Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de Software

Autor: Gabriel Alves Hussein
Orientadora: Profª. Drª Milene Serrano

Brasília, DF
2024

Gabriel Alves Hussein

DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientadora: Prof[a]. Dr[a] Milene Serrano

Brasília, DF

2024

Gabriel Alves Hussein

DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de Software/ Gabriel Alves Hussein. – Brasília, DF, 2024-

121 p. : il. (algumas color.) ; 30 cm.

Orientadora: Profª. Drª Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2024.

1. DASboard. 2. Documentação. I. Profª. Drª Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de Software

CDU 02:141:005.6

Gabriel Alves Hussein

DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de Software

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 9 de julho de 2024:

Profª. Drª Milene Serrano
Orientadora

Prof. Dr. Maurício Serrano
Examinador 1

Prof. Me. Rafael Fazzolino P. Barbosa
Examinador 2

Brasília, DF
2024

Agradecimentos

Meus agradecimentos vão, primeiramente e principalmente, à minha família, que sempre me apoiou durante minha trajetória, e que apesar dos problemas, sempre estão presentes e providenciaram uma educação de qualidade, além de uma boa criação.

Gostaria ainda de agradecer a todos os colegas, conhecidos e amigos, que me ajudaram de diferentes formas ao longo do processo de graduação na Universidade de Brasília, incluindo durante a escrita desta monografia.

Por fim, gostaria de agradecer à professora orientadora Milene Serrano por todo o apoio, não apenas durante a escrita do trabalho, mas também ensinamentos muito valiosos durante os anos que fui aluno e monitor de aulas ministradas por ela.

Resumo

A presença de Sistemas Legados em empresas antigas é muito comum, e sua presença é geralmente associada a problemas, sejam eles relacionados à incompleta ou falta de documentação, sejam por problemas de códigos muito extensos, nos quais não há constantes cuidados em relação a métricas de código e padrões arquiteturais. Essa realidade não desejada resulta em sistemas muito grandes, os quais apresentam impedimentos nas atividades diárias dos desenvolvedores/mantenedores desses sistemas. Esse trabalho procura mitigar tais problemas, conferindo uma abordagem combinada de soluções. Essa abordagem incorpora facilitadores no que compreende a documentação de Sistemas Legados, além de prover recursos visuais (*Dashboard*) visando o acompanhamento dessas documentações, incluindo a identificação de possíveis não conformidades que demandam atenção da equipe. No intuito de realizar um trabalho de natureza aplicada, com objetivos exploratórios, foi utilizado um Estudo de Caso. Nesse caso, trata-se de uma empresa de médio porte (cerca de 2.000 funcionários), com grande variedade de sistemas em seu portfólio (Sistemas Legados e sistemas mais atuais). Por fim, cabe ressaltar que, dentre os vários focos de atenção que poderiam ser conferidos aos Sistemas Legados, o trabalho atuou, predominantemente, na área de Arquitetura e Desenho de Software. Essa decisão foi tomada usando como base a relevância de uma arquitetura de software para a compeensão, a manutenção, e a evolução de qualquer sistema. Sendo assim, em resumo, a intenção foi acompanhamento facilitado de Sistemas Legados da empresa. Tem-se como solução um *dashboard* (semiautomatizado), capaz de apresentar a documentação de Sistemas Legado, orientando-se pelo Documento de Arquitetura (DAS), além de métricas de código específicas, em aderência à cultura da empresa.

Palavras-chave: Sistemas Legados. Arquitetura e Desenho de *Software*. *Dashboard*. Métricas de Análise de Código. Estudo de Caso.

Abstract

The presence of Legacy Systems in older companies is very common, your presence is usually associated to problems, be those related to incomplete or lack of documentation, or problems related to bloated codes, where there isn't enough care being put on the code metrics or architectural patterns. This unwanted reality results in large systems that present roadblocks on the day to day activities of the developers/mantainers in charge of those systems. This study employs a mitigation to those problems, with a mix of solutions. This approach incorporates facilitators of what encompass Legacy System documentation, while also providing visual resources (Dashboard) aiming to help keeping track of those documentations, including the identification of possible non-conformities that would demand the attention of the team. With the goal of creating a study of applied nature, with exploratory objectives, it was conducted a Case Study. In this case, it is about a medium-sized company (around 2.000 employees), with a big variety of systems in their portfolio (Legacy Systems and more modern systems). Lastly, it is worth noting that, among the various points that could be tackled about Legacy Systems, the approach acted, predominantly, on the Software Architecture field of expertise. This decision was taken using as basis the relevancy of software architeture for the comprehension, maintenance, and the evolution of a system. That way, in short, the intention was the facilized tracking of a company Legacy Systems. The result of this study is a Dashboard.

Key-words: Legacy Systems. Software Architecture. Dashboard. Code Metrics Analysis. Case Study.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API *Application Programming Interface*

BPMN *Business Process Model and Notation*

CLI *Command-Line Interface*

DAS Documento de Arquitetura de *Software*

JSON *JavaScript Object Notation*

JVM *Java Virtual Machine*

MVC *Model View Controller*

NNG *Nielsen Norman Group*

SQL *Structured Query Language*

TCC Trabalho de Conclusão de Curso

UI *User Interface*

UX *User Experience*

Sumário

1 Introdução

Nesse capítulo, é conferida uma breve Contextualização, no intuito de deixar mais claro sobre o domínio no qual o trabalho está inserido, bem como para apresentar o problema tratado no mesmo. Nesse sentido, é acordada uma definição para Sistemas Legados, uma vez que se procura contribuir para um acompanhamento mais adequado desse tipo de sistema. Sendo assim, o domínio de interesse são Sistemas Legados. Portanto, foi desenvolvido um *Dashboard*, através do qual uma equipe, alocada em uma empresa, tem maior facilidade para acompanhar um ou mais sistemas legados. Comumente, tais sistemas carecem de documentação, e incorrem em problemas de manutenção e dificuldades de evolução e integração com outras soluções computacionais. Na sequência, têm-se a Justificativa para a realização do trabalho; as Questões de Pesquisa e Desenvolvimento, que nortearam o trabalho, e os Objetivos atingidos no mesmo. Por fim, há uma breve noção quanto à Metodologia adotada no trabalho, e a apresentação da Organização da Monografia em capítulos.

1.1 Contextualização

Sistemas Legados são atuantes em muitas organizações, sejam empresas, instituições, dentre outros cenários de relevância (FRITOLA; SANTANDER, 2022). Entretanto, cabe ressaltar que Sistemas Legados possuem fragilidades em termos de documentação e código (FRITOLA; SANTANDER, 2022). Portanto, os autores acordam que esses sistemas, apesar de valiosos às organizações, dispõem de manutenção de alto custo, e impossibilidades de evolução e/ou integração com sistemas mais atuais, além de possuírem códigos não estruturados, dentre outras dificuldades para as equipes que lidam com os mesmos.

Corroborando para maior compreensão do perfil desses Sistemas Legados, Few (2006) revela que o desenvolvimento desse tipo de *software* baseia-se em grandes blocos de código, compondo uma arquitetura tipicamente monolítica. Sendo assim, sem a preocupação de isolar o acesso aos dados e à lógica de negócio, da camada de apresentação ao usuário. Nesse sentido, ainda com base nos autores, tem-se que uma arquitetura inadequada gera diversos problemas tecnológicos, seja de gestão ou integração, devido ao alto acoplamento e à baixa coesão.

Apesar dos problemas, os Sistemas Legados podem ser indispensáveis para uma dada organização (FRITOLA; SANTANDER, 2022). Sendo assim, essas empresas dependem de equipes especializadas, visando lidar com esses sistemas. Essa tarefa não é fácil, tampouco acessível economicamente (ANNETT, 2019), conforme ocorre na empresa que

foi utilizada como Estudo de Casos no decorrer deste trabalho, e cujo perfil será descrito mais adiante.

Como o intuito desse trabalho foi contribuir para o domínio dos Sistemas Legados, há necessidade de expor sobre a importância de uma documentação para um *software*. De acordo com Annett (2019), a documentação permite manter registros de como utilizar as funcionalidades desse *software*. Entretanto, esse mesmo autor comenta sobre a notória falta de documentação nos Sistemas Legados, uma vez que são aplicações muito antigas, em tecnologias obsoletas, comprometendo a rotina de negócios das empresas.

Diante do exposto, com esse trabalho, foi desenvolvida uma solução capaz de mitigar tais dificuldades, facilitando, dentre outras contribuições: o acompanhamento desses sistemas, e até mesmo a compreensão de como os mesmos podem trabalhar de forma integrada/colaborativa com outros sistemas mais modernos da empresa. Para que fosse conferida uma visão mais aplicada da solução, foi realizado um Estudo de Caso em uma empresa, na qual o autor desse trabalho atua como desenvolvedor. Doravante, e no intuito de manter oculta a identidade da empresa, a mesma será chamada Empresa A.

O perfil pleno da empresa ainda foi estudado e será apresentado com mais critério ao longo desse trabalho. Entretanto, pode-se mencionar que se trata de uma empresa de médio porte, envolvendo um quadro de 2.000 funcionários, e especializada no desenvolvimento de *software* no domínio bancário. Ressalta-se que há consentimento da empresa, em concordância com os estudos realizados nesse trabalho. A Empresa A possui um portfólio híbrido, com Sistemas Legados e sistemas mais atuais. No total, a equipe em que o autor trabalha conta com quatro Sistemas Legados, os quais incorrem em desperdício de tempo e recursos, conforme vivenciado pelo autor em sua atuação como desenvolvedor, no período compreendido de 13 de Janeiro de 2020 até o momento de elaboração dessa monografia.

Perante o cenário descrito sobre a Empresa A, e a inerente necessidade de uma medida de contorno para mitigar os impedimentos na rotina de negócios da empresa, o trabalho compreendeu o desenvolvimento de um *dashboard*, no intuito de facilitar o acompanhamento dos Sistemas Legados, e melhorar o entendimento sobre os mesmos. Ao se ter maior compreensão sobre o funcionamento desses sistemas, há possibilidade de maior integração desses sistemas com outros sistemas mais modernos da Empresa A. Por se tratar de um *dashboard*, ocorreu necessidade de estudo sobre quais dados considerar, e quais recursos utilizar para mais bem apresentá-los às equipes técnicas.

1.2 Justificativa

Conforme colocado anteriormente, a Empresa A possui Sistemas Legados que foram desenvolvidos há muitos anos, e que não possuem documentação adequada. Essa

realidade traz impedimentos para os novos funcionários, os quais apresentam dificuldades na compreensão dessas aplicações, atrasando a ambientação desses funcionários às rotinas de negócio, e incorrendo na necessidade de retrabalho em demandas de manutenção e evolução dessas funcionalidades. Adicionalmente, há perdas de tempo, desperdícios de recursos, e a imposição de cargas de trabalho mais exaustivas. Como pontos observados, além da falta de documentação, há contratempos para integração com novas tecnologias e diferentes sistemas da própria empresa à dificuldade de entender o funcionamento dos sistemas. Tais evidências foram percebidas pelo autor desse trabalho, e reforçadas pelo levantamento realizado junto aos funcionários da empresa, e disponível no Capítulo 5, Estudo de Caso.

Com que o desenvolvimento do *dashboard*, permitindo acompanhamento dos Sistemas Legados de forma visual, e baseado nas informações de um Documento de Arquitetura, além da apresentação de métricas de análise estática, tem-se uma solução pertinente para o problema contextualizado. Entretanto, tornou-se necessária uma análise mais adequada sobre a aceitabilidade dessa solução na Empresa A. Essa análise também foi objeto de estudo desse trabalho.

Cabe colocar que, na literatura especializada (JONES, 2015), é comum a indicação da migração de Sistemas Legados para soluções mais modernas. Entretanto, essa pode ser uma estratégia inválida para algumas empresas, como é o caso da Empresa A. Os Sistemas Legados da Empresa A são críticos, e fundamentais para se manter as rotinas de negócio da empresa. Tal particularidade demandou uma solução alternativa, que possibilitasse prolongar o uso dos Sistemas Legados de forma facilitada.

1.3 Questões de Pesquisa e Desenvolvimento

O presente trabalho compreendeu viés aplicado, demandando atividades de pesquisa e desenvolvimento. Portanto, ao final do trabalho, procurou-se responder os seguintes questionamentos:

Questão de Pesquisa: Quais documentações, com base na literatura especializada, são relevantes para facilitar o acompanhamento e a compreensão de Sistemas Legados?

Diante de estudos realizados, e dada a relevância da área de Arquitetura de *Software* para o pleno conhecimento de um sistema (FOWLER, 2019), acreditou-se que documentos que concentram diferentes visões arquiteturais do sistema, ou seja, o Documento de Arquitetura (DAS), representa um artefato considerado como ponto de partida nessa linha de pesquisa, junto à literatura, sobre documentações de Sistemas Legados. Demais informações sobre o DAS e as definições de Sistemas Legados serão conferidas no Capítulo 2 - Referencial Teórico.

Questão de Desenvolvimento: Como prover um *dashboard*, orientado a boas práticas de usabilidade, que permita auxiliar no acompanhamento e na compreensão de Sistemas Legados de uma empresa?

Segundo Oliveira; Cardoso (2015) e outros autores (FEW, 2006), há fatores que conferem usabilidade a um *dashboard*, sendo os principais:

- agradabilidade ao usuário;
- simplicidade na comunicação e na exposição de informações;
- facilidade de usar, orientando-se por padrões conhecidos;
- consistência no que é apresentado;
- informatividade, procurando auxiliar o usuário, e evitando dúvidas, e
- validação junto aos usuários, desde o seu desenho.

Nesse sentido, ocorreu a necessidade de um estudo mais criterioso sobre como desenvolver um *dashboard*, em atendimento a essas questões, que são inerentemente subjetivas. Além disso, esse desenvolvimento visou acompanhamento e compreensão de Sistemas Legados de uma empresa em particular. Sendo assim, ressaltam-se outros desafios impostos ao desenvolvimento, tais como:

- necessidade de levantamento e entendimento quanto à cultura da empresa, visando uma solução aderente ao portfólio de sistemas e aos funcionários dessa empresa;
- aproximação do autor ao contexto da empresa, não apenas na posição de um contratado, mas também como um profissional que visa conferir um apoio a um problema recorrente na empresa (i.e. dificuldade de se lidar com Sistemas Legados), e
- levantamento de algumas métricas de análise estática de código que agreguem valor ao suporte de apoio desenvolvido.

1.4 Objetivos

Orientando-se pelas Questões de Pesquisa e Desenvolvimento, procurou-se cumprir com o seguinte **Objetivo Geral:**

Acompanhamento facilitado de Sistemas Legados de uma empresa, procurando mitigar problemas como falta de documentação e dificuldades na compreensão desses sistemas. Teve como solução um suporte semiautomatizado e orientado a recursos visuais, com facilidade de uso e aderência à cultura da empresa.

Como o Objetivo Geral é abrangente, procurou-se atingi-lo através do cumprimento de alguns Objetivos Específicos, sendo:

- Objetivo_Específico_1: Estudo dos sistemas da Empresa A, tanto os Sistemas Legados quanto os sistemas mais modernos, com foco na documentação dos mesmos e na cultura da empresa;
- Objetivo_Específico_2: Levantamento das principais necessidades de documentação para os Sistemas Legados da empresa, considerando suas particularidades e limitações arquiteturais;
- Objetivo_Específico_3: Levantamento das principais métricas e tecnologias associadas, possivelmente, de análise estática de código, para auxiliar no acompanhamento e na compreensão desses Sistemas Legados;
- Objetivo_Específico_4: Estudo sobre Usabilidade em *dashboard*;
- Objetivo_Específico_5: Planejamento, Especificação, e Desenvolvimento de um *dashboard* para acompanhamento e compreensão desses Sistemas Legados usando como insumos os estudos e levantamentos anteriores, e
- Objetivo_Específico_6: Análise e Documentação dos resultados obtidos ao longo do trabalho, considerando o Estudo de Casos na Empresa A.

Salienta-se, para cumprimento desses objetivos, sobre a necessidade de orientação por metodologias de:

- pesquisa, para condução dos estudos e levantamentos;
- desenvolvimento, para construção da solução em si, e
- análise de resultados, para lidar mais adequadamente com os resultados obtidos.

1.5 Metodologia

No Capítulo 4 - Metodologia, serão tratados os aspectos metodológicos do trabalho de forma mais adequada. Entretanto, visando já esclarecer alguns pontos, o trabalho pode ser classificado, segundo GIL (2002), quanto:

- Aos objetivos, uma pesquisa exploratória, demandando investigação em contexto não conhecido completamente, tal como o próprio ambiente da Empresa A, além da necessidade de estudos e levantamentos junto à literatura especializada para maior familiaridade com os tópicos de interesse do trabalho;

- À abordagem, uma pesquisa híbrida (qualitativa e quantitativa), uma vez que se validou a solução junto à Empresa A com questionários de cunho mais qualitativo, ao mesmo tempo que ocorreu estudo de métricas, e apresentação das mesmas, incluindo as de viés quantitativo, e

- Aos procedimentos técnicos, pesquisa bibliográfica e Estudo de Casos, dado que demandou levantamentos junto à literatura para embasar a solução em padrões e práticas sólidas e já recomendadas na comunidade de *software*, além de ser aplicado em um contexto real, sendo esse a Empresa A.

1.6 Organização

A monografia está organizada em Capítulos, conforme consta a seguir:

- Capítulo 2 - Referencial Teórico: apresenta os principais conceitos e demais aspectos teóricos de relevância, os quais embasaram o trabalho como um todo;

- Capítulo 3 - Referencial Tecnológico: descreve as principais tecnologias, ferramentas e *frameworks*, ou seja, o arcabouço tecnológico que viabilizou não apenas o desenvolvimento da solução em si, mas também seu planejamento, sua especificação e sua análise;

- Capítulo 4 - Metodologia: classifica a pesquisa, e confere um detalhamento, em termos metodológicos, no intuito de facilitar a condução do trabalho na própria pesquisa, no desenvolvimento e na análise dos resultados obtidos;

- Capítulo 5 - Estudo de Casos: detalha o Estudo de Casos, usando como base uma adaptação do Protocolo acordado em GIL (2002), que delineou etapas para realização desse estudo (i.e. Formulação do Problema; Definição da Unidade-Caso; Determinação do Número de Casos; Elaboração do Protocolo; Coleta de Dados, e Redação do Relatório). Aqui, tem-se um conhecimento mais adequado sobre a Empresa A;

- Capítulo 6 - DASboard: apresenta o trabalho de forma mais ampla, procurando expor como surgiu a ideia e outras particularidades da mesma para que fiquem mais claras as intenções desse trabalho, e sejam apresentados os processos completos de planejamento e desenvolvimento do projeto até sua finalização;

- Capítulo 7 - Análise de Resultados: utiliza-se de coleta de dados da Empresa A com base no DASboard para tratar questões abordadas durante o trabalho, assim como evidenciar os impactos gerados pela inserção do trabalho em um ambiente real previamente analisado, e

- Capítulo 8 - Conclusão: acorda sobre os principais resultados obtidos com a realização do trabalho, bem como são retomadas as questões de pesquisas e os objetivos apresentados. Adicionalmente, procura-se mostrar novas ideias para melhorias futuras.

2 Referencial Teórico

Nesse capítulo, são conferidos os principais referenciais que embasaram o presente trabalho, com o intuito de realizar contribuições para a área de Engenharia de *Software*, especificamente ao que diz respeito aos Sistemas Legados. O capítulo encontra-se organizado em três seções principais, sendo as mesmas escritas orientando-se pela literatura: Sistemas Legados, focando na necessidade de uma documentação mais bem elaborada, o Documento de Arquitetura de *Software*; Arquitetura de *Software*, sendo essa uma estrutura de relevância para especificação e compreensão de um *software*, especialmente quando guiada por padrões e estilos arquiteturais; Métricas de *Software*, e Dashboard, com olhar mais aprofundado sobre o que é esse recurso, quais suas particularidades, e quais boas práticas permitem desenvolvê-lo com adequada usabilidade. Por fim, tem-se o Resumo do Capítulo.

2.1 Sistemas Legado

Visando conferir uma visão geral sobre Sistemas Legados, seguem subseções com focos, respectivamente, na definição e em aspectos de manutenibilidade.

2.1.1 Definição

Segundo Fritola; Santander (2022), os Sistemas Legados são sistemas antigos que continuam sendo utilizados pelas empresas. Comumente, esses sistemas muitas vezes também são muito grandes, de alta complexidade devido aos longos anos de alterações, e portanto, de difícil manutenção.

Outra definição, com base em literatura especializada (CROTTY; HORROCKS, 2017), permite considerar que um Sistema Legado, por vezes, é essencial para as atividades rotineiras da empresa, na qual está inserido, envolvendo altos riscos e constantes modificações. Segundo Crotty; Horrocks (2017), por vezes, esses sistemas são oriundos de uma época onde o processamento computacional e a capacidade de armazenamento não eram tão modernos. Dessa forma, a maior prioridade durante o desenvolvimento de tais sistemas era os mesmos serem eficientes e funcionais; ao contrário de compreensíveis e manuteníveis.

Complementarmente à definição, Crotty; Horrocks (2017) também citam uma forma não arbitrária de identificar Sistemas Legados. Nesse sentido, sugere-se a enumeração de pontos que realmente caracterizam um Sistema Legado, com destaque para:

- Avançada idade;
- Escassez ou ausência de documentação;
- Inadequação no gerenciamento de dados;
- Degradação na estrutura;
- Limitação na capacidade de suporte;
- Falta de clareza quanto ao entendimento de regras negociais;
- Altos custos de manutenção, e
- Fagilidades arquiteturais que dificultam ou impedem a evolução do sistema.

2.1.2 Manutenibilidade Evolutiva de *Software*

Segundo o Padrão IEEE 1219 (IEEE93), a manutenibilidade de *software* pode ser definida como alterações realizadas em um produto após entrega, para correções de falhas, melhoria de desempenho ou outros atributos, ou para adaptar o *Software* para uma mudança de ambiente. Essa definição também pode ser aplicada para caracterizar evolução de um produto de *Software*. Uma separação entre as definições foi feita por Bennet; Rajlich (2000), onde a manutenibilidade era definida como qualquer alteração no produto pós-entrega da última *release*, representando uma das últimas fases do ciclo de vida de um *software*. Já a evolução é um termo, acordado na área, e atuante após o produto ser considerado satisfatório, principalmente, em termos corretivos. Como a diferenciação entre manutenção e evolução não é de óbvia compreensão, há autores que sugerem o uso de um modelo de estágios. Tal modelo auxilia na melhor compreensão sobre o que é manutenção e o que é evolução de *software*, estabelecendo o momento de atuação de cada.

Este modelo de estágios, descrito por Bennet; Rajlich (2000), propõe uma forma de separar manutenção e evolução tomando como base diferentes estágios. Pode-se entender os estágios como momentos ao longo do ciclo de vida de um *software*. Um desses estágios é o desenvolvimento do produto. Nesse contexto, a manutenção do produto seria realizada ainda durante esse estágio, e a evolução seria aplicada durante os estágios seguintes.

Diante do exposto, a primeira versão do produto de *software* é oriunda do conhecimento adquirido pelos engenheiros responsáveis pelo seu desenvolvimento. É esperado, portanto, que se bem realizada, tende a ser mais fácil a condução da evolução, que ocorrerá mais adiante.

Após a primeira *release* do produto, a manutenção será mantida enquanto houver correções a serem feitas. Uma vez o produto visto como em estado satisfatório, demais melhorias ocorrerão já na evolução.

Cabe ressaltar que existem vários tipos de manutenção (LEMOS, 2011), conforme resumido a seguir:

- Preventiva: visa evitar o surgimento do problema (ex. investiga algo que impessa o funcionamento ou algo que provoque um desempenho indesejado que inviabilize o funcionamento), reduzindo a necessidade de manutenções corretivas;

- Corretiva: visa corrigir um defeito, uma falha, ou um problema observado no ativo de *software*;

- Preditiva: visa monitorar o ativo de *software*, usando como base tempo, tendências tecnológicas, e outros parâmetros/outras condições, no intuito de identificar possíveis comportamentos não esperados, e criar planos de ação para evitar problemas. Também conhecida como Manutenção Preventiva baseada na Condição, e

- Prescritiva: visa monitorar o ativo de *software*, similarmente como ocorre com a Manutenção Preditiva, mas acorda um plano de ação independentemente da ocorrência da um problema.

Há outros tipos de manutenção, mas não são focos diretos de atuação desse trabalho. Esse trabalho atuou, principalmente e temporalmente, na evolução de *software*, uma vez que os Sistemas Legados existem tem um tempo; já foram considerados produtos satisfatórios e, no momento, encontram-se dependentes de melhorias. Na prática, entretanto, o trabalho também compreende demandas de manutenção. Essa aparente confusão será mais bem explicada na sequência.

Segundo Ferreira; Leite (2013), a evolução de um produto de *software* é um fato que todos os desenvolvedores têm que lidar, uma vez que, na prática, compreende: necessidades de adaptação de uma empresa a diferentes circunstâncias; ou uma evolução direcionada à maior competitividade no mercado; ou modularização do produto para facilitar implementações. Independentemente da motivação, a evolução deve ser praticada, e pode ser mais custosa do que o desenvolvimento inicial de um produto.

Ferreira; Leite (2013) conferem uma proposta com a visão de reduzir o custo de evolução de um produto através de métodos de levantamento quanto às evoluções que podem ser necessárias durante o ciclo de vida de um dado produto (em estágios pós desenvolvimento satisfatório do mesmo), ao invés de realizar uma evolução muito mais cara e demorada no futuro. Nesse sentido, pode-se entender que os autores defendem especializações para evolução de *software* similares ao que ocorre na manutenção de *software*, como uma espécie de "evolução preventiva/preditiva/prescritiva". Entretanto, em se tratando de Sistemas Legados, não há essa possibilidade.

Sistemas Legados, como já explicado anteriormente, são produtos desenvolvidos e entregues há muito tempo, inerentemente acumulando um número grande de mudanças, adaptações, e correções, em diferentes épocas, e sem um senso de direção especificado previamente. Nesse caso, o foco concentrou-se em manter o produto funcionando, devido à criticidade do mesmo para os negócios da empresa. Portanto, pode-se dizer que há dificuldade de conduzir evolução em Sistemas Legados, mesmo os mesmos estando - temporalmente e teoricamente - situados pós desenvolvimento satisfatório. Ressalta-se que essa satisfação ocorreu no passado. No momento atual, é como se o Sistema Legado estivesse em estado permanente de manutenção, demandando, na prática, ser "revalidado", em termos satisfatórios, pelos seus usuários.

2.2 Arquitetura de *Software*

Produtos de *software* são constituídos de várias partes menores trabalhando em conjunto para viabilizar suas funcionalidades e atingir seus propósitos. Cada uma dessas pequenas partes possui particularidades, desenvolvidas orientando-se por estilos e padronizações, compondo a Arquitetura de *Software*.

De acordo com Fowler (2019), a palavra arquitetura no âmbito de *software* é utilizada para se referir à noção dos aspectos mais importantes do *design* interno de um sistema de *software*. Entretanto, na visão desse autor, a Arquitetura de *Software* é o entendimento conjunto do *design* do sistema, considerando as perspectivas dos desenvolvedores envolvidos na concepção de tal sistema. Há também a noção de que a Arquitetura de *Software* versa "Sobre as coisas que importam, nada mais".

Diante do exposto, é inferido que, ao se estabelecer uma Arquitetura de *Software*, se preocupa em priorizar o que é mais relevante ao sistema, além de prever possíveis melhorias e/ou necessidades, como por exemplo: ações corretivas ou evoluções.

Ainda de acordo com Fowler (2019), uma Arquitetura de *Software* incompleta ou mal feita tende a resultar em demasiado trabalho redundante, abandonado ou impeditivo; além de fatores que atrapalham a manutenção evolutiva. Tais situações não desejadas incorrem, frequentemente, em atrasos nas entregas de funcionalidades e melhorias, e em problemas e insucessos que desagradam os usuários.

Por fim, mas não menos importante, e considerando a relevância da Arquitetura de *Software* para o desenvolvimento de um sistema, cabe ressaltar que inconsistências e equívocos arquiteturais impactam sensivelmente o *modus operandi* de uma empresa, que depende do sistema, prejudicando sua eficiência e sua sustentabilidade (MARTIN, 2018).

Em um estudo de caso realizado por Martin (2018), foi avaliado que por mais que uma empresa tenha razoável efetivo de engenheiros desenvolvendo sistemas em expansão,

a sua eficiência de produção diminui e o custo de produção aumenta devido à falta de alinhamento claro na elaboração de um dado projeto de *software* em termos arquiteturais. Esse estudo, em uma empresa real que permanece anônima durante o relato, evidencia a importância de um adequado *design* interno para os produtos de *software*. Nesse sentido, para que um produto de *software* tenha longevidade, e seja sustentável financeiramente, é necessário fazer uso de estilos e padrões arquiteturais bem estabelecidos, considerando um nível de granularidade mais abrangente, além de padrões de projeto, já com foco em um nível de granularidade de menor escopo.

2.2.1 Padrões Arquiteturais

Padrões arquiteturais são definições presentes há muito tempo no âmbito da engenharia de *software* (GAMMA et al., 1994), porém sua definição pode diferenciar bastante. De acordo com Fowler (2019), e considerando um viés da engenharia civil e arquitetura de edificações, uma definição adequada para padrões de projeto compreende "cada padrão descreve um problema que ocorre de novo e de novo no nosso ambiente e também descreve a solução para esse problema, de forma em que você possa utilizar essa solução para cada instância do problema sem nunca repetir a forma de implementação". Apesar dessa definição não ser oriunda na área de engenharia de *software*, a mesma pode ser aplicada à Arquitetura de *Software* também.

O foco de um padrão, portanto, é identificar um problema específico que ocorre com frequência, e desenvolver a base de uma solução que se aplique a diferentes ramificações desse problema. Cada padrão pode ser independente dos outros, ou ainda podem ser utilizados em conjunto em busca de soluções refinadas, mais eficientes para problemas até mesmo não convencionais.

Os padrões arquiteturais, até pelo próprio nome, atuam no nível dos componentes e conectores arquiteturais, propondo estruturas e formas de comunicação orientadas a diretrizes e boas práticas. Existem vários padrões arquiteturais. No escopo desse trabalho, será aprofundado o padrão arquitetural MVC.

2.2.1.1 MVC

O Padrão arquitetural *Model View Controller* (MVC) é um dos padrões de projeto mais populares, sendo criado no fim da década de 70 como um *framework* para a plataforma Smalltalk (REENSKAUG, 1979).

O Padrão MVC propõe, em seu modelo conceitual, uma estrutura de projeto organizada três diferentes camadas:

- *Model*: Camada Arquitetural que representa a lógica negocial do projeto, não possuindo elementos visuais, contendo apenas as entidades e os serviços pertinentes ao

funcionamento do projeto;

- *View*: Camada Arquitetural que apresenta os elementos visuais, ou seja, a interface de usuário, utilizando os dados obtidos na Model para realizar a comunicação com o usuário da aplicação, não possuindo nenhuma implementação que não seja a disponibilização de informações pertinentes ao cliente/usuário, e
- *Controller*: Camada arquitetural que representa a mediadora entre as camadas *Model* e *View*, gerenciando as interações entre as camadas.

De acordo com Fowler (2019), há duas preocupações (na verdade, complementares) que procuraram ser mitigadas pelo Padrão MVC, sendo: propor uma separação entre a apresentação da aplicação e as entidades de domínio, e propor uma separação entre aspectos inerentes à lógica e elementos puramente visuais.

O interesse por mitigar tais preocupações é o diferencial do Padrão MVC em relação a outros padrões arquiteturais. Esse esforço em separar apresentação e domínio, e lógica e elementos visuais, é um dos princípios de *design* mais importantes no desenvolvimento de um *software*. Nesse contexto, o Padrão MVC propõe uma camada arquitetural dedicada à apresentação, sendo a *View*, e outra camada arquitetural dedicada à entidades de domínio, sendo a *Model*, além de uma camada arquitetural mediadora, sendo a *Controller*. Essa estratégia mantém as camadas mais coesas, com responsabilidades mais bem definidas, além de diminuir o acoplamento entre essas camadas. Alta Coesão e Baixo Acomplamento são padrões de projeto, conhecidos como GRASPs (LARMAN, 2004), e muito desejados em projetos de *software* que prezam por facilidades de manutenção evolutiva.

2.2.2 DAS - Documento de Arquitetura de *Software*

O DAS, ou Documento de Arquitetura de *Software*, é o documento que descreve a arquitetura do *software* desenvolvida ou em desenvolvimento, fornecendo uma visão geral do sistema, suas principais funcionalidades e as decisões arquiteturais tomadas durante o processo de desenvolvimento.

Segundo a definição de Bass; Kazman; Clements (2013), o documento de arquitetura de *software* é "uma descrição dos principais aspectos organizacionais do sistema, incluindo seus componentes, seus relacionamentos entre si e com o ambiente, e os princípios que orientaram o seu *design*, e a sua evolução ao longo do tempo". Ele fornece uma visão geral da arquitetura, destacando suas características principais e proporcionando um guia para os desenvolvedores envolvidos na realização do projeto.

A arquitetura de *software* é a base estrutural de um sistema e, portanto, sua documentação é essencial para facilitar a compreensão e a colaboração entre os desenvol-

vedores de uma equipe. Além disso, o documento de arquitetura de *software* serve como uma referência para futuras alterações, manutenções, atualizações e evoluções do sistema, ajudando a preservar a integridade da arquitetura ao longo do tempo.

A elaboração do documento de arquitetura de *software* envolve a análise dos requisitos do sistema, a definição dos componentes, a especificação de suas interfaces e interações, bem como a documentação das decisões de *design* e padrões utilizados. Ele pode incluir diagramas, modelos, descrições textuais e outras representações visuais para facilitar a compreensão da arquitetura.

É um documento de extrema importância, segundo Mariotti (2012), não importa o quão adequada às normas e simplificada a arquitetura seja. Para desenvolvedores que não possuem familiaridade, o projeto tende a se tornar confuso e trabalhoso quando o intuito é adaptá-lo, e até mesmo conhecê-lo. Portanto, a documentação torna-se essencial.

O Documento de Arquitetura de *Software* é comumente dividido em algumas seções, entre elas estão:

- Histórico de revisões: Identifica a versão do documento arquitetural e do código desenvolvido, incluindo dados sobre quem está responsável pelo projeto e o que foi trabalhado em cada etapa;
- Introdução: Apresenta o que será abordado no documento, com as seguintes subseções:
  - Propósito: Define a razão do DAS estar sendo criado, além de descrever brevemente a estrutura do documento;
  - Escopo: Descreve sobre o que o projeto engloba no contexto proposto, o que é afetado ou influenciado por ele;
  - Definições, siglas e abreviações: Provê as definições de todos os termos, siglas e abreviações estabelecidos durante a escrita do DAS, e
  - Referências: Lista todo o embasamento utilizado durante a escrita do documento arquitetural, identificando cada material consultado, sendo esse apresentado de forma adequada e de acordo com as normas da (ABNT, 2023).
- Objetivos arquiteturais e restrições: Descreve os requisitos de *software* e objetivos que podem ter impacto na arquitetura, indicando o que já foi previamente estabelecido para o desenvolvimento em termos qualitativos (ex. desempenho, segurança e usabilidade), além das restrições que podem ocorrer (ex. ferramentas a serem utilizadas e estrutura da equipe);
- Mecanismos arquiteturais: Consiste em uma listagem organizada dos mecanismos arquiteturais utilizados no projeto, classificados de acordo com a área envolvida,

no intuito de verificar e garantir que todas as preocupações técnicas relativas à arquitetura do sistema tenham sido cobertas e analisadas;

- Visão geral dos componentes: Destaca os principais componentes, formando uma compreensão breve sobre o que está sendo especificado;

- Visão de casos de uso: Descreve os cenários de potencial aplicação do projeto. Faz-se uso de Diagramas de Casos de Uso, bem como de especifiicações estendidas de Caso de Uso;

- Visão lógica: Apresenta as partes significativas da arquitetura do projeto, demonstrando a visão em subsistemas e pacotes. Para cada pacote, uma descrição detalhada sobre classes relevantes e suas utilidades. Pode conter diagramações estáticas (ex. Diagrama de Classes e Diagramas de Pacotes) e dinâmicas (ex. Diagrama de Atividades e Diagrama de Sequência), desde que as mesmas revelem aspectos de cunho lógico;

- Visão de processos: Mostra de forma organizada os processos do sistema, compreendendo uma visualização arquitetural sobre a decomposição dos processos do sistema, incluindo o mapeamento de classes e subsistemas para processos e encadeamentos, além de detalhamentos sobre sincronismo, assincronismo e outras nuances do Paradigma Concorrente e da programação paralela. Faz-se uso tanto de diagramações estáticas, quanto de diagramações dinâmicas, desde que as modelagens revelem processos;

- Visão de implantação: Descreve as partes físicas responsáveis pela execução e pela hospedagem do projeto descrito no documento arquitetural. Faz-se uso do Diagrama de Implantação, revelando nós de implantação e os principais protolocos para comunicação entre esses nós;

- Visão de implementação: Detalha a estrutura geral da implementação do projeto, decompondo o *software* em diferentes camadas e subsistemas no modelo de implementação, indicando ferramentas de integração como bases de dados e bibliotecas externas. Faz-se uso do Diagrama de Componentes, revelando os principais componentes arquiteturais e conectores arquiteturais do sistema em questão;

- Fundamentação: Compreende a necessidade de uma fundamentação de "como"e "porque"cada decisão arquitetural foi tomada;

- Tamanho e Desempenho: Descreve as principais características do *software* que impactam a arquitetura (ex. porte do sistema), assim como as metas de desempenho e suas restrições, e

- Qualidade de *Software*: Descreve como a arquitetura do *Software* contribui para as capacidades gerais do sistema, como: portabilidade, confiabilidade, dentre outras. Se as características são de grande importância como segurança e privacidade, elas devem ser claramente definidas.

2.3 Métricas de *Software*

As métricas desempenham um papel muito importante na avaliação de qualidade, desempenho e manutenibilidade de um sistema de *software*. Elas fornecem uma base objetiva para medir e analisar características e propriedades do *software*, permitindo uma compreensão mais aprofundada de sua qualidade. De acordo com Pressman; Maxim (2021), as métricas de *software* são fundamentais na engenharia de *software*, pois fornecem meios quantitativos de avaliar o desempenho, a qualidade e a eficiência de um sistema.

As métricas permitem identificar problemas e pontos de melhoria, possibilitando a tomada de decisões durante o processo de desenvolvimento e manutenção do *software*. Pressman; Maxim (2021) destacam a importância das métricas de *software* como uma ferramenta essencial para estimar esforço, tempo e recursos necessários para desenvolver e manter um sistema, auxiliando na gestão de projetos e no planejamento estratégico.

2.3.1 Métricas Gerais

Existe uma grande variação de métricas de *software* disponíveis para diferentes usos na avaliação de um projeto. As métricas a serem escolhidas variam de acordo com a necessidade dos sistemas e dos objetivos de desenvolvimento. Para este trabalho, as principais métricas consideradas foram:

- Linhas de Código: Conta o número total de linhas de código-fonte do sistema, e fornece uma medida geral do tamanho do *software*, podendo ser usadas para estimar o esforço necessário para desenvolvê-lo. No entanto, deve-se ter cuidado ao interpretar essa métrica, uma vez que o tamanho do código nem sempre reflete a complexidade ou a qualidade do *software*. Com o uso de geração de código, frequentemente oferecida pelas plataformas e pelos *frameworks* de desenvolvimento, há quantidade representativa de código gerado, automaticamente, e que não demandou esforço do programador. Já há momentos, que poucas linhas da lógica de programação demandam grande esforço desse profissional;

- Taxa de Defeitos: Essa métrica mede a quantidade de defeitos encontrados em um determinado período de tempo, geralmente expressa como o número de defeitos por linha de código ou por função. A taxa de defeitos é um indicador da qualidade do

software, e pode ser usada para identificar áreas problemáticas que precisam de mais atenção e melhorias;

- Complexidade Ciclomática: Acomplexidade ciclomática é uma métrica que mede a complexidade estrutural de um programa, contando o número de caminhos independentes de execução (MCCABE, 1976). Ela fornece uma indicação da dificuldade de entender, testar e manter o *software*. Uma complexidade ciclomática alta pode indicar a necessidade de refatoração ou simplificação do do código. Ressalta-se que essa métrica pode ser obtida via análise estática de código. Por isso, será também coberta mais adiante nesse capítulo, e
- Testes Unitários: A cobertura de testes mede a proporção do código ou funcionalidades do sistema que são exercidas pelos testes automatizados. Ela fornece uma medida da eficácia dos testes realizados, identificando áreas que não estão adequadamente cobertas. Uma alta cobertura de testes é desejável para garantir a qualidade e a confiabilidade do *software*.

2.3.2 Métricas de Análise Estática de Código

A análise estática de código é uma prática muito importante no processo de desenvolvimento de um produto, com o intuito de identificar possíveis problemas e melhorar a qualidade do código. Dentre as métricas existentes, para o contexto deste trabalho, destacam-se:

- Complexidade ciclomática (MCCABE, 1976);
- Code Smells (FOWLER, 2019);
- God object (MARTIN, 2018) e
- Lista longa de parâmetros (FOWLER, 2019).

A complexidade ciclomática (MCCABE, 1976) é uma medida que avalia a complexidade estrutural de um programa, sendo um indicativo da dificuldade de entendimento e manutenção do código da aplicação. Essa medida é amplamente utilizada para identificar partes do código que podem ser mais propensas a erros e exigir maior esforço para teste e manutenção.

De acordo com Fowler (2019), os *code smells* são padrões que indicam a presença de possíveis problemas no código que não são necessariamente impeditivos para o funcionamento mesmo. Entretanto, representam práticas não adequadas, que podem levar a problemas mais sérios no futuro. Os *code smells* podem incluir duplicação de código, código morto, classes ou métodos muito extensos, entre outros. Um dos maiores problemas de Sistemas Legados muito robustos dá-se pelo acúmulo desses problemas.

O *god object* refere-se à uma classe que possui muitas responsabilidades e funções atribuídas à ela (MARTIN, 2018), tornando-se difícil mantê-la, entendê-la e alterá-la. Nesse cenário não desejado, uma mudança em algo dessa classe pode gerar impactos em outras partes do sistema que possuem correlações com essa classe. Torna-se, portanto, difícil realizar manutenções e evoluções nesse sistema.

A lista longa de parâmetros, de acordo com Fowler (2019), é um indicativo de que uma função possui um número excessivo de argumentos, o que pode prejudicar a manutenção e as próprias leituras e compreensão do método. Trata-se, também, de uma evidência de *code smell*.

Essas métricas, quando aplicadas de forma adequada, contribuem para a detecção de possíveis problemas e auxiliam na melhoria contínua da qualidade do código-fonte.

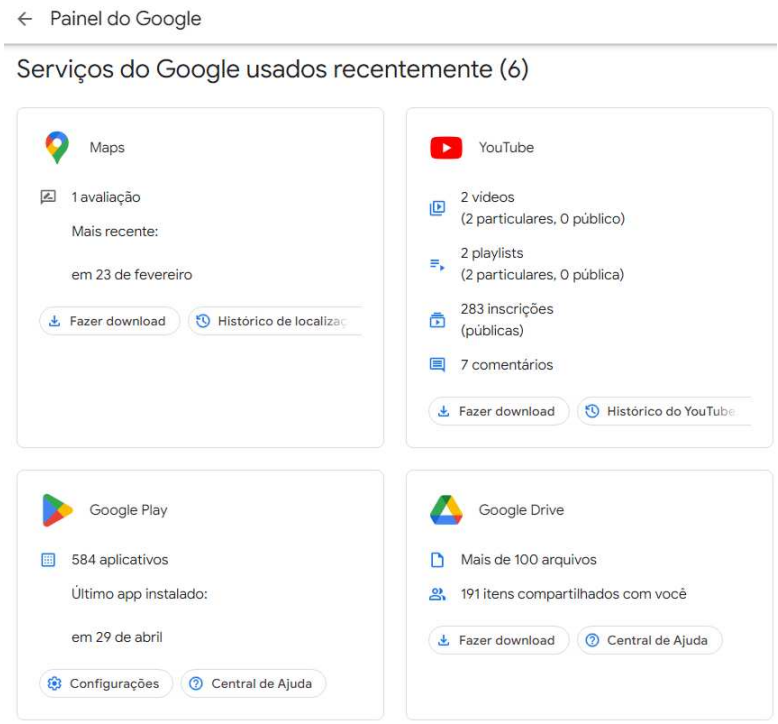
2.4 *Dashboard*

Segundo Tebaldi (2017), um *Dashboard* no contexto da Tecnologia da Informação é um painel que disponibiliza um conjunto pertinente de informações a seu usuário, sendo essas informações indicadores e suas métricas. O *Dashboard* pode ser utilizado para demonstrar a informação necessária para acompanhamento de qualquer produto de *software*. Trata-se de uma ferramenta visual que auxilia usuários a entenderem de forma mais simples o estado do produto e o que está sendo rastreado.

Um *Dashboard* pode ser utilizado para analisar desempenho do produto de *software* e, se atrelado à uma ferramenta de análise de código (SONARQUBE, 2023), pode conferir insumos relevantes para monitoramento e acompanhamento de outros aspectos, tais como: acoplamento, *code smells*, vulnerabilidades, dentre outros. O principal propósito desse recurso visual é ser dinâmico e objetivo para que não haja confusão ao analisar os dados. Sendo assim, é importante que o *Dashboard* esteja sempre sendo alimentado com novos dados, promovendo futuras análises do estado do produto.

Um exemplo de *Dashboard* pode ser visto na Figura 1.

Figura 1 – Exemplo de *Dashboard*



Fonte: Autor (via Google)

A Figura 1 ilustra o Google Dashboard de um perfil aleatório, não identificado, pois não é objetivo do autor expor o usuário em si. A ideia é apenas ilustrar um típico *Dashboard*, oferecido por uma empresa de renome (o Google), e que auxilia o usuário entender sobre o *Maps* (ex. última avaliação de local); YouTube (ex. número de inscrições e *playlist*); *Drive* (ex. número de arquivos e seus compartilhamentos); Google Play (ex. número de aplicativos instalados), dentre outras informações do perfil desse usuário.

Um *Dashboard* pode ainda ser muito atraente visualmente, com exposição de gráficos, em paleta de cores agradáveis, dentre outras práticas da área de *UI/UX Design*. Nesse sentido, há necessidade de um aprofundamento sobre como elaborar um *Dashboard* orientando-se por boas práticas de cunho visual. Cabe mencionar as orientações e diretrizes conferidas no Portal (NNG, 2021), com foco em Usabilidade.

Na apresentação de dados (NNG, 2021), há destaque para as seguintes boas práticas:

- Não detalhar demasiadamente os dados em um *Dashboard*, para que o usuário não passe horas analisando o que está na tela. Os dados devem ser simples de entender e visualizar para que o usuário consiga, com um olhar, identificar problemas e a situação atual do que está sendo disponibilizado. Os dados que são mais fáceis para se entender, considerando o cérebro humano, são dados apresentados em tamanho e posição em um espaço 2D (NNG, 2017). Porém, há certa dificuldade ao entender

dados disponibilizados em ângulo, cor ou área, isto é, comparativamente aos dados de mais fácil entendimento. Torna-se necessário evitar o uso excessivo de dados disponibilizados por meio de gráficos de pizza ou gráficos de medidores, optando, sempre que possível, por disponibilizar esses dados em gráficos de barras ou colunas;

- Desenvolver um *Dashboard* com uma boa usabilidade pode se tornar um desafio, dependendo das decisões tomadas em tempo de *design* e planejamento da interface, bem como das próprias funcionalidades. Sendo assim, é importante seguir práticas já estabelecidas de melhoria de usabilidade de usuários para prover uma boa experiência de usuário, facilitando a análise dos dados disponibilizados, e

- Utilizar o método *Design* Iterativo. Este método compreende diferentes iterações da interface em diferentes fontes para prover revisões sequenciais da interface do *Dashboard*, fazendo assim ajustes otimizados para o usuário. De acordo com o NNG (2017), esse método é mais bem utilizado quando pareado com outros dois métodos de otimização de usabilidade. Um dos métodos de otimização é o *Design* Paralelo, consistindo em um número X de diferentes interfaces desenvolvidas para o mesmo propósito e com diferentes artifícios visuais. Essas diferentes interfaces passam por revisão e testes com o usuário. Com base nos resultados obtidos, os melhores atributos de cada interface são migrados para uma interface final, com intenção de agradar o cliente da aplicação. Outro método de otimização é o Teste Competitivo, consistindo em testar o *Design* desenvolvido com as interfaces de outras companhias (ex. concorrentes de mercado), além de diferentes *Dashboards*. Por fim, integram-se os melhores elementos das interfaces, sem necessitar de criar novas iterações. Nesse contexto, a melhor forma de se guiar em busca de uma usabilidade aceitável, considerando esses métodos, dá-se pelo uso combinado deles, em ordem (NNG, 2011). Primeiramente, tem-se o Teste Competitivo, visando identificar as necessidades gerais do usuário. Na sequência é aplicado o *Design* Paralelo, conferindo diferentes soluções para essas necessidades. Por fim, várias iterações dessas interfaces seriam criadas para otimizar e refinar a experiência do usuário por meio do *Design* Iterativo.

No contexto deste trabalho, o *Dashboard* é utilizado para apresentação de insumos sobre os Sistemas Legados da Empresa A. Utiliza-se ainda para conferir maior clareza sobre as visões arquiteturais desses Sistemas Legados, em concordância com os apontamentos revelados nas seções anteriores. Adicionalmente, dentre os insumos de relevância, foram utilizadas métricas específicas, abordadas em seção anterior, orientada à literatura especializada, e que apresentou uma visão geral sobre métricas de análise estática de código.

2.5 Resumo do Capítulo

O capítulo apresentou o embasamento teórico principal que foi utilizado durante o desenvolvimento do trabalho. Embasamento esse que aborda os Sistemas Legados, a Arquitetura de *Software* e os Padrões de Projeto, assim como o Padrão Arquitetural MVC utilizado no desenvolvimento do *Dashboard*. Adicionalmente, demonstrou-se a definição e a importância de um Documento de Arquitetura, o qual acorda, dentre outros tópicos de relevância, visões arquiteturais. Cada visão confere detalhamentos arquiteturais específicos sobre um dado sistema, com ênfase para as partes: lógica, de dados, de processos, de implantação e de implementação, além de casos de uso.

Ocorreu ainda o levantamento de métricas, especificamente métricas de viés mais geral (ex. linhas de código e taxa de defeitos) e métricas de análise estática de código (ex. *code smells*, cobertura de código), as quais são utilizadas para evidenciar, via *Dashboard*, insumos referentes aos Sistemas Legados. Insumos que viabilizem análises mais adequadas sobre os Sistemas Legados em termos de baixa coesão, alto acoplamento, replicação de código, código morto, dentre outros problemas que dificultam sensivelmente a manutenção evolutiva do sistema.

Além disso, foi apontada a definição de um *Dashboard*, o qual é o produto final desse trabalho. No intuito de desenvolver um produto com adequada usabilidade, ocorreu um estudo sobre boas práticas atreladas ao desenvolvimento de *Dashboard* orientando-se por esse critério qualitativo, o que resultou no produto de estudo deste trabalho. Nesse estudo, procurou-se exemplificar um típico *Dashboard* de uma empresa de relevância, no caso o Google. Complementarmente, abordou-se sobre orientações e diretrizes recomendadas por autores da área, buscando identificar práticas concretas para prover uma solução computacional coerente, com destaque para: *Design* Iterativo, *Design* Paralelo e Teste Competitivo.

3 Referencial Tecnológico

Nesse capítulo, serão apresentadas as principais ferramentas e tecnologias que foram utilizadas para elaboração, condução, desenvolvimento e análise do trabalho como um todo. Começando pelas principais ferramentas, as quais viabilizaram o desenvolvimento e a análise do *Dashboard*, Ferramentas Associadas ao *Dashboard*, têm-se: versionamento/hospedagem com uso do GitHub; *frontend* do produto com utilização do Angular; *backend* do produto com uso da linguagem de programação de alto nível Java; armazenamento de dados com o *Firebase Realtime Database*, uma plataforma oferecida pela Google que possui serviços de Banco de Dados em tempo real; e coleta e análise de resultados com Google Forms. Ocorreu ainda a necessidade de uma ferramenta que viabilizesse a análise estática de código dos Sistemas Legados, orientando-se por métricas. Nesse caso, em Ferramenta de Análise Estática de Código, há menção ao SonarQube.

Entretanto, ainda existem outros apoios tecnológicos de relevância, e que merecem ser destacados. Em Ferramentas de Modelagem, há destaque para o Figma, que viabilizou a prototipação da solução; LucidChart, para elaboração de modelos UML, referentes às visões arquiteturais dos Sistemas Legados; e Bizagi, para modelagem de processos metodológicos na notação BPMN. Em Outros Apoios Tecnológicos, há destaque para Ferramentas de Comunicação (ex. Slack, Teams e Telegram); e Ferramentas de Escrita da Monografia (ex. Latex e Overleaf). Por fim, tem-se o Resumo do Capítulo.

3.1 Ferramentas Associadas ao *Dashboard*

Seguem as principais ferramentas que foram consumidas para viabilizar o desenvolvimento e a análise do *Dashboard*.

3.1.1 Angular

O Angular (ANGULAR, 2023) é uma ferramenta de desenvolvimento *open-source* *Frontend*, oferecida pelo Google, com base em *JavaScript* e *TypeScript*. Trata-se de uma ferramenta utilizada para criar aplicações *web* com alta escalabilidade, responsividade e interatividade. O Angular segue, primariamente, o Padrão MVC, onde possui componentes que representam a camada arquitetural *View*, dedicada à interface que permitirá a comunicação com o cliente que esteja utilizando a aplicação.

O Angular também oferece diferentes recursos para desenvolvimento, tais como integração com serviços API RESTful; injeção de dependências; validação dos formulários do *Frontend*, e manipulação de eventos. Adicionalmente, há uma extensa - e sempre em

expansão - biblioteca de componentes visuais, disponíveis aos desenvolvedores, chamada de Angular Material.

O Angular ainda oferece o Angular CLI, a ferramenta de interface de linha de comando que auxilia no manuseio de aplicativos Angular, sem necessidade de outras ferramentas para compilação e empacotamento. Por fim, cabe mencionar sobre a vasta documentação de todas as ferramentas disponíveis. O Angular foi utilizado, nesse trabalho, para desenvolvimento do *Frontend* do *Dashboard*.

3.1.2 Java

Java (ORACLE, 2023) é uma linguagem de programação de alto nível, versátil, e amplamente utilizada. Java possui características importantes que a tornaram muito popular, entre elas: portabilidade, segurança e robustez o suficiente para conseguir ser a base de sistemas empresariais de alto valor. É uma linguagem orientada a objetos, o que significa que produtos em Java são implementados com classes e objetos, e utilizando abstrações do modelo conceitual orientado a objetos, focado em reutilização facilitada dos componentes da aplicação. Outro atrativo da linguagem dá-se pelo seu gerenciamento automatizado de memória, evitando comuns problemas envolvendo alocação de memória.

Produtos Java possuem alta portabilidade. Sendo assim, a aplicação Java compilada pode ser executada em qualquer ambiente que possua a JVM, Máquina Virtual Java, disponível em diferentes sistemas operacionais, evitando a necessidade de implementações específicas. Assim como o Angular, Java possui uma grande e sempre em expansão biblioteca de dependências, disponíveis para uso. Essa biblioteca oferece uma abrangente variedade de funcionalidades para implementação na aplicação.

Java também possui diferentes *Frameworks*, os quais habilitam o desenvolvimento de aplicações de alto desempenho e suporte a *threads* múltiplas. A linguagem Java foi utilizada, nesse trabalho, para desenvolvimento do *Backend* do *Dashboard*.

3.1.3 Firebase Realtime Database

Firebase (GOOGLE, 2023a) é uma ferramenta de desenvolvimento de produtos *web* e *mobile*, fornecida pelo Google. Trata-se de uma ferramenta que oferece muitos serviços para auxiliar o desenvolvimento de aplicativos com maior facilidade. O principal recurso do Firebase de interesse desse trabalho é o banco de dados em tempo real, chamado *Firebase Realtime Database*.

O *Firebase Realtime Database* é um serviço oferecido pelo Firebase, que consiste em um banco de dados *NoSQL* baseado em *JSON*, o qual sincroniza em tempo real dados do cliente utilizando a aplicação. Quando há alguma alteração na aplicação, o *Firebase Realtime Database* atualiza automaticamente sua base de dados, em todas as

instâncias conectadas entre todos os clientes, permitindo uma visualização dinâmica e consistente entre todos os usuários do produto. Essa funcionalidade é comumente utilizada para aplicativos de conversa ou para jogos digitais entre vários jogadores.

O modelo do *Firebase Realtime Database* é baseado em árvores, onde os dados são separados por nós. Cada nó possui um "galho"que o identifica. Cada nó possui pares chave-valor no formato padrão *JSON*, onde a chave é um valor único, e o valor pode ser qualquer dado. O modelo em árvore possibilita uma organização detalhada, além de facilitar consultas aos dados utilizando os caminhos específicos de cada nó. Para o contexto deste trabalho o Firebase foi utilizado para armazenar os dados de análises realizadas pelo DASboard.

3.1.4 Google Forms

O Google Forms (GOOGLE, 2023b) é uma ferramenta de criação de formulários e questionários, disponibilizado pelo Google. O Google Forms permite que os usuários criem pesquisas, enquetes e formulários personalizados para coletar informações e dados pertinentes a diferentes questões e temas.

Com o Google Forms, os usuários podem criar perguntas de vários formatos, como texto curto, texto longo, múltipla escolha, lista suspensa, caixas de seleção e escala de classificação. Eles também têm a opção de adicionar imagens e vídeos para complementar as perguntas. Ao receber respostas, o Google Forms organiza de forma automática os dados em uma planilha de fácil visualização, auxiliando na análise e na leitura dos resultados.

No contexto deste trabalho, o Google Forms foi utilizado para coletar informações de desenvolvedores da Empresa A, referentes à situação de seus diferentes Sistemas Legados, incluindo documentação e problemas em geral (ex. dificuldade de entendimento). A intenção foi que essas informações guiassem o desenvolvimento do *Dashboard*, bem como o levantamento de insumos que foram disponibilizados aos desenvolvedores sobre os Sistemas Legados. para os desenvolvedores.

3.2 Ferramenta de Análise Estática de Código - SonarQube

O *SonarQube* (SONARQUBE, 2023) é uma ferramenta de análise estática de código, utilizada para avaliar a qualidade do código em projetos de *software*. Ele fornece um conjunto abrangente de ferramentas e métricas que ajudam a identificar e corrigir problemas no código de aplicações, incrementando segurança, legibilidade e manutenibilidade do código. O *SonarQube* analisa o código em busca de *bugs*, vulnerabilidades de segurança e boas práticas gerais de desenvolvimento. O *SonarQube* possui análises para várias tecnologias diferentes, incluindo a tecnologia utilizada no *Backend* deste trabalho.

A análise do código pelo *SonarQube* é baseada em regras predefinidas que abrangem padrões comuns de codificação, convenções de nomenclatura, complexidade do código, entre outros critérios. Ao identificar essas questões, o SonarQube fornece relatórios detalhados, permitindo que os desenvolvedores visualizem e compreendam os problemas em seu código, e tomem as medidas adequadas para corrigi-los.

O *SonarQube* também possui uma extensão que analisa o código em tempo real, durante desenvolvimento, que auxilia com apontamento de problemas durante a escrita do código, chamada *SonarLint*.

No contexto deste trabalho, o *SonarQube* foi utilizado para avaliar os maiores problemas dos Sistemas Legados da Empresa A, assim como realizar a análise do código desenvolvido pelo autor para o próprio *Dashboard* via *SonarLint* para maior qualidade do código desenvolvido, e também para integração com a aplicação para disponibilização de métricas do código analisado.

3.3 Ferramentas de Modelagem

Seguem as principais ferramentas que auxiliaram na modelagem de artefatos inerentes ao trabalho, com destaque para modelagem do protótipo do *Dashboard*; modelagens referentes às visões arquiteturais do Documento de Arquitetura dos Sistemas Legados, e modelagens dos processos metodológicos que orientaram as atividades desse trabalho.

3.3.1 Figma

O Figma (FIGMA, 2023) é uma ferramenta de *design* utilizada para criar interfaces de aplicações com uma grande variedade de recursos de criação de interfaces de usuário. O Figma também possui uma interface intuitiva, que permite criação de protótipos interativos.

Uma das principais vantagens do Figma é sua capacidade de funcionar diretamente no navegador, eliminando a necessidade de instalação de *software* adicional. Isso permite que os usuários acessem e trabalhem em seus projetos de *design* de qualquer lugar, facilitando a colaboração remota.

No contexto deste trabalho, o Figma foi utilizado para criar os diferentes protótipos de interface do *Dashboard*.

3.3.2 LucidChart

O Lucidchart (LUCID, 2023) é uma ferramenta de criação de diagramas e fluxogramas, possuindo um grande número de recursos para diferentes fins. O LucidChart permite que usuários criem diagramas para diferentes finalidades e propósitos.

Trata-se de uma ferramenta que possui variados modelos e formas pré-carregadas, sendo disponibilizadas gratuitamente bibliotecas de formas, ícones e elementos gráficos. Esses recursos podem ser facilmente integrados no diagrama em desenvolvimento, tornando a criação desses diagramas muito mais eficiente e simples. Assim como o Figma, o Lucidchart está disponível diretamente no navegador, acessível de qualquer dispositivo com conexão à internet, eliminando a necessidade de instalação de *software* adicional.

No contexto deste trabalho, o LucidChart foi utilizado para criação das diferentes visões arquiteturais do Documento de Arquitetura de *Software* dos Sistemas Legados da Empresa A, e também para especificação de artefatos mais técnicos visando exposição dos detalhamentos do projeto do próprio *Dashboard*.

3.3.3 Bizagi

O Bizagi (BIZAGI, 2023) é uma ferramenta de automação de processos de negócios que ajuda seus usuários a modelar, automatizar e melhorar seus fluxos de trabalho e processos cotidianos.

Uma das principais características do Bizagi é a sua interface intuitiva de modelagem de processos, permitindo criação de diagramas de fluxo de processos usando a notação BPMN para visualizar e documentar as etapas, decisões, atividades e interações envolvidas em um processo de negócio. Essa modelagem ajuda a identificar impedimentos, ineficiências e oportunidades de melhoria nos processos existentes.

No contexto deste trabalho, o Bizagi foi utilizado para modelagem de processos dos Sistemas Legados da Empresa A na notação BPMN (*Business Process Model and Notation*), além de auxiliar na modelagem dos processos do próprio *Dashboard*.

3.4 Outros Apoios Tecnológicos

Seguem as demais ferramentas que auxiliaram na comunicação entre os envolvidos no projeto, bem como na escrita dessa monografia.

3.4.1 Ferramentas de Comunicação

Para comunicação entre autor e orientadora, durante a elaboração desse trabalho, destacam-se:

- O Slack (TEAMS, 2023), para orientações periódicas, com rastros de revisões e dúvidas;
- O Microsoft Teams (MICROSOFT, 2023), para reuniões virtuais semanais sobre o andamento do trabalho, e

- O Telegram (TELEGRAM, 2023), para trocas de mensagens e avisos rápidos.

3.4.2 Ferramentas de Escrita da Monografia

Para escrita da monografia, destaca-se o uso do LaTeX (LATEX, 2023), sendo este uma linguagem de marcação para escrita de monografias comumente utilizada. Como principal vantagem tem-se a escrita em forma de texto simples, sem haver tanta preocupação com a formatação durante a escrita do texto. Adicionalmente, foi utilizado o Visual Studio Code (Microsoft, 2024) para escrita do texto no formato LaTeX e organização das pastas, além de ser utilizado para compilar a monografia em formato PDF.

3.5 Resumo do Capítulo

O capítulo apresentou as ferramentas que foram utilizadas no desenvolvimento deste trabalho tanto para o desenvolvimento do projeto do *Dashboard*, quanto para a análise dos dados coletados da Empresa A, bem como para modelagem, comunicação e escrita. Foi apresentada a tecnologia do *Frontend*, Angular, assim como a tecnologia escolhida para o *Backend*, Java. Na sequência, comentou-se sobre o Firebase e seu recurso, *Firebase Realtime Database*, sendo esses utilizados para o Banco de Dados do *Dashboard*.

Ocorreu ainda a exposição de outras ferramentas, referentes à análise de código, no caso, o SonarQube; bem como ferramentas para criação de artefatos de modelagem (ex. protótipo do *Dashboard*), e ferramentas de apoio à comunicação e à escrita.

A Tabela 1 revela as principais ferramentas, complementando com versão, descrição e *link* para informações adicionais.

Tabela 1 – Ferramentas Utilizadas no Trabalho

Nome da Ferramenta	**Descrição**	**Link**
Angular v14.1.3	Ferramenta utilizada para o *Frontend* do *Dashboard*	<https://angular.io>
Java 21	Linguagem de programação utilizada para desenvolver o *Backend* do *Dashboard*	<https://www.oracle.com/java>
Firebase Realtime Database 9.1.1	Ferramenta de banco de dados utilizando o *Firebase Realtime Database* para armazenar os dados e as métricas visualizadas no *Dashboard* (Via AngularFire)	<https://firebase.google.com/docs/database>

Tabela 1 – Continuação

Nome da Ferramenta	**Descrição**	**Link**
SonarQube 10.5.1	Ferramenta de análise estática de código para auxiliar a identificar problemas nos Sistemas Legados da Empresa A e no próprio desenvolvimento do *Dashboard*	<https://docs.sonarqube.org/latest/>
Figma	Ferramenta de prototipagem de interfaces de usuário para criação do protótipo do *Dashboard*	<https://www.figma.com>
LucidChart	Ferramenta de modelagem de diagramas UML para criação do Documento de Arquitetura de *Software* e análise das visões de Sistemas Legados da Empresa A	<https://www.lucidchart.com>
Bizagi	Ferramenta de modelagem de processos na notação BPMN para documentar processos metodológicos e outros fluxos de relevância.	<https://www.bizagi.com/en/platform/modeler>
Telegram	Aplicativo utilizado para trocas de mensagens entre autor e orientadora, bem como para avisos gerais rápidos.	<https://telegram.org>
LaTeX	Linguagem de marcação para desenvolvimento e escrita do TCC	<https://www.latex-project.org>
Slack	Ferramenta utilizada para revisões do TCC e comunicação geral	<https://slack.com>
Microsoft Teams	Ferramenta utilizada para reuniões semanais de acompanhamento do trabalho	<https://www.microsoft.com/en-us/microsoft-teams>

Tabela 1 – Continuação

Nome da Ferramenta	**Descrição**	**Link**
Visual Studio Code	Ferramenta de escrita do trabalho e organização de pastas e compilação do trabalho	<https://code.visualstudio.com>
Fonte: Autor		

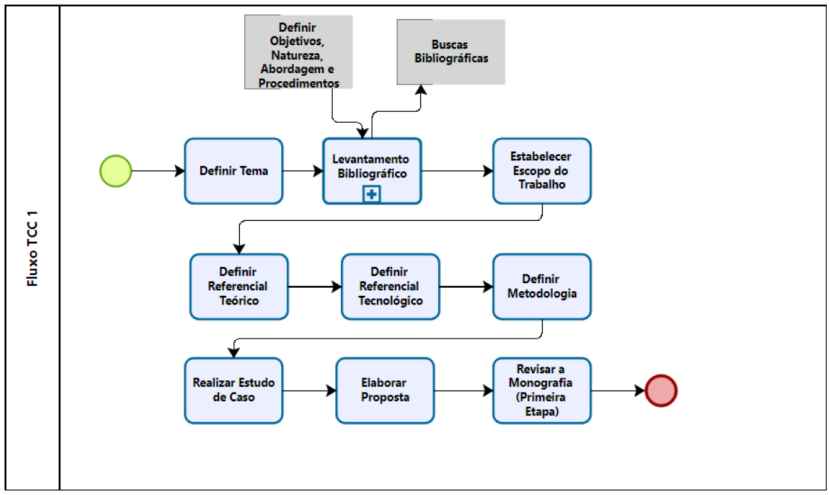
4 Metodologia

Neste capítulo, serão apresentados os principais aspectos metodológicos que direcionaram o trabalho. Há, primeiramente, a definição do Fluxo de Atividades que foram desenvolvidas no decorrer do trabalho. Na sequência, tem-se a definição das diferentes metodologias adotadas, sendo estas a Metodologia de Pesquisa, a Metodologia de Desenvolvimento e a Metodologia de Análise de Resultados. Complementa-se com menção aos Cronogramas de atividades que foram realizadas no decorrer do trabalho Por fim tem-se o Resumo do capítulo.

4.1 Fluxo de Atividades

O projeto foi dividido em duas etapas, primeira etapa do TCC e segunda etapa do TCC. As atividades e os subprocessos referentes à primeira etapa do TCC seguem um fluxo, sendo esse ilustrado na Figura 3, na notação BPMN, e descrito a seguir. Cabe mencionar que foram omitidos alguns artefatos gerados e consumidos pelas atividades e pelos subprocessos no nível de modelagem, mantendo-os apenas na descrição textual. A ideia é concentrar a atenção do leitor na cadência do fluxo em si.

Figura 2 – Fluxo de Atividades - Primeira Etapa do TCC



Fonte: Autor usando Bizagi

- Definir tema: Atividade que compeendeu a definição do tema a ser abordado no projeto. Juntamente com a orientadora, o autor propôs um tema alinhado às demandas da empresa, resultando no título desse trabalho "DASboard: *Dashboard* Centrado em Sistemas Legados de uma Empresa de *Software*";
- Levantamento bibliográfico: Subprocesso que compreendeu outras atividades para pleno conhecimento do arcabouço teórico necessário para a realização desse projeto.

Boa parte desse conhecimento foi relevante para a elaboração dessa monografia como um todo, além de fundamental para escrita do Capítulo 2. Esse subprocesso foi orientado pela Metodologia de Pesquisa. Status: Subprocesso Concluído;

- Estabelecer escopo do trabalho: Atividade que definiu sobre questão de pesquisa, justificativas, e principais objetivos do trabalho, resultando no Capítulo 1. Status: Atividade Concluída;

- Definir referencial teórico: Atividade que compreendeu a elaboração de uma base teórica capaz de embasar esse trabalho como um todo, com destaque para os conceitos de Sistemas Legados, Arquitetura de *Software*, Métricas de *Software*, *Dashboard*, e outros, apresentados no Capítulo 2. Status: Atividade Concluída;

- Definir referencial tecnológico: Atividade que compreendeu a elaboração de uma base tecnológica capaz de apoiar o desenvolvimento desse trabalho como um todo, com destaque para tecnologias *frontend, backend*, banco de dados, métricas de *software*, modelagens e comunicação, apresentadas no Capítulo 3. Status: Atividade Concluída;

- Definir metodologia: Atividade que compreendeu a especificação de aspectos metodológicos para guiar esse trabalho como um todo, com destaque para as seguintes metodologias: Metodologia de Pesquisa, Metodologia de Desenvolvimento, e Metodologia de Análise de Resultados, apresentadas no presente capítulo. Status: Atividade Concluída;

- Realizar estudo de caso: Atividade de relevância para apresentação adequada quanto ao Estudo de Caso da Empresa A, procurando esclarecer sobre os principais problemas observados pelo autor, assim como sobre artefatos específicos (ex. formulários de respostas e entrevistas conferidas junto à equipe de funcionários da empresa), sendo esse relato reportado no Capítulo 5. Status: Atividade Concluída;

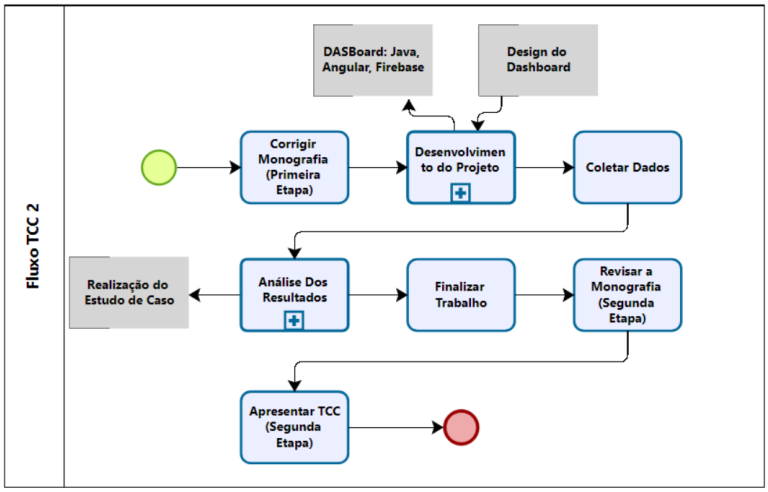
- Elaborar proposta: Atividade necessária para coerente apresentação da proposta desse trabalho, procurando adentrar em protótipos e arquitetura do *dashboard*, em versões ainda preliminares, conforme constam em parte no Capítulo 6. Status: Atividade Concluída, e

- Revisar a monografia (primeira etapa): Atividade que culminou em ajustes na escrita e nos conteúdos dessa monografia, seguindo as revisões conferidas pela orientadora. Status: Atividade Concluída.

As atividades e os subprocessos referentes à segunda etapa do TCC seguem um fluxo, sendo esse ilustrado na Figura 4, na notação BPMN, e descrito a seguir. Novamente,

optou-se por omitir alguns artefatos gerados e consumidos pelas atividades e pelos subprocessos no nível de modelagem, no intuito de concentrar a atenção do leitor na cadência do fluxo em si.

Figura 3 – Fluxo de Atividades - Segunda Etapa do TCC



Fonte: Autor usando Bizagi

- Corrigir monografia (primeira etapa): Atividade que consistiu em prover correções conforme orientações da banca examinadora. Ao final, tem-se a monografia revisada. Status: Atividade Concluída;

- Desenvolvimento do projeto: Subprocesso que compreendeu todas as atividades para pleno planejamento e cumprimento quanto às particularidades de desenvolvimento desse projeto. Esse subprocesso foi orientado pela Metodologia de Desenvolvimento. O resultado é o *Dashboard* em si, sendo esse especificado no Capítulo 6. Status: Subprocesso Concluído;

- Coletar dados: Atividade que consistiu na realização de uma nova coleta de dados referentes à implantação desse projeto na Empresa A. Os resultados são reportados no Capítulo 7. Status: Atividade Concluída;

- Análise dos resultados: Subprocesso que compreendeu outras atividades para pleno planejamento e cumprimento quanto à análise de resultados desse projeto. Esse subprocesso foi orientado pela Metodologia de Análise de Resultados. O *Dashboard* foi implantado na Empresa A, orientando-se pelo protocolo de Estudo de Casos estabelecido no Capítulo 5. Então, os resultados obtidos são reportados Capítulo 7. Status: Subprocesso Concluído;

- Finalizar trabalho: Atividade que contemplou a finalização do projeto, bem como tratamento de quaisquer *Bugs* e problemas reportados e coleta de *Feedback* posterior a implantação do trabalho. Status: Atividade Concluída;

- Revisar a monografia (segunda etapa): Atividade que visou ajustes na escrita e nos conteúdos dessa monografia, seguindo as revisões conferidas pela orientadora. Como resultado, tem-se a presente monografia. Status: Atividade Concluída, e
- Apresentar TCC (segunda etapa): Atividade que compreende a apresentação dos resultados obtidos na segunda etapa do TCC aos membros da banca examinadora. Status: Atividade A Ser Realizada.

4.2 Metodologia de Pesquisa

Segundo GIL (2002), pesquisa pode ser definida como o procedimento que possui o objetivo de gerar respostas aos problemas propostos, sendo esta necessária quando não há informação ou ferramentas suficientes para solucionar tais problemas. Há também a classificação da pesquisa em diferentes aspectos, sendo esses: objetivos, natureza, abordagem e procedimentos. Nos tópicos a seguir, essa classificação é apresentada para o caso da presente pesquisa, embasando-se - em todos os casos - em GIL (2002).

4.2.1 Objetivos

Em relação aos objetivos, a pesquisa é classificada como exploratória (GIL, 2002), realizando investigação em conjunto com a literatura especializada e levantamento bibliográfico para maior familiarização com o conteúdo abordado, além de utilização do método exploratório de estudo de caso para maior entendimento do problema, bem como de possíveis soluções que podem ser aplicadas à situação da Empresa A.

4.2.2 Natureza

A presente pesquisa pode ser classificada, quanto à natureza, como uma pesquisa aplicada. Sendo assim, os insumos gerados ao longo dessa pesquisa foram implantados em um estudo de casos, para aferir de forma mais concreta sobre os reais resultados e as mais relevantes contribuições desse trabalho. Em uma pesquisa aplicada, há participação constante do autor da pesquisa no objeto em estudo, exatamente como ocorre com o autor desse trabalho junto à Empresa A.

4.2.3 Abordagem

Em relação à abordagem de pesquisa, essa pode ser classificada como híbrida, sendo uma pesquisa quantitativa e qualitativa. A utilização de métricas quantificáveis, como o número de *Code smells* identificados no Sistema Legado estudado da Empresa A, assim como o número de *Bugs* encontrados, remete à natureza quantitativa da pesquisa. Por outro lado, a utilização de questionários e entrevistas durante o estudo de caso permite

atribuir à pesquisa um caráter mais qualitativo, uma vez que ocorreram validações com os funcionários da empresa considerando aspectos subjetivos e não quantificáveis, tais como: percepções, satisfações, impressões, dentre outros.

4.2.4 Procedimentos

Em relação aos procedimentos técnicos utilizados para realização da pesquisa, pode-se destacar o uso de levantamento bibliográfico, considerando a literatura para embasamento teórico da solução proposta e orientando-se por práticas e padrões recomendados pela comunidade de *Software*. Há ainda, em termos de procedimento, o uso de Estudo de Caso para investigar o contexto da Empresa A, bem como a situação dos Sistemas Legados encontrados nessa empresa. No Estudo de Caso, foi aplicada a Metodologia de Análise de Resultados.

4.2.5 Um Olhar Mais Aprofundado sobre Pesquisa Exploratória

Pesquisas exploratórias possuem o intuito de prover maior conhecimento sobre o problema, de forma que o desenvolvimento de soluções seja facilitado (GIL, 2002). O principal objetivo de pesquisa exploratória dá-se pela familiarização com o problema, possibilitando, consequentemente, maior variedade de soluções. A maior parte das pesquisas exploratórias envolve:

- Levantamento bibliográfico;
- Entrevistas;
- Estudo de caso, e
- Análise de exemplos que "estimulem a compreensão".

4.2.6 Levantamento Bibliográfico

De acordo com GIL (2002), o levantamento bibliográfico ocorre pela pesquisa desenvolvida com base em literatura especializada já criada e disponibilizada ao público. Corresponde a um dos pilares que guiam as pesquisas exploratórias. Dentre as vantagens, tem-se o fato de que, ao se buscar fontes especializadas, o pesquisador entra em contato com dados já coletados, analisados e explorados previamente, sem a necessidade de maior esforço. Tal vantagem auxilia no seu próprio objeto de estudo, uma vez que os dados já se encontram comprovados. Entretanto, em contrapartida, essa mesma vantagem pode se tornar um ponto negativo. Tudo depende da qualidade da pesquisa utilizada, podendo esta possuir dados inconsistentes ou equivocados. Diante do exposto, é importante a validação das fontes utilizadas durante a realização de pesquisas exploratórias.

4.2.6.1 Buscas Bibliográficas

A realização das buscas bibliográficas que compõem este trabalho permitiu, ao autor desse trabalho e à sua orientadora, o fechamento mais adequadado do tema e de seu escopo, e o conhecimento dos conteúdos constantes em bases científicas de relevância, tais como:

- IEEE;
- ACM - *Association for Computing Machinery*, e
- Scopus.

Ao acordar as bases de conhecimento a serem utilizadas de acordo com a necessidade do tema a ser abordado em cada tópico de interesse, foram então utilizados trechos de busca específicos para levantamento bibliográfico, sendo estes:

- *Software Architeture*;
- *MVC Architecture*;
- *Dashboard*;
- *Legacy Systems*;
- *Software Documentation*;
- *Software Maintenance*, e
- *Software Architeture Document*.

Ademais dos artefatos encontrados utilizando o método de busca apresentado, outras fontes foram encontradas por recomendação da orientadora, assim como algumas fontes eram de conhecimento do autor antes do início da escrita do trabalho.

4.2.7 Estudo de Caso

O estudo de caso é uma forma de pesquisa que investiga detalhada e minuciosamente um ou poucos objetos de estudo (GIL, 2002). O estudo de caso possui diferentes propósitos, entre eles:

- Explorar situações reais nas quais não existam limites definidos para suas existências;
- Preservar o caráter unitário do objeto de estudo;

- Descrever o contexto e a razão em que o estudo está sendo feito;
- Formular hipóteses ou desenvolver teorias sobre o objeto de estudo e a situação abordada, e
- Explicar as causas de determinada situação abordada no estudo de caso onde não há forma de determinar causa via meios comuns de pesquisa.

No contexto deste trabalho, o estudo de caso é especificado no Capítulo 5, onde o objeto de estudo compreende a Empresa A.

4.3 Metodologia de Desenvolvimento

Para o desenvolvimento do presente trabalho, foi utilizada uma metodologia combinada, com o *framework* de gerenciamento de projetos ágeis Scrum e o *framework* visual ágil Kanban (ATLASSIAN, 2023a).

O Scrum é a ferramenta utilizada para condução do processo de desenvolvimento ágil, considerando práticas específicas, tais como: elaboração de *Backlog* do Produto; realização de reuniões diárias, elaboração dos *Backlogs* das *Sprints*, dentre outras. Nesse processo, atividades a serem realizadas são estabelecidas e priorizadas (ATLASSIAN, 2023b).

Já o Kanban consiste em uma ferramenta que permite a visualização facilitada do processo de desenvolvimento ágil adotado pelo Scrum, indicando o status de cada atividade anteriormente definida no Scrum, e fazendo uso de um quadro para gerenciamento e exposição do processo como um todo (ATLASSIAN, 2023a).

O Scrum necessita de um *Backlog* bem definido para ser utilizado com maior eficiência. O *Backlog* do Produto representa uma lista de atividades para o time de desenvolvimento. Esta lista é classificada em diferentes níveis de prioridade e granularidade. O time de desenvolvimento define, então, quais atividades serão desenvolvidas, bem como uma janela de tempo fixa para que ocorra esse desenvolvimento. Essa janela é chamada de *Sprint*, sendo essas iterações contínuas do método Scrum, nas quais a equipe de desenvolvimento trabalha para entregar o que foi definido anteriormente (no início da *Sprint*). Como representa uma janela de tempo curta, é de extrema importância que, desde o *Backlog* do Produto, as atividades estejam bem definidas e priorizadas. Todo esse processo é acompanhado de forma dinâmica e visual via o Quadro Kanban (ATLASSIAN, 2023a).

Para o desenvolvimento do projeto proposto, a aplicação do método híbrido com Scrum e Kanban foi realizada da seguinte forma:

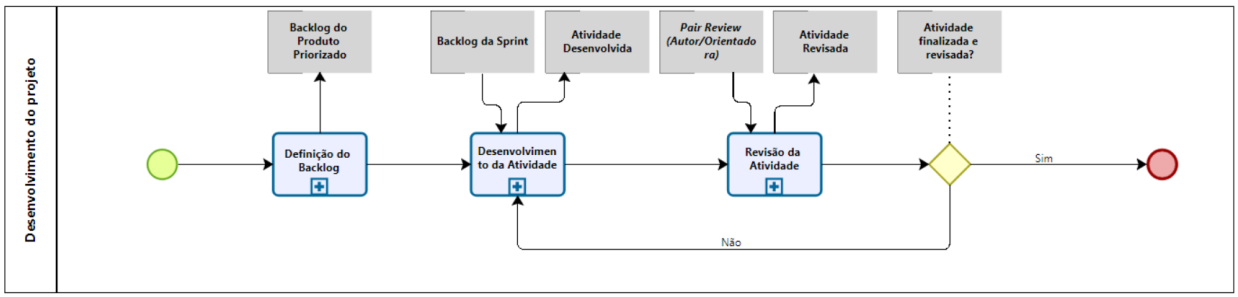
- Definição do *Backlog*: As atividades relacionadas ao desenvolvimento do projeto foram definidas, priorizadas e classificadas antes do início do desenvolvimento, e especificadas como tarefas a serem feitas até o fim do trabalho. Inicialmente, estavam na primeira coluna do Quadro Kanban (A Fazer), e foram movidas de acordo com seu status de desenvolvimento atual;

- Desenvolvimento da Atividade: Cada atividade, uma vez movida no Quadro Kanban, entrando em desenvolvimento (Em Desenvolvimento), deveria ser tratada no decorrer da *Sprint*, pelo desenvolvedor responsável, evitando atrasos. Ao finalizar, há novamente mudança no Quadro Kanban, no qual a atividade foi movida para a coluna de atividades finalizadas, porém não aprovadas (Feito), e

- Revisão da Atividade: Após a finalização da atividade, foi realizada uma revisão para averiguar possíveis erros ou manutenções necessárias. Uma vez concluída essa revisão, tem-se a aprovação, movendo-se a atividade para a coluna final do Quadro Kanban (Finalizada). Caso haja necessidade de alteração, a atividade era então movida, novamente, para o status anterior até que todos os ajustes sejam feitos e uma nova revisão seja realizada.

A Figura 4 demonstra o fluxo geral de atividades que foram realizadas para desenvolvimento do projeto proposto, com implementação do Scrum e do Kanban (adaptados). Cabe ressaltar que as etapas principais do processo foram definidas usando a abstração de Subprocesso da notação BPMN.

Figura 4 – Fluxo de Desenvolvimento do Projeto



Fonte: Autor usando Bizagi

4.4 Metodologia de Análise de Resultados

Para a análise de resultados obtidos no decorrer do trabalho, foi utilizado o Estudo de Caso. A principal vantagem do estudo de caso como metodologia de análise de resultados é a sua capacidade de fornecer uma compreensão aprofundada e detalhada do objeto em estudo. Ele permite que o pesquisador examine o contexto específico, as nuances e as

complexidades do fenômeno em questão, o que pode ser útil para explorar causas, efeitos e processos em profundidade.

Segundo GIL (2002), o Estudo de Caso não possui um consenso acordado em relação à sua estrutura e às etapas a serem seguidas. Porém, com base em outros autores YIN (2001) e STAKE (1995), é possível apontar um conjunto de etapas que pode ser adotado na maioria dos Estudos de Caso. Para o contexto deste trabalho, essas etapas foram:

- Formulação do problema: A formulação do problema foi a etapa inicial que definiu o direcionamento do Estudo de Caso e que tipos de dados seriam coletados e analisados ao fim do Estudo;

- Definição da unidade-caso: A unidade-caso refere-se a um objeto em um contexto definido previamente para o estudo. A unidade-caso deste trabalho pode ser classificada como **intrínseco**, onde o caso constitui o próprio objeto de estudo (STAKE, 1995);

- Determinação do número de casos: Estudos de Caso podem ser sobre um ou vários objetos de estudo. Geralmente, não é feita uma definição dos números de casos *a priori*. Entretanto, como o estudo de caso a ser utilizado é intrínseco, é possível definir um número previamente e incrementar progressivamente caso seja necessário;

- Elaboração do protocolo: O protocolo é a criação do plano, no qual são definidos os procedimentos de coletas de dados escolhidos, tais como: entrevistas e questionários;

- Coleta de dados: Em Estudos de Caso, os dados podem ser coletados via análise de documentos, entrevistas, questionários, artefatos, ou outro. No contexto deste trabalho, esses dados foram coletados via investigação do ambiente e do portfólio da Empresa A; de documentação presente nos projetos, bem como com base na visão dos funcionários, sendo esses entrevistados e questionados acerca dos Sistemas Legados presentes, e

- Avaliação e análise dos dados: Dentre as estratégias e técnicas de análise que foram empregadas para interpretar os dados coletados, incluem-se identificação de padrões; verificação e validação dos dados coletados; comparação dos dados coletados em relação aos padrões de *Software* acordados na comunidade, e análise dos dados contrastando a visão inicial adquirida antes da implantação da solução proposta, com a visão posterior, adquirida depois da implantação, por meio de entrevistas antes e após a implantação.

4.5 Cronogramas

Com base no exposto, foram definidos cronogramas orientados aos fluxos de trabalho estabelecidos para ambas as etapas do TCC, primeira e segunda. As Tabelas 1 e 2 apresentam esses cronogramas.

Tabela 2 – Cronograma de Atividades/Subprocessos - Primeira Etapa do TCC

	ABR/2023	**MAI/2023**	**JUN/2023**	**JUL/2023**
Definir tema	X			
Levantamento Bibliográfico	X			
Estabelecer escopo do trabalho	X			
Definir referencial teórico		X	X	
Definir referencial tecnológico		X	X	
Definir metodologia			X	X
Realizar estudo de caso				X
Elaborar proposta				X
Revisão do trabalho				X
Revisar a monografia (primeira etapa)				X

Tabela 3 – Cronograma de Atividades/Subprocessos - Segunda Etapa do TCC

	MAR/2024	**ABR/2024**	**MAI/2024**	**JUN/2024**	**JUL/2024**
Corrigir monografia (primeira etapa)	X				
Desenvolvimento do projeto	X	X	X	X	
Coletar dados				X	
Análise dos resultados				X	
Finalizar o trabalho					X

Tabela 3 – Continuação

	MAR/2024	**ABR/2024**	**MAI/2024**	**JUN/2024**	**JUL/2024**
Revisar a monografia (segunda etapa)					X
Apresentar TCC (segunda etapa)					X

4.6 Resumo do Capítulo

O capítulo apresentou as metodologias que guiaram a condução do trabalho. Primeiramente, há a exposição do fluxo do trabalho, desde a definição do tema até a apresentação à banca examinadora. Isso foi feito tanto para o escopo da primeira etapa, quanto para o escopo da segunda etapa do TCC. Após isso, a pesquisa foi classificada, sendo: exploratória nos objetivos; aplicada na natureza; qualitativa e quantitativa na abordagem, e envolvendo Pesquisa Bibliográfica e Estudo de Casos como procedimentos. Ocorreu ainda o detalhamento de metodologias específicas, dentre elas: Investigativa, com levantamento bibliográfico; de desenvolvimento, com a combinação adaptada de Scrum com Kanban, e de análise de resultados, com protocolo de Estudo de Casos. Por fim, tem-se os cronogramas, que conferem uma visão temporal acerca de ambas as etapas inerentes ao TCC (i.e. primeira e segunda etapas).

5 Estudo de Caso

Nesse capítulo, há a definição e a especificação do Estudo de Caso da Empresa A. Esse estudo auxiliou o trabalho, que permitiu a implantação da solução proposta em um caso real, ou seja, uma empresa de software que contém, em seu portfólio, Sistemas Legados relevantes. O capítulo orienta-se por um protocolo de Estudo de Casos. Esse protocolo estabelece etapas, sendo essas as norteadoras das seções desse capítulo. Inicialmente, ocorre a Formulação do Problema, onde há a definição do contexto do estudo. Depois, tem-se a Definição de Casos, etapa reponsável pela definição da unidade-caso a ser investigada, além da determinação do número de casos que serão tratados. Na sequência, encontra-se a Elaboração do Protocolo, onde foram definidos os métodos de coleta de dados e a finalidade de cada um deles. Como última etapa, há a Coleta de Dados, na qual os métodos definidos na etapa anterior serão utilizados, incorrendo no levantamento de insumos que embasaram o desenvolvimento do trabalho como um todo. Por fim, tem-se o Resumo do Capítulo.

5.1 Formulação do Problema

De acordo com GIL (2002), o estudo de caso começa pela formulação do problema que será observado e investigado durante a pesquisa. Trata-se de definir a situação ou fenômeno que seja possível de delinear causa e soluções para esta situação no decorrer da investigação. No contexto deste trabalho, o problema definido dá-se pela presença de Sistemas Legados presentes que são fundamentais para as atividades diárias da Empresa A. Sistemas esses que não possuem documentação adequada, nem estrutura arquitetural bem definida, resultando em problemas para a equipe de desenvolvimento referentes a desenvolvimento de novas funcionalidades, integração com outros sistemas, e manutenibilidade. Portanto, parte-se de evidências de que a Empresa A tem Sistemas Legados, e que esses não atendem às expectativas e aos anseios dos envolvidos na empresa (sejam desenvolvedores ou superiores na gestão).

5.2 Definição de Casos

Esta etapa do Estudo de Caso trata a definição da unidade-caso de estudo, assim como a determinação do número de casos a serem estudados. A unidade-caso refere-se a um objeto em um contexto pré-definido (GIL, 2002). A delimitação desta unidade-caso não é uma tarefa simples, sendo necessário estabelecer os limites do que será investigado. Um problema comum durante Estudos de Caso ocorre pelo fato dessa etapa resultar em dois

extremos: (i) uma delimitação muito abrangente que atrapalha a coleta de dados devido à grande variabilidade de casos, aumentando muito o escopo da pesquisa e deixando o pesquisador com muitas tarefas para conciliar, ou (ii) uma delimitação muito pequena, não permitindo uma coleta relevante de dados para análise. Neste trabalho, a unidade-caso foi definida como uma equipe de desenvolvimento específica da Empresa A. Lembrando que se trata de uma empresa de médio porte, com um quadro de 2.000 funcionários, e uma grande variedade de sistemas. Alguns desses sistemas são desenvolvidos pela equipe designada como Equipe B, que será referenciada ao longo deste trabalho.

Após a definição da unidade-caso, é necessário definir quantos casos serão estudados. A determinação do número de casos é dada considerando o contexto de pesquisa, assim como as limitações de pesquisa em termos de capacidade e tempo do pesquisador (GIL, 2002). Há uma tendência maior a utilizar um grande número de casos, quando há métodos mais refinados de pesquisa e maior disponibilidade de tempo para coleta e análise de dados. Para este trabalho, a pesquisa realizada com a Equipe B foi focada na análise de um Sistema Legado em específico, doravante referido como Sistema Legado X. Entretanto, haverá menção a outros sistemas utilizados pela Equipe B, conferindo maior apoio ao processo de investigação.

5.3 Elaboração do Protocolo

A etapa de elaboração de protocolo de um Estudo de Caso faz-se necessária, pois é a etapa que define os métodos de coleta de dados a serem utilizados, além de ser responsável por definir como esses métodos serão coletados e sua finalidade (GIL, 2002). Para este trabalho, primeiramente, foi relevante definir quais os principais dados eram apropriados ao desenvolvimento do projeto proposto. Adicionalmente, como esses dados seriam utilizados para estabelecer o ambiente composto pela Equipe B e o Sistema Legado X. Nesse sentido, foram utilizados os seguintes métodos para coleta de dados:

- Entrevista: Realizou-se uma entrevista com dois desenvolvedores da Equipe B, que possuem correlacões com o Sistema Legado X, bem como compreendem sobre suas utilizações nas atividades diárias da Empresa A. A ideia é levantar problemas comuns do Sistema Legado X, procurando entender um pouco mais sobre esse sistema na perspectiva de ambos os participantes, e com olhar orientado às demandas arquiteturais;

- Formulário: Para todos os integrantes da Equipe B, disponibilizou-se um formulário no Google Forms (GOOGLE, 2023b), no intuito de coletar dados de forma mais abrangente, e conhecer as percepções da equipe como um todo sobre o portfólio de sistemas da Empresa A, em especial, sobre o Sistema Legado X, e

- Análise no Sonar: Realizou-se uma análise no SonarQube (SONARQUBE, 2023), tomando como referência o Sistema Legado X, e visando identificar e conferir insumos mais concretos sobre os problemas presentes nesse sistema.

5.4 Coleta de Dados

Nesta etapa, realizou-se a coleta dos dados, orientando-se pelo protocolo previamente estabelecido. Ressalta-se sobre a pertinência de se usar diferentes formas de coleta de dados em uma pesquisa de Estudo de Caso, principalmente, considerando diferentes métodos de aquisição. Isso aumenta a qualidade dos resultados obtidos (YIN, 2001). Segue sobre a coleta realizada guiando-se pelos diferentes métodos propostos.

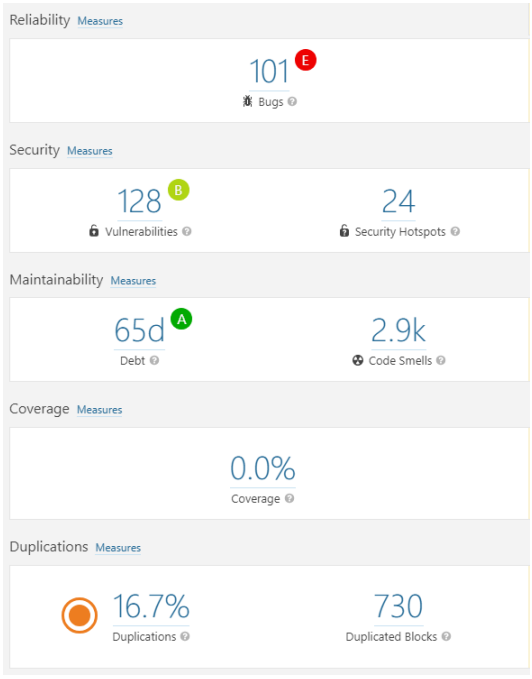
5.4.1 Análise no SonarQube

Para uma melhor demonstração do Sistema Legado X e dos problemas identificados, foi realizada uma análise no SonarQube (SONARQUBE, 2023) utilizando o Sistema Legado X. Trata-se de um Sistema Legado da Empresa A, com mais de dez anos, possuindo *Backend* e *Frontend*. Faz uso de Java e Jakarta Server Pages no *Backend* e no *Frontend*, respectivamente. Possui cerca de cinquenta e três mil linhas de código, além de depender de um grande número de bibliotecas, sejam essas externas à Empresa A, sejam bibliotecas gerais desenvolvidas pela própria Empresa A para integrar o Sistema Legado X com outros sistemas de interesse. Segue a análise feita no SonarQube (Figura 5), e breves explicações dos problemas observados.

Na Figura 5, é possível notar que há um grande número de problemas encontrados pelo SonarQube no Sistema Legado X, sendo apontados cento e um *bugs* observados, e que necessitam de correções. Além desses *bugs*, há muitas vulnerabilidades observadas no código. Estas não são tão graves quanto os *bugs*, uma vez que o SonarQube indicou a nota "B", a qual informa que, apesar de existirem vulnerabilidades, não há vulnerabilidade crítica o suficiente que necessite de correção imediata. Porém, ainda são problemas que necessitam de atenção.

Há considerável ocorrência de *code smells*, resultantes de anos de manutenções e evoluções de código sem documentação apropriada para guiar a padronização e o desenvolvimento do *Software*. Observa-se ainda que não há cobertura de testes alguma no Sistema Legado X, além de muitas linhas duplicadas. Essa duplicação implica em um sistema com mais linhas de código do que o necessário. Acredita-se que tais problemas podem ser justificados, em sua grande maioria, pela falta de documentação adequada, e por evoluções sem critérios na arquitetura, dado que o Sistema Legado X é resultado do esforço de dezenas de desenvolvedores ao longo dos anos de existência do sistema. Essa suposição é - de certa forma - confirmada pelo autor, que faz parte do corpo de funcioná-

Figura 5 – Análise SonarQube



Fonte: SonarQube

rios da empresa, tendo que lidar frequentemente com a pouca e obsoleta documentação do Sistema Legado X. Corrobrando ainda mais com essa premissa, há os reportes dos entrevistados, que apontam para conclusões semelhantes.

5.4.2 Formulário

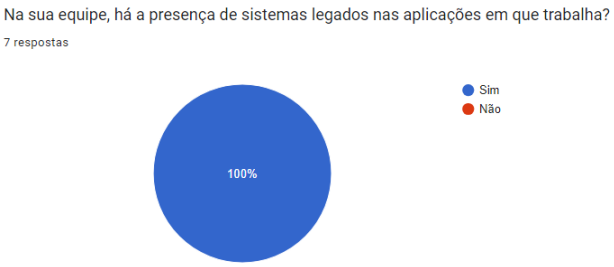
Para a coleta de informações acerca da Empresa A e seus Sistemas Legados utilizados na Equipe B, foi disponibilizado um formulário no Google Forms a todos os desenvolvedores da equipe. Esse formulário ficou disponível por um período de sete dias para respostas, além de oferecer um Termo de Consentimento Livre e Esclarecido (Apêndice A).

O formulário continha um total de sete perguntas realizadas e preenchidas pelos desenvolvedores. Para garantir validade e confiabilidade ao formulário, o acesso foi concedido apenas a pessoas da equipe que estão envolvidas diariamente no desenvolvimento e na manutenção do Sistema Legado X e associados. O número de respostas variou por pergunta, com predomínio de 7 respostas por pergunta. A escolha do Google Forms ocorreu pela facilidade de preenchimento das respostas por parte dos participantes, assim como a simples visualização das respostas obtidas pelo autor. Esse processo facilitou a coleta e a análise dos dados. Para conhecimento, seguem as perguntas realizadas e suas respectivas respostas.

A primeira pergunta refere-se à identificação de presença de Sistemas Legados na equipe em que o participante atua na Empresa A. Como retornos, constam 7 respostas.

Todos afirmam sobre a existência de sistemas legados, conforme ilustrado na Figura 6.

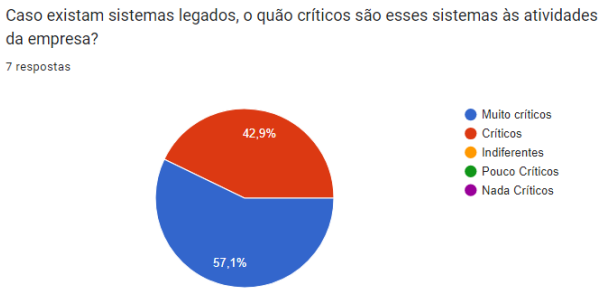
Figura 6 – Pergunta Formulário 1



Fonte: Autor usando Google Forms

A segunda pergunta possui o intuito de conhecer sobre a criticidade dos Sistemas Legados para as atividades diárias da Empresa A. Como retornos, constam 7 respostas. Em uma escala de cinco níveis de concordância, indo de "Muito Críticos"à "Nada Críticos", 57,1% dos respondentes (maioria) reconheceram como "Muito Críticos", e 42,9% dos respondentes reconheceram como "Críticos", conforme consta na Figura 7. Isso confere pertinência à condução do presente trabalho, uma vez que o mesmo procura mitigar alguns problemas comumente encontrados em Sistemas Legados.

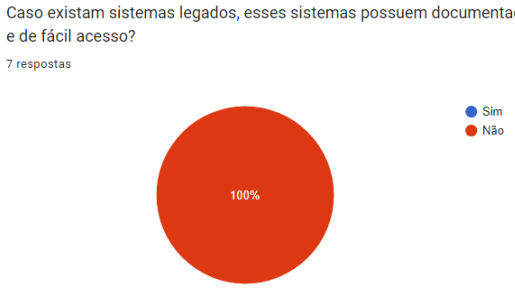
Figura 7 – Pergunta Formulário 2



Fonte: Autor usando Google Forms

A terceira pergunta objetiva saber se, na existência de Sistemas Legados, à documentação adequada e de fácil acesso. Como retornos, constam 7 respostas. Todos os respondentes acordaram sobre a falta de documentação adequada nos Sistemas Legados (vide Figura 8), confirmando uma suposição do autor, exposta anteriormente nessa monografia, com base na literatura especializada.

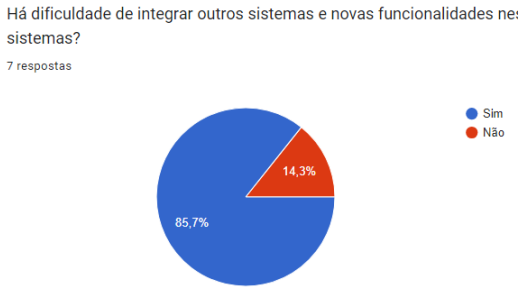
Figura 8 – Pergunta Formulário 3



Fonte: Autor usando Google Forms

A quarta pergunta aborda problemáticas de relevância expostas neste trabalho, sendo essas a dificuldade de integração dos Sistemas Legados com outros sistemas (mais atuais), bem como a dificuldade de se incorporar novas funcionalidades nesses Sistemas Legados. Como retornos, constam 7 respostas. 85.7% dos respondentes (maioria significativa da amostra) reconheceram como "Sim", e apenas 14,3% dos respondentes reconheceram como "Não", conforme ilustrado na Figura 9. Isso reforça ainda mais a pertinência quanto à condução do presente trabalho.

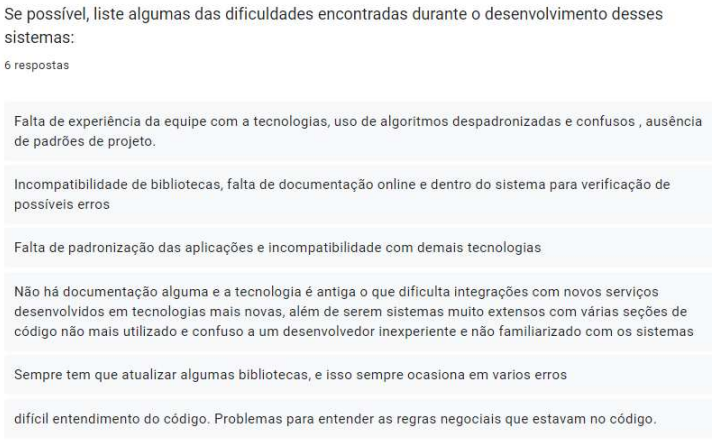
Figura 9 – Pergunta Formulário 4



Fonte: Autor usando Google Forms

A quinta pergunta confere grau de liberdade aos respondentes, sendo discursiva, e permitindo que os participantes exponham suas impressões sobre as dificuldades encontradas durante o desenvolvimento (ou uso) desses Sistemas Legados. Constam 6 retornos (vide Figura 10). Ao analisar as colocações, encontra-se total correspondência das respostas com os típicos problemas reportados na literatura, e já documentados nessa monografia em capítulos anteriores, com destaque para: código despadronizado e confuso; ausência de padrões de projeto; incompatibilidade de bibliotecas e tecnologias; falta de documentação; documentação legada; dificuldade de integração com novos serviços; sistemas extensos; código morto; necessidade de atualizações constantes, e código pouco compreensível, dentre outros.

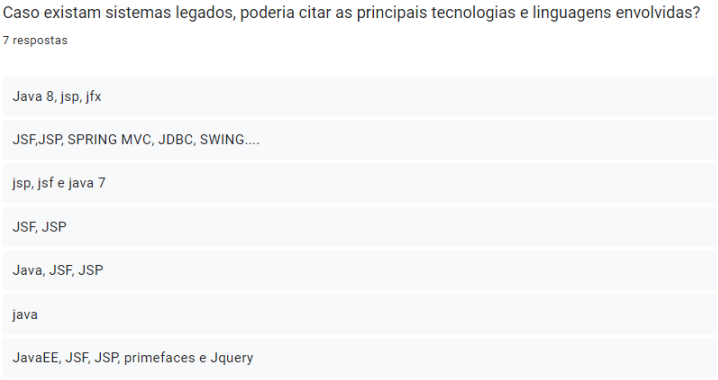
Figura 10 – Pergunta Formulário 5



Fonte: Autor usando Google Forms

A sexta pergunta, novamente, confere grau de liberdade aos respondentes, sendo discursiva, mas a intenção foi saber sobre as principais tecnologias e linguagens utilizadas nesses Sistemas Legados pela Equipe B. Constam 7 retornos (vide Figura 11). Ao analisar as colocações, há predomínio de tecnologias associadas à comunidade da linguagem Java e afins.

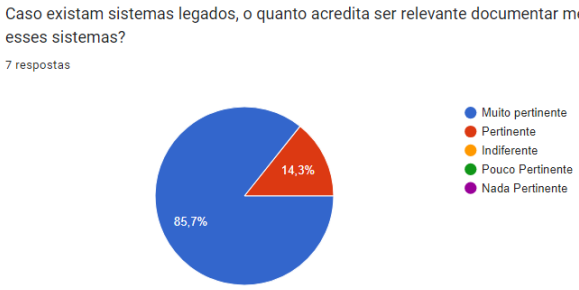
Figura 11 – Pergunta Formulário 6



Fonte: Autor usando Google Forms

Por fim, a sétima pergunta possui o intuito de saber a opinião dos participantes sobre a importância de se documentar melhor esses Sistemas Legados. Como retornos, constam 7 respostas. Em uma escala de cinco níveis de concordância, indo de "Muito Pertinente"à "Nada Pertinente", 85,7% dos respondentes (maioria representativa da amostra) reconheceram como "Muito Pertinente", e apenas 14,3% dos respondentes reconheceram como "Pertinente", conforme ilustrado na Figura 12. Conclui-se que documentação em Sistemas Legados é algo desejado na percepção dos participantes.

Figura 12 – Pergunta Formulário 7



Fonte: Autor usando Google Forms

Resumidamente, os retornos conferidos via formulário demonstram o ambiente da Equipe B, considerando os Sistemas Legados da Empresa A. Evidencia a necessidade e o desejo dos desenvolvedores em obterem documentação adequada capaz de auxiliá-los em suas atividades diárias (dependentes dos Sistemas Legados). Além disso, há exemplificação de problemas comuns em Sistemas Legados, sendo esses referentes à não padronização, e à dificuldade de compreensão de regras negociais decorrentes da falta de registro dessas regras, o que incorre em desenvolvedores inexperientes na equipe enfrentando dificuldades ao realizar manutenções.

5.4.3 Entrevista

Para coleta de dados acerca dos Sistemas Legados da Empresa A, foi conduzida uma entrevista com o propósito de investigar com mais precisão o Sistema Legado X da Equipe B. A entrevista foi elaborada com onze perguntas. Essas perguntas, além de definirem com precisão o perfil do entrevistado, também possuíam o intuito de demonstrar potenciais problemas encontrados no Sistema Legado X relacionados à documentação (ou falta dela), e aos padrões de projeto (que, se negligenciados ou mal utilizados, remetem aos problemas arquieturais de difícil tratamento).

A Entrevista possibilita um olhar mais aprofundado sobre o Sistema Legado X. Portanto, a seleção dos entrevistados é importante para abranger diferentes visões do sistema. Para esta Entrevista, os selecionados foram dois Analistas de maior senioridade no campo de *Software*; porém, com grande diferença de experiência na Empresa A. Um dos selecionados, o primeiro entrevistado, possui 10 meses de experiência na Empresa A. Já o segundo entrevistado possui 10 anos de experiência na Empresa A. A seleção foi feita visando a obtenção de respostas de similar experiência técnica. Entretanto, com diferentes graus de vivência neste ambiente em específico. Seguem as Entrevistas realizadas acerca do Sistema Legado X.

A primeira entrevista conduzida foi realizada com o Analista mais novo na Equipe B. Primeiramente, foi realizada uma pergunta para assegurar sobre o consentimento do

entrevistado acerca do objetivo da pesquisa e sua natureza: "Antes de prosseguirmos com o questionário, gostaríamos de esclarecer que esta entrevista é puramente acadêmica e confidencial. Suas respostas serão tratadas de forma anônima, sem a divulgação do seu nome ou da empresa em que você trabalha. Você está ciente e consente com essa condição de confidencialidade?"Após a validação por parte do entrevistado, a entrevista prosseguiu da seguinte forma:

Entrevistado A

Entrevistador: Autor Gabriel Alves Hussein

Pergunta 1:

Há quanto tempo você trabalha na Empresa A?

Resposta: 11 meses.

Pergunta 2:

Qual é a sua função ou cargo atual na empresa?

Resposta: Analista de Sistemas.

Pergunta 3:

Você tem experiência de trabalho com o sistema X da Empresa A? Se sim, por quanto tempo?

Resposta: Sim, desde que iniciei, 11 meses.

Pergunta 4:

Como você descreveria o estado atual do sistema X da Empresa A?

Resposta: É um Sistema Legado bem antigo com bastante problemas.

Pergunta 5:

O sistema X possui uma documentação abrangente e atualizada? Se sim, como você a acessa? Se não, como você lida com essa falta de documentação?

Resposta: Não possuímos documentação para os sistemas legados, logo é necessário estudo do código de maneira constante; porém nem sempre consegue-se entender todas as funcionalidades e regras negociais aplicadas.

Pergunta 6:

Quais são as principais dificuldades que você enfrenta ao lidar com o sistema X da Empresa A no seu trabalho diário?

Resposta: Falta de documentação, centralização de informação em uma só pessoa da equipe e falta de padronização de código.

Pergunta 7:

Você acredita que a falta de padrões de *software* adequados afeta a manutenção e a evolução do sistema X? Se sim, de que forma?

Resposta: Com certeza, pois se tratando de sistemas legados se torna muito onerosa sua manutenção, uma vez que, não existe nem padrão de código/arquitetura definidas. Determinados sistemas, incluindo o Sistema X, cresceram de forma exponencial e sem padrão, logo são necessárias muitas voltas para se chegar em uma implementação simples.

Pergunta 8:

Existe algum processo ou método específico que você utiliza para entender ou modificar o código do sistema X? Se sim, poderia descrever brevemente como é esse processo?

Resposta: Leitura do código, execução em debug e tentativa/erro, infelizmente.

Pergunta 9:

Como você se mantém atualizado sobre as mudanças e atualizações realizadas no sistema X? Existe alguma forma de anunciar mudanças para todos os membros da equipe?

Resposta: Somente via quadro de atividades, sendo o único histórico sobre as atividades.

Pergunta 10:

Você acredita que melhorias na documentação e nos padrões de *software* do sistema X poderiam ajudar a resolver problemas enfrentados durante o desenvolvimento? Por quê?

Resposta: A documentação da parte negocial seria de grande valia para melhoria da manutenção do sistema, pois o Sistema X é muito grande e tem regras específicas que nem sempre são detectadas somente com a leitura do código.

Pergunta 11:

Na sua opinião, quais seriam as principais ações que a Empresa A poderia tomar para melhorar o estado do sistema X e a sua documentação?

Resposta: Documentar inicialmente a parte negocial e pontos críticos do sistema, disponibilizar wiki acessível a todos os membros e criação de cultura de documentação por parte das áreas acima da nossa que possa ser aplicada a outras equipes também.

É possível observar na entrevista as dificuldades enfrentadas por um desenvolvedor com uma grande experiência técnica no campo de *Software*, porém com menor vivência em se tratando da Empresa A e do Sistema Legado X em específico. Há um grande foco do entrevistado em expor a falta de padronização no sistema, assim como a falta de documentação impedir a fluidez do desenvolvimento de novas funcionalidades.

Após a primeira entrevista, foi realizada uma segunda entrevista, com outro analista da Equipe B. Porém, este analista possui uma maior vivência na Empresa A, assim como com o Sistema Legado X.

Segundo entrevistado B

Entrevistador: Autor Gabriel Alves Hussein.

Pergunta 1:

Há quanto tempo você trabalha na Empresa A?

Resposta: Trabalho desde 02/05/2011, 12 anos.

Pergunta 2:

Qual é a sua função ou cargo atual na empresa?

Resposta: Analista de Programação Pleno - Líder Técnico.

Pergunta 3:

Você tem experiência de trabalho com o sistema X da Empresa A? Se sim, por quanto tempo?

Resposta: Sim, 10 anos.

Pergunta 4:

Como você descreveria o estado atual do sistema X da Empresa A?

Resposta: Sistema desenvolvido desde 2004 com tecnologia Java 1.4 atualmente 1.8, JSP. Em questão de padrões de projeto até que possuia boa padronização de arquitetura inicialmente, porém não acompanhou o avanço das tecnologias necessitando ser refatorado ao longo dos anos.

Pergunta 5:

O sistema X possui uma documentação abrangente e atualizada? Se sim, como você a acessa? Se não, como você lida com essa falta de documentação?

Resposta: Não, não possui documentação, se teve, se perdeu ao longo do tempo, se tem documentação, desconheço. Não lido. O que sei do sistema foi adquirido ao longo do tempo que trabalho com o sistema.

Pergunta 6:

Quais são as principais dificuldades que você enfrenta ao lidar com o sistema X da Empresa A no seu trabalho diário?

Resposta: Principalmente falta de documentação, fazendo com que eu tenha que explicar as mesmas coisas para cada novo desenvolvedor da equipe ao longo dos anos. Por ser antigo também há limitações tecnológicas.

Pergunta 7:

Você acredita que a falta de padrões de *software* adequados afeta a manutenção e a evolução do sistema X? Se sim, de que forma?

Resposta: Sim, as demandas chegam atropeladas na área e sem discussão adequada acerca do caminho a ser seguido, dessa forma além da ausência da documentação, faz que novos desenvolvedores que não conhecem o design *do sistema e desenvolvem da forma que acreditam ser adequada e não mantém um padrão consistente no projeto.*

Pergunta 8:

Existe algum processo ou método específico que você utiliza para entender ou modificar o código do sistema X? Se sim, poderia descrever brevemente como é esse processo?

Resposta: Seguimos o padrão de projeto Command para a maior parte da estrutura do Sistema X onde cada nova página necessita de uma Command de carregamento e outra de processamento, além de criar uma página JSP que age como o visual, para modificar utilizo do conhecimento adquirido ao longo dos anos.

Pergunta 9:

Como você se mantém atualizado sobre as mudanças e atualizações realizadas no sistema X? Existe alguma forma de anunciar mudanças para todos os membros da equipe?

Resposta: Normalmente quando se utiliza alguma tecnologia nova ou alguma técnica nova procura-se apresentar para todos da equipe, como um workshop*, porém fica cada vez mais difícil de realizar por conta dos incidentes e demandas urgentes do dia a dia, porém isto não é documentado.*

Pergunta 10:

Você acredita que melhorias na documentação e nos padrões de *software* do sistema X poderiam ajudar a resolver problemas enfrentados durante o desenvolvimento? Por quê?

Resposta: Resolver talvez não resolveria, mas diminuiria muito o gap *de conhecimento acerca do Sistema X, além de diminuir a dependência de passagem de informação entre desenvolvedores mais antigos para os desenvolvedores mais novos.*

Pergunta 11:

Na sua opinião, quais seriam as principais ações que a Empresa A poderia tomar para melhorar o estado do sistema X e a sua documentação?

Resposta: A documentação acredito que seja uma iniciativa que deveria estar incorporada intimamente no processo de desenvolvimento da equipe, onde essa etapa bem

como o esforço deveria ser considerado para realizar a entrega, mas essa etapa sempre é negligenciada e sacrificada para entregar o produto dentro do prazo que o cliente espera. Já houve iniciativas anteriores de documentar porém iniciar uma documentação não é tão difícil, o problema se dá pela manutenção e acompanhamento. Revisitar documentos e atualizá-los para ficarem condizentes com a situação atual do sistema é o mais difícil.

Nesta segunda entrevista, é possível observar que há uma maior facilidade em lidar com o Sistema Legado X do que para o caso do primeiro entrevistado. Isso se deve ao fato, desse segundo entrevistado já estar mais habituado ao ambiente da Empresa A. Porém, o problema dá-se pelo tempo perdido com demandas confusas sem documentação, e com a demora na passagem de conhecimento acerca das regras negociais e da estrutura geral do projeto para desenvolvedores mais novatos na Equipe B.

Com as entrevistas realizadas, são evidenciados os maiores problemas acerca do Sistema Legado X, que se dão pela não padronização ao longo dos anos, e falta de documentação acerca das funcionalidades e regras negociais presentes no sistema. Além disso, há difícil ambientação de novos membros da Equipe B, onde é necessário que desenvolvedores experientes com o sistema realizem constantes treinamentos para entendimento da aplicação e suas particularidades.

5.5 Resumo do Capítulo

O capítulo apresentou a definição de como é realizado o Estudo de Caso acerca da Empresa A. Há formulação do problema, onde foi apresentado que os Sistemas Legados da Empresa A são relevantes para as atividades diárias da empresa; porém, esses sistemas possuem problemas sérios acerca da falta de padronização arquitetural e da falta de documentação. Após isso, tem-se a definição da unidade-caso a ser investigada, assim como a determinação do número de casos a serem estudados. Sendo assim, foi apresentada a Equipe B e o Sistema Legado X, sendo esse o foco do Estudo de Caso. Também foi elaborado o protocolo do Estudo de Caso, onde são definidos os métodos de coletas de dados realizados durante o estudo. Aqui, optou-se pelo uso de formulários, entrevistas e análises do Sistema Legado X por ferramentas especializadas. Logo após a elaboração desses métodos, os mesmos foram utilizados para realizar a coleta de dados de fato, provendo vários dados, os quais auxiliaram na realização do trabalho como um todo.

6 DASboard

Esse capítulo possui o intuito de apresentar, mais detalhadamente, o foco desse trabalho. Nesse sentido, confere-se, inicialmente, um Contexto descrevendo a origem da ideia; a razão de ter sido escolhida; e como foi observado o problema que foi objeto de estudo ao longo desse trabalho. Retomam-se conceitos já abordados para melhor ambientação do leitor. Seguindo, há a fase de Desenvolvimento, sendo essa uma seção para conferir de forma mais clara as fases de desenvolvimento e desafios encontrados desde o período de idealização e prototipação até a finalização do desenvolvimento da aplicação. A seção comporta, dentre outras informações, a descrição de visões preliminares da aplicação, e como essas visões foram se alterando para se adequar ao estudo de caso e suas particularidades, evidenciando sobre suas principais *features*. Depois, será apresentada a Aplicação DASboard, onde se tem detalhamento sobre as telas do **DASboard**, com o propósito de prover uma noção mais clara e concreta sobre a aplicação. Aqui, constam aspectos que foram levantados com a coleta de dados junto aos membros da Equipe B da Empresa A sobre o Sistema Legado, conforme apresentado no Capítulo 5. Portanto, há detalhamento adicional acerca da arquitetura, além da reiteração sobre as tecnologias que foram utilizadas durante o desenvolvimento, em concordância com o perfil do Sistema Legado X e a cultura da Empresa A. Por fim, tem-se o Resumo do Capítulo.

6.1 Contexto

Conforme reportado anteriormente, o autor deste trabalho está inserido como desenvolvedor na Empresa A. Sendo assim, foi possível observar as dificuldades diárias enfrentadas pelos desenvolvedores em relação aos Sistemas Legados presentes na empresa. Isso ocorreu não apenas para o escopo da equipe em que o autor atua, mas também no âmbito geral da empresa. Com as crescentes necessidades de adições de funcionalidades aos Sistemas Legados, foi presenciado um grande gargalo no desenvolvimento desses sistemas, os quais demandam expansão/evolução contínua. Há falta de padronização; falta de documentação; dificuldade de integração com outros sistemas; complexidade percebida pela equipe quanto à compreensão dos sistemas, dentre outras não conformidades, chegando a impactar outras frentes de atuação da empresa. Essas frentes compreendem o uso de sistemas mais modernos, mas dependentes de várias funcionalidades que são providas pelos Sistemas Legados. Há ainda problemas de desempenho, pois são sistemas abrangentes, que se relacionam com vários outros, armazenando e processando grande quantidade de dados.

Aumentando continuamente a necessidade de maior agilidade nas entregas deman-

dadas para a Empresa A, e com problemas relacionados às funcionalidades dos Sistemas Legados acontecendo diariamente, observou-se, rapidamente, uma tendência de maiores gargalos. Além disso, há claro acúmulo de dificuldades, desde à impossibilidade de incorporar novas regras negociais, até atrasos nas entregas, e entregas cada vez mais trabalhosas à equipe.

Antes que esses problemas se tornem insustentáveis, no futuro próximo, o autor pensou em contribuir com esse cenário. Conforme levantamento realizado junto à equipe da Empresa A, detalhado no Capítulo 5, percebe-se o desejo dos desenvolvedores da Equipe B em termos, principalmente, de: documentação de regras negociais dos Sistemas Legados; arquitetura mais aderente aos padrões de projeto e outros facilitadores (ex. ausência de duplicação de código e código morto), e guias relacionados à configuração e à instalação dos Sistemas Legados para novos desenvolvedores. Entretanto, há evidente falta de incentivos e até mesmo de tempo para que essas atividades possam ser realizadas.

Diante do exposto, esse trabalho visou o acompanhamento facilitado de Sistemas Legados da Empresa A, procurando mitigar problemas como falta de documentação e dificuldades na compreensão das funcionalidades e regras negociais desses sistemas. Tem-se como solução um suporte semiautomatizado e orientado a recursos visuais, com facilidade de uso e aderência à cultura da empresa. Esse suporte, desenvolvido como *Dashboard*, tem foco no Documento de Arquitetura, em métricas e análises conferidas por ferramentas especializadas (ex. SonarQube), acompanhamento de funcionalidades do sistema e boas práticas de usabilidade.

Para a análise dos resultados dessa aplicação, utilizou-se o método de Estudo de Caso, conforme consta exposto no Capítulo de Metodologia, replicando os passos inerentes a cada etapa estabelecida pelo protocolo de Estudo de Caso para validar a solução proposta. Nesse caso, ocorreu uma análise comparativa reportada, procurando apresentar sobre as percepções da Equipe B antes e depois da implantação da presente proposta.

6.2 Desenvolvimento

O desenvolvimento da aplicação, que é objeto de estudo deste trabalho, o *Dashboard*, foi realizado visando em apresentar aos envolvidos, ou seja, aos integrantes da Equipe B, uma forma de acompanhar os Sistemas Legados, evidenciando problemas observados e acordando maior responsabilidade em termos de documentação desses sistemas, além de prover visualização para funcionalidades do sistema cadastradas pelos desenvolvedores.

Essa solução é orientada aos referenciais descritos e apontados nos Capítulos de Referencial Teórico e Referencial Tecnológico. Cabe destacar que o *Dashboard* seguiu o padrão arquitetural MVC, implementado usando Java e Angular, além do Firebase para as necessidades de armazenamento.

A proposta inicial contava que a aplicação Web, o **DASboard**, teria *features* distintas, que abordariam o mesmo propósito, sendo este o de informar ao usuário sobre atividades referentes à evolução de Sistemas Legados. Os dados exibidos pelo *Dashboard*, no âmbito documental, manteriam fundamentação na literatura especializada e na cultura da Empresa A. Adicionalmente, a visualização no **DASboard** deveria ser facilitada, evitando a exibição de complexidade sem necessidade; e prezando por uma maior dinamicidade na navegação, na busca por dados, e no uso da aplicação em geral. Para tanto, foi proposto que seriam utilizadas boas práticas de usabilidade e afins (NNG, 2012).

Entre as *features* inicialmente abordadas, podem ser destacadas:

- Clareza sobre as visões arquiteturais do sistema analisado, tratando de evidenciar quais visões estão documentadas, e os status atuais delas;
- Apresentação visual da análise das métricas estabelecidas para o código ao longo do tempo (ex. *code smells, bugs*), e
- Demonstração das regras negociais e funcionalidades documentadas, além de prover visualização para esses documentos.

6.2.1 Protótipos e *Design* de Tela DASboard

O **DASboard** é uma aplicação que possibilita o acompanhamento e a visualização de artefatos de documentação e análise de código em tempo real, para desenvolvedores da Equipe B na Empresa A. A seguir, constam apresentados protótipos iniciais do *Dashboard*, bem como dados e métricas que foram propostos para ele. Para ambientação e adequado desenvolvimento do *Dashboard*, foram desenvolvidas diferentes telas em contato direto com a Equipe B para atender necessidades e preferências em relação à utilização da aplicação, fazendo-se necessários os métodos apresentados no Capítulo de Referencial Teórico, seguindo o fluxo de *Design* Iterativo, *Design* Paralelo e, por fim, Teste Competitivo (NNG, 2011).

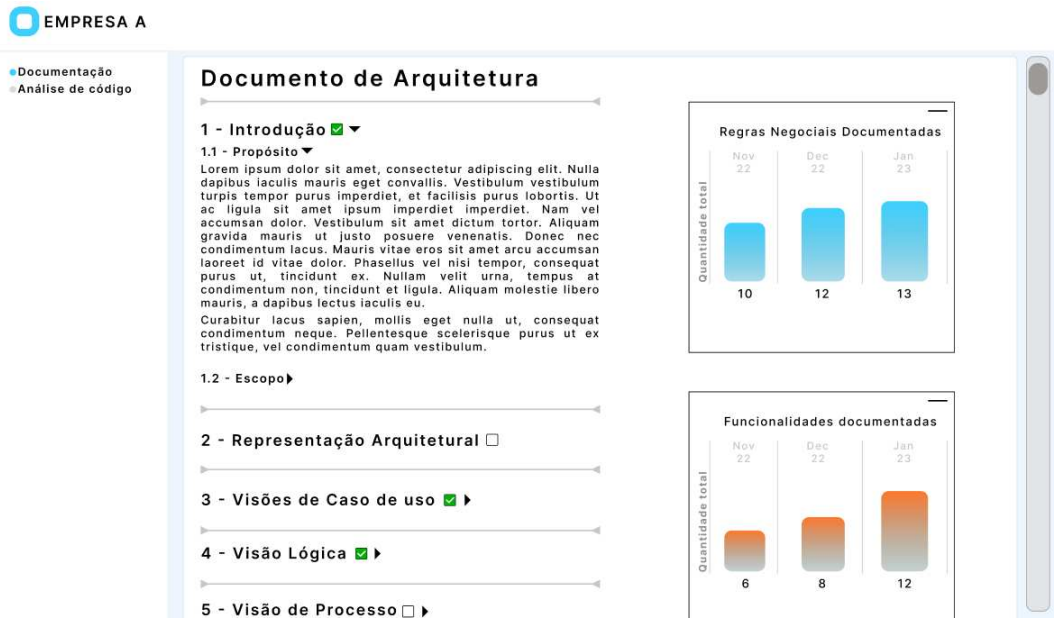
Inicialmente foram propostas duas telas para análise das métricas: a primeira tela, exposta na Figura 13, demonstra duas partes referentes à documentação, e a segunda tela, exposta na Figura 14, demonstra em tempo real sobre os problemas evidenciados na análise de código, além de outras funcionalidades.

Considerando a primeira tela (Figura 13), na primeira parte (mais à esquerda), tem-se maior foco no acompanhamento de um Documento de Arquitetura (DAS). Nesse contexto, enumeram-se as seções básicas de um DAS, permitindo reconhecer - claramente e visualmente - sobre a presença dessas seções para o caso do Sistema Legado em análise. Modelagens com os diferentes diagramas para cada visão arquitetural, assim como seções

de relevância que conferem detalhamento de cunho negocial, poderiam ser consultadas, editadas e analisadas pela equipe.

Ainda na primeira tela (Figura 13), mas agora com foco na segunda parte (mais à direita), tem-se olhar centrado em um problema específico, evidenciado durante a realização das etapas do protocolo de Estudo de Caso na Empresa A. Aqui, reportou-se sobre a dificuldade de entendimento quanto às regras negociais por parte de novos desenvolvedores da Equipe B. Nesse contexto, faz-se uso de recursos, no *Dashboard*, para que certas revelações visuais sejam alinhadas e especificamente dedicadas aos desenvolvedores novatos, direcionando-os diretamente para as principais regras negociais do Sistema Legado.

Figura 13 – Protótipo Seção de Documentação

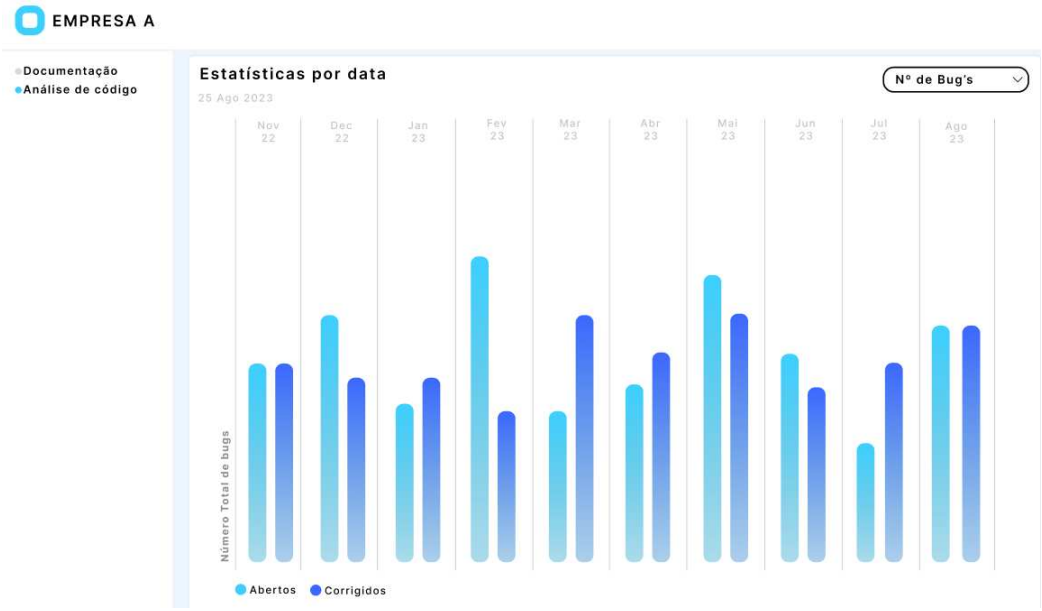


Fonte: Autor usando Figma

Considerando a segunda tela (Figura 14), tem-se maior foco nos problemas evidenciados via análise de código, sendo esses apresentados em tempo real, com possibilidade de tratamento. Há, para tanto, uma linha de tempo mensal para facilitar o acompanhamento. O *dropdown* presente na tela ajusta as informações sobre cada problema, considerando cada linha do tempo. No protótipo em particular, consta evidenciado para o caso do número de *bugs* presentes na aplicação em análise. Demais análises possuiriam gráficos similares (ex. *code smells*, linhas duplicadas). Portanto, o centro das atenções dá-se pela análise dos resultados provenientes das correções e documentação ao longo do tempo, permitindo compreender os impactos de cada melhoria na documentação, ou ainda em termos de padronizações arquiteturais no Sistema Legado X. Esses impactos causariam mudanças no comportamento das métricas ao longo da linha do tempo. A intenção era provocar mudanças comportamentais no próprio desenvolvimento da Equipe B, que teria

condições de acompanhar mais claramente as resolução de problemas no código.

Figura 14 – Protótipo Seção de Análise



Fonte: Autor usando Figma

6.2.1.1 Iterações de Interface

Como citado anteriormente, no Capítulo de Referencial Teórico, esses protótipos iniciais e iterações de interface passariam por revisões juntamente a Equipe B, seguindo-se pela metodologia de fluxo Teste Competitivo, *Design* Paralelo, e por fim o *Design* Iterativo.

6.2.1.1.1 Teste Competitivo

Inicialmente, após apresentação dos protótipos iniciais à Equipe B, foram coletados dados referentes as telas e necessidades mais pertinentes em relação ao acompanhamento da solução proposta. Isso ocorreu entre os desenvolvedores por meio de reuniões em conjunto. Nessa oportunidade, observou-se a necessidade de se criar uma nova tela (separada), dedicada à listagem de Funcionalidades e Regras Negociais. Isso colabora para um melhor acompanhamento, além de permitir um acompanhamento mais simples, via análise estática de código usando o SonarQube. Essa simplicidade é garantida, um vez que, há inerente interesse no acompanhamento evolutivo do código ao longo dos meses, mas não havia necessidade de observar melhorias visando a geração de novas versões. Diante do exposto, o protótipo conferido inicialmente poderia gerar complexidade demasiada no momento de acompanhamento, o que não era desejado.

6.2.1.1.2 *Design* Paralelo

Portanto, após a execução do Teste Competitivo, a fase de *Design* Paralelo foi iniciada. Algumas soluções foram apresentadas como a tela de Funcionalidades ser um espelho da antiga tela de Análise Estática de Código conforme já proposto no protótipo. Havia a possibilidade ainda da tela ser única, ou seja, sem a implementação do *Dropdown*. Nesse caso, a tela conteria os registros das funcionalidades e regras negociais mês a mês. Ocorreu mais uma opção, onde na tela seria possível filtrar o que foi registrado em cada mês, auxiliando os desenvolvedores com menos experiência na aplicação. Assim, seria mais fácil identificar regras e funcionalidades comuns, usando os filtros.

Já para a questão do cadastro, foram propostas duas soluções: o cadastro manual via código com uma *Tag* específica, o que viabiliza a identificação na aplicação; ou ainda o cadastro realizado de forma manual.

Para a tela de Análise Estática de Código, foram propostas soluções em que a tela conteria apenas uma análise rápida do código, sem a necessidade de submeter a versão à esteira de *Continuous Integration/Continuous Deploy* (CI/CD). Nessa esteira, têm-se validações de qualidade comumente realizadas. Isso implica em um grande número de reclamações, devido à lentidão e à indisponibilidade da plataforma utilizada. Outra solução proposta foi a implementação de gráficos simples, comparando a última análise com a análise atual de código para acompanhamento de evolução e introdução de novos problemas. Em relação às métricas apresentadas nessas telas, sugeriu-se algumas algumas métricas comuns no cotidiano da Equipe B, como *Bugs*, *Code Smells* e cobertura de código.

Após reuniões coletando as ponderações de desenvolvedores da Equipe B, foi decidido que a tela de Funcionalidades seria em tela única. Nessa tela, Funcionalidades e Regras Negociais seriam disponibilizadas por mês, além de permitir pesquisa e observação para saber onde as funcionalidades se encontravam no código e seus respectivos métodos de tratamento.

O método de cadastro foi preferido que os cadastros fossem realizados via código, permitindo semiautomatização durante *scan* da aplicação. Em um primeiro momento, sugeriu-se que esse *scan* fosse feito em uma nova tela dedicada às questões de cadastro do projeto, análise estática de código e varredura do código em busca de funcionalidades. Entretanto, optou-se por essa nova tela, chamada de *Home*, ser simples, havendo apenas a inserção de dados referentes ao cadastro do projeto visando a varredura. Nessa decisão, orientou-se pelo critério de simplicidade, com facilidade de uso e visualização.

Para a tela de análise estática de código, ficou estabelecido que as métricas seriam comparadas entre a última análise e a mais recente. Como métricas, optou-se por: *Bugs*, *Code Smells*, cobertura de testes e taxa de linhas duplicadas.

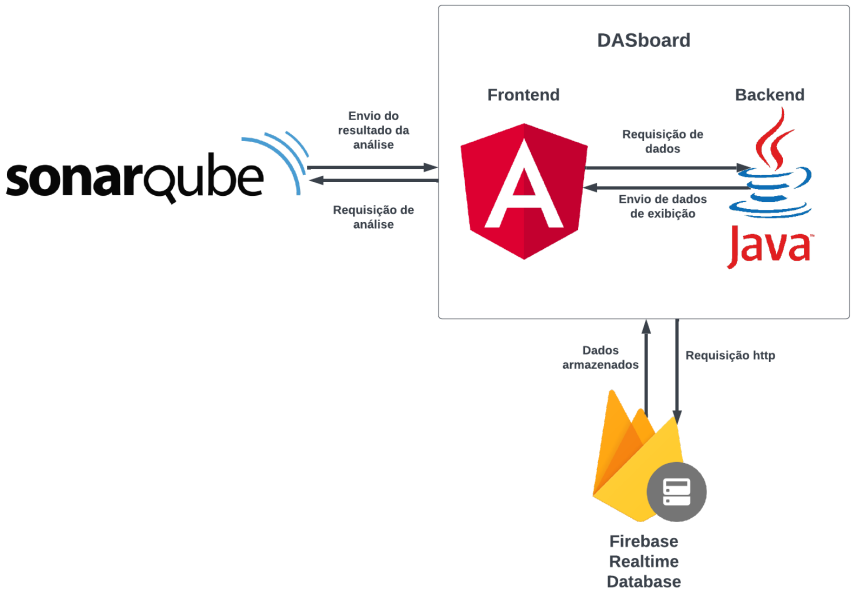
6.2.1.1.3 *Design* Iterativo

Para o *Design* Iterativo, algumas telas diferentes foram apresentadas aos desenvolvedores da equipe B para idealização de funcionalidades. Tais telas continham diferentes elementos, porém todos com o mesmo intuito. Por exemplo, na tela de Cadastro de Funcionalidades, disponibilizou-se dois modelos diferentes. Um com o gráfico de barras aparecendo apenas para meses em que ocorressem cadastro pelos usuários; e outro que mostrasse também os meses vazios, indicando progressão. Essas iterações foram acordadas por meio de reuniões e conversas durante o desenvolvimento do projeto.

6.2.2 Arquitetura DASBoard

O projeto desenvolvido conta com três componentes arquiteturais principais, além de ferramentas auxiliares. O apoio auxiliar mais pertinente é o uso de uma API para tratamento das métricas disponibilizadas no **DASboard**. A Figura 15 ilustra um modelo de relações para auxiliar na compreensão da representação arquitetural do DASBoard.

Figura 15 – Modelagem da Arquitetura do DASboard



Fonte: Autor usando LucidChart

O *Backend* é constituído pela linguagem Java, sendo responsável pela busca, pelo tratamento, e pelo envio dos dados que serão disponibilizados ao usuário. É responsável ainda pela comunicação com a base de dados, para manipulação dos dados que serão armazenados em função das análises ao longo do tempo. Além disso, o *Backend* tem como responsabilidade a aquisição dos dados oriundos da análise de código, realizada utiliando

a API do *SonarQube* (SONARSOURCE, 2023). Assim, métricas são disponibilizadas, juntamente com a documentação no **DASboard** e os cadastros realizados pelos usuários.

O *Frontend* consiste na utilização do Angular para a visualização dos dados enviados pelo *Backend*, além da filtragem - em tempo real - sobre as diferentes seções do *Dashboard*. Aqui, segue a lógica proposta de manter os dados consistentes na tela, sem alta complexidade envolvida, para que a utilização seja dinâmica e de fácil acesso aos integrantes da Equipe B.

A base de dados é gerenciada pelo Firebase, com o *Firebase Realtime Database*, utilizando-se das vantagens já previamente mencionadas em capítulos anteriores. Dentre essas vantagens, cabe mencionar a facilidade de utilização; a adequada segurança, e a integração entre a ferramenta do Firebase e o *Backend*, uma vez que há bibliotecas da solução de fácil acesso para a linguagem Java, o que provê uma solução combinada.

Por fim, mas não menos importante, têm-se que as tecnologias utilizadas possuem amplas documentações disponíveis. Isso facilita o desenvolvimento e a integração entre as tecnologias, diminuindo o tempo necessário para a aprendizagem e a resolução de erros que porventura ocorram durante o desenvolvimento da aplicação.

6.2.2.1 Organização de Pastas da Aplicação

O código do aplicativo está separado em pastas, havendo uma divisão de pastas entre *Frontend* e *Backend*. Essa divisão é orientada por pacotes.

Para o *Backend*, há uma divisão padrão para aplicações Java *Spring Boot*. A Seguir, constam breves explicações de cada divisão:

- *Config*: Pasta de configurações diversas da aplicação, seja de bibliotecas externas ou regras internas a serem seguidas por diferentes classes. Nela encontram-se a configuração de CORS(*Cross-origin resource sharing*) para comunicação segura entre *Frontend* e *Backend*, e também a configuração do Firebase para acesso e requisição da base de dados da aplicação;

- *Controller*: Camada de controle que gerencia as comunicações entre as duas camadas de *Model* e *View* da aplicação. Nela, são definidos os serviços utilizados pela camada visual para apresentar dados ao usuário. Conta ainda com uma subcamada *impl*, na qual é realizada a implementação desses serviços de forma mais detalhada, além da chamada direta para esses serviços;

- *Dto*: Pasta onde são armazenadas as classes de retorno e requisição da aplicação, e também para manipulação geral de dados realizados durante a execução dos serviços. Esses serviços, na sua grande maioria, são exportados ou recebidos como JSON, e seus valores chave são atrelados a seus respectivos atributos;

- *Model*: Pasta onde são armazenadas as classes que serão exportadas para o Firebase como entidades da base de dados. Todos os campos correspondem a um valor armazenado no Firebase. Seus identificadores são definidos na camada de serviços, sendo esta apenas para organizar as entidades gerais da aplicação, e

- *Service*: Camada onde a maior parte das atividades relacionadas ao *Backend* ocorre. Há a implementação de todos os serviços definidos na camada de controle, ocorrendo o recebimento dos dados externos de requisição para manipulação. Posteriormente, tem-se o envio de dados para a camada visual apresentar para o usuário.

Para o *Frontend*, há uma camada de configurações básicas de projetos do Angular, chamada de *src*. Nela estão as principais pastas, sendo essas referentes à cada funcionalidade, além da pasta de serviços, encontrada no caminho *core/services*. Nesse último caso, tem-se a camada que faz a comunicação dos serviços com a camada Controller, além de possuir um serviço específico para a tela de compartilhamento de dados entre diferentes telas.

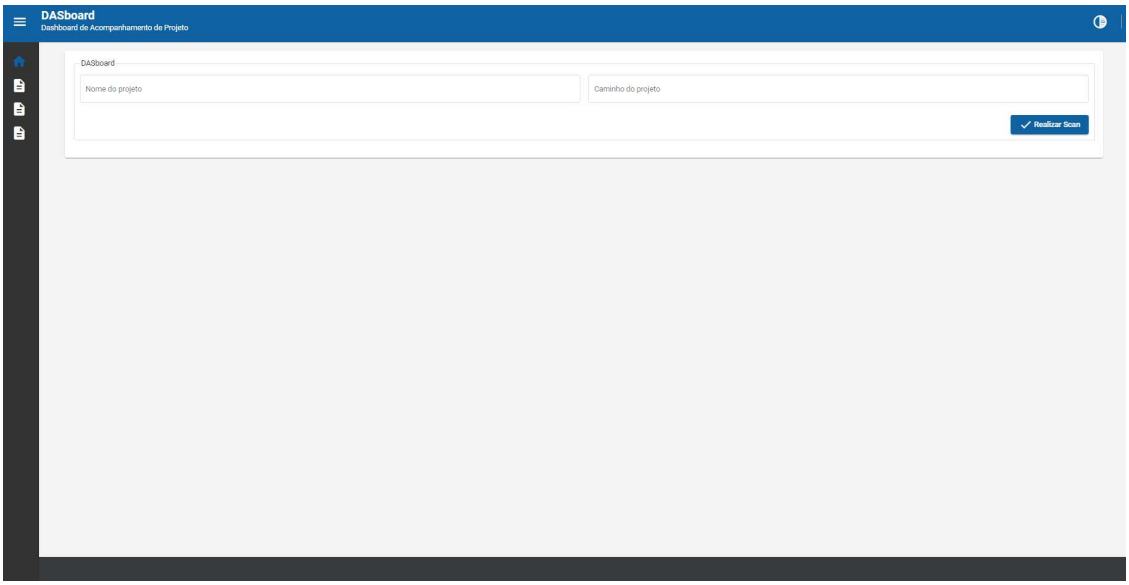
6.3 Aplicação DASboard

Nesta seção, serão conferidas alguns detalhes do processo de desenvolvimento e evolução da aplicação **DASboard**. Adicionalmente, será apresentada a aplicação em seu estado atual, ou seja, implementada para utilização dos membros da Equipe B. É importante ressaltar que o desenvolvimento foi feito juntamente com a análise de código estática do *SonarQube*, no intuito de evidenciar boas práticas de desenvolvimento, bem como código limpo no geral.

Como mencionado anteriormente, as telas e funcionalidades foram especificamente desenvolvidas para atender as necessidades da Equipe B referentes ao acompanhamento de documentação e à qualidade do Sistema X e demais sistemas legados de cunho similar presentes no cotidiano da empresa. As telas focam em tentar disponibilizar os dados de forma mais simples e objetiva ao usuário, referindo-se a aos embasamentos descritos no capítulo de Referencial Teórico. Sendo assim, evita-se desenvolver orientando-se por métricas muito complexas ou telas demasiadamente elaboradas, capazes de distrair o usuário. Lembrando que o intuito foi ser simples, intuitivo e de rápido acesso e uso.

Inicialmente, para o cadastro de dados da aplicação, eram requisitados apenas o nome dado ao projeto e o caminho do projeto nas pastas do usuário, conforme Figura 16:

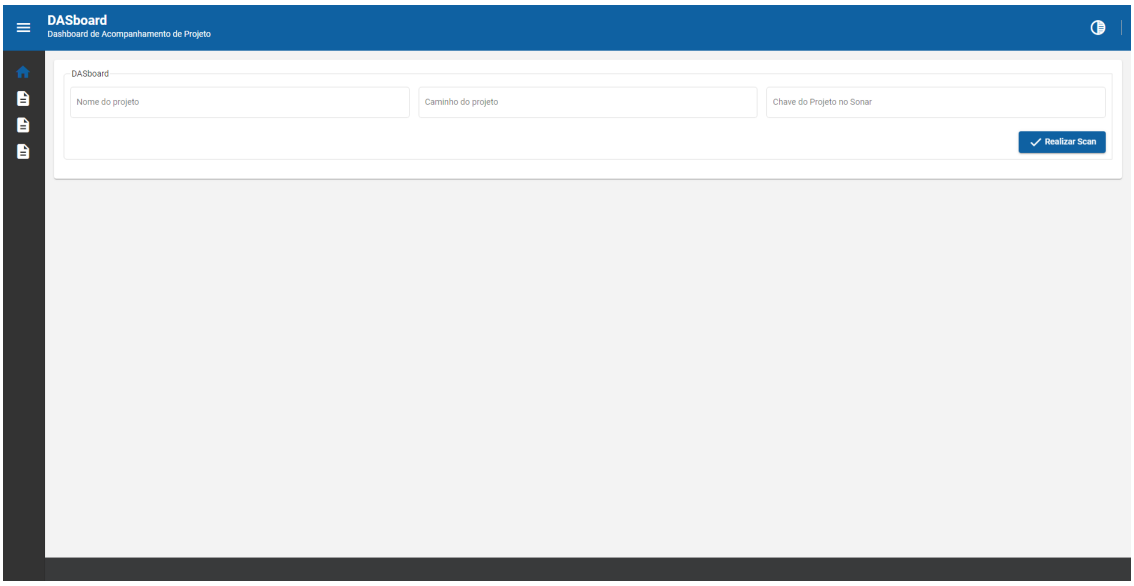
Figura 16 – Tela Inicial do DASboard



Fonte: Aplicação DASboard

Entretanto, após testes, foi observado que era mais interessante também informar a chave de projeto do *SonarQube*, uma vez que poderia habilitar a aplicação para gerar análise de diferentes projetos, indo além da análise do Sistema X. Nesse caso, deve-se configurar na esteira do *Sonar* da Empresa A, conforme ilustrado na Figura 17.

Figura 17 – Tela Inicial do DASboard Após Modificação

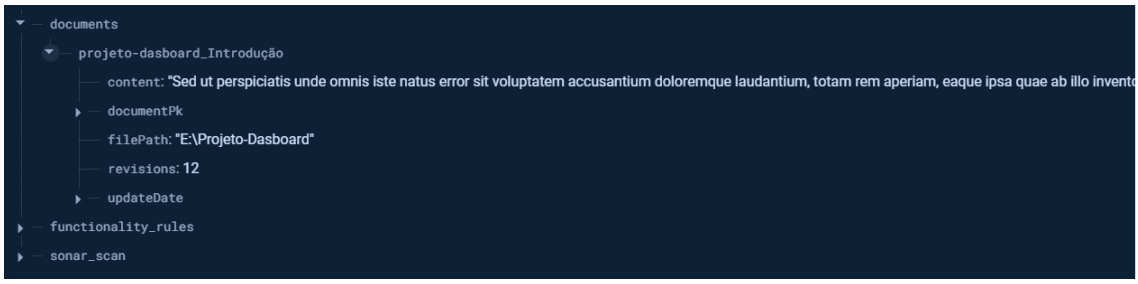


Fonte: Aplicação DASboard

Realiza-se, então, um *scan* do projeto informado, a partir do qual a aplicação irá resgatar o cadastro de funcionalidades e armazenar na base de dados do Firebase para disponibilização de métricas. Serão armazenados os dados do projeto para *scans* de análise estática de código, além do acompanhamento do documento de arquitetura.

Todos os dados são armazenados no Banco de Dados Não Relacional do Firebase de forma rápida e segura, conforme apresentado na Figura 18. Observa-se a estrutura geral dos dados armazenados no Firebase e alguns dados pertinentes que serão resgatados via serviço para futuras análises.

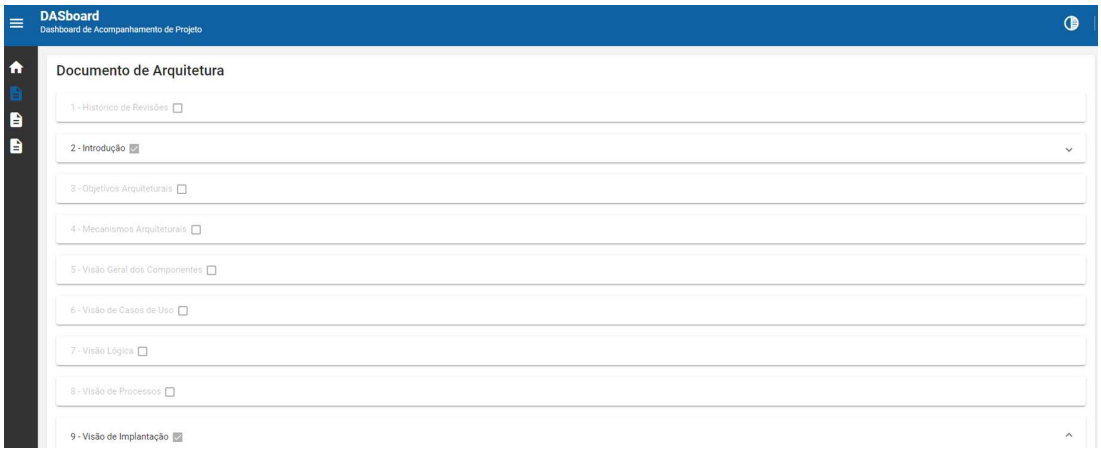
Figura 18 – Base de Dados Firebase



Fonte: Google Firebase Realtime Database

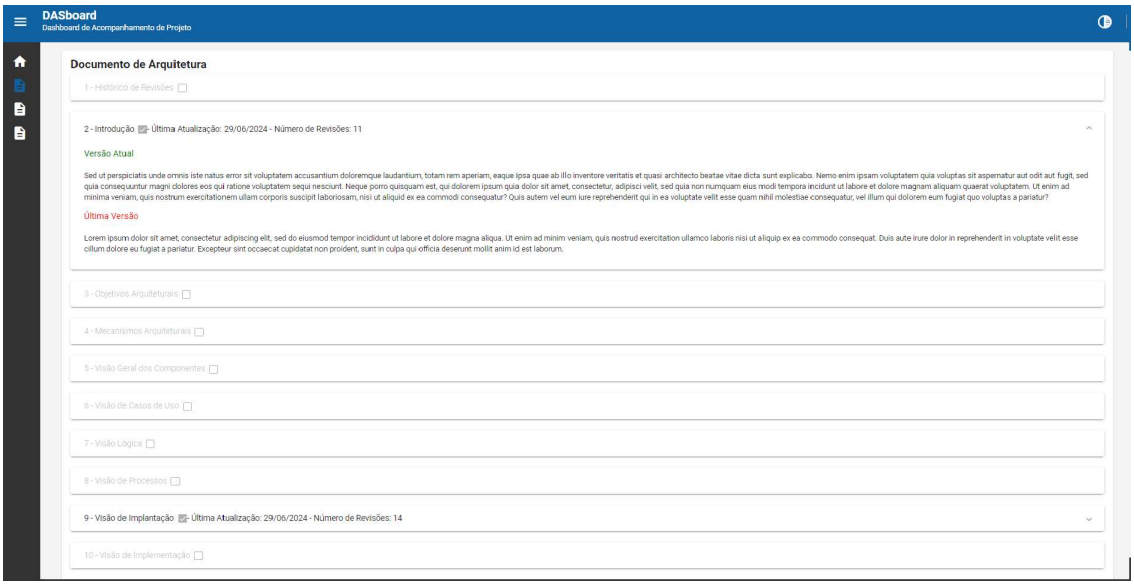
Na tela de acompanhamento da documentação do DAS, inicialmente, foi implementada uma solução simples, de acordo com os protótipos iniciais (Figura 13), conforme Figura 10. Porém, com o intuito de trazer um diferencial para a *feature* ao invés de só detalhar os estados e conteúdos do DAS, foi implementada uma possibilidade de conferência do conteúdo do DAS referente à última versão, sendo atribuída e disponibilizada uma data de alteração do documento juntamente com o número de mudanças registradas no **DASboard**. Isso viabiliza o acompanhamento de datas de alteração do documento, tanto para verificação/validação do documento (ex. verificar se ainda se mantém coerente com versões atuais); quanto para saber se este é constantemente atualizado e quais mudanças ocorreram, conforme consta na Figura 20. Na implementação atual da aplicação, é possível verificar e validar DAS, seja esse documento especificado em .txt, .docx e/ou .pdf.

Figura 19 – Tela de Acompanhamento de DAS Inicial



Fonte: Aplicação DASboard

Figura 20 – Tela de Acompanhamento de DAS Atualizada

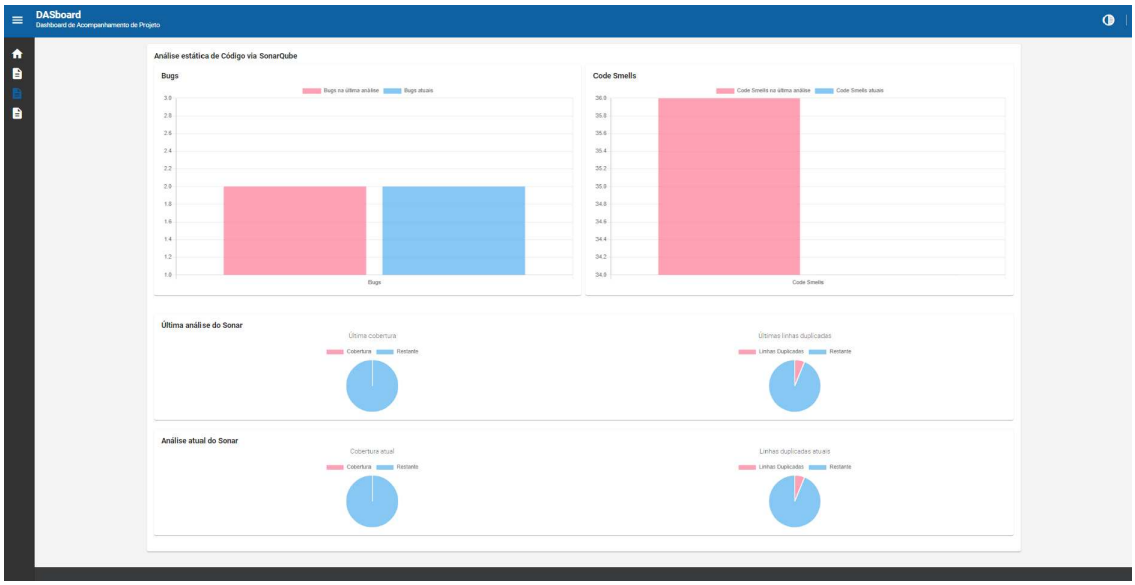


Fonte: Aplicação DASboard

Na tela de análise estática de código via *SonarQube*, conforme Figura 21, há uma disposição de quatro gráficos. Orientando-se pelas boas práticas de desenvolvimento de *Dashboards* mencionadas no Capítulo de Referencial Teórico há uso de gráficos simples apenas com os dados pertinentes, provendo rápida visualização para a comparação entre a última análise realizada com a análise atual. O intuito da tela dá-se pelo fato de haver uma necessidade de conferência da qualidade do código durante o desenvolvimento do mesmo, uma vez que, nas atividades cotidianas da Equipe B, a análise é feita somente durante gerações de versões via esteira CI/CD. Essa realização, primariamente, é feita pelo *Jenkins*. Trata-se de uma esteira em constante uso. Portanto, apresenta lentidões, representando um problema.

Uma vez que as aplicações legadas já possuem muitos problemas vinculados a elas, uma análise via aplicação facilita o controle de qualidade, permitindo melhorias constantes; ou ainda evitando a lentidão. Nesse caso, a comunicação é feita diretamente com o *SonarQube*, via integração com Java, por meio da biblioteca *Sonar Scanner*. Sendo assim, não há necessidade da esteira realizar essa comunicação durante a geração de novas versões, conforme ilustrado na Figura 22.

Figura 21 – Tela de Acompanhamento de Análise Estática de Código



Fonte: Aplicação DASboard

Figura 22 – Tela de integração com o Sonar via Java

```java
public SonarQubeAnalysisResultDto runSonarScanner(String projectPath, String projectKey) throws IOException {
    try {
        ProcessBuilder processBuilder = new ProcessBuilder("mvn.cmd", "sonar:sonar",
                "-Dsonar.host.url=" + sonarHostUrl,
                "-Dsonar.login=" + sonarToken);
        processBuilder.directory(new File(projectPath));
        Process process = processBuilder.start();

        BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String line;
        StringBuilder output = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            output.append(line).append("\n");
        }

        int exitCode = process.waitFor();
        if (exitCode != 0) {
            return new SonarQubeAnalysisResultDto(false, "Falha de análise sonarqube: " + exitCode + ".\n" + output);
        }
        return fetchAnalysisResults(projectKey);
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        return new SonarQubeAnalysisResultDto(false, "Análise sonarqube falhou com exceção: " + e.getMessage());
    }
}
```

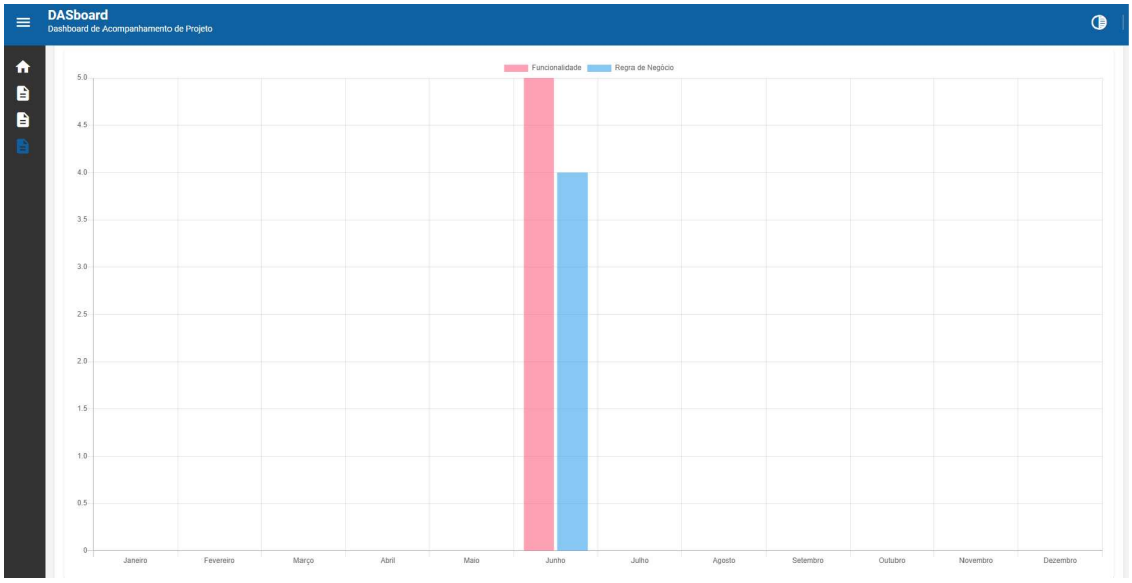
Fonte: Aplicação DASboard

Para a última tela, conforme ilustrado na Figura 23, para o acompanhamento de funcionalidades e regras negociais da aplicação, é provida uma disposição de gráficos de barras por mês. Em que cada mês, é conferida uma visualização de cadastros feitos em ambas modalidades. Essa conferência mensal serve para identificar a adesão da aplicação assim como a evolução de documentação da aplicação. Adicionalmente, é possível verificar por meio de buscas quais cadastros foram realizados e sua localização no código. Inicialmente, essa localização seria feita de forma direta, abrindo o local do código onde se encontram. Porém, após coleta de *Feedback* inicial, foi proposta uma busca no próprio **DASboard**, evidenciando exatamente onde se encontram esses cadastros. Isso ajuda, pois os membros da Equipe B utilizam diferentes formas para desenvolver, e um link direto para o código não seria uma solução universal.

O cadastro é feito de forma manual pelos desenvolvedores, por duas principais razões: (i) tanto para não haver um problema similar ao que já ocorre onde há uma

sobrecarga de informação referente ao conteúdo já presente no sistema em relação a um cadastro mais automatizado de todo o sistema; (ii) quanto para criar uma cultura de documentação enquanto se utiliza a aplicação. Nesse sentido, optou-se por realizar o cadastro usando *Tags* referentes ao **DASboard**, que são observadas durante o *Scan* da aplicação. A tela pode ser visualizada na Figura 24.

Figura 23 – Tela de Acompanhamento de Cadastro



Fonte: Aplicação DASboard

Figura 24 – Tela de Acompanhamento de Cadastro - Busca

Fonte: Aplicação DASboard

6.4 Resumo do Capítulo

Nesse capítulo, foi conferida uma visão mais detalhada sobre o que é o **DASboard**. Iniciou-se sobre a origem da ideia, contextualizando sobre o autor e seus anseios enquanto desenvolvedor na Empresa A. Em seguida, foi apresentada a idealização, o intuito, e o início do desenvolvimento da aplicação, demonstrando a descrição básica do que

a aplicação precisava atender e ser. Acordou-se sobre as *features* principais, que foram planejadas inicialmente e posteriormente implementadas, incluindo os propósitos de cada uma e a razão de terem sido incluídas.

Conferiu-se ainda maior aprofundamento quanto ao **DASboard**, com a apresentação dos primeiros protótipos das telas do *Dashboard*. Cada protótipo de tela foi descrito. Além disso, foi realizada uma descrição do método de criação de *Dashboard*, mencionado em capítulo anterior, mesclando três práticas de idealização das telas para atender de forma mais concretas a necessidades da Equipe B. Adiante, tem-se uma breve apresentação sobre os principais componentes arquiteturais do *Dashboard* e sua estrutura de pastas. Também retomam-se as principais tecnologias utilizadas no desenvolvimento do trabalho, com menção às tecnologias: Java, Angular, Firebase, e as APIs do *SonarQube*. Tecnologias essas pertinentes para a integração das métricas no *Dashboard*. Por fim, há a apresentação da aplicação em si, com todas suas funcionalidades e a razão de alterações em relação às ideias iniciais.

7 Análise de Resultados

Esse capítulo apresenta a análise dos resultados obtidos com a aplicação **DASboard** no ambiente da Equipe B da Empresa A, sob o ponto de vista dos desenvolvedores. Além disso, é exposta a análise do autor referente aos dados coletados e as impressões e opiniões dos usuários consultados. Confere-se, inicialmente, como foi realizado o processo de Coleta de Dados. Logo após, há a apresentação dos resultados, acompanhada pela Análise dos Resultados.

Essa análise orienta-se pelo método adotado e documentado no Capítulo 4. Lembrando ainda que outras etapas (mais iniciais), inerentes ao método de análise de resultados acordado na seção 4.4., encontram-se no Capítulo 5 - Estudo de Caso. No Estudo de Caso, é possível perceber como foram conduzidas as etapas de Formulação do Problema, Definição da Unidade-Caso, Determinação do Número de Casos e Elaboração do Protocolo. No presente capítulo, a intenção é mostrar de forma mais detalhada a Coleta de Dados e a Avaliação e Análise dos Dados. Por fim, tem-se o Resumo do Capítulo.

7.1 Coleta de Dados

Para a coleta de dados referentes ao **DASboard**, era necessário disponibilizar para os desenvolvedores da Equipe B meios de utilizar a aplicação, e ainda manter suas atividades comuns de trabalho sem interferência. Também era relevante considerar fatores de segurança da Empresa A, os quais impossibilitaram a implantação direta em sistemas como o Sistema X.

Diante do exposto, a coleta foi realizada de forma em que um projeto Java comum fosse disponibilizado para que os desenvolvedores pudessem utilizar a aplicação. A intenção era possibilitar que esse grupo de interessados tirasse suas impressões da solução como um todo. A utilização da aplicação foi feita de forma local, de forma individual (ou seja, cada desenvolvedor). Nesse caso, faz-se o uso simultâneo dos módulos *Frontend* e *Backend*.

Por fim, cabe colocar que o grupo de interessados foi consultado sobre o intuito da coleta. Todos aceitaram o Termo de Consentimento Livre e Esclarecido, especificado nos moldes do documento apresentado no (Apêndice A A). Não ocorreu preparação da equipe sobre a aplicação em si, mas os desenvolvedores sabiam que o autor desse trabalho estava realizando uma pesquisa para melhorar o acompanhamento e a compreensão dos sistemas legados da empresa.

7.2 Análise dos Resultados

A aplicação ficou disponível por 7 dias. Após o período, foram executadas as atividades de análise de resultados, também guiada pelo método mencionado no Capítulo 4 (seção 4.4.). As atividades iniciais da análise compreendem uma visão geral da aplicação pelos desenvolvedores da aplicação, acompanhada por entrevistas realizadas com o Entrevistado A e Entrevistado B.

Os entrevistados tiveram seus perfis definidos e entrevistas realizadas antes da implantação da solução, conforme consta colocado no Capítulo 5. Adicionalmente, foi realizada mais uma entrevista, com um novo integrante da Equipe B, chamado de Entrevistado C. Esse novo participante permitiu conferir retornos ainda mais precisos sobre a pertinência da aplicação. Isso ocorreu, pois esse participante teve sua inserção na Equipe B e na Empresa A muito recentemente. Portanto, ele possui uma impressão sobre os Sistemas Legados que revela várias dificuldades.

7.2.0.1 Entrevistas

A entrevista foi feita tentando captar uma visão geral sobre a aplicação, assim como as diferentes reações e sentimentos em relação à mesma. Como mencionado em capítulo anterior, a senioridade e a experiência com sistemas legados da Empresa A são fatores que impactam na recepção da aplicação e seus usos.

A entrevista possui 14 perguntas. Questionou-se, principalmente, a recepção dos entrevistados sobre diferentes funcionalidades, bem como a aplicação como um todo. Inicialmente, foi realizada a mesma pergunta realizada nas entrevistas documentadas no Capítulo 5. Isso foi importante para assegurar sobre o consentimento do entrevistado acerca do objetivo da pesquisa e sua natureza: "Antes de prosseguirmos com o questionário, gostaríamos de esclarecer que esta entrevista é puramente acadêmica e confidencial. Suas respostas serão tratadas de forma anônima, sem a divulgação do seu nome ou da empresa em que você trabalha. Você está ciente e consente com essa condição de confidencialidade?"Após a validação por parte do entrevistado, a entrevista prosseguiu da seguinte forma:

Entrevistado A

Entrevistador: Autor Gabriel Alves Hussein

Pergunta 1:

O que você achou da aplicação no geral?

Resposta: Achei a proposta da aplicação muito relevante, pois a maioria dos projetos legados carece de documentos. Quando cheguei por exemplo tive muita dificuldade de aprender as coisas que não fosse ir no código procurar tudo manualmente.

Pergunta 2:

A interface da aplicação foi intuitiva e fácil de usar?

Resposta: A interface ficou simples e isso me agrada.

Pergunta 3:

Quais foram suas impressões sobre a funcionalidade de acompanhamento do Documento de Arquitetura?

Resposta: Achei útil e, caso aprimorado, o versionamento se tornará melhor, talvez incluindo a pessoa que mudou o documento.

Pergunta 4:

Você achou útil ter a implementação do histórico de alterações?

Resposta: Muito relevante, porque possibilita saber se o documento tá atualizado ou não.

Pergunta 5:

Quais foram suas impressões sobre a visualização gráfica da análise de código estático do SonarQube?

Resposta: Gostei, gráficos sempre são bons para demonstrar evolução, porém acho que talvez ter a opção de comparar com diferentes branches seria uma boa adição.

Pergunta 6:

A apresentação dos gráficos foi intuitiva?

Resposta: Sim, não tive problemas para identificar nenhum dado.

Pergunta 7:

A apresentação dos gráficos o ajudaria a conseguir acompanhar a qualidade de suas mudanças de forma prática?

Resposta: Sim, já que o jenkins é bem lento na maioria das vezes, poder ver se a mudança não piorou a análise do sonar localmente já ajuda bastante.

Pergunta 8:

Quais foram suas impressões sobre a funcionalidade de visualização de funcionalidades e regras de negócio?

Resposta: É uma boa ideia para identificar funcionalidades e regras na aplicação sem ter que sair buscando toda vez que der erro, porém talvez uma busca por palavra-chave ajudaria a identificar mais rápido.

Pergunta 9:

Houve alguma informação ou recurso que você gostaria de ver nesta funcionalidade?

Resposta: Uma sugestão seria tentar mapear também o que ainda falta documentação, para saber um quantitativo de progressão da documentação, mas no geral a funcionalidade atende.

Pergunta 10:

Qual das funcionalidades você acha que usaria com mais frequência? Por quê?

Resposta: Acredito que a parte do sonar com maior frequência, porque é necessário manter um valor mínimo de qualidade e análise do sonar para gerar uma nova versão, e como disse às vezes o jenkins não colabora.

Pergunta 11:

Qual das funcionalidades você acha que usaria com menos frequência? Por quê?

Resposta: Acredito que a de documentação de arquitetura pela falta de costume que temos de fazer documentação assim, porém em uma aplicação com revisão de documento ela se torna indispensável, caso o costume de criar documentos de arquitetura comece essa funcionalidade se torna bem mais importante.

Pergunta 12:

Existe alguma funcionalidade que você gostaria de ver adicionada à aplicação?

Resposta: Acredito que seria necessário salvar as app já mapeadas e também uma forma de acessá-las com facilidade.

Pergunta 13:

Há algo que você gostaria de mudar ou melhorar em alguma das funcionalidades existentes?

Resposta: Melhoria no gráfico para inclusão do filtro de ano e incluir também os métodos não mapeados.

Pergunta 14:

Você usaria a aplicação no seu cotidiano? Se sim por qual razão?

Resposta: Sim, para ajudar o pessoal novo chegando e também para compartilhar conhecimento geral dos nossos sistemas mais antigos.

É possível observar que o Entrevistado A teve uma boa recepção em relação à aplicação. Mesmo que tenha apontado que há vários pontos de melhoria, há um reconhe-

cimento sobre os usos para a solução no ambiente que está inserido, seja por tratar de problemas infraestruturais, como a geração de versão via CI/CD apresentar problemas frequentes; quanto para disseminar conhecimento das aplicações para novos integrantes.

O Entrevistado A, no momento desta segunda entrevista, já possui bem mais experiência com os Sistemas Legados da Empresa A. Entretanto, observa-se, durante as duas entrevistas do Entrevistado A, que houve frustração quanto ao aprendizado desses sistemas, e que a solução pode ajudar integrantes novatos futuros a terem uma ambientação mais facilitada aos sistemas.

Após a primeira entrevista, foi novamente entrevistado o Entrevistado B. Lembrando que esse possui grande experiência na empresa, cerca de 13 anos, e que também teve contato com a aplicação durante o período de testes.

Entrevistado B

Entrevistador: Autor Gabriel Alves Hussein

Pergunta 1:

O que você achou da aplicação no geral?

Resposta: É uma boa ideia. Um problema que tenho é que com o passar do tempo tenho que explicar o mesmo sistema várias vezes para os novos desenvolvedores e isso consome muito tempo quando se acumula todos os treinamentos feitos. Como não há documentação, é algo que demora ainda mais.

Pergunta 2:

A interface da aplicação foi intuitiva e fácil de usar?

Resposta: Bem simples e fácil de ler, consegui entender tudo bem rápido.

Pergunta 3:

Quais foram suas impressões sobre a funcionalidade de acompanhamento do Documento de Arquitetura?

Resposta: No momento por ter muitos incidentes e problemas no dia a dia não temos muito tempo de desenvolver documentação de arquitetura desta forma, mas acho que é uma boa ideia. Deveríamos ter uma etapa no desenvolvimento para criar esses documentos, porém, por causa de prazos, acabamos não fazendo. Mas caso a aplicação seja implantada, tem a chance de pelo menos criar documentos simples para acompanhar por ela mais rapidamente.

Pergunta 4:

Você achou útil ter a implementação do histórico de alterações?

Resposta: Sim, dá para saber o que foi alterado desde a última versão e quando foi

alterado, ficando mais fácil de acompanhar.

Pergunta 5:

Quais foram suas impressões sobre a visualização gráfica da análise de código estático do SonarQube?

Resposta: É uma boa, rodar o sonar localmente ajuda a manter o controle antes de commitar e gerar versão.

Pergunta 6:

A apresentação dos gráficos foi intuitiva?

Resposta: Sim.

Pergunta 7:

A apresentação dos gráficos o ajudaria a conseguir acompanhar a qualidade de suas mudanças de forma prática?

Resposta: Por padrão, eu já tento validar minhas mudanças, rodar uma análise completa das mudanças ajudaria bastante.

Pergunta 8:

Quais foram suas impressões sobre a funcionalidade de visualização de funcionalidades e regras de negócio?

Resposta: Boa ideia para conseguir cadastrar para novos desenvolvedores entenderem sobre funcionalidades básicas, mas senti falta de conseguir achar todos métodos da aplicação de uma vez.

Pergunta 9:

Houve alguma informação ou recurso que você gostaria de ver nesta funcionalidade?

Resposta: Buscar todos os métodos, e conseguir pesquisar métodos por substring.

Pergunta 10:

Qual das funcionalidades você acha que usaria com mais frequência? Por quê?

Resposta: Se eu pudesse buscar todos os métodos eu gostaria de cadastrar pela própria aplicação uma funcionalidade para um método específico.

Pergunta 11:

Qual das funcionalidades você acha que usaria com menos frequência? Por quê?

Resposta: Se nada mudar em relação as demandas atropeladas, a de documentação,

mas eu gostei da ideia.

Pergunta 12:

Existe alguma funcionalidade que você gostaria de ver adicionada à aplicação?

Resposta: Talvez na mesma tela de documentação de arquitetura ou em outra tela, mas um acompanhamento das releases em produção seria bem útil.

Pergunta 13:

Há algo que você gostaria de mudar ou melhorar em alguma das funcionalidades existentes?

Resposta: Eu gostaria se a atividade de cadastro estivesse atrelada à geração de versões. Após fazer algo na aplicação, um dos requisitos de aprovação deveria ser o cadastro para a gente conseguir acompanhar no dashboard. Não sei se seria algo manual que criaríamos como ritual ou se tem como fazer integrando o dashboard e o bitbucket.

Pergunta 14:

Você usaria a aplicação no seu cotidiano? Se sim por qual razão?

Resposta: Sim, mesmo que tenha o que melhorar acho que ajuda bastante e salvaria muito tempo no longo prazo, além de talvez conseguir diminuir o gargalo de incidentes se o sistema como um todo for documentado.

O Entrevistado B também demonstra percepção positiva em relação à aplicação. Assim como na entrevista anterior, há apontamentos de melhoria e novas funcionalidades. Entretanto, é possível inferir a intenção quanto ao uso da aplicação, caso fosse algo rotineiro na jornada de trabalho. Mesmo o Entrevistado B sendo o integrante com maior experiência e conhecimento pleno de todos os sistemas, há interesse de gradualmente documentar os sistemas para usos diversos: seja para facilitar treinamentos; acompanhar *releases* como melhoria mencionada, ou até mesmo para inserir a aplicação como parte da metodologia de desenvolvimento da Equipe B. Essa prática seria de suma relevância para aprovação de novas mudanças nesses sistemas.

Por fim, tem-se a entrevista com o Entrevistado C. Trata-se de um desenvolvedor com boa experiência na área de tecnologia, porém está inserido na Equipe B desde Março de 2024. Encontra-se, portanto, com as dificuldades mencionadas nessa monografia quando tratado sobre os Sistemas Legados, com destaque para sua falta de documentação e padronizações.

Entrevistado C

Entrevistador: Autor Gabriel Alves Hussein

Pergunta 1:

O que você achou da aplicação no geral?

Resposta: De modo geral gostei da aplicação. Penso que ela pode ser muito útil para o monitoramento de documentações, algo que não é feito aqui para esses sistemas mais antigos.

Pergunta 2:

A interface da aplicação foi intuitiva e fácil de usar?

Resposta: Sim.

Pergunta 3:

Quais foram suas impressões sobre a funcionalidade de acompanhamento do Documento de Arquitetura?

Resposta: Gostei, bastante interessante a forma que é realizada a análise da documentação, acredito que ainda seja possível evoluir o software nesse sentido, fazendo uma análise mais profunda do documento.

Pergunta 4:

Você achou útil ter a implementação do histórico de alterações?

Resposta: Sim, dá para identificar o que foi mudado recentemente bem rápido.

Pergunta 5:

Quais foram suas impressões sobre a visualização gráfica da análise de código estático do SonarQube?

Resposta: É um ótimo complemento para a aplicação a integração com o sonar para a verificação analítica do código.

Pergunta 6:

A apresentação dos gráficos foi intuitiva?

Resposta: Sim, deu para entender o que está sendo mostrado tranquilamente.

Pergunta 7:

A apresentação dos gráficos o ajudaria a conseguir acompanhar a qualidade de suas mudanças de forma prática?

Resposta: Sim, já que geralmente só vemos gerando versão.

Pergunta 8:

Quais foram suas impressões sobre a funcionalidade de visualização de funcionalidades e regras de negócio?

Resposta: Ótima funcionalidade. Como acabei de entrar, ainda estou muito per-

dido. Porém, se algo parecido estivesse sendo utilizado com frequência, ia ficar muito mais fácil conseguir acompanhar tudo. Às vezes eu, pego algum chamado ou demanda de manutenção, e fico muito tempo perdido buscando onde arrumar porque não sei onde ficam as coisas. Como nem sempre consigo apoio do pessoal por estar todo mundo muito ocupado, algo assim ia salvar muito tempo.

Pergunta 9:

Houve alguma informação ou recurso que você gostaria de ver nesta funcionalidade?

Resposta: Acho que buscar manualmente as funcionalidades por nome.

Pergunta 10:

Qual das funcionalidades você acha que usaria com mais frequência? Por quê?

Resposta: Com certeza a de funcionalidades. Como alguém sem muita experiência no sistema, isso é algo que usaria todo dia.

Pergunta 11:

Qual das funcionalidades você acha que usaria com menos frequência? Por quê?

Resposta: Acho que a do sonar, só porque achei as outras mais úteis.

Pergunta 12:

Existe alguma funcionalidade que você gostaria de ver adicionada à aplicação?

Resposta: No momento não.

Pergunta 13:

Há algo que você gostaria de mudar ou melhorar em alguma das funcionalidades existentes?

Resposta: Buscar manualmente as funcionalidades e regras de negócio, e talvez conseguir ver quem cadastrou cada coisa.

Pergunta 14:

Você usaria a aplicação no seu cotidiano? Se sim por qual razão?

Resposta: Com certeza, tanto para cadastrar o que eu faço; quanto para achar mais fácil o que a aplicação faz.

É possível observar que para um integrante muito recente na Equipe B, há uma clara preferência em relação à funcionalidade de cadastro de Funcionalidades e Regras

Negociais, uma vez que é o ponto onde se encontram as maiores dificuldades. Há também uma boa recepção em relação à aplicação, indicando que a solução possui espaço na Equipe B, além de clara evidência dessa solução poder contribuir para a criação de uma cultura coletiva de documentação, sem impactar diretamente no tempo utilizado para as atividades que já são realizadas diariamente.

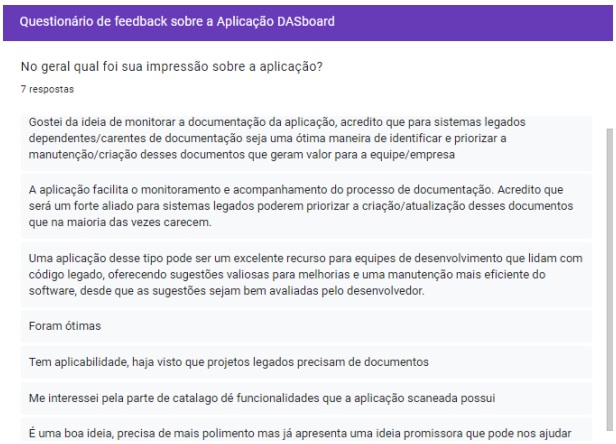
7.2.0.2 Formulário

Concluída a entrevista, também foi disponibilizado um formulário, no Google Forms, para coleta das impressões sobre a aplicação tomando como base os demais integrantes da Equipe B, que não foram entrevistados. O formulário ficou disponível por 5 dias após o período de testes da aplicação, além de oferecer um Termo de Consentimento Livre e Esclarecido (Apêndice A). O intuito do formulário deu-se para tentar identificar padrões, considerando as diferentes visões sobre a aplicação, tanto com base nos dados coletados na entrevista; quanto via questionário. Além disso, foi possível identificar o sentimento geral em relação ao **DASboard** e a própria proposta da aplicação.

O formulário continha um total de oito perguntas realizadas e preenchidas pelos desenvolvedores. Para garantir validade e confiabilidade ao formulário, o acesso foi concedido apenas a pessoas da equipe que estão envolvidas diariamente no desenvolvimento e na manutenção do Sistema Legado X e associados. O número de respostas variou por pergunta, com predomínio de 7 respostas por pergunta. A escolha do Google Forms ocorreu pela facilidade de preenchimento das respostas por parte dos participantes, assim como a simples visualização das respostas obtidas pelo autor. Esse processo facilitou a coleta e a análise dos dados. Para conhecimento, seguem as perguntas realizadas e suas respectivas respostas.

A primeira pergunta refere-se às impressões gerais sobre a aplicação dos demais desenvolvedores da Equipe B. Constaram 7 respostas como retorno. É possível observar que, assim como nas entrevistas, há uma boa aceitação a solução, por diferentes razões, conforme ilustrado na Figura 25.

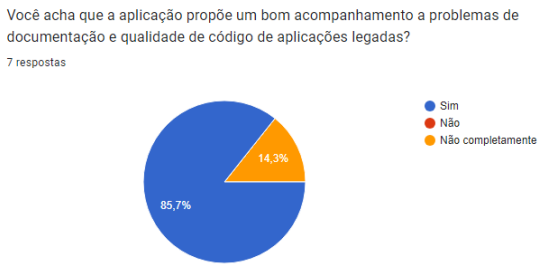
Figura 25 – Pergunta Formulário 1



Fonte: Autor usando Google Forms

A segunda pergunta tenta verificar de forma objetiva o sentimento geral em relação à proposta em si da aplicação, independentemente da implementação. Foram conferidas 7 respostas como retorno. 85,7% dos respondentes (maioria) informaram que "Sim", e 14,3% dos respondentes informaram que "Não Completamente". Dessa forma, é possível aferir sobre a pertinência da solução proposta ao ambiente do Estudo de Caso da Empresa A, conforme ilustrado na Figura 26.

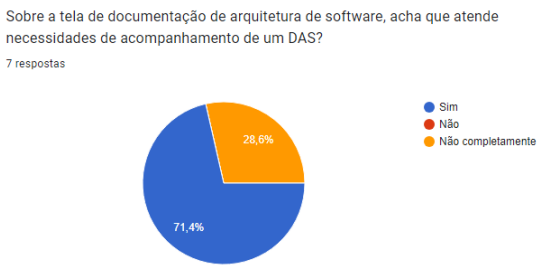
Figura 26 – Pergunta Formulário 2



Fonte: Autor usando Google Forms

A terceira pergunta procura verificar opiniões sobre uma funcionalidade específica da aplicação, neste caso, a de tela de documentação de arquitetura de *software*. A intenção é entender se essa ideia atende as necessidades do acompanhamento de um DAS. Houve um total de 7 respostas como retorno. 71,4% dos respondentes (maioria) informaram que "Sim", enquanto que 28,6% dos respondentes informaram que "Não Completamente". Essa percepção está alinhada à visão geral observada durante as entrevistas. Em resumo, há espaço para essa solução específica. Entretanto, de acordo com os entrevistados e respondentes do formulário, há pontos de melhoria a serem considerados, há pontos de melhoria conforme ilustrado na Figura 27.

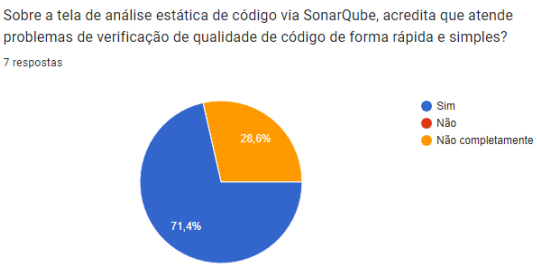
Figura 27 – Pergunta Formulário 3



Fonte: Autor usando Google Forms

A quarta pergunta procura verificar opiniões sobre uma funcionalidade específica da aplicação, neste caso, a de tela de análise estática de código. A ideia é compreender se as necessidades de validação de qualidade de código e acompanhamento como um todo do estado do código são atendidas. Houve um total de 7 respostas como retorno. 71,4% dos respondentes (maioria) informaram que "Sim", enquanto que 28,6% dos respondentes informaram que "Não Completamente". Novamente, há um alinhamento com visões observadas mais a fundo em entrevistas realizadas anteriormente. Em resumo, há pontos de melhoria a serem considerados. Entretanto, é uma solução que seria relevante para auxiliar em problemas de infraestrutura, assim como em análises rápidas e práticas, conforme ilustrado na Figura 28.

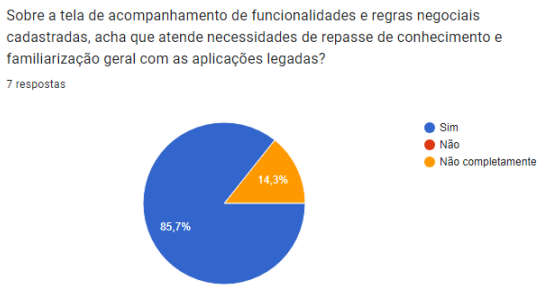
Figura 28 – Pergunta Formulário 4



Fonte: Autor usando Google Forms

A quinta pergunta procura verificar opiniões sobre uma funcionalidade específica da aplicação, neste caso, a de tela de acompanhamento de funcionalidades e regras negociais. A ideia é compreender se as necessidades de criar documentação em relação aos sistemas legados são atendidas. Houve um total de 7 respostas como retorno. 85,7% dos respondentes (maioria) informaram que "Sim", enquanto que 14,3% dos respondentes informaram que "Não Completamente". Trata-se da solução com maior popularidade entre as apresentadas, novamente alinhando com o observado nas entrevistas realizadas. O perfil da Equipe B possui muitos integrantes novos, portanto é uma funcionalidade que indica a pertinência do presente trabalho como um todo, conforme ilustrado na Figura 29.

Figura 29 – Pergunta Formulário 5

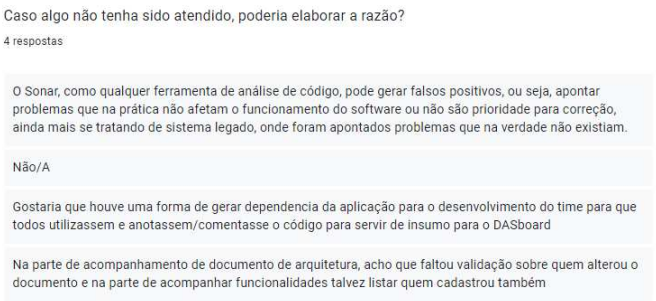


Fonte: Autor usando Google Forms

A sexta pergunta confere liberdade aos respondentes para tentar identificar quais funcionalidades não atenderam algum problema específico enfrentado pelos desenvolvedores da Equipe B, tendo a intenção de coletar dados referentes a possíveis omissões na solução. Foram, no total, 4 respostas, conforme ilustrado na Figura 30.

Ao analisar as respostas é possível observar que ocorreram apontamentos referentes à uma análise mais precisa do *SonarQube*, atendendo questões mais pertinentes aos sistemas específicos. Por exemplo, uma forma de identificar usuários que cadastraram funcionalidades ou alteraram o documento de arquitetura desde sua última versão. Outro ponto que se destacam em relevância é o apontamento realizado pelo Entrevistado B, que comentou sobre a funcionalidade de cadastro de funcionalidades ser diretamente atrelada ao fluxo de desenvolvimento da Equipe B, visando a geração de dados cadastrados para a solução, independente da forma como essa solução seja implementada (i.e. seja via código; seja manualmente). Permite-se afirmar que a proposta teve relevância, uma vez que gerou interesse em ser utilizada no cotidiano da Equipe B, para justamente auxiliar problemas presentes nos sistemas utilizados.

Figura 30 – Pergunta Formulário 6

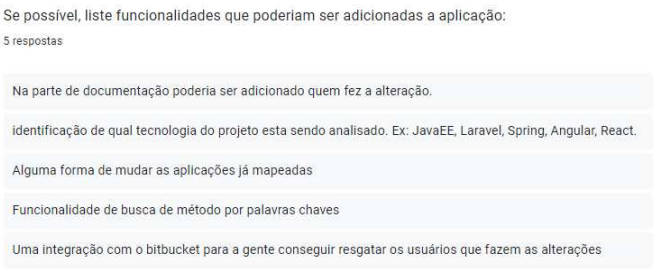


Fonte: Autor usando Google Forms

A sétima pergunta, novamente, confere liberdade aos respondentes. Entretanto, desta vez, com intuito de identificar novas soluções a serem adicionadas ao *dashboard* para melhor acompanhamento de métricas e dados dos Sistemas Legados presentes. Conforme ilustrado na Figura 31, foram 5 respostas. Dentre elas, há novamente menção ao

usuário que sugeriu alteração no DAS presente na aplicação; juntamente com outra resposta que sugeriu uma integração mais bem especificada com o *Bitbucket*, plataforma de armazenamento e gerenciamento de código utilizado pela Empresa A. Há ainda menção ao ponto do Entrevistado A, coletado via entrevista, no qual seria adequado permitir mudanças mais rapidamente entre projetos já analisados anteriormente, bem como buscas diretas por funcionalidade. Conclui-se que, enquanto houve boa receptividade da solução, e maior engajamento para uso futuro da mesma, há considerável relevância do trabalho realizado, e espaço para melhorias futuras.

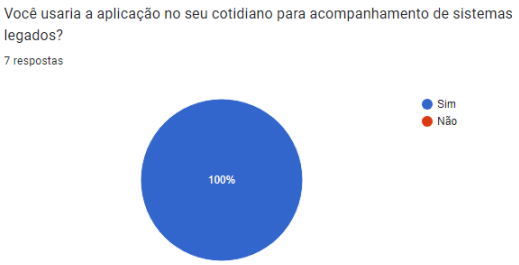
Figura 31 – Pergunta Formulário 7



Fonte: Autor usando Google Forms

Por fim, a oitava pergunta, de caráter objetivo, aborda o interesse dos respondentes em se utilizar a solução no seu cotidiano para acompanhar os Sistemas Legados. Teve ao todo 7 respostas. Todos os respondentes informaram que "Sim"conforme ilustrado Figura 32. Isso reforça o fato de que o estudo é pertinente, e a aplicação pode se tornar algo para auxiliar a Equipe B.

Figura 32 – Pergunta Formulário 8



Fonte: Autor usando Google Forms

7.2.0.3 Conclusão da Análise

Resumidamente, utilizando-se dos insumos coletados pelas experiências e visões dos integrantes da Equipe B, é possível perceber que o objeto de estudo deste trabalho, assim como a solução produzida baseado nesse estudo, são válidos como para lidar com problemas referentes a Sistemas Legados. Entretanto, cabem pontos de melhoria. De toda forma, a proposta e a implementação atendem e auxiliam no ambiente referido, com base

no grande interesse despertado nos interessados, principalmente, provendo novas ideias e formas de integrar a aplicação no cotidiano da Equipe B. Isso permitirá que, de forma gradual, se documente esses sistemas, contribuindo para melhor compreensão dos mesmos por parte de futuros integrantes na empresa.

7.3 Resumo do Capítulo

Nesse capítulo, foram apresentados os resultados obtidos provenientes do estudo de caso na Empresa A, onde a Equipe B era o objeto de estudo, assim como a ferramenta de coleta e análise de resultados da pesquisa, como mencionado anteriormente no Capítulo 4. Ocorreu, inicialmente, a implantação para os desenvolvedores para os desenvolvedores realizarem testes. Isso permitiu a coleta de dados via entrevistas com dois integrantes já entrevistados anteriormente, e um novo integrante. Esse último conferiu uma visão muito relevante para a análise da pertinência desse estudo, e do ferramental de apoio produzido pelo autor.

Ocorreu ainda a coleta via formulário, conferindo retorno de outros integrantes que já haviam respondido o formulário anterior. Ambas as análises foram documentadas, seja no Capítulo 5, seja no presente capítulo. Ressalta-se, por fim, sobre a pertinência do trabalho, bem como sobre suas futuras melhorias.

8 Conclusão

Esse capítulo possui como intenção apresentar as considerações finais sobre esse trabalho, considerando a solução desenvolvida e os retornos referentes à utilização da solução, além de retomar as questões e objetivos inicialmente especificados no Capítulo 1. Inicialmente, há um breve resumo sobre o Contexto Geral definido para o trabalho e reflexões sobre o mesmo. Após isso, será exposto sobre o *status* atual do trabalho, retomando Questionamentos e Objetivos. Há destaque ainda para as Contribuições e Fragilidades da solução apresentada. Por fim, constam ideias para Trabalhos Futuros, visando evoluções para a solução atual.

8.1 Contexto Geral

Sistemas Legados são sistemas ainda muito presentes em empresas de grande porte com muitos anos de existência, muitas vezes sendo sistemas especialmente relevantes às atividades gerais das empresas. Mesmo com essa importância, esses sistemas comumente possuem uma grande deficiência em relação à documentação e ao código (FRITOLA; SANTANDER, 2022). Esses sistemas também possuem como característica um código-fonte muito longo, proveniente de anos de mudanças e adições, onde não é mais viável realizar reestruturações ou migrações desse código para aplicações mais modernas e com alto desempenho, seja por questões de tempo ou monetárias (FEW, 2006).

Sabendo disso, surgiu o interesse de prover uma solução que pudesse auxiliar o acompanhamento do código como um todo, assim como prover formas de incentivar e criar documentações de fácil acesso e utilização. Em termos de acesso e facilidade de uso, optou-se por incrementar a solução com o desenvolvimento de um *dashboard*. Nesse contexto, fez-se uso de um Estudo de Caso, cujo público alvo foi uma equipe específica de uma empresa de médio porte do domínio financeiro. Essa equipe reportou, ao longo da coleta de dados inicial desse projeto, vários problemas referentes aos sistemas legados, conforme exposto no Capítulo 5, e corroborado pela leitura de materiais teóricos, reunidos no Capítulo 2.

8.2 Questionamentos e Objetivos

A seguir, são retomadas as questões inicialmente apresentadas no trabalho, assim como os objetivos específicos e geral especificados para serem cumpridos ao longo do projeto.

8.2.1 Questão de Pesquisa

A questão de pesquisa, abordada no Capítulo 1, era referente a quais documentações são relevantes para facilitar o acompanhamento e a compreensão dos Sistemas Legados. Ao longo do trabalho, foram oferecidas diferentes ideias para viabilizar a documentação de *software*, provendo acompanhamento de um dado sistema como um todo. No Capítulo 2, há um embasamento sobre formas de documentação nesse sentido, com destaque para o Documento de Arquitetura (DAS), uma vez que o mesmo reuni várias visões arquiteturais em um só artefato. Esse embasamento consta acordado orientando-se por referências bibliográficas da área de interesse desse projeto. Já no Capítulo 6, ocorreu uma validação, junto aos integrantes de uma equipe específica da empresa em estudo, para compreender melhor sobre a pertinência do acompanhamento do *software* via documentos/artefatos. Na percepção desse grupo de interessados, a proposta de documentar, usando um DAS, é interessante, e pode complementar bem os projetos da empresa, além de facilitar a integração de novos membros sobre o escopo de cada sistema. Entretanto, reportam que a empresa não tem, em suas práticas já estabelecidas, a cultura de prover documentação sobre os sistemas.

Durante a pesquisa, também foi possível observar que o Documento de Arquitetura de *Software* e a Análise Estática de Código firmaram-se como as documentações mais relevantes para se analisar e acompanhar um sistema, ao longo do seu ciclo de visa. Portanto, ambos foram utilizados para compor a solução desenvolvida nesse trabalho.

8.2.2 Questão de Desenvolvimento

A questão de desenvolvimento abordada durante o trabalho questionava como prover um *dashboard* que permitiria auxiliar e acompanhar Sistemas Legados. Essa questão foi trabalhada com diferentes vieses, no decorrer desse trabalho. Destacam-se, entretanto: (i) no Capítulo 2, as boas práticas no desenvolvimento de um *dashboard*, visando eficiência, facilidade de uso, dentre outros critérios qualitativos. Aqui, ocorreu um levantamento junto à comunidade especializada, via Pesquisa Bibliográfica; (ii) no Capítulo 3, os ferramentais dedicados às atividades de desenvolvimento e teste (ex. linguagem de programação, frameworks, plataforma de testes, ferramentas de versionamento e hospedagem), e (iii) nos Capítulos 6 e 7, os vários insumos coletados com a realização do Estudo de Caso, cabendo mencionar as métricas e demais informações relevantes para o adequado acompanhamento e auxílio no que tange os Sistemas Legados. Ressalta-se ainda que a solução, com adequada exposição de suas particularidades (i.e. arquitetura, organização de pastas do código, protótipo, telas refinadas ao longo do desenvolvimento, dentre outros), encontra-se no Capítulo 5. Tais esforços revelam com clareza como foi provido o *dashboard* almejado como solução.

8.2.3 Objetivos Alcançados

Foram especificados como objetivos do trabalho:

- Estudo dos sistemas da Empresa A, tanto os Sistemas Legados quanto os sistemas mais modernos, com foco na documentação dos mesmos e na cultura da empresa, *Status*: Cumprido, apresentado no Capítulo 5;
- Levantamento das principais necessidades de documentação para os Sistemas Legados da empresa, considerando suas particularidades e limitações arquiteturais, *Status*: Cumprido, apresentado no Capítulo 5;
- Levantamento das principais métricas e tecnologias associadas, possivelmente, de análise estática de código, para auxiliar no acompanhamento e na compreensão desses Sistemas Legados, *Status*: Cumprido, apresentado nos Capítulo 2 e Capítulo 6;
- Estudo sobre Usabilidade em *dashboard*, *Status*: Cumprido, apresentado no Capítulo 2 - Seção 2
- Planejamento, Especificação, e Desenvolvimento de um *dashboard* para acompanhamento e compreensão desses Sistemas Legados usando como insumos os estudos e levantamentos anteriores, *Status*: Cumprido, apresentado no Capítulo 6, e
- Análise e Documentação dos resultados obtidos ao longo do trabalho, considerando o Estudo de Casos na Empresa A, *Status*: Cumprido, apresentado no Capítulo 7.

Atingindo-se os objetivos específicos, também é possível analisar o objetivo geral do trabalho, que era o acompanhamento facilitado de Sistemas Legados de uma empresa, procurando mitigar problemas como falta de documentação e dificuldades na compreensão desses sistemas, utilizando-se dos insumos coletados durante a fase de análise de resultados. Portanto, pode-se afirmar que o Objetivo Geral foi cumprido.

8.3 Contribuições e Fragilidades

Como mencionado anteriormente, a utilização de um Sistema Legado, principalmente sistemas como o Sistema X da Empresa A, é bastante complicada devido à total falta de documentação. O presente trabalho procurou conferir uma solução capaz de mitigar este problema. A solução, com base na análise dos resultados obtidos, foi bem aceita pelos integrantes da empresa. Portanto, deu-se um passo à frente, tendo a empresa hoje um suporte para apoiar o acompanhamento dos projetos orientando-se por artefatos relevantes (Documento de Arquitetura e Análise Estática de Código).

Pode-se apontar como contribuições extras: o engajamento gerado com a implantação da solução, onde percebeu-se diferentes reações dos membros da empresa. Em sua

maioria, reações muito positivas, confirmando o interesse no uso da aplicação na rotina de trabalho da empresa; a facilidade de acesso às informações, obtida via *dashboard* projetado orientando-se por boas práticas para torná-lo objetivo e intuitivo.

Entretanto, é possível apontar algumas fragilidades na solução. Uma delas dá-se pela pouca experiência do desenvolvedor com algumas ferramentas utilizadas na aplicação, aliado ao curto prazo conferido para realização do projeto como um todo. Isso dificultou evoluir algumas funcionalidades à tempo de atender os prazos acordados no cronograma de atividades, apresentado no Capítulo 4.

Há ainda questões referentes à não integração da aplicação com plataformas de armazenamento e versionamento de código, sendo essa uma funcionalidade sugerida durante o período de testes. Isso agregaria, com certeza, muito valor à solução. Por fim, ainda podem ser mencionadas como fragilidade o não atendimento de algumas sugestões de melhorias, apontadas pelos participantes na fase de análise de resultados. Entretanto, o motivo para não atendimento foi justamente o curto prazo para entrega do projeto. De toda forma, foram cumpridas as demandas planejadas pelo autor, bem como são anotadas como ideias para trabalhos futuros esses casos não atendidos a contento nessa primeira versão da solução.

8.4 Trabalhos Futuros

Considerando as fragilidades apontadas, há espaço para futuros trabalhos referentes à solução apresentada, conforme segue:

- Integração com plataformas de armazenamento e versionamento de código, para assim conseguir implementar várias funcionalidades que agregariam valor à aplicação. Dentre as funcionalidades viabilizadas com essa integração, têm-se o conhecimento quanto ao usuário que realizou um dado cadastro de funcionalidades/regras negociais; ou o conhecimento sobre os usuários que alteraram Documentos de Arquitetura ou reportes sobre análise estática de código, filtrando-os por tags e branches do sistema analisado;

- Aprimoramento visual como um todo da aplicação, após estudo mais aprofundado sobre as ferramentas de *Frontend*, com foco em apresentar uma solução mais elegante, ainda que mantendo a simplicidade e a facilidade de manuseio dos dados presentados;

- Busca de métodos ainda não mapeados para análise gradual de documentação do sistema;

- Ainda no contexto de a busca de métodos não mapeados, busca de métodos por palavras chave e cadastro desses métodos diretamente pelo **DASboard**, conferindo assim uma forma mais fácil de identificar e cadastrar na aplicação, e
- Forma de alterar entre diferentes projetos analisados com mais facilidade, podendo também alterar entre diferentes *Branches* do mesmo projeto em caso de integração com plataformas de armazenamento de código fonte.

Apêndices

APÊNDICE A – Termo de Consentimento

Você está sendo convidado(a) para participar, como voluntário(a), em uma pesquisa científica. Caso você não queira participar, não há problema algum. Você não precisa explicar porque, e não haverá punição por isso. Você tem todo o direito de não querer participar do estudo, basta fechar o formulário. Este TCLE refere-se ao TCC cujo objetivo é Entender o Estado de Documentação em Sistemas Legado de uma empresa. A coleta de dados para a presente pesquisa será realizada por meio desse Questionário online, constituído por 2 Seções, com perguntas (de múltipla escolha e/ou discursivas). Estima-se que você precisará de aproximadamente 5 minutos. A precisão de suas respostas é determinante para a qualidade da pesquisa. Você não será remunerado, visto que sua participação nesta pesquisa é de caráter voluntária. Caso decida desistir da pesquisa, você poderá interromper o que preenchimento do formulário, e sair do mesmo a qualquer momento, sem nenhuma restrição ou punição, bastando não enviar as respostas já preenchidas. PRINCIPAIS BENEFÍCIOS DO ESTUDO: Melhorias contínuas no TCC, o qual tem como base Documentação de Sistemas Legados. Os dados coletados serão utilizados para fins de divulgação dos resultados obtidos em Monografias de TCCs, bem como em publicações em Eventos Científicos, como embasamento/insumos para os estudos acadêmicos realizados. Para entrar em contato, por gentileza, encaminhe um email para: gabrielhussein83@gmail.com

Eu, concordo em participar voluntariamente do presente estudo como participante. Poderei deixar o preenchimento do Questionário, a qualquer momento, sem dar explicação, bastando apenas não enviar o formulário. Esta decisão não me trará penalidade. Posso imprimir ou gerar um pdf do TCLE e do Questionário para ter uma cópia.

Referências

ABNT. Associação brasileira de normas técnicas. 2023. Disponível em: <https://www.abnt.org.br/>. Citado na página 35.

ANGULAR. Angular. 2023. Disponível em: <https://angular.io/>. Citado na página 43.

ANNETT, R. *Working with Legacy Systems: A practical guide to looking after and maintaining the systems we inherit*. [S.l.]: Packt, 2019. v. 1. Citado 2 vezes nas páginas 21 e 22.

ATLASSIAN. Kanban: How the kanban methodology applies to software development. 2023. Disponível em: <https://www.atlassian.com/agile/kanban>. Citado na página 57.

ATLASSIAN. What is scrum and how to get started. 2023. Disponível em: <https://www.atlassian.com/agile/scrum>. Citado na página 57.

BASS, L.; KAZMAN, R.; CLEMENTS, P. *Software Architecture in Practice*. [S.l.]: Addison-Wesley, 2013. v. 3. Citado na página 34.

BENNETT, K.; RAJLICH, V. Software maintenance and evolution: A roadmap. 2000. Disponível em: <http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalbennett.pdf>. Citado na página 30.

BIZAGI. Bizagi. 2023. Disponível em: <https://www.bizagi.com/en/platform/modeler>. Citado na página 47.

CROTTY, J.; HORROCKS, I. Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. 2017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210832716301260>. Citado na página 29.

FERREIRA, M. G.; LEITE, J. C. S. do P. Requirements engineering with a perspective of software evolution. 2013. Disponível em: <https://ceur-ws.org/Vol-1005/erbr2013_submission_31.pdf>. Citado na página 31.

FEW, S. *Working with Legacy Systems: A practical guide to looking after and maintaining the systems we inherit*. [S.l.]: O'Reilly Media, 2006. v. 1. Citado 3 vezes nas páginas 21, 24 e 109.

FIGMA. Figma. 2023. Disponível em: <https://www.figma.com/>. Citado na página 46.

FOWLER, M. A guide to material on martinfowler.com about software architecture. 2019. Disponível em: <https://martinfowler.com/architecture/>. Citado 6 vezes nas páginas 23, 32, 33, 34, 38 e 39.

FRITOLA, R.; SANTANDER, V. Documentando requisitos de sistemas legados: um estudo de caso utilizando técnicas da engenharia de requisitos orientada a objetivos. Porto Alegre, Brasil, 2022. Disponível em: <https://sol.sbc.org.br/index.php/eres/article/view/18459/18292>. Citado 3 vezes nas páginas 21, 29 e 109.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: GoF, 1994. v. 1. Citado na página 33.

GIL, A. *Como Elaborar Projetos de Pesquisa*. [S.l.]: Atlas, 2002. v. 4. Citado 8 vezes nas páginas 25, 26, 54, 55, 56, 59, 63 e 64.

GOOGLE. Firebase realtime database. 2023. Disponível em: <https://firebase.google.com/docs/database>. Citado na página 44.

GOOGLE. Google forms. 2023. Disponível em: <https://workspace.google.com/intl/pt-BR/lp/forms/>. Citado 2 vezes nas páginas 45 e 64.

JONES, P. *Modernizing Legacy Applications in PHP*. [S.l.]: O'Reilly Media, 2015. v. 1. Citado na página 23.

LARMAN, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. [S.l.]: Pearson, 2004. v. 3. Citado na página 34.

LATEX. The latex project. 2023. Disponível em: <https://www.latex-project.org/>. Citado na página 48.

LEMOS, M. A. Qualidade na manutenção. 2011. Disponível em: <https://abepro.org.br/biblioteca/enegep2011_tn_sto_135_859_18052.pdf>. Citado na página 31.

LUCID. Lucidchart. 2023. Disponível em: <https://www.lucidchart.com/pages/>. Citado na página 46.

MARIOTTI, F. S. Como documentar a arquitetura de software. 2012. Disponível em: <http://www.linhadecodigo.com.br/artigo/3343/como-documentar-a-arquitetura-de-software.aspx>. Citado na página 35.

MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. [S.l.]: Prentice Hall, 2018. v. 1. Citado 3 vezes nas páginas 32, 38 e 39.

MCCABE, T. A complexity measure. IEEE, 1976. Disponível em: <https://ieeexplore.ieee.org/document/1702388>. Citado na página 38.

MICROSOFT. Visual studio code. 2023. Disponível em: <https://code.visualstudio.com/>. Citado na página 47.

NNG. Parallel and iterative design + competitive testing = high usability. 2011. Disponível em: <https://www.nngroup.com/articles/parallel-and-iterative-design/>. Citado 2 vezes nas páginas 41 e 79.

NNG. Best application designs. 2012. Disponível em: <https://www.nngroup.com/articles/best-application-designs/>. Citado na página 79.

NNG. Dashboards: Making charts and graphs easier to understand. 2017. Disponível em: <https://www.nngroup.com/articles/dashboards-preattentive/>. Citado 2 vezes nas páginas 40 e 41.

NNG. Data visualizations for dashboards. 2021. Disponível em: <https://www.nngroup.com/videos/data-visualizations-dashboards/>. Citado na página 40.

OLIVEIRA, M.; CARDOSO, E. Medição de usabilidade e eficiência de dashboards: Framework desenvolvida usando os princípios de desenho de business intelligence e interação homem-máquina. 15ª conferência da associação portuguesa de sistemas de informação. Lisboa, Portugal, 2015. Disponível em: <https://repositorio.iscte-iul.pt/bitstream/10071/27763/1/conferenceobject_27694.pdf>. Citado na página 24.

ORACLE. Java. 2023. Disponível em: <https://www.oracle.com/java/>. Citado na página 44.

PRESSMAN, R.; MAXIM, B. *Engenharia De Software: Uma Abordagem Profissional*. [S.l.]: Bookman, 2021. v. 8. Citado na página 37.

REENSKAUG, T. Models-views-controllers. 1979. Disponível em: <https://folk.universitetetioslo.no/trygver/1979/mvc-2/1979-12-MVC.pdf>. Citado na página 33.

SONARQUBE. Sonarqube. 2023. Disponível em: <https://docs.sonarsource.com/sonarqube/latest/>. Citado 3 vezes nas páginas 39, 45 e 65.

SONARSOURCE. Sonarsource. 2023. Disponível em: <https://next.sonarqube.com/sonarqube/web_api/api/alm_integrations>. Citado na página 84.

STAKE, R. E. *The Art of Case Study Research*. [S.l.]: Sage, 1995. v. 1. Citado na página 59.

TEAMS, M. Microsoft teams. 2023. Disponível em: <https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>. Citado na página 47.

TEBALDI, P. C. O que é um dashboard? o guia completo e definitivo! 2017. Disponível em: <https://www.opservices.com.br/o-que-e-um-dashboard/>. Citado na página 39.

TELEGRAM. Telegram. 2023. Disponível em: <https://telegram.org/>. Citado na página 48.

YIN, R. K. *Estudo de caso: planejamento e métodos*. [S.l.]: Bookman, 2001. v. 2. Citado 2 vezes nas páginas 59 e 65.