um fornecedor.

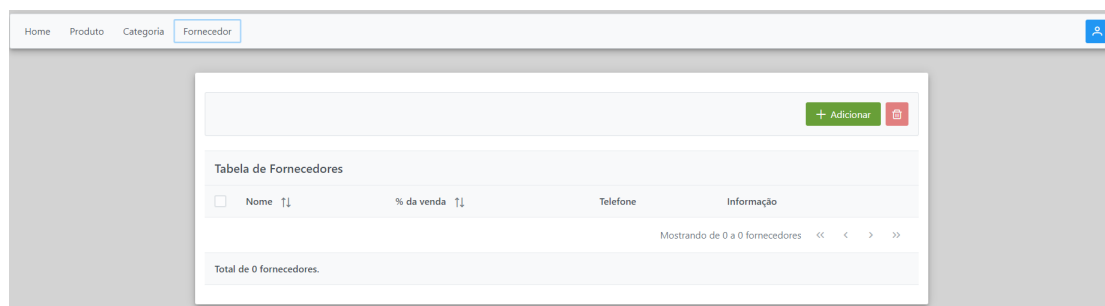


Figura 6.7: Página dos fornecedores na aplicação web.

6.2.4.1 Criando, Editando e deletando um fornecedor

Como o processo de criação, edição e remoção de um fornecedor é muito parecido com a de categoria, iremos apenas mostrar o que é diferente. No caso da janela flutuante, podemos observar na figura 6.8 que existem outros campos a serem preenchidos, tais como o nome do fornecedor, a porcentagem do valor da venda que iria pro fornecedor, o seu telefone para contato e uma informação breve. Após o preenchimento obrigatório das informações, deve-se apertar em “Salvar” caso desejasse adicionar na aplicação ou no “X” no canto superior a direita para fechar a janela.



A imagem mostra uma janela flutuante intitulada "Adicionar Fornecedor" com um botão de fechar "X" no canto superior direito. O formulário contém os seguintes campos:

- Nome:** Um campo de texto com o valor "Josimar".
- Porcentagem da venda:** Um campo de texto com o valor "10%".
- Telefone:** Um campo de texto com o valor "(21) 9 9999-9999".
- Informações:** Um campo de texto com o valor "isso é uma informação".

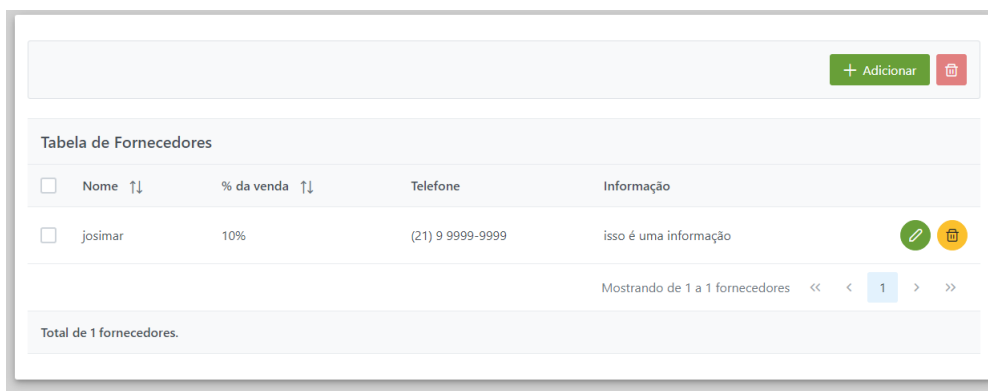
No canto inferior direito da janela, há um botão verde com o ícone de uma seta para cima e o texto "Salvar".

Figura 6.8: Janela flutuante da tela de fornecedores.

Após uma adição ou edição, a lista de fornecedores será atualizada na página principal do fornecedor e poderemos ver as informações alteradas conforme a figura 6.9. No exemplo mostrado, podemos observar que o Josimar foi criado e as suas informações mais importantes estão à mostra para uma rápida análise.

6.2.5 Tela de Produtos

A tela de produtos se assemelha com a de categoria na estrutura como pode ser visto na figura 6.10, o que muda são os campos na janela flutuante ao adicionar ou



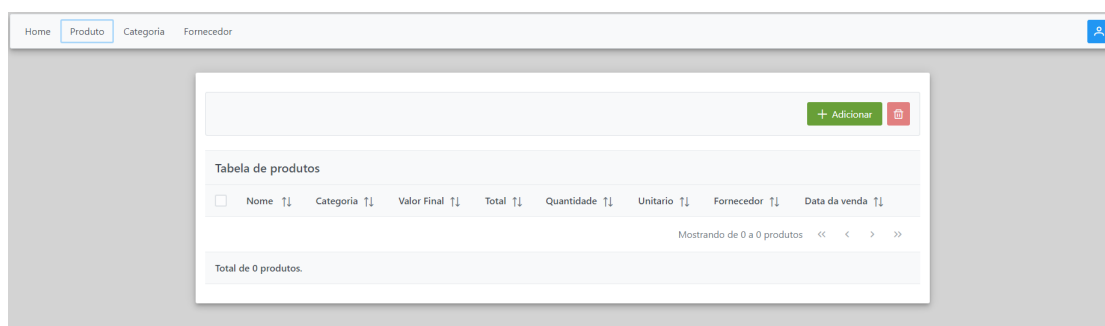
<input type="checkbox"/>	Nome ↑↓	% da venda ↑↓	Telefone	Informação
<input type="checkbox"/>	josimar	10%	(21) 9 9999-9999	isso é uma informação

Mostrando de 1 a 1 fornecedores

Total de 1 fornecedores.

Figura 6.9: Tabela de fornecedores listando o novo fornecedor após ter sido criado.

editar um produto e que todo produto precisa estar relacionado a uma categoria e a um fornecedor obrigatoriamente.



<input type="checkbox"/>	Nome ↑↓	Categoria ↑↓	Valor Final ↑↓	Total ↑↓	Quantidade ↑↓	Unitário ↑↓	Fornecedor ↑↓	Data da venda ↑↓
--------------------------	---------	--------------	----------------	----------	---------------	-------------	---------------	------------------

Mostrando de 0 a 0 produtos

Total de 0 produtos.

Figura 6.10: Página dos produtos na aplicação web.

6.2.5.1 Criando, Editando e deletando um produto

Como o processo de criação, edição e remoção de um produto é muito parecido com a de categoria, iremos apenas mostrar o que é diferente. No caso da janela flutuante, podemos observar na figura 6.11 que existem outros campos a serem preenchidos, tais como o nome do produto, a categoria a qual ele pertence, qual o seu fornecedor, valor da venda por preço unitário, a quantidade, a data em que esse produto foi vendido ou a data que foi adicionado no sistema e uma informação breve. Após o preenchimento obrigatório das informações, deve-se apertar em “Salvar” caso desejasse adicionar na aplicação ou no “X” no canto superior a direita para fechar a janela.

Após uma adição ou edição, a lista de fornecedores será atualizada na página principal do produto e poderemos ver as informações alteradas conforme a figura

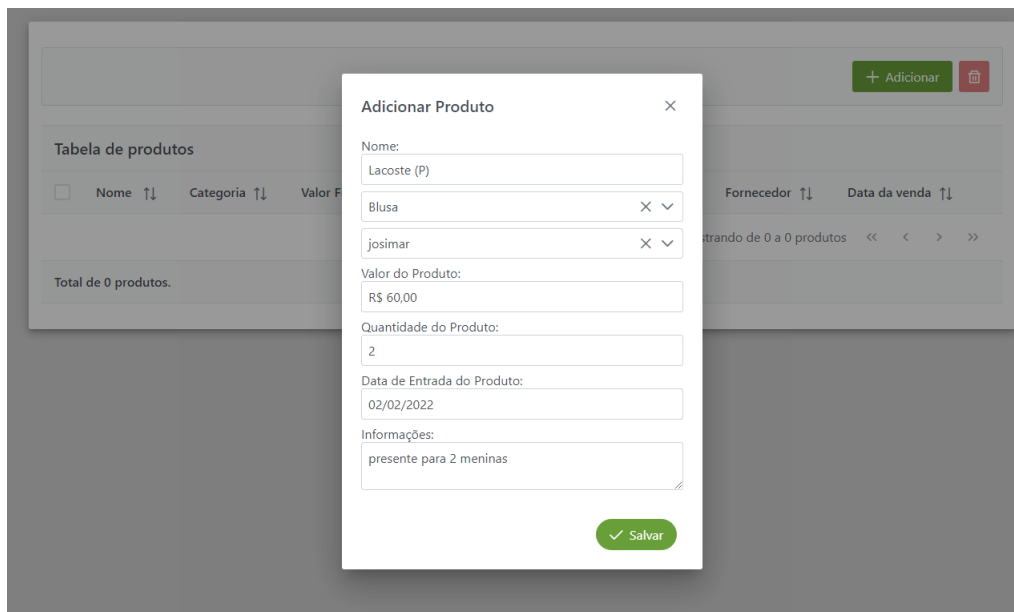


Figura 6.11: Janela flutuante da tela de produtos.

6.12. No exemplo mostrado, podemos observar que a blusa “Lacoste (p)” foi criado e as suas informações mais importantes estão a mostra para uma rápida análise, além do fato de termos informações amais como o valor total da venda (valor da venda x quantidade) e o valor final que é o total menos a parte que irá pro fornecedor, que também é mostrado na tabela com o nome de “Fornecedor”.

	Nome ↑↓	Categoria ↑↓	Valor Final ↑↓	Total ↑↓	Quantidade ↑↓	Unitario ↑↓	Fornecedor ↑↓	Data da venda ↑↓
<input type="checkbox"/>	Lacoste (P)	Blusa	R\$ 108,00	R\$ 120,00	2	R\$ 60,00	josimar	02/02/2022

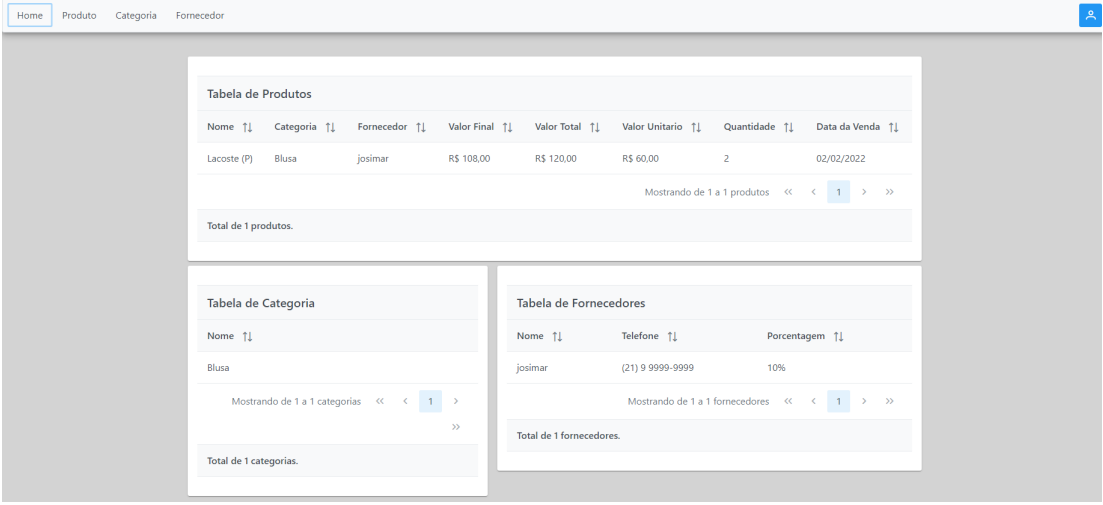
Mostrando de 1 a 1 produtos

Total de 1 produtos.

Figura 6.12: Tabela de produtos listando o novo produto após ter sido criado.

Ao termino de todas as adições dos dados, podemos ir para a pagina inicial que veremos um resumo de todos os dados já inseridos pelo usuário, tais como fornecedores, categorias e produtos (figura 6.13). Lembrando que essa visualização é apenas

de forma micro e para uma análise mais profunda no âmbito macro, deve-se utilizar a aplicação de dashboard que será discutida na próxima sessão.



The screenshot shows a web application interface with a navigation bar at the top containing links for Home, Produto, Categoria, and Fornecedor. The main content area displays three tables:

Nome	Categoria	Fornecedor	Valor Final	Valor Total	Valor Unitario	Quantidade	Data da Venda
Lacoste (P)	Blusa	josimar	R\$ 108,00	R\$ 120,00	R\$ 60,00	2	02/02/2022

Mostrando de 1 a 1 produtos

Total de 1 produtos.

Nome
Blusa

Mostrando de 1 a 1 categorias

Total de 1 categorias.

Nome	Telefone	Porcentagem
josimar	(21) 9 9999-9999	10%

Mostrando de 1 a 1 fornecedores

Total de 1 fornecedores.

Figura 6.13: tela principal listando todas as informações adicionadas do usuário em tabelas distintas.

6.2.6 Tela de Dashboard

Para a análise da aplicação, utilizaremos o Grafana, que é uma aplicação web de análise de código aberto multiplataforma e visualização interativa da web e será considerado o nosso módulo de visualização de dados. Para acessá-lo, devemos colocar o link <http://localhost:8084> no nosso navegador e aparecerá a tela de login dele. Na figura 6.14, podemos ver a tela padrão de login dele e se entrarmos com o usuário e senha criados pelo administrador, podemos ver a tela principal com as informações macro que acabamos de criar na aplicação web (figura xxx).

6.2.6.1 Tela principal

De acordo com a figura 6.15, a tela principal apresenta inúmeras informações sobre a saúde das vendas do empreendimento. A tela é subdividida em agrupamentos de painéis, campos envolvidos em laranja, em cada uma delas é possível clicar e expandi-las. A primeira mostra 3 listas sobre informações gerais das vendas, categorias e fornecedores, igualmente como está na aplicação web (figura 6.16). A segunda já está expandida por padrão e mostra informações macro como os valores totais e reais ganhos em um gráfico de série temporal, quantidade de produtos vendidos

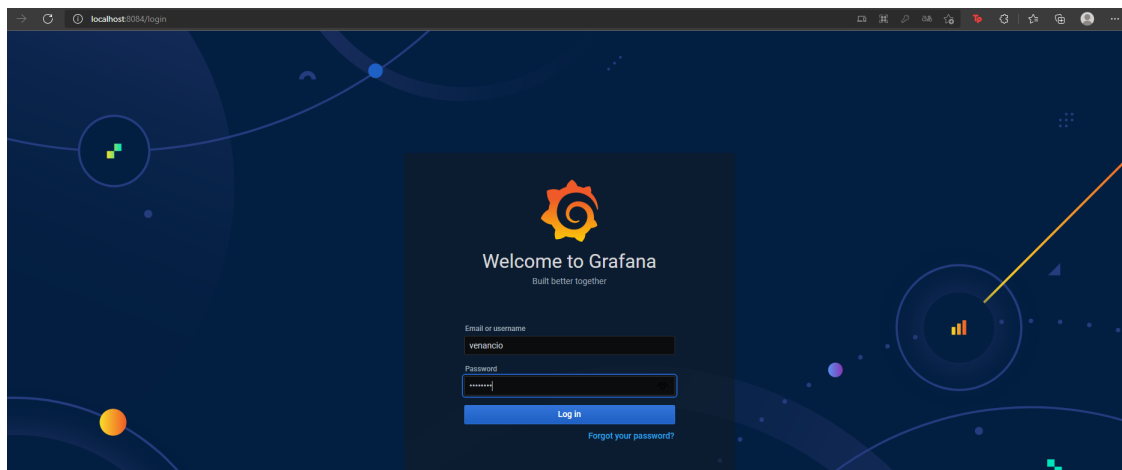


Figura 6.14: Tela de login do módulo de análise.

também no mesmo tipo de gráfico e valor quantitativo de quantas informações foram adicionadas na aplicação. A terceira e quarta são parecidas no formato, elas mostram o lucro total por categoria e valor total do fornecedor em série temporal, gráfico de pizza e lista (figura 6.16).

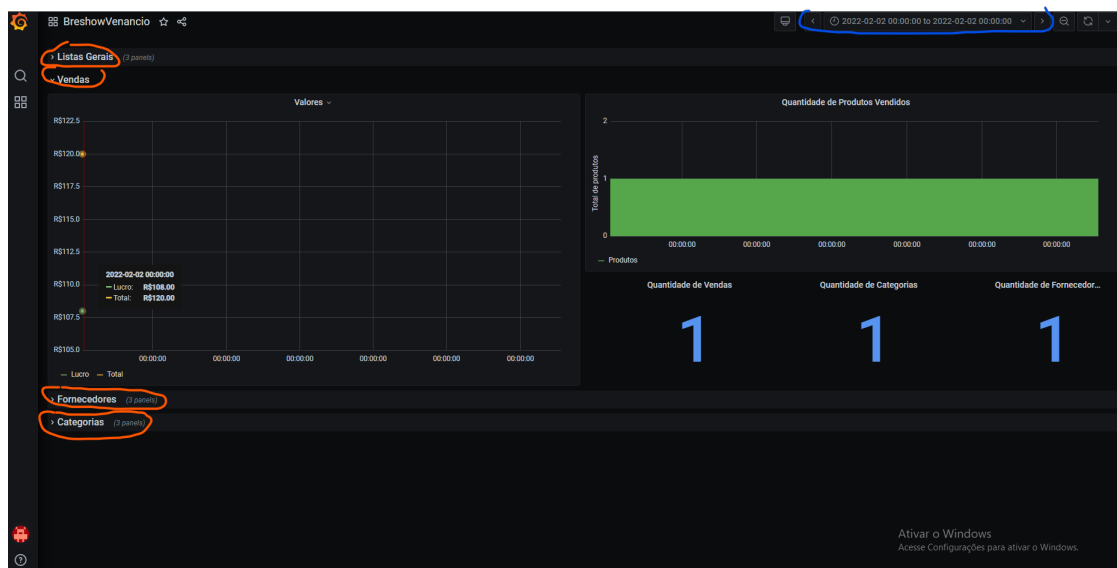


Figura 6.15: Tela principal do usuário.

Além disso, todos os dados e painéis estão interligados a um filtro temporal que pode ser encontrado circulado de azul no topo da tela a direita na figura 6.15. Uma vez que esse filtro é alterado para o espaço temporal desejado, todos os valores são recalculados e filtrados para aquele intervalo, ou seja, o usuário pode pegar informações totais de um dia, um mês, mês passado, três meses ou um ano. Além

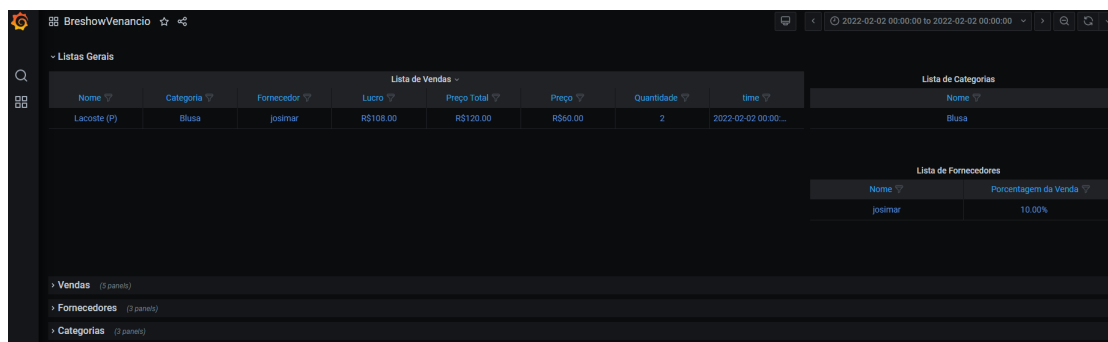


Figura 6.16: Painel de listas gerais expandido.

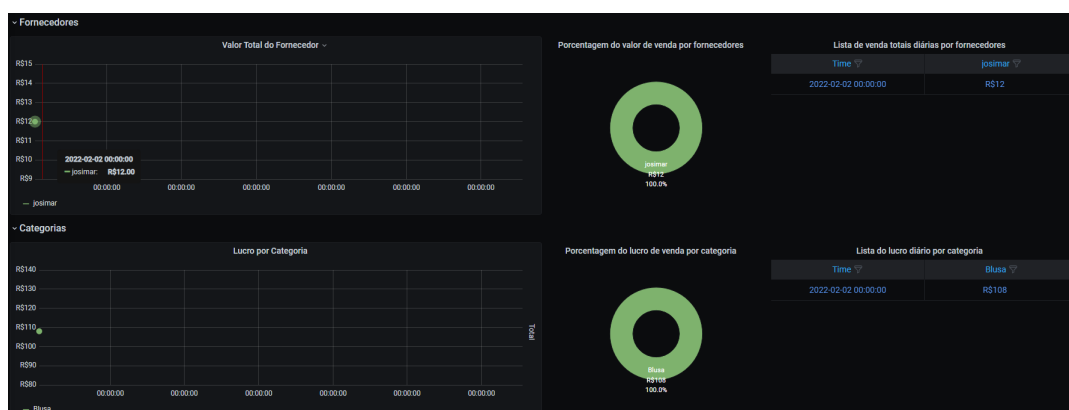


Figura 6.17: Painel de fornecedores e categorias expandido.

disso, o sistema é atualizado automaticamente e pode ser configurado no botão com a seta pra baixo o tempo de atualização automática dos dados conforme a figura 6.18, por exemplo, se for setado para 5 segundos, o sistema irá recarregar as informações aplicadas na aplicação web nele.

6.3 Administrador

Uma conta de administrador serve para gerenciar as contas dos usuários da aplicação web utilizando o módulo de gerenciamento de usuários. Além disso, é responsável pela administração de criação e manutenção das contas dos usuários no módulo de análise e gráficos (dashboard).

6.3.1 Tela de Gerenciamento de Usuários

Para esse módulo, utilizá-se o sistema Keycloak como estrutura principal do nosso módulo de gerenciamento de usuários, um produto de software com a finalidade de

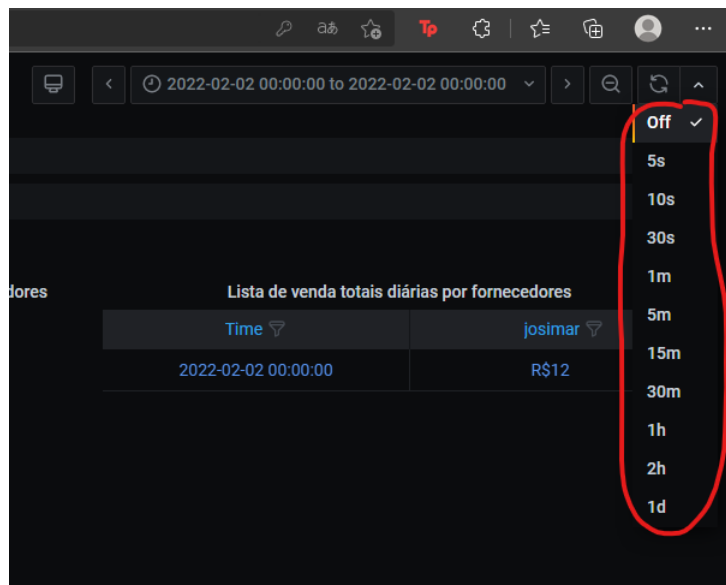


Figura 6.18: Painel de fornecedores e categorias expandido.

gerenciamento de identidades e acessos voltados para aplicativos. O administrador utiliza uma senha e usuário padrão que está na documentação do módulo no Github para entrar no gerenciador.

Primeiramente, o indivíduo deve acessar o link <http://localhost:28080/> e clicar em “Administration Console” (figura 6.19) para entrar na tela de login da administração (figura 6.20), após colocar os dados de administrador que estão, por padrão, na documentação do GitHub e clicar em entrar, uma tela principal irá aparecer com várias opções de navegação numa coluna a esquerda (figura 6.22).

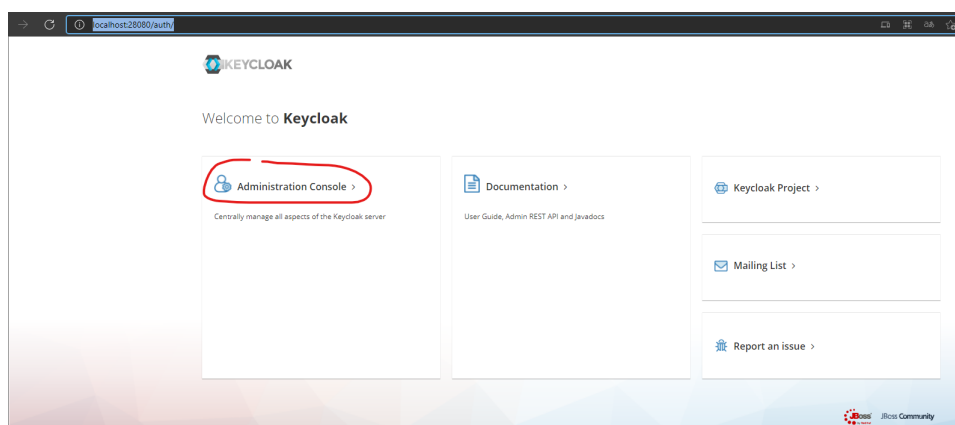


Figura 6.19: Pagina Inicial do Keycloak.

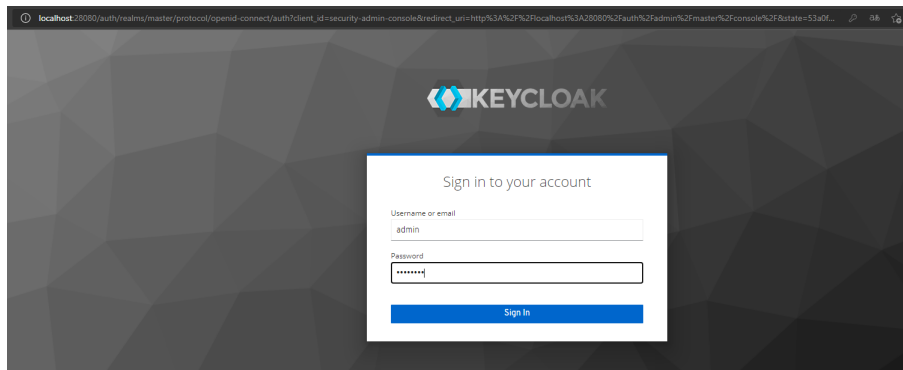


Figura 6.20: Pagina de login da área de administração.

De acordo com a figura 6.22, o círculo em vermelho representa as opções de configurações do BreshowAD, tais como, configuração de segurança de rota, ajuste dos acessos de aplicações externas, roles, grupos. O círculo azul representa a parte voltada para o gerenciamento de usuários do módulo, nele é possível visualizar informações dos usuários, assim como, assim como ter o controle para fazer o seu logout de aplicações, resetar o seu login ou até mesmo criar uma nova senha. Vale ressaltar que a parte de segurança é muito importante, logo não existe a possibilidade do administrador ver a senha que o usuário colocou, apenas resetar ela ou criar uma nova. O círculo amarelo representa a aba de “Users” e nele está o usuário que acabamos de criar no subtópico acima com as informações importantes colocada por este usuário.

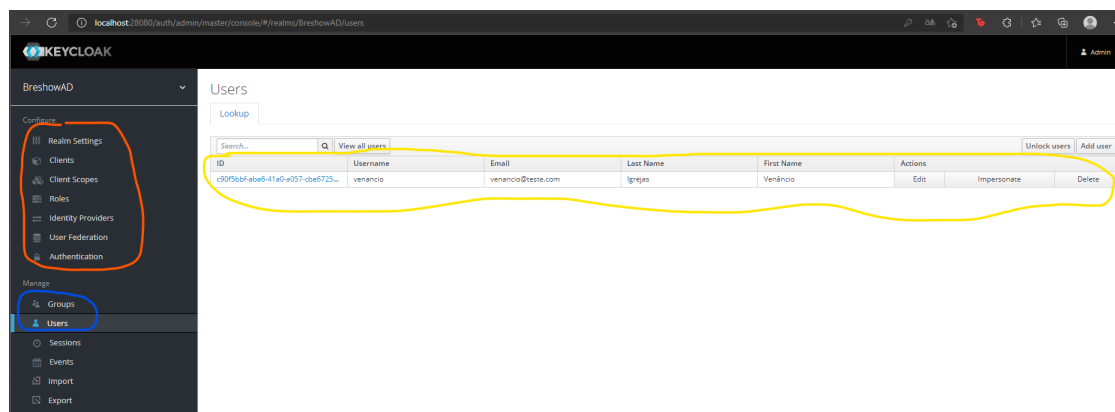


Figura 6.21: Pagina Principal do Keycloak.

Ao clicar no usuário que aparece na figura 6.22, abrimos a tela daquele usuário com as suas informações pessoal, onde podemos editar caso necessário. temos também

a parte de credenciais que é reponsavel pelo reset da senha e o seu gerenciamento (figura 6.23), os grupos de usuários aos quais ele pode pertencer e o campo de sessão (“Sessions”) que permite você ver quais aplicações ele está logado e se está ativo, além de poder encerrar a sessão, obrigando-o a se deslogar.

The screenshot shows the 'Venancio' user management page. At the top, there's a breadcrumb trail: Details (selected), Attributes, Credentials, Role Mappings, Groups, Consents, Sessions. Below this, the user's details are listed in a form-like structure:

- ID: c90f5bbf-aba6-41a0-a057-cbe672549795
- Created At: 1/31/22 5:25:01 PM
- Username: venancio
- Email: venancio@teste.com
- First Name: Venâncio
- Last Name: Igrejas
- User Enabled: ON (toggle)
- Email Verified: OFF (toggle)
- Required User Actions: Select an action...
- Impersonate user: Impersonate button
- Buttons: Save, Cancel

Figura 6.22: Tela de gerenciamento do usuário Venâncio.

The screenshot shows the 'Venancio' user management page with the 'Credentials' tab selected. The breadcrumb trail is: Details, Attributes (selected), Credentials, Role Mappings, Groups, Consents, Sessions.

Manage Credentials

Position	Type	User Label	Data	Actions
< >	password		Show data...	Delete Save

Reset Password

Password:

Password Confirmation:

Temporary: ON (toggle)

Reset Password button

Credential Reset

Reset Actions: Select an action...

Expires in: 12 Hours (dropdown)

Reset Actions Email: Send email button

Figura 6.23: Pagina de credenciais do usuário Venâncio.

6.3.2 Tela de Dashboard

Para a aplicação de dashboard, o administrador é importante para gerenciar o acesso de cada usuário em suas respectivas vendas. Ele tem o acesso a todos os Dashboard dos usuários, pois é ele os disponibiliza e ajusta. A criação de dashboard para um usuário novo é feito manualmente, em contra partida, é só o administrador copiar uma já existente e alterar a variável global do painel para o nome do usuário que ele deseja criar.

Na tela de administrador do servidor (figura 6.24) é a area em que se cria usuários para a aplicação de dashboard, vale ressaltar que o usuário da aplicação web não é o mesmo que a do dashbord e pra isso, o administrador deve cria-los nessa tela.

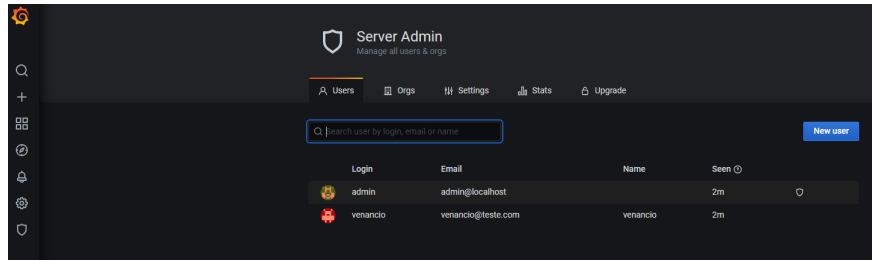


Figura 6.24: Área de gerenciamento de usuários do módulo de dashboard.

Supondo que o usuário foi criado e o painel dele também fora através do método sugerido no início dessa sessão, na tela de configuração do dashboard é possível adicionar o usuário como apenas leitor e assim, ele só poderá analisar os gráficos de vendas dele, conforme mostrado na figura 6.24.

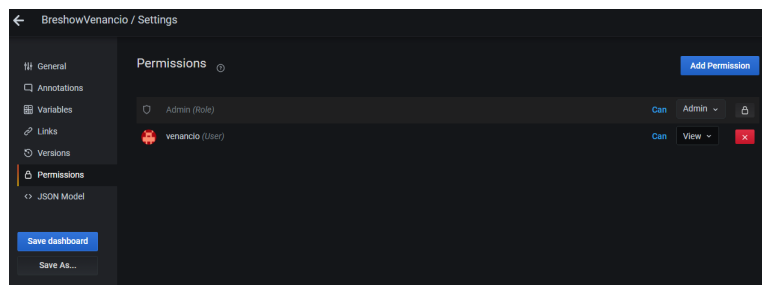


Figura 6.25: Área de configuração .

Capítulo 7

Conclusões e Trabalhos Futuros

O seguinte capítulo está dividido em duas sessões, a primeira contemplará a conclusão do trabalho, mostrando as dificuldades e problemas ao longo do desenvolvimento do projeto, assim como mudança nas estratégias de escolhas e um resumo. A segunda parte está voltada para os trabalhos futuros, onde é abordado quais serão os próximos passos e possíveis melhorias no sistema para que possa torna-lo mais robusto e intuitivo para o usuário.

7.1 Conclusão

Na fase atual o qual se encontra, O projeto consegue ajudar o micro empreendedor a analisar as suas vendas de forma macro através da aplicação de dashboard. Para a inserção de dados na base, a aplicação web está minimamente funcional e operacional, onde o usuário consegue fazer as operações básicas de inserção, alteração, visualização e deletar dos dados das suas respectivas vendas. Assim como, o administrador consegue orquestrar e gerenciar usuários da aplicação e do sistema como um todo.

Mesmo com todos os problemas ao longo do desenvolvimento, o projeto funciona da forma como foi proposto, onde ele é mais robusto e mais simples de ser utilizado do que o Excel, porém não apresenta uma complexidade tão grande e desnecessárias das ERPs no mercado que são voltadas para grandes empresas, além disso o projeto é totalmente gratuito e de código aberto. O usuário que for utilizá-lo precisará

apenas preencher os dados dos fornecedores, categorias e das vendas, sendo eles intuitivo na plataforma web e no final, poderá ver toda a análise macro das suas vendas no módulo de dashboard. Apenas na hora da instalação do projeto e na sua configuração que pode haver algo mais complicado de configurar, porém temos a documentação para isso.

7.2 Trabalhos Futuros

Para os trabalhos futuros, existem algumas pendências e melhorias do projeto. Primeiramente, as pendências estão relacionados a problemas apresentados ao longo do desenvolvimento e não foram possíveis resolvê-los dentro do prazo estipulado do trabalho. O fato do projeto não poder estar em produção por problemas nas ligações entre os serviços, está relacionado a aplicação web, que são o frontend e o backend, não estarem dentro de contêineres como estão as aplicações de banco de dados, análise e do gerenciador de usuários. Outro ajuste necessário seria a configuração de rotas seguras dentro do keycloak, o gerenciador de usuário, afinal como o projeto roda localmente, não há problema de invasão pela rede, mas ao colocar em produção, é de extrema importância a configuração correta das rotas.

Em relação as melhorias, pode-se criar outros módulos no padrão de micro serviços como módulo de estoque, financeiro e funcionários. Pelo fato de ter sido explorado a descentralização de serviços no sistema, a escolha da arquitetura e framework se torna uma livre escolha do indivíduo. Outro ponto a destacar seria a aplicação do conhecimento de designer da experiência do usuário na aplicação web, afinal apenas existe um protótipo de tela que está apenas funcional, mas sem nenhum acabamento. Ainda falando sobre a aplicação web, é possível adicionar mais recursos como, por exemplo, botão de exportação das tabelas em extensões de arquivos do Excel, alteração de um grupo de vendas que apresentem um mesmo campo, duplicação da venda para evitar rescrever campos iguais inúmeras vezes e dentre outros recursos.

Referências Bibliográficas

- [1] MARTIN, R. C., “The Clean Code Blog”, [urlhttps://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html](https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html), 2012, [Online; accessed 26-April-2022].
- [2] MARZALL, L. F., OTHERS, “Implantação de um sistema de gestão ERP gratuito em uma empresa de pequeno porte com foco na melhoria de indicadores de desempenho da produção”, , 2016.
- [3] FERNANDES, D. L., “Implementação de uma Ferramenta de Gestão de Projetos numa Consultora”, , 2017.
- [4] UTAMI, S. S., SUSILO, H., RIYADI, “Analisis Penerapan Enterprise Resource Planning (ERP)”, *Jurnal Administrasi Bisnis (JAB)*, v. 33, pp. 65–170, Abril 2016.
- [5] PAHL, C., “Containerization and the paas cloud”, *IEEE Cloud Computing*, v. 2, n. 3, pp. 24–31, 2015.
- [6] SATYANARAYANA, S., “Cloud computing: SAAS”, *Computer Sciences and Telecommunications*, , n. 4, pp. 76–79, Dezembro 2012.
- [7] MARTIN, R. C., “The Principles of OOD”, [urlhttp://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod](http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod), 2005, [Online; accessed 22-April-2022].
- [8] OLORUNTOBA, S., “SOLID: The First 5 Principles of Object Oriented Design”, [urlhttps://www.digitalocean.com/community/conceptual_articles/solid-the-first-five-principles-of-object-oriented-design](https://www.digitalocean.com/community/conceptual_articles/solid-the-first-five-principles-of-object-oriented-design), 2020, [Online; accessed 22 – April – 2022].

- [9] NAIDU, D., “SOLID Principles In C”, url[https://www.sharpcorner.com/UploadFile/damubetha/solid-principles-in-C-Sharp/:text=SOLID%20design%20principles%20in%20C%23,Dependency%20Inversion%20Principle%20\(DIP\).](https://www.sharpcorner.com/UploadFile/damubetha/solid-principles-in-C-Sharp/:text=SOLID%20design%20principles%20in%20C%23,Dependency%20Inversion%20Principle%20(DIP).), 2021, [Online; accessed 22-April-2022].
- [10] PAIXAO, J. R. D., “O que é SOLID: O guia completo para você entender os 5 princípios da POO”, url<https://medium.com/desenvolvendo-com-paixao/o-que-%C3%A9-solid-o-guia-completo-para-voc%C3%AA-entender-os-5-princ%C3%ADpios-da-poo-2b937b3fc530>, 2019, [Online; accessed 22-April-2022].
- [11] DEACON, J., “Model-view-controller (mvc) architecture”, *Online*/[Citado em: 10 de março de 2006.] <http://www.jdl.co.uk/briefings/MVC.pdf>, v. 28, 2009.
- [12] HIGOR, “Introdução ao Padrão MVC”, url<https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>, 2013, [Online; accessed 26-April-2022].
- [13] HERNANDEZ, R., “The Model View Controller Pattern – MVC Architecture and Frameworks Explained”, url<https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>, 2021, [Online; accessed 26-April-2022].
- [14] MARTIN, R. C., GRENNING, J., BROWN, S., *et al.*, *Clean architecture: a craftsman’s guide to software structure and design*, n. s 31. Prentice Hall, 2018.
- [15] FONSECA, E. D., “Introdução a Clean Architecture”, url<https://imasters.com.br/back-end/introducao-clean-architecture>, 2018, [Online; accessed 26-April-2022].
- [16] SAUBER, S., “Feature Folder Structure in ASP.NET Core”, url<https://scottsauber.com/2016/04/25/feature-folder-structure-in-asp-net-core/:text=A%20feature%20folder%20structure%20is,poor%20man’s%20feature%20folder%20structure>, 2016, [Online; accessed 26-April-2022].
- [17] SHEHAB, E., SHARP, M., SUPRAMANIAM, L., *et al.*, “Enterprise resource planning: An integrative review”, *Business process management journal*, , 2004.

- [18] KIM, H., BOLDYREFF, C., OTHERS, “Open source ERP for SMEs.”, , 2005.
- [19] TERMINANTO, A., HIDAYANTO, A. N., “Implementation and configurations open source ERP in ecommerce module (A case study on SME)”. In: *8th International Conference on Industrial Engineering and Operations Management, IEOM 2018*, pp. 1224–1233, IEOM Society, 2018.
- [20] WEBERP, “WebERP”, url<https://www.weberp.org/>, 2022, [Online; accessed 21-June-2022].
- [21] COMPIERE, “Compiere”, url<http://www.compiere.com/products/product-demos/>, 2022, [Online; accessed 21-June-2022].
- [22] JFIRE, “Jfire”, url<https://web.archive.org/web/20100910033244/http://www.jfire.net/>, 2022, [Online; accessed 21-June-2022].
- [23] ERP5, “ERP5”, url<https://www.erp5.com/>, 2022, [Online; accessed 21-June-2022].
- [24] ODOO, “Odoo”, url<https://www.odoo.com/>pt_BR, 2022, [Online; accessed 21-June-2022].
- [25] ACCOUNTING, F., “Front Accounting”, url<https://frontaccounting.com/>, 2022, [Online; accessed 21-June-2022].
- [26] OFBIZ, “OFBiz”, url<https://ofbiz.apache.org/>, 2022, [Online; accessed 21-June-2022].
- [27] JPIRE, “JPIRE”, url<http://www.jfire.net/>, 2022, [Online; accessed 21-June-2022].
- [28] XTUPLE, “Xtuple”, url<https://www.xtuple.com/>, 2022, [Online; accessed 21-June-2022].
- [29] LIMANTARA, N., JINGGA, F., “Open source ERP: ODOO implementation at micro small medium enterprises:(A case study approach at CV. XYZ in module purchasing and production)”. In: *2017 International Conference on Information Management and Technology (ICIMTech)*, pp. 340–344, IEEE, 2017.

- [30] MIRAGEM, B., “A Lei Geral de Proteção de Dados (Lei 13.709/2018) e o direito do consumidor”, *Revista dos Tribunais*, v. 1009, 2019.
- [31] GITHUB, “GitHub”, url<https://github.com/>, 2022, [Online; accessed 26-April-2022].
- [32] DOCKER, “Docker”, url<https://www.docker.com/>, 2022, [Online; accessed 26-April-2022].
- [33] “PostgreSQL”, url<https://www.postgresql.org/>, 2022, [Online; accessed 26-April-2022].
- [34] DOCKER, “Docker Hub”, url<https://hub.docker.com/>, 2022, [Online; accessed 26-April-2022].
- [35] HEAT, R., “Keycloak”, url<https://www.keycloak.org/>, 2022, [Online; accessed 26-April-2022].
- [36] AUTH0, “Jwt”, url<https://jwt.io/>, 2022, [Online; accessed 26-April-2022].
- [37] MYSLIWIEC, K., “NestJs”, url<https://nestjs.com/>, 2022, [Online; accessed 26-April-2022].
- [38] INFORMATICS, P., “primeNG”, url<https://www.primefaces.org/primeng/>, 2021, [Online; accessed 26-April-2022].
- [39] GOOGLE, “Angular”, url<https://angular.io/>, 2022, [Online; accessed 26-April-2022].
- [40] “TypeORM”, url<https://typeorm.io/>, 2022, [Online; accessed 26-April-2022].
- [41] LABS, G., “Grafana”, url<https://grafana.com/>, 2022, [Online; accessed 26-April-2022].