UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO HENRIQUE PONTES SILVA

ROMANEIOAPP:
UM SISTEMA PARA CONTROLE DE PEDIDOS NO ATACADO

CAMPINA GRANDE - PB

2023

EDUARDO HENRIQUE PONTES SILVA

ROMANEIOAPP:
UM SISTEMA PARA CONTROLE DE PEDIDOS NO ATACADO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Wilkerson de Lucena Andrade.

CAMPINA GRANDE - PB

2023

EDUARDO HENRIQUE PONTES SILVA

ROMANEIOAPP:
UM SISTEMA PARA CONTROLE DE PEDIDOS NO ATACADO

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Wilkerson de Lucena Andrade

Orientador – UASC/CEEI/UFCG

Professor Dr. Carlos Eduardo Santos Pires

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de Fevereiro de 2023.

CAMPINA GRANDE - PB

ABSTRACT

The growth of the horticultural sector was a decisive factor for the implementation of systems for information storage and management. However, there is still a lack of accessible systems for users of this sector, considering that a large part of them are totally or functionally illiterate. With the aim of meeting the usual needs (low cost, easy usability and understanding), we propose the RomaneioApp. With this tool, it will be possible to guarantee efficiency in the management of orders in a simple and clear way.

RomaneioApp: Um sistema para controle de pedidos no atacado

Eduardo Henrique Pontes Silva
E. H. Pontes
eduardohenrique38@live.com
eduardo.henrique.silva@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

Dr. Wilkerson de Lucena Andrade
wilkersonatcomputacao.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

RESUMO

O crescimento do setor hortifrutigranjeiro foi um fator decisivo para a implementação de sistemas para armazenamento e gerenciamento de informações. Contudo, ainda faltam sistemas acessíveis aos usuários de tal setor, tendo em vista que grande parte são analfabetos totais ou funcionais. Com o intuito de suprir as necessidades usuais (baixo custo, fácil usabilidade e compreensão), propomos o RomaneioApp. Com esta ferramenta, será possível garantir a eficiência na gestão dos pedidos de forma simples e clara.

ACM Reference Format:
Eduardo Henrique Pontes Silva, E. H. Pontes, and Dr. Wilkerson de Lucena Andrade. 2023. RomaneioApp: Um sistema para controle de pedidos no atacado. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Repositórios
https://github.com/eduhique/romaneioApp
https://github.com/eduhique/romaneioApp-frontend
[1]

1 INTRODUÇÃO

A comercialização de produtos é dividida em duas categorias distintas: atacado e varejo. Ambas possuem características e públicos próprios. O comércio atacadista oferece produtos em grandes quantidades, geralmente para revenda, enquanto o varejo vende em quantidades menores, mas diretamente ao público-alvo. Juntos, esses modelos de negócios atendem a diferentes demandas de mercado e movimentam a economia.

Dentro do mercado de comercialização de produtos, podemos destacar o ramo hortifrutigranjeiro. A atividade em hortifrutigranjeiros está ligada ao mercado produtor de frutas, legumes e hortaliças. Nos últimos anos, tal mercado tem crescido exponencialmente. Devido a esse crescimento, a necessidade de informatização e utilização de sistemas tornou-se primordial. A implementação de sistemas é essencial para aumentar a produtividade, segurança e operações eficientes.

Muitos dos sistemas disponibilizados oferecem aplicações genéricas e robustas, dificultando o acesso por usuários desse ramo, que na maioria das vezes possuem baixa escolaridade. As aplicações para venda direta que são chamadas de "Ponto de Venda", geralmente são pagas, caras e difíceis de manusear. Logo, a falta de conhecimento formal e o alto custo, tornam as aplicações inacessíveis fazendo com que os usuários optem por fazer o gerenciamento à mão. A finalidade do presente trabalho é realizar o desenvolvimento de um sistema simplificado, de baixo custo, fácil usabilidade e adequado ao mercado de atacado hortifrutigranjeiro.

O texto é dividido em cinco partes: a seção um apresenta uma introdução sobre o sistema e os fomentos de seu desenvolvimento; a seção dois aborda uma aplicação semelhante à proposta neste trabalho; a seção três aborda a solução que o sistema representa trazendo informações coletadas junto a usuários e explorando a experiência do autor; a seção quatro apresenta o relato da experiência de desenvolvimento do sistema; e na seção cinco expomos propostas de como avançar no desenvolvimento do sistema como ideias de trabalhos futuros.

2 TRABALHOS RELACIONADOS

Pode-se destacar um sistema semelhante à aplicação proposta neste trabalho: Hortigestão. O Hortigestão[1] é um sistema que conta com acompanhamento de estoque, gerenciamento de notas fiscais, gestão de contas e emissão de boletos.

Apesar de ser uma ferramenta aparentemente completa, apresenta algumas deficiências, tais como:

- A falta de informações no site e a dificuldade em obtê-las deixa em dúvida se o sistema proposto realmente supre as necessidades;
- O fato do sistema incluir outros módulos que não tem relação direta com a gestão de pedidos desviam o foco da aplicação e o torna pesado a longo prazo;
- A aplicação integra funcionalidades como emissão de nota fiscal e controle de caixa, podendo aumentar o custo. Além

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

disso, é dividida em planos que se baseiam na noção de licenciamento de dispositivos.

3 SOLUÇÃO

O Romaneio[18] é um documento utilizado no transporte logístico que tem por objetivo sumarizar os produtos transportados por um veículo de carga. Este documento é bastante utilizado para fiscalização aduaneira e interestadual, bem como na conferência destes produtos na entrega final. Os profissionais do ramo hortifrutigranjeiro denominam romaneio como sendo o resumo de todos os pedidos de seus clientes, pois se utilizam desse documento para realizar os seus pedidos junto aos fornecedores e também como guia para gerenciamento do seu estoque de produtos durante a venda, tendo em vista que na maioria das vezes os produtos são vendidos enquanto são descarregados do veículo em que são transportados.

O RomaneioApp, que teve seu nome inspirado na definição de romaneio, é um aplicativo de controle de pedidos e romaneios focados no mercado de atacado hortifrutigranjeiro, onde é possível realizar a gestão de pedidos tanto com o foco na venda, como no controle de estoque junto a fornecedores.

No mercado, existem algumas aplicações que realizam funcionalidades semelhantes e todos fazem outras funções além destas, geralmente vinculadas ao sistema tradicional de ponto de venda e gestão comercial. Logo, as funcionalidades principais da aplicação abordadas neste trabalho são tratadas como uma função secundária que na boa parte das vezes não funciona como esperado ou é limitada.

A maioria desses sistemas ou não possuem aplicação mobile/acesso remoto ou este é bastante limitado, prejudicando a portabilidade e forçando com que o usuário tenha uma infraestrutura básica que irá funcionar como uma espécie de servidor. Além disso, esses sistemas são caros e de alto custo mensal.

Portanto, considerando o baixo custo, a portabilidade e simplicidade, a aplicação se apresenta como uma solução simples e objetiva que possui um baixo custo e pode ser adaptada para ambientes remotos. O código fonte base será disponibilizado gratuitamente de maneira integral e os únicos custos são os relativos a implantação e customização da ferramenta.

3.1 Funcionalidades

A experiência prévia de um dos autores juntamente de conversas com familiares que são do ramo hortifrutigranjeiro resultou na base das funcionalidades. A partir dessas conversas foi realizado o levantamento dos requisitos que geraram as funcionalidades basilares do sistema.

3.1.1 Gerenciamento de usuário.

O gerenciamento de usuário é restrito a determinados perfis de usuário. Um usuário do tipo administrador é cadastrado previamente e pode criar outros usuários bem como gerenciá-los. Os perfis disponíveis são: caixa, conferente, gerente e administrador.

3.1.2 Gerenciamento de Tipos primitivos.

O gerenciamento de tipos primitivos permite que o usuário possa criar os tipos unitários mais básicos da aplicação, tais como: quilograma e unidade. Esta funcionalidade tem por objetivo permitir ao usuário poder criar tipos personalizados para os seus produtos, bem como automatizar a conversão de unidade. Por exemplo, se considerarmos que um determinado produto é contabilizado com a unidade saco e cada saco possui 20 quilogramas, logo tem-se dois tipos primitivos, saco e unidade.

3.1.3 Gerenciamento de Produtos.

O sistema permite criar, editar e apagar produtos desde que não estejam vinculados a um pedido. Além do nome do produto é possível criar e gerenciar uma tabela de conversão entre tipos primitivos para permitir a conversão automática do tipo unitário do produto.

3.1.4 Gerenciamento de Cliente.

O sistema permite criar, editar e apagar clientes. Além do nome do cliente, são necessárias informações como tipo de cliente (varejo, atacado ou outros) e bairro/cidade de lotação. Há também campo adicional e opcional de comentários.

3.1.5 Gerenciamento de Romaneio.

No gerenciamento de romaneio, além de ser possível criar, editar e apagar um romaneio, também é possível definir um romaneio como ativo e visualizar um relatório de resumo de todos os pedidos vinculados ao romaneio para facilitar o controle de estoque e compras junto a fornecedores.

3.1.6 Gerenciamento de pedidos.

O usuário da aplicação deve poder criar um pedido vinculando um cliente e um romaneio. O pedido criado serve como estatística para o relatório do romaneio e cada item do pedido pode ter qualquer uma das unidades e escolher qualquer um dos tipos primitivos cadastrados no produto escolhido, dando mais liberdade ao usuário.

3.2 Arquitetura

A arquitetura do sistema segue o padrão MVC (Model-View-Controller) com as referentes camadas de frontend, backend e banco de dados, efetuando o referido padrão e comunicando-se entre si, conforme ilustrado na Figura 1.

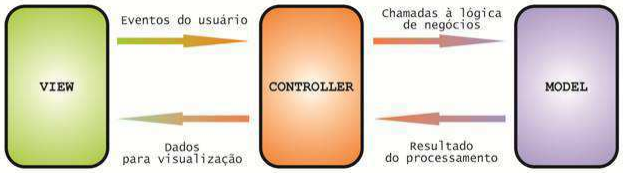

Figura 1: Arquitetura do sistema seguindo o padrão MVC[10].

3.2.1 *Frontend.*

A princípio, a interface do usuário tinha sido implementada em Angular[2] 13 que é um framework voltado ao desenvolvimento de sistemas de informação para a linguagem de programação Javascript. A fim de refinar a estratégia Lazy loading houve a migração para o Angular 15. A técnica Lazy loading (Figura 2) identifica recursos e carrega-os quando há necessidade, melhorando assim, o tempo de carregamento da página. Além disso, traz uma estrutura mais organizada, código mais limpo e fácil manutenção. Para refletir a técnica adotada, o repositório foi estruturado em módulos, conforme a Figura 3.

Ademais, a biblioteca de estilização PrimeNG[15] foi utilizada por ser bastante conhecida, possuir diversos componentes de código aberto e pela afinidade do autor.

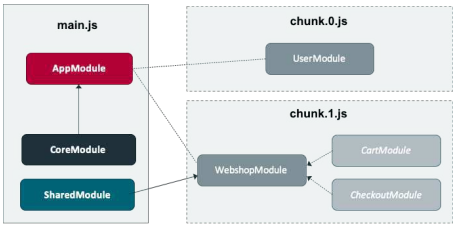

Figura 2: Exemplo arquitetural do Lazy loading no Angular.

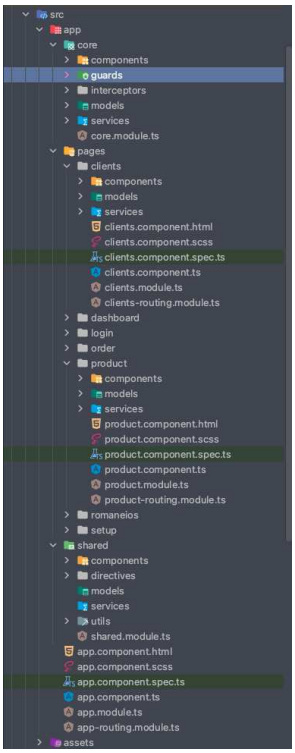

Figura 3: Estrutura do projeto Angular.

3.2.2 *Backend.*

O servidor foi desenvolvido utilizando o paradigma de orientação a objetos na linguagem Java 11, fazendo uso do framework Spring Boot[7]. Assim, o backend foi utilizado no sistema como o intermediário entre as informações que devem ser armazenadas no banco de dados e aquelas que são repassadas para a interface com o usuário, sendo o desenvolvimento possível graças à simplicidade de implementação de um serviço web utilizando o framework trazendo agilidade para o processo de desenvolvimento.

O framework Spring Security[6] foi utilizado como sistema de autenticação e autorização de alto nível. E também o JWT[17] (JSON Web Token) que transmitiu e autenticou informações com segurança.

Para a comunicação com o frontend foram criadas funções de comutação de informações que seguiram o padrão REST (Representational State Transfer) API(Aplication Interface). Também foi utilizado o padrão de projeto DTO para transferência de dados, desta forma permitindo a otimização da comunicação entre o cliente e o servidor. Portanto, o desenvolvimento que seguiu o padrão MVC, teve sua estrutura de diretórios dispostos no mesmo padrão aplicado ao frontend conforme ilustrado na Figura 4.

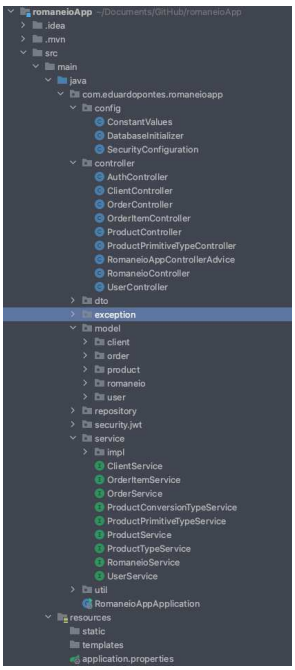

Figura 4: Estrutura do projeto Spring Boot.

3.2.3 *Banco de Dados.*

As informações do sistema que devem ser persistidas na base dados podem ser projetadas para se adequar ao modelo objeto-relacional de banco de dados. Portanto, optou-se por utilizar o PostgreSql[16] como sistema gerenciador por ser de código aberto e ter uma integração com o SpringBoot através do JDBC[13] (Java

Database Connectivity). Assim, com base nos pressupostos levantados foi definido o modelo de dados a ser implementado utilizando as tecnologias mencionadas anteriormente.

3.3 Sistema

Para atender os requisitos de usabilidade e simplicidade, o sistema foi dividido em 5 Módulos principais: Controle de Acesso, Configurações, Produtos, Clientes, Romaneios e Pedidos.

3.3.1 Controle de Acesso.

O módulo de Controle de Acesso tem por objetivo permitir a entrada de um usuário e realizar o controle de acesso baseado no perfil de usuário cadastrado previamente. Para fins de controle são considerados 5 perfis: Master, Administrador, Gerente, Conferente e Caixa. Cada perfil limita o acesso a determinadas funções, com o objetivo de manter o controle e integridade do sistema.

Por fim, além do controle de acesso, há também o fluxo de autorização que é realizado por meio da tela de login, no momento em que o usuário realiza o acesso ao sistema usando um nome de usuário e senha cadastrados anteriormente para uso das funcionalidades do sistema, conforme ilustrado nas Figuras 5 e 6.

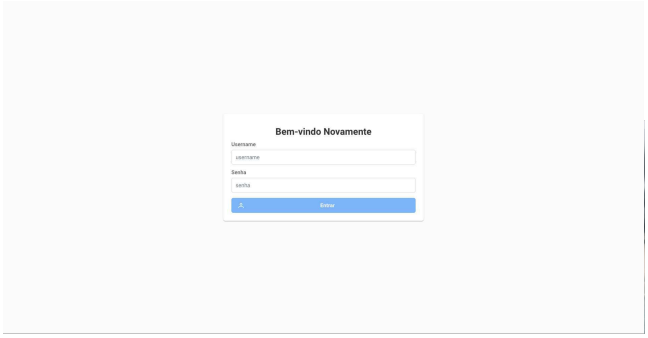

Figura 5: Tela de login.

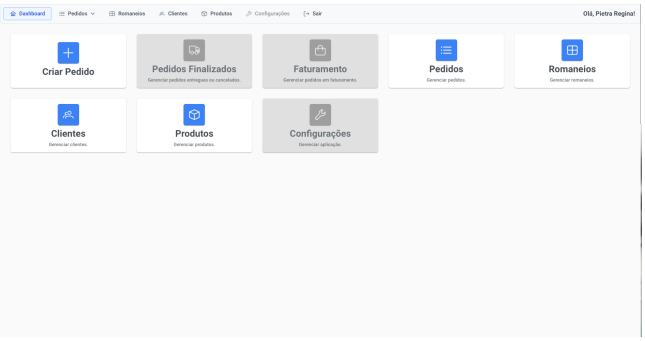

Figura 6: Tela de Dashboard em um usuário com restrição de acesso.

3.3.2 Configurações.

Este módulo é a parte do sistema responsável pelo gerenciamento de usuários e tipos primitivos, pois são funcionalidades que afetam todo o sistema. O acesso a este módulo é restrito a usuários do tipo Master, Administrador e Gerente.

O gerenciamento de usuário consiste na listagem, cadastro e edição dele. O cadastro é o momento em que são registrados na base de dados informações básicas e o tipo de perfil dele, com o objetivo de gerar um acesso deste usuário ao sistema (Figura 7). A edição é um processo semelhante ao cadastro, onde é possível gerir determinadas informações de um usuário bem como desabilitá-lo temporariamente ou excluí-lo definitivamente (Figura 8).

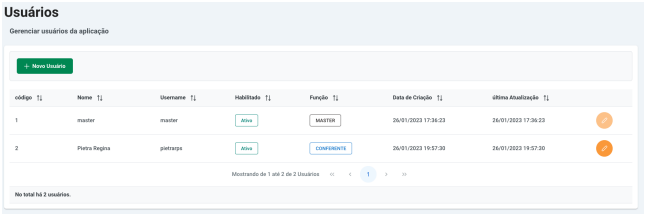

Figura 7: Listagem de usuários.

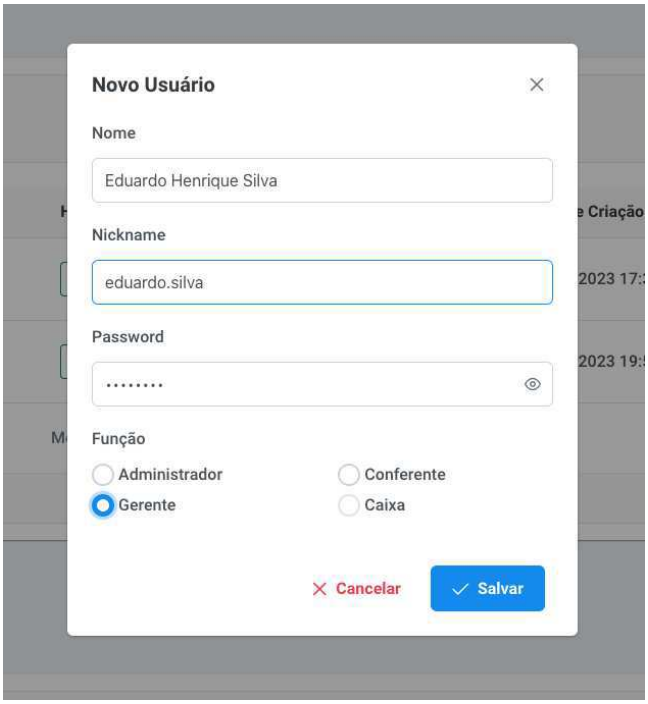

Figura 8: Cadastro e edição de usuários.

O gerenciamento de tipos primitivos é uma funcionalidade de manutenção de informações simples, sendo possível listar, criar, editar e excluir tipos primitivos (Figuras 9 e 10). A exclusão depende de que ele não esteja vinculado a nenhum produto.

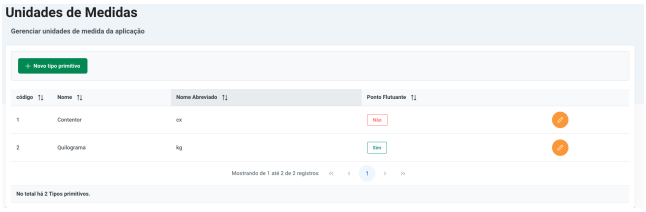

Figura 9: Listagem de tipos primitivos.

Figura 10: Cadastro e edição de tipos primitivos.

3.3.3 *Produtos.*

O módulo de produtos permite listar e gerenciar os produtos do sistema através de uma tela de listagem (Figura 11) O acesso a esta tela é permitido a todos os usuários, mas apenas os usuários master, administradores e gerentes podem gerenciar os produtos. O cadastro de produtos tem como pré-requisito a existência de pelo menos um produto primitivo cadastrado. Para a realização do gerenciamento,

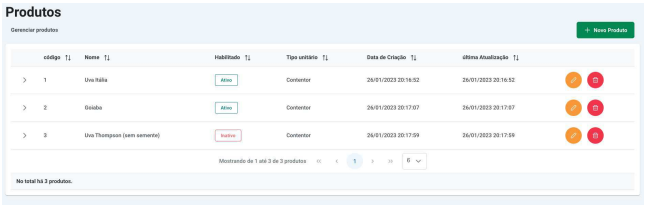

Figura 11: Listagem de produtos.

na tela de produtos, existem três botões dedicados a criação, edição e exclusão (Figura 12). Ao clicar no botão "Novo Produto" é possível criar um produto, onde dentre outras informações é necessário selecionar um tipo primitivo principal e adicionar outros tipos primitivos secundários, bem como a razão de conversão entre eles. A edição de um produto é um processo equivalente ao de criação.

3.3.4 *Clientes.*

Já o módulo de clientes permite exibir e gerenciar as informações relativas aos clientes. O acesso a esta tela também é permitido a todos os usuários, mas apenas os usuários master, administradores

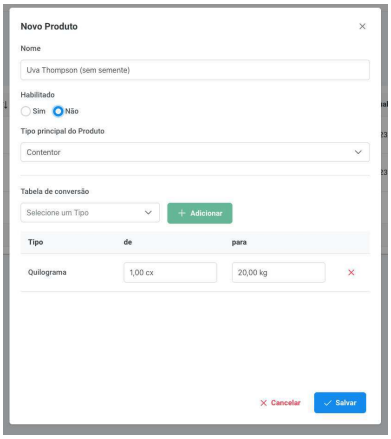

Figura 12: Cadastro e edição de produtos.

e gerentes podem gerenciar (Figura 13). Um cliente pode ser classificado em três tipos: atacado, varejo e outros. A tela de clientes

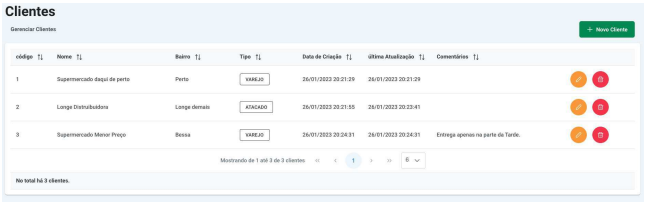

Figura 13: Listagem de clientes.

possui botões dedicados ao gerenciamento, tais como criar um cliente, editar e excluir, conforme ilustrado na Figura 14. Na tela de criação e edição é possível enviar a base dados informações sobre os clientes que o sistema deve persistir.

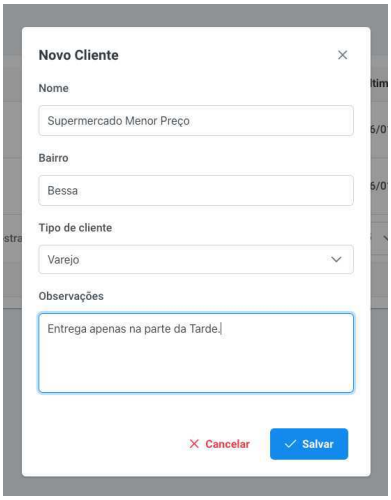

Figura 14: Cadastro e edição de produtos.

3.3.5 *Romaneios.*

O módulo de romaneio é responsável pela listagem de romaneios, gerenciamento e relatórios, conforme ilustrado na Figura 15. O acesso a esta tela também é permitido a todos os usuários, mas apenas os usuários master, administradores e gerentes podem gerenciar. Uma vez que um romaneio é criado é possível colocá-lo como ativo (Figura 16). Uma vez que existe um romaneio ativo é

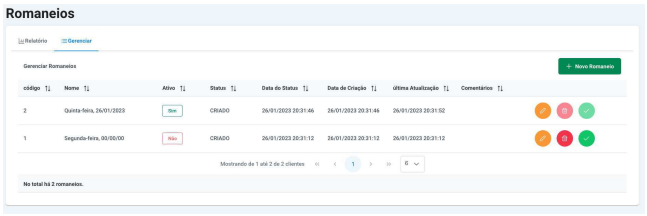

Figura 15: Listagem de Romaneios.

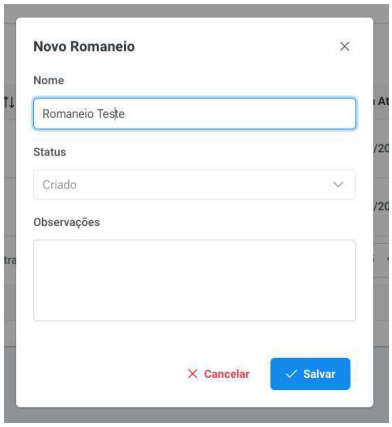

Figura 16: Cadastro e edição de Romaneios.

possível visualizar o relatório dele. O relatório é o agrupamento de todos os pedidos pelos produtos escolhidos, mostrando o quantitativo total de cada produto, bem como os clientes que pediram. Desta forma é possível facilitar as tomadas de decisões logísticas e administrativas, conforme Figura 17.

3.3.6 *Pedidos.*

Por fim, o módulo de pedidos é dividido em duas telas. A primeira tela é dedicada a criação de pedidos e permite relacionar um romaneio, cliente e uma lista de produtos (Figura 18). Um usuário também deve ser vinculado ao pedido, permitindo a delegação dos pedidos. A segunda tela é destinada a exibição e gerenciamento dos pedidos (Figura 19). Esta tela é personalizada de acordo com o tipo de usuário que a acessa. Os usuários do tipo master, administrador e gerente podem visualizar todos os pedidos de um romaneio. Já os demais usuários podem ver apenas os pedidos destinados a ele.

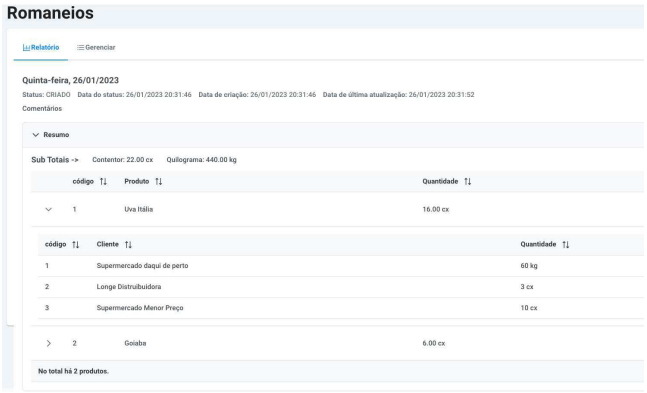

Figura 17: Relatório de um romaneio ativo.

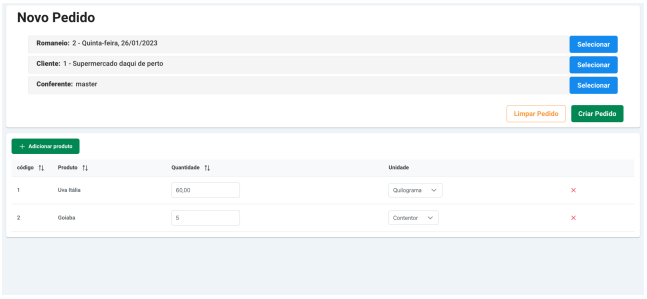

Figura 18: Tela de criação de pedido.

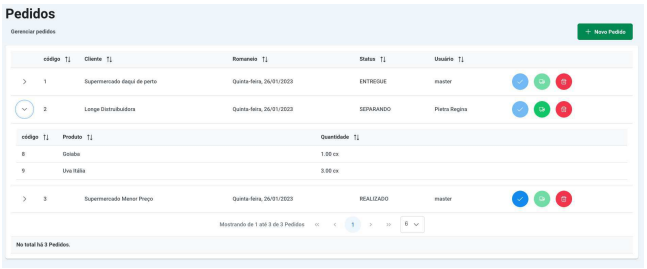

Figura 19: Listagem de pedidos.

3.4 Ambiente de Implantação

Para os propósitos deste trabalho, a aplicação foi disponibilizada para testes usando um servidor próprio do usuário em uma infraestrutura On-Site. Numa infraestrutura On-Site ou On-premise[3] todos os componentes de hardware e softwares são gerenciados pelo o usuário. Tal escolha se deu devido as preocupações do usuário quanto a segurança de acesso durante o desenvolvimento.

Além disso, foi utilizado o Docker[4] para automatizar, modularizar e isolar as instâncias da aplicação, sendo 3 instâncias configuradas:

- Uma para o banco de dados PostgreSQL;
- Uma para o Java e Spring boot;
- Uma para a aplicação Angular sendo provida pelo NGINX[5].

4 EXPERIÊNCIA

Esta seção irá detalhar o processo de desenvolvimentos, os desafios envolvidos e a avaliação da aplicação desenvolvida.

4.1 Processo de desenvolvimento

O processo de desenvolvimento foi dividido em etapas, sendo a primeira delas o levantamento de requisitos. O levantamento de requisitos partiu da experiência prévia do primeiro autor e de conversas informais com os potenciais usuários, levantando gargalos encontrados no dia a dia que podem ser sanados com funcionalidades da aplicação. A última fase desta etapa consistiu em transformar estes requisitos em funcionalidades em potencial bem como levantar quais delas são consideradas essenciais para validação da aplicação. Com isso, o processo de desenvolvimento e entregas adotado foi uma adaptação da metodologia *Scrum* que partiu da divisão da aplicação em dois grandes entregáveis (*milestones*) e *Sprints* que duraram cerca de uma semana com entregas que gerassem valor, partindo do backend e evoluindo até o desenvolvimento das telas.

Após esta etapa, foi realizado um levantamento das tecnologias mais adequadas para o desenvolvimento e foi considerado como requisito a afinidade com ela, a facilidade de manutenção e o custo operacional de uma infraestrutura dedicada. Com isso, para o frontend foi analisado o que é mais comum no mercado e levantou-se que basicamente duas tecnologias são usadas em larga escala, são elas Angular e ReactJS[11]. Angular foi a tecnologia escolhida devido ao seu amplo uso no meio corporativo e por ser um framework de fato e já possuir integrado todas as bibliotecas necessárias para o desenvolvimento. A tecnologia para backend escolhida foi o Spring Boot com JAVA devido a larga experiência do autor com uso desta tecnologia. Por fim, o banco de dados escolhido teve como base a facilidade de implantação e manutenção tendo em vista que tanto o backend, como o banco de dados podem ser providos por serviços PaaS (Plataforma as a Service) tais como o Heroku[8].

A terceira etapa, consistiu na escolha dos padrões de projetos bem como a estruturação do mesmo e a preparação do ambiente de desenvolvimento. No backend, a padronização do código foi mantida pela IDE da JetBrains chamada Intellij[9] que faz um excelente trabalho nesse sentido e o padrão adotado para a estrutura do código foi a que melhor sintetiza o padrão de projeto MVC adotado. No Frontend, a padronização do código foi mantida pelas bibliotecas ESlint[19] e Prettier[14] que são referência na comunidade para esse objetivo, além de ser recomendada pelos frameworks. Por fim, a estrutura de código do frontend reflete as melhores práticas observadas na comunidade que preza por uma estrutura de código escalável e de fácil manutenção.

A implementação partiu do backend devido a complexidade do desenho do modelo de banco de dados que possuem um relacionamento complexo. O desenvolvimento do frontend foi iniciado após a conclusão dos principais módulos da aplicação. Como a implantação do sistema dependia de uma ação manual, numa estrutura Onpremisse, o código foi mantido na branch principal do GitHub[12] e implantado sempre que uma funcionalidade estava completa e funcional. E com o objetivo de simular o ambiente produtivo, uma infraestrutura similar foi era utilizada através do Docker para manter o ambiente de desenvolvimento o mais próximo possível do ambiente produtivo.

4.2 Desafios

O primeiro grande desafio no desenvolvimento do sistema foi o tempo de implementação. Devido ao período final de pandemia e a problemas familiares do primeiro autor, se fez necessário mudar a linha de pesquisa e para atender as restrições de tempo houve a necessidade de reduzir o desenvolvimento do sistema a um mínimo produto viável (MVP) que pudesse validar a ideia central do sistema e permitir a evolução dele.

Outro desafio foi conciliar o desenvolvimento apertado e o tempo do usuário potencial com as apresentações e consultorias necessárias com o usuário para o alinhamento de requisitos. Mas, no final todas as funcionalidades que entraram no escopo foram analisadas e validadas a tempo da entrega deste trabalho.

O desenvolvimento deste projeto exercitou todas as habilidades e conhecimentos adquiridos durante a formação. Partindo do backend e banco de dados, pode-se exercitar a programação orientada a objetos e o desenho de bancos de dados relacionais. Além disso, o backend trouxe desafios que permitiram exercitar bastante os conhecimentos adquiridos em Estrutura de dados e Algoritmos, desde a escolha das estruturas de dados para armazenamentos de objetos até a análise da complexidade dos algoritmos para não degradar a performance do sistema. Um fato interessante é que as versões recentes de Java implementam interfaces funcionais, originadas do paradigma funcional que também é abordado na graduação e foi utilizado neste projeto.

As disciplinas de Programação II, Projeto de Software, Banco de Dados, Estrutura de Dados e Algoritmos, Engenharia de Software, Paradigmas de Linguagens de Programação, Projeto em Computação e as optativas específicas Princípios de Desenvolvimento Web e Gestão de Projetos foram essenciais para comportar o conhecimento e formação necessária para atingir o objetivo.

4.3 Avaliação

A avaliação do sistema deu-se através de entrevista com dois potenciais consumidores que atuam em subsegmentos distintos. O primeiro atua com vendas de produtos em atacado sendo seus itens comercializados principalmente em contentores (caixas), sacos ou unidades. O segundo usuário trabalha com fornecimento para supermercados, logo seus pedidos têm como unidade principal o quilograma, sendo uma porção maior do que a do primeiro usuário. Devido a necessidade em reduzir o desenvolvimento, ambos chegaram à conclusão de que suas maiores dificuldades são na sintetização dos pedidos de seus clientes para formular um pedido junto a seus fornecedores, portanto este MVP visa atender esta funcionalidade.

Como uma das preocupações deste projeto é a simplicidade e facilidade de uso por usuários leigos, o MVP proposto se atentou em utilizar nomenclatura e processos já conhecidos pelos usuários. Logo, com um tutorial básico, ao longo da avaliação os mesmos já conseguiam explorar o potencial máximo da aplicação.

O processo de avaliação foi realizado presencialmente em uma entrevista informal na qual os usuários relataram suas considerações à medida que progrediam no uso da aplicação. Os usufrutuários testaram o fluxo, desde a criação de usuários até a de pedidos.

Por causa da complexidade das funcionalidades, a avaliação foi dividida em duas etapas. A primeira etapa focou na avaliação do gerenciamento de usuários, tipos primitivos e produtos. Já a segunda, consistiu na validação do restante da aplicação, principalmente a criação de pedidos e exibição do relatório de um romaneio. Ao todo as duas etapas tiveram uma duração de mais ou menos uma hora de uso.

Ao navegar pelo sistema ambos relataram terem tido uma boa experiência de uso e elogiaram a simplicidade do sistema na maioria dos pontos. Um dos usuários solicitou uma customização em sua versão final do sistema que restringe o acesso a funcionalidade de exibição de relatório de romaneios a gerentes, administradores e o master.

Ademais, ambos elogiaram, agradeceram e disseram estar ansiosos pela evolução do sistema. Apesar de afirmarem que o sistema em sua condição atual supriria suas demandas, os usuários relataram algumas possíveis melhorias para maximizar seus resultados, dentre elas:

- Melhorar a visualização e experiência de usuário ao visualizar as tabelas existentes no sistema em dispositivos mobile;
- Uma página ou função que permita ver o histórico de pedidos anteriores de um cliente;
- Adicionar na barra de menu uma visualização de qual romaneio está ativo e um botão para alterá-lo.

Por fim, foi encontrado apenas um defeito, onde no cadastro independente da escolha do tipo de cliente, apenas o que já estava pré-selecionado era cadastrado. Constatou-se rapidamente o problema que também foi facilmente resolvido. Todos os requisitos essenciais e a maioria dos requisitos não essenciais foram atendidos, faltando apenas a implementação da tela de alteração de senha e o aprimoramento do fluxo de status de pedidos pela adição dos status "Em Faturamento"e "Faturado".

5 TRABALHOS FUTUROS

Este trabalho teve por objetivo desenvolver uma aplicação que simplifique e auxilie comerciantes do ramo hortifrutigranjeiro a gerenciar seus pedidos. Ademais, mesmo com os desafios encontrados e complexidade da construção da solução foi possível implementar um MVP que atendesse uma parte significativa do dia a dia desses profissionais.

Outrossim, por se tratar de uma aplicação focada no comércio, mesmo que a aplicação esteja completa é sempre possível evoluí-la, seja construindo novas funcionalidades, seja melhorando e refinando as já existentes. Além das melhorias ponderadas em seções anteriores, outras funcionalidades podem ser adicionadas, como uma ferramenta de controle de estoque, o uso de ciência de dados para levantar gráficos e relatórios relevantes tal como a relação entre os produtos mais vendidos e a região em que os pedidos são feitos; média móvel dos produtos mais pedidos, dentre outros.

Por fim, considerando as limitações de tempo, o principal desafio no desenvolvimento desta aplicação foi não deteriorar a performance ao longo do desenvolvimento, uma vez que as entidades nelas envolvidas possuem uma grande complexidade. Em trabalhos futuros será possível implementar as melhorias sugeridas, bem como realizar a divisão das telas de pedidos para tornar os tipos de usuários ainda mais funcionais. Além disso, é possível levantar um estudo de caso quanto aos benefícios do uso de microsserviços para melhor o desempenho da aplicação como o todo, bem como a relação custo-benefício.

REFERÊNCIAS

[1] JustNew Sistemas Cloud. 2023. HORTIGESTÃO. Retrieved 2023-01-31 from https://hortigestao.com.br/
[2] Google. 2016. Angular 2+. Retrieved 2023-01-26 from https://angular.io/
[3] Red Hat. 2022. IaaS x PaaS x SaaS. Retrieved 2023-01-25 from https://www.redhat.com/pt-br/topics/cloud-computing/iaas-vs-paas-vs-saas
[4] Docker Inc. 2013. Docker. Retrieved 2023-01-26 from https://www.docker.com/
[5] F5 Inc. and Igor Sysoev. 2004. NGINX. Retrieved 2023-01-26 from https://www.nginx.com
[6] VMware Inc. 2004. Spring Security. Retrieved 2023-01-26 from https://spring.io/projects/spring-security
[7] VMware Inc. 2014. Spring Boot. Retrieved 2023-01-26 from https://spring.io/projects/spring-boot
[8] Adam Wiggins James Lindenbaum and Orion Henry. 2007. Heroku. Retrieved 2023-01-26 from https://www.heroku.com/
[9] JetBrains. 2007. IntelliJ IDEA. Retrieved 2023-01-26 from https://www.jetbrains.com/pt-br/idea/
[10] Marcio. 2011. Padrão MVC - Java Magazine. Retrieved 2023-01-26 from https://www.devmedia.com.br/padrao-mvc-java-magazine/21995
[11] Inc Meta Platforms. 2013. React: A JavaScript library for building user interfaces. Retrieved 2023-01-26 from https://reactjs.org/
[12] Microsoft. 2008. Github. Retrieved 2023-01-26 from https://github.com/
[13] Oracle. 1997. Java JDBC API. Retrieved 2022-10-28 from https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/
[14] Prettier. [n. d.]. Prettier. Retrieved 2023-01-26 from https://prettier.io/
[15] Primetek. [n. d.]. PrimeNG. Retrieved 2023-01-26 from https://www.primefaces.org/primeng/
[16] Open Source. 1986. PostgreSQL: The World's Most Advanced Open Source Relational Database. Retrieved 2022-10-28 from https://www.postgresql.org/
[17] Open Source. 2015. JSON Web Token (JWT). Retrieved 2023-01-26 from https://www.rfc-editor.org/rfc/rfc7519
[18] Patrus Transportes. 2017. Entenda já o que é romaneio de carga! Retrieved 2023-02-15 from https://patrus.com.br/blog/entenda-ja-o-que-e-romaneio-de-carga/#:~:text=O%20romaneio%20de%20carga%20%C3%A9,os%20dados%20referentes%20%C3%A0%20distribui%C3%A7%C3%A3o.
[19] Nicholas C. Zakas. 2013. ESLint: Find and fix problems in your JavaScript code. Retrieved 2023-01-26 from https://eslint.org/