LEONARDO JUNIO YURI DA SILVA BASSO

REFORMULAÇÃO DA PLATAFORMA CONNECT

LAVRAS – MG

2023

LEONARDO JUNIO YURI DA SILVA BASSO

REFORMULAÇÃO DA PLATAFORMA CONNECT

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

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG

2023

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Basso, Leonardo Junio Yuri da Silva
Reformulação da Plataforma Connect / Leonardo Junio Yuri da Silva Basso. – Lavras : UFLA, 2023.
42 p. : il.

Relatório de Estágio (graduação) – Universidade Federal de Lavras, 2023.
Orientador: Prof. Dr. Maurício Ronny de Almeida Souza.
Bibliografia.

1. Fullstack. 2. Desenvolvimento WEB. 3. Componentização de Software. 4. Testes de unidade. 5. Plataforma Cloud. 6. Plataforma de Planejamento e Programação Integrada. I. Souza, Maurício Ronny de Almeida. II. Título.

CDD-808.066

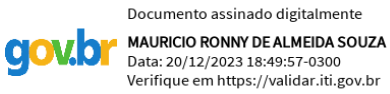
LEONARDO JUNIO YURI DA SILVA BASSO

REFORMULAÇÃO DA PLATAFORMA CONNECT

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

APROVADA em 08 de Dezembro de 2023.

Prof. Dr. Maurício Ronny de Almeida Souza	UFLA
Prof. Dr. Paulo Afonso Júnior	UFLA
Alexandre de Maia Júnior	dti



Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG
2023

Agradeço aos meus pais, Zilda e Francesco, que me deram a vida e me apoiaram em todos os momentos da minha trajetória acadêmica, dando incentivo e suporte em busca dos meus sonhos. Aos meus irmãos, Rosimara e George, que são minha zona de conforto e meus amigos, que me incentivaram e me ajudaram a superar os desafios. Aos meus colegas e professores, que compartilharam comigo o conhecimento e a experiência, que me fizeram crescer como pessoa e como profissional. A todos vocês, minha eterna gratidão.

AGRADECIMENTOS

Agradeço primeiramente a Deus por suas incontáveis obras maravilhosas operadas em minha vida e que sem Ele nada seria possível. A Ele toda honra e toda glória.

Agradeço à minha família, Zilda, Francesco, Rosimara e George, que convivem comigo diariamente e me deram todo suporte e incentivo para realizar meus sonhos.

Quero reconhecer o valor dos meus mestres, desde o ensino médio até o ensino superior, que me transmitiram o saber e me inspiraram a buscar o conhecimento.

Agradeço aos meus colegas de infância que cresceram junto comigo por sempre me mostrarem que a vida não é preto e branco e que sempre podemos aprender mais.

Aos meus amigos e exemplos da Comp Júnior, onde tive a minha primeira experiência profissional e me desenvolvi como pessoa e como profissional.

Quero agradecer aos meus companheiros do CIMVEST, que me apoiaram na construção de uma sociedade melhor por meio da educação financeira e me fizeram um líder melhor.

Meu obrigado aos colegas e mentores da dti, na qual com acolhimento, paciência e leveza me ajudaram a construir a base de minha carreira profissional.

Quero agradecer à Universidade Federal de Lavras, que me acolheu e me ofereceu oportunidades e estrutura para eu me formar e me aprimorar.

Por fim, a todas as pessoas próximas de mim na qual seria arriscado citar nomes e, inadvertidamente, omitir alguém. Vocês me proporcionaram sempre um ombro amigo, conforto e incetivo para superar os desafios que a vida apresenta. Obrigado por tornarem o fardo da caminhada mais leve.

"Stay Hungry, Stay Foolish"
(Steve Jobs)

RESUMO

Este documento tem como objetivo apresentar as atividades desenvolvidas durante estágio na empresa dti, focando especialmente nas experiências adquiridas na área de desenvolvimento web *fullstack*. Durante esse período, o estagiário contribuiu na reformulação da plataforma de gestão e visualização integrada Connect, uma ferramenta essencial para a visualização, manipulação de dados e tomada de decisões por parte dos clientes. Ao longo do estágio, foram aplicados conceitos adquiridos na graduação em Ciência da Computação e em atividades extra-curriculares, contribuindo de maneira significativa para a evolução da plataforma. Além de relatar o processo de desenvolvimento da Connect, este trabalho também aborda as experiências profissionais e pessoais vivenciadas durante esse período, enriquecendo a trajetória profissional de forma substancial.

Palavras-chave: Fullstack. Desenvolvimento WEB. Componentização de Software. Testes de unidade. Plataforma Cloud. Plataforma de Planejamento e Programação Integrada.

ABSTRACT

This document aims to present the activities developed during an internship at the company dti, focusing especially on the experiences acquired in the area of *fullstack* web development. During this period, the intern contributed to the reformulation of the Connect integrated management and visualization platform, an essential tool for visualization, data manipulation and decision making by clients. Throughout the internship, concepts acquired during the Computer Science degree and in extra-curricular activities were applied, contributing significantly to the evolution of the platform. In addition to reporting the development process of Connect, this work also addresses the professional and personal experiences experienced during this period, substantially enriching the professional trajectory.

Keywords: Full stack. Web development. Software Componentization. Unitary tests. Cloud Platform. Integrated Planning and Programming Platform.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este documento relata a experiência do estagiário na reformulação da plataforma Connect, um sistema desenvolvido pela empresa dti que atua como plataforma de visualização e tomada de decisão referente às regras de negócio sob a operação ponta a ponta de uma empresa que atua no ramo de mineração. O sistema faz parte da plataforma de planejamento integrado IBP (Integraded Business Planning) que abrange diversas áreas de negócio, como financeiro, logística, marketing, demanda e produção.

No entanto, o Connect tinha suas fundações em códigos legados, ou seja, códigos antigos que apresentavam dificuldades de manutenção e baixo desempenho devido ao uso de tecnologias obsoletas. Isso gerava problemas para a empresa contratante em relação ao desempenho e à integridade dos dados exibidos. Além disso, o usuário precisava utilizar várias plataformas diferentes para realizar as contribuições, a coleta, a gestão e visualização de dados, o que afetava a sua produtividade.

A reformulação da plataforma Connect teve como objetivo concentrar mais áreas de negócio na plataforma, aumentar o desempenho e trazer mais confiabilidade e integridade aos dados manipulados e apresentados, superando os problemas causados pelos códigos legados e apresentando uma nova interface de usuário, repaginada e mais veloz para auxilio nas tomadas de decisão do cliente.

O estagiário colaborou na reformulação da plataforma Connect no período compreendido entre agosto de 2022 e janeiro de 2023, totalizando 720 horas de contribuição. Neste documento, o estagiário apresenta e destaca os principais aspectos abordados durante o projeto, os desafios enfrentados, as contribuições técnicas ao projeto, bem como analisa como a aplicação de metodologias ágeis e a adoção de tecnologias modernas foram elementos chaves no êxito deste projeto.

1.1 Sobre a dti

A empresa dti é uma empresa de consultoria digital, fundada em 2009 em Belo Horizonte, Minas Gerais, e até o ano de 2023 conta com mais de 1200 colaboradores trabalhando de forma presencial e remota por todo o país. Com um portólio de clientes nacionais e internacionais, a dti se destaca no mercado de trabalho por sua organização e meticulosidade nas

metodologias ágeis, mas também a empresa se destaca com cultura acolhedora, ambiente de trabalho e colegas de trabalho sempre dispostos a ajudar ao próximo.

A estrutura organizacional da empresa segue uma abordagem inspirada no framework SAFe (*Scaled Agile Framework*), que é projetado para escalar práticas ágeis em organizações de grande porte. Nesse contexto, a empresa é fragmentada em diversas enterprises, cada uma delas formada por alianças, que, por sua vez, agrupam uma ou mais tribos. Essas tribos são unidades organizacionais que refletem a filosofia ágil, promovendo a autonomia e a agilidade (AGILE, 2023).

Dentro desse arranjo, as enterprises atuam como unidades autônomas, desfrutando de uma considerável independência em suas operações cotidianas. Cada enterprise pode conter uma ou mais tribos, e estas, por sua vez, são compostas por aproximadamente vinte a quarenta profissionais, como exibido na figura 1.1. Esta distribuição proporciona um ambiente propício para a colaboração eficaz e a entrega ágil de projetos.

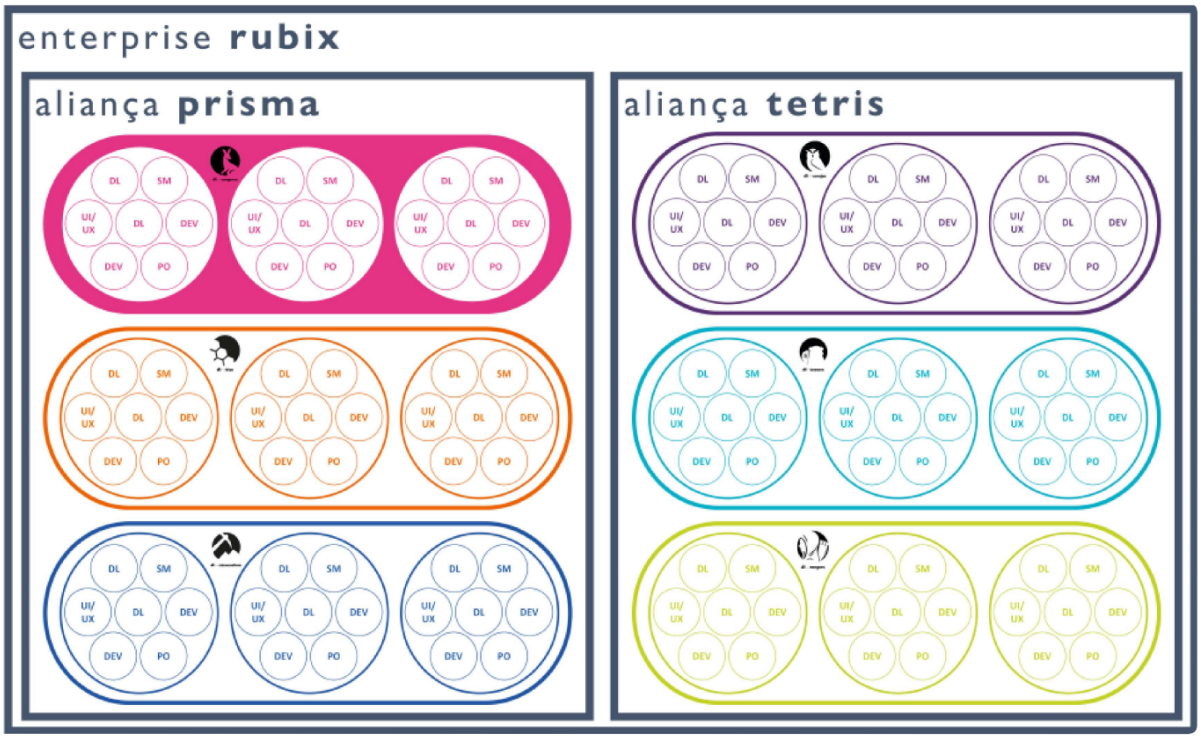
As tribos, por sua vez, subdividem-se em squads, pequenas equipes multifuncionais que operam de acordo com as diretrizes específicas estabelecidas para cada tribo. Essa estrutura modular e escalável permite a adaptação rápida às mudanças nas demandas do mercado e a entrega contínua de valor ao cliente.

Durante o período de estágio, o envolvimento do autor ocorreu na *enterprise* denominada "Prisma", mais precisamente na tribo Cangurus, composta por cerca de trinta membros e seus respectivos squads. Nesse contexto, os *squads* adotavam a metodologia *Team Topologies* [1], uma abordagem que visa organizar as equipes de forma eficaz, alinhando-se à arquitetura de sistemas (SKELTON; PAIS; MALAN, 2019). Destaca-se que, no âmbito dessa metodologia, foi implementado o conceito de "*Stream Aligned Teams*". Esse conceito implica que cada equipe (ou squad, no contexto da empresa) está alinhada com os fluxos de trabalho e a jornada final do cliente, garantindo assim a entrega de valor diretamente ao usuário final (SKELTON; PAIS; MALAN, 2019) .

Durante o projeto, o estagiário aplicou metodologias ágeis, como Scrum e Kanban, detalhadas com maior profundidade na Seção 2.2. Essas práticas ágeis proporcionaram um ambiente dinâmico e adaptável, contribuindo para a eficiência das operações e para a entrega contínua de valor ao cliente.

[1] https://teamtopologies.com/key-concepts

Figura 1.1 – Estrutura da enterprise



Fonte: Do autor (2023)

1.2 Organização do Trabalho

Ao longo do presente trabalho, será relatado as experiências e atividades desenvolvidas durante o período de estágio. Este documento está organizado em capítulos, sendo o capítulo 1 uma introdução sobre o escopo do projeto, os objetivos e a motivação da reformulação da plataforma Connect. Também é feita uma breve descrição da empresa dti, sua estrutura organizacional e sua abordagem de trabalho baseada em metodologias ágeis. O Capítulo 2 aborda os principais conceitos e tecnologias que foram utilizados na reformulação da plataforma Connect, tais como *React*, *TypeScript*, *StoryBook*, Testes Unitários, entre outros. O capítulo também explica como essas tecnologias se relacionam e contribuem para o desenvolvimento de uma plataforma moderna, performática e confiável. O Capítulo 3 descreve em detalhes a plataforma Connect, sua arquitetura, suas funcionalidades e seus benefícios para o cliente. O capítulo também relata os desafios enfrentados durante o processo de reformulação, as soluções adotadas e as atividades realizadas pelo estagiário em cada etapa do projeto, identificando e justificando o uso de tecnologias específicas para a construção do novo Connect. Por fim, o Capítulo 4 conclui o trabalho apresentando uma avaliação do projeto e de como as disciplinas aprendidas ao

longo do curso, aliadas às atividades extra-curriculares, foram importantes para a contribuição na plataforma, mas também para o crescimento pessoal e profissional do estagiário.

2 CONCEITOS E TECNOLOGIAS

A reformulação do Connect, alinhada com a integração de tecnologias modernas, foi resultado de uma sólida arquitetura de software cuidadosamente planejada para atingir escalabilidade, segurança e eficiência ao compararmos a versão posterior da plataforma, resultando em um patamar superior de desempenho e funcionalidades. Nesse sentido, a aplicação de tecnologias e linguagens de programação modernas não inclui apenas uma abordagem holística ao desenvolvimento web, como *frontend*, *backend* e *fullstack*, mas na incorporação estratégica de ferramentas de ponta. Principalmente o uso da biblioca React com Typescript proporcionou uma base sólida para a construção da plataforma com interface de usuário moderna e funcionais.

Além da demonstração da efetividade dos testes de software, a adoção e implementação da componentização foi essencial para a escalabilidade do projeto. Ao usar este conceito de desenvolvimento de software, não apenas facilitou a modularidade e a reutilização de código mas também teve um papel fundamental na agilidade do desenvolvimento. Outro aspecto relevante foi a arquitetura de software sólida e a priorização criteriosa das atividades a serem desenvolvidas, fatores que foram abordados com sucesso ao longo do projeto.

Por fim, é necessário destacar a importância das estratégias metodológicas, fornecendo uma visão abrangente do processo de reformulação do Connect. Uma das estratégias utilizadas foi a adoção do Scrum, uma metodologia ágil que visa a entrega rápida e contínua de valor para o cliente, com foco na colaboração, na adaptação e na melhoria contínua (SANDRINO, 2014). O Scrum permitiu que a equipe do projeto gerenciasse o processo de desenvolvimento de forma eficiente e eficaz, entregando um produto de alta qualidade, com funcionalidades relevantes e alinhadas com as demandas do mercado, em um curto espaço de tempo.

Nas seguintes Seções deste Capítulo, serão detalhados algumas das tecnologias e metodologias que desempenharam papéis cruciais ao longo de todo o ciclo de desenvolvimento do projeto.

2.1 Desenvolvimento Web

O Connect opera como uma aplicação web, o que implica que seu desenvolvimento envolve conceitos e práticas inerentes à área de desenvolvimento web. Esta esfera da programação é responsável por viabilizar a funcionalidade do site, adaptando-se aos requisitos específicos do

cliente e/ou usuário. No âmbito do desenvolvimento web, o foco recai principalmente na funcionalidade do site, abrangendo desde a codificação do código-fonte até a marcação textual que estrutura o conteúdo (ROUSE, 2020). A amplitude dessa área é ampla, incluindo desde a criação de páginas de texto simples até aplicativos web complexos, redes sociais e soluções para negócios eletrônicos.

A hierarquia do desenvolvimento web compreende a codificação do lado do cliente (*frontend*), a codificação do lado do servidor (*backend*) e a aplicação de tecnologias de banco de dados. Não obstante, o desenvolvedor web pode especificar nas áreas de desenvolvimento *frontend*, *backend* e *fullstack*.

O desenvolvimento *frontend* é responsável pela interface do usuário, concentrando-se na parte visual e interativa do aplicativo, onde elementos como layout, botões, gráficos e imagens são projetados para garantir uma experiência de usuário envolvente e intuitiva (FLANAGAN, 2002). O desenvolvimento *frontend* também abrange níveis mais aprofundados e complexos ao utilizar-se de requisições para o *backend*, estratégias de otimização e reúso de software para garantir um aplicação de qualidade. A maioria dos desenvolvedores web *frontend* usa linguagem de marcação de hipertexto (HTML), *Cascading Style Sheets* (CSS) e JavaScript para desenvolver sites (MOZILLA, 2023).

Ao desenvolvimento *backend* confere-se o cérebro por trás da aplicação, ou seja, o *backend* lida com as camadas de lógica de negócios e a manipulação segura de dados armazenados. Aqui, estrutura de dados complexas são gerenciadas, operações de banco de dados são executadas e a lógica de funcionamento do sistema é implementada. De acordo com uma pesquisa da W3Techs (W3TECHS, 2023b), os desenvolvedores *backend* usam linguagens como Java, PHP, C# e Python no âmbito de desenvolver aplicações web.

A harmonização entre o *frontend* e *backend* é essencial para assegurar que as solicitações do usuário sejam processadas de forma eficiente e que as informações recuperadas sejam apresentadas de forma concisa. A abordagem *fullstack*, na qual foi adotada durante o desenvolvimento do Connect, oferece uma visão mais abrangente do fluxo de funcionamento do sistema. Isso implica na atuação de desenvolvimento de código tanto nas camadas visíveis ao usuário quanto das engrenagens ocultas do *backend* (NORTHWOOD, 2018). Ao criar um projeto que faça uso devido da sinergia entre o *frontend* e o *backend*, é possível criar qualidade técnica

resultando em uma plataforma sólida e adaptável, capaz de atender as demandas variadas e dinâmicas das regras de negócio do cliente, como é o caso do novo Connect.

2.1.1 Tecnologias de Desenvolvimento

Ao autor e sua equipe durante o desenvolvimento da reformulação do Connect, foi dedicado a atuação no *frontend*. Tal atuação fez uso de tecnologias e ferramentas modernas como Javascript, React, Typescript, Storybook, testes unitários, uso do Sonarqube e da plataforma *cloud* Azure.

O JavaScript foi a linguagem de programação mais usada durante o desenvolvimento. Desde sua criação em 1995 até hoje (2023), o JavaScript é uma das linguagens presentes no coração da web e está presente em mais de noventa e oito por cento dos *websites* (W3TECHS, 2023a). Essa popularidade favoreceu o uso de bibliotecas e *frameworks* criados a partir da linguagem, como JQuery, React, Angular e o TypeScript. Essas ferramentas refletem o poder da linguagem no desenvolvimento web, sendo a linguagem mais usada atualmente pelos programadores ao redor do mundo (STACKOVERFLOW, 2023).

Uma das principais desvantagens do uso de Javascript é o o fato dela ser uma linguagem de programação com tipagem dinâmica [1]. Isso pode levar a erros de tipo que só são descobertos durante a execução do programa, o que pode ser problemático especialmente em projetos grandes e complexo como é o caso da plataforma Connect (TYPESCRIPT, 2023).

Diante dessa situação, foi utilizado a biblioteca de Javascript de *frontend* React aliado ao Typescript, melhorando a qualidade, segurança e manutenibilidade do código, proporcionando uma experiência de desenvolvimento mais eficiente.

O React é uma biblioteca grátis e de código aberto de JavaScript usada no *frontend* para criação de interfaces visuais baseada em componentes (META, 2023b). A biblioteca é mantida pela Meta (Facebook) e hoje é a escolha mais popular pelos desenvolvedores ao se tratar de tecnologias para *frontend* (STACKOVERFLOW, 2023).

Já o TypeScript é uma linguagem de programação de código aberto criada pela Microsoft (MICROSOFT, 2023c). Ela é um superconjunto sintático estrito de JavaScript que adiciona tipagem estática à linguagem. Com isso, o programador deve definir o tipo de cada variável e

[1] Tipagem dinâmica é uma característica de determinadas linguagens de programação, que não exigem declarações de tipos de dados, pois são capazes de escolher que tipo utilizar dinamicamente para cada variável, podendo alterá-lo durante a compilação ou a execução do programa (TRATT, 2009).

função. Isso facilita a prevenção de erros e bugs, pois o compilador do TypeScript verifica a consistência dos tipos e fornece *feedback* e soluções para o correto funcionamento do código.

A ênfase na qualidade do código foi sustentada por práticas de testes unitários que visam verificar individualmente cada componente do sistema, garantindo que funcionem conforme o esperado de forma isolada. O uso de testes unitários é uma das tendências contemporâneas no desenvolvimento web devido à sua capacidade de fortalecer a integração do código e agilizar o processo de identificar e corrigir possível falhas (MICROSOFT, 2023a). A utilização de bibliotecas como Jest e React-Testing-Library foram de grande valia no desenvolvimento do projeto. O Jest (META, 2023a) é uma estrutura de testes desenvolvida pelo Facebook (Meta) que oferece uma ampla gama de recursos, incluindo suporte integrado para asserções, simulação de funções e geração de relatórios detalhados. Enquanto isso, o React-Testing-Library (DODDS, 2023) é uma biblioteca que se concentra em testes que se aproximam da experiência do usuário, encorajando os desenvolvedores a escrever testes que se assemelham às interações do usuário com componentes React, garantindo uma maior confiabilidade e manutenibilidade dos testes.

Para o desenvolvimento das interfaces de usuário (UI) da plataforma, foi utilizado o Storybook (CHROMATIC, 2023). Essa ferramenta de *workshop* permite aos desenvolvedores criar e testar componentes de interface do usuário (UI). Ela fornece uma maneira fácil de isolar componentes, visualizar diferentes estados e realizar testes. O uso dessa ferramenta permitiu que a equipe desenvolvesse e testasse componentes de UI de forma isolada e interativa. Isso foi valioso pois o Connect consistia de várias telas com diversos componentes interdependentes, facilitando assim o reúso de código, documentação de desenvolvimento e principalmente a capacidade de testar o componente em um ambiente isolado antes dele ser integrado ao projeto principal.

A integração contínua com o uso da ferramenta SonarQube, foi uma poderosa ferramenta para avaliação da qualidade de código que se tornou uma referência para garantir padrões elevados (SONARSOURCE, 2023).

Todo o projeto do Connect foi desenvolvido utilizando-se do ecossistema Microsoft, integrado na plataforma de computação em nuvem líder no setor, a plataforma Azure (MICROSOFT, 2023b). Essa escolha proporcionou uma infraestrutura escalável e confiável para armazenar o projeto, fazer o uso de pipelines para subida de código nos ambientes de desenvolvimento, testes e apoio para ritos da metodologias ágeis como o uso de quadros Kanbam, linha

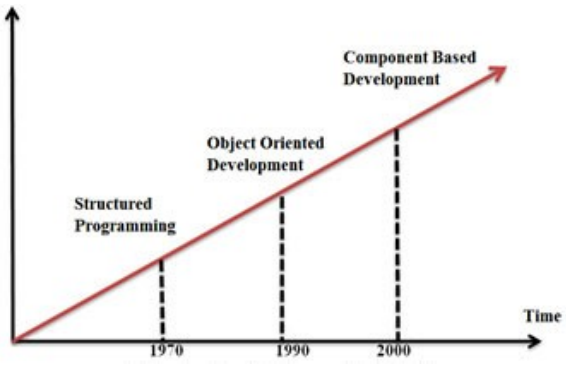
de *burndown* para acompanhamento da Sprint e *planning poker* das atividades futuramente desenvolvidas. Adicionamente, a exibição de métricas pelo Microsoft Azure, durante todo o desenvolvimento do projeto auxiliou na realização de testes unitários em todos as páginas e componentes, tanto no Connect em si como também nos componentes Storybook, mantendo cobertura de código maior que oitenta por cento em toda a plataforma.

2.1.2 Componentização de software

Desde A Crise do Software na década de 70 (DIJKSTRA, 1972) até os dias de hoje, o desenvolvimento de software passou por várias metodologias e modelos de abordagem como o modelo cascata até para uma abordagem orientada a componentes altamente gerenciáveis (GULIA; PALAK, 2023).

Inicialmente, os softwares eram construídos usando uma abordagem funcional, centrada nas funções e procedimentos. Essa abordagem é chamada de programação estruturada, e foi amplamente usada até a década de 1970. Com a introdução e popularização da programação orientada a objetos na década de 1990, houve uma mudança significativa na indústria de software, trazendo vantagens revolucionárias, como abstração, encapsulamento, polimorfismo e herança (GULIA; PALAK, 2023), como é possível observar na figura 2.1. A programação orientada a objetos usa objetos como unidades básicas de código, que possuem atributos (dados) e métodos (funções) relacionados. Essa abordagem permite o reuso, a modularidade e a organização de código, mas também pode introduzir problemas de desempenho, acoplamento e herança múltipla.

Figura 2.1 – Evolução de abordagens em desenvolvimento de software



Fonte: (GULIA; PALAK, 2023)

Entretanto, nem todos os problemas foram solucionados. Posteriormente, surgiu e começou a se popularizar o desenvolvimento de software baseada em componentes, concentrando-se na utilização de esforços anteriores para construir componentes (GULIA; PALAK, 2023). Cada componente representa um conjunto de serviços, e a combinação desses componentes forma o software completo. Essa abordagem é chamada de desenvolvimento baseado em componentes, e visa aumentar a interoperabilidade, a qualidade e a manutenção do software, mas também requer uma boa arquitetura, documentação e gerenciamento de componentes. A figura 2.1 mostra a evolução do desenvolvimento de software ao longo do tempo, desde a programação estruturada até o desenvolvimento baseado em componentes, indicando uma tendência crescente ou evolução dessa abordagem.

A evolução do ciclo de vida de desenvolvimento de software demonstra a transição dos profissionais para a modularidade e a construção de software mais coeso e escalável como foi o caso da abordagem de desenvolvimento utilizada no Connect. Além de fazer uso de prática de modularização de código, divisão em funções, organização da estrutura do projeto de maneira hierárquica e modular e blocos de código com funcionalidades específicas e não repetitivas, o foco em componentização, especialmente aliadas a linguagem React trouxe benefícios ao desenvolvimento do projeto. O uso da ferramenta de *workshop* [2] Storybook permitiu o isolamento, criação e testes de componentes facilitando todo o fluxo de desenvolvimento no *frontend* e adicionando ganhos na produtividade e eficiência tanto da empresa tanto ao cliente.

2.2 Metodologias Ágeis

Nesta Seção, será detalhado como a empresa dti utiliza metodologias ágeis para o desenvolvimento de seus projetos, destacando a metodologia "dti flow", que é uma inovação dentro do paradigma ágil. A incorporação dessas abordagens visam garantir maior personalização, flexibilidade, colaboração eficiente e entrega iterativa, elementos cruciais para enfrentar os desafios dinâmicos do cenário atual de desenvolvimento de software.

Destacando-se entre as metodologias ágeis mais reconhecidas, o Scrum permanece como um pilar fundamental na gestão de projetos complexos. O Scrum é uma das metodologias ágeis mais reconhecidas e utilizadas no mundo, com uma taxa de adoção global de sessenta e dois

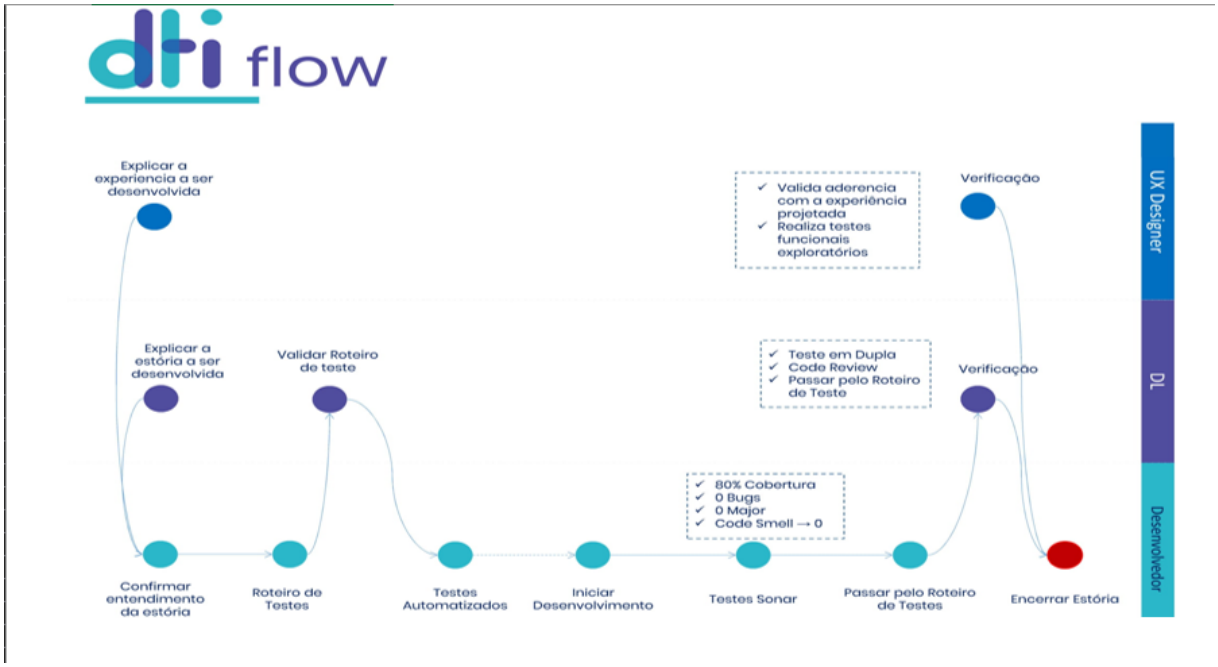
[2] Ferramentas de *workshop* são ferramentas que ajudam os desenvolvedores a colaborar e compartilhar ideias. Elas podem ser usadas para organizar reuniões, documentar projetos e gerenciar tarefas.

por cento em empresas de tecnologias, segundo a pesquisa “State of Agile Report 2023” da VersionOne (DIGITAL.AI, 2023). O Scrum é caracterizado por seus *sprints* curtos e reuniões diárias de acompanhamento, que permitem uma resposta rápida às mudanças nos requisitos do projeto. O Scrum também define papéis, artefatos e eventos que facilitam a gestão e a entrega de projetos complexos (ALLIANCE, 2001).

Outra metodologia ágil que se relaciona com o Scrum é o Kanban, que se baseia na visualização do fluxo de trabalho, usando um quadro com colunas que representam os diferentes estágios do processo, como “a fazer”, “em andamento” e “feito”. Cada tarefa é representada por um cartão que se move ao longo das colunas conforme o seu progresso. O Kanban também limita o número de tarefas que podem estar em cada coluna, para evitar o acúmulo de trabalho e garantir a qualidade. O Kanban permite uma maior transparência, eficiência e melhoria contínua no desenvolvimento de projetos complexos, pois facilita a identificação de gargalos, prioridades e dependências (ATLASSIAN, 2023).

No âmbito específico da empresa, é essencial destacar a metodologia ágil específica adotada. O “Manifesto Ágil” destaca a importância dos indivíduos e interações, entrega contínua de software funcional, colaboração com o cliente e resposta flexível às mudanças (ALLIANCE, 2001).

Figura 2.2 – Fluxo de desenvolvimento do dti flow



Fonte: Do autor (2023)

A estratégia da dti em adotar o dti Flow revela um compromisso com a inovação dentro do paradigma ágil. Essa metodologia, ao agregar-se às práticas existentes, busca proporcionar uma abordagem mais ajustada às necessidades específicas da empresa e de seus projetos. O dti flow (figura 2.2) aborda os processos relacionados ao desenvolvimento de código durante as sprints pela equipe de desenvolvimento, sendo responsáveis principalmente os desenvolvedores, o desenvolvedor líder e o UX Designer da equipe. O dti flow consiste em cinco estágios, que são:

- Explorar e experienciar o ser desenvolvido: Neste estágio, o desenvolvedor compreende o que deve ser desenvolvido a partir da atividade escrita disponível por meio da plataforma Azure. O desenvolvedor também explora as possíveis soluções, ferramentas e tecnologias que podem ser utilizadas para o desenvolvimento.

- Validar Requisito do teste: Nesta etapa, o desenvolvedor valida o seu roteiro do teste, que consiste em uma planilha na qual é inserido os caminhos possíveis para fazer o teste de validação.

- Testes e Desenvolvimento: O desenvolvedor realiza o desenvolvimento do código, seguindo as boas práticas de programação e os padrões de qualidade definidos pela empresa. O desenvolvedor também realiza os testes unitários, de integração e de interface, para garantir que o código atende aos requisitos e aos critérios de aceitação.

- Confirmar entrega da atividade: Neste estágio, o desenvolvedor confirma a entrega da atividade, verificando se o código está de acordo com o esperado pelo cliente e pelo desenvolvedor líder. O desenvolvedor também realiza a documentação, a revisão e o versionamento do código, para facilitar a manutenção e a evolução do projeto.

- Encerrar Estória: Por fim, o desenvolvedor retira a atividade do fluxo, indicando que ela está concluída e pronta para ser entregue ao cliente. O desenvolvedor também atualiza o status da atividade no quadro de gestão do projeto, para manter a transparência e a comunicação com a equipe.

A estratégia da dti em adotar o dti Flow revela um compromisso com a inovação dentro do paradigma ágil. Portanto, podemos concluir que a empresa utiliza metodologias ágeis de forma

estratégica e inovadora, ao adotar o dti flow como uma adição aos métodos convencionais, como o Scrum e o Kanban. O dti flow proporciona uma abordagem mais personalizada, flexível, colaborativa e iterativa, que se adapta às necessidades específicas da empresa e de seus projetos. Por fim, é possível concluir que o uso ferramentas de metodologia ágil de forma otimizada não são apenas uma abordagem teórica, mas sim uma prática incorporada na cultura de desenvolvimento utilizada no Connect.

3 REFORMULAÇÃO DA PLATAFORMA CONNECT

A Plataforma IBP (*Integrated Business Platform*) é uma solução que centraliza as informações oficiais do cliente, reduz o trabalho manual de consolidação de dados e aumenta o tempo disponível para análises de negócio, por meio de uma visão integrada do plano de produção, cenários analíticos e gestão de riscos e oportunidades em todos os pontos da cadeia produtiva do cliente. A plataforma permite que o negócio gere planos/resultados mais assertivos e ganhe mais confiança nas informações que estão sendo trabalhadas. Por meio da captura simples e prática dos dados, a plataforma ainda oferece visões flexíveis e personalizadas às diferentes cadeias de produção do Adquirente.

A Plataforma suporta o gerenciamento das rotinas operacionais e a gestão estratégica de performance do planejamento da cadeia de suprimentos, isto é, a gestão do fluxo de bens e serviços que passa pelo transporte e armazenagem, estoque e embarque, aquisição de matéria-prima, fornecimento do produto, até a chegada ao consumidor final. Isso só é possível pela ampla gama de serviços e áreas de negócio que a plataforma IBP mantida ao cliente contempla. Essa plataforma integrada é adotada pela empresa contratante e abrange diversas áreas de negócios, como financeiro, logística, marketing, vendas, demanda e produção.

Dentro do ecossistema IBP, o Connect é uma plataforma de gestão e visualização integrada que faz parte da solução oferecida ao cliente. O Connect assume o papel de uma plataforma abrangente para visualização e tomada de decisões relacionadas às regras de negócio da empresa contratante. Sua atuação abrange a operação ponta a ponta na área de mineração, permitindo a análise de contribuições específicas relacionadas à mineração na plataforma. Essa ferramenta possibilita que os usuários criem ambientes de trabalho personalizados, adaptados às suas necessidades e perfis profissionais específicos.

Neste capítulo, será descrito o processo de reformulação da plataforma Connect, na qual melhorou consideravelmente a estrutura de código, atualizou as tecnologias, modernizou a aparência e trouxe novas funcionalidades chaves.As atividades de desenvolvimento foram exercidas durante o mês de Agosto de 2022 até Janeiro de 2023 pelo estagiário na empresa dti. Durante este período, o estagiário atuou majoritariamente desenvolvendo telas do sistema com o uso do *framework* React e TypeScript para Javascript, desenvolvimento e uso de componentes isolados com o *workshop* de *frontend* Storybook, testes unitários com Jest e participando de reuniões com a empresa contratante e ritos referentes à metodologia ágil.

3.1 Sobre o Connect

O Connect assume o papel de uma plataforma abrangente para visualização e tomada de decisões relacionadas às regras de negócio da empresa contratante. Sua atuação abrange a operação ponta a ponta na área de mineração, permitindo a análise de contribuições específicas relacionadas à mineração na plataforma. Essa ferramenta possibilita que os usuários criem ambientes de trabalho personalizados, adaptados às suas necessidades e perfis profissionais específicos. Dentro desses ambientes personalizados, os usuários têm a capacidade de consolidar informações provenientes de diversas fontes de dados. Essa integração eficiente e a apresentação visualmente acessível dos dados beneficiam significativamente gestores e analistas, facilitando a visualização e a tomada de decisões informadas.

A partir dele, é possível que os usuários criem ambientes de trabalho personalizados, adaptados aos seus perfis de trabalho específicos. Dentro desses ambientes, os usuários têm a capacidade de agregar informações de várias fontes de dados. Adicionando *widgets*[1] específicos para cada uma das áreas de trabalho, gráficos relacionados à manipulação dos dados recebidos por meio de um *plugin* para o Microsoft Excel, na qual o usuário poderia contribuir com dados referentes às regras de negócio por meio de planilhas. Também é possível adicionar na área de trabalho itens como atalhos para equipes, *email* ou *websites* de uso recorrente pelo usuário.

Esses ambientes de área de trabalho personalizados pelo usuário poderiam ser compartilhados com outros usuários da plataforma, tornando assim o compartilhamento de informações e visualização do fluxo de trabalho e tomada de decisão mais acessível.

Na figura 3.1 é possível ver uma área de trabalhado no antigo Connect, com vários *widgets* selecionados pelo usuário. Também é possível visualizar o menu lateral, a possibilidade de troca de idioma, visualização de notificações e acesso às configurações de conta no menu ao canto superior direito. Um pouco abaixo existe o submenu na qual o usuário pode adicionar mais itens, como outros *widgets*, duplicar a área de trabalho, excluir ou compartilhá-la com outros usuários.

O Connect é uma plataforma que visa facilitar o acesso, a integração e a apresentação dos dados de forma visualmente acessível e interativa. Ela beneficia gestores e analistas que precisam tomar decisões informadas sobre o planejamento da cadeia de suprimentos do cliente.

[1] *Widgets* no contexto do Connect são micro-frontend que funcionam de maneira independente e podem ser adicionados nas áreas de trabalho do Connect

Figura 3.1 – Área de trabalho do antigo Connect

Fonte: Do autor (2023)

No entanto, o Connect apresentava algumas limitações e problemas que comprometiam sua usabilidade e funcionalidade na qual resultou em uma reformulação do sistema.

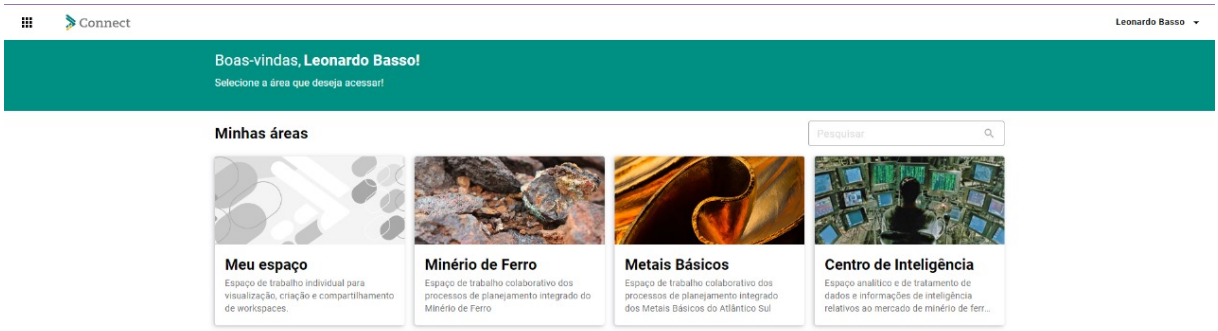
A necessidade de reformulação da plataforma veio de um desafio da empresa contratante que enfrentava desafios significativos em relação à coleta e gestão de dados em diferentes áreas, como contribuições para minérios, criação de equipes e validação de contribuições. Essas tarefas exigiam acessar diversos projetos diferentes na qual estavam construídos sobre uma API legado, antiga e muitas vezes lenta. Isso resultava em uma incompatibilidade na visualização desses dados pelo Connect na qual muitas vezes trazia dados incoerentes, dificultando assim a confiabilidade da plataforma. Também, o antigo design da interface de usuário usado pela plataforma era pouco claro ao cliente, na qual o mesmo precisava recorrer ao suporte. Esses problemas e as soluções propostas serão discutidos na próxima Seção (Seção 3.2).

3.2 Novo Connect

Através das novas funcionalidades, o Connect atualizado proporciona um fluxo de trabalho mais dinâmico. Após o *login*, o usuário é imediatamente redirecionado para a página inicial do novo Connect.

Nesta área, existe um Card também intitulado ’Meu Espaço’, conforme a figura 3.2, que levará a uma página com o mesmo nome. Caso o usuário faça parte de equipes de negócios, outros *cards* associados a diferentes áreas de negócios da plataforma serão exibidos para seleção.

Figura 3.2 – Página inicial do novo Connect



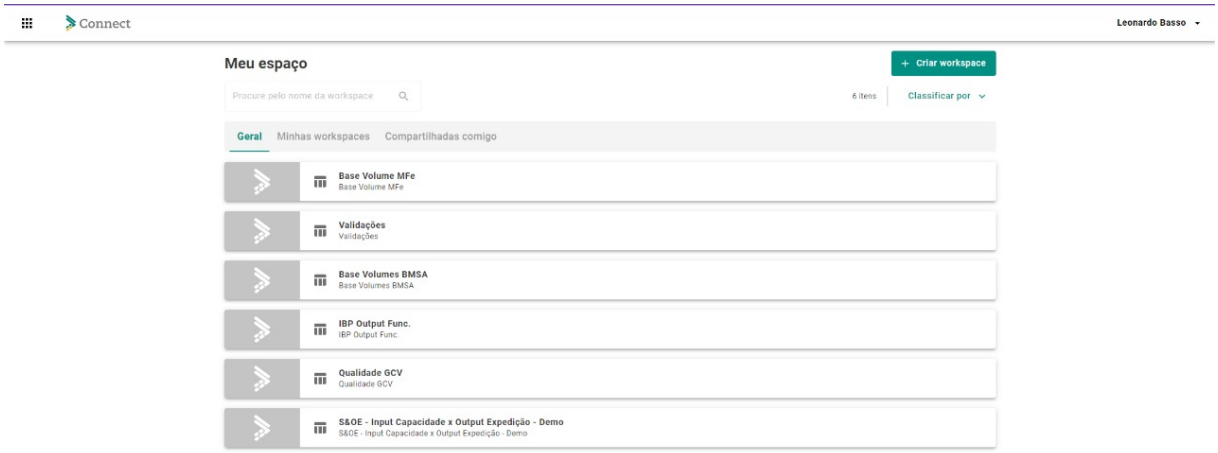
Fonte: Do autor (2023)

A figura 3.3 exibe a página ’Meu Espaço’, na qual o usuário poderá visualizar várias *workspaces*. Cada uma dessas *workspaces* é uma área de trabalho virtual na qual é possível adicionar itens personalizáveis, como *widgets* e gráficos, entre outros. É possível compartilhar essas *workspaces*, e, nesta área, também é possível visualizar as *workspaces* compartilhadas com o usuário logado.

Os *cards* da plataforma estão relacionados a áreas de negócios específicas, como ’Minério de Ferro’, ’Metais Básicos’ e outras. Ao clicar em um desses *cards* da plataforma, é possível acessar a área de negócios correspondente, onde também é possível adicionar *widgets* relacionados à plataforma e visualizar e compartilhar *workspaces* com os usuários que fazem parte da equipe dessa plataforma."

Outra funcionalidade na qual foi desenvolvida do zero para o novo Connect é o "Catálogo de Itens", que consiste em um menu de lista com os diversos itens que podem ser adici-

Figura 3.3 – Área "Meu Espaço"do novo Connect



Fonte: Do autor (2023)

onados em cada uma das *workspaces*. Cada um desses itens representam um *microfrontend*[2], que podem ser *widgets*, *charts* (gráficos) e até *cards* de atalhos para outras páginas web criado pelo usuário relacionados à área de negócio.

Todas essas novas funcionalidades agregadas compõe o novo Connect e estão desenvolvidas sob o ecosistema Microsoft. Ao estar incorporado no plataforma cloud Microsoft Azure, o Connect se beneficia de uma infraestrutura robusta e altamente escalável, proporcionando um ambiente confiável para o gerenciamento de fluxos de trabalho dinâmicos. Isso é especialmente útil uma vez em que o Connect necessita de recursos como integração contínua, análise de métricas e automação de implantação. A partir do Microsoft Azure, também foi usado automação dos pipelines entre os ambientes de desenvolvimento - *Develop* (desenvolvimento), UAT (*User Acceptance Testing*) e PROD (produção), automatizando compilação, teste, implantação, métricas de testes e *deploy* nestes ambientes.

Não obstante, a plataforma Azure foi especialmente importante no trabalho de engenharia de software do projeto, facilitando os ritos de metodologias ágeis pertencentes à este *framework* Scrum mas também aos ritos específicos da empresa dti.

[2] Um microfrontend é uma forma de arquitetar uma aplicação web em módulos ou funções independentes, que podem ser desenvolvidos, testados e implantados por equipes diferentes, usando tecnologias diferentes, mas que se integram para formar um produto coeso (GEERS, 2023)

Nas próximas seções será detalhado as etapas de desenvolvimento das seguintes novas funcionalidades: desenvolvimento da área "Meu Espaço"(Seção 3.2.1), desenvolvimento dos componentes de Card usados na área "Meu Espaço"(seção 3.2.2), uso do *workshop* de desenvolvimento de componentes no Storybook (Seção 3.2.3), a importância de testes unitários para o sucesso do projeto (seção 3.2.4) e o uso de Sonar para avaliar e agir ao uso de cobertura de código, *code smells*, entre outros (seção 3.2.5).

3.2.1 Desenvolvimento da página “Meu Espaço”

O desenvolvimento da página “Meu Espaço” no Connect utilizou diversas tecnologias e componentes para criar uma interface funcional e amigável. Essa página permite que os clientes, após efetuarem login e clicarem no Card “Meu Espaço”, acessem rapidamente todas as *workspaces* das quais participam, facilitando o uso das funcionalidades mais frequentes na plataforma (figura 3.3). O objetivo dessa inovação foi proporcionar aos usuários uma experiência mais prática e eficiente ao acessarem o sistema. As principais tecnologias utilizadas foram React e StoryBook.

O React foi escolhido principalmente pela sua capacidade de manipular lógica e estado. O React também fornece *hooks*, que são funções que permitem que os componentes usem o estado e outros recursos sem escrever uma classe, proporcionando simplicidade, desempenho e flexibilidade ao desenvolvimento. O StoryBook foi usado para fornecer um conjunto de componentes que seguem as diretrizes da material UI. A partir do StoryBook, foram usados componentes como botões, guias, campos de texto e cartões que podem ser facilmente personalizados e integrados ao React.

A página “Meu Espaço” é composta por um cabeçalho, uma caixa de pesquisa, uma barra de abas e uma lista de *workspaces*. O cabeçalho exibe o título da página e um botão para criar ou duplicar um *workspace*. A caixa de pesquisa permite ao usuário filtrar os *workspaces* por um termo de pesquisa. A barra de abas permite ao usuário alternar entre diferentes categorias de *workspaces*: geral, favoritos, compartilhados e arquivados. A lista de *workspaces* exibe os *workspaces* que correspondem aos filtros, usando um componente de Card para cada *workspace*.

O componente Card é um dos componentes mais importantes e complexos da página. Ele é baseado no componente Card do StoryBook, que é um componente que exibe informações

e ações para um item de catálogo. O componente Card exibe o nome, a descrição, a imagem e as ações de cada *workspace*. As ações incluem abrir, editar, duplicar, favoritar, compartilhar e arquivar o *workspace*. O componente Card também lida com a lógica e o estado de cada ação, como atualizar os dados do *workspace*, mostrar uma caixa de diálogo de confirmação ou chamar uma API.

O desenvolvimento da página "Meu Espaço" enfrentou algumas dificuldades e desafios, como buscar e exibir os *workspaces* de maneira confiável e com bom desempenho. Isso exigiu o uso de técnicas de paginação, cache e tratamento de erros. Lidar com as interações do usuário e mudanças de estado para os filtros, as abas e as ações exigiu o uso de *hooks*, *callbacks* e variáveis de estado.

O desenvolvimento da página "Meu Espaço" também implementou algumas otimizações e melhores práticas, como o uso de *hooks useMemo* e *useCallback* para memorizar valores e funções e evitar nova renderização e recálculo desnecessários. O uso do TypeScript para adicionar anotações de tipo e verificar erros e bugs em tempo de compilação. E por fim, o uso de ferramentas de formatação e *linting* de código, como Prettier e ESLint, para garantir um estilo de código consistente e limpo.

A página "Meu Espaço" é resultado da utilização de diversas tecnologias e componentes para criar uma interface funcional e amigável. A página permite ao usuário visualizar, filtrar e gerenciar seus *workspaces*, usando um componente de cartão para cada *workspace*. O desenvolvimento envolveu superar algumas dificuldades e aplicar algumas otimizações, para garantir uma aplicação robusta e de alta qualidade. Com seu foco na usabilidade e na personalização, essa página se destaca como um elemento fundamental para melhorar a entrega de valor aos clientes da plataforma.

3.2.2 Desenvolvimento do componente Card usado na área "Meu Espaço"

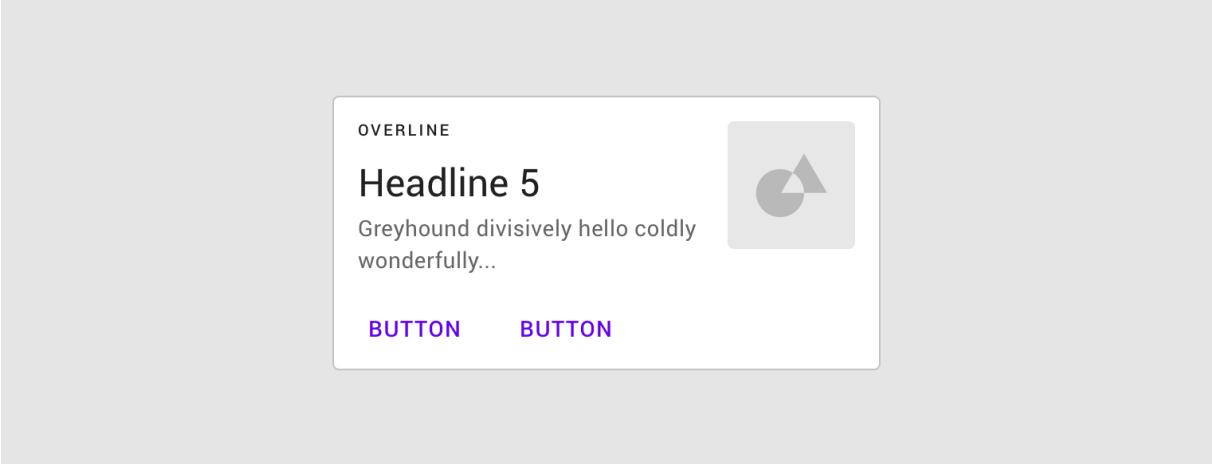
A importância do desenvolvimento de componentização em projetos grandes e complexos como o novo Connect, é fundamental para a organização, manutenção e escalabilidade do código. Primeiramente, a abordagem de componentização permite dividir o aplicativo em partes menores, conhecidas como componentes, cada uma responsável por uma funcionalidade específica. Isso facilita o desenvolvimento colaborativo, já que diferentes equipes ou desenvolvedores podem trabalhar em componentes isolados, reduzindo conflitos de código e melhorando

a eficiência. Além disso, a reutilização de componentes é uma vantagem significativa, pois economiza tempo e recursos, uma vez que funcionalidades similares podem ser implantadas em todo o projeto.

Um dos principais componentes usados pelo Connect na qual o estagiário participou do desenvolvimento foi o componente de Card usado pelas páginas de "Meu Espaço"e também de áreas como plataformas.

Uma vez que este componente seria utilizado em muitas páginas mas também para facilitar o reúso, modificação e teste, o componente Card foi desenvolvido no StoryBook (seção 3.2.3).

Figura 3.4 – Exemplo de um componente Card



Fonte: (Material Design, 2023)

O Card consistia em um retângulo responsivo desenvolvido em React com TypeScript e utilizando uma biblioteca de CSS chamada "Styled Components", na qual foi essencial para a personalização do componente. Para guiar o desenvolvimento do componente, o Card veio da biblioteca de componentes de interface de usuário para React chamada de Material UI (MUI, 2023), que seguia as diretrizes do popular Material Design desenvolvido e mantido pela Google. É possível visualizar um Card de exemplo na figura 3.4.

Para desenvolver o Card, foi criado uma pasta dentro do projeto do Storybook na qual era composto pelo arquivo de desenvolvimento, arquivo de desenvolvimento do stories, arquivo css e o arquivo para os testes unitários do componente individual.

O código mostrado na figura 3.5 é um exemplo de como criar o componente Card usando o Storybook. O código define um componente React chamado Card que recebe um título, ícones

e conteúdo como propriedades. O componente usa uma classe de estilo para definir a aparência do cartão e renderiza o título, os ícones e o conteúdo dentro de um elemento div.

Figura 3.5 – Código do desenvolvimento do Card no Storybook

```
16 > export interface CardProps {···
40   }
41
42   const Card = (props: CardProps) => {
43     const { title, icons, date, children, className } = props;
44
45     const classes = useStyles();
46     return (
47       <Paper className={clsx(classes.cardContainer, className)} elevation={3}>
48         <div className={classes.headerCard}>
49           {title}
50           <div className={classes.flexDiv}>
51             {icons?.map((icon) => (
52               <IconButton
53                 onClick={icon.clickAction}
54                 className={classes[icon.type]}
55                 key={icon.name}
56                 color="main"
57                 size="default"
58                 variant="text"
59                 shape="circular"
60               >
61                 {icon.icon}
62               </IconButton>
63             ))}
64           </div>
65         </div>
66         <div className={classes.changeDate}>{formatDate(date)}</div>
67         {children}
68       </Paper>
69     );
70   };
71
72   export { Card };
73
```

Fonte: Do autor (2023)

Após a criação do Card, o Storybook possibilita a elaboração do arquivo de histórias (também conhecido como stories). Esse arquivo desempenha um papel importante ao permitir a dinamicidade e personalização proporcionadas pela ferramenta Storybook (conforme detalhado na Seção 3.2.3). Embora o arquivo de histórias seja relativamente simples, sua visualização na ferramenta é de extrema importância para todas as partes envolvidas no projeto. Ele serve como um recurso fundamental para a manipulação e planejamento do design, bem como para a adoção das melhores práticas conforme o projeto evolui em direção à sua forma final.

Outro código essencial é o de estilização do componente. O CSS desempenha um papel singular no Storybook, pois, para desenvolver o código, é preciso criar todas as possibilidades que serão exibidas e personalizadas na visualização no Storybook. Isso inclui diferentes cores, tamanhos, fontes e como o componente se adapta a variáveis como textos, ícones e imagens.

Por fim, o arquivo de teste é um elemento fundamental e benéfico para o ciclo de desenvolvimento do projeto, conforme detalhado na seção 3.2.4. Os testes unitários foram concebidos para abranger uma variedade de casos de uso, garantindo o correto funcionamento do componente. Na imagem a seguir, é possível observar dois casos: o primeiro diz respeito à exibição correta das propriedades, como botão, título e data. O segundo caso é um teste de exceção, no qual um mock[3] é utilizado para verificar se o componente reage de forma apropriada.

O código mostrado na figura 3.6 é um exemplo de como testar o componente Card usando o Storybook e a biblioteca React Testing Library. O código define um teste que verifica se o componente Card renderiza corretamente o título, o subtítulo, o texto e o botão do cartão.

Figura 3.6 – Casos de testes de unidade do componente Card no Storybook

```
23
24  describe("<Card />", () => {
25    it("should have icons, title and date", () => {
26      render(
27        <Card icons={icons} title="Douglas Pereira" date="2021-06-14 16:21" />,
28      );
29
30      const buttonElement = screen.getAllByRole("button");
31      const cardTitleElement = screen.getByText("Douglas Pereira");
32      const cardDateElement = screen.getByText(formatDate("2021-06-14 16:21"));
33      expect(cardTitleElement).toBeInTheDocument();
34      expect(cardDateElement).toBeInTheDocument();
35      expect(buttonElement).toHaveLength(2);
36    });
37
38    it("shouldnt have icons, should have title, a invalid date and a input children", () => {
39      render(
40        <Card title="Titulo" date="data invalida">
41          <input type="text" placeholder="Card Children" />
42          <input type="text" placeholder="Card Children" />
43        </Card>,
44      );
45
46      const inputElement = screen.getAllByRole("textbox").length;
47      const buttonElement = screen.queryByRole("button");
48      const cardTitleElement = screen.getByText("Titulo");
49      const cardDateElement = screen.getByText("-");
50      expect(cardTitleElement).toBeInTheDocument();
51      expect(cardDateElement).toBeInTheDocument();
52      expect(buttonElement).not.toBeInTheDocument();
53      expect(inputElement).toBe(2);
54    });
55  });
56
```

Fonte: Do autor (2023)

[3] Um mock é um objeto falso que simula o comportamento de um objeto real para fins de teste. Um mock permite testar uma unidade de código de forma isolada, sem depender de outras partes do sistema ou de recursos externos.

3.2.3 Desenvolvimento do componente Card no StoryBook

O desenvolvimento de um componente de Card no Storybook oferece inúmeras vantagens para o projeto Connect. O Storybook é uma ferramenta de desenvolvimento que permite criar, visualizar e testar componentes de forma isolada do aplicativo principal. Para começar, o estagiário envolvido no desenvolvimento do componente Card cria um novo componente no Storybook, o que envolve a definição das propriedades, estilos e comportamentos desejados. Isso permite que o componente seja desenvolvido e refinado separadamente do projeto principal, garantindo uma abordagem mais controlada e iterativa, como mostrado na figura 3.7.

Além disso, o Storybook facilita a documentação e a visualização dos componentes. Com anotações, exemplos e histórias, é possível descrever em detalhes como o componente deve ser usado e quais casos de uso específicos ele aborda. Isso é extremamente útil para os desenvolvedores que irão integrar o componente em várias partes do projeto Connect, pois eles têm acesso a uma documentação clara e exemplos prontos para uso.

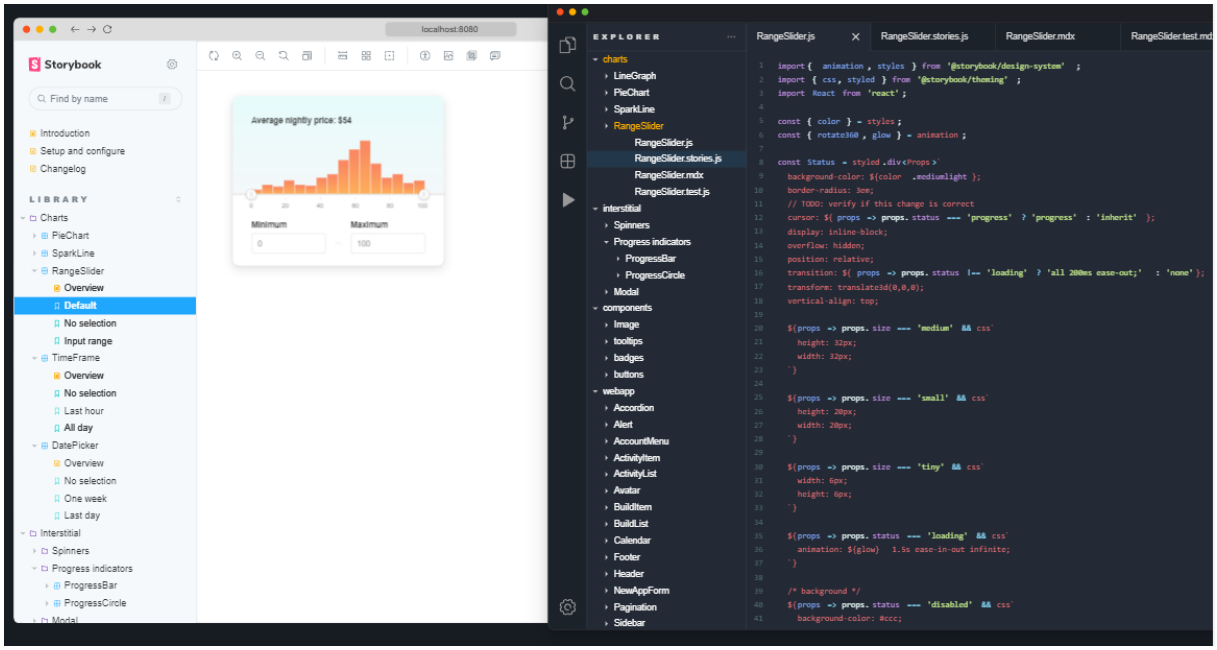
Para utilizar o componente Card no projeto principal, a equipe simplesmente o importa como um módulo externo. Graças à componentização, o código do componente Card é desacoplado do código do projeto principal, o que torna a integração simples e segura. Os desenvolvedores podem passar as propriedades necessárias para personalizar o componente Card de acordo com as necessidades de cada página. Como o componente foi testado no Storybook, a equipe tem confiança de que ele funcionará conforme o esperado, economizando tempo e minimizando erros de implementação.

O desenvolvimento de componentes no Storybook foi uma abordagem eficaz para criar e reutilizar componentes, como o Card e outros inúmeros componentes desenvolvidos utilizando Material UI no Connect. Este processo simplificou o desenvolvimento, tornando a documentação mais acessível e melhorando a manutenção do código, permitindo uma integração suave desses componentes no projeto principal.

3.2.4 Desenvolvimento de testes de unidade

Os testes de unidade desempenharam um papel fundamental no desenvolvimento do novo Connect.

Figura 3.7 – Exemplo do fluxo de desenvolvimento no StoryBook



Fonte: (CHROMATIC, 2023)

Uma vez em que o Connect antigo os esforços eram direcionados para a correção, manutenção e atuação nos muitos *bugs* que surgiam na plataforma, durante o planejamento e desenvolvimento da reformulação, colocamos uma meta alta de pelo menos oitenta por cento de cobertura de código e casos cobertos pelos testes unitários, garantindo que cada componente comportasse corretamente, tanto nos componentes criados no Storybook como nos componentes e páginas individuais do Connect.

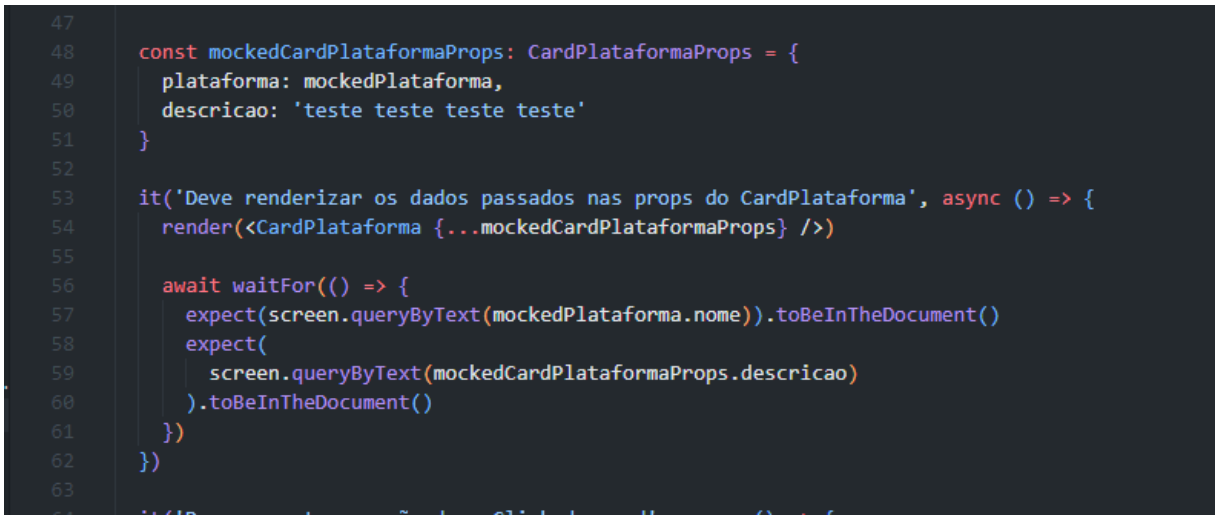
A criação dos testes unitários foram desenvolvidas usando duas bibliotecas conhecidas e eficientes - Jest e React-testing-library. A implementação ocorria geralmente envolvendo a criação de "casos de teste"que especificam o comportamento esperado de um componente React.

Por exemplo, um teste pode verificar se um componente de botão dispara uma função de clique ou se um formulário realiza uma validação de entrada corretamente. Os testes podem ser executados automaticamente sempre que houver uma alteração no código, o que ajuda a identificar regressões e problemas de forma rápida e eficaz.

Um exemplo usado durante o desenvolvimento do projeto com a biblioteca react-testing-library foi o teste de unidade para o componente de Card que levava para a página de plataforma. No teste, foi desenvolvido um mock com as informações esperadas para serem renderizadas

quando forem passadas por prop para o Card. Após isto, foi criado um *await*[4] para o Card ser renderizado e verificado se o Card estava sendo exibido corretamente e com as informações corretas.

Figura 3.8 – Caso teste de exibição do Card plataforma



Fonte: Do autor (2023)

O uso de testes unitários, principalmente no *frontend*, são numerosos. Eles aumentam a confiabilidade de código, identificam problemas mais cedo no ciclo de desenvolvimento, economizando tempo e recursos no longo prazo. Além disso, fornecem documentação viva do comportamento do código, o que facilita a colaboração entre desenvolvedores e o entendimento de terceiros sobre o sistema. Também promovem a manutenção contínua e a evolução de um aplicativo com maior segurança, já que as mudanças podem ser feitas com a segurança de que os testes identificarão problemas potenciais.

No entanto, é importante notar que os testes unitários não residem apenas em sua capacidade de detecção de erros mas também incentivam uma escrita de código mais limpo e modular, uma vez que componentes bem testados tendem a ser mais desacoplados e focados em uma única responsabilidade.

Tudo isso contribui para a construção de uma base sólita no desenvolvimento de aplicações complexas e robustas como o novo Connect.

[4] O *await* em JavaScript é um operador que faz com que uma função assíncrona espere pela resolução de uma Promise antes de continuar sua execução. Ele permite escrever código assíncrono de forma mais simples e legível

3.2.5 Análise de qualidade de código pelo Sonar

O uso da ferramenta SonarQube (também chamada de Sonar) foi de extrema importância para o fluxo de desenvolvimento da aplicação. O Sonar é uma ferramenta amplamente reconhecida no mundo do desenvolvimento de software para garantir a qualidade de código, identificar problemas e vulnerabilidades e melhorar a manutenibilidade do projeto. Embora a ferramenta seja comumente associada ao uso no *backend*, ela desempenha um papel vital no desenvolvimento *frontend*, sendo usada para analisar códigos Javascript, Typescript e suas bibliotecas e *frameworks*.

A principal vantagem do uso, como exibido na figura 3.9 é a capacidade da ferramenta de identificar erros, problemas de segurança, duplicações, *bugs*, cobertura de testes e outros aspectos de qualidade de código como *code smells* (características do código que indicam a presença de problemas de design ou possível área de melhoria, como por exemplo uma função muito longa ou uma classe com muitas responsabilidades).

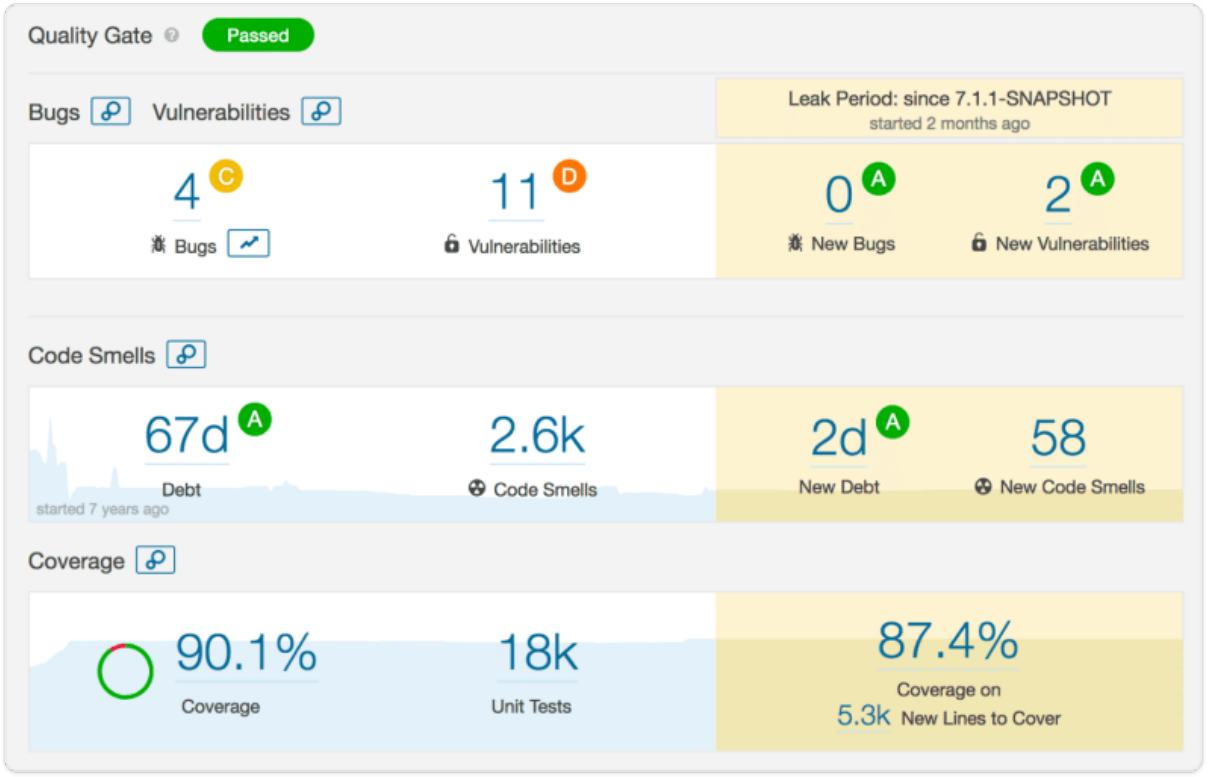
A implementação do SonarQube foi realizada pelo time de arquitetura da tribo na qual o estagiário trabalhou, sendo configurado e adicionado na etapa de análise do pipeline de CI/CD (integração contínua e entrega contínua) na plataforma Azure na qual era executado a cada *pull request*[5] aceita para o projeto.

A partir do link gerado para a visualização pelo pipeline, era possível acessar a página do Sonar no navegador e ter acesso à todas as informações citadas anteriormente e também comparativos dos novos códigos adicionados e também comparativos com outras *branchs*[6]. Também era possível executar o sonar localmente a partir da integração da ferramenta com os *webpacks*[7] utilizados no projeto.

O uso do SonarQube, em conjunto com as tecnologias empregadas e o comprometimento dos desenvolvedores, permitiu que o novo Connect fosse lançado sem a presença de *bugs* recorrentes. Além disso, a combinação de uma cobertura de testes unitários superior a oitenta por cento e a quase inexistência de vulnerabilidades de segurança e code smells em

[5] Uma pull request é um pedido para mesclar alterações de código com o repositório principal do projeto.
[6] Uma branch é uma ramificação de um repositório de código-fonte, que permite criar uma cópia do código e modificá-lo de forma independente da versão principal.
[7] Webpacks são ferramentas que permitem compilar módulos JavaScript (arquivos, imagens, fontes, JS, CSS, HTML, etc.) para uso em um navegador

Figura 3.9 – Exemplo da página principal do SonarQube



Fonte: Do autor (2023)

cada commit[8] para a branch principal do projeto demonstra a importância de realizar um sólido levantamento arquitetural. Estabelecer metas e adotar metodologias eficazes desde o início do desenvolvimento de um novo projeto se revelou essencial para garantir o sucesso e a integridade do produto.

3.3 Considerações Finais

O Connect proporcionou uma ampla gama de experiências, tecnologias, aprendizados e desenvolvimento de habilidades técnicas e inter-pessoais, devido à sua natureza como um projeto grande e complexo, envolvendo uma das maiores empresas do Brasil. Isso também se deve à cultura da empresa, que enfoca a colaboração, a eficiência na metodologia ágil e as oportunidades de crescimento.

O autor adquiriu um conhecimento profundo sobre o fluxo de engenharia de software em um projeto, desde a concepção da arquitetura até as entregas contínuas e validações com o

[8] Um commit é uma ação de registrar as alterações feitas em um repositório de código-fonte, como o Git.

cliente. Essas interações permitiram uma aproximação do autor ao perfil de um profissional em forma de "T", termo utilizado para descrever indivíduos que possuem habilidades especializadas em uma área específica, representadas pela parte vertical do "T", e habilidades mais amplas em diversas áreas relacionadas, representadas pela parte horizontal do "T". Esses profissionais são cada vez mais demandados pelo mercado devido à capacidade de liderar projetos, solucionar problemas complexos e contribuir para o sucesso geral de uma organização.

No entanto, em um mercado cada vez mais especializado, a especialização se torna imperativa para enfrentar desafios maiores. O projeto trouxe clareza a esse respeito, destacando a importância de concentrar-se no aprendizado de um punhado de tecnologias específicas e aprofundar o conhecimento nelas. Além disso, manter-se atualizado, como a profissão exige, não é mais uma escolha, mas uma necessidade para aqueles que buscam alcançar o sucesso.

3.3.1 Principais desafios do Projeto

A atuação na reformulação de uma plataforma grande e complexa como o Connect é uma tarefa complexa e repleta de desafios. Durante o desenvolvimento do projeto, o autor encontrou diversos desafios como a criação de um projeto do zero em contraste com a atuação em manuntenção de códigos legados do projeto passado. Temas como o aprendizado de novas tecnologias, compreensão e aprofundamento de regras de negócio do cliente e a criação de testes unitários eficazes com alta cobertura de código foram alguns dos obstáculos superados durante o tempo em que o estagiário trabalhou no novo Connect.

Um dos desafios primordiais na qual o estagiário encontrou foi o primeiro contato com a tecnologia React e Typescript. Embora experiências anteriores em empresa júnior e também em estudos em disciplinas durante a graduação, a modernez e complexidade do React representaram um ponto de partida desafiador na busca por entender os conceitos fundamentais da tecnologia. A ótima liderança do time de desenvolvimento combinado com colegas de trabalho proativos e dispostos a ajudar, facilitaram a busca pela proficiência necessária para contribuir efetivamente para o projeto.

Compreender as regras do cliente também não se revelou uma tarefa fácil. Uma plataforma de planejamento e programação integrada lida com processos complexos e específicos do setor. Fornecer soluções que atendessem às necessidades do cliente exigia uma compreensão profunda dessas regras de negócio. A colaboração próxima com os stakeholders e a equipe de

desenvolvimento era fundamental para esclarecer requisitos e garantir que as soluções propostas estivessem alinhadas com as expectativas do cliente.

A criação de testes unitários eficazes era um dos desafios técnicos mais críticos. Garantir que o código fosse robusto, seguro e livre de erros requeria testes abrangentes. No entanto, a elaboração de testes de unidade de alta qualidade exigia um profundo conhecimento do código, bem como a capacidade de identificar os casos de teste relevantes. A cobertura de código era um objetivo essencial, mas alcançá-la sem comprometer a eficiência e a simplicidade dos testes era um desafio constante. O trabalho em equipe se mostrou essencial neste aspecto do desenvolvimento ao compartilhar opiniões e casos de uso para o sistema na qual não poderiam ter sido pensadas tão rapidamente de maneira individual.

A realização de testes e validações contínuos era essencial para garantir a qualidade do projeto. A necessidade de feedback rápido e a identificação precoce de problemas eram cruciais. A implementação de um fluxo de trabalho de integração contínua e a criação de processos de revisão eram desafios importantes para manter a qualidade e a consistência ao longo do desenvolvimento.

A reformulação de uma plataforma é um empreendimento desafiador, mas também uma oportunidade de aprendizagem e crescimento. Três desafios críticos foram enfrentados pela equipe de desenvolvimento: colaboração interdisciplinar, adaptação a mudanças contínuas e gestão eficiente do tempo.

A colaboração interdisciplinar foi essencial para garantir que as soluções propostas estivessem alinhadas com as necessidades do cliente e com os padrões de usabilidade. A equipe trabalhou em estreita colaboração com outros departamentos, incluindo os stakeholders do cliente, designers de UX/UI e especialistas em regras de negócio. A comunicação constante e a troca de conhecimentos foram fundamentais para o sucesso do projeto.

A adaptação a mudanças contínuas foi outro desafio crítico. À medida que novas informações e feedbacks eram incorporados ao projeto, a equipe precisava ser ágil e flexível para ajustar as estratégias e a implementação. A prática de gerenciamento de mudanças eficiente se mostrou vital para assegurar que o projeto permanecesse alinhado com as expectativas em constante evolução.

A gestão eficiente do tempo foi fundamental para o sucesso do projeto e do desenvolvimento pessoal do estagiário. O estagiário aprendeu a equilibrar prazos e priorizar tarefas, habilidades essenciais para qualquer profissional.

No final, apesar dos desafios significativos e embora o estagiário não estivesse mais trabalhando na empresa para ver os resultados e métricas do projeto, a equipe conseguiu enfrentar os obstáculos e entregar com sucesso a reformulação do Connect. A combinação de aprendizado contínuo, colaboração interdisciplinar, gerenciamento eficaz de mudanças, documentação adequada e foco na motivação da equipe foi fundamental para alcançar esse resultado.

4 CONCLUSÃO

O desenvolvimento do novo Connect, apresentado neste trabalho de conclusão de curso, reflete na convergência de diversos conhecimentos adquidos ao longo da formação no curso de Ciência da Computação pela Universidade Federal de Lavras. A aplicação prática de conceitos e técnicas aprendidas durante as disciplinas, além das atividades extra-curriculares desempenhadas pelo autor, foram de importância crucial nos alicerces do início da vida profissional na área de desenvolvimento de software do estagiário.

O uso dos conceitos aprendidos em disciplinas como Engenharia de Software, Estrutura de Dados, Banco de Dados, Sistemas Distribuídos, Processos de Software, entre outras, foram importantes no processo do desenvolvimento da interdisciplinaridade do autor, algo importante em um mundo atual que cada vez os profissionais que vão além do seu escopo de trabalho se destacam.

O entendimento profundo das necessidades do cliente não envolveu apenas aspectos técnicos mas também uma compreensão holística do contexto de negócios. A capacidade de traduzir requisitos complexos em soluções técnicas e eficazes é uma habilidade que foi cultivada ao longo de disciplinas de análise de algoritmos e resolução de problemas computacionais.

Este relatório de estágio não é apenas um reflexo do aprendizado do estagiário e da demonstração de conceitos aprendidos na faculdade ao mundo real, mas também atesta a capacidade transformadora na qual a tecnologia se mostra como ponto de partida.

O poder transformador da tecnologia se mostra presente no projeto de desenvolvimento do novo Connect, trazendo otimização e eficiência operacional ao cliente, gerando valor significativo e impulsionando a transformação digital das empresas atendidas.

Por fim, a bagagem de experiência adquirida durante o curso com os desafios vencidos, nas amizades e projetos desenvolvidos na empresa júnior, o caráter empreendedor, a interdisciplinaridade e a liderança obtidas no núcleo de estudos, bem como a torcida, o apoio e a ajuda recebidos durante essa fase, foram de extrema importância para que o estagiário seguisse em seu objetivo de aprender mais sobre o mundo afim de torná-lo melhor.

REFERÊNCIAS

AGILE, I. S. **Scaled Agile Framework (SAFe) for Lean Enterprises**. 2023. Disponível em: <https://scaledagileframework.com/about/>.

ALLIANCE, A. **Manifesto for Agile Software Development**. Agile Alliance, 2001. Disponível em: <http://agilemanifesto.org/>.

ATLASSIAN. **Kanban**: How the kanban methodology applies to software development. Atlassian, 2023. Disponível em: <https://www.atlassian.com/agile/kanban/>.

CHROMATIC. **Build UIs without the grunt work**. Chromatic, 2023. Disponível em: <https://storybook.js.org/>.

DIGITAL.AI. **16th State of Agile Report**. Digital.ai, 2023. Disponível em: <https://digital.ai/resource-center/analyst-reports/state-of-agile-report/>.

DIJKSTRA, E. W. The humble programmer. **Communications of the ACM**, v. 15, p. 859–866, 1972. Disponível em: <https://cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF>.

DODDS, K. C. **React Testing Library**. KentC.Dodds, 2023. Disponível em: <https://testing-library.com/docs/react-testing-library/intro/>.

FLANAGAN, D. **JavaScript**: The definitive guide. O'Reilly, 2002. (Definitive Guides). ISBN 9780596000486. Disponível em: <https://books.google.com.br/books?id=vJGlu9t9LNYC>.

GEERS, M. **Micro Frontends**: extending the microservice idea to frontend development. 2023. Disponível em: <https://micro-frontends.org/>.

GULIA, P.; PALAK. Component based software development life cycle models: A comparative review. **Orient. J. Comp. Sci. and Technol**, v. 10, 2023. Disponível em: <http://www.computerscijournal.org/?p=5982>.

Material Design. **Cards**: Cards contain content and actions about a single subject. 2023. Disponível em: <https://m2.material.io/components/cards>.

META. **Jest é um poderoso Framework de Testes em JavaScript com um foco na simplicidade**. Meta Platforms, 2023. Disponível em: <https://jestjs.io/pt-BR/>.

META. **The library for web and native user interfaces**. Meta, 2023. Disponível em: <https://react.dev/>.

MICROSOFT. **Melhores práticas de teste de unidade com .NET Core e .NET Standard**. Microsoft, 2023. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/core/testing/unit-testing-best-practices>.

MICROSOFT. **O que é o Azure?** Microsoft, 2023. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure/>.

MICROSOFT. **TypeScript is JavaScript with syntax for types**. Microsoft, 2023. Disponível em: <https://www.typescriptlang.org/>.

MOZILLA. **The web and web standards**: learn web development: Mdn. MDN Web Docs, 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards>.

MUI. **Material UI SAS**: The react component library you always wanted. 2023. Disponível em: <https://mui.com>.

NORTHWOOD, C. **The Full Stack Developer**: Your essential guide to the everyday skills expected of a modern full stack web developer. Apress, 2018. ISBN 9781484241523. Disponível em: <https://books.google.com.br/books?id=vvd6DwAAQBAJ>.

ROUSE, M. **What is web development?**: Definition from techopedia. 2020. Disponível em: <https://www.techopedia.com/definition/23889/web-development>.

SANDRINO, K. M. **Análise da Aplicação de Metodologias Ágeis**: Um estudo de caso em empresa de desenvolvimento de software. 59 p. Monografia (Trabalho de Conclusão de Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) — Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2014. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/16848/3/PG_COADS_2013_2_05.pdf>.

SKELTON, M.; PAIS, M.; MALAN, R. **Team Topologies**: Organizing business and technology teams for fast flow. IT Revolution, 2019. (Business book summary). ISBN 9781942788812. Disponível em: <https://books.google.com.br/books?id=oFdRuAEACAAJ>.

SONARSOURCE. **SonarQube**. SonarSource S.A, 2023. Disponível em: <https://docs.sonarsource.com/sonarqube>.

STACKOVERFLOW. **Most popular technologies**: 2023 developer survey. stackoverflow, 2023. Disponível em: <https://survey.stackoverflow.co/2023/#technology-most-popular-technologies>.

TRATT, L. Dynamically typed languages. **Advances in Computers**, v. 77, p. 149–184, jul. 2009.

TYPESCRIPT. **What Problems Can TypeScript Solve**. TypeScript, 2023. Disponível em: <https://www.typescriptlang.org/why-create-typescript>.

W3TECHS. **Usage statistics of JavaScript as client-side programming language on websites**. W3Techs, 2023. Disponível em: <https://w3techs.com/technologies/details/cp-javascript>.

W3TECHS. **Usage statistics of server-side programming languages for websites**. W3Techs, 2023. Disponível em: <https://w3techs.com/technologies/overview/programming_language>.