NATHAN ARAÚJO SILVA

DESENVOLVIMENTO DE UM SISTEMA DE ATENDIMENTO PARA UMA REDE DE LABORATÓRIOS

LAVRAS - MG

2023

NATHAN ARAÚJO SILVA

DESENVOLVIMENTO DE UM SISTEMA DE ATENDIMENTO PARA UMA REDE DE LABORATÓRIOS

Trabalho de conclusão de curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. Dr. Antonio Maria Pereira De Resende
Orientador

LAVRAS - MG

2023

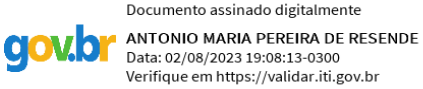
NATHAN ARAÚJO SILVA

DESENVOLVIMENTO DE UM SISTEMA DE ATENDIMENTO PARA UMA REDE DE LABORATÓRIOS

Trabalho de conclusão de curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação, para a obtenção do título de Bacharel.

APROVADA em 21/07/2023.

Prof. Dr. Antonio Maria Pereira De Resende	UFLA
Prof. Janderson Rodrigo de Oliveira	UFLA
Rafael Ângelo Martins de Oliveira	DTI



Prof. Dr. Antonio Maria Pereira De Resende
Orientador

LAVRAS - MG
2023

Dedico principalmente a Deus, à minha família, aos meus pais, com ênfase em minha mãe Magda, minha vó Hidê, e minha madrinha Mara, sem vocês nada disso seria possível, obrigado pela paciência e apoio nesses anos ausentes.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, que vem me abençoando desde o momento em que consegui a aprovação na universidade, me guiando para escolhas certas e me protegendo!
Também agradeço a toda a minha família, em especial, minha mãe Magda, minha vó Hidê e minha tia Mara, que sempre se fizeram presente durante toda minha graduação, tendo paciência devido à ausência neste tempo e sempre orando por mim. Menção especial para minha madrinha Mara, que foi quem me deu o meu primeiro computador, e desde então, ao primeiro contato com a tecnologia, não consegui pensar em outra área para seguir como profissional em minha vida.
Agradecimento especial aos amigos que fiz durante a graduação, em especial, Igor e Breno, efetuamos uma parceria que se estendeu do início ao final do curso, sempre fazendo grupos quando possível e estudando juntos!

“Não espere pela oportunidade. Crie-a”
(Sêneca)

RESUMO

O presente documento tem o propósito de relatar as atividades desenvolvidas durante o estágio do autor na empresa Dti Digital, iniciado em 04/01/2022 e ainda em andamento. Durante esse período, o foco foi no desenvolvimento de um sistema de atendimento para uma rede de laboratórios, utilizando tecnologias como React.js e suas bibliotecas internas, bem como Typescript. Além disso, foi realizado o desenvolvimento de um BFF (*Back-end for Front-end*) em CSharp, utilizando o *framework* .NET. As atividades abrangeram o desenvolvimento em ambos os lados, *front-end* e *back-end*, visando a criação de uma solução completa e funcional para o sistema de atendimento da rede de laboratórios.

O objetivo do desenvolvimento do sistema foi aprimorar o atendimento prestado aos usuários, buscando melhorar sua segurança por meio da implementação de diversas funcionalidades que reduzissem as falhas operacionais. Além disso, o sistema foi projetado para ser ágil e dinâmico em seus fluxos, proporcionando uma experiência de uso rápida e eficiente. A interface do sistema foi desenvolvida de forma a ser simples e intuitiva, permitindo que qualquer pessoa possa utilizá-lo com facilidade.

Durante o estágio, foi evidente o significativo crescimento do estagiário em termos de conhecimentos na área de desenvolvimento de *software*. Além disso, suas habilidades de negociação e comunicação também foram aprimoradas. A experiência proporcionou ao estagiário um ambiente propício para expandir seus conhecimentos e aplicar suas habilidades, resultando em um desenvolvimento profissional substancial.

Palavras-chave: Desenvolvimento de software. Dti Digital. BFF. Reactjs. Typescript. CSharp. .NET. Cache.

ABSTRACT

This document aims to report the activities carried out during the author's internship at Dti Digital, which started on January 4, 2022 and is still ongoing. During this period, the focus was on developing a customer service system for a network of laboratories, using technologies such as React.js and its internal libraries, as well as TypeScript. Additionally, a Back-end for Front-end (BFF) was developed in CSharp using the .NET framework. The activities encompassed both front-end and back-end development, aiming to create a comprehensive and functional solution for the laboratory network's customer service system.

The goal of the system development was to enhance the quality of service provided to users by improving security through the implementation of various features to reduce operational failures. Additionally, the system was designed to be agile and dynamic in its flows, providing a fast and efficient user experience. The system's interface was developed to be simple and intuitive, enabling easy usage for all users.

Throughout the internship, there was a significant growth in the intern's knowledge in software development. Furthermore, their negotiation and communication skills were also improved. The experience provided an environment conducive to expanding knowledge and applying skills, resulting in substantial professional development.

Keywords: Software Development. Dti Digital. BFF. Reactjs. Typescript. CSharp. .NET. Cache.

LISTA DE FIGURAS

Figura 2.1 – Dti Flow 17
Figura 3.1 – Fluxo do Scrum 20
Figura 3.2 – Exemplo do Board Azure 21
Figura 3.3 – Arquitetura Projeto 22
Figura 3.4 – Comparação Java e CSharp 24
Figura 4.1 – Exemplo teste unitário 27
Figura 4.2 – Exemplo de variável de ambiente “GENERATE SOUCE MAP" 27
Figura 4.3 – Exemplo de tag CSP 28
Figura 4.4 – Exemplo de estória antes do refinamento 31
Figura 4.5 – Exemplo de estória após o refinamento 32
Figura 4.6 – Tela de Conclusão 36
Figura 4.7 – Tela de Dados do pagador 37
Figura 4.8 – Modal de seleção do pagador 38
Figura 4.9 – Modal de seleção do pagador após filtro dos dados 39
Figura 4.10 – Formulário de Dados do pagador com os dados preenchidos após seleção do pagador 40
Figura 4.11 – Exemplo de Uso *useState* 41
Figura 4.12 – Exemplo de uso RHF 42
Figura 4.13 – Exemplo de uso Yup 43

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Empresa 10
1.2 Objetivos 11
1.3 Estrutura do restante deste documento 11
2 Descrição do local de trabalho 12
2.1 Organização Descentralizada da empresa 12
2.2 Organização Centralizada da empresa 12
2.2.1 Keepers 13
2.2.2 Marketing 13
2.2.3 Chapter 13
2.2.4 Pessoas 13
2.2.5 Round 13
2.2.6 Ignitions 14
2.3 Operation Canvas 14
2.4 Pilares de maestria 14
2.5 Guildas: 15
2.6 Funções comuns na organização 15
2.7 Dti Flow 17
3 Metodologias e Ferramentas Técnicas 19
3.1 Scrum 19
3.2 Azure Devops 20
3.3 Integração com outras aplicações 21
3.4 React 22
3.5 TypeScript 22
3.6 Arquitetura Back-End 23
3.7 .NET 23
3.8 Postman 24
3.9 Dbeaver 25
3.10 REST 25
4 Atividades Desenvolvidas 26
4.1 Ritos da empresa 26

4.1.1 Check de Engenharia 26
4.1.2 Check de Segurança 27
4.1.3 Check de Produto e Design 28
4.1.4 Check de Operações 29
4.2 Refinamento Técnico 30
4.3 Desenvolvimento de interfaces (Front-End) 33
4.3.1 Possibilitando o pagamento para Pessoa Jurídica (PJ) 33
4.3.2 Utilização de biblioteca para validar formulário 40
4.4 Familiarização com o padrão arquitetural REST 43
4.5 Considerações Finais 43
5 CONCLUSÃO 45
REFERÊNCIAS 47

1 INTRODUÇÃO

Este documento descreve o trabalho de conclusão de curso do autor, sendo realizado na modalidade de Estágio Supervisionado. Este capítulo traz uma introdução sucinta sobre a empresa onde o estágio foi realizado, a Dti Digital, suas principais características, sua estrutura, seu principal ramo de negócios da empresa, e as funções desempenhadas pelo autor durante o período de estágio.

1.1 Empresa

O estágio em questão foi realizado de Janeiro de 2022 e continua até o atual momento em que este trabalho está sendo escrito, com contrato até o mês de Janeiro de 2024. O estágio foi realizado na Dti Digital[1], uma empresa pertencente ao grupo *WPP Group*, uma empresa ágil, pela utilização da metodologia ágil Scrum em seus processos, facilitando o trabalho realizado no dia-a-dia. Além disso, outro fato sobre a empresa, é que ela possui um sistema de organização interno descentralizado.

Atualmente, a Dti presta serviços para diversos clientes que atuam em redes de laboratórios, aplicativos de comida, e-commerces, entre outros. Hoje a empresa possui em torno de 1000 funcionários, divididos em setores diferentes como Tecnologia, Financeiro, Recursos Humanos e *Marketing*. A matriz da empresa localiza-se em Belo Horizonte, onde ocorrem algumas reuniões presenciais ao longo do ano. A empresa, apesar de sua sede ampla, aceita e incentiva o trabalho *home-office*.

Durante o período de estágio, o estagiário pode acompanhar de perto o desenvolvimento de um sistema de atendimento para uma rede de laboratórios. O mesmo pode participar dos ritos do *squad*, realizar manutenção no código, e desenvolver novas funcionalidades.

Um sistema de atendimento é, no contexto do projeto em que o estagiário atuou, uma plataforma de cadastro de informações referentes a pedidos, na área de saúde. Alguns fluxos presentes neste sistema são: cadastro de clientes, preenchimento de convênio/plano de saúde do cliente, cadastro do exame a ser realizado pelo cliente, pagamento dos exames realizados, entre outros.

[1] <https://www.dtidigital.com.br/>

A intenção do estágio é capacitar gradativamente o estagiário como um desenvolvedor hábil e para a realização de projetos importantes na área escolhida por ele. Tal experiência é adquirida gradativamente, de acordo com cada tarefa designada ao estagiário.

1.2 Objetivos

Dentre os objetivos do estágio, estava a introdução do estagiário no mercado de trabalho, bem como seu crescimento como desenvolvedor, especialmente em termos de lógica de programação e resolução de problemas, além da melhoria de habilidades consideradas *soft skills*, como comunicação, negociação e liderança. Com esses objetivos em mente, o estagiário desempenhou várias atividades e assumiu diferentes papéis ao longo do período de estágio. Isso incluiu a resolução de *bugs* relatados pelos usuários durante a fase de homologação do sistema, na criação de rotas, utilizando os dados provenientes do *back-end* e manipulando-os no *front-end*, além de refinar tecnicamente algumas tarefas antes de serem incluídas em uma *sprint*, bem como o desenvolvimento de testes unitários.

1.3 Estrutura do restante deste documento

O restante trabalho está organizado da seguinte forma: (ii) o Capítulo 2 apresenta a organização interna da empresa, (iii) no Capítulo 3 serão descritas as tecnologias utilizadas pelo estagiário, e (iv) no Capítulo 4 são apresentadas as atividades desenvolvidas durante o período de estágio, e por fim (V) no Capítulo 5, serão apresentadas as considerações finais deste trabalho.

2 DESCRIÇÃO DO LOCAL DE TRABALHO

Neste capítulo serão apresentados alguns detalhes sobre a empresa, como uma especificação de sua divisão interna, que mescla duas formas: centralizada e descentralizada, além de descrever algumas funções comuns na organização.

2.1 Organização Descentralizada da empresa

Visando aumentar a agilidade e a autonomia das equipes, a Dti adota uma estrutura combinada de setores centralizados e equipes distribuídas. A distribuição ocorre entre *enterprizes*, alianças, tribos e *squads*, cada um com um nível de especificidade e foco maior que o anterior. Abaixo, cada papel subestrutura pertencente a estrutura principal descentralizada, será descrito abaixo, para um melhor entendimento.

Squad: O *Squad* como a unidade de trabalho mais específica, composta por uma equipe multidisciplinar que efetivamente desenvolve e lida diretamente com projetos e clientes. Um *Squad* possui cerca de 6 a 7 integrantes.

Tribo: Nas Tribos são onde são alocados os *squads*. Elas funcionam como suborganizações, com autonomia para gerenciar as equipes — incluindo contratações, orçamentos e projetos abrangentes. Uma Tribo possui 20 a 40 integrantes.

Aliança: As Alianças seriam as formas de organização mais abrangentes, responsáveis por gerenciar várias tribos de uma perspectiva ampla. Tal conjunto de tribos é criado muito por interesse no contexto em que as mesmas estão inseridas, tornando-se comum uma aliança com grande parte de suas tribos/ *squads* pertencentes a um mesmo cliente. Uma Aliança possui cerca de 100 pessoas.

Enterprize: O conceito de *enterprize* acaba se assemelhando um pouco em relação ao conceito de aliança, porém, o que as difere, seria o fato de uma Enterprize ser um conjunto de alianças e não de tribos. Uma *Enterprize* possui cerca de 300 pessoas.

2.2 Organização Centralizada da empresa

Como mencionado anteriormente, a Dti também possui setores centralizados, para servir como referência para as equipes. Esses setores não mantêm interação direta com as equipes, contudo, provêm suporte às operações que estas realizam. Eles pertencem a um *Núcleo habi-*

litador, como é chamado pela Dti. Abaixo, cada papel pertencente a esta estrutura é descrito, para um melhor entendimento.

2.2.1 Keepers

Conhecido como a tribo de suporte e infraestrutura, a *Keepers* se divide em dois *squads*: o *Facilities*, sendo responsável pelos escritórios da Dti, cuidando de logísticas de entregas, realizando compras e tendo pleno conhecimento acerca de todos os eventos ocorridos no âmbito físico da Dti; e o Infra, responsável por cuidar da infraestrutura de redes da Dti, como o ambiente virtual, computadores, licenças de *software* e segurança da rede.

2.2.2 Marketing

O Marketing possui uma estrutura centralizada, para posicionamento da Dti e geração de demanda e, habilitadora, para amparar a execução descentralizada. Sua missão é educar o mercado e fortalecer a autoridade da Dti, materializando a sua especialização, através do *true agile* — genuína adoção dos princípios ágeis, enfatizando valores, flexibilidade, comunicação efetiva e auto-organização da equipe.

2.2.3 Chapter

Seria uma estrutura local presente em cada aliança, cujo propósito seria ligado ao contexto que foram criados. Não tem o foco de resolver problemas genéricos, mas sim para resolver problemas do contexto em que estão.

2.2.4 Pessoas

A subestrutura de pessoas se considera um *Chapter*, cuja definição foi estabelecida logo acima. O *Chapter* de Pessoas é um coletivo horizontal que congrega líderes com competências e responsabilidades semelhantes no que tange à gestão de pessoas. Cada Enterprize possui pessoas referências nas competências de Recursos Humanos e Administração. Estas, junto as principais lideranças técnicas e de gestão, formam a estrutura do *Chapter* de Pessoas da Dti.

2.2.5 Round

Round seria, das subestruturas centralizadas, a mais inovadora. Seu propósito é ser uma plataforma digital de interação, de toda a empresa. Algumas funcionalidades presentes

nessa plataforma são: programa de doação de moedas fictícias entre os *crafters* que podem ser trocadas por prêmios; seção para agendamento de 1-1's — são reuniões individuais entre líder e membro da equipe para promover comunicação aberta, *feedback*, e desenvolvimento profissional; programa de mentoria, onde você pode selecionar um tema desejado e encontrar um mentor na própria Dti, para auxílio semanal na busca de conhecimento, etc.

2.2.6 Ignitions

Ignitions são palestras introdutórias, com a intenção de apresentar a cultura da Dti para novos integrantes, mostrar como os processos são organizados e principalmente, apresentá- los aos valores da empresa e forma de atuar.

2.3 Operation Canvas

O *Operation* Canvas é um documento cujo objetivo principal consiste em ser um acordo informal da equipe em relação aos seus processos e práticas. Nele são descritas e detalhadas algumas particularidades da maneira como a equipe opera, incluindo alguns dos rituais realizados, os *status* das estórias, os horários dos ritos, as maneiras de comunicação utilizadas pela equipe, entre outros. Regularmente, é ideal que esse documento seja atualizado na presença de todos os membros da equipe. Além de garantir a concordância do time sobre a forma com a qual todos se comprometem a trabalhar, esse documento se torna uma ferramenta muito útil, principalmente para os novos integrantes da equipe, dado que o *Operation* Canvas proporciona acesso simplificado e prático aos processos peculiares presentes em cada equipe, sua utilização facilita a integração e adaptação diante das mudanças de contexto.

2.4 Pilares de maestria

Os pilares de maestria representam as disciplinas técnicas fundamentais para a entrega de valor direta para o cliente, através da criação de ativos digitais, sendo a base do modelo de negócio da Dti.

Engenharia: O Pilar da Engenharia na Dti é o responsável por garantir a qualidade da entrega, alcançada através da formação contínua dos *crafters* e da execução da metodologia materializada no Dti Flow. Busca-se aprimorar sempre para escrever um código limpo e projetar as arquiteturas corretas para cada desafio.

Produto: O Pilar de Produto na Dti é o guardião da metodologia da Dti, com o propósito de assegurar que a criação dos ativos digitais gere efetivamente resultados para o negócio, é imprescindível direcionar esforços ao entendimento aprofundado dos problemas e necessidades dos clientes. A disciplina de produto é fundamental para orientar a construção do produto certo, segundo a Dti.

Design: O Pilar de Design na Dti é o guardião da metodologia para criar ativos digitais com foco nas pessoas. A metodologia vai desde a pesquisa para descoberta de oportunidades até a tradução dessas oportunidades em experiências fáceis de usar, acessíveis e apropriadas para cada contexto. O Pilar de *Design* é fundamental para a empresa conseguir gerar valor de verdade.

Operações: O Pilar de Operações na Dti é o guardião da metodologia e fluxo geral de trabalho, além de ser um pilar agregador dos outros três pilares técnicos: *Design*, Engenharia e Produto, instanciando seus métodos e ferramentas, na prática.

2.5 Guildas:

Uma Guilda é um grupo de pessoas de diferentes tribos e orientados por afinidade, a fim de compartilhar experiências, aprendizados e melhores praticas em tópicos de interesse em comum. É uma maneira pela qual a organização mantém a unidade, a qualidade do serviço e uma padronização mínima do processo.

2.6 Funções comuns na organização

As principais funções desempenhadas na Dti digital são *Product Owner*, *Product Designer*, Desenvolvedor, Desenvolvedor Líder, Arquiteto de *Software*, Scrum *Master*, *Tech Manager*. Na organização cada uma dessas funções desempenham um conjunto de responsabilidades e estão descritas com maior detalhe, permitindo o leitor maior compreensão da empresa.

Product Owner: É sua responsabilidade garantir a consistência conceitual das novas funcionalidades, correções de bugs ou melhorias, de modo a alinhá-las com a visão estabelecida para o produto ou projeto. Além disso, ele tem a responsabilidade de assegurar a qualidade final das entregas, sendo o único com autoridade para aprovar as estórias como concluídas. Embora a preocupação com o produto deva ser de toda a equipe, o *Product Owner* tem essa função como principal.

Product Designer: Em conjunto com o *Product Owner*, o *Product Designer* desempenha um papel fundamental para garantir a qualidade do produto. O trabalho desse profissional vai além da estética, envolvendo uma abordagem estratégica no desenvolvimento de soluções que visam resolver os problemas dos usuários. Dessa forma, a importância do *Product Designer* é que ele contribui para a experiência do usuário, agregando valor e qualidade ao produto.

Desenvolvedor: O desenvolvedor desempenha um papel importante na parte técnica do *software*, como acontece na maioria das empresas atuais, utilizando linguagens de programação e outras tecnologias disponíveis para implementar os projetos recebidos pela empresa. As divisões mais tradicionais de categorias entre os desenvolvedores, como júnior, pleno e sênior, não são adotadas como referência na Dti, porém, há formas de alavancagem e reconhecimento de habilidades e evolução técnica durante a trajetória do desenvolvedor na empresa, podendo crescer e receber pequenas promoções ao longo do tempo, como maiores salários ou novas responsabilidades.

Desenvolvedor Líder: É entendido que o Desenvolvedor Líder, tenha menos responsabilidades práticas de programação, a fim de dedicar seu tempo e esforços em apoiar os outros membros da equipe de desenvolvimento. Assim, há várias direções que um desenvolvedor líder pode seguir. Por exemplo, alguns possuem um conjunto maior de habilidades técnicas podendo ajudar em questões mais complexas de desenvolvimento, enquanto outros tendem a se concentrar mais nas habilidades de comunicação, e gestão, auxiliando na mitigação de empecilhos, principalmente, gerados entre a equipe e o cliente.

Arquiteto de Software: O profissional que assume essa função é responsável por criar soluções em um nível mais abstrato, baseando-se em padrões de projeto e boas práticas, além de considerar o contexto e as necessidades específicas de cada aplicação. Ele auxilia a equipe técnica na manutenção do código, visando manter o projeto na arquitetura proposta, e propõem soluções baseadas em boas práticas de programação, facilitando o trabalho da equipe de desenvolvimento, mediante a dificuldade imposta pela complexidade dos problemas que surgem durante o dia-a-dia de trabalho.

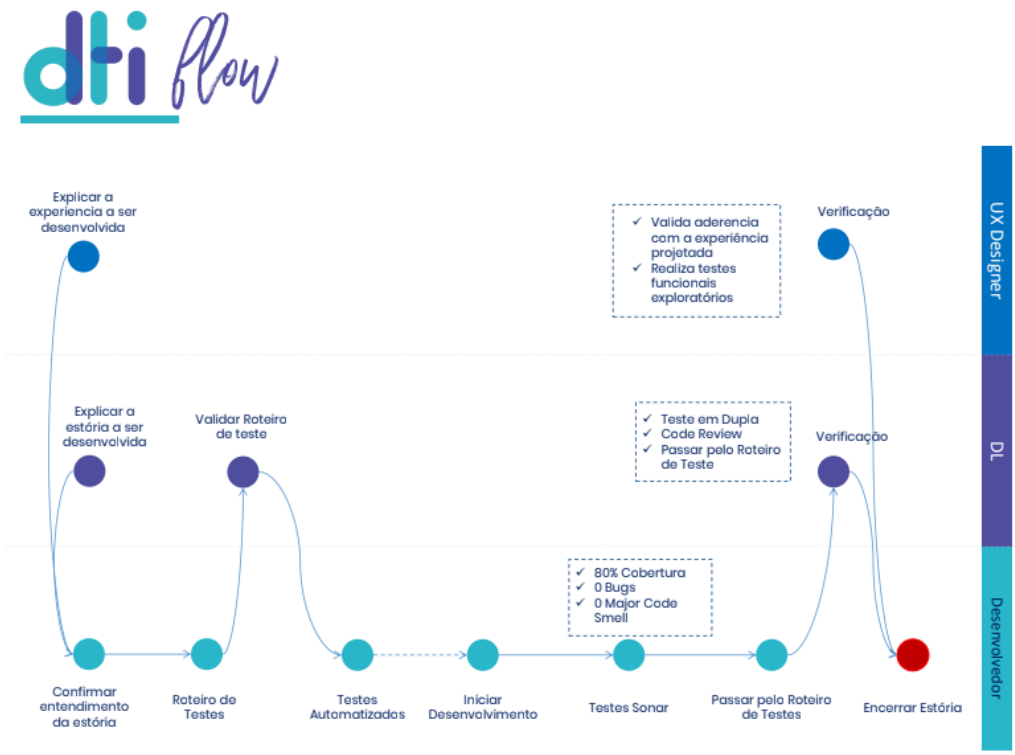
Scrum Master: Scrum *Master*, conforme o próprio nome diz, é aquele que deve assegurar que equipe esteja seguindo fielmente, os ritos do Scrum (CASAGRANDE, 2022). Ele tem papel fundamental no auxílio para identificação de possíveis falhas em alguns processos desta metodologia ágil, para a equipe ter uma operação de entrega contínua e dinâmica.

Tech Manager: Este tem a função principal de alinhar as equipes, considerando suas capacidades, forças, fraquezas e interesses. É papel fundamental do *Tech Manager* alinhar as diversas oportunidades, com os interesses dos colaboradores, sempre procurando aquelas mais satisfatórias ou desafiadoras.

2.7 Dti Flow

O Dti *Flow* se integra ao Scrum, expandindo na própria empresa, uma vez que cada atividade desenvolvida dentro da *sprint* segue um fluxo bem definido, agilizando seu processo de aprovação. Esse fluxo começa com o entendimento da história, passa pelo planejamento e validação de um roteiro de testes, avança para a fase de desenvolvimento e implementação de testes automatizados e, por fim, chega à etapa de validação do que foi realizado. Após a conclusão dessas etapas, uma história é considerada pronta.

Figura 2.1 – Dti Flow



Fonte: Dti Digital

É importante ressaltar que, embora o Scrum e o Dti *Flow* sejam amplamente utilizados na empresa, eles não são obrigatórios e inflexíveis para todas as equipes. Pelo contrário, a Dti

incentiva e promove a adaptação e utilização de métodos convencionais que melhor se adéquem ao contexto de cada projeto em particular.

3 METODOLOGIAS E FERRAMENTAS TÉCNICAS

O projeto no qual o estagiário esteve envolvido durante o seu período de estágio consiste essencialmente em uma aplicação *Front-End* que consome dados provenientes de um *Back-End For Front-End* (BFF). O BFF, por sua vez, tem a função exclusiva de receber dados provenientes de uma API externa, uma vez que a equipe trabalhava em colaboração com parceiros de outra empresa, atendendo a solicitação do cliente. Tais parceiros eram responsáveis por fornecer e executar a maioria das operações relacionadas à consulta e ao tratamento de informações no banco de dados.

3.1 Scrum

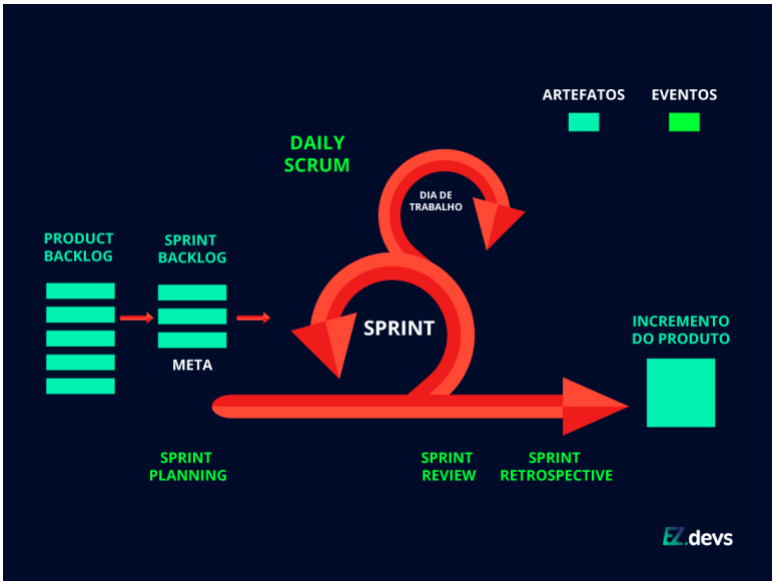
O Scrum é uma abordagem ágil que se baseia na utilização de *sprints*, sendo ciclos de produção, visando garantir revisão e aprimoramento contínuos para alcançar sempre o melhor resultado possível. A colaboração é essencial na implementação dessa estratégia.

Como um dos *frameworks* mais populares, sua popularidade se deve à sua praticidade. Ao adotar ciclos de produção mais curtos, os gestores podem acompanhar o progresso de um projeto gradualmente. Dessa forma, por meio de revisões frequentes, o trabalho final entregue tende a ser de alta qualidade (CASAGRANDE, 2022).

A Figura 3.1 apresenta o fluxo do Scrum segundo (CASAGRANDE, 2022). Nesta figura pode-se observar do lado esquerdo o *Product Backlog* que contém todos os requisitos e demandas do sistema. Em seguida, tem-se a *Sprint Backlog* que seriam as funcionalidades desenvolvidas em determinada *Sprint*. A *Sprint Review* ocorre no final de cada *Sprint*, em que a equipe apresenta o trabalho concluído durante a Sprint aos stakeholders e outras partes interessadas. O objetivo principal da *Sprint Review* é obter feedback dos stakeholders sobre o incremento do produto desenvolvido durante a *Sprint*. Por outro lado, a *Sprint Retrospective* ocorre após a *Sprint Review* e visa a melhoria contínua do processo de desenvolvimento. Durante a retrospectiva, a equipe Scrum reflete sobre a sprint anterior e identifica o que funcionou bem, os desafios enfrentados e oportunidades de melhoria.

A imagem abaixo auxilia no entendimento do fluxo do Scrum.

Figura 3.1 – Fluxo do Scrum



Fonte: Junior (2019)

A metodologia ágil Scrum está presente em todos os *squads* da Dti, onde seus principais rituais, como *daily*, *retrospective*, *sprint*, *planning* e *product backlog*, são constantemente utilizados e seguidos. O estagiário participou ativamente de todos os ritos do Scrum realizados pela sua equipe, o que contribuiu para sua familiarização com essa metodologia ágil.

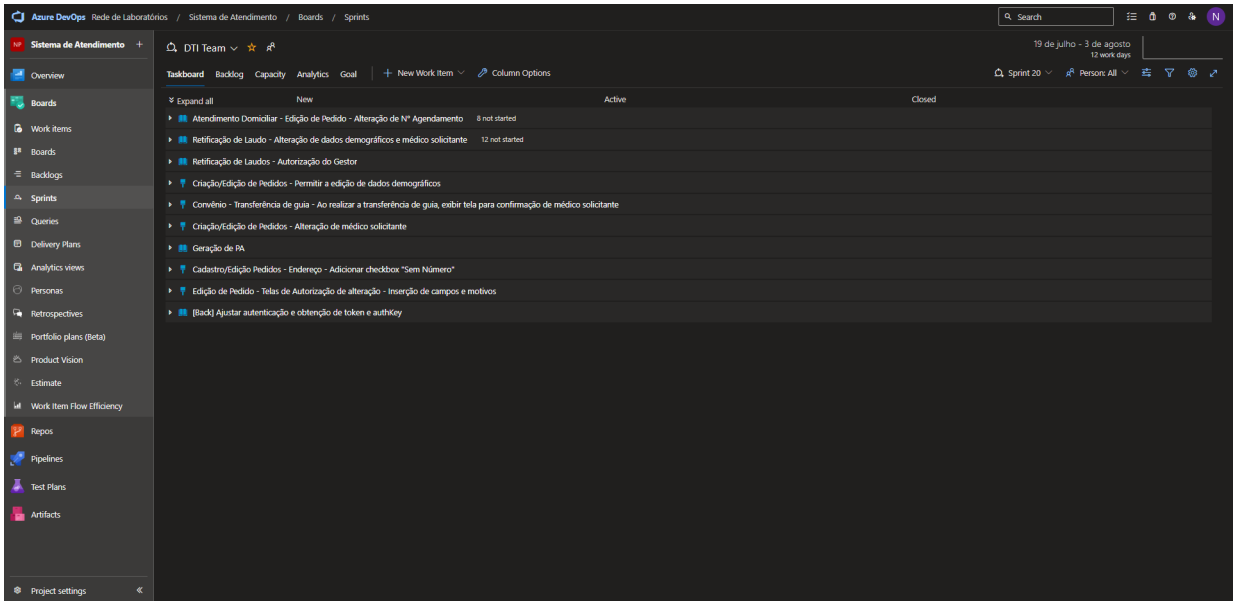
3.2 Azure Devops

O Azure DevOps fornece suporte a uma cultura colaborativa e a um conjunto de processos que reúnem desenvolvedores, gerentes de projetos e colaboradores para o desenvolvimento de software. Permite que as organizações criem e aprimorem produtos em um ritmo mais rápido do que seria possível com abordagens tradicionais de desenvolvimento de software (MICROSOFT, 2022).

A plataforma Azure DevOps mostrou-se uma excelente ferramenta para gerenciar um produto. Ela oferece um espaço para a escrita de histórias com descrições detalhadas sobre as regras de negócio das funcionalidades a serem desenvolvidas. Dentro dessas histórias, é possível criar tarefas com descrições mais técnicas do que precisa ser efetuado, agilizando o trabalho do desenvolvedor vinculado a elas.

Abaixo encontra-se um exemplo do *board* do Azure *Devops*, contendo a listagem de algumas estórias de usuário.

Figura 3.2 – Exemplo do Board Azure



Fonte: Azure Dti Digital + Cliente

Adicionalmente, na plataforma Azure DevOps, o estagiário teve a possibilidade de realizar o controle e versionamento de código por meio do Git (KRIGER, 2022), com armazenamento de *branches* e criação de *pull requests* para aprovação. Após a aprovação do *pull request*, o código era incorporado à *branch* de desenvolvimento, denominada "*develop*". Além desse fluxo, também foi estabelecida uma *branch* específica para a fase de homologação com o cliente, que era uma réplica da *branch* de produção.

3.3 Integração com outras aplicações

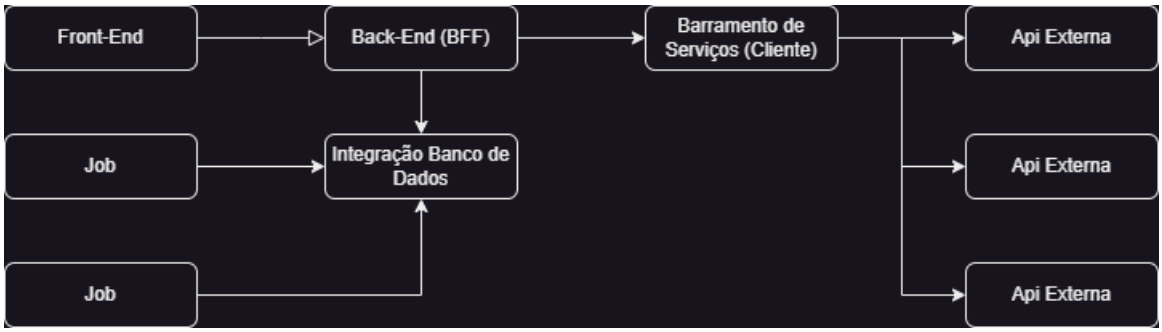
Conforme mencionado na introdução deste capitulo, o projeto do Sistema de Atendimento incorporava dados provenientes de parceiros externos, conforme a preferência do cliente, por meio de um *Back-End For Front-End* (BFF), que fornecia os dados para o *Front-End*. Devido à sua extensa complexidade e abrangência, esse projeto recorria a várias APIs (REDHAT, 2023) externas, relacionadas às funcionalidades presentes no sistema, tais como: API de Pagamento, API de Agendamentos, API de Faturamento, API de Autorização, API de Autenticação, entre outras.

Todas as APIs (REDHAT, 2023) externas empregadas no decorrer do desenvolvimento do projeto desempenharam um papel fundamental na prevenção da sobrecarga da aplicação. Além disso, considerando que o desenvolvimento do Sistema de Atendimento tinha como fundamento um Sistema Legado (TEBET, 2021), várias dessas APIs já se encontravam prontas no

momento do início do desenvolvimento, o que proporcionou facilidade e agilidade no processo de obtenção de dados para fluxos essenciais da aplicação.

A ilustração a seguir representa o mecanismo de comunicação estabelecido entre as aplicações no âmbito do projeto.

Figura 3.3 – Arquitetura Projeto



Fonte: Autor deste documento

3.4 React

O React (NEVES, 2023) é um *framework* JavaScript (PEREIRA, 2023) desenvolvido pelo Facebook cujo objetivo é criar interfaces de usuário em aplicativos web. Reconhecido por sua facilidade de uso, flexibilidade e escalabilidade, o React é amplamente adotado por empresas de tecnologia renomadas, como Facebook, Instagram e Airbnb. Sua popularidade se deve à sua capacidade de simplificar o processo de criação de interfaces interativas e responsivas, permitindo que os desenvolvedores construam aplicativos web modernos e eficientes.

O React é uma tecnologia amplamente empregada atualmente, e o estagiário só teve a oportunidade de entrar em contato com ela graças ao seu período de estágio, ressaltando, assim, a relevância dessa experiência no desenvolvimento do estagiário.

3.5 TypeScript

Considerando a utilização de recursos avançados que são amplamente empregados em projetos de grande envergadura, tais como tipagem estática, forte e automática, orientação a objetos, bem como a capacidade de identificar e corrigir erros em tempo real durante o processo de desenvolvimento, optou-se pelo Typescript (MELO, 2021). O Typescript, desenvolvido pela Microsoft em 2012, foi concebido com o propósito de incorporar recursos e ferramentas que não estão nativamente disponíveis na linguagem JavaScript (PEREIRA, 2023). Tais recursos

incluem a tipagem estática, na qual os tipos das variáveis são explicitamente definidos no código, além do suporte à orientação a objetos.

3.6 Arquitetura Back-End

Conhecida também como arquitetura *Ports and Adapters*, a arquitetura hexagonal (TRINDADE, 2019) é uma abordagem que organiza o código em camadas distintas, cada uma com suas responsabilidades, visando isolar completamente a lógica da aplicação do mundo externo. Esse isolamento é alcançado através do uso de Portas e Adaptadores, onde as Portas são as interfaces expostas pelas camadas de baixo nível, e os Adaptadores são as implementações dessas interfaces.

Essa abordagem de isolamento se aplica tanto à entrada de dados quanto à saída. Isso significa que o código deve ser independente da forma de acesso aos dados e também dos mecanismos de persistência, envio de notificações e outros aspectos relacionados.

Dessa forma, a aplicação se torna mais flexível, permitindo a substituição ou troca desses componentes sem afetar a lógica central da aplicação.

3.7 .NET

A tecnologia adotada para o desenvolvimento do back-end foi o .NET (AMAZON, 2023), que consiste em uma plataforma de desenvolvimento aberta capaz de criar aplicativos para desktop, web e dispositivos móveis, permitindo sua execução nativa em diferentes sistemas operacionais.

O ecossistema .NET engloba uma variedade de ferramentas, bibliotecas e linguagens, fornecendo suporte ao desenvolvimento de software moderno, escalável e altamente eficiente. A plataforma .NET é sustentada por uma comunidade ativa de desenvolvedores dedicados à sua manutenção e suporte contínuos.

Ao ser empregado em conjunto com a linguagem CSharp (GUEDES, 2018), o .NET possibilitou ao estagiário uma prática consistente dos conceitos abordados na disciplina de Práticas de Programação Orientadas a Objetos. O CSharp, por ser uma linguagem orientada a objetos, assemelha-se ao Java (ORACLE, 2023), que foi utilizado na referida disciplina. Dessa forma, a utilização dessas tecnologias proporcionou ao estagiário uma experiência robusta e alinhada com os princípios e práticas da programação orientada a objetos.

A imagem abaixo retrata algumas semelhanças entre as duas tecnologias (Java e CSharp):

Figura 3.4 – Comparação Java e CSharp

Fonte: Araujo (2023)

Conforme observado no exemplo anterior, há uma similaridade entre as tecnologias Java e CSharp quando se trata da instanciação de um novo objeto. No que se refere à atribuição de valores, nota-se que o Java emprega uma função específica chamada “setNome”, geralmente criada na classe do objeto, enquanto o CSharp permite realizar a atribuição de forma mais direta, como demonstrado no exemplo. O mesmo princípio se aplica à recuperação de valores, em que o Java utiliza o método “getNome”, conforme exemplificado anteriormente.

Apesar das pequenas diferenças mencionadas anteriormente, o estagiário não enfrentou grandes desafios ao fazer a transição do Java, aprendido durante o curso universitário, para o CSharp, utilizado durante o período de estágio. Essa facilidade é resultado de um aprendizado sólido sobre os princípios da Programação Orientada a Objetos (HENRIQUE, 2023), adquirido durante o curso de Ciência da Computação. Esses princípios foram inicialmente abordados na disciplina de Estrutura de Dados e posteriormente aprofundados na disciplina de Práticas de Programação Orientadas a Objetos, conforme mencionado anteriormente.

3.8 Postman

Quando trabalha-se com aplicações que se comunicam, é uma prática recomendada testar as requisições que um sistema deve aceitar antes de iniciar a implementação das rotas. Embora as APIs geralmente forneçam documentações detalhadas, pode ser que ocorra a perda de informações durante manutenções que deixam de ser documentadas. Seria frustrante descobrir

que uma não funciona devido à ausência de algum campo desconhecido após a sua implementação estar completa.

Dito isso, foi utilizado o Postman (VERSIANI, 2023) no projeto, sendo justamente um cliente que auxilia no teste de requisições de algumas API, facilitando o desenvolvimento de manutenção das aplicações que as consomem.

3.9 Dbeaver

Apesar do projeto se basear principalmente no consumo de informações provenientes de APIs externas e sua posterior transmissão para o Front-End, em algumas situações pontuais tornou-se necessário o acesso direto ao banco de dados. Essa necessidade ocorreu devido à indisponibilidade de informações essenciais para o desenvolvimento de fluxos específicos do Front-End por meio das APIs externas já existentes. Assim, mesmo com o Back-End atuando principalmente como um BFF, houve a eventual realização de acesso direto ao banco de dados. Para consultas, inserção ou remoção de dados no banco, foi utilizado o Dbeaver (PPLWARE, 2017).

3.10 REST

A aplicação back-end do sistema foi desenvolvida seguindo o padrão arquitetural REST (REDHAT, 2020)). O REST define um conjunto de princípios para a criação de APIs web escaláveis e padronizadas. Ao adotar o REST, o sistema garante uma estrutura coerente e eficiente para expor e manipular recursos, utilizando os verbos HTTP de forma semântica. Essa abordagem proporciona uma arquitetura flexível, de fácil manutenção e facilita a comunicação entre diferentes sistemas, permitindo uma expansão e evolução contínuas.

4 ATIVIDADES DESENVOLVIDAS

Neste capítulo são destrinchadas algumas atividades desenvolvidas pelo estagiário durante o seu período de estágio na Dti. Tais tarefas que serão citadas englobam tanto aspectos técnicos, quanto aspectos voltados para gestão de processos.

Embora o autor deste documento tenha se concentrado principalmente no desenvolvimento de software, também foi de extrema importância que o mesmo se familiarizasse e adotasse os ritos comuns dentro da Dti, juntamente com sua equipe, a fim de garantir que o projeto seguisse os padrões exigidos pela empresa, que valoriza a agilidade.

4.1 Ritos da empresa

Nesta seção, são apresentados alguns rituais comuns da empresa, que fazem parte da rotina de trabalho, sendo executados uma vez a cada *sprint*, e visam atuar como *checklists* regulares, com diversas perguntas de múltipla escolha, para validar a operação completa de um *squad*. Esses rituais abordam questões fundamentais para o desenvolvimento de software, como segurança, qualidade de código, qualidade do produto e operações.

4.1.1 Check de Engenharia

O *Check* de Engenharia é direcionado principalmente para a equipe técnica, incluindo desenvolvedores e arquitetos. Como o nome sugere, visa abordar aspectos estruturais do software e características relevantes para a equipe técnica, como manutenibilidade, modularidade, testes automatizados, logs, ambientes, publicações, versionamento, documentação, entre outros.

Neste check, o estagiário pode contribuir, como dono do check, isto é, aquele que marca a agenda do check, que era realizado em duplas alternadas durante cada sprint. O foco principal do check seria verificar algumas questões arquiteturais de projeto e, além disso, acompanhar as métricas fornecidas pelo Sonar Qube a cada sprint, assim, comparando-as e propondo ações de melhora, caso houvesse surgimento de *code smells, bugs e diminuição da cobertura de código*.

Era acordado com a equipe uma quantidade mínima de aceite para bugs, code smells e vulnerabilidades no código-fonte, identificadas no SonarQube. Quando essas métricas excediam o limite, a equipe criava planos de ação com tarefas abordadas durante a *Sprint*. O estagiário se envolveu no desenvolvimento de alguns desses planos de ação, permitindo que

ele contribuísse para a melhoria das métricas do SonarQube, como o aumento da cobertura de código por testes unitários no front-end, desenvolvendo testes de componentes.

Segue abaixo uma imagem exemplificando o desenvolvimento de um desses testes:

Figura 4.1 – Exemplo teste unitário

```
5   import React from 'react'
6   import { render, screen } from '@testing-library/react'
7   import Button from './Button'
8
9   describe('<Button />', () => {
10    it('should render the button', () => {
11      render(<Button href="/home" text="Click me"/>)
12
13      const button = screen.getByRole('button')
14
15      expect(button).toBeInTheDocument()
16    })
```

Fonte: (VELTRONI, 2022)

4.1.2 Check de Segurança

O Check de Segurança é geralmente conduzido com o auxílio de um especialista na área de segurança computacional, visando verificar se o projeto segue normas como a LGPD (Lei Geral de Proteção de Dados). Além disso, esse processo tem a finalidade de identificar possíveis vulnerabilidades no projeto e propor soluções para mitigá-las.

O estagiário inicialmente participou como ouvinte no *Check* de Segurança, mas ao adquirir conhecimento, conseguiu contribuir ativamente. Foi rodado um *pentest*, e identificadas algumas falhas de seguranças, e o estagiário pode assumir algumas demandas, chamadas de planos de ação para mitigar as falhas encontradas. Algumas dessas demandas foram: minificação do código *front-end* para que o sistema de build ou a ferramenta responsável pela compilação do código não criasse os arquivos de *sourcemaps*, economizando espaço e protegendo o código-fonte original do projeto contra acesso não autorizado, onde foi criada uma variável de ambiente no arquivo *.env*, chamada *"GENERATE SOURCE MAP = false"*.

Segue abaixo uma imagem de exemplo da variável de ambiente criada:

Figura 4.2 – Exemplo de variável de ambiente "GENERATE SOUCE MAP"

```
GENERATE_SOURCEMAP=false
```

Fonte: Código do Projeto

Além disso, houve a configuração da tag *Csp*, *Content Security Policy*, que permite que os proprietários de *websites* especifiquem uma série de regras e diretrizes sobre como o navegador deve se comportar em relação aos recursos carregados e executados em suas páginas. Essas regras ajudam a evitar que recursos não confiáveis ou potencialmente maliciosos, como *scripts*, estilos, imagens e *plugins*, sejam carregados e executados em uma página da web. Para isso, foi criada uma *tag* no arquivo de configuração do projeto, chamado de "web.config".

Segue abaixo uma imagem de exemplo da tag de segurança criada:

Figura 4.3 – Exemplo de tag CSP

Fonte: (COLOMBO, 2018)

4.1.3 Check de Produto e Design

Ao combinar dois pilares altamente complementares, o *Check* de Produto e Design promove discussões sobre o valor entregue ao cliente, como esse valor pode ser medido, como ele se manifesta visualmente e como a equipe compreende o que está sendo desenvolvido. Esse momento é importante para avaliar como as atuações dos principais responsáveis por essas duas áreas (*Product Owner* e *Product Designer*) são percebidas e assimiladas pelo restante da equipe.

Apesar do *Check* de Produto e Design focar principalmente nas questões relacionadas ao produto e ao design, como os processos de descoberta e usabilidade do usuário, é de extrema importância que a equipe de desenvolvimento participe desse check. Isso permite que a equipe de produto tenha uma visão abrangente que envolve os processos de negócio, a escrita de histórias de usuário e o protótipo do projeto. O estagiário teve a oportunidade de contri-

buir nessa área e adquirir experiência, utilizando conhecimentos adquiridos na disciplina de Interação Humano-Computador cursada durante sua graduação.

Durante o *Check* de Produto e *Design*, o estagiário, juntamente com sua equipe, se reunia para discutir maneiras mais eficazes de aprimorar a geração de valor do produto, assim como aperfeiçoar a frequência das entregas. Nesse encontro, eram analisados os resultados obtidos até o momento, identificando oportunidades de aprimoramento nos processos e na experiência do usuário. As discussões focavam em como agregar mais valor aos clientes por meio de novas funcionalidades, otimizações e ações estratégicas, bem como em como aumentar a agilidade das entregas para responder mais prontamente às demandas do mercado. Esse espaço de análise e reflexão era essencial para impulsionar a qualidade e a inovação contínua do produto, fortalecendo a competitividade da empresa no cenário atual.

4.1.4 Check de Operações

O *Check* de Operações abrange os processos em si, englobando todos os pilares mencionados anteriormente. A Dti adota um fluxo recomendado que guia todo o processo de trabalho, integrando a metodologia ágil. O Check de Operações questiona fatores relevantes sobre os rituais de forma geral e, acima de tudo, promove o alinhamento da equipe em relação à sua situação atual, bem como aos entendimentos e divergências de perspectivas entre todos os membros.

Este *Check* está diretamente relacionado à forma como a equipe opera. Nesse contexto, o estagiário teve a oportunidade de contribuir na participação e, em algumas ocasiões, até na apresentação desse *Check* para o *squad*. O estagiário pôde abordar questões como a felicidade e a integração com o time, além de promover críticas construtivas em conjunto com toda a equipe, identificando processos que não estavam sendo executados corretamente e propondo ações de melhoria.

Entre os diversos planos de ação desenvolvidos para aprimorar a eficiência da equipe durante a *Sprint*, o estagiário destacou uma sugestão: a divisão das reuniões de Refinamento Técnico. Observando que a reunião de duas horas realizada anteriormente não estava sendo tão produtiva quanto o desejado, o estagiário propôs dividir esse encontro em sessões menores e mais focadas. Essa abordagem permitiria uma análise mais detalhada dos aspectos técnicos do projeto, tornando as discussões mais concisas e efetivas. Com a nova dinâmica proposta pelo estagiário, a equipe poderia otimizar o tempo e obter resultados mais significativos durante o processo de refinamento técnico, aprimorando, assim, a operação do time ao longo da *Sprint*.

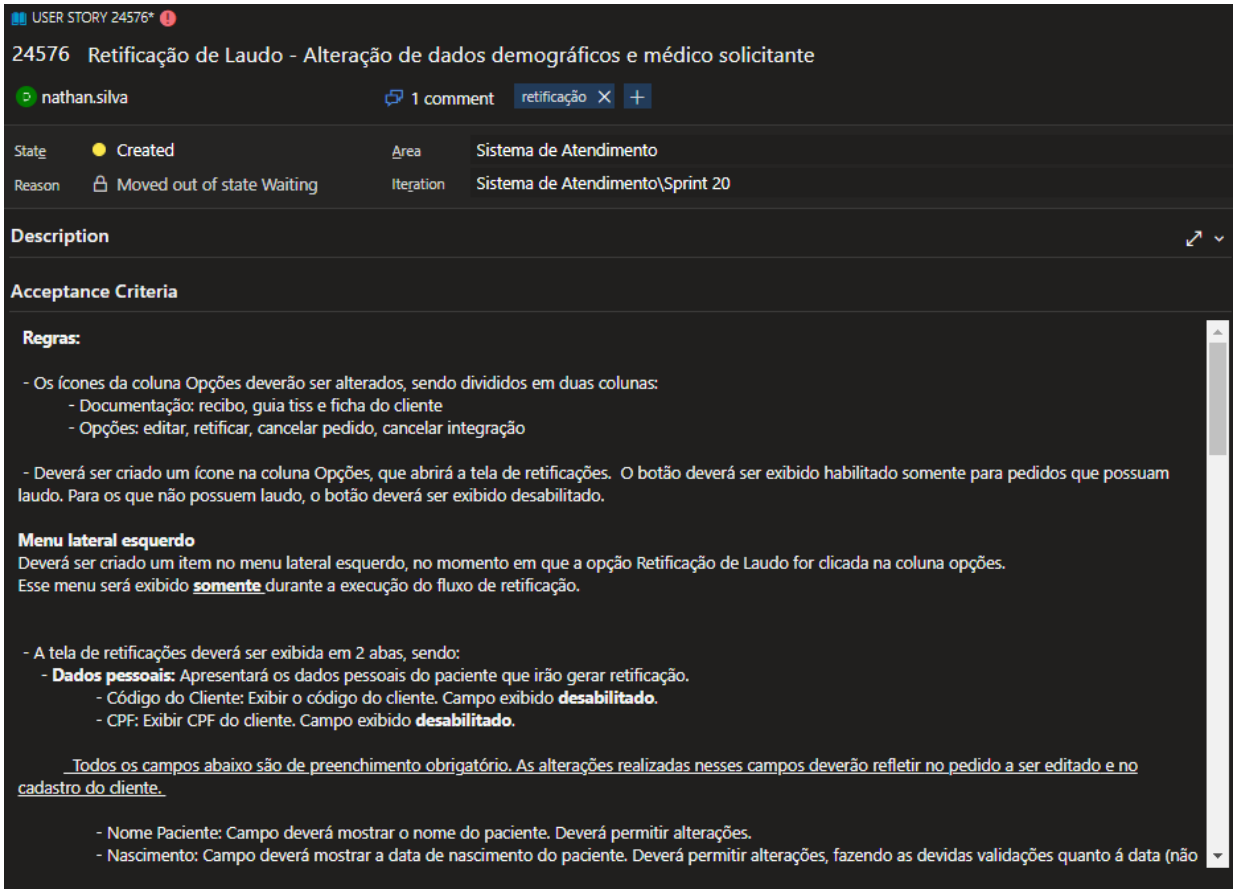
4.2 Refinamento Técnico

Durante o estágio, o estagiário desempenhou um papel significativo nos refinamentos técnicos das tarefas antes de serem incorporadas à *sprint*. Inicialmente, o estagiário participava como observador nesses processos, adquirindo familiaridade com as práticas e abordagens utilizadas pela equipe. À medida que adquiria conhecimento e confiança, o estagiário assumiu gradualmente mais responsabilidades nessa etapa, até conseguir conduzir de forma autônoma o refinamento das estórias.

Durante os refinamentos técnicos, a equipe se reunia para revisar as estórias de usuário planejadas para a próxima *sprint*. Nesse processo, eram analisados detalhadamente os requisitos, funcionalidades e comportamentos esperados para cada estória. Eram discutidos e esclarecidos os aspectos técnicos das estórias, como a arquitetura, a lógica de negócio envolvida, as integrações com outros sistemas e as possíveis dependências. O estagiário, sob a orientação da equipe, acompanhava essas discussões, absorvendo os conhecimentos compartilhados e contribuindo com perguntas e observações pertinentes.

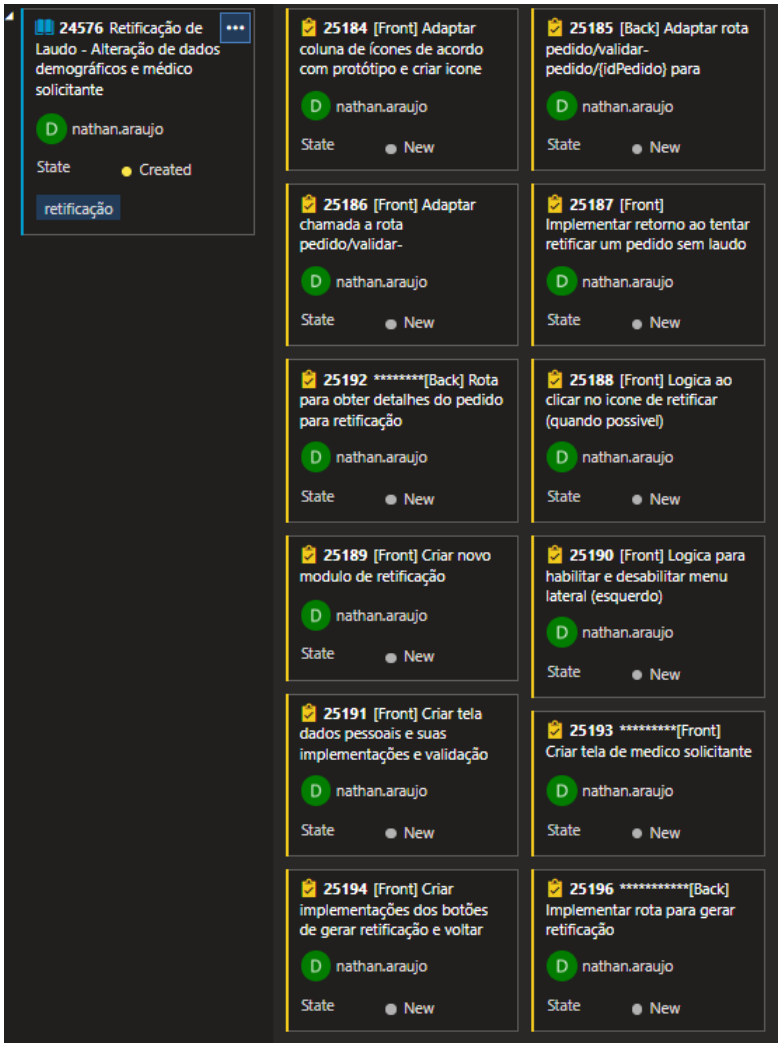
Abaixo serão exemplificadas imagens das estórias antes do refinamento técnico, apenas com as regras de negócio, e depois do refinamento, com as tarefas já criadas.

Figura 4.4 – Exemplo de estória antes do refinamento



Fonte: Azure Dti Digital + Cliente

Figura 4.5 – Exemplo de estória após o refinamento



Fonte: Azure Dti Digital + Cliente

À medida que o estagiário se familiarizava com o projeto e ganhava confiança em suas habilidades, ele começou a assumir responsabilidades mais ativas nos refinamentos técnicos. Ele passou a analisar as estórias, identificar possíveis problemas ou inconsistências, sugerir melhorias e propor soluções técnicas adequadas. Esse processo de refinamento permitiu ao estagiário aprimorar suas habilidades de análise crítica, compreender os desafios técnicos do projeto e contribuir efetivamente para o planejamento e a definição das tarefas a serem desenvolvidas. Essa prática proporcionava um alinhamento técnico consistente entre os membros da equipe, garantindo uma compreensão compartilhada dos requisitos e facilitando o desenvolvimento eficiente das funcionalidades.

A oportunidade de participar ativamente dos refinamentos técnicos proporcionou ao estagiário uma valiosa exposição aos aspectos práticos e desafios do desenvolvimento de software. Ao trabalhar em estreita colaboração com a equipe, ele teve a oportunidade de aprender com

profissionais experientes e ampliar sua compreensão sobre as melhores práticas e abordagens utilizadas no contexto do projeto.

4.3 Desenvolvimento de interfaces (Front-End)

O *front-end* do sistema de atendimento foi desenvolvido utilizando as tecnologias React e TypeScript. Essas tecnologias são amplamente reconhecidas e utilizadas na construção de interfaces de usuário modernas e robustas. A combinação dessas tecnologias permitiu o desenvolvimento de um *front-end* poderoso, com componentes reutilizáveis, gerenciamento eficiente de estados e integração com APIs de *back-end*.

O estagiário continua envolvido no projeto até a data atual em que este documento está sendo redigido, ou seja, aproximadamente 1 ano e 6 meses de dedicação ao projeto. Com essa extensiva experiência, é inviável descrever todas as atividades desempenhadas pelo estagiário no *front-end* do projeto. No entanto, algumas atividades foram selecionadas por exigirem maior dificuldade e tempo de desenvolvimento por parte do estagiário. Essas atividades também proporcionaram um valor significativo para o produto final, o Sistema de Atendimento.

4.3.1 Possibilitando o pagamento para Pessoa Jurídica (PJ)

O projeto no qual o estagiário estava envolvido se tratava de uma aplicação muito completa e com diversas funcionalidades. O objetivo dessa completude era facilitar o atendimento realizado pelas atendentes que trabalhavam na rede de laboratórios, proporcionando-lhes, de forma centralizada, diversas ações em um único sistema, de forma moderna e fácil de ser utilizada.

Dentre as diversas funcionalidades presentes no sistema de atendimento, uma delas consistia na capacidade de efetuar o pagamento de um pedido realizado. Para uma compreensão mais precisa, o termo “pedido” utilizado neste documento refere-se a um conjunto de exames associados a um cliente e a um convênio. No caso em que determinado convênio não cobria um determinado exame, o pagamento seria processado diretamente por meio do Sistema de Atendimento.

Conforme mencionado anteriormente, já havia a funcionalidade de pagamento no pedido, a qual foi desenvolvida previamente por outros membros da equipe envolvida no projeto. No entanto, essa funcionalidade estava restrita apenas a pagamentos realizados por pessoas físicas, ou seja, por indivíduos. Surgiu, então, a necessidade de implementar o pagamento por

pessoa jurídica, que representa uma entidade, como uma empresa, sociedade ou organização. Esse desafio foi atribuído ao estagiário.

Inicialmente, utilizou-se como base o que já havia sido desenvolvido para o pagamento por pessoa física. No entanto, foram incorporadas as peculiaridades relacionadas às pessoas jurídicas no que diz respeito ao tratamento dos dados. Por exemplo, pessoas jurídicas possuem um CNPJ, enquanto pessoas físicas possuem um CPF. A utilização do React foi de grande utilidade para simplificar o desenvolvimento dessa funcionalidade. Como os dados eram armazenados em um estado global (React-Redux), eles podiam ser acessados em qualquer parte da aplicação. Dessa forma, os dados de pagamento, como os exames a serem pagos e os respectivos valores, estavam facilmente disponíveis para o estagiário durante o desenvolvimento.

A princípio, após a conclusão do pedido, havia duas formas de redirecionamento para o fluxo de pagamento: havia a possibilidade de redirecionamento direto, por meio de um botão, ou a opção de colocar o pedido como pendente de pagamento, direcionando-o para uma espécie de fila.

Ao ingressar no fluxo de pagamento, o usuário se deparava com um formulário contendo os dados do pagador, que, inicialmente, correspondia ao próprio cliente responsável pelo pedido. No entanto, também era oferecida a opção de selecionar outro pagador, uma pessoa física já cadastrada no sistema. Essa seleção era realizada por meio de um modal exibido na tela, o qual proporcionava filtros de pesquisa para facilitar a identificação do pagador desejado.

Para viabilizar o pagamento de pessoa jurídica, o estagiário empregou uma lógica que operava da seguinte maneira: foi incorporado um *Radio Group*, fornecido pela biblioteca de componentes React, MUI (MUI, 2023). Esse componente indicava se o pagador era uma pessoa física ou uma pessoa jurídica. Com base na seleção, os filtros eram ajustados de acordo com cada tipo de pagador, e uma rota de pesquisa específica era acessada, considerando o tipo de pagador selecionado e os filtros preenchidos.

Dessa forma, a obtenção da lista de pagadores conforme o tipo e filtros selecionados, se tornou dinâmica, e não houve a necessidade de grandes mudanças no componente que já era utilizado anteriormente.

Após a implementação da funcionalidade de retornar a lista de pagadores PJ (pessoa jurídica) conforme os filtros selecionados, o estagiário utilizou outro componente da biblioteca MUI, denominado *Data Grid*. Esse componente consiste em uma tabela que exibe os dados preenchidos. Após a obtenção dos dados do *back-end*, o *Data Grid* era renderizado logo abaixo

dos filtros. No entanto, antes da realização da consulta, ou seja, quando nenhum dos dados necessários havia sido filtrado, um espaço em branco era exibido abaixo dos filtros, indicando a ausência de dados a serem exibidos.

Após a implementação do *Data Grid* e o preenchimento dos dados provenientes do back-end, cada item da lista exibia uma opção de seleção. Ao clicar nessa opção, o usuário era redirecionado para um formulário mais completo contendo os dados detalhados do pagador selecionado. Esses dados não eram os mesmos obtidos anteriormente, que eram mais simplificados e destinados apenas à exibição na tabela. Os dados presentes nesse formulário mais completo eram mais abrangentes e dependiam da seleção do pagador para a obtenção das informações específicas relacionadas a ele.

A obtenção dos dados foi realizada por meio do uso do *hook useEffect()* (REACT, 2023) do React. Esse *hook* funciona da seguinte maneira: ele executa um trecho específico de código após a renderização do componente na tela ou quando alguma variável listada em seu *array* de dependências sofre uma alteração. A seguir, é apresentado um exemplo de código ilustrativo:

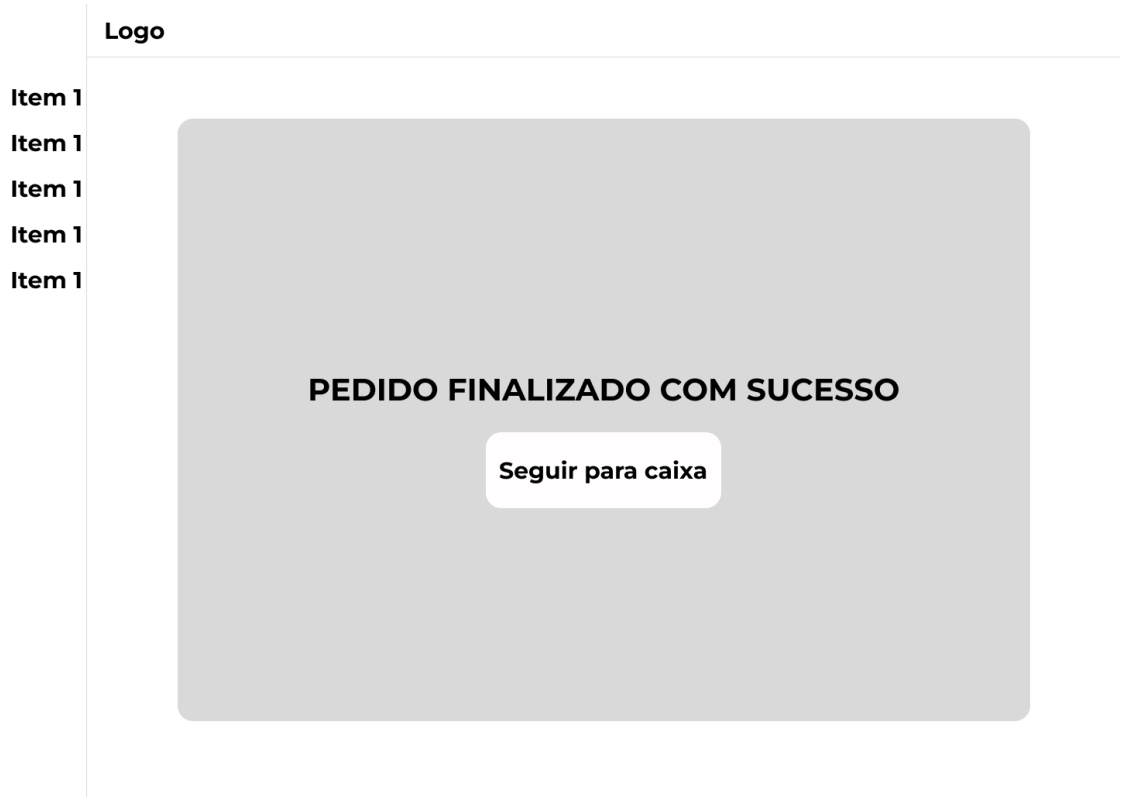
Então, conforme descrito acima, segundo o tipo de pagador selecionado, um formulário diferente seria renderizado, e após a renderização, os dados do pagador selecionados eram obtidos em uma função chamada dentro do *useEffect()*, e assim o formulário era preenchido na tela com os dados retornados do back-end.

O processo de preenchimento quase automático mencionado foi viabilizado, na maioria, pelo uso do *react-redux*, mencionado anteriormente. Por meio dessa integração, o estagiário pôde armazenar os dados retornados do *back-end* no estado global da aplicação, em um objeto específico designado para esse propósito. Assim, ao atribuir o atributo *value* aos campos do formulário, era possível fazer referência direta aos respectivos atributos correspondentes no estado global, facilitando significativamente o processo de preenchimento dos dados.

Segue abaixo um exemplo do fluxo completo desenvolvido:

Na Figura 4.6, é exibida a tela de conclusão, que representa o estado posterior à finalização da criação de um pedido.

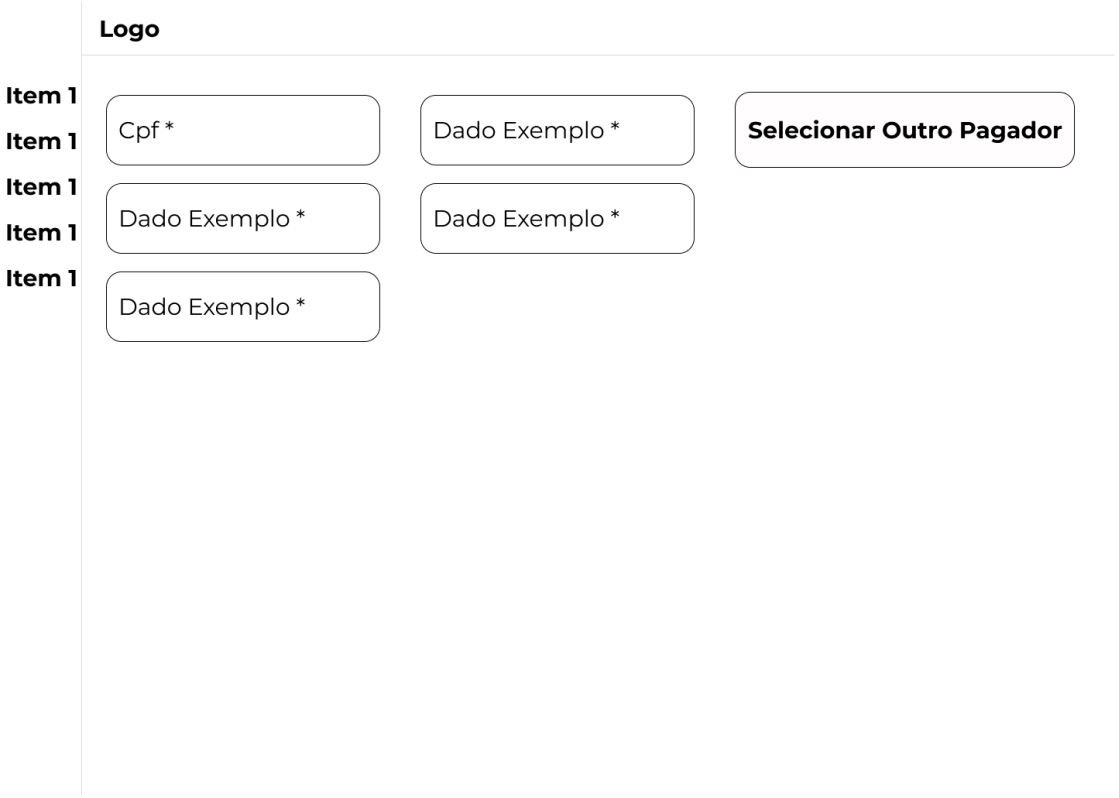
Figura 4.6 – Tela de Conclusão



Fonte: Autor deste documento

Na Figura 4.7, é apresentada a interface de dados do pagador, na qual o usuário é redirecionado após clicar no botão "Seguir para caixa"na tela de conclusão.

Figura 4.7 – Tela de Dados do pagador



Fonte: Autor deste documento

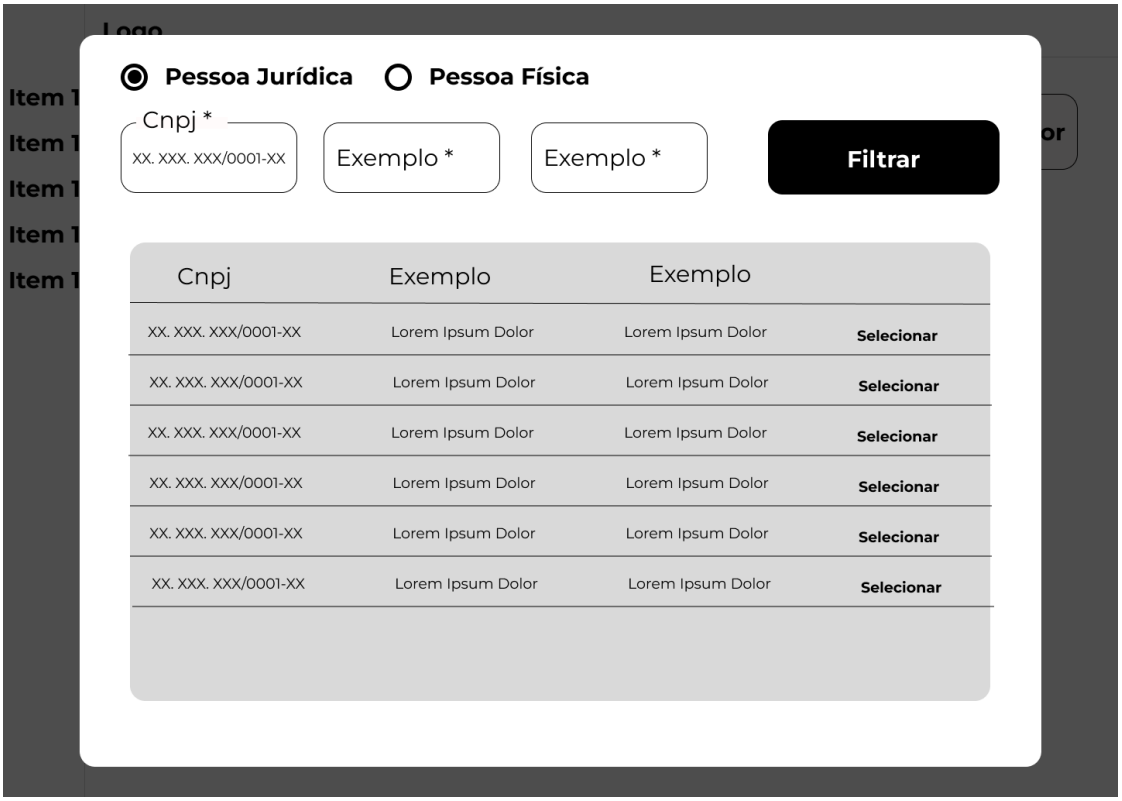
Na Figura 4.8, é exibido o modal de seleção do pagador, para o qual o usuário é redirecionado após clicar no botão “Selecionar outro pagador” na tela de dados do pagador.

Figura 4.8 – Modal de seleção do pagador

Fonte: Autor deste documento

Na Figura 4.9, é apresentado o modal de seleção do pagador, no qual os dados estão filtrados após o clique no botão "Filtrar".

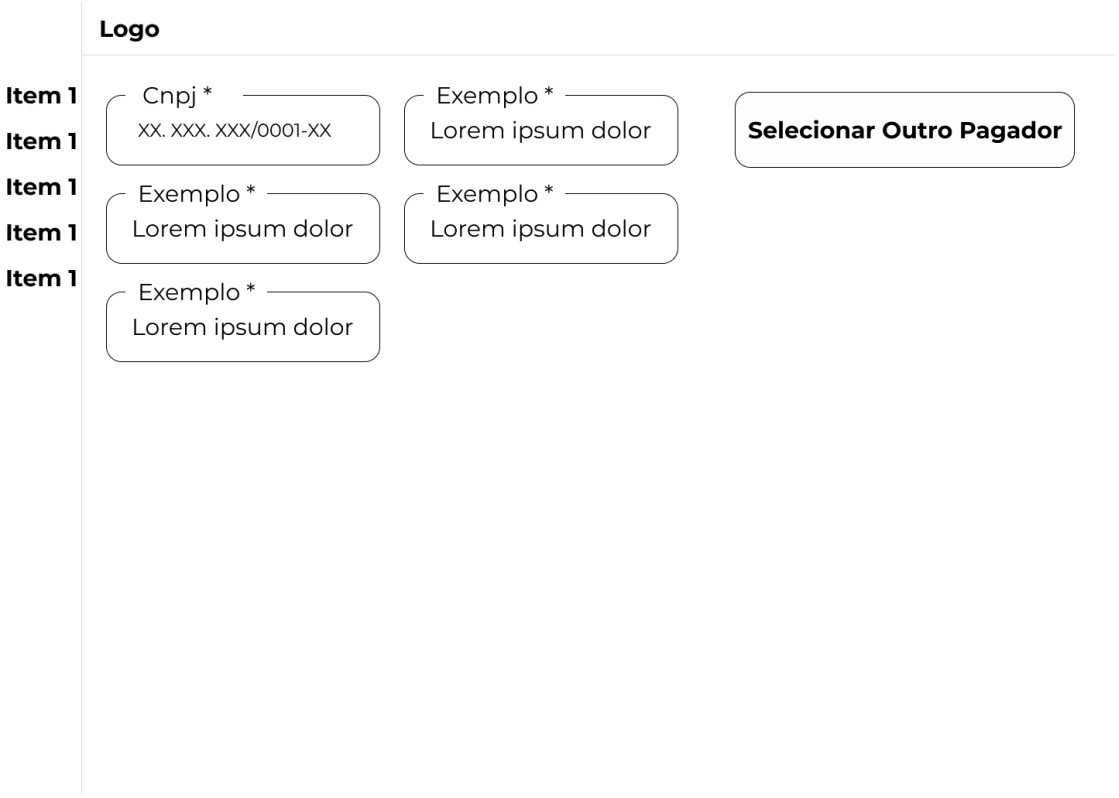
Figura 4.9 – Modal de seleção do pagador após filtro dos dados



Fonte: Autor deste documento

Na Figura 4.10, é exibido o formulário de dados do pagador, contendo as informações preenchidas após a seleção do pagador no modal anterior.

Figura 4.10 – Formulário de Dados do pagador com os dados preenchidos após seleção do pagador



Fonte: Autor deste documento

4.3.2 Utilização de biblioteca para validar formulário

Quando o estagiário foi introduzido no projeto do Sistema de Atendimento, as validações dos formulários eram feitas via outro *hook* do React chamado *useState()* (SILVA, 2023). Esse hook era uma funcionalidade que possibilitava a criação de estado em um componente usando uma função e encarregava-se de gerenciar o estado local do componente, retornando um *array* como resultado. Essa abordagem foi adotada para facilitar a implementação das validações nos formulários no projeto.

Eis abaixo um exemplo de código:

Figura 4.11 – Exemplo de Uso *useState*

```
import React, { useState } from 'react';
function ListaDeRepositorios() {
  const [repositorio, setRepositorio] = useState([]);
  …
 return (
    <>
      ...
    </>
  );
}
export default ListaDeRepositorios;
```

Fonte: Silva (2023)

Os *states* eram utilizados para verificar a obrigatoriedade de um campo, disparar erros e impedir a submissão do formulário caso houvesse algum erro no campo, entre outras funcionalidades. No entanto, essa abordagem era considerada custosa e tinha um impacto significativo no desempenho de uma aplicação React.

O time decidiu adotar uma estratégia gradual durante as *sprints* para implementar o uso de uma biblioteca própria de validação de formulários. Essa ação tinha como objetivo reduzir o peso da aplicação e melhorar a velocidade de execução. A biblioteca escolhida para essa finalidade foi o *react-hook-forms*, também conhecido como RHF (BEEKAI, 2023). Além disso, a equipe utilizou a biblioteca Yup (FORMIK, 2020) em conjunto com o RHF para realizar a validação dos formulários já existentes no projeto. Essa combinação de bibliotecas foi considerada adequada para atender às necessidades de validação eficientemente e melhorar o desempenho global da aplicação.

O estagiário foi encarregado de assumir a tarefa de adequar o formulário de cadastro de convênio com a biblioteca de formulários, uma vez que ele já tinha experiência prévia com essa biblioteca em projetos pessoais. A escolha desse formulário específico para a adequação foi feita visando iniciar a migração gradual do uso da biblioteca nos diferentes componentes do sistema. Essa decisão considerou a familiaridade do estagiário com a biblioteca e a importância do formulário de cadastro de convênio no contexto do projeto.

A ideia era que o Yup atuasse nas validações específicas de cada campo, como a tipagem do mesmo, número de caracteres e sua obrigatoriedade, por exemplo, enquanto o RHF atuava na

captura de possíveis erros encontrados em cada campo, baseando-se no *schema* fornecido pelo Yup. Ao realizar a submissão do formulário, erros eram disparados na tela, indicando ao usuário os campos que deveriam ter seus dados corrigidos, caso houvesse alguma inconsistência no preenchimento. Essa abordagem combinada entre o Yup e o RHF visava garantir uma validação abrangente e eficiente dos dados do formulário, proporcionando uma melhor experiência ao usuário ao identificar e corrigir possíveis erros antes da submissão.

Eis abaixo um exemplo de código do uso das bibliotecas (*react-hook-forms*, e *Yup*):

Figura 4.12 – Exemplo de uso RHF

```
import React from "react";
import { useForm } from "react-hook-form";

export default function Form({ defaultValues, children, onSubmit }) {
  const methods = useForm({ defaultValues });
  const { handleSubmit } = methods;

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      {React.Children.map(children, child => {
        return child.props.name
          ? React.createElement(child.type, {
              ...{
                ...child.props,
                register: methods.register,
                key: child.props.name
              }
            })
          : child;
      })}
    </form>
  );
}
```

Fonte: Beekai (2023)

Figura 4.13 – Exemplo de uso Yup

```
import React from 'react';
import { Formik, Form, Field } from 'formik';
import * as Yup from 'yup';

const SignupSchema = Yup.object().shape({
  firstName: Yup.string()
    .min(2, 'Too Short!')
    .max(50, 'Too Long!')
    .required('Required'),
  lastName: Yup.string()
    .min(2, 'Too Short!')
    .max(50, 'Too Long!')
    .required('Required'),
  email: Yup.string().email('Invalid email').required('Required'),
});
```

Fonte: Formik (2020)

4.4 Familiarização com o padrão arquitetural REST

O estagiário foi incumbido a se familiarizar com o padrão REST, utilizando os conhecimentos adquiridos na disciplina de Sistemas Distribuídos para compreender e aplicar os princípios do mesmo.

O estagiário demonstrou habilidade ao aplicar os princípios do REST. Ele conseguiu entregar um trabalho consistente e alinhado às diretrizes estabelecidas, seguindo um roteiro pré-definido. As atividades desempenhadas envolviam consumir dados de uma API externa, realizar os tratamentos necessários e repassá-los adequadamente para o *front-end*. O estagiário desempenhou suas tarefas de maneira competente, assegurando a integração eficiente entre os sistemas e atendendo aos requisitos estabelecidos.

4.5 Considerações Finais

Durante sua atuação no *front-end* em React, o estagiário demonstrou habilidade e facilidade em se adaptar às tarefas designadas. No entanto, foi identificado um desafio específico relacionado à sua familiarização com o padrão REST e as rotas *back-end* desenvolvidas.

Um dos maiores desafios enfrentados pelo estagiário foi sua participação no refinamento técnico. No começo, ele encontrou certa dificuldade em contribuir efetivamente durante essas

sessões. No entanto, com o passar do tempo, ele pôde aprimorar sua visão de negócio do produto, compreendendo melhor as necessidades dos usuários e aperfeiçoando suas habilidades em identificar as melhores formas técnicas de desenvolvimento.

Além disso, o estagiário também se esforçou para melhorar sua comunicação com a equipe durante os refinamentos. Ele percebeu a importância de uma comunicação clara e eficiente, especialmente ao discutir questões técnicas e propor soluções. Ao longo do estágio, suas habilidades de comunicação evoluíram consideravelmente, permitindo uma interação mais fluida e produtiva com os demais membros da equipe.

A Dti Digital, empresa onde foi realizado o estágio, possui uma política de *feedbacks*, que consiste em formulários de autoavaliação nos quais os funcionários podem solicitar avaliações de seus colegas de equipe. Nesse sentido, após receber diversos feedbacks, pode-se concluir que o estágio apresentou evolução nos seguintes pontos, conforme os objetivos e intenções iniciais estabelecidos para o estágio:

- Habilidades de negociação, melhoria na comunicação e proatividade;
- Conhecimentos técnicos aprimorados;
- Contribuir para o crescimento da empresa.
- Introdução no mercado de trabalho, para vivenciar o dia-a-dia com problemas reais para resolução.

Durante o estágio, a empresa demonstrou paciência e cautela em relação ao estagiário, permitindo que ele se adaptasse em seu próprio ritmo, a fim de alcançar os objetivos mencionados anteriormente com sucesso. Um exemplo da flexibilidade da empresa em relação à adaptação do estagiário foi a oportunidade concedida para estudos quando ele ingressou na empresa, e somente após esse período ele foi designado para um projeto.

No geral, o estagiário, embora tenha enfrentado alguns desafios no início, demonstrou grande determinação e capacidade de aprendizado ao superá-los. A experiência adquirida durante esse estágio certamente contribuiu para o crescimento profissional do estagiário. Sua capacidade de se adaptar a novas tecnologias e enfrentar desafios técnicos são atributos valiosos que serão de grande utilidade em sua carreira futura.

5 CONCLUSÃO

É inegável que a área da tecnologia é altamente dinâmica e está em constante evolução. Tecnologias amplamente utilizadas hoje podem se tornar obsoletas e irrelevantes em breve. Diante dessa dinamicidade, as grades curriculares das universidades muitas vezes não conseguem acompanhar todas as mudanças e inovações que ocorrem. Nesse contexto, o estágio se torna uma oportunidade essencial. O estagiário pode constatar que algumas tecnologias e práticas que ele teve acesso durante esse período não teriam sido abordadas em sua formação acadêmica, caso não fosse pelo estágio. O estágio proporciona ao estagiário a exposição a novas tecnologias e práticas em tempo real, permitindo um aprendizado atualizado e aplicado diretamente no contexto profissional.

No entanto, é importante ressaltar que alguns conceitos aprendidos na universidade desempenharam um papel crucial na motivação do estagiário em ingressar na área de Engenharia de Software, embora nem todos pudessem ser aplicados eficazmente durante o curso. Essa lacuna foi preenchida durante o período de estágio, proporcionando ao estagiário a oportunidade de praticar e vivenciar conceitos que antes eram abordados de maneira mais teórica. Um exemplo significativo disso foi o framework Scrum (CASAGRANDE, 2022), que foi introduzido durante disciplinas como Gerência de Projetos de Software e Processos de Software de forma teórica, e posteriormente praticado de maneira mais incisiva e detalhada ao longo de todo o período em que o estagiário esteve envolvido no projeto do Sistema de Atendimento durante seu estágio.

A participação do estagiário nos refinamentos técnicos foi de grande importância para o seu desenvolvimento profissional. Essa experiência permitiu ao estagiário aprimorar seus conhecimentos teóricos, proporcionando uma base sólida para a aplicação prática desses conhecimentos. Além disso, estar envolvido no processo de refinamento permitiu ao estagiário compreender melhor as necessidades do projeto e contribuir de forma mais efetiva para a definição e planejamento das tarefas.

É fundamental enfatizar que os conceitos teóricos aprendidos durante o curso de Ciência da Computação, juntamente com os conceitos práticos vivenciados pelo estagiário durante o período de estágio, desempenharam um papel crucial em seu desenvolvimento profissional na área de Desenvolvimento de Software. A combinação dessas duas vertentes proporcionou uma base sólida de conhecimento teórico, complementada pela aplicação prática desses conceitos em um ambiente real de trabalho. Essa abordagem integrada permitiu ao estagiário adquirir habilida-

des e competências relevantes, impulsionando seu crescimento profissional e sua capacidade de lidar com os desafios do desenvolvimento de software.

REFERÊNCIAS

AMAZON, A. O que é o .net? **https://aws.amazon.com/pt/what-is/net/**, 2023.

ARAUJO, E. C. de. Java e c.net - um breve e introdutório estudo comparativo de suas sintaxes e convenções. **http://www.linhadecodigo.com.br/artigo/1620/java-e-csharpnet-um-breve-e-introdutorio-estudo-comparativo-de-suas-sintaxes-e-convencoes.aspx**, 2023.

BEEKAI. Build complex and accessible forms. **https://www.react-hook-form.com/advanced-usage/**, 2023.

CASAGRANDE, E. O que é o scrum, quais os benefícios e como usar nas empresas. **https://pt.semrush.com/blog/scrum/**, 2022.

COLOMBO, G. Identificando vulnerabilidades client-side em seu projeto web. **https://medium.com/minuto-frontend/identificando-vulnerabilidades-client-side-em-seu-projeto-web-c12607ecc3b4**, 2018.

FORMIK, D. Validation schema. **https://formik.org/docs/guides/validation**, 2020.

GUEDES, M. O que é e como começar com c (c sharp)? **https://www.treinaweb.com.br/blog/o-que-e-e-como-comecar-com-c-sharp**, 2018.

HENRIQUE, J. Poo: o que é programação orientada a objetos? **https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos**, 2023.

JUNIOR. Fluxo scrum. **https://kodus.io/scrum-na-pratica-entendendo-o-fluxo/**, 2019.

KRIGER, B. O que é git? **https://kenzie.com.br/blog/o-que-e-git/**, 2022.

MELO, D. O que é typescript? [guia para iniciantes]. **https://tecnoblog.net/responde/o-que-e-typescript-guia-para-iniciantes/**, 2021.

MICROSOFT, C. O que é o azure devops? **https://learn.microsoft.com/pt-br/azure/devops/user-guide/what-is-azure-devops?view=azure-devops**, 2022.

MUI. Radio group. **https://mui.com/material-ui/react-radio-button/**, 2023.

NEVES, V. O que é react js? **https://www.alura.com.br/artigos/react-js**, 2023.

ORACLE. O que é tecnologia java e por que preciso dela? **https://www.java.com/pt-BR/download/help/whatis***java.html*, 2023.

PEREIRA, R. A. P. S. J. V. Guia de javascript: o que é e como aprender a linguagem mais popular do mundo? **https://www.alura.com.br/artigos/javascript**, 2023.

PPLWARE. Dbeaver: Uma excelente ferramenta para gerir bases de dados. **https://pplware.sapo.pt/software/dbeaver-nunca-tao-facil-gerir-bases-dados/**, 2017.

REACT, C. Using the effect hook. **https://legacy.reactjs.org/docs/hooks-effect.html**, 2023.

REDHAT. Api rest. **https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api**, 2020.

REDHAT. O que é api? **https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces**, 2023.

SILVA, J. C. B. da. React hooks: o que é e como funcionam? **https://www.alura.com.br/artigos/react-hooks**, 2023.

TEBET, I. Sistemas legados: como e por que introduzir técnicas modernas. **https://www.objective.com.br/insights/sistemas-legados/**, 2021.

TRINDADE, L. Desvendando a arquitetura hexagonal. **https://medium.com/tableless/desvendando-a-arquitetura-hexagonal-52c56f8824c**, 2019.

VELTRONI, L. E. P. Introdução a testes unitários com jest e testing library react.js. **https://dev.to/eduardoopv/introducao-a-testes-unitarios-com-jest-e-testing-library-reactjs-491n**, 2022.

VERSIANI, R. O que é o postman? **https://enotas.com.br/blog/postman/**, 2023.