CENTRO TECNOLÓGICO

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO TÍTULO DE BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO.

Wesley Mayk Gama Luz

Implementação de Métricas Ágeis integradas à Plataforma Jira

Florianópolis

2024

Wesley Mayk Gama Luz

Implementação de Métricas Ágeis integradas à Plataforma Jira

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina para obtenção do título de Bacharel em Ciências da Computação.
Orientador: Prof. Dr. Ricardo Pereira e Silva

Florianópolis
2024

Este trabalho é dedicado a meu pai, Eduardo Muniz Luz (*in memoriam*). Que seu legado como cristão, pai e esposo fique para sempre em nossos corações.

AGRADECIMENTOS

Em primeiro lugar a Deus, que me capacitou, guiou e sustentou diante de todos os anos de estudos e em meio à todas as dificuldades enfrentadas.

Ao meu orientador Ricardo Pereira e Silva, por topar fazer parte das ideias que deram origem ao tema, se fazendo presente durante todo o desenvolvimento deste trabalho. Se colocando à disposição para tirada de dúvidas e me auxiliando a tomar as melhores decisões.

Aos membros da banca, Dr. Professor Jean Hauck e Professor Dr. Professor Raul Wazlawick, por aceitarem fazer parte da banca avaliadora.

À minha esposa, que tem me dado todo apoio e incentivo durante o percurso.

À minha família, que sempre me deram todo suporte que puderam durante minha graduação. Em especial à minha irmã, que apostou em minha educação desde pequenino.

À minha equipe de trabalho, que me proporcionou vivências e aprendizados que me permitiram se apaixonar pelas áreas que deram origem ao tema deste T.C.C.

"Não importa o que aconteça, continue a nadar."
(STANTON, Andrew. 2003 - PROCURANDO NEMO)

RESUMO

Caracterizado por uma expansão nos últimos anos, o mercado de produção de software convive com a crescente demanda por processos e metodologias que comportem uma ascenção em relação à qualidade do software produzido. As metodologias ágeis têm se disseminado cada vez mais nas organizações modernas e, portanto, métricas, medidas e indicadores ágeis são abordagens cada vez mais necessárias para a avaliação dos resultados. Para gerir os referidos processos, as equipes relacionadas e respectivas tarefas diversas plataformas são utilizadas, tendo em vista que a gestão de forma manual pode se tornar custosa, lenta e ineficaz. Dentre elas, neste trabalho destaca-se a plataforma Jira, que possui vasto ferramental para gerenciamento de projetos, especialmente quando utilizantes de *Scrum* e *Kanban*. A coleta de métricas não presentes de forma nativa na Plataforma Jira, de forma automatizada, pode agilizar e auxiliar na gestão. Neste trabalho é proposta uma ferramenta que visa coletar dados dos projetos de forma automatizada, integrada à Plataforma Jira, com o objetivo de estabelecer métricas ágeis signficativas, acompanhar em tempo real o alcance de objetivos macro da organização, evidenciar possíveis melhorias de fluxo, identificar possíveis gargalos e agregar melhorias em equipes em ambientes ágeis.

Palavras-chave: Processos de produção de Software. Abordagens ágeis. Automatização de avaliação de processos. Qualidade de software. Métricas, medidas e indicadores ágeis. SCRUM. Kanban. Jira.

ABSTRACT

Characterized by an expansion in recent years, the software production market coexists with the growing demand for processes and methodologies that entail an ascension in relation to the quality of the software produced. Agile methodologies have been increasingly disseminated in modern organizations and, therefore, agile metrics, measures and indicators have become increasingly necessary approaches for evaluating results. To manage said processes, the related teams and respective tasks, several platforms are used, considering that manual management can become costly, slow and ineffective. Among them, in this work stands out the Jira platform, which has vast set of tools for project management, especially when using SCRUM and Kanban. Collecting metrics that are not natively present on the Jira Platform, in an automated way, can streamline and assist in management. In this work a tool is proposed that aims to collect project data in an automated way, integrated with the Jira Platform, with the objective of establishing meaningful agile metrics, monitoring in real time the achievement of the organization's macro objectives, highlighting possible flow improvements, identify potential bottlenecks and add improvements to teams in agile environments.

Keywords: Software production procesess. Agile approaches. Process evaluation automation. Software quality. Metrics, measures and agile indicators. SCRUM. Kanban. Jira.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O mercado de produção de software se intensificou a partir da década de 1960 ao redor do mundo. Com o advento dos microcomputadores na década de 1970 a demanda por produção de software cresceu ainda mais, fazendo com que todos os processos que envolvem a dinâmica existente nesse contexto fossem cada vez mais elaborados, estudados e melhorados. Esse movimento deu origem ao termo atualmente conhecido como "Engenharia de Software". O referido termo surgiu numa tentativa de contornar a crise do software, marcada por software de baixa confiabilidade, manutenibilidade precária e desempenho um tanto quanto questionável (PRESSMAN, 2021). A atribuição desse termo ao contexto de desenvolvimento de software trouxe consigo um tratamento caracterizado por semelhanças à processos comuns da engenharia, que por sua vez são caracterizados por possuir um planejamento sistemático, produção controlada, avaliação constante e qualidade mensurável.

A avaliação de processos de software é um procedimento de medição subjetivo que envolve o julgamento de pessoas qualificadas para identificação quantitativa de pontos fortes e fracos nos processos (EMAM, 1998). A prática da avaliação geralmente possui o objetivo de se ter ciência dos resultados cumulativos com base em dados e amostragens do passado, e também um panorama da situação atual com dados recentes dos processos executados pelos times de uma organização. A partir disto são determinados pontos de maior prioridade a serem melhorados considerando, principalmente, as metas de melhoria para o futuro de uma organização.

A partir da década de 1990 toda a sistematica de produção de software, assim como sua burocracia e vagarosidade herdada dos processos de engenharia, foi modificada com o surgimento das denominadas metodologias ágeis para desenvolvimento de software. O termo “ágil”, nesse caso, indica um estado de prontidão, com reações rápidas às mudanças e maior capacidade de adaptação (MARTIN, 2020) agregando uma nova perspectiva acerca do desenvolvimento de software juntamente com uma mudança cultural. Promover a cultura ágil consiste em disseminar cotidianamente um conjunto de valores, princípios e práticas que flexibilizam o desenvolvimento de software com o intuito de responder às constantes mudanças do mercado, enfocando no que é realmente importante (LARMAN, 2003).

Em contextos ágeis e não ágeis as tomadas de decisões significativas estão presentes em diversos momentos do processo de desenvolvimento de software, entretanto, nos contextos ágeis essas decisões necessitam ser tomadas com ainda mais rapidez para manter o ritmo de vazão e adaptação à mudanças do projeto.

Um dos principais objetivos da medição é apoiar a tomada de decisão em relação a projetos e processos, de modo a também viabilizar o alcançe aos objetivos organizacionais. Segundo a ISO/IEC 12207 (ISO/IEC, 2008) o propósito da Medição é coletar e analisar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar o efetivo gerenciamento dos processos e demonstrar objetivamente a qualidade dos produtos.

O *Scrum* é um *framework* para gestão de projetos na qual é possível empregar técnicas

e processos, de maneira a tornar clara a eficaz as práticas de gestão de produto e do desenvolvimento. (SUTHERLAND, 2016).Notado por se adequar muito bem à qualquer natureza de projeto, é também popularmente utilizado para trabalhar com projetos complexos e extensos. Dentre os frameworks difundidos atualmente o SCRUM é o mais comumente utilizado por organizações de desenvolvedores de softwares e sistemas. Caracterizando-se por um método de gestão de trabalho realizado a partir de pequenos ciclos de atividades dentro de um projeto. Cada ciclo de atividade é planejado previamente e se chama *Sprint*, composto por um período de tempo predefinido em que as tarefas devem ser realizadas pela equipe. Essa maneira de gestão permite potencializar o trabalho em equipe, acompanhar a evolução do produto, sempre com foco na qualidade da produção e nos prazos estipulados. Visa conseguir mais resultados com menos gente, tempo e recursos, mas com qualidade melhor (SUTHERLAND, 2016). Os ciclos de atividades dentro de um projeto são capazes de evidenciar à gerencia das equipes e da organização diversos dados que, quando bem avaliados e medidos, rendem indicadores extremamente úteis sobre os produtos, projetos e processos.

Segundo McGarry (MCGARRY, 2002), a medição é o nível mais importante em um projeto. A medição de software ajuda o gerente de projetos a fazer um trabalho melhor. Ajuda a definir e implementar planos mais realistas, alocar adequadamente recursos escassos para implementar esses planos e monitorar com precisão o progresso e o desempenho em relação a esses planos.

Segundo a ISO/IEE 15939 (ISO/IEE, 2017), uma medida é definida em termos de um atributo e o método para quantificá-lo. Uma medida base é funcionalmente independente de outras medidas e captura informações sobre um único atributo. A medida derivada é uma medida definida como uma função de dois ou mais valores de medidas base, de modo a capturar informações sobre mais de um atributo ou o mesmo atributo de várias entidades.

A métrica ágil deve reforçar princípios ágeis, medir resultados e não saídas, seguir tendências e não números, responder uma pergunta específica para uma pessoa real e pertencer a um conjunto pequeno de métricas e diagnósticos (HARTMANN, 2006).

Visando auxiliar na organização diária diversas equipes em ambientes de desenvolvimento ágil fazem uso do método *Kanban*. Este método é uma abordagem enxuta para o desenvolvimento de software ágil. Trata-se de uma palavra japonesa que significa "cartão visual". Portanto, no contexto de desenvolvimento de software ele é relativamente novo, enquanto no processo produtivo o termo já é utilizado há mais de meio século (ALBINO, 2017). Conforme Boeg (2010), a maioria dos especialistas concorda que o *Kanban* é um método de gestão de mudanças, dando ênfase em alguns princípios, como visualizar o trabalho em andamento, limitar o trabalho em progresso e tornar explícitas as políticas sendo seguidas.

Diversas organizações vêm implantando a OKR para atingir objetivos organizacionais. OKR (do inglês, Objective and Key-Results), é um framework de definição e gerenciamento de objetivos. O framework tem dois componentes básicos: o objetivo (o que queremos alcançar) e um conjunto de resultados-chaves (como saber se estamos chegando lá) (CASTRO, 2015). O propósito do OKR é criar alinhamento na organização. Assim, objetivos ambiciosos

e resultados-chaves mensuráveis são estabelecidos e precisam ser visíveis para todos os níveis da organização (NIVEN, 2016). Entretanto, nem sempre esses resultados-chaves são visualizados e sensibilizam as equipes envolvidas de maneira efetiva e diariamente. Para isso, existem ferramentas de gestão que buscam auxiliar nesse processo.

Lançado em 2002, pela empresa Australiana Atlassian, a plataforma de código fechado Jira tem se consolidado como uma das principais ferramentas de gestão ágil de projetos de qualquer natureza, oferendo suporte a diversas metodologias ágeis, como *Scrum* e *Kanban*. Essa ferramenta de gerenciamento de projetos tem sido vastamente disseminada no contexto equipes de desenvolvimento de software. Destacando-se por disponibilizar de maneira clara e facilitada o monitoramento de tarefas e acompanhamento dos projetos garantindo o gerenciamento de todas as demandas, subdemandas e atuantes em um único local.

Apesar das ferramentas e técnicas que viabilizam um ganho no gerenciamento das tarefas e projetos, as equipes de desenvolvimento frequentemente passam por dificuldades quando buscam obter medidas e indicadores claros e de maneira facilitada. Uma vez que, através de toda a massa de dados armazenados dos times e projetos ao longo do tempo, deveria ser possível abstrair informaões relevantes que contribuam para aumentar eficiência, melhorar estimativas, diminuir gargalos, mitigar riscos, auxiliar em decisões significativas e evidenciar todo o esforço empregado.

Este trabalho visa o auxílio à gestão e tomada de decisão em equipes de desenvolvimento ágeis que utilizam a Plataforma Jira, através do desenvolvimento de uma ferramenta integrada à API disponível do Jira para coletar e processar uma massa de dados já existente de projetos de uma organização real que utiliza o Jira diariamente, de modo a viabilizar a apresentação de metricas e indicadores significatos de maneira facilitada e intuitiva em uma aplicação de fácil acesso.

Com isso, espera-se que a partir da coleta de dados de tarefas, subtarefas, projetos, desenvolvedores, sprints e outras informações presentes no Jira da organização, aplicados a necessidade de cada equipe ágil e demanda de alta gestão, seja possível gerar indicadores e resultados significativos que auxiliarão as equipes e gestores em vários quesitos de melhorias nos processos, otimização de estimativas e desenvolvimento das tarefas.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Através da análise e implementação de métricas ágeis utilizadas no gerenciamento de projetos por equipes ágeis, a partir de dados coletados de forma automatizada por meio de integração junto à plataforma Jira, busca-se implementar uma ferramenta que tem como objetivo automatizar o processo de análise do histórico de dados, de acordo com cada equipe, possibilitando uma obtenção de indicadores significativos através de interpretação das métricas. Viabilizando, portanto, uma melhor visualização dos processos e fluxos de software nos ambientes dos times e, consequentemente, uma avaliação facilitada. Busca-se também alimentar em tempo real estruturas de obtenção de resultados para acompanhamento de objetivos macro da organização, tais como *Key Results* ligadas aos times de software.

A aplicação desenvolvida será utilizada em equipes ágeis da empresa de tecnologia específica, à qual o autor do trabalho é colaborador. Essas equipes utilizam metodologia *Scrum* para cerimonias e cotidiano, alinhada ao método *Kanban* nos seus processos internos de desenvolvimento, e OKR para acompanhamento e regimento de objetivos macro a serem alcançados.

1.1.2 Objetivos Específicos

- Realizar a análise das métricas ágeis utilizadas por equipes em ambientes de desenvolvimento ágil.
- Analisar e modelar uma ferramenta com base na análise de métricas significativas para equipes ágeis.
- Desenvolver uma Ferramenta Web com integração à plataforma Jira.
- Avaliar a ferramenta desenvolvida, sob perspectiva do autor, em sua utilização por equipes ágeis de desenvolvimento de software na organização-alvo.

1.2 MÉTODO DE PESQUISA

A construção de conhecimento para a realização do trabalho se dá através de pesquisa em artigos científicos, materiais didáticos, publicações, dissertações e teses com informações relevantes e recentes da área de pesquisa em bases conhecidas como ACM Digital Library, IEEExplore, bem como indexadores tais como DBLP e Google Scholar.

A pesquisa que fora realizada, caracteriza-se por uma pesquisa quantitativa e exploratória que visa embasar capítulos como Introdução, Fundamentação Teórica e Revisão Sistemática. A partir do conhecimento construído por meio da pesquisa busca-se quantificar problemas práticos de organizações por meio da geração de dados numéricos ou informações que possam ser transformadas em estatísticas utilizáveis, demonstração de padrões/comportamentos e ou tendências dos fatores analisados.

Foram executadas etapas de Análise de Requisitos, Modelagem e Desenvolvimento de uma ferramenta/aplicação, que leve em consideração os estudos realizados, com o propósito de levantar indicadores e métricas ágeis sobre as equipes.

Haverá também a etapa de avaliação da utilidade e funcionalidade da ferramenta desenvolvida. Nesta etapa busca-se definir o objetivo da avaliação, planejar a avaliação, executar a avaliação e consequentemente analisar os resultados obtidos.

O desenvolvimento tende a ser realizado com recursos pessoais e da universidade. Isto é, máquinas, espaço e conexão à rede.

1.3 ESTRUTURA DO TRABALHO

O trabalho está estruturado da seguinte forma:

O capítulo 1 apresenta uma introdução geral sobre todos os pontos relevantes que embasam este trabalho, tão como os objetivos macro e micro que deseja-se alcançar, além de breve resumo sobre o método de pesquisa utilizado para desenvolvimento do trabalho. O capítulo 2 apresenta os conceitos e informações que fundamentam teoricamente este trabalho. O capítulo 3 apresenta a maneira como foi avaliado o estado da arte dos tópicos abordados, através da revisão sistemática. No capítulo 4 é definida a proposta do trabalho, tão como os objetivos que deseja-se alcançar com a mesma.

No capítulo 5 é feito o levantamento de requisitos e concepção inicial da modelagem da ferramenta. O capítulo 6 apresentará toda a descrição da fase de desenvolvimento da ferramenta. O capítulo 7 discorrerá acerca da análise e avaliação dos resultados obtidos através da utilização da ferramenta construída. E, por fim, o capítulo 8 aborda a conclusão do trabalho e respectivas sugestões de evoluções da ferramenta desenvolvida em formato de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados de maneira conceitual os principais tópicos abordados ao longo deste trabalho. Através de revisão bibliográfica com enfoque nos temas que permeiam a Engenharia de Software, Metodologias Ágeis e Métricas de Software, busca-se fundamentar os conceitos e abordagens utilizados.

2.1 PROCESSO DE PRODUÇÃO DE SOFTWARE

O conceito de "processo" pode ser definido como um conjunto de atividades inter-relacionadas ou interativas que transformam entradas em saídas (ISO/IEC, 2011). Quando aplicado ao contexto de produção de software, um "processo de software" pode ser descrito como uma coleção de padrões que definem um conjunto de atividades, subatividades, ações, comunicações, comportamentos, tarefas de trabalho, subtarefas de trabalho e produtos de trabalho necessários ao desenvolvimento de softwares de computador (PRESSMAN, 2010). Ou ainda, de maneira mais simplificada, como um conjunto estruturado de atividades exigidas para desenvolver um sistema de software (SOMMERVILLE, 1995).

Humphrey define as seguintes razões para a definição e aplicação cotidiana de processos padronizados em uma organização de software (HUMPHREY, 1989).

- Redução dos problemas relacionados a treinamento, revisões e suporte à ferramentas.
- As experiências adquiridas nos projetos são incorporadas ao processo padrão e contribuem para melhorias em todos os processos definidos.
- Economia de tempo e esforço na definição de novos processos adequados aos projetos.

Diversas organizações adotam modelos de processos com o intuito de estabelecer padrão sobre seus processos, de modo a evitar variações indesejadas em suas práticas de produção de software. Um modelo de processo apresenta uma filosofia, ou uma forma geral de comportamento, baseada na qual processos específicos podem ser definidos. Assim, quando uma empresa decide adotar um processo, ela deve inicialmente buscar um modelo de processo e adaptar a filosofia e práticas recomendadas para criar seu próprio processo. A partir daí, todos os projetos da empresa deverão seguir este processo definido (WAZLAWICK, 2013)

O estabelecimento de processos de software padronizados dentro de uma empresa tem um papel fundamental para conseguir a garantia de qualidade do produto desenvolvido. Pela combinação de padrões, uma equipe de software pode construir um processo que melhor satisfaça às necessidades de um projeto (PRESSMAN, 2010). Portanto, a utilização de processos padronizados é um fator extremamente relevante para o alcançe de objetivos da organização, de forma controlada, alinhados à uma maior produtividade.

2.1.1 Avaliação de Processos de Software

A implementação de modelos de processos de software somente não basta para galgar melhoras em muitos dos aspectos de uma organização. É necessário, também, que os modelos de processos adotados sejam frequentemente avaliados em busca de melhorias constantes. A avaliação geralmente ocorre com o objetivo de se conhecer a situação atual dos processos executados pela organização. Assim, são determinados pontos de maior prioridade a serem melhorados considerando, principalmente, as metas de melhoria da organização e os benefícios que cada melhoria pode introduzir (ANACLETO; WANGENHEIM; SALVIANO, 2005).

A ISO (ISO/IEC, 2011) determina a avaliação de processo como uma avaliação disciplinada dos processos de uma unidade organizacional em relação a um modelo de avaliação de processos. Um modelo de avaliação de processos é um modelo adequado para avaliar a capacidade do processo, com base em um ou mais modelos de referência de processo.

Como este trabalho irá abordar processos de software estabelecidos de forma autônoma por equipes ágeis da organização foco desse estudo, é importante ressaltar que, assim como aponta Sommerville (SOMMERVILLE, 2010), "Não existe algo como um processo de software 'ideal' ou 'padrão' que seja aplicável a todas as organizações ou para todos os produtos de software de um tipo particular". Portanto, no desenvolvimento deste trabalho não será utilizado um modelo de referência específico para a avaliação de processo e a avaliação a ser realizada vai de acordo com aspectos interpretativos do que a organização foco deste trabalho necessita ou pode ser melhorada.

2.1.2 Automatização de avaliação de processos de software

A implementação de automatização de avaliação de processos de software pode ocorrer de forma parcial ou de forma total. Em ambos os modos, é possível discorrer acerca de aspectos positivos e negativos.

A avaliação totalmente automatizada parte do princípio de que os indicadores de avaliação são quase que totalmente pré-definidos, integrados a um processo de software maduro e a avaliação é feita usando critérios também já pré-definidos para interpretar os dados de medição. Organizações que possuem um processo estatisticamente estável podem consideradar inclusive os limites de controle para o processo como critérios de avaliação na automação da avaliação (JARVINEN, 2000). Esse cenário tende a funcionar apenas em condições consideravelmente limitadas dado que essa maturidade de processos não é regra dentre as organizações. Os critérios costumam ser incorporados em uma ferramenta, como um sistema especialista ou software de terceiros, que muitas das vezes funcionam de maneira fixa e sem muita margem à alterações, de modo a dificultar organizações que necessitam de alguma dinamicidade nos processos e nos indicadores de avaliação. Visto algumas das limitações da avaliação totalmente automatizada, Jervinen (JARVINEN, 2000) destaca algumas vantagens na avaliação parcialmente automatizada: "A avaliação parcialmente automatizada parece mais promissora do que tentar

automatizar totalmente a avaliação".

A avaliação parcialmente automatizada, por permitir uma flexibilidade maior em seu processo, costuma fazer uso de medições de software como evidência de apoio para a avaliação da capacidade de software, de modo a abrir novas possibilidades de avaliação. As avaliações podem ser realizadas com menor frequência, conforme demanda sazonal de gestores, ou com maior frequência, pois os dados podem ser coletados continuamente sob demanda (BOSSE, 2019). O custo da avaliação é reduzido, uma vez que menos pessoas são necessárias para realizar uma alteração no modo que a avaliação ocorre e as próprias avaliações, quando continuamente executadas, se tornam parte integrante do trabalho das equipes (BOSSE, 2019). As principais dificuldades associadas à avaliação parcialmente automatizada dizem respeito ao fato de que coletar dados de medição de software possuem custo considerável e construir a coleta de medições como parte do processo de software demanda tempo (JARVINEN, 2000).

De acordo com o que foi visto, nesse trabalho opta-se pela avaliação parcialmente automatizada como ponto chave, especialmente dado o cenário e contexto da organização foco para utilização da ferramenta.

2.2 ABORDAGENS ÁGEIS

Nos dias de hoje, as empresas operam em um ambiente global, com mudanças rápidas e frequentes. Deste modo, precisam responder a novas oportunidades e novos mercados, a mudanças nas condições econômicas e ao surgimento de produtos e serviços concorrentes. Softwares fazem parte de quase todas as operações de negócios, assim, novos softwares necessitam ser desenvolvidos rapidamente para obterem proveito de novas oportunidades e responder às pressões competitivas (SOMMERVILLE, 2010).

Ainda neste âmbito, de acordo com Sommerville (SOMMERVILLE, 2010), processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida, projetar, construir e testar o sistema não estão adaptados ao desenvolvimento rápido de software. Com as mudanças nos requisitos ou a descoberta de problemas nos requisitos, o projeto do sistema ou sua implementação precisa ser refeito ou retestado. Como consequência, um processo convencional em cascata ou baseado em especificações costuma ser demorado, e o software final é entregue ao cliente bem depois do prazo acordado.

Os modelos ágeis de desenvolvimento de software seguem uma filosofia diferente dos modelos prescritivos. Apesar de serem usualmente mais leves, é errado entender os métodos ágeis como modelos de processos meramente menos complexos ou simplistas. Não se trata de apenas de simplicidade, mas de focar mais nos resultados do que no processo (WAZLAWICK, 2013).

Os princípios de desenvolvimento de software ágil que são seguidos e defendidos surgiram dos princípios tradicionais de desenvolvimento de software e de várias experiências baseadas nos sucessos e fracassos dos projetos de software (RAO; NAIDU; CHAKKA, 2011).

Conforme o Manifesto Ágil (BECK et al., 2001), O movimento ágil não é anti-metodologia; Na verdade, é a restauração da credibilidade da palavra. Também é a restauração de um equilíbrio: adotar a modelagem, mas não apenas para arquivar alguns diagramas em um repositório corporativo empoeirado.

Determinados pilares do Manifesto Ágil (BECK et al., 2001) afirmam o seguinte:

"Estamos descobrindo maneiras melhores de desenvolver software fazendo isso e ajudando os outros a fazer isso. Valorizamos:

- Indivíduos e interações sobre processos e ferramentas.
- Software funcionando sobre documentação completa.
- Colaboração do cliente sobre negociação de contrato.
- Responder às mudanças sobre seguir um plano."

Portanto, este trabalho irá focar em equipes ágeis, ou seja, equipes de desenvolvimento de software que fazem uso das abordagens ágeis no dia a dia. Essas equipes foco utilizam princípios de *Scrum*.

2.2.1 Scrum

O *Scrum* é uma estrutura de gerenciamento de projetos ágeis que tem por objetivo gerenciar os processos de desenvolvimento de software por um viés focado nas pessoas e aderente a ambientes em que requisitos surgem e se alteram de maneira rápida e com alguma frequência (SCHWABER; BEEDLE, 2002).

O *Scrum* prevê alguns artefatos, sendo os principais e mais difundidos: *Product Backlog* e *Sprint Backlog*.

O *Product Backlog* é o documento mutável definido pelo cliente no início do projeto, onde estão contidas suas características esperadas em forma de lista ordenada por prioridade. O *Sprint Backlog* é um documento gerado na reunião de planejamento com a divisão do *Product Backlog* em partes para serem implementados (SBROCCO; MACEDO, 2012).

Os processos do *Scrum* se dividem em várias iterações chamadas de *Sprint*. O processo iterativo tem o objetivo de minimizar riscos, oferecendo aos usuários uma rápida avaliação do que está sendo construído. É estabelecido uma faixa de tempo fixo, onde o time deve trabalhar para atingir o objetivo especificado para a iteração.

O *Scrum* prevê que dentro de cada *sprint* ocorram algumas cerimônias, sendo as principais e mais disseminadas: *Sprint Planning, Sprint Review, Daily Scrum* e *Sprint Retrospective*.

A cerimônia de *Sprint Planning* costuma ser realizada no primeiro dia do Sprint, nesse primeiro planejamento o líder do processo, chamado de *Scrum* master, repassa com a equipe as tarefas que necessitam ser avaliadas e estimadas, tendo em vista que as mesmas já passaram por alinhamentos e priorizações junto ao cliente através da atuação em conjunto ao *Product Owner* (representa os interesses de todos os envolvidos no sucesso do projeto, também conhecidos como *Stakeholders*). Nessa reunião, a equipe deve definir quanto trabalho será necessário para cada tarefa baseado em diversos fatores de análise, e então é decidido qual das demandas será realizada no sprint em questão.

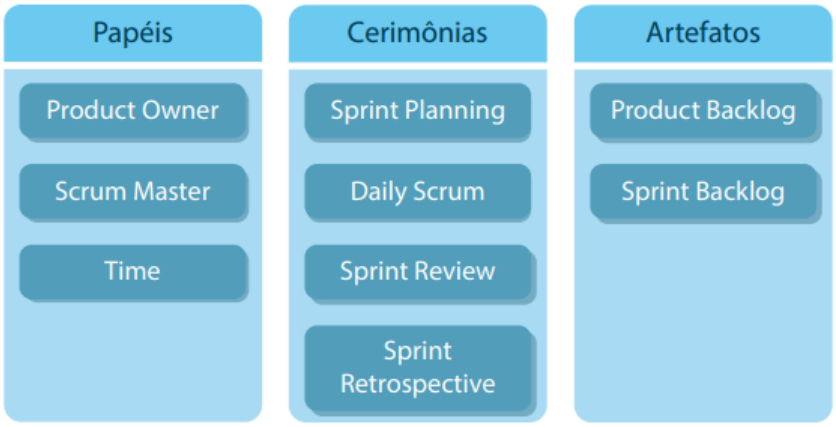
Durante todos os dias do sprint são realizadas reuniões diárias (*Daily Scrum*). Toda a equipe participa das reuniões diárias para a manutenção do foco da equipe. Durante a reunião, todos os membros da equipe compartilham informações, descrevem seu progresso desde a última reunião, os problemas que têm surgido e o que está planejado para o dia seguinte. Isso garante que todos na equipe saibam o que está acontecendo e, se surgirem problemas, poderão replanejar o trabalho de curto prazo para lidar com eles (SOMMERVILLE, 2010).

No último dia do sprint, é feita uma reunião de entrega e fechamento do sprint, que validará os itens concluídos no ciclo que se fecha e poderá dar novos direcionamentos ao ciclo que se seguirá. Essa reunião é chamada de reunião de revisão (*Sprint Review*).

Posteriormente à reunião de revisão é feita uma reunião de retrospectiva (*Sprint Retrospective*), onde será avaliado que foi aprendido e os ajustes necessários ao projeto, objetivando a melhoria contínua.

Abaixo, na Figura 1, segue-se um esquemático de Papéis, Cerimônias e Artefatos mais difundidos no *Scrum*:

Figura 1 – Fundamentos básicos do *Scrum*
Fonte: https://conexiam.com/pt/desenvolvimento-agil-e-arquitetura-empresarial/. Acessado em: 22/05/2023



Rising e Janoff (RISING; JANOFF, 2000), que aplicaram a metodologia ágil *Scrum* juntamente a suas cerimônias padrão em pequenas equipes ágeis ao longo do tempo, relatam uma série de resultados positivos após algumas iterações dentro das equipes. Sendo esses alguns dos pontos destacados:

- O produto se torna uma série de partes gerenciáveis.
- O progresso é feito, mesmo quando os requisitos não são estáveis.
- Tudo é visível para todos.
- A comunicação da equipe melhora e compartilham sucessos ao longo do caminho.
- Os clientes vêem a entrega pontual de incrementos e obtêm feedback frequente sobre como o produto realmente funciona.
- Um relacionamento com o cliente se desenvolve, a confiança e o conhecimento cresce, e é criada uma cultura onde todos esperam que o projeto seja bem-sucedido.

2.2.2 Kanban

Segundo Anderson (ANDERSON; CARMICHAEL, 2015), o *Kanban* é um método de organização e gerenciamento de serviços profissionais que possui como uma de suas diretrizes de funcionamento limitar o trabalho em andamento para melhorar os resultados. Um sistema *Kanban* é um meio de equilibrar a demanda de trabalho a ser realizado com a capacidade disponível para iniciar um novo trabalho.

A palavra "*Kanban*"possui origem japonesa e significa "Cartão Visual". Uma pesquisa pela palavra no Google retorna mais de 5 milhões de resultados; isto porque a palavra também é utilizada para descrever o sistema que vem sendo utilizado há décadas pela Toyota para visualmente controlar e equilibrar a linha de produção. Portanto, embora sistemas *Kanban* sejam

conceito relativamente novo em Tecnologia da Informação, vêm sendo utilizados por mais de 50 anos no sistema de produção Lean na Toyota (BOEG, 2010).

Segundo Anderson (ANDERSON; CARMICHAEL, 2015), o *Kanban* é sustentado em seis prioridades, sendo elas:

- Visualizar o fluxo de trabalho.
- Limitar o trabalho em progresso .
- Medir e gerenciar o fluxo de trabalho.
- Tornar as políticas explícitas.
- Desenvolver loops de feedback.
- Melhorar de forma colaborativa e contínua.

Boeg (BOEG, 2010) aponta que existem diversas abordagens para o *Kanban* com diferentes ênfases, entretanto a maioria dos especialistas concorda que o *Kanban* é um método de gestão de mudanças que dá ênfase aos seguintes princípios:

- Visualizar de maneira clara o trabalho em andamento;
- Visualizar cada passo em sua cadeia de valor, do conceito geral até software que se possa lançar;
- Limitar o Trabalho em Progresso (WIP – *Work in Progress*), restringindo o total de trabalho permitido para cada estágio;
- Tornar explícitas as políticas sendo seguidas;
- Medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessas decisões;
- Identificar oportunidades de melhorias, criando uma cultura Kaizen, na qual a melhoria contínua é responsabilidade de todos.

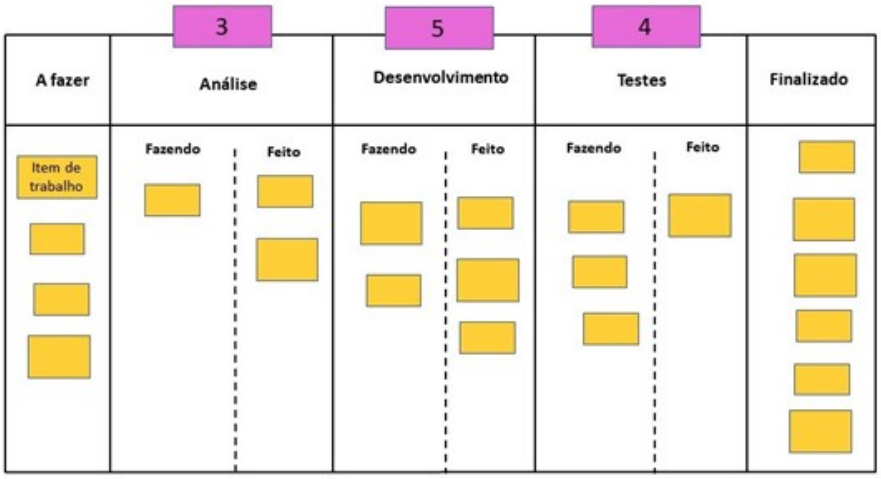
O quadro *Kanban* é composto de colunas com os estados onde o cartão *Kanban* está situado. Segundo Peinado (PEINADO, 2007), o cartão *kanban* é responsável pela comunicação e pelo funcionamento de todo o sistema, nele devem estar contidos as informações mínimas para o bom funcionamento da linha de produção. Sendo necessário, ele poderá conter um número maior de informações, desde que sejam importantes para a área específica, onde se pretende implementar o sistema *Kanban*.

Segundo Boeg (BOEG, 2010), visualizar seu fluxo de trabalho gera uma série de benefícios, sendo os mais importantes:

- Foco no “todo”: Fica visível exatamente como o seu trabalho afeta as outras pessoas e vice-versa;
- Transparência: Todos sabem exatamente o que está acontecendo e nenhuma informação é escondida;
- Identificação de desperdícios: Surgirá naturalmente um questionamento da razão pela qual as coisas são feitas;

A Figura 2, abaixo, ilustra um exemplo de quadro *Kanban* comumente utilizado em contextos de desenvolvimento de software, em que é possível visualizar explicitamente itens de trabalho e seus respectivos estados:

Figura 2 – Exemplo de quadro *Kanban*.
Fonte: https://www.projectbuilder.com.br/blog/. Acessado em: 23/05/2023



2.3 MÉTRICAS DE SOFTWARE

Medição ou mensuração é o processo pelo qual números ou símbolos são associados aos atributos das entidades do mundo real, com o objetivo de descrevê-la de acordo com um conjunto de regras claramente definidas (FENTON; PFLEEGER, 1997).

Mensuração de software é a quantificação de uma característica do produto ou dos processos envolvidos em sua concepção e construção (VAZQUEZ; SIMOES; ALBERT,). Sendo assim, métricas de software referem-se a qualquer tipo de medida de software, processo ou documentação relacionada.

Segundo Pressman (PRESSMAN, 2010) a mensuração é aplicada no processo de desenvolvimento de software ou atributos de um produto com o objetivo de melhorá-lo de forma contínua.

O propósito para mensuração de software é coletar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos desenvolvidos na organização e em seus projetos, de forma a apoiar os objetivos organizacionais (MPS-BR, 2007).

2.3.1 Métricas, medidas e indicadores

Ao discutir o papel das métricas no acompanhamento de projetos ágeis, faz-se necessário conhecer as diferenças entre os conceitos de medidas, métricas e indicadores.

Uma medida é a avaliação de um atributo segundo um método de medição especifico, funcionalmente independente de todas as outras medidas e capturando informação sobre um único atributo (MCGARRY, 2002).

Conforme a IEEE (IEEE; STD; 610.12, 1990), uma métrica é um método para determinar se um sistema, componente ou processo possui um certo atributo. Um indicador é um dispositivo ou variável que pode ser configurado para um determinado estado com base no resultado de um processo ou ocorrência de uma determinada condição.

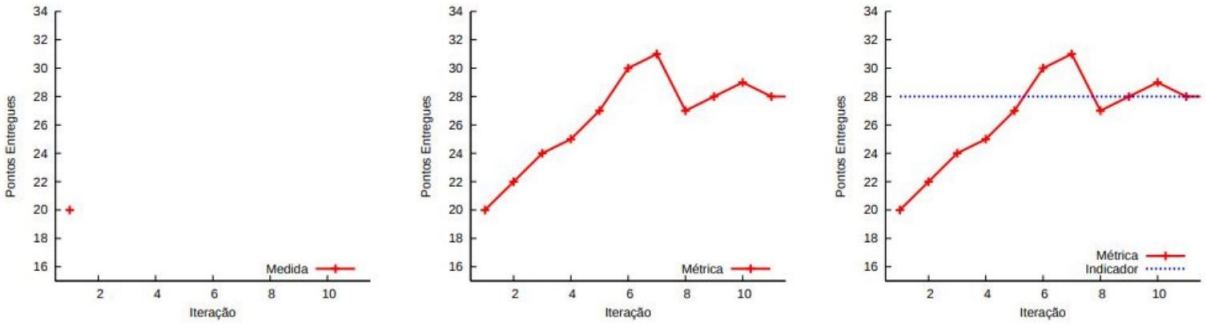
Com as métricas os gerentes conseguem informações quantitativas que auxiliam na tomada de decisões e para obter uma visão melhor do trabalho que está sendo realizado, podendo assim desenvolver um software mais confiável (INTHURN, 2001).

Semelhante as medições do mundo físico, as métricas de software podem ser dividas em diversas categorias: medidas diretas e indiretas (PRESSMAN, 2021) e métricas de controle ou preditivas (SOMMERVILLE, 2004).

Medidas diretas são aquelas que incluem custo, esforço, quantidade de linhas de código produzidas e total de defeitos registrados. São fáceis de serem reunidas e avaliadas. Já as medidas indiretas são aquelas obtidas a partir de outras métricas. São mais difíceis de serem avaliadas, uma vez que analisa a funcionalidade, qualidade, complexidade, eficiência, confiabilidade, manutenibilidade do software, entre outras (ROSA, 2011).

A figura abaixo ilustra exemplo para uma Medida (gráfico a), para uma Métrica (gráfico b) e para um Indicador (gráfico c):

Figura 3 – Exemplos de métrica, medida e indicador.
Fonte: (SATO, 2007)



(a) Medida (b) Métrica (c) Indicador

Métricas de controle geralmente são associadas a processos de software. São dados quantitativos, como por exemplo, esforço e tempo dedicado ao processo. Métricas preditivas são associadas aos produtos de software, como por exemplo, tamanho do código, complexidade do software, número de atributos e classes (SOMMERVILLE, 2004).

O domínio das métricas de software pode ser dividido em: métricas da produtividade, métricas da qualidade e métricas técnicas (ROSA, 2011). Métricas da produtividade concentram-se na saída do processo de engenharia de software, métricas da qualidade permite indicar o nível de resposta do software às exigências implícitas, e por fim as métricas técnicas concentram-se nas características do software (PRESSMAN, 2021).

2.3.2 Métricas ágeis

Com relação às métricas aplicadas com objetivo de medir desempenho em equipes ágeis, Sato (SATO, 2007) e Cohn (COHN, 2005) citam uma série de critérios que uma boa métrica ágil deve ter, tais como: reforçar princípios ágeis, envolvendo toda a equipe, seguir tendências e não números; pertencer a um conjunto pequeno de métricas e diagnósticos; ser facilmente coletada, deixar claro os fatores que a influenciam para evitar manipulações, facilitar a melhoria do processo e fornecer feedbacks.

Conforme Albino (ALBINO, 2017), outra característica importante sobre métricas de negócio é que elas devem ser compreensíveis, isto é, elas não podem ser complicadas e todos devem compreender o que elas representam. A métrica de negócio deve também ser uma relação ou uma taxa. Números absolutos não devem ser usados. A quantidade de usuários pode ser pouco útil, porém, a porcentagem de usuários ativos diários já poderá trazer tendências de crescimento. Razões e taxas são comparativas, o que ajuda a tomar melhores decisões.

Algumas métricas ágeis também destacadas por Albino (ALBINO, 2017), especialmente quando se trata de métricas utilizadas dentro do desenvolvimento do método *Kanban*, são:

- *Work in Progress*.
- *Cumulative Flow Diagram*.
- *Lead time*.
- *Throughput*.

Nestre trabalho aprofundaremos de forma prática em algumas dessas métricas destacadas, especificamente nas sessões de análise, desenvolvimento e avaliação da plataforma proposta.

2.4 OBJECTIVES AND KEY RESULTS (OKR)

Um dos objetivos finais deste trabalho é a captura dos indicadores obtidos na avaliação do processo. Esses indicadores servirão como dados de entrada para o acompanhamento das OKRs, pois representam uma forma de medir os resultados dos processos e são amplamente utilizados na organização-alvo da aplicação a ser desenvolvida neste trabalho.

A *Objectives and Key Results* - OKR, é uma estrutura de pensamento crítico e disciplina contínua que busca garantir que os funcionários trabalhem juntos, concentrando seus esforços para fazer contribuições mensuráveis que impulsionam a empresa para frente (NIVEN, 2016).

OKR é uma estrutura de definição de metas criada pela Intel e adotada por várias empresas do Vale do Silício. O Google é o caso mais famoso, tendo adotado o OKR em seu primeiro ano. Twitter, LinkedIn, Dropbox e Oracle estão entre outros adotantes (CASTRO, 2015).

Também segundo Castro (CASTRO, 2015), essas são algumas das principais características que tornam a OKR única:

- Simplicidade: para permitir ciclos frequentes de estabelecimento de metas , o processo é extremamente simples. Os próprios OKRs devem ser simples e fáceis de entender.
- Cadência mais curta: Em vez de usar um processo anual de planejamento estático, o OKR usa ciclos mais curtos de definição de metas , permitindo um planejamento dinâmico e uma adaptação mais rápida às mudanças.
- *Open Source*: OKR é um framework de código aberto para que as empresas o adaptem a cada cultura e contexto, criando diferentes versões dele.
- Objetivos a longo prazo: objetivos que tiram o time da zona de conforto e fazem com que repensem o modo como trabalham para atingir o máximo desempenho.

Conforme aponta Luna (LUNA et al., 2017), um primeiro ciclo de OKR incia-se a partir da análise em como uma organização “quer ser reconhecida” (visão institucional), permitindo então a identificação do “seu propósito” (missão institucional) e de seus objetivos estratégicos. A partir destas informações essenciais o time procura identificar ao menos três objetivos que uma vez alcançados pelos times ou departamentos auxiliarão a organização a cumprir sua missão.

Para cada objetivo, então, são identificadas de uma a três métricas (*key results*) que permitirão medir o progresso daquele objetivo. E só então, são elaboradas as ações para alcançar os resultados descritos pelas métricas estipuladas. A partir daí, os OKRs são distribuídos no tempo e monitorados periodicamente através de reuniões que avaliam os ciclos de execução, também conhecidas como *Scorings*.

A figura 4 abaixo ilustra uma possível configuração de ciclos de OKR:

Figura 4 – Exemplo de Ciclos OKR.
Fonte: https://linkedin.com/pulse/okrs-objectives-key-results-edalmo-araujo. Acessado em: 24/05/2023

Os ciclos de monitoramento ocorrem dentro das iterações de avaliação, visando avaliar como os objetivos estão sendo alcançados e realizando ajustes sempre que necessário. Já os ciclos de reflexão estratégica ocorrem sempre que a estratégia organizacional precisa ser adequada à uma nova realidade (LUNA et al., 2017).

2.5 PLATAFORMA JIRA

JIRA é uma plataforma de gerenciamento de projetos desenvolvida pela Atlassian Corporation a partir de 2002. É mais comumente utilizada para gerenciamento de projetos de software, mas devido a seus recursos avançados de personalização é altamente utilizada para diversos contextos, como por exemplo: gestão de ordens de serviço, gestão de chamados de clientes e gestão de propostas comerciais.

A plataforma fornece um conjunto de ferramentas poderoso e maduro que viabilizam customizações para atender às necessidades específicas de cada projeto ou organização. Esse conjunto é composto por campos personalizados, tipos de demandas, fluxos de trabalho, notificações e diferentes interfaces de usuário.

2.5.1 Jira API

Jira API (*Application Programming Interface*) diz respeito a uma Interface de comunicação pelo qual diversas aplicações ou ferramentas podem interagir remotamente, via protocolo REST (*Representational State Transfer*), com a plataforma Jira.

Figura 5 – Logo Jira API.
Fonte: https://www.atlassian.com/software/jira. Acessado em: 25/05/2023

O servidor da plataforma Jira provê esse tipo de conectividade com o objetivo de permitir com que desenvolvedores construam integrações entre a plataforma Jira e a mais variada natureza de aplicações, e também para viabilizar a construção de automatizações não presentes de forma nativa da ferramenta.

O acesso aos recursos ocorre por meio de requisições HTTP (*Hypertext Transfer Protocol*) com a utilização de JSON (*JavaScript Object Notation*) como formato de comunicação padrão no tráfego dentro das requisições. São disponibilizados os métodos de comunicação HTTP padrão, como *GET, PUT, POST e DELETE*, além de outros detalhados de maneira mais profunda na documentação da ferramenta.

Há dois principais conceitos de segurança que devem ser considerados ao acessar quaisquer API's que possuam dados sensíveis, que são:

- Autenticação: determina a identificação de quem acessa.
- Autorização: determina quais ações podem ser perfomadas por quem acessa.

A autenticação padrão no Jira API ocorre através de um PAT (*Personal Access Token*) responsável por autenticar chamadas à API da plataforma Jira, que são válidos por um tempo

determinado. A criação dos tokens de autenticação ocorre através de comunicação com a própria API REST, e os mesmos podem ser renovados de maneira facilitada quando necessário.

A autorização, ou nível de permissão de acesso aos recursos, é controlada por meio das permissões previamente definidas e vinculadas ao usuário específico que fará o acesso, dentro do painel de configurações da plataforma Jira.

Os recursos disponibilizados através da Jira API viabilizam com que a ferramenta proposta nesse trabalho possa consumir os dados dos projetos e equipes ágeis existentes na Plataforma Jira em tempo real, de maneira automatizada e segura.

3 ESTADO DA ARTE

3.1 INTRODUÇÃO

Através da técnica de pesquisa denominada revisão sistemática busca-se analisar e resumir o escopo das pesquisas atuais e estado da arte em determinadas áreas de conhecimento, identificando lacunas nos tópicos em questão e fornecendo embasamento para novas atividades de pesquisa na área abordada. Vale ainda ressaltar que a condução da revisão sistemática foi realizada de forma imparcial, de acordo com uma estratégia de pesquisa pré-definida que visa permitir avaliar a integridade deste projeto de pesquisa. Uma série de atividades distintas são conduzidas em diferentes fases do processo para a realização da revisão sistemática. As fases devem obedecer a uma ordem sequencial de execução, porém, as atividades iniciadas durante o desenvolvimento do protocolo podem ser refinadas em fases posteriores.

3.2 MATERIAIS E MÉTODOS

A partir do projeto de TCC proposto e da demanda por artigos que viessem a demonstrar o panorama atual acerca de gestão de equipes ágeis, métricas ágeis e processos de software foram definidas fontes de pesquisa, palavras chave e critérios específicos.

As fontes de buscas foram selecionadas considerando sua importância para a área de pesquisa, interface amigável, quantidade e qualidade de resultados obtidos. As bases de busca escolhidas são Google Scholar e DBLP. A partir das questões de pesquisas foram definidas as palavras chaves e seus sinônimos para compor as estratégias (*strings*) de buscas. No esquemático da Figura 6 consta histórico de evolução da *string* de busca de modo a ficar cada vez mais assertivo no que diz respeito às informações que desejava-se encontrar:

Figura 6 – Esquemático de Refinamento da *String* de Busca.
Fonte: Elaborado pelo Autor.

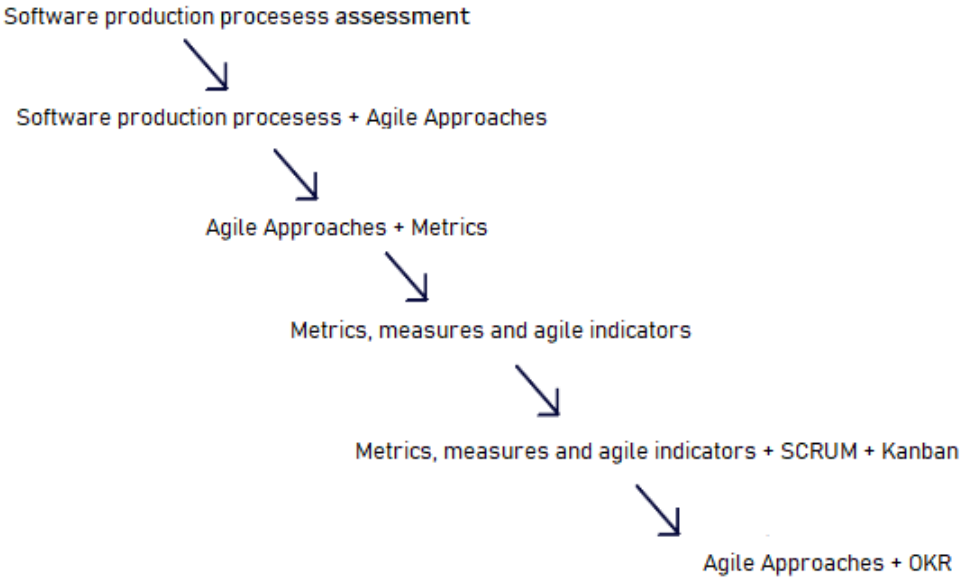
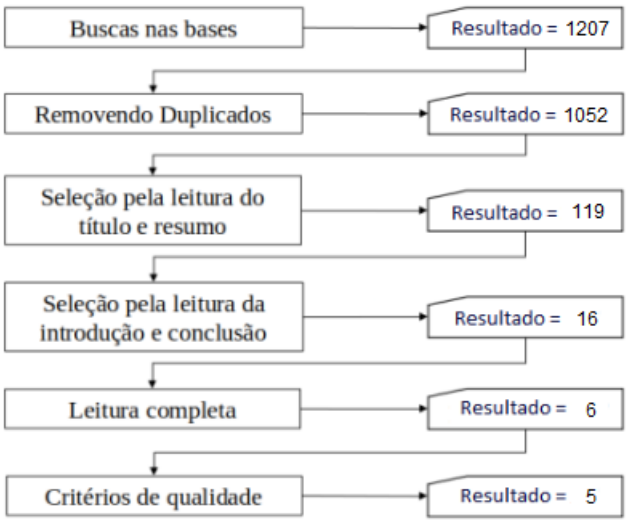


Figura 7 – Seleção de Artigos.
Fonte: Elaborado pelo Autor.



3.3 RESULTADOS

A Figura 7 representa os totalizadores de estudos selecionados para extração de dados de ambas as bases somados (DBLP e Google Scholar). Nota-se que a maioria dos artigos são da base Google Scholar, pois a mesma possui um acervo maior. É possível observar também que houve diminuição significativa da quantidade de artigos na etapa de seleção pela leitura do título e resumo, também foi levado em conta relevância dos artigos em questão de acordo com classificação do próprio acervo de busca de artigos.

Na figura 8 constam os estudos selecionados ao final do processo de mapeamento.

Figura 8 – Relação de Estudos selecionados ao final do mapeamento.
Fonte: Elaborado pelo Autor.

Estudo	Título Original
(Adali, Top e Demirörs. 2017)	Assessment of Agility in Software Organizations with a Web-Based Agility Assessment Tool
(Adali, Top e Demirörs. 2016)	Evaluation of Agility Assessment Tools: A Multiple Case Study
(Homchuenchom, Piyabunditkul e Lichter. 2012)	SPIALS-II: A light-Weight Software Process Improvement SelfAssessment Tool
(Suteeca, Ramingwong. 2021)	The Visualization of ISO/IEC29110 on SCRUM under EPF Composer
(Chagas, Mendes e Cortés. 2013)	Uma abordagem Baseada em Agentes de Apoio à Gestão de Projetos de Software

3.4 ANÁLISE E DISCUSSÃO

Após leitura dos artigos selecionados, foram definidas as informações que possuíam maior relevância para complementar a proposta deste trabalho e fomentar discussão acerca dos temas pertinentes.

A abordagem de Adali, Top e Demirörs (2016) avalia 11 ferramentas de avaliação de agilidade (entre frameworks baseados em questionários e sistemas propriamente ditos) com base em 9 critérios pré-definidos pelos autores em um estudo de caso múltiplo. Os critérios que foram elencados são: cobertura, disponibilidade, capacidade de orientação, registo da avaliação, relatórios automatizados, comparabilidade, diferentes modos de utilização, diferentes escopos de projetos e extensibilidade. O objetivo principal do estudo de Adali, Top e Demirörs (2016) foi de examinar as ferramentas de avaliação de agilidade existentes no momento da realização do estudo para fornecer uma visão geral sobre possíveis usos, vantagens e desvantagens.

Como resultado da avaliação realizada pelos autores, nenhuma das ferramentas de avaliação da agilidade estudadas foi capaz de satisfazer plenamente todos os critérios. Evidenciando, portanto, uma ausência de ferramentas capazes de atender em totalidade os critérios estabelecidos pelos autores e indicando tendências que podem ser seguidas na implementação da ferramenta alvo deste trabalho, com base no atendimento à ausências de outros sistemas e também tendo como referências pontos fortes dos sistemas avaliados através do estudo de Adali, Top e Demirörs (2016) .

Outra abordagem de Adali, Top e Demirörs (2017), possivelmente motivados por todo o panorama que observaram no artigo anteriormente referido, propõe uma ferramenta web integrativa de avaliação da agilidade. A abordagem consiste num modelo estruturado e abrangente de avaliação da agilidade (AgilityMod), e uma ferramenta online de avaliação da agilidade (AssessAgility) que possui a capacidade de orientar e automatizar os processos de avaliação de

ambientes ágeis, incluindo fases de planejamento, condução e registro de ocorrências relevantes de modo a fornecer uma maneira abrangente de avaliar processos. Entretanto, a ferramenta proposta por Adali, Top e Demirörs (2017) pressupõe que organizações alvo se adaptem ao modelo de agilidade proposto pelos autores para que possam se integrar à ferramenta proposta e passem a ter seus processos avaliados de maneira automatizada.

Já a abordagem de Homchuenchom, Piyabunditkul e Lichter (2012) utiliza tarefas do *Scrum* selecionadas e modeladas com BPMN (*Business Process Model and Notation*) em uma plataforma BPMS (*Business Process Management System*). Porém o trabalho de Homchuenchom, Piyabunditkul e Lichter (2012) não aborda uma avaliação automatizada sobre um modelo Scrum já definido, mas tem como objetivo estabelecer um roteiro que pode ser replicado pelas organizações para efetuar uma autoavaliação. De modo a gerar como resultado uma compreensão e visão detalhada do processo *Scrum* empírico sendo praticado. Isto é, busca-se auxiliar equipes ágeis a visualizar melhor o fluxo que praticam de modo a identificar melhorias que venham a impactar suas tarefas de maneira positiva.

A maior parte dos trabalhos que utilizam questionários como entrada de informações, como é o caso de Homchuenchom, Piyabunditkul e Lichter (2012) e grande parte do ferramental visto em Adali, Top e Demirörs (2016), e se baseiam nos modelos de referência SCRUM e/ou CMMI. Então, a partir dos dados coletados nas perguntas, formulam indicativos de aceitação e qualidade de processos de software.

Suteeca e Ramingwong (2016) evidenciam que existem várias incompatibilidades fortes entre normas e padrões estabelecidos e a abordagem ágil ao desenvolvimento de software. Por exemplo, a necessidade identificada nas normas de criar muitos artefatos de qualidade não está em conformidade com muitas das filosofias de agilidade. Uma vez que os métodos ágeis centram no software funcional em detrimento da documentação, a utilização de métodos ágeis alinhados com normas estabelecidas acaba por se tornar difícil de implementar.

Suteeca e Ramingwong (2016) afirmam também que não existem diretrizes acessíveis a organizações de menor porte para que implementem de maneira autônoma a ISO/IEC 29110, uma norma sobre desenvolvimento de software construída com enfoque especialmente para microempresas em ambientes *Scrum*, resultando na constante necessidade de recorrer a consultores externos. A partir dessas motivações Suteeca e Ramingwong (2016) se propuseram a analisar vários cenários de implementação da ISO/IEC 29110 em microempresas, gerando um ferramental denominado EPF (*Eclipse Process Framework*) que tem como principal objetivo a implementação efetiva e conveniente da norma ISSO/IEC 29110 no desenvolvimento de software *Scrum* especialmente em organizações de pequeno porte e com pouca informação referente a padronizações formais. As bibliotecas e aplicações Web que foram o resultado da implementação de Suteeca e Ramingwong (2016) visam auxiliar também na preparação para a certificação da referida norma.

Chagas, Mendes e Cortés (2013) evidenciam que modelos de melhoria de processo de software podem contribuir significativamente na qualidade do produto e evolução das organizações. No entanto, reiteram acerca do grande desafio que é para determinadas organizações a

implementação destes modelos. Então, Chagas, Mendes e Cortés (2013) propoem um sistema web multi-agente para auxiliar na gestão do conhecimento e automação de tarefas recomendadas pelo modelo CMMI (*Capability Maturity Model Inte-gration*) nível 2.

Segundo Russel e Norvig (RUSSELL; NORVIG, 2010), um agente é uma entidade que pode perceber o seu ambiente por meio de sensores e agir sobre ele por meio de atuadores. Agentes de software são programas automatizados que executam ações de forma autônoma e, se necessário, se comunicam com outros agentes a fim de atingir os seus objetivos.

A ferramenta proposta por Chagas, Mendes e Cortés (2013) se diferencia por disponibilizar recursos automatizados para gerenciar projetos de software e também suas avaliações, de forma proativa em tempo de execução, contemplando as práticas do CMMI. Foi observado pelos autores que a conclusão de etapas de projetos pode automaticamente já disparar a avaliações e geração de métricas da etapa finalizada em paralelo a novas etapas de projeto que se iniciam. A ferramenta de Chagas, Mendes e Cortés (2013), entretanto, não pode ser utilizada com outras ferramentas de gerenciamento de projetos a não ser que a ferramenta possua o mesmo padrão de modelagem de banco de dados de projetos utilizado como modelo na implementação de sua ferramenta, conforme citado pelos mesmos em sua prospecção de trabalhos futuros.

Apesar de nem todos os trabalhos apresentarem uma ferramenta para automatizar a avaliação de processos de softwares propriamente dita, todos apresentaram um modelo para seus casos de estudo. De um modo abrangente, todos os modelos apresentam as etapas planejamento e gerenciamento do projeto, análise dos dados e formulação de melhorias ou recomendações. A maior parte dos estudos utilizaram *SCRUM* e CMMI como modelo de referência. Alguns trabalhos apontaram mais de um modelo utilizado.

Vale ressaltar que quatro dos cinco estudos realizaram avaliações de processos utilizando como base modelos ágeis (*Scrum e AgilityMod*) o que levanta indícios de uma preocupação com alinhamento do processo, mesmo para os modelos ágeis.

Outro ponto a ser destacado é que muitas das ferramentas criadas ou citadas nos estudos não foram colocadas em prática em organizações reais para obtenção de resultados, diferenciando-se da ferramenta que propomos no presente trabalho.

3.5 CONCLUSÃO

Ao término desta revisão sistemática, foi possível sintetizar parte do conhecimento científico atual sobre as áreas pesquisadas e direcionar de forma mais precisa o foco da pesquisa e da ferramenta em desenvolvimento. Os conceitos relacionados a processos de produção de software, medição em organizações ágeis e a aplicação automatizada de métricas, medidas e indicadores ágeis identificados na revisão forneceram insumos essenciais para fundamentar e inspirar a proposta deste trabalho.

As análises revelaram limitações significativas nos trabalhos avaliados, como a ausência de utilização prática em organizações reais, a falta de integração com ferramentas amplamente utilizadas no mercado e a dependência de modelos de referência como SCRUM e CMMI.

Tais deficiências dificultam a validação empírica e limitam a aplicabilidade dos modelos e ferramentas em contextos organizacionais diversificados. Além disso, a maioria das soluções apresentadas foca em aspectos isolados do processo ágil, sem oferecer uma visão integrada, o que reforça a necessidade de avanços na área.

No contexto deste trabalho, busca-se abordar algumas das lacunas identificadas, conforme descrito a seguir:

- **Ausência de utilização prática em ambientes organizacionais reais:** A aplicação proposta foi projetada para uso em organizações de mercado, atendendo a cenários reais de operação.
- **Falta de integração com ferramentas amplamente utilizadas no mercado:** A solução desenvolvida se integra à plataforma Jira, uma ferramenta consolidada e amplamente difundida no mercado.
- **Necessidade de avaliação automatizada ou parcialmente automatizada:** A aplicação proposta oferece funcionalidades de avaliação automatizada, com atualização de métricas em tempo real, otimizando o monitoramento de processos.
- **Dependência de modelos específicos, como SCRUM e CMMI:** A ferramenta apresenta flexibilidade para adaptação a diferentes metodologias e contextos organizacionais, sem estar restrita a um modelo específico.

Contudo, o trabalho apresenta algumas limitações também observadas em grande parte dos trabalhos analisados:

- **Restrição ao uso com ferramentas de gerenciamento de projetos compatíveis com o padrão de modelagem do Jira:** A solução foi desenvolvida considerando a estrutura do Jira como base.
- **Cobertura limitada aos processos avaliados:** A aplicação concentra-se nos processos relevantes para a avaliação automatizada de métricas, sem abranger integralmente todos os processos de produção de software.

Essas delimitações foram consideradas na construção da solução, que tem como objetivo principal oferecer uma abordagem prática, adaptável e direcionada às necessidades específicas de avaliação de métricas ágeis em ambientes reais.

3.6 AMEAÇAS À VALIDADE

Foram verificadas as seguintes ameaças à validade da pesquisa realizada:

- As abordagens automatizadas para avaliações de processos, principalmente em empresas de pequeno e médio porte, ainda estão em fase de pesquisa e estudos, os relatos acerca dos resultados encontrados na sua implantação ainda são escassos.
- Durante as fases inicias de pesquisa e refinamento das *strings* de busca na revisão sistemática foi encontrada grande quantidade de artigos. Portanto, algum estudo relevante pode não ter sido incluído até a seleção final desta revisão.
- Nos estudos verificados ocorre uma ausência de implantações práticas em ambientes ágeis empresariais, resultando numa baixa quantidade de resultados de implantação e de eficácia verificada das ferramentas propostas.

4 PROPOSTA

A proposta deste Trabalho de Conclusão de Curso consiste na implementação de uma aplicação web voltada à geração de métricas ágeis e customizadas, com integração direta à plataforma Jira via Jira API. Essa aplicação explora um vasto banco de dados de projetos acumulados ao longo do tempo, pertencente a equipes da organização da qual o autor faz parte, e busca transformar esses dados em informações significativas para a análise e melhoria de processos.

Por meio do acompanhamento em tempo real das atualizações de demandas registradas no Jira, a ferramenta visa monitorar o alcance de objetivos estratégicos da organização, com destaque para o cumprimento de OKRs. Adicionalmente, a solução proposta busca identificar possíveis gargalos, evidenciar oportunidades de melhoria no fluxo de trabalho e oferecer perspectivas ainda não exploradas sobre os times e projetos. Dessa forma, o objetivo é auxiliar tanto na tomada de decisões operacionais quanto estratégicas, promovendo melhorias no cotidiano das equipes ágeis de desenvolvimento e na gestão organizacional, por meio de uma visualização clara e rápida dessas informações.

O desenvolvimento da aplicação busca endereçar lacunas frequentemente observadas na literatura e em soluções existentes. Entre essas lacunas, destacam-se a ausência de utilização prática em ambientes organizacionais reais, uma vez que a aplicação foi concebida para uso direto em uma organização de mercado, permitindo sua observação em cenários reais. Além disso, aborda a falta de integração com ferramentas amplamente utilizadas no mercado, oferecendo uma solução que se integra nativamente ao Jira, uma plataforma consolidada e amplamente reconhecida no setor. A necessidade de avaliação automatizada ou parcialmente automatizada também foi contemplada, com a implementação de funcionalidades que permitem atualização de métricas em tempo real, otimizando o monitoramento e a gestão de processos. Por fim, a dependência de modelos específicos, como SCRUM e CMMI, foi mitigada pela flexibilidade da ferramenta, que possibilita adaptação a diferentes metodologias e contextos organizacionais, sem se limitar a frameworks predefinidos.

Após a implementação, a ferramenta foi disponibilizada para uso prático no contexto de equipes ágeis de desenvolvimento em uma organização de mercado, na qual o autor deste trabalho é colaborador. Esse processo permitiu a análise dos impactos e resultados obtidos, sob a perspectiva do autor. A observação realizada teve como objetivo identificar os benefícios proporcionados pela aplicação, tais como a melhoria na rotina operacional das equipes, a detecção de oportunidades para otimização de processos e o fortalecimento do suporte às decisões estratégicas nos âmbitos de negócios e gestão.

5 ANÁLISE

Este capítulo visa discorrer brevemente acerca do processo, produto e equipes da organização que serviu como base para desenvolvimento da aplicação proposta neste trabalho. Contempla também a análise das métricas e OKRs que fazem sentido no cotidiano das equipes ágeis aqui descritas e que, portanto, agregam valor ao sistema, além de apresentar também a análise de requisitos funcionais e não funcionais do desenvolvimento.

5.1 CONTEXTO

A organização alvo deste trabalho é focada em tecnologia inovadora para o mercado financeiro, porém, com produtos de software em outras diversas áreas tais como previdência complementar privada, consórcios, bancos, inteligência de negócios e meios de pagamento. Atualmente possui filiais físicas em diferentes estados do Brasil e tem se destacado nos últimos anos por um crescimento significativo em fatia de mercado em todas as suas vertentes de software.

Ao longo deste estudo foi observado o cotidiano recente de três equipes de desenvolvimento da vertical de previdência privada da organização, de modo que foi refletido na aplicação-alvo desse estudo os dados de produção de software dessas equipes com a devida anonimização/descaracterização dos dados sensíveis de modo a não identificar clientes, projetos e/ou colaboradores, porém sem perder o valor dos dados ponto de vista de construção de métricas e possibilidade de abstração de indicadores significativos.

É importante destacar que o autor deste trabalho integra uma das equipes mencionadas anteriormente e que a aplicação desenvolvida neste trabalho busca, de forma efetiva, contribuir para o aprimoramento do cotidiano dessas equipes através da adoção da solução construída.

5.2 PROCESSO DA ORGANIZAÇÃO

Para a organização em foco neste estudo, todas as demandas de alterações de software exigem a criação de uma "tarefa" na plataforma JIRA, abrangendo desde correções de bugs até melhorias e novos desenvolvimentos. Cada tarefa está vinculada a um projeto, que, por sua vez, é composto por outras tarefas. Essas tarefas são alocadas em sprints, que pertencem exclusivamente a um time e possuem datas definidas de início e término.

A documentação de histórico produzida na plataforma JIRA através das tarefas alocadas em sprints é utilizada como uma base para toda a documentação de liberação ao usuário final. Isso inclui revisões de código, testes cruzados ou automatizados e também especificidades encontradas ao longo da execução das tarefas.

Cada tarefa criada na plataforma JIRA possui, por padrão, uma estrutura de campos *default* semelhante ao ilustrado na Figura 9, abaixo:

Figura 9 – Campos da plataforma Jira utilizados pelo *Reporter* de uma tarefa.
Fonte: https://www.atlassian.com/software/jira. Acessado em 26/05/2023.

Field Name	Description
Summary	A one-line description of the request
Priority	Urgent, Important, Normal, or Low. Urgent issues may be handled as patch releases
Component	The product within the project, chosen from a project-specific list
Category	Software, Operational Data, Infrastructure, Documentation.
Description	A freeform text field describing the request
Affects Versions	What software version this request relates to
Environment	Where the issue manifests (main facility, side lab, etc.)
Origin	Design Review, Coding, Developer Unit Test, Operations, Offline Tests, etc.
Reporter	Who requested the change (auto filled)
Recommend-ation	If a particular fix is needed, it can be specified here
Wrap-around	A flag indicating that this was a data change that originated in the production environment.

Quando uma tarefa na plataforma Jira é atualizada, os “observadores”, que podem ser desde líderes de equipe e gestores associados ao projeto ou demanda especificada como também outros desenvolvedores, recebem automaticamente um e-mail de notificação da atualização.

A Figura 10, abaixo, descreve um fluxo padrão de trabalho (*workflow*) para uma tarefa de software genérica na organização, onde as caixas e as setas são estados e transições das tarefas, respectivamente:

Neste fluxo padrão, destacam-se principalmente os estados de *Assigned, In Progress, Ready for Code Review, Ready for Test* e *Test Passed*.

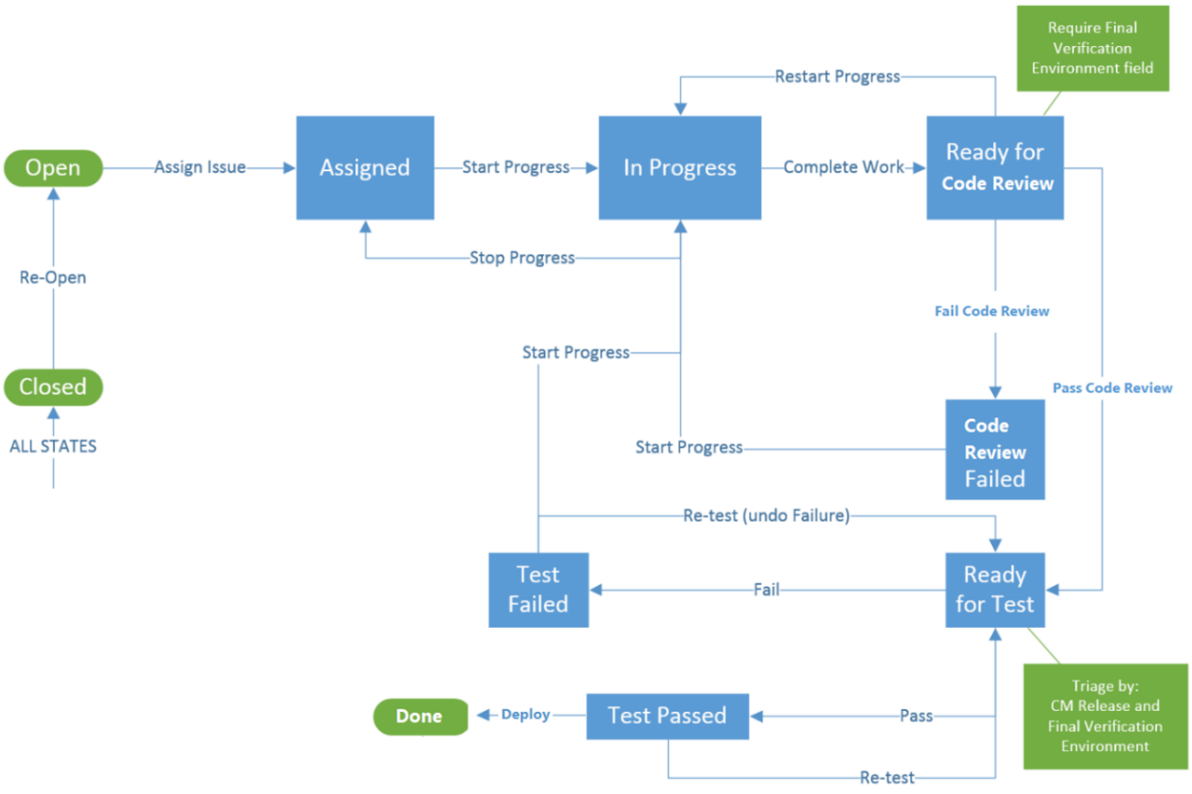
No estado *Assigned* uma demanda já possui um atuante definido, possui critérios de aceite e requisitos de desenvolvimento bem definidos, e já pode ter sua atuação iniciada.

Quando no estado *In Progress*, a demanda já encontra se em andamento, com atuação de uma ou mais pessoas da equipe, ao passo que o produto final desse estado é definido como requisitos da tarefa atendidos e documentação de desenvolvimento concluída.

No estado *Ready for Code Review* a demanda encontra-se finalizada do ponto de vista de desenvolvimento, e é de responsabilidade de outro membro da equipe (diferente dos que atuaram originalmente no desenvolvimento) a revisão de código construído e aprovação da publicação do código em ambiente de validação.

Quando em *Ready for Test*, o desenvolvimento já encontra-se publicado em abiente de testes, e os responsáveis pela realização dos mesmos tem acesso ao escopo e documentação da tarefa, visando aprofundar as validações.

Figura 10 – *Workflow* Jira com devidas adaptações.
Fonte:(FISHER; D.J.KONING; A.P.LUDWIGSEN, 2013)



Ao final, quando em *Test Passed*, é dado seguimento na publicação da nova *feature* em ambiente produtivo e a tarefa é considerada como finalizada.

Para fins de facilidade de entendimento e desenvolvimento, na aplicação-alvo deste trabalho foram abstraídos os dados referentes à tarefas, de modo que uma tarefa possua somente os atributos relevantes para o contexto de geração das métricas.

5.3 PRODUTO

A vertical de Previdência, unidade de pertencimento dos times neste trabalho relatados, produz software para gestão de previdência complementar para as Entidades Gestoras de Previdência Privada.

Dentro deste escopo de software constantemente estão envolvidas demandas de meios de pagamentos que envolvem interações bancárias, pois participantes de um plano de previdência necessitam diferentes meios para realizar suas contribuições mensais aos seus respectivos planos e também, quando do momento de sua aposentadoria ou resgate de recursos no plano, demandam meios para recebimento automatizado dos valores.

Outra área de domínio constante em projetos desse segmento é customização multiplataforma dos produtos. Pois, participantes dos planos de previdência necessitam acompanhar seus saldos, projeções, informes anuais e demais informações através de computadores com diferentes navegadores e dispositivos móveis das mais diferentes naturezas.

Além dos tópicos anterioremente citados, a preocupação com a segurança cibernética dos produtos e demandas de infraestrutura automatizada de produção e publicação de software também geram demandas constantes no cotidiano dos times, visando o enriquecimento dos produtos e amadurecimento do respectivo ecossistema de tecnologia envolvido.

5.4 EQUIPES

As três equipes de desenvolvimento de software que forneceram os dados utilizados na ferramenta proposta neste trabalho são compostas por cinco membros cada, totalizando 15 colaboradores.

Entre os colaboradores, constam desenvolvedores e validadores com diferentes níveis de experiência. Desenvolvedores de maior senioridade assumem, além das demandas de desenvolvimento de software, algumas demandas relacionadas à infraestrutura e *DevOps* de forma esporádica.

As tarefas possuem um atributo denominado "tag", que identifica a natureza da tarefa. A seguir, são apresentados alguns exemplos das possíveis tags atribuídas a uma tarefa:

- Front-end
- Back-end
- Mobile
- Infra
- DevOps
- Bug
- Teste

Tarefas denotadas pela tag "Bug"correspondem a demandas atreladas ao ambiente produtivo e que demandam, portanto, resoluções mais tempestivas.

Cada equipe possui sua própria *sprint* de desenvolvimento, que costuma ter duração de até 2 semanas independente do time ao qual pertence, e com o apoio das equipes de gestão e líderes são realizadas periodicamente reuniões de planejamento de novas sprints e revisão de sprints finalizadas.

5.5 ANÁLISE DE OKR

A organização em estudo utiliza a metodologia OKR (Objectives and Key Results) para definir e monitorar objetivos que impactam diretamente os principais resultados estratégicos do negócio. Dessa forma, ao atingir os objetivos estabelecidos, os colaboradores são

recompensados ao final de cada trimestre, seja por meio de promoções, ajustes salariais, premiações individuais ou coletivas e aumento na participação nos lucros.

Nesse contexto, foram mapeadas três OKRs específicas e diretamente influenciadas pelas equipes de desenvolvimento mencionadas neste estudo, visando proporcionar um acompanhamento mais tangível e acessível na rotina dos times. O intuito é que essas metas sejam monitoradas no dia a dia, e não apenas revisadas ao final de cada trimestre.

As OKRs selecionadas para acompanhamento por meio da aplicação desenvolvida estão descritas a seguir:

- *Total Sprint Deliveries*
- *Bug Resolution Time*
- *Bug Tracker for Completed Projects*

A OKR ***Total Sprint Deliveries*** refere-se à meta de conclusão média de 80% das tarefas planejadas a cada sprint dentro de um período de tempo estabelecido para avaliação. O objetivo desta OKR é incentivar as equipes a manterem uma média elevada de entregas concluídas, promovendo a consistência nos resultados.

A OKR ***Bug Resolution Time***, por sua vez, mede o tempo de resolução de tarefas identificadas com a tag "Bug"imediatamente após o registro de problemas no ambiente de produção. O objetivo estabelecido é corrigir 75% dos bugs reportados em até três dias após sua identificação.

Já a OKR ***Bug Tracker for Completed Projects*** monitora o percentual de projetos finalizados (ou seja, projetos que concluíram todas as tarefas associadas) que geram registros de bugs em produção. A meta estabelecida para esta OKR é garantir que, no máximo, 20% dos projetos finalizados originem ao menos uma tarefa identificadas como "Bug"após sua publicação em ambiente produtivo.

5.6 ANÁLISE DE MÉTRICAS

Na fase de análise, foram realizados levantamentos para selecionar as métricas ágeis mais adequadas ao contexto cotidiano das equipes de desenvolvimento, levando em consideração aquelas amplamente difundidas na literatura da área e que já agregam valor ao processo de monitoramento e análise de produção atualmente. Este conjunto foi classificado como "Métricas Ágeis"na aplicação em desenvolvimento, visando facilitar o acesso a esses indicadores essenciais para o acompanhamento das atividades das equipes.

Adicionalmente, foram propostas métricas customizadas coletivas e individuais, definidas a partir de aspectos específicos do negócio e da produção de software que não são usualmente contemplados pelas métricas ágeis convencionais. Na aplicação aqui descrita, essas métricas foram categorizadas como "Métricas Customizadas"e foram elaboradas para oferecer

recortes e análises direcionados aos dados organizacionais. O objetivo é fornecer, por meio de uma análise interpretativa, indicadores que sustentem a tomada de decisões gerenciais e estratégicas significativas, contribuindo para a melhoria contínua e o crescimento da organização.

5.6.1 Métricas ágeis

Dentre as métricas ágeis que frequentemente agregam valor ao processo atual de análise e monitoramento das equipes da organização, foram selecionadas as seguintes métricas para implementação na aplicação desenvolvida neste trabalho:

- Burndown Chart
- Burnup Chart
- Work in Progress (WIP)
- Velocity

Os critérios para seleção das métricas citadas acima são:

- Opinião dos gestores das equipes.
- Frequência de uso por gestores e membros dos times.
- Viabilidade de implementação, considerando o esforço técnico necessário em relação à frequência de utilização nas equipes.

O ***Burndown Chart*** é uma representação gráfica utilizada para monitorar o progresso de um sprint ou projeto ágil ao longo do tempo. Esse gráfico exibe a quantidade de trabalho restante em relação ao tempo decorrido, permitindo que a equipe visualize rapidamente se está no ritmo adequado para concluir as tarefas planejadas. No eixo horizontal, estão representados os dias ou unidades de tempo do sprint, enquanto o eixo vertical indica a quantidade de trabalho restante, geralmente medida em pontos de história ou horas de trabalho. A linha de progresso ideal desce linearmente, e a comparação com o progresso real fornece insights valiosos sobre o andamento do sprint e possíveis ajustes necessários para alcançar as metas.

Exemplo ilustrativo conforme Figura 11 a seguir:

O ***Burnup Chart*** é uma ferramenta visual usada para acompanhar o progresso de uma equipe ágil em direção à conclusão do trabalho previsto. Diferente do Burndown Chart, o Burnup Chart apresenta duas linhas no gráfico: uma que indica o total de trabalho a ser feito e outra que representa o trabalho já realizado. Com o tempo avançando no eixo horizontal e o volume de trabalho no eixo vertical, esse gráfico permite que a equipe observe tanto o avanço acumulado quanto qualquer alteração no escopo do projeto. O Burnup Chart é particularmente útil para identificar tendências de produtividade e para oferecer uma visão clara do impacto de ajustes no escopo ao longo do desenvolvimento.

Exemplo ilustrativo conforme Figura 12 a seguir:

Figura 11 – Exemplo ilustrativo - Burndown Chart.
Fonte: https://www.clariostechnology.com/productivity/blog/burnupvsburndownchart/.
Acessado em dezembro de 2024

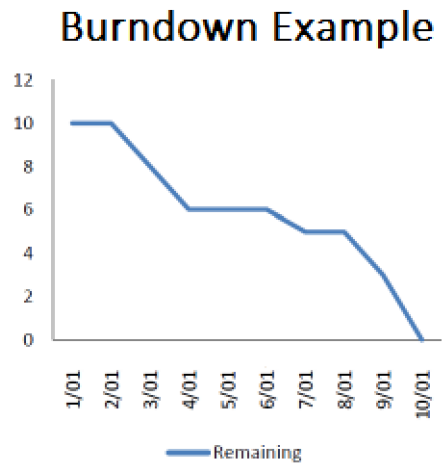
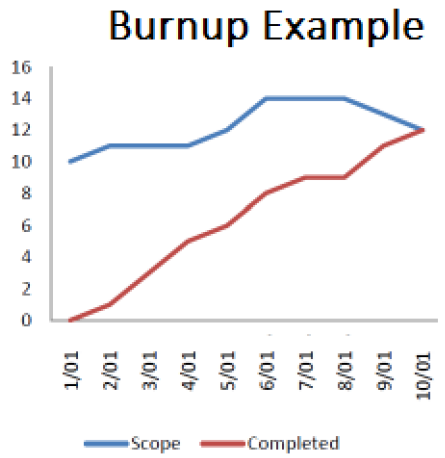


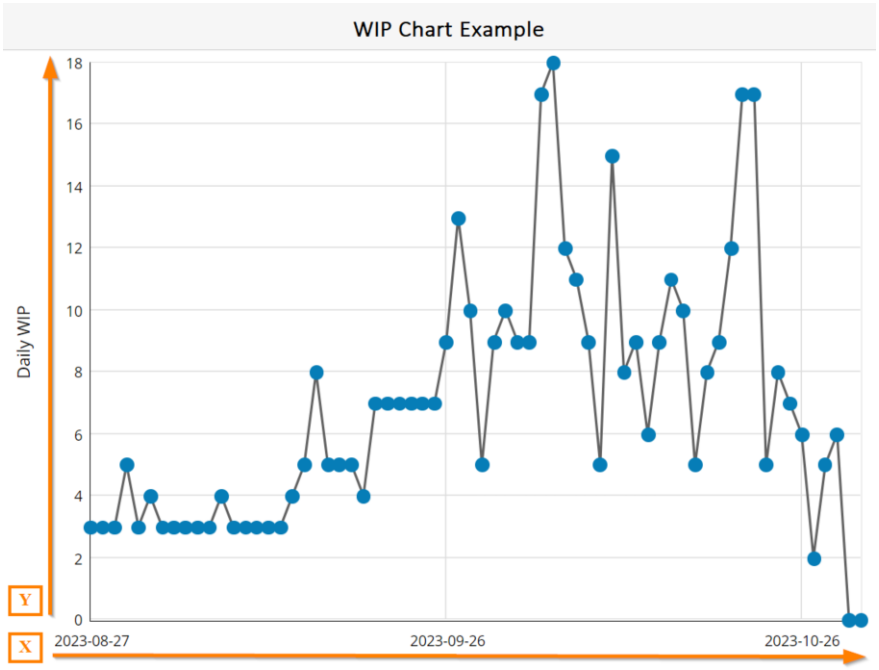
Figura 12 – Exemplo ilustrativo - Burnup Chart.
Fonte: https://www.clariostechnology.com/productivity/blog/burnupvsburndownchart/.
Acessado em dezembro de 2024



Work in Progress (WIP), ou Trabalho em Progresso, é uma métrica ágil que mede a quantidade de trabalho atualmente em andamento, ou seja, tarefas que foram iniciadas, mas ainda não concluídas. Essa métrica é essencial para o gerenciamento de fluxo, pois um WIP excessivo pode indicar que a equipe está sobrecarregada, comprometendo a eficiência e a qualidade do trabalho. Controlar o WIP permite que a equipe foque em concluir as tarefas em andamento antes de iniciar novas, promovendo um fluxo de trabalho mais equilibrado e produtivo. Na prática, o WIP é frequentemente limitado para manter o ritmo e assegurar que o time mantenha a consistência na entrega de valor.

Exemplo ilustrativo conforme Figura 13 a seguir:

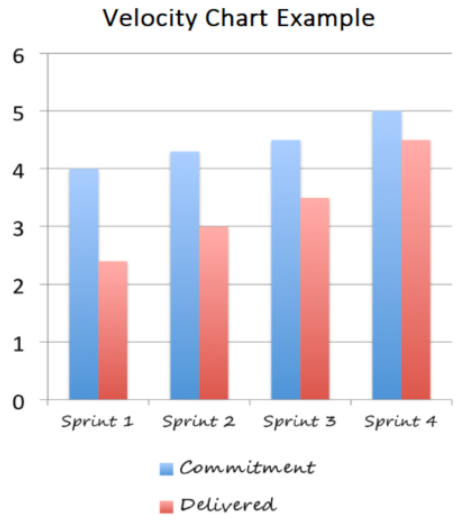
Figura 13 – Exemplo ilustrativo - WIP.
Fonte: https://knowledgebase.businessmap.io/hc/en-us/articles/360014707379-The-WIP-Run-Chart. Acessado em dezembro de 2024



O ***Velocity*** é uma métrica que representa a quantidade de trabalho que uma equipe ágil conclui em cada sprint, medido geralmente em pontos de história. Esse gráfico exibe a velocidade média da equipe ao longo de vários sprints, permitindo previsões mais precisas sobre a capacidade de entrega e o ritmo de desenvolvimento. Com base no histórico da velocidade, as equipes podem estimar a quantidade de trabalho que é realista planejar para sprints futuros, facilitando a gestão de expectativas e o planejamento de entregas. O Velocity Chart é uma ferramenta útil para identificar variações de produtividade e ajustar a carga de trabalho conforme necessário para manter a equipe em um ritmo sustentável.

Exemplo ilustrativo conforme Figura 14 a seguir:

Figura 14 – Exemplo ilustrativo - Velocity Chart.
Fonte: https://www.agile-scrum.be/whats-great-scrum-methodology/velocity-chart/.
Acessado em dezembro de 2024



5.6.2 Métricas Customizadas

Dentre as métricas customizadas a serem implementadas, destaca-se o potencial de agregar valor à organização em duas em específico que estão listadas e melhor descritas abaixo:

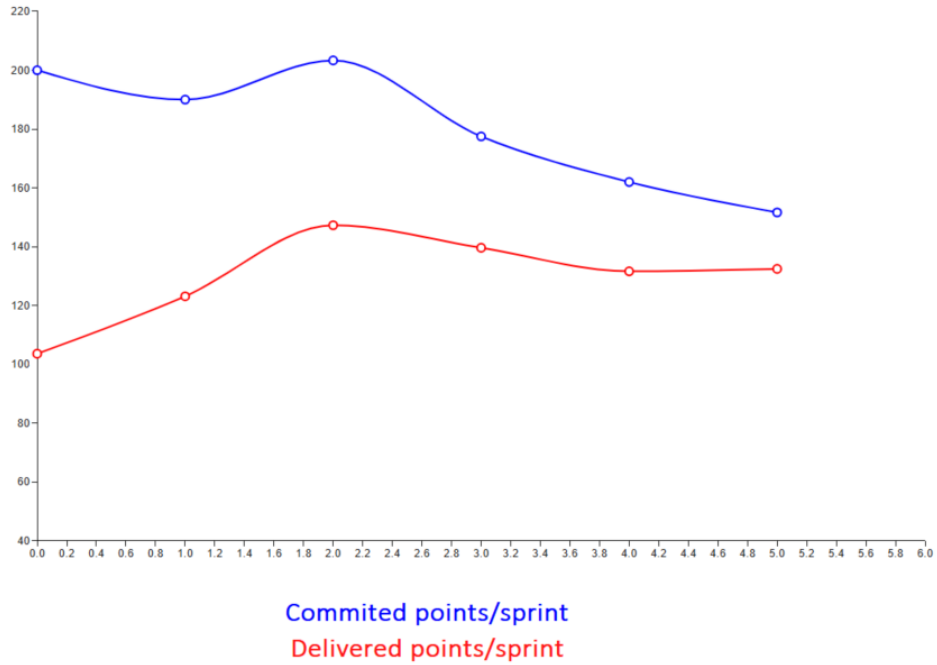
- *Moving Average Velocity*
- *Bugs Per Client*

A métrica de **média móvel de velocidade**, também denominada como *moving-average-velocity* na aplicação, fornece uma análise contínua da capacidade de entrega de uma equipe ágil ao longo de diversos sprints, utilizando o histórico de sprints anteriores para calcular a média de trabalho concluído e o previsto para os próximos períodos. Essa métrica é especialmente útil para monitorar a produtividade ao suavizar as variações entre sprints individuais, ajudando a identificar o ritmo de desenvolvimento da equipe. Por meio dessa média móvel, a equipe e os gestores conseguem observar o desempenho médio ao longo do tempo, o que permite fazer previsões mais consistentes sobre a velocidade com que o backlog será executado e, portanto, possibilita um planejamento mais eficaz das entregas futuras.

Além disso, essa média atua como um indicador das oscilações de produtividade dentro do time. Em períodos em que a entrega da equipe é estável, o gráfico exibirá uma curva menos acentuada, enquanto flutuações significativas na produtividade tornam a curva mais pronunciada, destacando os sprints de baixa ou alta produtividade. Essa visibilidade sobre a consistência e as variações na velocidade de entrega facilita a identificação de fatores que podem impactar o desempenho da equipe, possibilitando ajustes no planejamento e intervenções para mitigar riscos, de forma a sustentar um fluxo de trabalho mais previsível e eficiente.

Exemplo ilustrativo conforme Figura 15 a seguir:

Figura 15 – Exemplo ilustrativo - Moving Average Velocity.
Fonte: Elaborado pelo autor.

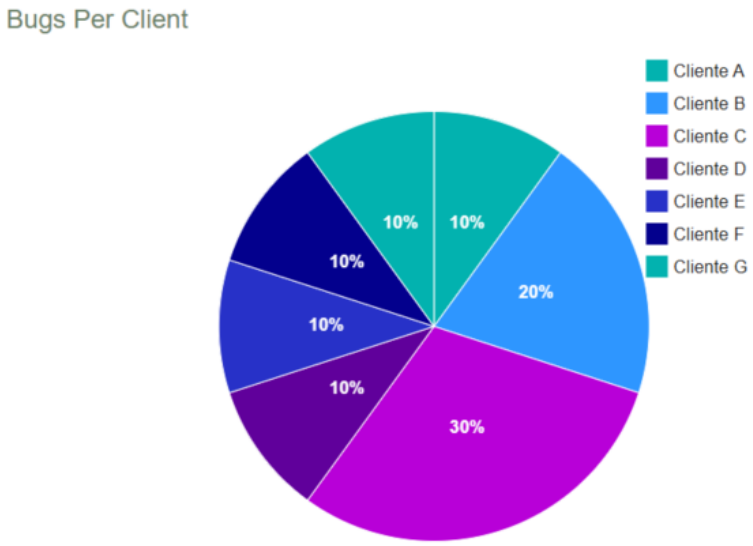


A métrica de **Bugs por cliente**, também descrita como *Bugs Per Client* permite a visualização da quantidade de tarefas marcadas como 'bug' para cada cliente em um determinado intervalo de datas, configurável pelo usuário na interface da aplicação. Esse gráfico possibilita à organização identificar os clientes que mais demandam esforços para correção de falhas em ambientes produtivos, fornecendo uma visão clara e detalhada sobre a qualidade e estabilidade das soluções oferecidas a cada cliente. Essa métrica auxilia no monitoramento dos requisitos e necessidades específicas de cada cliente, permitindo que a organização reconheça os pontos críticos que podem estar gerando um maior número de relatos de bugs.

Além de destacar os clientes que apresentam maior incidência de bugs, a métrica *Bugs Per Client* pode revelar padrões importantes que contribuem para uma análise estratégica da complexidade das demandas. Clientes com volumes elevados de bugs podem indicar processos de negócio mais complexos ou revelar lacunas na equipe de desenvolvimento responsável, que podem ser sanadas com maior suporte técnico ou treinamentos específicos sobre as regras de negócio. Identificar esses padrões ao longo dos trimestres permite o desenvolvimento de planos de ação direcionados, tanto no aprimoramento técnico das equipes quanto na abordagem comercial, promovendo um relacionamento mais proativo e eficaz com os clientes e, ao mesmo tempo, fortalecendo a qualidade do produto entregue.

Exemplo ilustrativo conforme Figura 16 a seguir:

Figura 16 – Exemplo ilustrativo - Bugs per Client.
Fonte: Elaborado pelo autor.



5.6.3 Métricas Individuais

Foram selecionadas quatro perspectivas significativas para as métricas individuais, com foco na análise de dados referentes a colaboradores específicos. Essas métricas foram escolhidas para proporcionar uma visão abrangente e detalhada do desempenho individual, abordando aspectos essenciais para o monitoramento e desenvolvimento das habilidades e contribuições dos membros da equipe, conforme lista abaixo.

- Distribuição percentual de atuação por cliente
- Distribuição percentual de atuaçãor por Projeto
- Distribuição percentual de atuação por Tipo de Demanda
- Pontuação entregue (performance) por tipo de tarefas

A métrica de **Distribuição Percentual de Atuação por Cliente** permite monitorar o percentual de tempo e esforço que um colaborador dedica a cada cliente, proporcionando uma visão detalhada sobre o foco individual nas demandas específicas de diferentes clientes. Através da análise do histórico de sprints, essa métrica permite identificar colaboradores que têm se especializado nas regras de negócio ou no contexto de um cliente em particular. Essa especialização pode ser um indicativo estratégico, pois colaboradores com experiência concentrada em um cliente específico podem oferecer suporte mais qualificado, além de agregar valor na continuidade de projetos futuros para o mesmo cliente.

Na **Distribuição Percentual de Atuação por Projeto**, a métrica indica de forma clara quais projetos têm demandado mais tempo e atenção de cada colaborador. Esse monitoramento

é valioso para identificar profissionais que possuem maior familiaridade com determinado projeto, facilitando a alocação de recursos e a designação de mentores para novas iniciativas similares. Em casos onde um projeto semelhante é demandado por outro cliente ou o mesmo projeto surge para o mesmo cliente, essa métrica permite uma seleção fundamentada de colaboradores para suporte e mentoria ao desenvolvimento, otimizando o tempo de adaptação e aumentando a eficiência da equipe na implementação."

Por meio da **Distribuição Percentual de Atuação por Tipo de Demanda**, a análise se aprofunda na natureza das atividades em que o colaborador atua, considerando o atributo 'tag' das tarefas. A diversificação nas naturezas de demanda atendidas pode refletir o nível de experiência e senioridade do colaborador, uma vez que profissionais mais seniores tendem a atuar em uma variedade maior de tipos de demanda. Por outro lado, um perfil de atuação mais restrito a tipos específicos de tarefas pode indicar um colaborador em fase inicial na organização ou com menor senioridade. Com base nesses dados, a gestão pode vir a desenvolver planos de ação direcionados, tanto para promover o desenvolvimento do colaborador quanto para alinhar suas atividades às expectativas de crescimento e aprimoramento profissional do mesmo.

A métrica individual de **Performance por Tipo de Tarefa** permite a visualização, por meio de gráficos, do aumento ou diminuição na quantidade de entregas realizadas por um determinado colaborador. Essa análise pode ser feita com base em tipos específicos de tarefas e em sprints filtradas conforme a necessidade. Dessa forma, é possível identificar os tipos de tarefas em que o colaborador demonstra maior facilidade de entrega ao longo do tempo, bem como aquelas em que há oportunidades para aprimoramento de suas habilidades.

5.7 LEVANTAMENTO DE REQUISITOS

Esta seção apresenta os requisitos funcionais e não funcionais definidos durante a fase de concepção da aplicação-alvo deste trabalho. Através da contribuição informal de membros das equipes de desenvolvimento ágil, gestores dos times e integrantes da equipe comercial, foi possível identificar funcionalidades que agregam valor significativo ao sistema.

Além disso, são descritas ao longo desta seção as histórias de usuário, que se relacionam direta ou indiretamente com alguns dos requisitos funcionais mapeados.

A definição dos requisitos não funcionais baseou-se em pesquisas, conhecimento técnico prévio, opiniões de membros das equipes de desenvolvimento e nas características da integração planejada, visando facilitar o desenvolvimento da ferramenta.

5.7.1 Requisitos funcionais

De acordo com Sommerville (SOMMERVILLE, 2011), requisitos funcionais são declarações de serviços que o sistema deve oferecer, como o sistema deve reagir a entradas específicas e como ele deve se comportar em determinadas situações. Em alguns casos, esses requisitos também podem indicar explicitamente o que o sistema não deve fazer.

Os requisitos funcionais listados a seguir foram definidos com base na análise do cotidiano de planejamento e desenvolvimento das equipes, além de sessões de *brainstorming* realizadas com membros específicos dos times, gestores e integrantes da equipe comercial.

A tabela abaixo (Figura 17) apresenta os Requisitos Funcionais mapeados:

Figura 17 – Tabela de Requisitos Funcionais.
Fonte: Elaborado pelo Autor.

RF01	O sistema deve possuir uma interface de login que permita acesso à visualização dos dados de produção de software das equipes apenas a usuários com as devidas permissões de acesso.
RF02	O sistema deve restringir a visualização de dados aos times e projetos para os quais o usuário logado possui permissão de acesso.
RF03	O sistema deve exibir métricas ágeis já integradas à plataforma Jira, comumente utilizadas para acompanhamento e análise gerencial, centralizando as principais métricas e indicadores em um único ambiente.
RF04	O sistema deve exibir métricas e indicadores baseados em dados individuais dos membros das equipes de desenvolvimento, permitindo a análise por projetos, clientes e escopo de tarefas (ex.: Front-end, Back-end, Infraestrutura, DevOps e demais escopos).
RF05	O sistema deve apresentar métricas personalizadas, com o objetivo de ilustrar aspectos relevantes dos dados para times de negócios e gestão por meio de amostragens significativas.
RF06	O sistema deve permitir o acompanhamento em tempo real das OKRs registradas, facilitando a visualização constante dessas metas no cotidiano das equipes.
RF07	O sistema deve ser capaz de consumir, em tempo real, dados disponibilizados por meio da interface de integração API da plataforma Jira.

Conforme o descrito por Cohn (COHN, 2004), histórias de usuário devem ser descrições breves e simples de funcionalidades contadas da perspectiva de quem deseja uma nova capacidade, geralmente um usuário ou cliente do sistema. Visando capturar o que o usuário deseja fazer com o sistema e o valor que ele espera obter.

A tabela logo mais abaixo (Figura 18) ilustra as histórias de usuário mapeadas:

5.7.2 Requisitos não funcionais

Ainda de acordo com Sommerville (SOMMERVILLE, 2010), requisitos não funcionais devem ser restrições sobre os serviços ou funções oferecidos pelo sistema. Eles podem incluer restrições de tempo, espaço, confiabilidade, segurança e ressaltar padrões de qualidade que o sistema deve atender. Diferente dos requisitos funcionais, que descrevem o que o sistema faz, os requisitos não funcionais descrevem como o sistema deve se comportar.

Na tabela abaixo (Figura 19) estão ilustrados os requisitos não funcionais mapeados:

Figura 18 – Tabela de Histórias de Usuário.
Fonte: Elaborado pelo Autor.

História de Usuário	Descritivo	RF referentes
HU01	Como integrante de uma equipe de desenvolvimento ágil, gestor ou membro da equipe comercial, desejo visualizar informações sobre tarefas, projetos e sprints para obter uma visão clara do histórico recente das equipes de desenvolvimento de maneira prática e acessível.	RF01, RF02 e RF07.
HU02	Como integrante de uma equipe de desenvolvimento ágil, gestor ou membro da equipe comercial, desejo continuar visualizando determinadas métricas ágeis que já utilizamos rotineiramente na plataforma Jira.	RF03 e RF07.
HU03	Como gestor de equipes ágeis ou membro da equipe comercial desejo visualizar métricas e indicadores individuais dos membros das equipes de desenvolvimento de software.	RF04 e RF07
HU04	Como integrante de uma equipe de desenvolvimento ágil, gestor ou membro da equipe comercial, desejo visualizar métricas personalizadas relacionadas aos projetos e sprints das equipes, que destaquem aspectos relevantes dos dados, para que, por meio da análise dessas informações, eu possa tomar melhores decisões significativas.	RF05 e RF07
HU05	Como integrante de uma equipe de desenvolvimento ágil, gestor ou membro da equipe comercial, necessito visualizar informações atualizadas sobre o cumprimento das OKRs predefinidas, a fim de acompanhar em tempo real o progresso no atingimento das metas estabelecidas.	RF06 e RF07

Figura 19 – Tabela de Requisitos não Funcionais.
Fonte: Elaborado pelo Autor.

RNF01	A aplicação deve ser desenvolvida como um sistema web, permitindo acesso simplificado a partir de qualquer dispositivo, seja desktop ou móvel, compatível com os navegadores Google Chrome, Mozilla Firefox e Microsoft Edge.
RNF02	O back-end do sistema deve ser implementado na linguagem Dart, garantindo robustez e eficiência na execução das funcionalidades do sistema.
RNF03	O front-end da aplicação deve ser desenvolvido em React, facilitando a criação e exibição dos gráficos necessários para a visualização das métricas.
RNF04	O sistema deve utilizar a tecnologia Docker para simplificar a hospedagem e execução em diferentes servidores e ecossistemas, promovendo maior flexibilidade e portabilidade.
RNF05	A aplicação deve consumir dados da plataforma Jira por meio de integração via API, possibilitando a comunicação direta e a obtenção de informações relevantes para o sistema.

5.8 CONCLUSÃO DE ANÁLISE

Neste capítulo, foram discutidos aspectos relacionados aos processos, produtos e equipes da organização utilizada como referência para o desenvolvimento da aplicação apresentada neste trabalho. Além disso, foram analisadas as métricas e os OKRs relevantes para o dia a dia das equipes ágeis, os quais agregam valor ao sistema, juntamente com a análise dos requisitos funcionais e não funcionais necessários para o desenvolvimento.

Com base nessas informações, foram estabelecidos os insumos iniciais para dar continuidade à etapa de desenvolvimento da aplicação proposta, bem como definidas as diretrizes essenciais para alinhar os objetivos do sistema de forma estruturada.

6 DESENVOLVIMENTO

Este capítulo apresenta a arquitetura geral da ferramenta, juntamente com o descritivo das tecnologias que foram utilizadas. Discorre também acerca das decisões significativas de projeto, configuração do ambiente para respectiva execução, detalhes da automatização do consumo de dados da plataforma Jira e informações de banco de dados.

Ao final é apresentada breve documentação de navegação da ferramenta, ilustrando e descrevendo principais telas e menus visando facilitar a utilização, além de um detalhamento sobre as métricas contidas na aplicação e sua forma de utilização.

6.1 ARQUITETURA GERAL DA APLICAÇÃO

O sistema em foco nesse trabalho trata-se de uma aplicação web com diferentes linguagens no *front-end* e no *back-end*. Para o *back-end* a linguagem escolhida foi Dart[1], enquanto que para o front-end a linguagem escolhida foi Javascript, através do *framework* React [2].

Dart é uma linguagem de programação desenvolvida pela Google, focada em oferecer desempenho elevado e eficiência para aplicações modernas, especialmente para projetos voltados ao back-end. Combinando tipagem estática opcional e suporte nativo à programação assíncrona, Dart permite a criação de sistemas robustos e escaláveis, adequados para arquiteturas de microsserviços e processamento de dados em tempo real. Sua compilação AOT (*Ahead-of-Time*) proporciona tempos de execução otimizados, enquanto a interoperabilidade com diversas plataformas torna-a uma linguagem versátil. Esses atributos posicionam Dart como uma alternativa promissora para projetos que exigem alto desempenho, manutenibilidade e escalabilidade no backend.

React é uma biblioteca JavaScript de código aberto, desenvolvida pelo Facebook, projetada para a construção de interfaces de usuário interativas e eficientes. Com uma arquitetura baseada em componentes, React promove a reutilização de código e a modularidade, permitindo que desenvolvedores criem interfaces de maneira organizada e escalável. Seu sistema de renderização virtual melhora o desempenho ao atualizar somente os elementos necessários da página em vez de recarregar toda a interface, resultando em experiências de usuário mais rápidas e fluidas.

Tanto o *front-end* quanto o *back-end* executam de maneira conteinerizada através do Docker[3], isolando cada parte da aplicação em ambientes independentes e consistentes. Essa abordagem permite um gerenciamento simplificado de dependências e configurações, além de escalabilidade e distribuição de recursos eficientes. Com o Docker, o *front-end* e o *back-end* podem se comunicar de maneira integrada e segura por meio de redes de containers, otimizando todo o processo.

[1] Site da tecnologia citada: https://dart.dev/
[2] Site da tecnologia citada: https://react.dev/
[3] Site da tecnologia citada: https://www.docker.com/

Docker é uma plataforma de virtualização baseada no conceito de containers, projetada para simplificar a criação, o desenvolvimento e a implantação de aplicações. Através desse conceito anteriormente citado, o Docker permite que softwares sejam empacotados com todas as suas dependências e configurações, garantindo que o ambiente de execução seja o mesmo em desenvolvimento, teste e produção. Entre os principais benefícios do Docker, destacam-se a consistência no ciclo de vida do software, a escalabilidade rápida e flexível, e o uso eficiente de recursos do sistema, que permite que múltiplos containers rodem simultaneamente com um menor consumo de memória em comparação com máquinas virtuais.

O padrão de projeto adotado para a aplicação em questão é o MVC (*Model, View e Controller*), uma arquitetura amplamente utilizada para estruturar aplicações de forma modular e organizada. No padrão MVC, a camada Model representa os dados e a lógica de negócio, View é responsável pela interface e exibição ao usuário, e Controller atua como intermediário, gerenciando as interações entre Model e View. Esse padrão promove a separação de responsabilidades, facilitando a manutenção e escalabilidade do sistema, ao mesmo tempo que melhora a testabilidade e a reutilização de código (FOWLER, 2003).

Abaixo está uma ilustração do código da classe HomeController (Figura 20). Ela atua, neste exemplo citado, como a camada de controller no padrão MVC da aplicação. Ela é responsável por se comunicar com a API e buscar dados, encapsulando a lógica de comunicação e preparação dos dados para que a view (Home.js) apenas consuma os resultados.

Figura 20 – Trecho de código - Exemplo de Controller.
Fonte: Elaborado pelo Autor.

```
async getSprints() {

    try {
        const response = await axios.post(`${HomeController.apiURL}/sprints/`, {})

        return response.data['sprints']
    } catch (error) {
        console.log(error)

        return []
    }
}
```

Neste exemplo de código acima `getSprints` faz chamadas à API para recuperar dados e os retornam prontos para serem utilizados pela camada de visualização

Em seguida, é apresentado um trecho do arquivo "Home.js", Essa classe descreve um exemplo de camada de *view* da aplicação e utiliza os métodos da classe `HomeController` para obter e exibir os dados na interface. Ela se preocupa apenas em exibir os dados que recebe do controller.

No exemplo abaixo (Figula 21), a camada de *view* faz uso de `homeController.getSprints()` para buscar os dados e atualizar o estado. A interface exibe os dados diretamente usando `sprints.map` e `teams.map`, sem precisar lidar com a lógica de comunicação com a API.

Figura 21 – Trecho de código - Exemplo de View.
Fonte: Elaborado pelo Autor.

```
useEffect(() => {
    const fetchSprints = async () => {
        const sprintList = await homeController.getSprints()
        setSprints(sprintList)
    }
    fetchSprints()
}, [])

return (
    <Box padding="32px">
        <Typography variant="h5">Sprints e Equipes</Typography>
        <Box>
            <Typography variant="h6">Sprints</Typography>
            {sprints.map((sprint, index) => (
                <Typography key={index}>{sprint.name}</Typography> // Exibindo os nomes dos sprints
            ))}
        </Box>
```

No exemplo mais abaixo (Figura 22) a classe `SprintModel` apresenta uma representação estrutural inicial de uma sprint e pode incluir métodos para manipular ou formatar informações específicas do sprint.

Figura 22 – Trecho de código - Exemplo de Model.
Fonte: Elaborado pelo Autor.

```
export class SprintModel {
    constructor(data) {
        this.id = data.id;
        this.name = data.name;
        this.startDate = new Date(data.startDate);
        this.endDate = new Date(data.endDate);
        this.status = data.status;
    }

    // Método para verificar se o sprint está ativo
    isActive() {
        const now = new Date();
        return now >= this.startDate && now <= this.endDate;
    }
```

A camada de modelo (Model) é representada pelas classes que definem a estrutura dos dados que quando trafegados através das camadas da aplicação são organizados em formato JSON. Esses dados são retornados para a visualização (view) por meio de requisições processadas pela classe de controle mencionada anteriormente. Neste caso, a camada Model utiliza estruturas JSON genéricas, permitindo que a estrutura padrão dos dados que a aplicação espera seja facilmente reutilizada tanto para consultas a um banco de dados quanto para a API do JIRA, tema que será aprofundado na seção mais adiante.

6.2 DECISÕES DE PROJETO

Para garantir a consistência e a disponibilidade dos dados no sistema, foi adotada a estratégia de manter uma instância local de banco de dados destinada ao armazenamento de uma amostragem dos dados relacionados a equipes, sprints, projetos e tarefas. Essa decisão arquitetural foi motivada pela possibilidade de indisponibilidade ou instabilidade nas respostas da API do Jira, que constitui a fonte primária de dados da aplicação.

Com essa abordagem, busca-se mitigar potenciais riscos de interrupção, assegurando que informações essenciais permaneçam acessíveis aos usuários. A obtenção dos dados do Jira é realizada de forma síncrona e em tempo real. No entanto, em situações de indisponibilidade da API, a instância local do banco de dados assume automaticamente a responsabilidade de responder às requisições.

É importante destacar, contudo, que a atualização da instância local ocorre exclusivamente por meio de um gatilho acionado manualmente por um desenvolvedor com privilégios de administrador no banco de dados e acesso ao código-fonte. Esse processo envolve a execução de um script específico que atualiza os dados referentes aos últimos três meses.

Com essa estrutura, a aplicação pode continuar operando com dados possivelmente atualizados mesmo em momentos de instabilidade ou falhas temporárias na conexão com a API, reduzindo o impacto nas operações do usuário e garantindo a confiabilidade na apresentação das métricas e indicadores de produtividade das equipes ágeis. Além disso, essa estratégia possibilita tempos de resposta mais rápidos e uma experiência de uso contínua, pois consultas recorrentes são realizadas diretamente na base de dados local, desde que recentemente atualizada, dispensando requisições constantes à API e promovendo eficiência no consumo de dados.

Outro aspecto relevante nas decisões de projeto diz respeito à visualização de informações sensíveis da organização em um contexto externo, de modo a proteger dados sensíveis como nomes de clientes, dados de colaboradores, informações sigilosas de projetos e tarefas, bem como descrições técnicas que possam revelar propriedade intelectual. Para mitigar riscos de exposição dessas informações confidenciais e atender aos requisitos da Lei Geral de Proteção de Dados (LGPD), optou-se por implementar um repositório de dados espelhado dentro da plataforma Jira, porém, contendo uma versão anonimizada dos dados sensíveis. Essa abordagem permite preservar a integridade e a utilidade dos dados necessários para o cálculo de métricas e indicadores de produtividade, sem comprometer a segurança das informações confidenciais. Assim, o banco de dados local que armazena um subconjunto das informações do Jira também reflete esta política de anonimização, garantindo que os dados sensíveis sejam devidamente descaracterizados, mas sem prejuízo à qualidade e precisão das análises a serem realizadas.

6.3 CONFIGURAÇÃO DO AMBIENTE

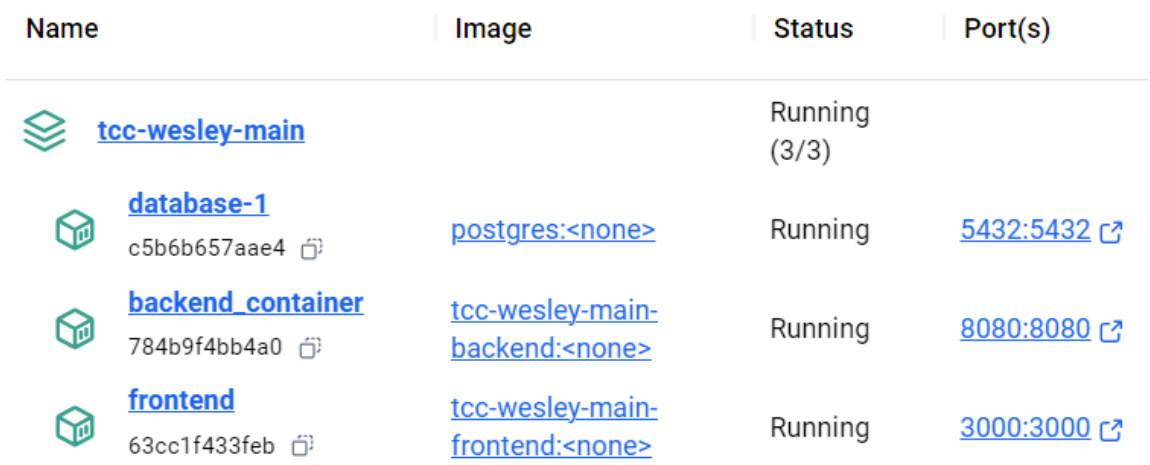
A configuração do ambiente para execução da aplicação web objeto deste trabalho é simplificada graças ao uso de contêineres Docker. Esse processo de containerização permite

uma instalação e execução padronizadas e facilita a replicação do ambiente.

Para inicializar o ambiente e executar o projeto, basta navegar até o diretório raiz do projeto via prompt de comando ou terminal e utilizar o comando `docker compose up --build`. Esse comando configura e inicia os contêineres necessários para a aplicação, compreendendo três instâncias principais: uma para o front-end, outra para o back-end e uma terceira para o banco de dados, que utiliza PostgreSQL.

Após o tempo de inicialização, a aplicação estará disponível e acessível a partir da tela de login, que pode ser acessada diretamente pelo navegador na URL: `http://localhost:3000/`.

Figura 23 – Ilustração de containeres docker em execução.
Fonte: Docker Desktop.



A interrupção do ambiente é realizada de maneira igualmente simples, utilizando o comando `docker compose down` no terminal ou prompt de comando, o que encerra a execução dos contêineres. Para remover o histórico e liberar os recursos ocupados pela aplicação, retornando o ambiente ao estado inicial, pode-se utilizar o comando `docker system prune -a`. Essa operação elimina arquivos temporários e dados residuais dos contêineres utilizados, garantindo que o ambiente esteja pronto para futuras execuções.

6.4 CONSUMO JIRA API

O Jira oferece uma interface para desenvolvedores que desejam criar aplicações integradas com sua plataforma, chamada de API REST. Por padrão, todas as solicitações para `https://{sua_instancia_jira}.atlassian.net/rest/api/` recebem a versão ativa da API REST do Jira.

Neste trabalho, são realizadas diversas requisições do tipo GET para a API do Jira. Cada recurso, como dados de projetos, sprints e tarefas, é acessado por meio de um endpoint específico, que pode incluir headers de autorização e Content-Type, conforme exigido pela autenticação e documentação da API do Jira. A API permite uma integração segura e eficiente,

utilizando autenticação Basic Auth com token de API, garantindo acesso controlado aos dados do Jira.

6.4.1 Implementação das requisições

Nesta subseção foi ilustrado exemplo de uma implementação de requisição para a API REST do Jira, de modo a exemplificar através de uma requisição o padrão que é replicado nas demais.

O método `fromJson` é um construtor de fábrica da classe Sprint que cria uma instância da classe a partir de um objeto JSON (um Map em Dart). Ele mapeia os valores JSON para os atributos da classe, garantindo valores padrão caso o JSON forneça `null`. Esse método torna a criação de instâncias de Sprint mais segura e resiliente contra dados incompletos no JSON.

Figura 24 – Trecho de código - Instância de Sprint a partir de um JSON.
Fonte: Elaborado pelo Autor.

```dart
// Método para criar uma instância de Sprint a partir de um JSON
factory Sprint.fromJson(Map<String, dynamic> json) {
  return Sprint(
    code: json['code'] ?? '',
    points: json['points'] ?? 0,
  );
}
```

A seguir temos um exemplo de definição de uma interface (*abstract class*) e uma implementação concreta dessa interface. A classe concreta `SprintRepositoryJira` implementa `SprintRepository` e usa dados de autenticação como `jiraBaseUrl`, `apiToken` e `username` para se conectar ao Jira API. Esses valores são inicializados via construtor nomeado, onde `required` exige que sejam fornecidos na criação da instância.

Essa estrutura promove certa flexibilidade, permitindo diferentes implementações de `SprintRepository` para obter dados de sprints de diversas fontes. Na aplicação proposta, essas fontes dizem respeito à API do Jira ou à instâncias de bancos de dados pré-definidas.

No trecho de código seguinte (Figura 26) é ilustrada a definição de um método para autenticação via headers de autorização e um método para obter uma lista de sprints a partir da API do Jira:

Através da geração dos headers de autorização para a chamada ao Jira API são codificados `username` e `apiToken` em `base64`, criando um `token Basic Auth`, que por sua vez, retorna um mapa com os `headers` de `Authorization` e `Content-Type`. Na sequência é feita uma requisição `HTTP GET` para a API de sprints do Jira em que o URL de requisição é construído através dos parâmetros `jiraBaseUrl` e `boardId`. Em caso de sucesso, o corpo da resposta JSON é decodificado e dados são mapeados para objetos Sprint usando o construtor fromJson.

Figura 25 – Trecho de código - Contrato que define métodos a serem sobrescritos.
Fonte: Elaborado pelo Autor.

```dart
// Contrato que define quais métodos um repositório de sprints precisa sobrescrever
abstract class SprintRepository {
  Future<List<Sprint>> getAllSprints();
}

// Repositório que usa o Jira API
class SprintRepositoryJira implements SprintRepository {
  final String jiraBaseUrl;
  final String apiToken;
  final String username;

  SprintRepositoryJira({
    required this.jiraBaseUrl,
    required this.apiToken,
    required this.username,
  });
```

Através da replicação deste padrão em requisições do projeto, o código encapsula a lógica de autenticação e de requisição `HTTP` tornando o acesso aos dados via Jira API mais seguro e eficiente.

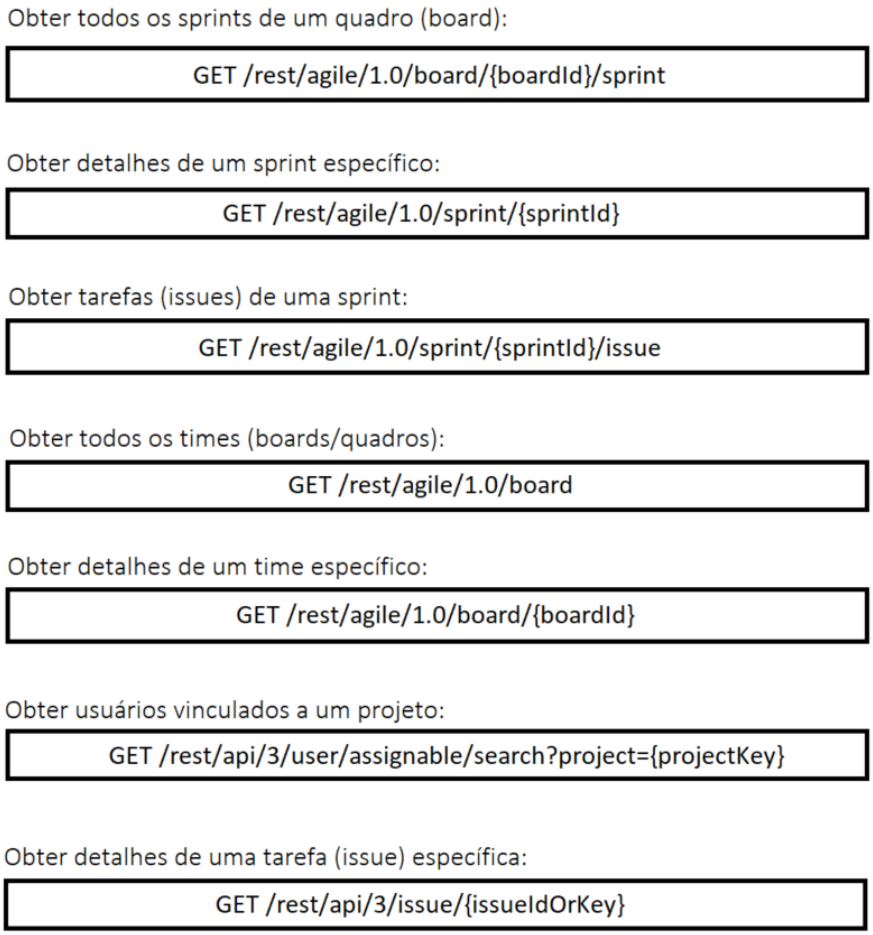
Figura 26 – Trecho de código - Obter *headers* de autorização para requisição ao Jira e posterior exemplo de requisição.
Fonte: Elaborado pelo Autor.

```
  // Método para obter headers de autorização para requisição ao Jira
Map<String, String> _getHeaders() {
  final basicAuth =
      'Basic ' + base64Encode(utf8.encode('$username:$apiToken'));
  return {
    'Authorization': basicAuth,
    'Content-Type': 'application/json',
  };
}

@override
Future<List<Sprint>> getAllSprints() async {
  final response = await http.get(
    Uri.parse('$jiraBaseUrl/rest/agile/1.0/board/{boardId}/sprint'),
    headers: _getHeaders(),
  );

  if (response.statusCode != 200) {
    throw Exception('Falha ao carregar sprints: ${response.body}');
  }

  final List<dynamic> data = jsonDecode(response.body)['values'];

  return data.map((json) => Sprint.fromJson(json)).toList();
}
```

6.4.2 Métodos utilizados

Ao longo do desenvolvimento da aplicação-alvo dados de sprints, times, membros de times e tarefas são consumidos da API REST do Jira. Abaixo está uma lista do principais métodos/endpoints disponibilizados pela API que são utilizados para a referida integração na aplicação:

Figura 27 – Lista dos principais endpoints para integração.
Fonte: Elaborado pelo Autor.



6.5 BANCO DE DADOS

A aplicação desenvolvida conta com uma instância local de banco de dados, implementada por meio de containerização Docker, o que garante maior controle e disponibilidade dos dados. Essa arquitetura permite o armazenamento de informações essenciais para o funcionamento contínuo da ferramenta, mesmo diante de eventuais instabilidades na rede ou na fonte externa de dados. Além disso, o banco de dados local possibilita o gerenciamento eficiente de dados de autenticação e autorização, essenciais para a segurança da aplicação e para o controle de acesso dos usuários.

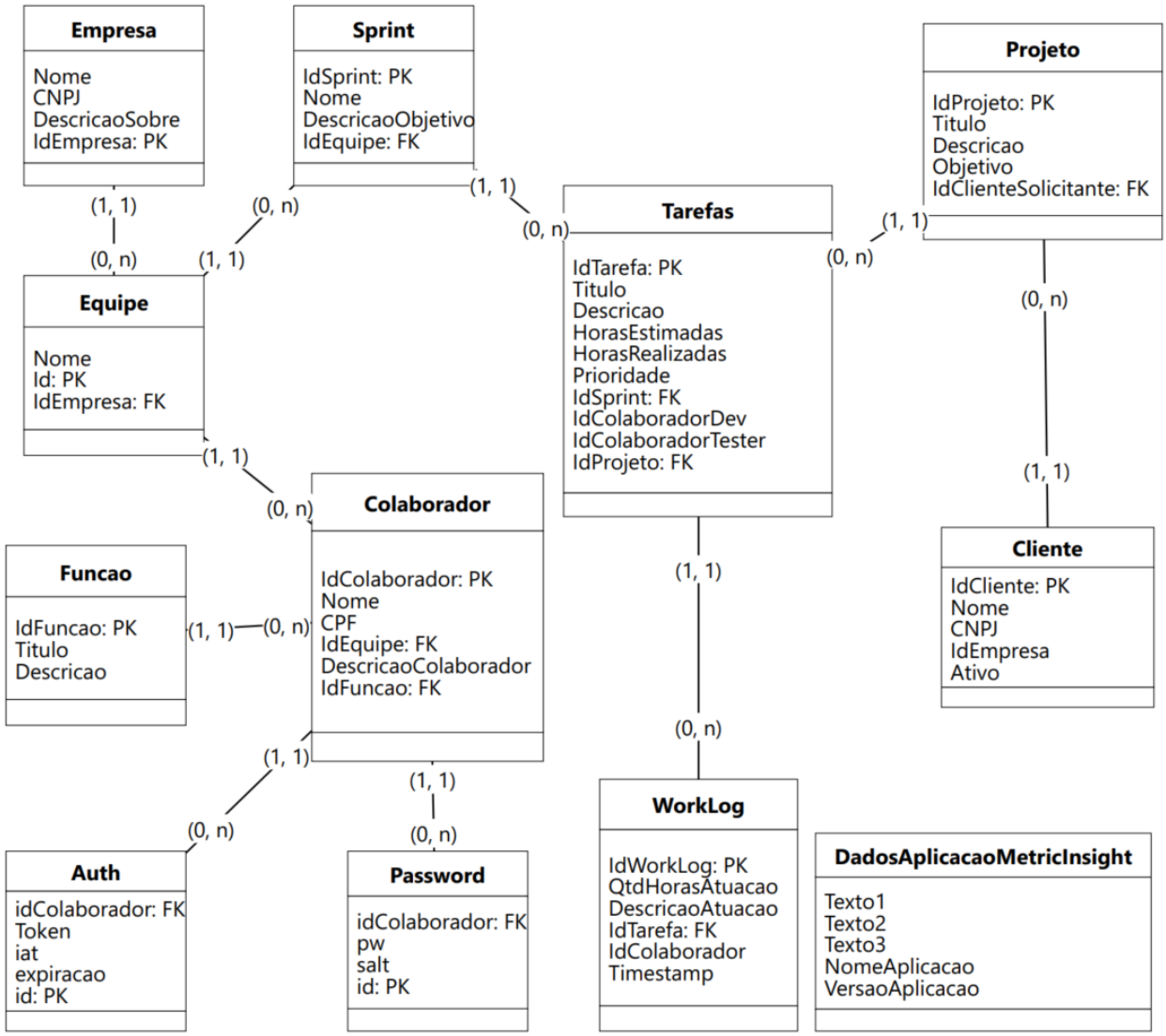
O banco de dados escolhido para o projeto foi o PostgreSQL, uma solução relacional amplamente adotada pela sua robustez, confiabilidade e suporte a uma ampla gama de tipos de dados e operações complexas. A configuração em Docker facilita a replicação do ambiente de desenvolvimento e permite que o PostgreSQL seja executado de maneira isolada, o que contribui para a escalabilidade e manutenção simplificada do sistema. Essa abordagem de conteinerização oferece flexibilidade para ajustes de configuração e facilita a integração contínua, tornando o ambiente de banco de dados mais seguro e estável.

O modelo de dados contempla tabelas fundamentais para o contexto de gestão ágil,

como Times, Projetos, Tarefas e Colaboradores, estabelecendo relacionamentos que refletem a estrutura organizacional e o fluxo de trabalho das equipes. Esses relacionamentos permitem uma visão integrada dos dados, facilitando a análise e interpretação das métricas e indicadores necessários para o monitoramento dos times ágeis.

O diagrama do banco de dados abaixo (Figura 28) ilustra essas conexões e demonstra como as tabelas se inter-relacionam para atender aos requisitos da aplicação:

Figura 28 – Modelagem do Banco de Dados.
Fonte: Elaborado pelo Autor.



6.6 TELAS E MENUS

A seguir, são apresentadas capturas de telas e menus relevantes da aplicação, com o objetivo de introduzir o usuário à estrutura e à navegação básica do sistema. Essas imagens ilustram as principais funcionalidades e opções de menu disponíveis, proporcionando uma visão inicial da interface e dos recursos de monitoramento de métricas para equipes ágeis.

A Figura 29 exibe a Tela de Login da aplicação, ponto inicial de acesso ao sistema. Em seguida, a Figura 30 apresenta a Tela Inicial, acessada após o login, que contém um breve descritivo sobre a motivação e os objetivos da aplicação.

É importante destacar que o login na aplicação é realizado por meio das mesmas credenciais organizacionais utilizadas para acessar a plataforma Jira, as quais também são empregadas nas requisições para comunicação com a API.

As permissões relacionadas ao escopo de visualização de cada usuário no sistema são determinadas com base em suas características específicas, especialmente quanto ao seu perfil — se é um gestor ou não — e à equipe à qual pertence.

As figuras subsequentes apresentam as seguintes telas e menus principais da aplicação:

- Menu Principal do Sistema: permite a navegação para as principais funcionalidades da aplicação.
- Menu de Gerenciamento de Conta: disponibiliza opções relacionadas ao gerenciamento da conta do usuário logado.
- Menu de Descrição dos Times e Membros: apresenta um detalhamento dos times e de seus respectivos membros.
- Menu de Detalhamento das Sprints e Tarefas: exibe uma visão detalhada das sprints e das tarefas associadas a cada uma.
- Menu de Descrição dos Projetos: mostra os projetos em andamento e finalizados, com as respectivas tarefas vinculadas.
- Menu de Acesso às Métricas e Monitoramento de OKRs: reúne as métricas ágeis padrão, métricas customizadas e individuais, além do monitoramento de OKRs (*Objectives and Key Results*) para avaliação de desempenho e resultados.

Figura 29 – Tela de Login Metric Insight.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

Figura 30 – Tela Inicial Metric Insight.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

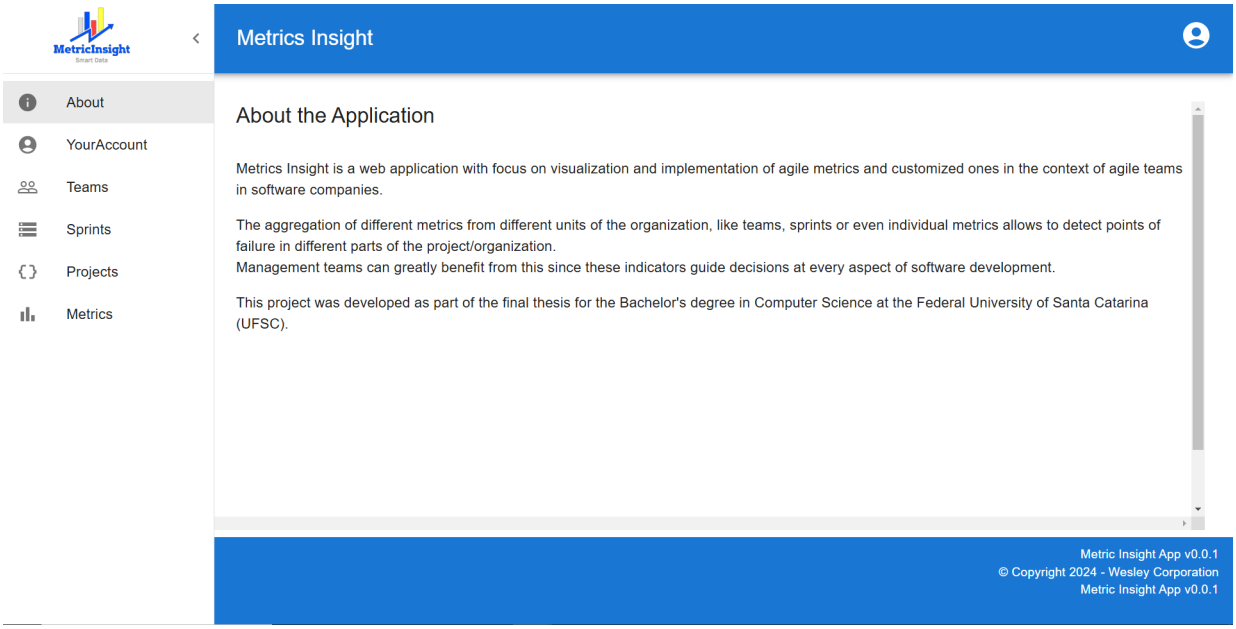


Figura 31 – Menu principal da Aplicação.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

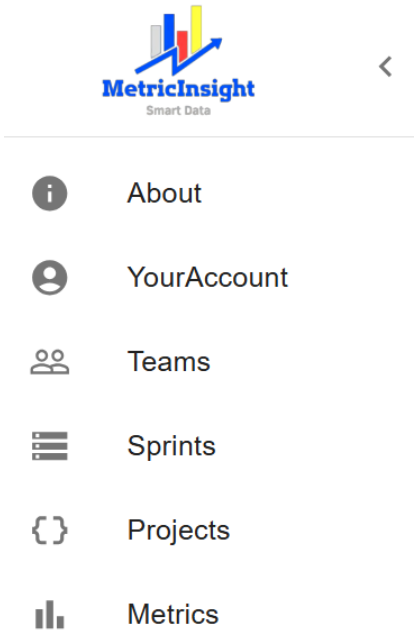


Figura 32 – Tela de Times.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

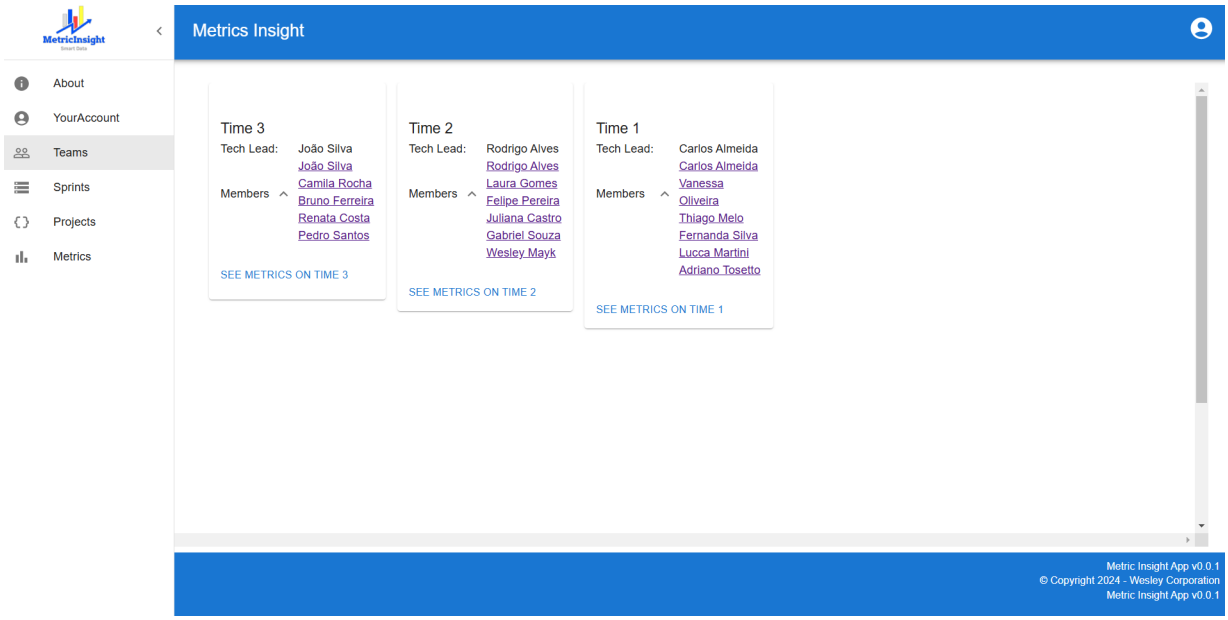
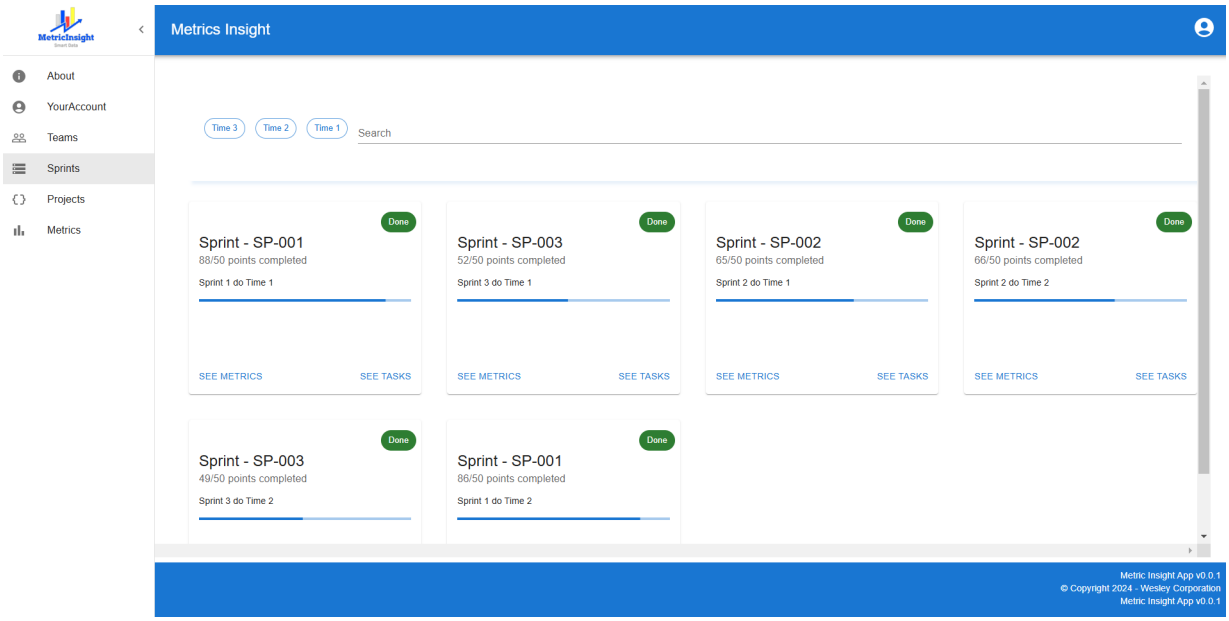


Figura 33 – Tela de Sprints.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



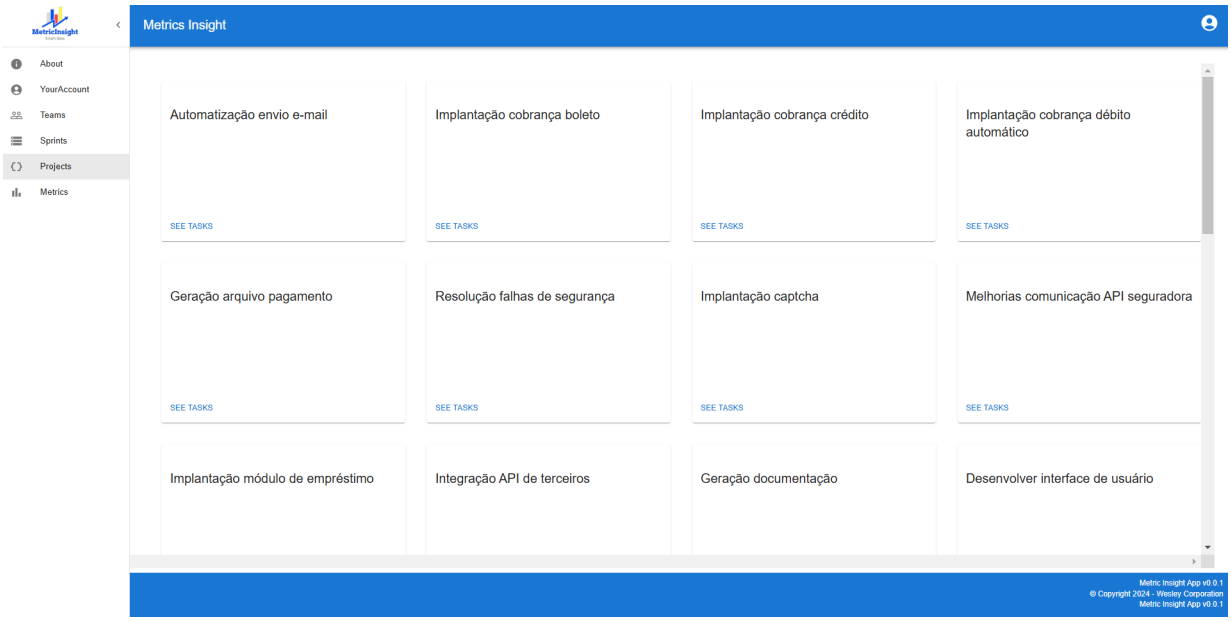
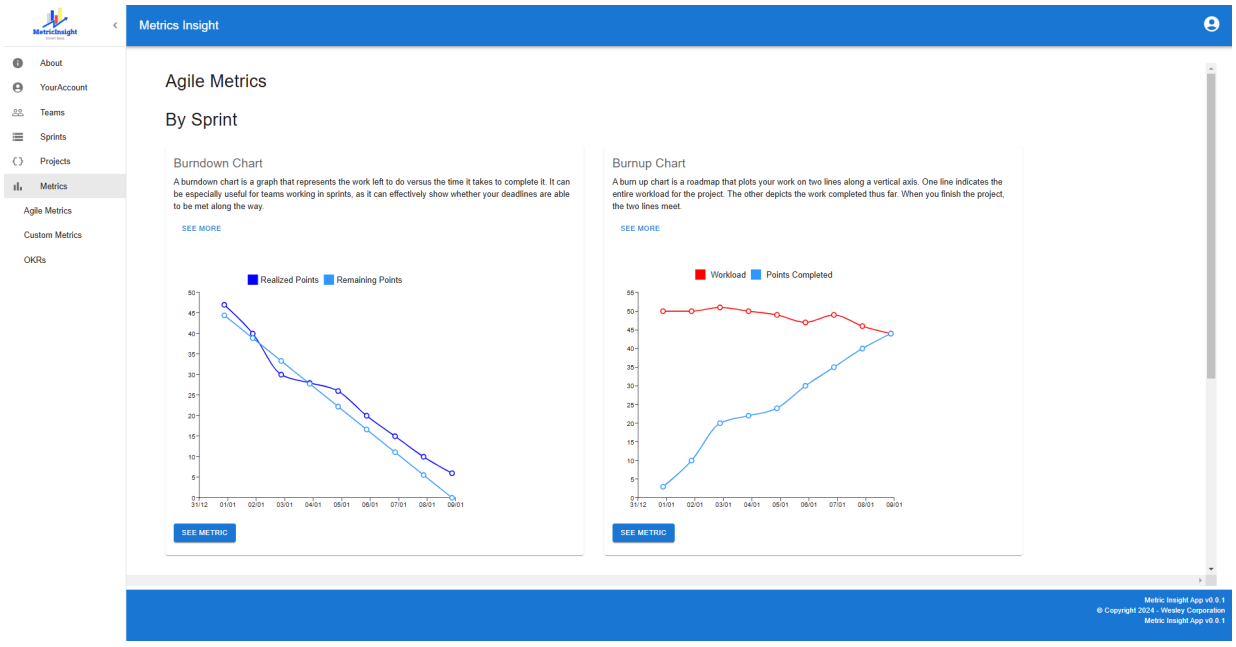


Figura 34 – Tela de Projetos.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

Figura 35 – Menu de Métricas.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



6.7 MÉTRICAS IMPLEMENTADAS

Esta seção tem como objetivo ilustrar as métricas anteriormente descritas no capítulo de análise deste trabalho e, posteriormente, implementadas na aplicação desenvolvida, abrangendo tanto métricas ágeis tradicionais quanto métricas customizadas e individuais.

6.7.1 Métricas Ágeis

Figura 36 – Burndown Chart
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

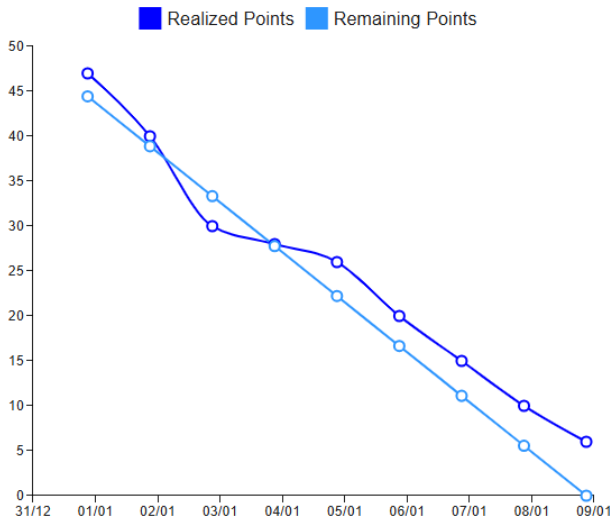


Figura 37 – Parâmetros alteráveis no Burndown Chart
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

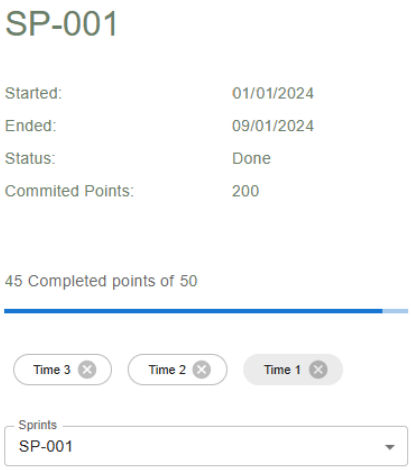
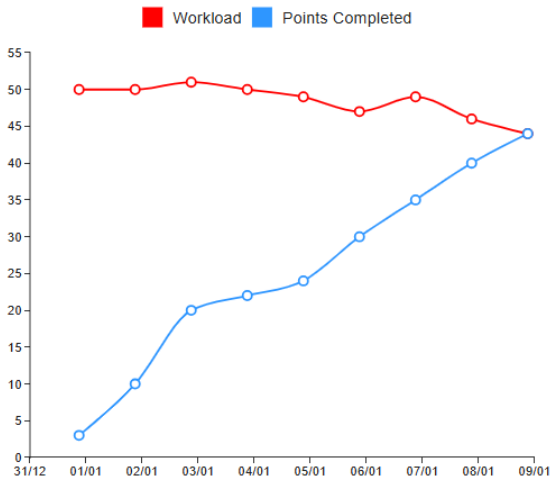
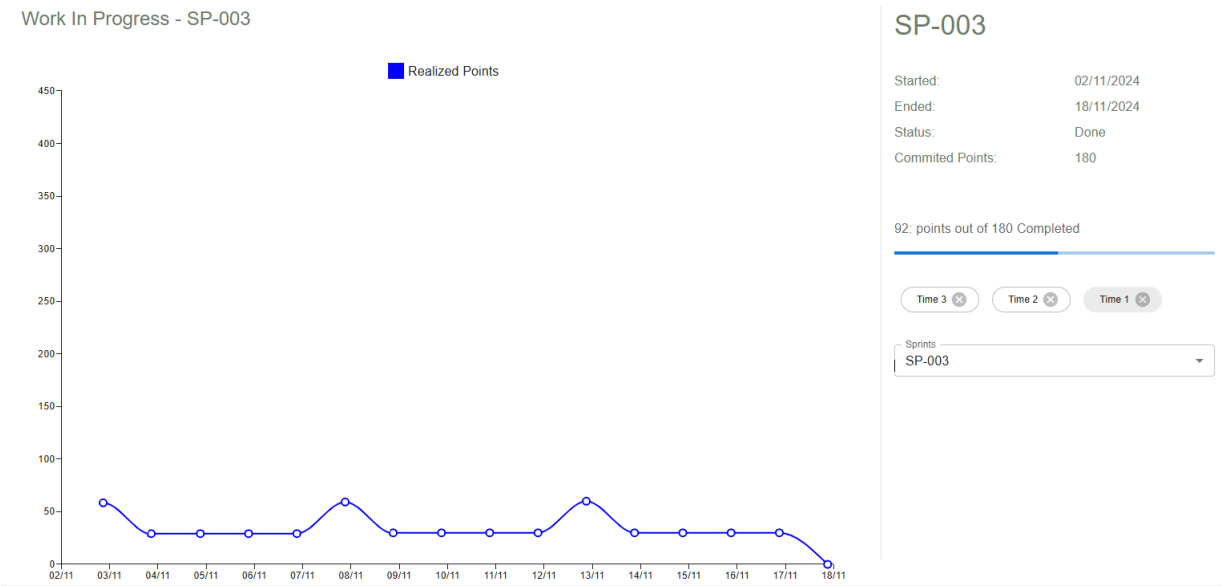


Figura 38 – Burnup Chart
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



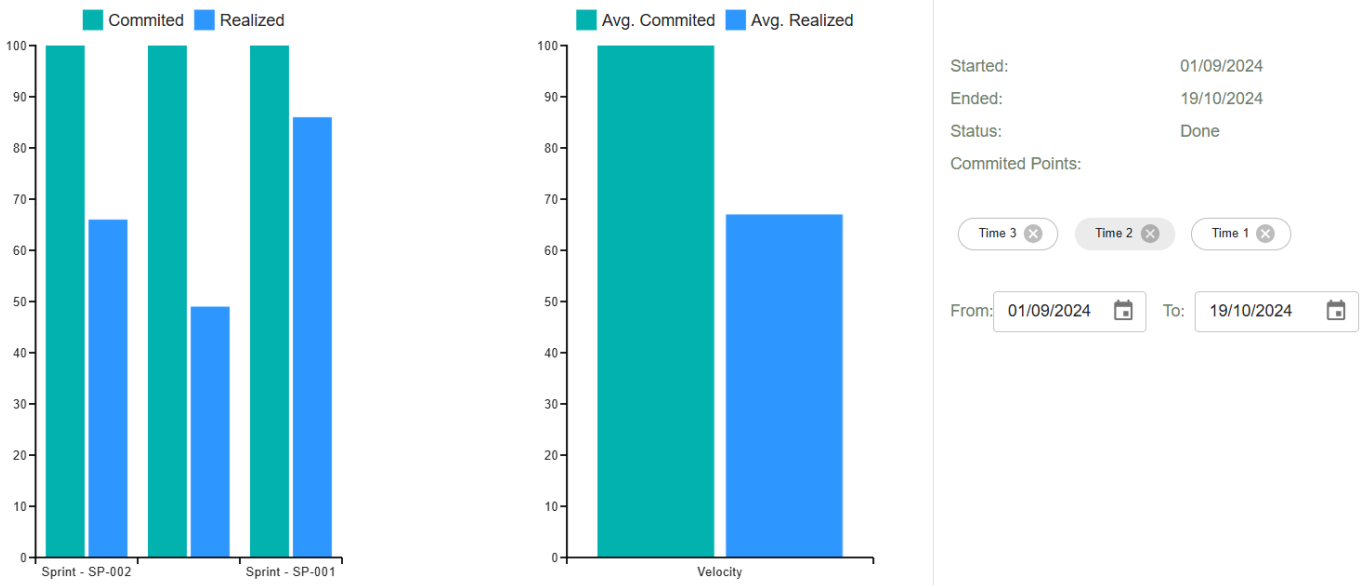
Para o *Burnup Chart* (Figura 38) aplicam-se os mesmos parametros alteráveis existentes para o *Burndow Chart* (Figura 37). Isto é, sendo possível filtrar entre diferentes times e diferentes sprints do respectivo time selecionado.

Figura 39 – Work in Progress
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



Referente a métrica *Work in Progress* vista acima (Figura 39), destaca-se o fato e que uma linha estável indica que o fluxo de trabalho está equilibrado, com entradas e saídas bem gerenciadas. De modo que ilustra que o time não está concluindo mais tarefas do que inicia e também não está iniciando mais tarefas do que conclui.

Figura 40 – Velocity Chart
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



6.7.2 Métricas Customizadas

Figura 41 – Bugs por Cliente.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

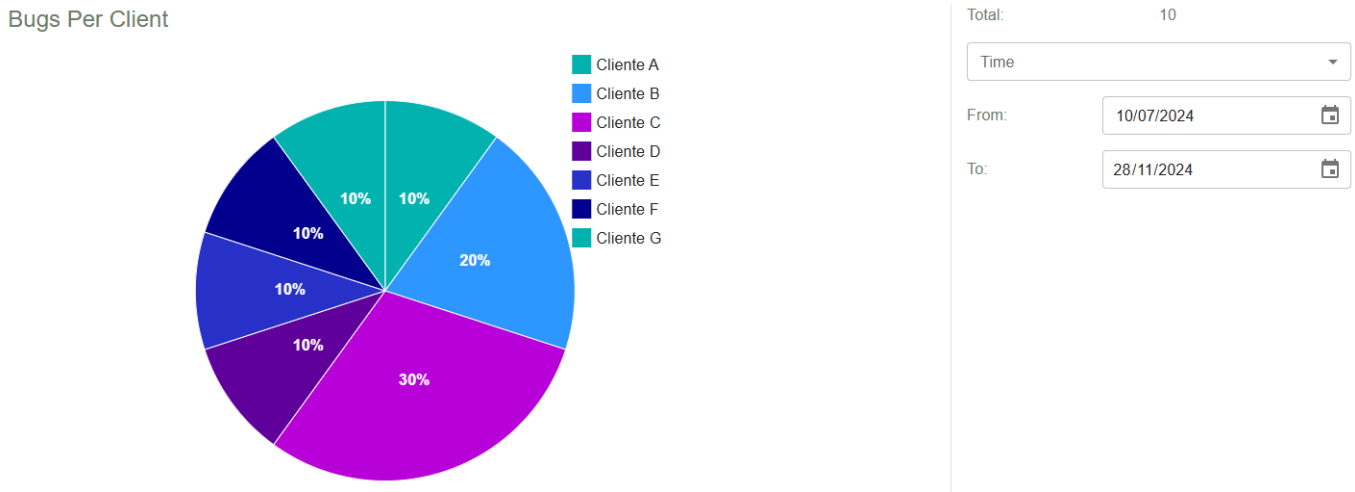
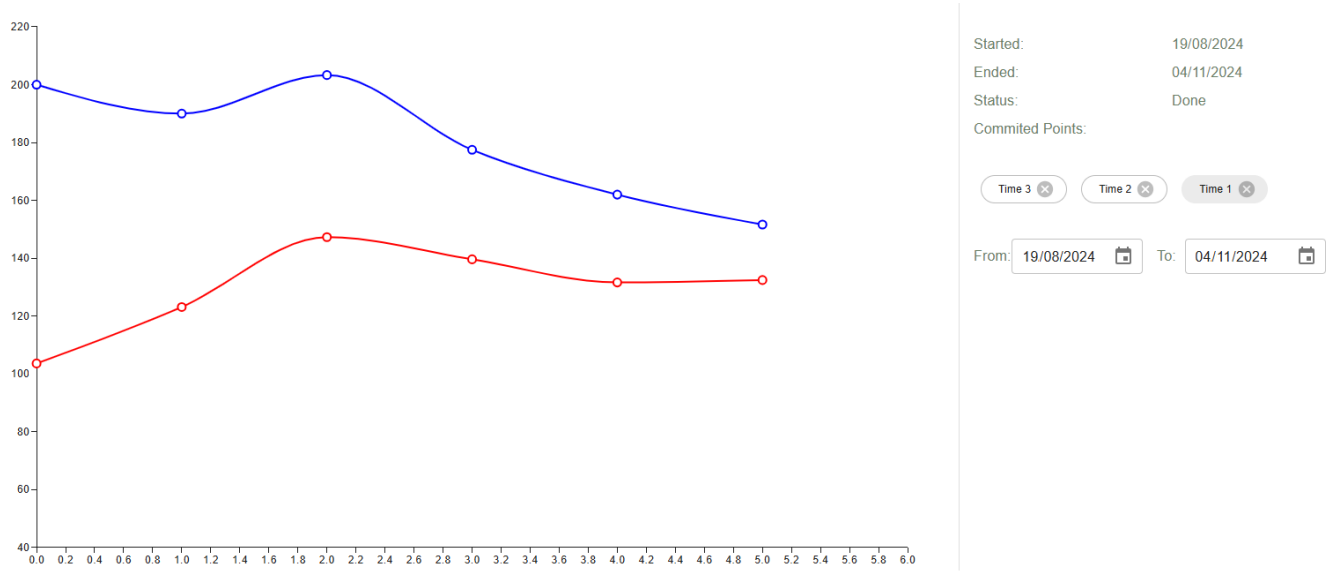


Figura 42 – Média Móvel.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



A métrica acima, *Moving Average Velocity* (Figura 42), ilustra através de duas linhas no gráfico uma tendência dos pontos comprometidos e realizados com base no histórico de sprints recentes(conforme filtro).

Na figura abaixo (Figura 43) é descrita parte da implementação desta métrica.

O código em questão implementa um componente React chamado `MovingAvrgVelocityChart`, que exibe um gráfico de linha com as médias móveis de dois históricos de dados: pontos comprometidos e pontos realizados. Ele recebe três parâmetros:

1. `commitedPointsHistory`: histórico de pontos comprometidos.
2. `realizedPointsHistory`: histórico de pontos realizados.
3. `K`: período para cálculo da média móvel.

Para cada ponto da série, o código calcula a média dos últimos `K` valores (ou menos, se estiver no início da série) e armazena essas médias em arrays separados. Por fim, o gráfico é renderizado com duas linhas:

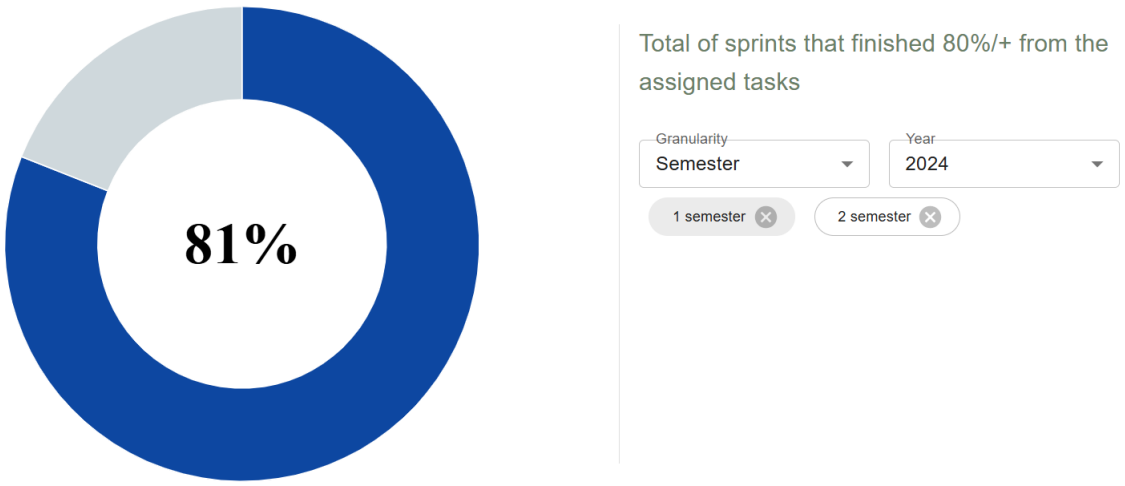
- Azul: média móvel dos pontos comprometidos.
- Vermelha: média móvel dos pontos realizados.

O gráfico facilita a visualização da tendência de desempenho, para a equipe selecionada, ao longo do tempo.

Figura 43 – Implementação Média Móvel.
Fonte: Elaborado pelo Autor.

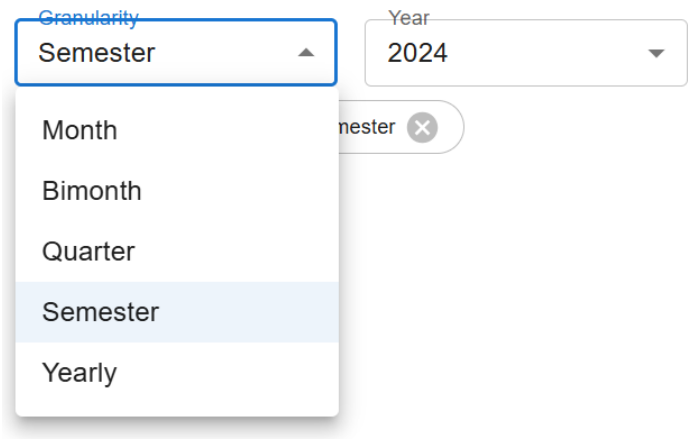
```
export const MovingAverageVelocityChart = ({ commitedPointsHistory, realizedPointsHistory, K }) => {

  const commitedPointsMovingAverages = []
  const realizedPointsMovingAverages = []

  for (let i = 0; i < commitedPointsHistory.length; i++) {
    const n = Math.min(K, i)

    const avg0 = commitedPointsHistory.slice(i - n, i + 1).reduce((acc, curr) => acc + curr, 0) / (n+1)
    const avg1 = realizedPointsHistory.slice(i - n, i + 1).reduce((acc, curr) => acc + curr, 0) / (n+1)

    commitedPointsMovingAverages.push(avg0)
    realizedPointsMovingAverages.push(avg1)

  }

  return (
    <LineChart
      xAxis={[{ data: range(0, commitedPointsHistory.length, 1) }]}
      series={[
        {
          data: commitedPointsMovingAverages,
          color: 'blue'
        },
        {
          data: realizedPointsMovingAverages,
          color: 'red'
        }
      ]}
```

Figura 44 – Exemplo de gráfico do cumprimento de OKR.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



Acerca do gráfico que ilustra o cumprimento de determinada OKR em tempo real, visto na Figura 44, destaca-se o fato da possibilidade de visualização em diferentes granularidades da informação conforme ilustrado na Figura 45 a seguir.

Figura 45 – Granularidade do Gráfico OKR.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



6.7.3 Métricas Individuais

Figura 46 – Participation Report por Cliente e por Projeto.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.

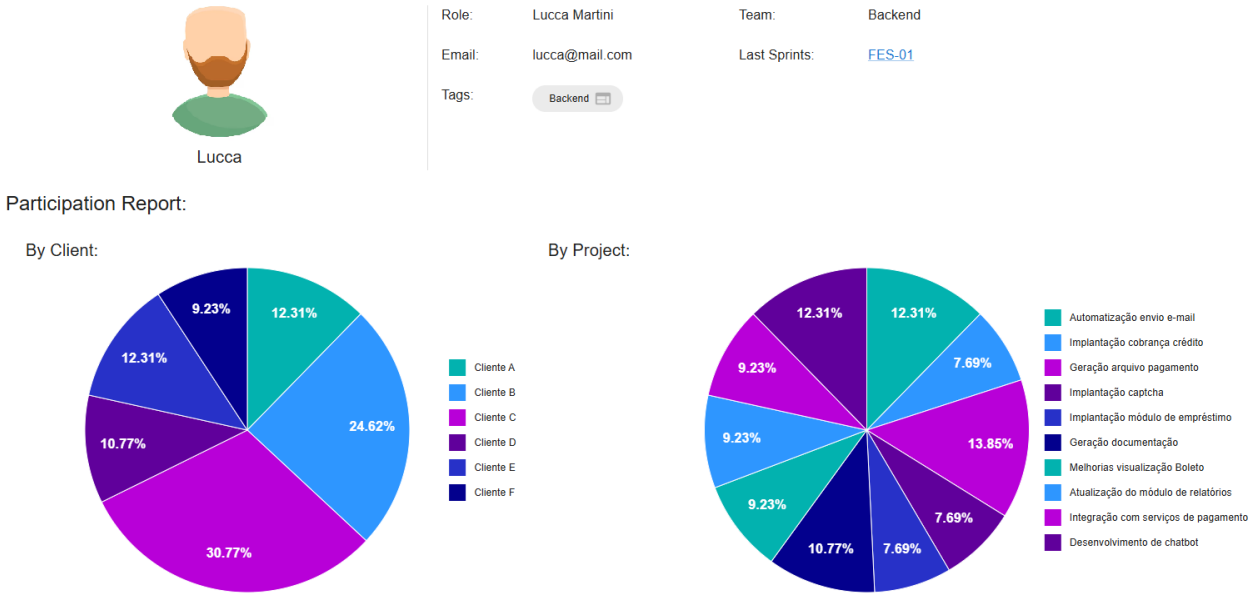
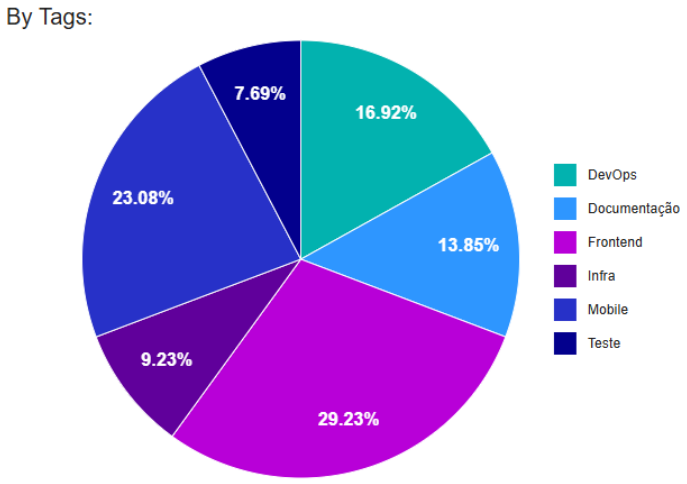
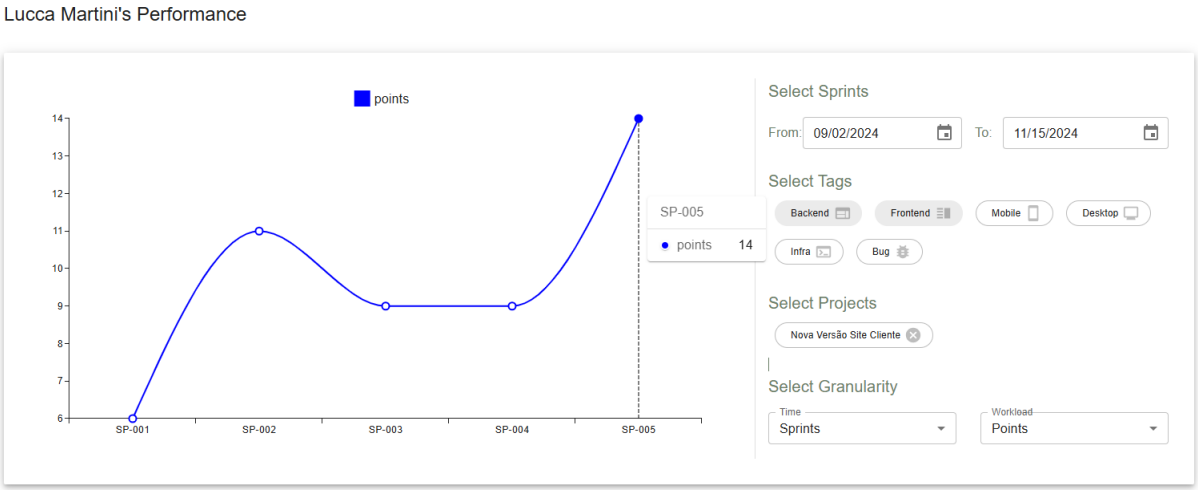


Figura 47 – Participation Report por Tags de tarefas.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



Na Figura 48 abaixo, é ilustrada a utilização da métrica de Performance Individual para um colaborador específico. A imagem mostra a quantidade de pontos atribuídos às tarefas concluídas ao final de cada sprint, considerando o intervalo de datas filtrado, e levando em conta as tags de tarefas *Frontend* e *Backend* também filtradas.

Figura 48 – Performance Individual por Tag de Tarefa, por Sprint e por Projeto.
Fonte: *Screenshot* aplicação desenvolvida pelo Autor.



7 AVALIAÇÃO

Este capítulo apresenta uma avaliação dos resultados obtidos com a implementação da aplicação desenvolvida, enfocando seus impactos na organização em que a ferramenta foi disponibilizada para utilização. O objetivo é relatar as observações do autor quanto à utilização do sistema por diferentes perfis de usuários, através de interações informais do autor com as equipes de desenvolvimento ágil, gestores das equipes e membros da equipe comercial. São analisados os efeitos práticos das funcionalidades implementadas, considerando como estas têm sido aplicadas no contexto organizacional cotidianamente.

7.1 ANÁLISE DOS RESULTADOS

A partir da análise de uso das ferramentas e métricas fornecidas pela aplicação, foi possível identificar melhorias significativas nos processos internos, especialmente em termos de eficiência, visibilidade e tomada de decisão. A aplicação demonstrou potencial para otimizar a visão geral de tarefas e sprints, facilitar o acompanhamento de projetos e aprimorar a comunicação entre áreas, contribuindo para ganhos de produtividade, qualidade no atendimento ao cliente e decisões mais assertivas.

Nas subseções seguintes, foram apresentadas análises dos aspectos mencionados, organizadas de forma segregada por equipe, visando ilustrar os ganhos identificados e os impactos específicos em cada contexto.

7.1.1 Equipes ágeis

Nas equipes ágeis, observou-se uma alta frequência de acessos diários à aplicação, especialmente para o acompanhamento da evolução das OKRs (Objectives and Key Results) diretamente relacionadas ao desempenho dos times. A cada sprint concluída ou demanda resolvida que impactasse os resultados das metas estabelecidas, a atualização em tempo real das métricas proporcionada pela aplicação tornou-se um grande atrativo para os colaboradores.

Considerando que o atingimento das metas definidas nas OKRs semestrais influencia diretamente os ganhos financeiros, como o aumento nos percentuais de PLR (Participação nos Lucros e Resultados), distribuídos ao final de cada semestre, a capacidade da aplicação de ilustrar em tempo real a evolução dos indicadores associados às OKRs transformou-se em um importante fator motivacional. Os times passaram a monitorar regularmente as metas, empenhando-se em alcançá-las e acompanhando as mudanças explícitas nas métricas apresentadas pela aplicação.

A possibilidade de visualizar métricas e indicadores de desempenho individuais na aplicação gerou grande interesse entre os membros das equipes. Por meio das amostragens apresentadas, os colaboradores podem obter um parâmetro claro de sua performance, com gráficos que segregam o desempenho por natureza de tarefa (Frontend, Backend, Mobile, entre

outras) em cada sprint. A aplicação permite analisar detalhadamente a pontuação entregue em cada sprint, categorizada por tipo de tarefa, cliente e projeto, além de fornecer um panorama comparativo entre o tempo estimado e o tempo realizado das atividades. Isso permite aos colaboradores refletirem sobre os fatores que influenciaram eventuais desvios e identificarem oportunidades de melhoria.

Outro aspecto relevante na adoção diária da aplicação pelas equipes ágeis é a funcionalidade de visualização das tarefas das sprints em formato de lista, similar ao oferecido pela plataforma Jira, destacando prioridades, responsáveis, descrição e tecnologias envolvidas em cada demanda. Assim, os colaboradores, que já acessam a aplicação regularmente para acompanhar OKRs e métricas individuais, tendem a utilizá-la também para o monitoramento de tarefas e sprints, reduzindo a frequência de alternância entre diferentes ferramentas e aumentando a eficiência no fluxo de trabalho.

7.1.2 Equipe de Negócios

Na equipe de negócios, responsável pela definição de direcionamentos para a evolução comercial do produto e pelo contato com os clientes, destaca-se, entre as funcionalidades da aplicação desenvolvida, a visualização de métricas relacionadas às características dos projetos e tarefas, segregadas por cliente. Dessa forma, a equipe de negócios pode obter uma compreensão facilitada sobre o esforço consumido em cada projeto e por cada cliente, com base em faixas de datas específicas ou outros filtros a serem definidos.

É válido destacar a utilização da métrica customizável de "bugs por cliente", que ilustra quais clientes têm gerado um maior número de demandas relacionadas a problemas em ambientes produtivos ao longo de um período específico. Com isso, foi possível à equipe de negócios elaborar um plano de ação focado nos clientes que frequentemente lideram os indicadores de reportação de bugs. Esse plano inclui estratégias como a revisão da abordagem de atendimento, um melhor entendimento das regras de negócios, ajustes nas equipes ou nos desenvolvedores responsáveis pelas demandas, além de mentorias e treinamentos específicos. Essas ações visam capacitar a equipe, reduzindo, assim, a quantidade de bugs gerados e refletidos pela métrica.

Por meio da utilização da ferramenta, foi possível identificar, por exemplo, um cliente cujo número de bugs reportados permaneceu elevado ao longo do tempo, independentemente das diferentes abordagens e planos de ação adotados. Essa constatação, frequentemente observada por meio da ferramenta desenvolvida, indica que o esse cliente pode não justificar mais o investimento contínuo e significativo de esforços para sua manutenção na carteira de clientes pois o mesmo deixa de ser lucrativo para a organização.

Outro aspecto relevante da utilização da aplicação é o acesso frequente da equipe de negócios às informações sobre as tarefas e demandas em andamento nas sprints dos times, em tempo real. Esse acesso permite que a equipe forneça posicionamentos mais assertivos aos clientes. Anteriormente, era necessário um contato ativo com o desenvolvedor ou gestor responsável pela demanda para obter atualizações sobre o progresso, especialmente devido à

limitação no número de usuários que podem acessar a plataforma Jira. Como membros das equipes comercial e de negócios não possuem acesso à plataforma Jira, consequentemente não possuíam visibilidade direta das sprints de desenvolvimento. Portanto, a aplicação-alvo deste trabalho passou a suprir essa lacuna, proporcionando uma visão informatizada e acessível do andamento das atividades nos times a todos os interessados.

7.1.3 Equipe de Gestão

Na equipe de Gestão dos times de desenvolvimento, observou-se uma grande utilidade da aplicação, em parte devido ao fato de os gestores das equipes serem os principais responsáveis pela definição das métricas que a aplicação, alvo deste trabalho, deveria ou não incluir.

Dentre os aspectos que agregaram valor ao cotidiano da gestão, destaca-se, primeiramente, as métricas individuais relacionadas à performance e à natureza das atuações de cada membro do time. Por meio dessas amostragens, é possível acompanhar a evolução de cada colaborador e definir planos de ação para que determinados membros atinjam os objetivos esperados, tanto do ponto de vista de seus gestores quanto do próprio colaborador.

Além disso, as métricas individuais possibilitaram a criação de um parâmetro relativo para a efetivação de promoções com base na senioridade de um colaborador específico. Observou-se que, ao longo de diversas sprints, as características de atuação desse colaborador mantiveram um padrão consistente, evidenciado pela conclusão de tarefas em diferentes contextos (Frontend, Backend, DevOps, Mobile, Documentação, entre outras). Considerando que outros desenvolvedores de maior senioridade formal na organização apresentam características de métricas individuais semelhantes, esse padrão contribuiu para defesa de reconhecimento formal do aumento de senioridade do colaborador citado.

A centralização de acesso a informações sobre métricas ágeis, customizadas, individuais, além de dados sobre sprints, projetos e tarefas, foi também destacada como um ponto positivo na utilização cotidiana da ferramenta pela equipe de gestão. De maneira semelhante ao mencionado anteriormente no contexto das equipes de desenvolvimento, para a equipe de gestão a frequência de alternância entre diferentes ferramentas foi significativamente reduzida. No caso da gestão, esse impacto positivo é ainda mais relevante, pois, antes do desenvolvimento da aplicação abordada neste trabalho, os gestores precisavam recorrer a múltiplas planilhas para obter indicadores, medidas e métricas necessárias para orientar a gestão das equipes e apoiar a tomada de decisões.

Outro impacto positivo observado pela equipe de gestão foi o aumento do engajamento das equipes em relação ao cumprimento das metas pré-estabelecidas nas OKRs. A facilidade de visualização e o acompanhamento em tempo real dos objetivos têm reduzido a necessidade de ações frequentes de conscientização e reforço das metas por parte da gestão. Dessa forma, a aplicação passou a atuar como um motivador natural no dia a dia das equipes, incentivando o foco e o alinhamento com os objetivos estabelecidos, e influenciando diretamente o fator anímico dos colaboradores.

É válido ressaltar que a equipe de gestão frequentemente sugere novas métricas e melhorias para a aplicação, garantindo uma contribuição contínua para a evolução da ferramenta mencionada e alinhamento com os objetivos organizacionais.

7.2 CONSIDERAÇÕES FINAIS

Conforme os pontos analisados ao longo deste capítulo, a avaliação da aplicação denominada "Metric Insight", que disponibiliza métricas ágeis, customizadas e individuais de forma integrada à plataforma Jira por meio de consumo de dados em tempo real, pode ser considerada positiva.

A aplicação demonstrou resultados práticos e concretos em diferentes equipes da organização, mostrando-se útil no contexto de desenvolvimento de software em times ágeis e cumprindo com êxito o seu papel inicialmente estabelecido.

8 CONCLUSÃO

O presente trabalho aborda a análise e implementação de uma aplicação web destinada a facilitar o dia a dia das equipes ágeis de desenvolvimento de software e seu ecossistema, por meio da apresentação de métricas ágeis, customizadas e individuais relevantes para a organização, com base em dados consumidos em tempo real via integração por API com a plataforma Jira.

A ferramenta proposta também se compromete a fornecer informações atualizadas em tempo real sobre o progresso no cumprimento dos objetivos organizacionais relacionados ao desenvolvimento e à manutenção de software, monitorados por meio de OKRs.

Inicialmente, foi realizada a fundamentação teórica sobre os conceitos diretamente relacionados ao tema deste trabalho, com o objetivo de embasar conceitualmente, de acordo com a literatura da área, tópicos como Processo de Produção de Software, Avaliação de Processos de Software, Abordagens Ágeis (como Scrum e Kanban), Métricas de Software, OKR e a plataforma Jira.

Em seguida, foi conduzida uma análise do estado da arte sobre o tema do trabalho, utilizando a técnica de mapeamento sistemático da literatura. A partir dessa análise, foi possível complementar a proposta do presente estudo, identificando aspectos relevantes para o desenvolvimento da ferramenta proposta, baseando-se em lacunas encontradas na literatura. Além disso, buscou-se replicar elementos positivos de outras ferramentas identificadas durante a revisão sistemática, que automatizam partes do processo de medição e avaliação da produção de software.

Na sequência, foi realizada a etapa de análise detalhada para o desenvolvimento da aplicação proposta, com o objetivo de contextualizar o processo da organização em que a ferramenta será utilizada, incluindo os produtos desenvolvidos, a composição das equipes, as métricas relevantes para o contexto e os OKRs diretamente influenciados pelas equipes de desenvolvimento de software. Em seguida, foram definidos os requisitos funcionais e não funcionais da aplicação, estabelecendo a base para a fase de desenvolvimento do sistema proposto.

Durante a fase de desenvolvimento, a ferramenta proposta foi implementada, realizando a coleta automatizada de dados sobre produtos, equipes e tarefas para utilização nas métricas escolhidas, por meio de integração via API com a plataforma Jira. Paralelamente, foi elaborada a documentação técnica, abrangendo a arquitetura da aplicação, as configurações do ambiente de execução, as especificidades do consumo da API do Jira e a modelagem do banco de dados. Ainda nesta etapa, foi produzida a documentação não técnica, destinada a ilustrar a utilização da ferramenta proposta, com exemplos das telas, dos menus e das métricas implementadas na aplicação. Por fim, após a conclusão do desenvolvimento, a aplicação foi disponibilizada para utilização cotidiana pelas equipes da organização mencionada no trabalho.

Ao final, é apresentada a avaliação da aplicação proposta sob a perspectiva do autor. Os resultados descritos neste capítulo foram obtidos por meio da observação da utilização da ferramenta pelos colaboradores da organização, pertencentes às equipes de desenvolvimento,

negócios e gestão. A percepção dos resultados alcançados foi complementada por diálogos e interações não formais com os usuários da ferramenta.

Dessa forma, conclui-se que o propósito da aplicação proposta, que visa apresentar métricas ágeis significativas, customizadas e individuais, relevantes para o contexto das equipes ágeis de desenvolvimento de software da organização, por meio de integração automatizada com a plataforma Jira, foi plenamente alcançado. Os requisitos funcionais e não funcionais identificados na fase de análise do desenvolvimento foram atendidos. Ademais, entre as lacunas identificadas na revisão sistemática do estado da arte, relacionadas a aspectos não contemplados por outros trabalhos, aquelas selecionadas foram devidamente preenchidas. Os resultados observados demonstraram impactos positivos concretos e práticos em diversas equipes da organização, evidenciando a utilidade da aplicação no contexto do desenvolvimento de software em equipes ágeis.

8.1 LIMITAÇÕES

Segue, abaixo, a relação de tópicos identificados como limitações do trabalho realizado, os quais podem servir como base para futuras melhorias da ferramenta e para o desenvolvimento de trabalhos subsequentes:

- Restrição ao uso com ferramentas de gerenciamento de projetos compatíveis com o padrão de modelagem de dados da plataforma Jira.
- A aplicação concentra-se nos processos relevantes, na perspectiva da organização alvo, para a avaliação automatizada de métricas, não abrangindo integralmente todos os processos de produção de software possíveis.
- Limitação na replicação da experiência fluida proporcionada ao usuário ao acessar a ferramenta por meio de dispositivos móveis, em contraste com a experiência extremamente positiva observada ao utilizá-la em navegadores padrão através de computadores.
- Não foi conduzido um estudo de caso formal que incluísse a aplicação de questionários avaliativos para obtenção de *feedback* dos usuários.
- Dependência de ação manual de desenvolvedor para atualizar instância de banco de dados local.

8.2 TRABALHOS FUTUROS

Com base nas limitações citadas na seção anterior, recomendam-se os seguintes tópicos para exploração em trabalhos futuros:

- Implementar integração com outros sistemas além da plataforma Jira.
- Permitir o consumo de dados de organizações distintas da organização-alvo deste trabalho.
- Desenvolver uma extensão de navegador que facilite a visualização de gráficos relevantes, como o gráfico em tempo real do cumprimento das OKRs da organização.
- Aprimorar a responsividade da aplicação, visando uma melhor utilização em dispositivos móveis.
- Submeter a aplicação a um estudo de caso formal sobre sua utilização, conforme as diretrizes do Comitê de Ética em Pesquisa com Seres Humanos (CEPSH), elaborando questionários avaliativos e registrando *feedbacks* dos usuários.
- Implementar novas métricas ágeis, customizadas e individuais, além das já abordadas neste trabalho.

- Definir rotina automatizada de atualização da instância de banco de dados local da aplicação, de modo a não depender de gatilho disparado manualmente.

REFERÊNCIAS

ALBINO, R. Métricas Ágeis -. In: **Obtenha melhores resultados em sua equipe.** [S.l.: s.n.], 2017. ISBN 978-85-5519-276-0.

ANACLETO, A.; WANGENHEIM, C.; SALVIANO, C. Um método de avaliação de processos de software em micro e pequenas empresas. In: **Anais do IV Simpósio Brasileiro de Qualidade de Software**. Porto Alegre, RS, Brasil: SBC, 2005. p. 147–161. ISSN 0000-0000. Disponível em: https://sol.sbc.org.br/index.php/sbqs/article/view/16160.

ANDERSON, D.; CARMICHAEL, A. **Essential Kanban Condensed**. Lean-Kanban University, 2015. ([Essential Kanban]). ISBN 9780984521425. Disponível em: https://books.google.com.br/books?id=fHMCDQEACAAJ.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: http://www.agilemanifesto.org/.

BOEG, J. **Kanban em 10 passos**. C4Media, Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt, 2010. Disponível em: https://www.infoq.com/br/minibooks/priming-kanban-jesper-boeg/.

BOSSE, J. Métricas ágeis integradas à Plataforma Github - Trabalho de Conclusão de Curso do Departamento de Informática e Estatística - Universidade Federal de Santa Catarina, Florianópolis. 2019.

CASTRO, F. **Agile Goal Setting with OKR - Objectives and Key Results. InfoQ, Disponível em: https://www.infoq.com/articles/agile-goals-okr/. [Acesso em: 21 de Junho de 2023]**. [S.l.], 2015.

COHN, M. **User Stories Applied: For Agile Software Development**. Boston, MA, USA: Addison-Wesley Professional, 2004. ISBN 0321205685.

COHN, M. **Agile Estimating and Planning**. Pearson Education, 2005. (Robert C. Martin Series). ISBN 9780132703109. Disponível em: https://books.google.com.br/books?id=BuFWHffRJssC.

EMAM, e. a. Cost implication of interrater agreement for software process assessments. **Technical Report ISERN-98-14. Fraunhofer IESE.**, 1998.

FENTON, N.; PFLEEGER, S. **Software Metrics: A Rigorous and Practical Approach**. International Thomson Computer Press, 1997. (Computer Science Series). ISBN 9781850322757. Disponível em: https://books.google.com.br/books?id=tiEiAQAAIAAJ.

FISHER, J. M.; D.J.KONING; A.P.LUDWIGSEN. Utilizing atlassian jira for large-scale software development management*. In: . [s.n.], 2013. Disponível em: https://api.semanticscholar.org/CorpusID:114987096.

FOWLER, M. **Patterns of Enterprise Application Architecture**. [S.l.]: Addison-Wesley, 2003.

HARTMANN, e. a. Appropriate agile measurements:. In: **Using metrics and diagnostics to deliver business value.** [S.l.]: Agile 2006 Conference, 2006. p. 126–131.

HUMPHREY, W. S. **Managing the software process**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989.

IEEE; STD; 610.12. Ieee standard glossary of software engineering terminology. **IEEE Std 610.12-1990**, p. 1–84, 1990.

INTHURN, C. **Qualidade & teste de software: engenharia de software**. Visual Books, 2001. ISBN 9788575020265. Disponível em: https://books.google.com.br/books?id=E4pKAAAACAAJ.

ISO/IEC. **International Standard - Systems and software engineering — Software life cycle processes. 12207:2008**. [S.l.], 2008.

ISO/IEC. **Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models 25010:2011**. [S.l.], 2011.

ISO/IEE. **International Standard - Systems and software engineering Measurement process. 15939:2017(E)**. [S.l.], 2017.

JARVINEN, J. Measurement based continuous assessment of software engineering processes. **VTT PUBLICATIONS**, TECHNICAL RESEARCH CENTRE OF FINLAND ESPOO, v. 4, n. 2, p. 6, 2000.

LARMAN, C. Agile and iterative development. In: **Agile and Iterative Development: A Manager's Guid.** Boston, USA: Pearson Education, 2003. p. 25–28.

LUNA, A. et al. Uma abordagem para o gerenciamento estratégico Ágil em saúde utilizando pes, okr e mangve. **Revista Eletrônica da Estácio Recife**, v. 3, n. 2, 2017. Disponível em: https://reer.emnuvens.com.br/reer/article/view/146/47.

MARTIN, C. R. Desenvolvimento Ágil limpo. In: **Desenvolvimento Ágil Limpo - De volta às Origens**. New York, USA: ALTA BOOKS, 2020. p. 161–163. ISBN 978-85-508-1500-8.

MCGARRY, J. **Practical Software Measurement: Objective Information for Decision Makers**. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0201715163.

MPS-BR. **Softex - Melhoria de Processo do Software Brasileiro**. [S.l.], 2007.

NIVEN et al. Driving focus, alignment, and engagement with okrs. In: **Objectives and Key Results**. [S.l.]: John Wiley Sons, 2016. v. 1. ISBN 978-11-192-5239-9.

PEINADO, J. Compreendendo o kanban: um ensino interativo ilustrado. **Revista DaVinci. Curitiba–PR**, v. 4, n. 1, p. 133–146, 2007.

PRESSMAN, R. Software engineering. In: **A Practitioner's Approach**. New York, USA: McGraw Hill, 2010.

PRESSMAN, R. S. Engenharia de software. In: **Engenharia de Software - Uma abordagem profissional**. New York, USA: AMGH EDITORA LTDA, 2021. (PODS '89), p. 140–149. ISBN 978-65-5804-011-8.

RAO, K. N.; NAIDU, G. K.; CHAKKA, P. A study of the agile software development methods, applicability and implications in industry. **International Journal of Software Engineering and its applications**, v. 5, n. 2, p. 35–45, 2011.

RISING, L.; JANOFF, N. The scrum software development process for small teams. **IEEE Software**, v. 17, n. 4, p. 26–32, 2000.

ROSA, J. Desenvolvimento de um software para métricas em rastreabilidade de requisitos de software - Proposta de Trabalho de Diplomação - Universidade Tecnológica Federal do Paraná, Cornélio Procópio. 2011.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. Pearson, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780132071482. Disponível em: https://books.google.com.br/books?id=nj-wPwAACAAJ.

SATO, D. T. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. **Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo**, v. 139, 2007.

SBROCCO, J. D. C.; MACEDO, P. D. **Metodologias Ágeis: ENGENHARIA DE SOFTWARE SOB MEDIDA**. [s.n.], 2012. ISBN 9788536503981. Disponível em: https://books.google.com.br/books?id=GAkpLgEACAAJ.

SCHWABER, K.; BEEDLE, M. **Agile software development with Scrum**. [S.l.]: Prentice Hall PTR, 2002.

SOMMERVILLE, I. **Software Engineering**. 5. ed. USA: Addison Wesley Longman Publishing Co., Inc., 1995. ISBN 0201427656.

SOMMERVILLE, I. **Software Engineering**. 7. ed. Pearson/Addison-Wesley, 2004. (International computer science series). ISBN 9780321210265. Disponível em: https://books.google.com.br/books?id=fIJQAAAAMAAJ.

SOMMERVILLE, I. **Software Engineering**. 8. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston, MA, USA: Pearson, 2011. ISBN 9780137035151.

STANTON, A. [S.l.]: PIXAR, Andrew. 2003 – PROCURANDO NEMO.

SUTHERLAND, J. Scrum. In: **Scrum : The art of doing twice the work in half the time**. [S.l.]: LEYA, 2016. ISBN 978-85-441-0088-2.

VAZQUEZ, C.; SIMOES, G.; ALBERT, R. **Analise De Pontos De Funçao: MEDIÇAO,ESTIMATIVAS E GERENCIAMENTO DE PROJETOS DE**. ERICA. ISBN 9788571948990. Disponível em: https://books.google.com.br/books?id=THFUPgAACAAJ.

WAZLAWICK, R. Engenharia de software: conceitos e práticas. In: . [S.l.]: Elsevier Brasil, 2013. ISBN 978-85-3526-084-7.

.1 APÊNDICE A - ARTIGO

Implementação de Métricas Ágeis integradas à Plataforma Jira

Wesley Mayk Gama Luz[1]

[1]Instituto de Informática e Estatística – Universidade Federal do Santa Catarina (UFSC)
Florianópolis – SC – Brasil

w_mayk007@hotmail.com

Abstract. *The expanding software production market has increased the demand for agile methodologies and metrics to improve software quality and evaluate results. Platforms like Jira are widely used to manage processes, teams, and tasks efficiently, especially with SCRUM and Kanban. This work proposes a tool integrated with Jira to automate project data collection, establish agile metrics, monitor progress in real time, identify bottlenecks, and enhance team performance in agile environments.*

Resumo. *O mercado de produção de software tem crescido junto à demanda por metodologias que aprimorem a qualidade e a avaliação de resultados, tornando métricas ágeis essenciais. Para otimizar a gestão de processos e tarefas, plataformas como o Jira são amplamente adotadas por sua eficiência no gerenciamento de projetos com Scrum e Kanban. Este trabalho propõe uma ferramenta integrada ao Jira para automatizar a coleta de dados, estabelecer métricas ágeis, monitorar objetivos em tempo real, identificar gargalos e contribuir para melhorias em equipes ágeis.*

1. Introdução

O mercado de software começou a se expandir na década de 1960 e, com o advento dos microcomputadores nos anos 1970, a demanda por produção de software cresceu substancialmente. Isso levou à criação da Engenharia de Software, uma tentativa de solucionar a crise de software marcada por baixa confiabilidade e desempenho questionável. Com o passar dos anos, a Engenharia de Software adotou um modelo mais sistemático e controlado, inspirado nos processos da engenharia tradicional, e focado na avaliação constante e na melhoria da qualidade do produto. A medição de software tornou-se fundamental para identificar pontos fortes e fracos nos processos e apoiar a melhoria contínua, especialmente à medida que as metodologias ágeis começaram a ser adotadas nos anos 1990.

As metodologias ágeis, como Scrum e Kanban, trouxeram uma nova abordagem ao desenvolvimento de software, priorizando flexibilidade, adaptação rápida às mudanças e eficiência. O Scrum, por exemplo, divide o trabalho em ciclos curtos chamados Sprints, permitindo que as equipes se concentrem em entregas incrementais e monitorando constantemente o progresso do projeto. A medição de métricas ágeis tornou-se essencial para tomar decisões informadas sobre o andamento dos projetos e identificar melhorias. Além disso, métodos como o Kanban ajudaram as equipes a gerenciar mudanças e visualizar o trabalho em andamento de maneira mais eficiente. No entanto, mesmo com o uso dessas ferramentas, muitas equipes de desenvolvimento

enfrentam desafios ao tentar obter métricas claras e acessíveis para aprimorar seus processos e melhorar a produtividade.

Este trabalho propõe uma solução para ajudar na gestão e tomada de decisões em equipes ágeis que utilizam a plataforma Jira, uma das ferramentas mais populares para gerenciamento de projetos. A proposta é o desenvolvimento de uma ferramenta integrada à API do Jira que automatiza a coleta e o processamento de dados dos projetos. Com isso, a ferramenta visa gerar métricas e indicadores significativos de forma intuitiva e acessível, permitindo que equipes e gestores identifiquem gargalos, melhorem estimativas e otimizem seus processos de desenvolvimento. Ao integrar dados de tarefas, subtarefas, sprints e projetos, a ferramenta proporcionará uma visão mais clara e objetiva do desempenho, auxiliando na melhoria contínua e na tomada de decisões estratégicas.

2. Fundamentação Teórica

O Scrum é uma metodologia ágil amplamente aplicada em equipes de desenvolvimento de software, que divide o trabalho em ciclos curtos chamados Sprints. Os benefícios dessa abordagem incluem a entrega contínua de incrementos, aumento da visibilidade do progresso, e melhoria na comunicação da equipe. Os clientes recebem feedback frequente, promovendo um relacionamento de confiança, enquanto a equipe tem maior controle sobre o trabalho e a adaptação às mudanças nos requisitos. Esse modelo favorece o desenvolvimento colaborativo e promove uma cultura de sucesso dentro do projeto.

As métricas de software são essenciais para a mensuração de atributos de produtos e processos, com o objetivo de melhorar a qualidade do desenvolvimento. Existem diferentes tipos de métricas, incluindo diretas, que medem dados facilmente acessíveis, como custo e defeitos, e indiretas, que avaliam aspectos mais complexos, como qualidade e manutenibilidade. A aplicação de métricas ajuda a gerenciar o fluxo de trabalho, reduzir custos e otimizar a produtividade. Dentro das metodologias ágeis, as métricas devem ser simples, facilmente coletadas e fornecer feedback útil para a melhoria do processo.

O conceito de OKRs (Objectives and Key Results) é uma abordagem popular para definição de metas, visando garantir alinhamento estratégico e medição de resultados em organizações. OKRs são baseados em objetivos claros, com indicadores-chave para avaliar o progresso. O ciclo de definição e avaliação de OKRs permite ajustes rápidos e um foco contínuo no desempenho. Empresas como Google e Intel adotaram com sucesso esse modelo para impulsionar sua evolução organizacional, promovendo uma cultura de alta performance e inovação.

Por fim, a plataforma Jira, desenvolvida pela Atlassian, oferece um conjunto robusto de ferramentas para o gerenciamento de projetos de software. Através da Jira API, é possível integrar e automatizar processos com outras ferramentas, facilitando o acesso e a manipulação de dados em tempo real. O uso da API do Jira permite que as equipes ágeis monitorarem o progresso do projeto e coletem dados de maneira

automatizada, além de garantir segurança no acesso através de autenticação e autorização adequadas.

3. Estado da Arte

Após a leitura de determinados artigos selecionados que retratam o estado da arte referente ao tema de estudo, foi possível identificar as informações mais relevantes para complementar a proposta deste trabalho, com foco na avaliação de processos de produção de software.

Embora as ferramentas e modelos propostos nas pesquisas mostrem a importância de automatizar e avaliar os processos ágeis, muitos deles não foram testados em organizações reais, o que limita sua aplicabilidade prática. Além disso, várias ferramentas dependem de modelos específicos, como Scrum e CMMI, dificultando sua adaptação a diferentes contextos organizacionais. O estudo também revela a necessidade de uma visão integrada, já que muitas soluções tratam aspectos isolados do processo ágil, sem considerar a totalidade do ciclo de desenvolvimento.

A conclusão desta revisão sistemática, pelo qual foi analisado o estado da arte, aponta para as lacunas identificadas nos estudos revisados, como a falta de aplicação prática em ambientes reais e a limitação em termos de integração com ferramentas amplamente utilizadas no mercado. No entanto, a proposta deste trabalho visa superar essas limitações ao oferecer uma solução prática que se integra ao Jira, uma ferramenta consolidada no mercado. A ferramenta proposta também oferece funcionalidades de avaliação automatizada de métricas, com atualização em tempo real, adaptando-se a diferentes metodologias e contextos organizacionais, sem depender de modelos específicos como o Scrum ou o CMMI.

4. Proposta do Trabalho

Este Trabalho de Conclusão de Curso propõe a implementação de uma aplicação web para geração de métricas ágeis e customizadas, integrada ao Jira via Jira API. A ferramenta utiliza um banco de dados de projetos acumulados, visando transformar dados em informações úteis para análise e melhoria de processos nas equipes da organização.

A aplicação monitora em tempo real o progresso de demandas no Jira, com foco no cumprimento de OKRs, identificando gargalos, oportunidades de melhoria no fluxo de trabalho e oferecendo novas perspectivas sobre os times e projetos. O objetivo é apoiar decisões operacionais e estratégicas, melhorando as equipes ágeis e a gestão organizacional.

O desenvolvimento da aplicação aborda lacunas encontradas em outras soluções, como a falta de aplicação prática em ambientes reais e a ausência de integração com ferramentas amplamente usadas. A solução foi criada para se integrar ao Jira e oferecer

avaliação automatizada de métricas, com flexibilidade para adaptar-se a diferentes metodologias e contextos organizacionais.

Após a implementação, a ferramenta foi testada em uma organização de mercado, permitindo analisar os impactos observados, como melhorias nas rotinas operacionais das equipes, identificação de oportunidades para otimização de processos e suporte fortalecido às decisões estratégicas.

5. Aspectos da fase de Análise

Este capítulo visa discorrer brevemente acerca do processo, produto e equipes da organização que serviu como base para desenvolvimento da aplicação proposta neste trabalho. Contempla também a análise das métricas e OKRs que fazem sentido no cotidiano das equips ágeis descritas e que, portanto, agregam valor ao sistema, além de apresentar também a análise de requisitos funcionais e não funcionais do desenvolvimento.

A organização foco deste trabalho é especializada em tecnologia inovadora para o mercado financeiro, com produtos de software em áreas como previdência complementar, consórcios, bancos, inteligência de negócios e meios de pagamento. Com filiais em diversos estados do Brasil, a empresa tem apresentado um crescimento significativo em participação de mercado. Este estudo analisou o cotidiano de três equipes de desenvolvimento da vertical de previdência privada, utilizando dados de produção de software dessas equipes de forma anonimizada, a fim de construir métricas e indicadores significativos. O autor faz parte de uma dessas equipes e a aplicação desenvolvida busca aprimorar o trabalho diário das equipes por meio da adoção da solução proposta.

As três equipes de desenvolvimento de software, compostas por cinco membros cada, totalizando 15 colaboradores, incluem desenvolvedores e validadores com diferentes níveis de experiência. Desenvolvedores mais seniores também assumem responsabilidades relacionadas à infraestrutura e DevOps. As tarefas são identificadas por tags que indicam sua natureza, como Front-end, Back-end, Mobile, Infra, DevOps, Bug e Teste, sendo que as tarefas "Bug" requerem soluções rápidas devido à sua relação com o ambiente produtivo. Cada equipe segue uma sprint de desenvolvimento de até 2 semanas, com planejamento e revisões periódicas de sprints realizadas com o apoio das equipes de gestão e líderes.

A organização em estudo adota a metodologia OKR (Objectives and Key Results) para definir e monitorar objetivos estratégicos que impactam diretamente os resultados do negócio. Atingir esses objetivos resulta em recompensas trimestrais, como promoções, ajustes salariais, premiações e aumento na participação nos lucros. No contexto das equipes de desenvolvimento, foram mapeadas três OKRs específicas para proporcionar um acompanhamento mais acessível e constante das metas, com o objetivo de monitorá-las durante as sprints, e não apenas ao final de cada trimestre.

As OKRs selecionadas incluem: "Total Sprint Deliveries", que visa garantir a conclusão média de 80% das tarefas planejadas a cada sprint; "Bug Resolution Time",

que busca corrigir 75% dos bugs reportados em até três dias após sua identificação; e "Bug Tracker for Completed Projects", que monitora o percentual de projetos finalizados que geram bugs em produção, com a meta de limitar esse número a 20%. Essas metas têm como objetivo melhorar a consistência nas entregas, otimizar a resolução de problemas e reduzir o número de bugs em projetos finalizados.

Ainda na fase de análise, foi realizado um levantamento para selecionar as métricas ágeis mais adequadas ao contexto das equipes de desenvolvimento, considerando as amplamente difundidas na literatura e aquelas que já agregam valor ao monitoramento e análise de produção. Essas métricas foram classificadas como "Métricas Ágeis" na aplicação em desenvolvimento, com o objetivo de facilitar o acesso aos indicadores essenciais para o acompanhamento das atividades das equipes.

Além das métricas ágeis, também foram propostas métricas customizadas, tanto coletivas quanto individuais, definidas a partir de aspectos específicos do negócio e da produção de software que não são abordados pelas métricas convencionais. Essas métricas, denominadas "Métricas Customizadas", foram elaboradas para oferecer análises direcionadas aos dados organizacionais, com o objetivo de apoiar a tomada de decisões gerenciais e estratégicas, promovendo a melhoria contínua e o crescimento da organização.

Entre as métricas ágeis selecionadas para implementação na aplicação estão o Burndown Chart, Burnup Chart, Work in Progress (WIP) e Velocity. A escolha dessas métricas levou em consideração a opinião dos gestores das equipes, a frequência com que são utilizadas pelos membros das equipes e a viabilidade de implementação, ponderando o esforço técnico necessário em relação à frequência de uso no ambiente organizacional.

Dentre as métricas customizadas implementadas, destacam-se duas que agregam valor significativo à organização: Moving Average Velocity e Bugs Per Client. A Moving Average Velocity oferece uma análise contínua da capacidade de entrega de uma equipe ao longo de vários sprints, utilizando o histórico de sprints anteriores para calcular uma média do trabalho concluído e o previsto para os próximos períodos. Essa métrica é útil para suavizar variações entre sprints, permitindo uma previsão mais precisa sobre o ritmo de desenvolvimento e facilitando o planejamento das entregas futuras.

Além de ajudar na previsão de produtividade, a média móvel também indica oscilações no desempenho da equipe. Quando a entrega é estável, a curva é suave, enquanto variações mais acentuadas revelam sprints de maior ou menor produtividade. Essa visibilidade facilita a identificação de fatores que impactam o desempenho, permitindo ajustes no planejamento e intervenções que ajudem a manter um fluxo de trabalho mais eficiente e previsível.

Neste capítulo, foram abordados os processos, produtos e equipes da organização de referência para o desenvolvimento da aplicação, além das métricas e OKRs relevantes para as equipes ágeis, que agregam valor ao sistema. Também foram analisados os requisitos funcionais e não funcionais necessários para o desenvolvimento, estabelecendo os insumos iniciais e as diretrizes essenciais para alinhar os objetivos do

sistema de maneira estruturada e dar continuidade à fase de desenvolvimento da aplicação proposta.

6. Aspectos de Desenvolvimento

Este capítulo descreve a arquitetura da ferramenta, as tecnologias utilizadas, decisões de projeto, configuração do ambiente, automatização de consumo de dados do Jira e informações sobre o banco de dados. Ao final, é apresentada uma documentação de navegação da ferramenta, detalhando telas, menus e as métricas implementadas, com o objetivo de facilitar sua utilização pelos usuários.

A aplicação desenvolvida é uma ferramenta web com tecnologias distintas para o front-end e back-end. O back-end foi construído utilizando Dart, uma linguagem de programação da Google, focada em desempenho e eficiência para sistemas modernos, enquanto o front-end usa Javascript com o framework React, desenvolvido pelo Facebook, para interfaces de usuário interativas e eficientes. Dart se destaca por sua alta performance, tipagem estática e suporte a programação assíncrona, enquanto React facilita a criação de interfaces escaláveis e rápidas.

O sistema é containerizado utilizando Docker, o que permite a execução do front-end e do back-end de forma isolada em ambientes independentes, garantindo a consistência e facilitando a escalabilidade. Docker proporciona uma virtualização eficiente, oferecendo consistência no ciclo de vida do software e uso otimizado de recursos, o que melhora a comunicação entre as partes da aplicação.

O padrão de projeto adotado para a arquitetura da aplicação é o MVC (Model, View, Controller). Esse padrão facilita a organização do código, promovendo uma separação clara de responsabilidades entre dados (Model), interface de usuário (View) e lógica de controle (Controller). O uso do MVC melhora a manutenção, escalabilidade, testabilidade e reutilização do código, permitindo maior flexibilidade no desenvolvimento da aplicação.

Em resumo, a combinação de Dart no back-end, React no front-end, e Docker para a containerização, juntamente com o padrão MVC, foi escolhida para garantir um sistema robusto, escalável e de fácil manutenção. Essas tecnologias e práticas permitem que a aplicação seja eficiente, flexível e capaz de atender às necessidades de integração e monitoramento em tempo real das métricas ágeis e customizadas propostas no estudo.

Para garantir a consistência e disponibilidade dos dados, foi implementada uma estratégia que envolve manter uma instância local de banco de dados com uma amostragem dos dados essenciais relacionados a equipes, sprints, projetos e tarefas. Isso foi feito para mitigar riscos de falhas ou instabilidade na API do Jira, que é a fonte primária de dados. A instância local é atualizada manualmente por um desenvolvedor autorizado e, em caso de indisponibilidade da API, ela assume a responsabilidade de responder às requisições, garantindo que as métricas e indicadores de produtividade das equipes possam ser acessados mesmo em momentos de falha da API. Essa estratégia

também permite melhorar a performance, pois consultas recorrentes são feitas diretamente no banco local atualizado.

Além disso, para proteger dados sensíveis e atender à Lei Geral de Proteção de Dados (LGPD), foi adotada a prática de anonimizar informações confidenciais, como nomes de clientes e dados de colaboradores. A plataforma Jira mantém uma versão anonimizada dos dados sensíveis, garantindo que as informações necessárias para análise de métricas sejam preservadas sem comprometer a segurança dos dados. Assim, o banco de dados local também reflete essa política de anonimização, permitindo que as métricas e indicadores de produtividade sejam calculados de forma precisa, sem expor dados confidenciais da organização.

7. Avaliação

Este capítulo avalia os resultados da implementação da aplicação desenvolvida, destacando os impactos observados na organização. O objetivo é relatar as experiências do autor com a utilização da ferramenta por diferentes perfis de usuários, incluindo equipes de desenvolvimento ágil, gestores e membros da equipe comercial. A análise abrange os efeitos das funcionalidades implementadas, considerando como essas ferramentas têm sido aplicadas no dia a dia da organização.

A partir da análise do uso das ferramentas e métricas, foram identificadas melhorias significativas nos processos internos da organização, especialmente em eficiência, visibilidade e tomada de decisão. A aplicação demonstrou ser eficaz na otimização da visão geral de tarefas e sprints, no acompanhamento de projetos e na facilitação da comunicação entre as áreas. Esses benefícios contribuíram para ganhos de produtividade, melhoria na qualidade do atendimento ao cliente e decisões mais assertivas. A análise foi detalhada e organizada por equipe, evidenciando os ganhos específicos em cada contexto.

7.1. Equipes ágeis

Nas equipes ágeis, a aplicação foi amplamente acessada para monitorar as OKRs (Objectives and Key Results), com uma atualização em tempo real das métricas que se tornaram um atrativo significativo para os colaboradores. Cada sprint concluída ou demanda resolvida impactava diretamente os resultados das metas, e como o atingimento dessas metas influenciava os ganhos financeiros, como a participação nos lucros, a visualização das métricas em tempo real se tornou um fator motivacional importante. As equipes passaram a se engajar mais ativamente para alcançar as metas e acompanhar as mudanças nas métricas.

A funcionalidade de exibir métricas e indicadores de desempenho individuais também gerou grande interesse, permitindo aos colaboradores analisar detalhadamente seu desempenho, segmentado por tipo de tarefa, cliente e projeto. Isso proporcionou uma visão clara sobre a pontuação entregue em cada sprint, comparando tempo

estimado e realizado. Além disso, a aplicação ofereceu uma visualização das tarefas das sprints em formato de lista, facilitando o acompanhamento de prioridades e responsáveis. Com isso, as equipes passaram a utilizar a ferramenta com mais frequência, melhorando a eficiência ao reduzir a alternância entre diversas plataformas e consolidando o monitoramento das tarefas e OKRs em um único local.

7.2. Equipe de Negócios

Na equipe de negócios, responsável pelo desenvolvimento comercial do produto e pelo contato com os clientes, a aplicação foi essencial para visualização de métricas relacionadas aos projetos e tarefas, especialmente com a segmentação por cliente. Isso permitiu que a equipe compreendesse melhor o esforço envolvido em cada projeto, com filtros por datas e outras variáveis. A métrica customizável de "bugs por cliente" destacou clientes com maior número de problemas, facilitando a criação de planos de ação focados na revisão de abordagens de atendimento, ajustes nas equipes e treinamentos específicos para reduzir os erros.

A utilização dessa ferramenta também ajudou a identificar clientes cujos números de bugs reportados eram consistentemente elevados, sugerindo que o investimento contínuo na manutenção desses clientes poderia não ser mais justificável, devido à baixa lucratividade gerada. Essa análise permitiu à equipe de negócios tomar decisões mais estratégicas sobre a viabilidade de continuar investindo tempo e recursos em certos clientes, com base na análise detalhada de suas métricas.

Outro benefício da aplicação foi o acesso em tempo real às informações sobre as tarefas e demandas em andamento nas sprints das equipes de desenvolvimento. Anteriormente, a equipe de negócios precisava de contato ativo com desenvolvedores ou gestores para obter atualizações sobre o progresso, o que era limitado devido à falta de acesso à plataforma Jira. Com a nova aplicação, a equipe comercial e de negócios passou a ter visibilidade direta sobre o andamento das atividades, permitindo-lhes fornecer posicionamentos mais rápidos e precisos aos clientes, sem depender de intermediários.

7.3. Equipes de Gestão

Na equipe de Gestão dos times de desenvolvimento, a aplicação foi altamente valorizada, especialmente porque os gestores eram os principais responsáveis por definir as métricas a serem utilizadas. As métricas individuais relacionadas à performance de cada membro do time permitiram o acompanhamento detalhado da evolução dos colaboradores, o que facilitou a criação de planos de ação para que os objetivos fossem atingidos. Além disso, essas métricas ajudaram na definição de parâmetros para promoções com base na senioridade, como evidenciado pelo desempenho consistente de certos colaboradores em diversas áreas, o que contribuiu para a defesa do aumento de senioridade de alguns membros.

A centralização das informações sobre métricas ágeis, dados sobre sprints, projetos e tarefas, também foi um ponto positivo significativo para a equipe de gestão, já que antes os gestores precisavam recorrer a múltiplas planilhas para obter as métricas necessárias. Com a aplicação, a alternância entre diferentes ferramentas foi reduzida, o que otimizou o processo de tomada de decisões e orientações para as equipes. Isso aumentou a eficiência e a rapidez na gestão das equipes de desenvolvimento, permitindo aos gestores se concentrarem mais no acompanhamento e menos na busca por dados dispersos.

Outro impacto positivo foi o aumento do engajamento das equipes com as metas estabelecidas nas OKRs, devido à facilidade de visualização e acompanhamento em tempo real dos objetivos. A aplicação passou a atuar como um motivador natural, incentivando o foco nas metas e promovendo o alinhamento com os objetivos organizacionais. A equipe de gestão também contribuiu ativamente para a evolução contínua da ferramenta, sugerindo novas métricas e melhorias, o que garantiu o alinhamento contínuo da aplicação com as necessidades da organização e seus objetivos estratégicos.

8. Conclusão

O presente trabalho analisa a implementação de uma aplicação web projetada para facilitar o dia a dia das equipes ágeis de desenvolvimento de software, focando na apresentação de métricas ágeis personalizadas e individuais, com base em dados coletados em tempo real da plataforma Jira. A ferramenta proposta visa proporcionar informações atualizadas sobre o progresso das equipes, monitorando o cumprimento de objetivos organizacionais por meio de OKRs (Objectives and Key Results). A aplicação tem como principal objetivo melhorar o acompanhamento das equipes e fornecer dados precisos para apoiar a tomada de decisões, contribuindo diretamente para o desempenho organizacional.

A pesquisa iniciou com uma fundamentação teórica abrangente, abordando temas como o Processo de Produção de Software, Avaliação de Processos, Abordagens Ágeis (como Scrum e Kanban), Métricas de Software, OKRs e a plataforma Jira. Esse embasamento teórico foi crucial para entender o cenário de desenvolvimento ágil e as melhores práticas para a implementação de uma solução que atendesse às necessidades específicas das equipes de software. A revisão da literatura, realizada através de um mapeamento sistemático, ajudou a identificar lacunas existentes e elementos positivos de outras ferramentas que automatizam a medição e avaliação da produção de software, os quais foram incorporados no design da aplicação.

Após a análise teórica, a pesquisa seguiu para a etapa de análise detalhada do contexto da organização em que a ferramenta seria implementada, compreendendo seus produtos, equipes e métricas relevantes para o desenvolvimento ágil de software. A partir disso, foram definidos os requisitos funcionais e não funcionais da aplicação, que formaram a base para o desenvolvimento do sistema. Este processo garantiu que a ferramenta fosse adequada ao ambiente organizacional e que as métricas a serem

utilizadas refletissem as necessidades das equipes e os objetivos estratégicos da empresa.

A fase de desenvolvimento da aplicação consistiu na implementação da coleta automatizada de dados sobre produtos, equipes e tarefas, por meio da integração com a API do Jira. Durante esse processo, foi elaborada a documentação técnica, que detalhou a arquitetura da aplicação, as configurações do ambiente de execução e a modelagem do banco de dados. Além disso, foi produzida uma documentação não técnica, voltada para os usuários finais, com exemplos de telas, menus e métricas implementadas. Após a conclusão do desenvolvimento, a aplicação foi disponibilizada para uso pelas equipes da organização.

A avaliação da aplicação foi realizada a partir da observação do uso da ferramenta pelas equipes de desenvolvimento, negócios e gestão, além de interações informais com os usuários. Os resultados observados indicaram impactos positivos em várias equipes da organização, evidenciando a eficácia da aplicação na melhoria do acompanhamento de tarefas e no alcance das metas estabelecidas. A ferramenta permitiu uma maior visibilidade das métricas, facilitando o monitoramento das OKRs e contribuindo para um ambiente de trabalho mais eficiente e alinhado.

Conclui-se que a aplicação alcançou seus objetivos, proporcionando métricas ágeis significativas, customizadas e individuais, relevantes para as equipes de desenvolvimento de software. A ferramenta atendeu aos requisitos definidos e preencheu as lacunas encontradas na revisão da literatura. Os impactos práticos observados nas equipes destacam a utilidade da aplicação, que demonstrou ser um recurso valioso para a gestão de desempenho em ambientes de desenvolvimento ágil, contribuindo para o alcance de objetivos organizacionais e melhoria contínua nas operações de software.

References

RISING, L.; JANOFF, N. The scrum software development process for small teams. IEEE Software, v. 17, n. 4, p. 26–32, 2000.

ROSA, J. Desenvolvimento de um software para métricas em rastreabilidade de requisitos de software - Proposta de Trabalho de Diplomação - Universidade Tecnológica Federal do Paraná, Cornélio Procópio. 2011.

SATO, D. T. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, v. 139, 2007.

SCHWABER, K.; BEEDLE, M. Agile software development with Scrum. [S.l.]: Prentice Hall PTR, 2002.

SOMMERVILLE, I. Software Engineering. 5. ed. USA: Addison Wesley Longman Publishing Co., Inc., 1995. ISBN 0201427656.

SOMMERVILLE, I. Software Engineering. 8. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1.

SOMMERVILLE, I. Software Engineering. 9. ed. Boston, MA, USA: Pearson, 2011. ISBN 9780137035151.

SUTHERLAND, J. Scrum. In: Scrum : The art of doing twice the work in half the time. [S.l.]: LEYA, 2016. ISBN 978-85-441-0088-2.

VAZQUEZ, C.; SIMOES, G.; ALBERT, R. Analise De Pontos De Funçao: MEDIÇAO, ESTIMATIVAS E GERENCIAMENTO DE PROJETOS DE. ERICA. ISBN 9788571948990. Disponível em: https://books.google.com.br/books?id=THFUPgAACAAJ.

WAZLAWICK, R. Engenharia de software: conceitos e práticas. In: . [S.l.]: Elsevier Brasil, 2013. ISBN 978-85-3526-084-7.

.2 APÊNDICE B - CÓDIGO FONTE

Selected files

86 printable files

backend\bin\backend.dart
backend\lib\data\actions\login.dart
backend\lib\data\db\models\employee\employee.dart
backend\lib\data\db\connector.dart
backend\lib\external\actions\clients.dart
backend\lib\external\actions\employee.dart
backend\lib\external\actions\login.dart
backend\lib\external\actions\projects.dart
backend\lib\external\actions\sprints.dart
backend\lib\external\actions\teams.dart
backend\lib\external\handlers\middlewares\authorize.dart
backend\lib\external\handlers\clients.dart
backend\lib\external\handlers\employee_handlers.dart
backend\lib\external\handlers\login.dart
backend\lib\external\handlers\projects.dart
backend\lib\external\handlers\sprints.dart
backend\lib\external\handlers\team_handler.dart
backend\lib\external\routers\router.dart
backend\lib\backend_config.dart
frontend\src\components\metrics\charts\burndown_chart.js
frontend\src\components\metrics\charts\burnup_chart.js
frontend\src\components\metrics\charts\date_formatters.js
frontend\src\components\metrics\charts\utils.js
frontend\src\components\metrics\charts\velocity.js
frontend\src\components\metrics\charts\work_in_progress_chart.js
frontend\src\components\metrics\customized\charts\moving_average_velocity_chart.js
frontend\src\components\shared\drawer_aware\drawer_aware.js
frontend\src\components\shared\hooks\use_local_storage.js
frontend\src\components\shared\tags\mm_base_tag.js
frontend\src\components\shared\tags\priority_tags.js
frontend\src\components\shared\tags\project_tags.js
frontend\src\components\shared\tags\sprint_status.js
frontend\src\components\shared\tags\task_status.js
frontend\src\components\shared\tags\task_tags.js
frontend\src\components\shared\tags\team_tag.js
frontend\src\components\shared\mm_circular_progress.js
frontend\src\components\shared\mm_date_range_picker.js
frontend\src\components\shared\mm_paper.js
frontend\src\components\shared\mm_select.js
frontend\src\components\shared\simple_lable.js
frontend\src\components\sprint_card\index.js
frontend\src\pages\home\about\about_page.js
frontend\src\pages\home\account\account_page.js
frontend\src\pages\home\controller\router\protected_router.js
frontend\src\pages\home\controller\home_controller.js
frontend\src\pages\home\employee\charts\participation_pie_chart.js
frontend\src\pages\home\employee\charts\performance_chart.js
frontend\src\pages\home\employee\components\select.js
frontend\src\pages\home\employee\employee_full_info_page.js
frontend\src\pages\home\employee\employee_list_page.js
frontend\src\pages\home\metrics\bpc\bugs_per_client.js
frontend\src\pages\home\metrics\burndown\burndown_page.js
frontend\src\pages\home\metrics\burndown\burnup_page.js
frontend\src\pages\home\metrics\moving_average_velocity\moving_average_velocity_page.js
frontend\src\pages\home\metrics\velocity\velocity_page.js
frontend\src\pages\home\metrics\wip\wip_page.js
frontend\src\pages\home\metrics\metric_base_many_sprints.js
frontend\src\pages\home\metrics\metric_base_single_sprint.js
frontend\src\pages\home\project\project_card.js
frontend\src\pages\home\project\projects.js
frontend\src\pages\home\project\task_list_page.js
frontend\src\pages\home\sprints\metrics\agile\agile_metrics.js
frontend\src\pages\home\sprints\metrics\components\metric_card.js
frontend\src\pages\home\sprints\metrics\costume\costume_metrics.js
frontend\src\pages\home\sprints\metrics\okr\ok_metric_base.js
frontend\src\pages\home\sprints\metrics\okr\okr_metrics.js
frontend\src\pages\home\sprints\metrics\types\metrics_card_base.js
frontend\src\pages\home\sprints\metrics\metrics_page.js
frontend\src\pages\home\sprints\search\utils.js
frontend\src\pages\home\sprints\tasks\tasks_list\task_accordion\employee_info_popover.js
frontend\src\pages\home\sprints\tasks\tasks_list\task_accordion\task_accordion.js
frontend\src\pages\home\sprints\tasks\tasks_list\tasks_list.js
frontend\src\pages\home\sprints\index.js
frontend\src\pages\home\teams\team_card.js
frontend\src\pages\home\teams\teams_page.js
frontend\src\pages\home\filter_area.js
frontend\src\pages\login\login_page.js
frontend\src\state\providers\auth_provider.js
frontend\src\App.css
frontend\src\App.js
frontend\src\App.test.js
frontend\src\home_drawer.js
frontend\src\home.js
frontend\src\index.css
frontend\src\index.js
docker-compose.yml

backend\bin\backend.dart

```dart
import 'dart:io';

import 'package:backend/external/db/init_connection.dart';
import 'package:backend/external/routers/router.dart';
import 'package:shelf/shelf.dart';
import 'package:shelf/shelf_io.dart';
import 'package:shelf_cors_headers/shelf_cors_headers.dart';
```

```dart


void main(List<String> arguments) async {
    initConnection();

    final ip = InternetAddress.anyIPv4;

  final overrideHeaders = {
    ACCESS_CONTROL_ALLOW_ORIGIN: 'http://localhost:3000',
    'Content-Type': 'application/json;charset=utf-8'
  };

    // Configure a pipeline that logs requests.
    final handler = Pipeline().addMiddleware(corsHeaders(headers: overrideHeaders)).addMiddleware(logRequests()).addHandler(globalRouter);

    // For running in containers, we respect the PORT environment variable.
    final port = int.parse(Platform.environment['PORT'] ?? '8080');
    final server = await serve(handler, ip, port);
    print('Server listening on port ${server.port}');
}

```

backend\lib\data\actions\login.dart

```dart
import 'package:backend/data/db/models/employee/employee.dart';

abstract class AccessToken<T> {
  T get token;
  List<String> get permissions;
}

class LoginResult<T> {

  LoginResult({required this.employee, required this.token});

  final AccessToken<T> token;
  final Employee employee;
}

class ActionResult<Data, Error> {




  ActionResult.success(Data data): _data = data, _error = null, success = true;
  ActionResult.error(Error error): _data = null, _error = error, success = false;

  final Data? _data;
  final Error? _error;
  final bool success;

  Data? get data => _data;
  Error? get error => _error;
}

typedef LoginAction = Future<ActionResult<LoginResult, String>> Function(String email, String password);
```

backend\lib\data\db\models\employee\employee.dart

```dart


class Employee {

  Employee({required String name, required String email, required String position}) {
    _name = name;
    _email = email;
    _position = position;
  }

  late final String _name;
  late final String _email;
  late final String _position;

  String get name => _name;
  String get email => _email;
  String get position => _position;

  Map get json => {
    'name': name,
    'email': email,
    'position': position,
  };
}
```

backend\lib\data\db\connector.dart

```dart
class Endpoint {

  Endpoint({
    required this.host,
    required this.database,
    required this.username,
    required this.password,
    required this.port
  });

  final String host;
  final String database;
  final String username;
  final String password;
  final int port;
}

abstract class QueryResult {}

// class QueryBuilder {
//   String sql = '';

//   QueryBuilder select();
// }

class QueryBuilder {
  String sql = '';

  QueryBuilder select(List<String> tableFields) {
    sql += 'SELECT ${tableFields.join(',')}';
    return this;
  }

  QueryBuilder from(String tableName) {
    sql += 'FROM $tableName';
    return this;
  }

  QueryBuilder where(Map<String, String> conditions) {
    sql += conditions.entries.map((entry) => '${entry.key} = ${entry.value}').join('and');



    return this;
  }
}

abstract class Connector {
  Connector({required this.endpoint});

  final Endpoint endpoint;

  Future<bool> connect();
}
```

backend\lib\external\actions\clients.dart

```dart
import 'dart:developer';

import 'package:backend/data/actions/login.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:postgres/postgres.dart';

Future<ActionResult<Map<String, dynamic>, String>> getClientsBugCountAction(
    DateTime from, DateTime to) async {

  final String sql = """
    with TasksInRange as (
      select * from "Task" t where
      extract(epoch from date_trunc('day', end_ts)) >= @begin and extract(epoch from date_trunc('day', end_ts)) <= @end
    ),
    BugTasks as (
      select * from TasksInRange t join "TaggedTask" tt on tt.task_id = t.id where tt.tag = 'Frontend'
    )
    select c."name", count(t.id) from "Client" c join "Project" p ON c.id = p.client_id join BugTasks t on t.project_id  = p.id group by c.id
  """;

  final result = await conn.execute(Sql.named(sql), parameters: {
    'begin': from.millisecondsSinceEpoch / 1000,
    'end': to.millisecondsSinceEpoch / 1000
  });

  final Map<String, dynamic> ret = {
    for (var mapped in result.map((r) => r.toColumnMap())) mapped['name']: mapped['count']
```

```
  };

  return ActionResult.success(ret);
}

```

backend\lib\external\actions\employee.dart

```
import 'dart:developer';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/data/db/models/employee/employee.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';
import 'package:shelf/shelf_io.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<ActionResult<int, String>> _getEmployeeId(String email) async {
    final fetchBasicInfoSql = """
      select e.id from "Employee" e where e."email" = @email
  """;

  final result = await conn
      .execute(Sql.named(fetchBasicInfoSql), parameters: {'email': email});

  if (result.isEmpty) {
    return ActionResult.error('User not found');
  }

  var resultMap = result.first.toColumnMap();

  return ActionResult.success(resultMap['id']);
}

Future<ActionResult<List<Map<String, dynamic>>, String>> getAllEmployees() async {
  final fetchBasicInfoSql = """
select e.*, team.name as team_name, STRING_AGG(distinct s.code , ',') AS sprints from "Employee" e
join "Team" team on team.id  = e.team_id




join (select * from "Task" t order by t.begin_ts) t on e.id = t.employee_dev_id join "Sprint" s on s.id = t.sprint_id
group by e.id, team.name
  """;

  final result = await conn.execute(Sql.named(fetchBasicInfoSql));

  // final lastSprintsResult = await getLastSprintsFromEmployee(email, 3);
  return ActionResult.success(
      result.map((ResultRow row) {
      final mappedInfo = row.toColumnMap();
      return {
        'name': mappedInfo['name'],
        'email': mappedInfo['email'],
        'position': mappedInfo['position'],
        'teamName': mappedInfo['team_name'],
        'lastSprints': mappedInfo['sprints'].toString().split(',')
      };
    }).toList());
}

Future<ActionResult<Map<String, dynamic>, String>> getEmployeeBasicInfo(
    String email) async {
  final fetchBasicInfoSql = """
      select e.*, t."name" as team_name from "Employee" e join "Team" t on t.id = e.team_id where e."email" = @email
  """;

  final result = await conn
      .execute(Sql.named(fetchBasicInfoSql), parameters: {'email': email});

  final mappedInfo = result.first.toColumnMap();

  final lastSprintsResult = await getLastSprintsFromEmployee(email, 3);

  return ActionResult.success({
    'name': mappedInfo['name'],
    'email': mappedInfo['email'],
    'position': mappedInfo['position'],
    'teamName': mappedInfo['team_name'],
    'lastSprints': lastSprintsResult.success ? lastSprintsResult.data : []
  });
}
```

```
Future<ActionResult<List<String>, String>> getLastSprintsFromEmployee(
    String email, int k) async {
  const String fetchIdFromEmployeeSql =
      'select e.id from "Employee" e where e.email = @email';

  final queryResultEmployee = await conn
      .execute(Sql.named(fetchIdFromEmployeeSql), parameters: {'email': email});

  final employeeColumnMap = queryResultEmployee.first.toColumnMap();
  final employeeId = employeeColumnMap['id'];

  const String selectLastKSprintsSql = """
    with TasksFromEmployee as (
      select * from "Task" t where t.employee_dev_id = @employee_id
    ),
    SprintsIdFromEmployee as (
      select TasksFromEmployee.sprint_id as id from TasksFromEmployee group by TasksFromEmployee.sprint_id
    )
    select s.code from SprintsIdFromEmployee join "Sprint" s on s.id = SprintsIdFromEmployee.id order by s.end_ts limit @k
  """;

  final queryResult = await conn.execute(Sql.named(selectLastKSprintsSql),
      parameters: {'employee_id': employeeId, 'k': k});

  final lastSprints = queryResult.map((ResultRow r) {
    final rowMap = r.toColumnMap();
    return rowMap['code'] as String;
  }).toList();

  return ActionResult.success(lastSprints);
}

Future<ActionResult<Map<String, dynamic>, String>> _getEmployeeParticipationByProjectFromSprint(int sprintUUID, int employeeUUID) async {
    final String sql = """
    select p.*, sum(t.points) as participation_points from "Task" t
      join "Project" p on p.id = t.project_id
    where t.sprint_id = @sprintUUID and (t.employee_dev_id = @employeeUUID) group by p.id
  """;

  final queryResult = await conn.execute(Sql.named(sql), parameters: {'employeeUUID': employeeUUID, 'sprintUUID': sprintUUID});
```

```
  final Map<String, dynamic> result = {
    for (final map in queryResult.map((ResultRow row) => row.toColumnMap())) map['title']: map['participation_points']
  };

  return ActionResult.success(result);
}

Future<ActionResult<Map<String, dynamic>, String>> _getEmployeeParticipationByTagFromSprint(int sprintUUID, int employeeUUID) async {
  final String sql = """
    select tt.tag, sum(t.points) as participation_points from "Task" t
      join "TaggedTask" tt on t.id = tt.task_id
    where t.sprint_id = @sprintUUID and (t.employee_dev_id = @employeeUUID) group by tt.tag
  """;

  final queryResult = await conn.execute(Sql.named(sql), parameters: {'employeeUUID': employeeUUID, 'sprintUUID': sprintUUID});

  final Map<String, dynamic> result = {
    for (final map in queryResult.map((ResultRow row) => row.toColumnMap())) map['tag']: map['participation_points']
  };

  return ActionResult.success(result);
}

Future<ActionResult<Map<String, dynamic>, String>> _getEmployeeParticipationByClientFromSprint(int sprintUUID, int employeeUUID) async {
  final String sql = """
  select c.name, sum(t.points) as participation_points from "Task" t
    join "Project" p on p.id = t.project_id join "Client" c on c.id = p.client_id
  where t.employee_dev_id = @employeeUUID and t.sprint_id = @sprintUUID group by c.id

  """;

  final queryResult = await conn.execute(Sql.named(sql), parameters: {'employeeUUID': employeeUUID, 'sprintUUID': sprintUUID});

  final Map<String, dynamic> result = {
    for (final map in queryResult.map((ResultRow row) => row.toColumnMap())) map['name']: map['participation_points']
  };

  return ActionResult.success(result);
}
```

```
Future<ActionResult<Map<String, dynamic>, String>>
    getEmployeeParticipationReportFromSprint(
        String sprintCode, String employeeEmail) async {
  final sqlFetchEmployee =
      'select * from "Employee" e where e."email" = @email';
  final sqlFetchSprint = 'select * from "Sprint" s where s."code" = @code';


  final queryResultEmployee = await conn.execute(Sql.named(sqlFetchEmployee),
      parameters: {'email': employeeEmail});
  final queryResultSprint = await conn
      .execute(Sql.named(sqlFetchSprint), parameters: {'code': sprintCode});

  if (queryResultEmployee.isEmpty) {
    return ActionResult.error('UserNotFound');
  }

  if (queryResultSprint.isEmpty) {
    return ActionResult.error('SprintNotFound');
  }

  final int employeeUUID = queryResultEmployee.first[0] as int;
  final int sprintUUID = queryResultSprint.first[0] as int;

  final participationByClientResult = await _getEmployeeParticipationByClientFromSprint(sprintUUID, employeeUUID);
  final participationByProjectResult = await _getEmployeeParticipationByProjectFromSprint(sprintUUID, employeeUUID);
  final participationByTagResult = await _getEmployeeParticipationByTagFromSprint(sprintUUID, employeeUUID);

  Map<String, dynamic> result = {};

  if (participationByClientResult.success) result['participationByClient'] = participationByClientResult.data;
  if (participationByProjectResult.success) result['participationByProject'] = participationByProjectResult.data;
  if (participationByTagResult.success) result['participationByTag'] = participationByTagResult.data;

  return ActionResult.success(result);
}

Future<ActionResult<Map<String, dynamic>, String>>
    getEmployeePerformance(String email, List<String> projects, List<String> tags) async {

      return ActionResult.success({'': {}});
```

```
    }

Future<ActionResult<Map<String, dynamic>, String>>
    getEmployeePerformancePointsBySprints(String email, List<String> projects, List<String> tags, DateTime from, DateTime to) async {

      final projectsParams = {
          for (final pair in projects.indexed) 'P${pair.$1}': pair.$2
      };

      final tagsParams = {
          for (final pair in tags.indexed) 'T${pair.$1}': pair.$2
      };

      final userResult = await _getEmployeeId(email);

      if (!userResult.success) {
        return ActionResult.error(userResult.error!);
      }

      final int userId = userResult.data!;

      final sql = """
        WITH SprintsInRange AS (
          SELECT *
          FROM "Sprint" s
          WHERE extract(epoch from date_trunc('day', begin_ts)) >= @begin
            AND extract(epoch from date_trunc('day', end_ts)) <= @end
        ),
        TaskFromEmployee AS (
          SELECT t.id, t.sprint_id, t.points AS task_points, t.end_ts
          FROM "Task" t
          WHERE t.employee_dev_id = @userId
            AND t.project_id IN (SELECT id FROM "Project" p WHERE p.title IN (${projectsParams.keys.map((e) => '@$e',).join(',')}))
        ),
        TaggedTasksFromEmployee AS (
          SELECT *
          FROM TaskFromEmployee t
          WHERE t.id IN (
            SELECT tt.task_id
            FROM "TaggedTask" tt
            WHERE tt.task_id = t.id
```

```
            AND tt.tag IN (${tagsParams.keys.map((e) => '@$e',).join(',')})
          )
        )

        select sum(t.task_points) as total_points, s.code from TaggedTasksFromEmployee t join SprintsInRange s on t.sprint_id = s.id group by s.code
  """;

  final result = await conn.execute(Sql.named(sql), parameters: {
    ...projectsParams,
    ...tagsParams,
    'userId': userId,
    'begin': from.millisecondsSinceEpoch / 1000,
    'end': to.millisecondsSinceEpoch / 1000
  });

  final Map<String, dynamic> ret = {
    for (var mapped in result.map((r) => r.toColumnMap())) mapped['code']: mapped['total_points']
  };

  return ActionResult.success(ret);
}

Future<ActionResult<Map<String, dynamic>, String>>
    getEmployeePerformancePointsByDay(String email, List<String> projects, List<String> tags, DateTime from, DateTime to) async {

      final projectsParams = {
          for (final pair in projects.indexed) 'P${pair.$1}': pair.$2
      };

      final tagsParams = {
          for (final pair in tags.indexed) 'T${pair.$1}': pair.$2
      };


      final userResult = await _getEmployeeId(email);

      if (!userResult.success) {
        return ActionResult.error(userResult.error!);
      }

      final int userId = userResult.data!;




      final sql = """
        WITH SprintsInRange AS (
          SELECT *
          FROM "Sprint" s
          WHERE extract(epoch from date_trunc('day', begin_ts)) >= @begin
            AND extract(epoch from date_trunc('day', end_ts)) <= @end
        ),
        TaskFromEmployee AS (
          SELECT t.id, t.sprint_id, t.points AS task_points, t.end_ts
          FROM "Task" t
          WHERE t.employee_dev_id = @userId
            AND t.project_id IN (SELECT id FROM "Project" p WHERE p.title IN (${projectsParams.keys.map((e) => '@$e',).join(',')}))
        ),
        TaggedTasksFromEmployee AS (
          SELECT *
          FROM TaskFromEmployee t
          WHERE t.id IN (
            SELECT tt.task_id
            FROM "TaggedTask" tt
            WHERE tt.task_id = t.id
              AND tt.tag IN (${tagsParams.keys.map((e) => '@$e',).join(',')})
          )
        )
        select extract(epoch from date_trunc('day', t.end_ts)) as day,
          sum(task_points) total from TaggedTasksFromEmployee t join SprintsInRange s on t.sprint_id = s.id group by date_trunc('day', t.end_ts)
  """;

  final result = await conn.execute(Sql.named(sql), parameters: {
    ...projectsParams,
    ...tagsParams,
    'userId': userId,
    'begin': from.millisecondsSinceEpoch / 1000,
    'end': to.millisecondsSinceEpoch / 1000
  });

  final Map<String, dynamic> ret = {
    for (var start = from; start.compareTo(to) <= 0; start = start.add(Duration(days: 1))) start.millisecondsSinceEpoch.toString(): 0
  };

  for (var mapped in result.map((r) => r.toColumnMap())) {
```

```
    int dayInMilliseconds = (double.parse(mapped['day']) * 1000).toInt();
    ret[dayInMilliseconds.toString()] = mapped['total'];
  }

  return ActionResult.success(ret);
}
```

backend\lib\external\actions\login.dart

```
import 'dart:developer';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/data/db/models/employee/employee.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

class JWTToken extends AccessToken<String> {

  JWTToken({required token}): _token = token;

  @override
  List<String> get permissions => ['readAll'];

  @override
  get token => _token;

  final String _token;
}

Future<ActionResult<LoginResult, String>>  login(
    String email, String password) async {
  final query = await conn.execute(
    Sql.named(
        'SELECT id, name, position FROM "Employee" e join "Password" p on e."id" = p."employee_id" WHERE email=@email'),
    parameters: {'email': email},
  );

  if (query.isEmpty) {
    return ActionResult.error('Email not found');
  }

  print('here');

  final userId = query.first[0] as int;

  final String name = query.first[1] as String;

  final String position = query.first[2] as String;

  final employee = Employee(
    name: name,
    email: email,
    position: position,
  );

  final user = query.first;

  final issued_at = DateTime.now();
  final expires_at = issued_at.add(Duration(hours: 1));

  final payload = {
    'iss': '',
    'sub': BackendConfig.subject.auth,
    'user': {'name': employee.name},
    'iat': issued_at.toIso8601String(),
    'eat': expires_at.toIso8601String(),
    'jti': ''
  };

  final jwt = JWT(payload, header: {});

  final token = jwt.sign(authTokenSecret);

  final queryToken = await conn.execute(
      Sql.named(
          'insert into "Auth" values (@user_id, @token, @issued_at, @expires_at)'),
```

```
      parameters: {
        'user_id': userId,
        'token': token,
        'issued_at': issued_at.toIso8601String(),
        'expires_at': expires_at.toIso8601String(),
      });

  final loginResult = LoginResult(employee: employee, token: JWTToken(token: token));

  return ActionResult.success(loginResult);
}

```

backend\lib\external\actions\projects.dart

```
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:postgres/postgres.dart';


Future<ActionResult<bool, String>> _projectExists(int projectId) async {
  final fetchProject = """
      select id from "Project" p where p.id = @projectId
  """;

  final result = await conn
      .execute(Sql.named(fetchProject), parameters: {'projectId': projectId});

  return ActionResult.success(result.isEmpty);
}

Future<ActionResult<List<Map<String, dynamic>>, String>> getProjects() async {
  final fetchProjectsSql = 'select * from "Project" p';
  final result = await conn.execute(Sql.named(fetchProjectsSql));

  final projectsList = result.map((ResultRow row) {
    final columnMap = row.toColumnMap();

    return {
      'id': columnMap['id'],
      'title': columnMap['title'],
      'description': columnMap['description'],
      'goal': columnMap['goal']
    };
  }).toList();

  return ActionResult.success(projectsList);
}

Future<ActionResult<List<Map<String, dynamic>>, String>> getTasksFromProject(
    int projectId) async {
  final fetchTasksSql = """
    with TasksFromProject as (
      select
        t.id, s.code as sprint_code, project_id, employee_dev_id, employee_qa_id, title,
        t.description, extract (epoch from t.creation_ts) as creation_ts,
        extract (epoch from t.begin_ts) as begin_ts,
        extract (epoch from t.end_ts) as end_ts,
        t.points
      from "Task" t join "Sprint" s on t.sprint_id = s.id
      where t.project_id = @projectId
    ),
    TagAggregate as (
      select tt.task_id, string_agg(tt.tag, ',') as tags from "TaggedTask" tt group by tt.task_id
    ),
    TaggedTasks as (
      select tfs.*, tt.tags from TasksFromProject tfs join TagAggregate tt on tfs.id = tt.task_id
    )
    select
           tt.*,
           dev.name as dev_name,
           dev.email as dev_email,
           qa.name as qa_name,
           qa.email as qa_email
    from TaggedTasks tt
      join "Employee" dev on tt.employee_dev_id = dev.id
      join "Employee" qa on tt.employee_qa_id = qa.id
  """;

  final result = await conn
      .execute(Sql.named(fetchTasksSql), parameters: {'projectId': projectId});

```

```
  if (result.isEmpty) {
    return ActionResult.error('ProjectNotFound');
  }

  final tasks = result.map((ResultRow r) {
    final map = r.toColumnMap();
    final creationTs = double.parse(map['creation_ts']) * 1000;
    final tags = map['tags'] as String;
    final beginTs = double.parse(map['begin_ts']) * 1000;
    final endTs = double.parse(map['end_ts']) * 1000;

    return {
      'id': map['id'],
      'sprintCode': map['sprint_code'],
      'title': map['title'],
      'description': map['description'],
      'devName': map['dev_name'],
      'devEmail': map['dev_email'],
      'qaName': map['qa_name'],
      'qaEmail': map['qa_email'],
      'linkedTs': creationTs.toInt(),
      'beginTs': beginTs.toInt(),
      'endTs': endTs.toInt(),
      'points': map['points'],
      'tags': tags.split(',')
    };
  }).toList();

  return ActionResult.success(tasks);
}

```

backend\lib\external\actions\sprints.dart

```
import 'dart:developer';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';




final authTokenSecret = SecretKey(BackendConfig.secret);

Future<ActionResult<Map<String, dynamic>, String>> getSprint(
    String sprintCode) async {
  final sql = """
      with SS as (select
            s.id,
            s.team_id,
            s.code,
            s.description,
            extract(epoch from s.begin_ts) as begin_date,
            extract(epoch from s.end_ts) as end_date,
          s.points,
            sum(t.points) as points_completed
          from "Sprint" s join "Task" t on s.id = t.sprint_id where s.code = @code group by s.id
      )
      select SS.*, t.name as team_name from SS join "Team" t on t.id = SS.team_id
    """;

  final query =
      await conn.execute(Sql.named(sql), parameters: {'code': sprintCode});
  final columnMap = query.first.toColumnMap();

  final double millisecondsBegin = double.parse(columnMap['begin_date'] as String) * 1000;
  final double millisecondsEnd = double.parse(columnMap['end_date'] as String) * 1000;

  final beginDate =
      DateTime.fromMillisecondsSinceEpoch(millisecondsBegin.toInt());
  final endDate = DateTime.fromMillisecondsSinceEpoch(millisecondsEnd.toInt());

  final code = columnMap['code'] as String;
  final description = columnMap['description'] as String;
  final teamName = columnMap['team_name'] as String;
  final int pointsCommited = columnMap['points'] as int;
  final int pointsCompleted = columnMap['points_completed'] as int;

  final result = {
    'title': 'Sprint - $code',
    'beginDate': beginDate.millisecondsSinceEpoch,
    'endDate': endDate.millisecondsSinceEpoch,
```

```
    'code': code,
    'description': description,
    'teamName': teamName,
    'pointsTotal': pointsCommited,
    'pointsDone': pointsCompleted,
    'done': endDate.compareTo(DateTime.now()) < 0
  };

  return ActionResult.success(result);
}

Future<ActionResult<List<dynamic>, String>> getSprints() async {
  final query = await conn.execute(Sql.named("""
      with SS as (select
            s.id,
            s.team_id,
            code,
            s.description,
            extract(epoch from s.begin_ts) as begin_date,
            extract(epoch from s.end_ts) as end_date,
          s.points,
            sum(t.points) as points_completed
          from "Sprint" s join "Task" t on s.id = t.sprint_id group by s.id
      )
      select SS.*, t.name as team_name from SS join "Team" t on t.id = SS.team_id

"""));

  if (query.isEmpty) {
    return ActionResult.error('error');
  }

  // id, project_id, team_id, code, description, begin, end
  final double millisecondsBegin =
      (double.parse(query.first[4] as String) * 1000);
  final double millisecondsEnd =
      (double.parse(query.first[5] as String) * 1000);

  final dateBegin =
      DateTime.fromMillisecondsSinceEpoch(millisecondsBegin.toInt());
  final dateEnd = DateTime.fromMillisecondsSinceEpoch(millisecondsEnd.toInt());



  final sprints = query.map((row) {
    final columnsMap = row.toColumnMap();
    final code = columnsMap['code'];
    final title = 'Sprint - $code';
    final description = columnsMap['description'];
    final teamName = columnsMap['team_name'];
    final points = columnsMap['points'];
    final pointsCompleted = columnsMap['points_completed'];

    return {
      'title': title,
      'teamName': teamName,
      'beginDate': millisecondsBegin.toInt(),
      'endDate': millisecondsEnd.toInt(),
      'code': code,
      'description': description,
      'pointsTotal': points,
      'pointsDone': pointsCompleted,
      'done': dateEnd.compareTo(DateTime.now()) < 0
    };
  }).toList();

  return ActionResult.success(sprints);
}

Future<ActionResult<List<dynamic>, String>> getTasksFromSprint(
    String sprintCode) async {
  const String fetchSprintSql =
      'select id from "Sprint" s where s.code = @code';
  final result = await conn
      .execute(Sql.named(fetchSprintSql), parameters: {'code': sprintCode});

  if (result.isEmpty) {
    return ActionResult.success([]);
  }

  int sprintId = result.first[0] as int;

  const String fetchTasksSql = """
        with TasksFromSprint as (
```

```
            select
                t.id, s.code as sprint_code, project_id, employee_dev_id, employee_qa_id, title,
                t.description, extract (epoch from t.creation_ts) as creation_ts,
                extract (epoch from t.begin_ts) as begin_ts,
                extract (epoch from t.end_ts) as end_ts,
                t.points
            from "Task" t join "Sprint" s on t.sprint_id = s.id
        where t.sprint_id = @sprintId
          ),
          TagAggregate as (
            select tt.task_id, string_agg(tt.tag, ',') as tags from "TaggedTask" tt group by tt.task_id
          ),
          TaggedTasks as (
            select tfs.*, tt.tags from TasksFromSprint tfs join TagAggregate tt on tfs.id = tt.task_id
          )
          select
          tt.*,
          dev.name as dev_name,
          dev.email as dev_email,
          qa.name as qa_name,
          qa.email as qa_email
      from TaggedTasks tt
            join "Employee" dev on tt.employee_dev_id = dev.id
            join "Employee" qa on tt.employee_qa_id = qa.id
      """;

  final query = await conn
      .execute(Sql.named(fetchTasksSql), parameters: {'sprintId': sprintId});

  final tasks = query.map((ResultRow r) {
    final map = r.toColumnMap();
    final creationTs = double.parse(map['creation_ts']) * 1000;
    final tags = map['tags'] as String;
    final beginTs = double.parse(map['begin_ts']) * 1000;
    final endTs = double.parse(map['end_ts']) * 1000;

    return {
      'id': map['id'],
      'sprintCode': map['sprint_code'],
      'title': map['title'],
      'description': map['description'],
      'devName': map['dev_name'],
      'devEmail': map['dev_email'],
      'qaName': map['qa_name'],
      'qaEmail': map['qa_email'],
      'linkedTs': creationTs.toInt(),
      'beginTs': beginTs.toInt(),
      'endTs': endTs.toInt(),
      'points': map['points'],
      'tags': tags.split(',')
    };
  });

  return ActionResult.success(tasks.toList());
}

Future<ActionResult<Map<String, dynamic>, String>> getPointsDoneByDay(
    String sprintCode) async {
  String sql = """
            select
            id, extract(epoch from date_trunc('day', begin_ts)), extract(epoch from date_trunc('day', end_ts)), points
        from "Sprint" s where code = @code""";

  final querySprint =
      await conn.execute(Sql.named(sql), parameters: {'code': sprintCode});

  final columnMap = querySprint.first.toColumnMap();

  final secondsSinceEpochBegin =
      double.parse(querySprint.first[1] as String) * 1000;
  final secondsSinceEpochEnd =
      double.parse(querySprint.first[2] as String) * 1000;

  final sprintId = querySprint.first[0] as int;
  var currentDay =
      DateTime.fromMillisecondsSinceEpoch(secondsSinceEpochBegin.toInt());
  final endDay =
      DateTime.fromMillisecondsSinceEpoch(secondsSinceEpochEnd.toInt());

  Map<String, int> points = {};

  while (currentDay.compareTo(endDay) <= 0) {
```

```
    final int currentSecondsSinceEpoch = currentDay.millisecondsSinceEpoch;
    points[currentSecondsSinceEpoch.toString()] = 0;
    currentDay = currentDay.add(Duration(days: 1));
  }

  sql =
      """select extract(epoch from date_trunc('day', end_ts)) as finished_date, sum(t.points) as points_done from "Task" t
    where sprint_id = @sprint_id group by finished_date""";

  final tasksQuery =
      await conn.execute(Sql.named(sql), parameters: {'sprint_id': sprintId});

  for (final row in tasksQuery) {
    final secondsSinceEpoch = (double.parse(row[0] as String) * 1000).toInt();
    points[secondsSinceEpoch.toString()] = row[1] as int;
  }

  return ActionResult.success({'pointsHistory': points, 'totalPoints': columnMap['points']});
}

Future<ActionResult<List<Map<String, dynamic>>, String>>
    getPointsDoneFromLastKSprints(String sprintCode, String k) async {
  final sql = """
            select LastKSprints.code, sum(t.points) from
                (select * from
                    (select id, code, begin_ts from "Sprint" order by begin_ts desc) as OrdSprint
                        WHERE OrdSprint.begin_ts <= (select begin_ts from "Sprint" where code = @code) limit @K) as LastKSprints
                            join "Task" t on t.sprint_id = LastKSprints.id
                                group by LastKSprints."code"
            """;

  final query = await conn
      .execute(Sql.named(sql), parameters: {'code': sprintCode, 'K': k});

  final pointsHistory = query
      .map((row) => {
            row[0] as String: {
              'pointsDone': row[1] as int,
              'pointsCommited': 45
            }
          })
```

```
      .toList();

  return ActionResult.success(pointsHistory);
}

Future<ActionResult<Map<String, String>, String>> getWorkScopeFromSprintHistory(
    String sprintCode) async {
  final sqlSprint = """
            SELECT
                s.id,
                extract(epoch from date_trunc('day', begin_ts)) as sprint_begin_ts,
                extract(epoch from date_trunc('day', end_ts)) as sprint_end_ts
            FROM "Sprint" s
            WHERE s.code = @code
        """;

  final querySprint = await conn
      .execute(Sql.named(sqlSprint), parameters: {'code': sprintCode});

  final sprintId = querySprint.first[0] as int;
  final sprintBegin = DateTime.fromMillisecondsSinceEpoch(
      (double.parse(querySprint.first[1] as String) * 1000).toInt());
  final sprintEnd = DateTime.fromMillisecondsSinceEpoch(
      (double.parse(querySprint.first[2] as String) * 1000).toInt());

  final sql = """
        WITH SelectedSprint as (
            SELECT *
            FROM "Sprint" s
            WHERE s.code = @code
        )
        select
            extract(epoch from date_trunc('day', t.creation_ts)) as day_change,
            sum(t.points)
        from SelectedSprint s join "Task" t on
            s.id = t.sprint_id group by day_change
    """;

  final query =
      await conn.execute(Sql.named(sql), parameters: {'code': sprintCode});

```

```
  final Map<String, String> result = {
    for (var current = sprintBegin;
        current.compareTo(sprintEnd) <= 0;
        current = current.add(Duration(days: 1)))
      current.millisecondsSinceEpoch.toString(): '0',
    for (var v in query)
      epochColumnToEpochKey(v[0] as String): (v[1] as int).toString()
  };

  return ActionResult.success(result);
}

Future<ActionResult<Map<String, int>, String>> getWorkInProgressHistoryFromSprint(
    String sprintCode) async {
  final sql =
    """
      WITH range_values AS (
        SELECT date_trunc('day', begin_ts) as minval,
              date_trunc('day', end_ts) as maxval
        FROM "Sprint" s where s.code = @code),

      day_range AS (
        SELECT generate_series(minval, maxval, '1 day'::interval) as day
        FROM range_values
      )

      select extract(epoch from day) as day, sum(t.points) as wip_points from day_range
        left join "Task" t on "day_range".day >= date_trunc('day', t.begin_ts) and "day_range".day <= date_trunc('day', t.end_ts)
        group by day
        order by day
    """;

    final result = await conn.execute(Sql.named(sql), parameters: {'code': sprintCode});

    Map<String, int> transformer(ResultRow r) {
      final columnMap = r.toColumnMap();
      final dayEpoch = double.parse(columnMap['day']) * 1000;
      return {
        dayEpoch.toInt().toString(): columnMap['wip_points'] ?? 0
      };
    }




    final Map<String, int> workInProgressHistory = {
      for (ResultRow r in result) ...transformer(r)
    };

    return ActionResult.success(workInProgressHistory);
}

String epochColumnToEpochKey(String epoch) {
  return (double.parse(epoch) * 1000).toInt().toString();
}

```

backend\lib\external\actions\teams.dart

```
import 'dart:developer';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/data/db/models/employee/employee.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<ActionResult<List<Map<String, dynamic>>, String>> getTeams() async {
  final fetchBasicInfoSql = """
      select t.*,
        string_agg(e."name", ',') as team_member_names,
        string_agg(e."email", ',') as team_member_emails
      from "Team" t join "Employee" e on t.id = e.team_id
      --join "Sprint" s on s.team_id = t.id
      group by t.id
  """;

  final result = await conn
      .execute(Sql.named(fetchBasicInfoSql));

  Map<String, dynamic> transformer(ResultRow row) {
    final columnMap = row.toColumnMap();
```

```
    final teamName = columnMap['name'];
    final teamMemberNames = (columnMap['team_member_names'] as String).split(',').toList();
    final teamMemberEmails = (columnMap['team_member_emails'] as String).split(',').toList();

    return {
      'teamName':  teamName,
      'membersNames': teamMemberNames,
      'memberEmails': teamMemberEmails,
    };
  }

  final teams = result.map((ResultRow row) => transformer(row)).toList();

  return ActionResult.success(teams);
}
```

backend\lib\external\handlers\middlewares\authorize.dart

```
import 'package:backend/backend_config.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:shelf/shelf.dart';

Middleware authorize() => (Handler innerHandler)  {
  return (request) async {
    final token = request.headers['Authorization'];

    if (token == null) return Response.unauthorized('');

    final tokenChecked = JWT.tryVerify(token, SecretKey(BackendConfig.secret));

    if (tokenChecked == null) return Response.unauthorized('');

    return innerHandler(request);
  };
};
```

backend\lib\external\handlers\clients.dart

```
import 'dart:convert';
import 'dart:developer';
```

```
import 'package:backend/backend_config.dart';
import 'package:backend/external/actions/clients.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:shelf/shelf.dart';
import 'package:shelf_router/shelf_router.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<Response> getClientsBugCountHandler(Request request) async {
    final params = request.url.queryParameters;
    final millisecondsSinceEpochBegin = int.parse(params['millisecondsSinceEpochBegin'] as String);
    final millisecondsSinceEpochEnd = int.parse(params['millisecondsSinceEpochEnd'] as String);

    final result = await getClientsBugCountAction(DateTime.fromMillisecondsSinceEpoch(millisecondsSinceEpochBegin), DateTime.fromMillisecondsSinc↩
eEpoch(millisecondsSinceEpochEnd));

    if (!result.success) {
      return Response.badRequest();
    }

  return Response.ok(jsonEncode(result.data));
}
```

backend\lib\external\handlers\employee_handlers.dart

```
import 'dart:convert';
import 'dart:developer';
import 'dart:io';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/actions/employee.dart';
import 'package:backend/external/actions/login.dart';
import 'package:backend/external/actions/sprints.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';
import 'package:shelf/shelf.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);
```

```dart

Future<Response> getEmployeeBasicInfoHandler(Request request, String email) async {
  final employeeInfoResult = await getEmployeeBasicInfo(email);

  if (!employeeInfoResult.success) {
    return Response.badRequest();
  }

  return Response.ok(jsonEncode({
    'info': employeeInfoResult.data
  }));
}

Future<Response> getEmployeeParticipationReportFromSprintHandler(Request request, String employeeEmail, String sprintCode) async {
  final result = await getEmployeeParticipationReportFromSprint(sprintCode, employeeEmail);

  if (!result.success) {
    return Response.badRequest();
  }
  return Response.ok(jsonEncode({
    'participation': result.data,
  }));
}

Future<Response> getEmployeePerformanceHandler(Request request, email) async {
  final params = jsonDecode(await request.readAsString());
  final granularity = params['granularity'] as String;
  final millisecondsSinceEpochBegin = params['range']['begin'] as int;
  final millisecondsSinceEpochEnd = params['range']['end'] as int;
  final projects = (params['projects'] as List<dynamic>).map((project) => project as String).toList();
  final tags = (params['tags'] as List<dynamic>).map((tag) => tag as String).toList();

  final _handler = granularity == 'days' ? getEmployeePerformancePointsByDay : getEmployeePerformancePointsBySprints;

  final ret = await _handler(
      email,
      projects,
      tags,
      DateTime.fromMillisecondsSinceEpoch(millisecondsSinceEpochBegin),
      DateTime.fromMillisecondsSinceEpoch(millisecondsSinceEpochEnd),
  );




  return Response.ok(jsonEncode(ret.data));
}

Future<Response> getAllEmployeesHandler(Request request ) async {
  final employees = await getAllEmployees();
  return Response.ok(jsonEncode(employees.data));
}

```

backend\lib\external\handlers\login.dart

```dart
import 'dart:convert';
import 'dart:io';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/actions/login.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';
import 'package:shelf/shelf.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<Response> authHandler(Request request) async {
    final body = jsonDecode(await request.readAsString());
    final email = body['user'] as String;
    final password = body['password'] as String;

    final result = await login(email, password);

    if (!result.success) {
        return Response.badRequest();
    }

    final loginResult = result.data as LoginResult;
    final user = loginResult.employee;
    final token = loginResult.token.token;

    return Response.ok(jsonEncode({
```

```
        'token': token,
    'user': {
      'name': user.name,
      'email': user.email,
    },
    }));
}

Future<Response> testAuthorizationHandler(Request request) async {
    return Response.ok('This request is successfully authorized');
}

```

backend\lib\external\handlers\projects.dart

```
import 'dart:convert';

import 'package:backend/external/actions/projects.dart';
import 'package:shelf/shelf.dart';

Future<Response> getProjectsHandler(Request request) async {
  final result = await getProjects();

  return Response.ok(jsonEncode(result.data));
}

Future<Response> getTasksFromProjectHandler(Request request, String projectId) async {
  final result = await getTasksFromProject(int.parse(projectId));

  return Response.ok(jsonEncode(result.data));
}
```

backend\lib\external\handlers\sprints.dart

```
import 'dart:convert';
import 'dart:developer';
import 'dart:io';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/actions/employee.dart';




import 'package:backend/external/actions/login.dart';
import 'package:backend/external/actions/sprints.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';
import 'package:shelf/shelf.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<Response> getSprintHandler(Request request, String sprintCode) async {
  final result = await getSprint(sprintCode);

  if (!result.success) {
    return Response.badRequest();
  }

  final sprint = result.data;

  return Response.ok(jsonEncode({
    'sprint': sprint
  }));
}

Future<Response> getSprintsHandler(Request request) async {

  final result = await getSprints();

  if (!result.success) {
    return Response.badRequest();
  }

  final sprints = result.data;

    return Response.ok(jsonEncode({
        'sprints': sprints,
    }));
}

Future<Response> getTasksFromSprintHandler(Request request, String sprintCode) async {

  final result = await getTasksFromSprint(sprintCode);
```

```

  if (!result.success) {
    return Response.badRequest();
  }

  final tasks = result.data;

    return Response.ok(jsonEncode({
        'tasks': tasks,
    }));
}

Future<Response> getPointsDonePerDay(Request request, String sprintCode) async {

  final result = await getPointsDoneByDay(sprintCode);

  if (!result.success) {
    return Response.badRequest();
  }

  return Response.ok(jsonEncode({
    'data': result.data
  }));
}

Future<Response> getPointsFromLastKSprintsHandler(Request request, String sprintCode, String k) async {
  final result = await getPointsDoneFromLastKSprints(sprintCode, k);

  if (!result.success) {
    return Response.badRequest();
  }

  return Response.ok(jsonEncode({
    'history': result.data!
  }));
}

Future<Response> getWorkScopeFromSprintHistoryHandler(Request request, String sprintCode) async {
  final result = await getWorkScopeFromSprintHistory(sprintCode);

  if (!result.success) return Response.badRequest();




  return Response.ok(jsonEncode({
    'workloadHistory': result.data
  }));
}

Future<Response> getWorkInProgressHistoryFromSprintHandler(Request request, String sprintCode) async {
  final result = await getWorkInProgressHistoryFromSprint(sprintCode);

  if (!result.success) {
    return Response.badRequest();
  }

  return Response.ok(jsonEncode({
    'workInProgressHistory': result.data
  }));
}

```

backend\lib\external\handlers\team_handler.dart

```
import 'dart:convert';
import 'dart:developer';
import 'dart:io';

import 'package:backend/backend_config.dart';
import 'package:backend/data/actions/login.dart';
import 'package:backend/external/actions/employee.dart';
import 'package:backend/external/actions/login.dart';
import 'package:backend/external/actions/sprints.dart';
import 'package:backend/external/actions/teams.dart';
import 'package:backend/external/db/init_connection.dart';
import 'package:dart_jsonwebtoken/dart_jsonwebtoken.dart';
import 'package:postgres/postgres.dart';
import 'package:shelf/shelf.dart';

final authTokenSecret = SecretKey(BackendConfig.secret);

Future<Response> getTeamsHandler(Request request) async {
  final employeeInfoResult = await getTeams();

```

```
  if (!employeeInfoResult.success) {
    return Response.badRequest();
  }

  return Response.ok(jsonEncode({
    'teams': employeeInfoResult.data
  }));
}
```

backend\lib\external\routers\router.dart

```
import 'package:backend/external/handlers/clients.dart';
import 'package:backend/external/handlers/employee_handlers.dart';
import 'package:backend/external/handlers/login.dart';
import 'package:backend/external/handlers/projects.dart';
import 'package:backend/external/handlers/sprints.dart';
import 'package:backend/external/handlers/team_handler.dart';
import 'package:shelf_router/shelf_router.dart';

final globalRouter = Router()
    /* sprints */
  ..post('/login', authHandler)
  ..post('/sprints/', getSprintsHandler)
  ..get('/sprints/<code>', getSprintHandler)
  ..get('/sprint/<code>/tasks', getTasksFromSprintHandler)
  ..get('/sprint/<code>/story-points-history', getPointsDonePerDay)
  ..get('/sprint/<code>/story-points-from-last-k-sprints/<k>', getPointsFromLastKSprintsHandler)
  ..get('/sprint/<code>/work-scope-history', getWorkScopeFromSprintHistoryHandler)
  ..get('/sprint/<code>/work-in-progres-history', getWorkInProgressHistoryFromSprintHandler)
  /* Employees */
  ..get('/employee', getAllEmployeesHandler)
  ..get('/employee/<email>/basic-info', getEmployeeBasicInfoHandler)
  ..get('/employee/<email>/participation-report/sprint/<code>', getEmployeeParticipationReportFromSprintHandler)
  ..post('/employee/<email>/performance-report', getEmployeePerformanceHandler)
  /* Teams */
  ..get('/teams', getTeamsHandler)
  /* Clients */
  ..get('/clients/bugs-counts', getClientsBugCountHandler)
  /* Projects */
  ..get('/projects', getProjectsHandler)
  ..get('/projects/<projectId>/tasks', getTasksFromProjectHandler);
  // ..post('/test-authorization-route/', Pipeline().addMiddleware(authorize()).addHandler(testAuthorizationHandler));
```

backend\lib\backend_config.dart

```
class BackendConfig {
  static const String secret = 'change this later';
  static const Subject subject = Subject();
}

class Subject {
  const Subject();
  static const AUTH_SUB = 'Authentication';

  String get auth => Subject.AUTH_SUB;
}
```

frontend\src\components\metrics\charts\burndown_chart.js

```
import { LineChart } from "@mui/x-charts"
import { defaultDateFormatter } from "./date_formatters";

const range = (start, stop, step) =>
    Array.from({ length: (stop - start) / step + 1 }, (_, i) => start + i * step);


export const BurndownChart = ({ totalPoints, culmulativePointsDonePerDay, days, config = {width: 800, height: 650} }) => {

    let { width, height } = config

    const daysInSprint = culmulativePointsDonePerDay.length
    const xAxisData = range(1, daysInSprint, 1)
    const yAxisData = culmulativePointsDonePerDay.map((points) => totalPoints - points)
    const pointsExpectedPerDay = totalPoints / daysInSprint

    const culmulativePointsExpected = range(1, daysInSprint, 1).map(value => totalPoints - pointsExpectedPerDay * value)

    const yAxisDataExpected = range(1, daysInSprint, pointsExpectedPerDay)

```

```
    return <LineChart
        xAxis={[{ data: days, scaleType: 'time', valueFormatter: defaultDateFormatter }]}
        series={[
            {
                data: yAxisData,
                color: 'blue',
                label: 'Realized Points'
            },
            {
                data: culmulativePointsExpected,
                label: 'Remaining Points'
            }
        ]}
    />
}
```

frontend\src\components\metrics\charts\burnup_chart.js

```
import { LineChart } from "@mui/x-charts"
import { defaultDateFormatter } from "./date_formatters";

export const BurnUpChart = ({ workloadHistory, culmulativePointsDonePerDay, days, config = {width: 800, height: 650} }) => {

    let { width, height } = config

    return <LineChart
        slotProps={{
            legend: {
                padding: {
                    // top: 150,
                    bottom: 190
                    // left: 600
                },
                // direction: 'row',
                // position: { vertical: 'bottom', horizontal: "right" },
                markGap: 10,
                itemGap: 10
            }
        }}
        xAxis={[{ data: days, scaleType: 'time', valueFormatter: defaultDateFormatter }]}
        series={[
            {
                data: workloadHistory,
                color: 'red',
                label: 'Workload'
            },
            {
                data: culmulativePointsDonePerDay,
                label: 'Points Completed'
            }
        ]}
    />
}
```

frontend\src\components\metrics\charts\date_formatters.js

```
export const defaultDateFormatter = (date) => {
    const month = date.getMonth() + 1
    const day = date.getDate()

    const appendZero = (value) => value < 10 ? `0${value}`: value

    return `${appendZero(day)}/${appendZero(month)}`
}

export const sprintXAxisFormater = (sprintCode) => sprintCode

```

frontend\src\components\metrics\charts\utils.js

```
export const deltaListToCummulativeList = (diffsList) => {
    let sum = 0
    return diffsList.map((diff) => {
        sum += diff
        return sum
    })
}
```

frontend\src\components\metrics\charts\velocity.js

```
import { Box } from "@mui/material"
import { BarChart } from "@mui/x-charts"
```

```

export const Velocity = ({
  sprintsNames,
  commitedPointsFromLastKSprints,
  realizedPointsFromLastKSprints,
  config = { width: 200, height: 200 }
}) => {

  const series = [{
    data: commitedPointsFromLastKSprints,
    label: 'Commited'
  },
  {
    data: realizedPointsFromLastKSprints,
    label: 'Realized'
  }]

  const expectedVelocity = commitedPointsFromLastKSprints.reduce((acc, curr) => acc + curr, 0) / commitedPointsFromLastKSprints.length
  const realizedVelocity = realizedPointsFromLastKSprints.reduce((acc, curr) => acc + curr, 0) / realizedPointsFromLastKSprints.length

  const { width, height } = config

  return (
    <Box display='flex' flexDirection='row' flex={1} height={'fit-content'} boxSizing={'border-box'}>
      <Box flex={1}>
        <BarChart
          xAxis={[{ scaleType: 'band', data: sprintsNames }]}
          series={series}
          width={width}
          height={height}
        />
      </Box>
      <Box>

        <BarChart
          xAxis={[{ scaleType: 'band', data: ['Velocity'] }]}
          series={[{ data: [expectedVelocity], label: 'Avg. Commited' }, { label: 'Avg. Realized', data: [realizedVelocity] }]}
          sx={{ flexGrow: 1 }}
          width={width}
          height={height}
        />




      </Box>
    </Box>
  )
}
```

frontend\src\components\metrics\charts\work_in_progress_chart.js

```
import { LineChart } from "@mui/x-charts"
import { defaultDateFormatter } from "./date_formatters"

export const WIPChart = ({ workInProgressPointHistory, days }) => {
    return <LineChart
        xAxis={[{ data: days, scaleType: 'time', valueFormatter: defaultDateFormatter }]}
        series={[
            {
                data: workInProgressPointHistory,
                color: 'blue',
                label: 'Realized Points'
            },
        ]}
    />
}
```

frontend\src\components\metrics\customized\charts\moving_average_velocity_chart.js

```
import { Box } from "@mui/material"
import { BarChart, LineChart } from "@mui/x-charts"
import { axisClasses } from '@mui/x-charts/ChartsAxis';
import { range } from "../../../../util/functions";

export const MovingAverageVelocityChart = ({ commitedPointsHistory, realizedPointsHistory, K }) => {

  const commitedPointsMovingAverages = []
  const realizedPointsMovingAverages = []

  for (let i = 0; i < commitedPointsHistory.length; i++) {
    const n = Math.min(K, i)

    const avg0 = commitedPointsHistory.slice(i - n, i + 1).reduce((acc, curr) => acc + curr, 0) / (n+1)
    const avg1 = realizedPointsHistory.slice(i - n, i + 1).reduce((acc, curr) => acc + curr, 0) / (n+1)

```

```
    commitedPointsMovingAverages.push(avg0)
    realizedPointsMovingAverages.push(avg1)

  }

  return (
    <LineChart
      xAxis={[{ data: range(0, commitedPointsHistory.length, 1) }]}
      series={[
        {
          data: commitedPointsMovingAverages,
          color: 'blue'
        },
        {
          data: realizedPointsMovingAverages,
          color: 'red'
        }
      ]}
      // width={500}
      // height={300}
    />
  );
}
```

frontend\src\components\shared\drawer_aware\drawer_aware.js

```
import { Box } from "@mui/material"
import DrawerContext from "../../../pages/home/home/drawer_context"
import { useContext } from "react"

const paddingLeft = 64
const widthOpen = 240 + paddingLeft
const widthClosed = paddingLeft

export const DrawerAware = ({ children }) => {
    const isDrawerOpen = useContext(DrawerContext)

    const _toPx = (value) => `${value}px`
    const _getPaddingLeft = () => _toPx(isDrawerOpen ? widthOpen : widthClosed)

    return (




        <Box width={'100%'} display={'flex'} flexDirection={'row'} height={`calc(100vh - 220px)`} bgcolor="" sx={{overflow: 'scroll'}}>
            <Box width={_getPaddingLeft()} sx={{ transitionDuration: '500ms' }} />
            { children }
        </Box>
    )
}

```

frontend\src\components\shared\hooks\use_local_storage.js

```
import { useState } from "react";

export const useLocalStorage = (keyName, defaultValue) => {
    const [storedValue, setStoredValue] = useState(() => {
      try {
        const value = window.localStorage.getItem(keyName);
        if (value) {
          return JSON.parse(value);
        } else {
          window.localStorage.setItem(keyName, JSON.stringify(defaultValue));
          return defaultValue;
        }
      } catch (err) {
        return defaultValue;
      }
    });
    const setValue = (newValue) => {
      try {
        window.localStorage.setItem(keyName, JSON.stringify(newValue));
      } catch (err) {
        console.log(err);

      }
      setStoredValue(newValue);
    };
    return [storedValue, setValue];
  };
```

frontend\src\components\shared\tags\mm_base_tag.js

```
import { Chip, Grid2 } from "@mui/material"
```

```
import { Children, cloneElement, createContext, useContext, useEffect, useState } from "react"

export const MMBaseTag = ({ tag, icon, selectable = false, onSelectionChange, isSelected, valueTag = null }) => {
    const [selected, setSelected] = useState(isSelected)
    const [labelSelected, setLabelSelected] = useContext(SelectionContext)
    const hasContext = () => setLabelSelected !== undefined && setLabelSelected !== null

    const onSelectionChangeHandler = () => {
        if (selectable) {
            if (hasContext()) {
                if (labelSelected !== tag) {
                    setLabelSelected(tag)
                    onSelectionChange(valueTag == null ? tag : valueTag, true)
                }
            } else {
                setSelected(!selected)
            }
        }
    }

    const getVariant = () => {
        if (hasContext()) return labelSelected == tag ? "filled" : "outlined"

        return selected ? "filled" : "outlined"
    }

    return (
        <Chip
            label={tag}
            onClick={onSelectionChangeHandler}
            onDelete={() => { }}
            deleteIcon={icon}
            variant={getVariant()}
            sx={{
                fontSize: '12px',
                padding: '8px'
            }}
        />
    )
}
```

```
const SelectionContext = createContext([]);

export const MMTagGroup = ({ children, onSelectionChange }) => {
    const [selectedElementKey, setSelectedElementKey] = useState(null)

    const onSelectionHandler = (tag, _) => {
        onSelectionChange(tag)
    }

    return (
        <SelectionContext.Provider value={[selectedElementKey, setSelectedElementKey ]}>
            <Grid2 direction={'row'} container spacing={0} columns={{ xs: 2, sm: 2, md: 3, lg: 3, xl: 4 }}>

                {Children.map(children, child =>
                    <Grid2 padding={'8px'}> {cloneElement(child, {onSelectionChange: onSelectionHandler})} </Grid2>
                )}
            </Grid2>
        </SelectionContext.Provider>
    )
}
```

frontend\src\components\shared\tags\priority_tags.js

```
import { Chip } from "@mui/material"
import GroupsIcon from '@mui/icons-material/Groups';
import DeleteIcon from '@mui/icons-material/Delete';
import MonitorIcon from '@mui/icons-material/Monitor';
import WebIcon from '@mui/icons-material/Web';
import VerticalSplitIcon from '@mui/icons-material/VerticalSplit';
import SmartphoneIcon from '@mui/icons-material/Smartphone';
import TerminalIcon from '@mui/icons-material/Terminal';
import BugReportIcon from '@mui/icons-material/BugReport';

export const PriorityTag = ({priority, color}) => {
    return (
        <Chip color={color} label={priority} variant="outline" sx={{fontWeight: 'bold'}} />
    )
}

export const LowPriority = () => {
```

```
    return <PriorityTag priority={'Low'} color="primary"/>
}

export const MediumPriority = () => {
    return <PriorityTag priority={'Medium'} color="warning"/>
}

export const HighPriority = () => {
    return <PriorityTag priority={'High'} color="error"/>
}


export const getPriorityChip = priority => {
    var _componentMap = {
        'low': <LowPriority />,
        'medium': <MediumPriority />,
        'high': <HighPriority />,
    }

    if (Object.keys(_componentMap).includes(priority)) {
        return _componentMap[priority]
    }

    return <></>
}
```

frontend\src\components\shared\tags\project_tags.js

```
import { Chip } from "@mui/material"
import { useState } from "react"

export const ProjectTags = [
    'Nova Versão Site Cliente'
]

export const ProjectTag = ({ tag, backgroundColor, icon, variant, selectable = false }) => {
    const [selected, setSelected] = useState(false)
    return (
        <Chip
            label={tag}
            onClick={() => setSelected(!selected)}




            variant={selected || !selectable ? "filled" : "outlined"}
            sx={{
                color: '',
                fontSize: '12px',
                padding: '8px'
            }}
        />
    )
}
```

frontend\src\components\shared\tags\sprint_status.js

```
import { Chip } from "@mui/material";

export const InProgressLabel = (props) => {
  return (
    <Chip size="medium" label="In Progress" color="primary" {...props} />
  )
}
export const DoneLabel = (props) => {
  return (
    <Chip size="medium" label="Done" color="success" {...props} />
  )
}

```

frontend\src\components\shared\tags\task_status.js

```
// Todo, progress, review, testing, done

import { Chip } from "@mui/material"

export const ToDoTag = ({ }) => {
    return (
        <Chip label={'To Do'} variant="outlined" color=""/>
    )
}

export const ProgressTag = ({ }) => {
    return (
        <Chip label={'Progress'} variant="filled" color="primary"/>
```

```
    )
}

export const ReviewTag = ({ }) => {
    return (
        <Chip label={'Review'} variant="filled" color="warning"/>
    )
}

export const TestingTag = ({ }) => {
    return (
        <Chip label={'Testing'} variant="filled" color="warning"/>
    )
}

export const CompletedTag = ({ }) => {
    return (
        <Chip label={'Done'} variant="filled" color="success"/>
    )
}

```

frontend\src\components\shared\tags\task_tags.js

```
import { Chip } from "@mui/material"
import MonitorIcon from '@mui/icons-material/Monitor';
import WebIcon from '@mui/icons-material/Web';
import VerticalSplitIcon from '@mui/icons-material/VerticalSplit';
import SmartphoneIcon from '@mui/icons-material/Smartphone';
import TerminalIcon from '@mui/icons-material/Terminal';
import BugReportIcon from '@mui/icons-material/BugReport';
import { useEffect, useState } from "react";

export const TaskTags = [
    'Backend',
    'Frontend',
    'Mobile',
    'Desktop',
    'Infra',
    'Bug',
]



export const TaskTag = ({ tag, backgroundColor, icon, variant, selectable = false, onSelectionChange }) => {
    const [selected, setSelected] = useState(false)

    const onSelectionChangeHandler = () => {
        if (selectable) {
            setSelected(!selected)
        }
    }

    useEffect(() => {
        if (selectable) onSelectionChange(selected)
    }, [selected])

    return (
        <Chip
            label={tag}
            onClick={onSelectionChangeHandler}
            onDelete={() => { }}
            deleteIcon={icon}
            variant={selected || !selectable ? "filled" : "outlined"}
            sx={{
                color: '',
                fontSize: '12px',
                padding: '8px'
            }}
        />
    )
}

export const FrontendTag = (props) => {
    return <TaskTag tag={'Frontend'} {...props} icon={<VerticalSplitIcon />} />
}

export const BackendTag = (props) => {
    return <TaskTag {...props} tag={'Backend'} icon={<WebIcon />} />
}

export const MobileTag = (props) => {
    return <TaskTag {...props} tag={'Mobile'} icon={<SmartphoneIcon />} />
}
```

```

export const InfraTag = (props) => {
    return <TaskTag {...props} tag={'Infra'} icon={<TerminalIcon />} />
}

export const DesktopTag = (props) => {
    return <TaskTag {...props} tag={'Desktop'} icon={<MonitorIcon />} />
}

export const BugTag = (props) => {
    return <TaskTag {...props} tag={'Bug'} icon={<BugReportIcon />} />
}

export const getTaskTagByName = (name, selectable = false, props = {}) => {
    var _componentMap = {
        'Backend': <BackendTag selectable={selectable} {...props} />,
        'Frontend': <FrontendTag selectable={selectable} {...props} />,
        'Mobile': <MobileTag selectable={selectable} {...props} />,
        'Desktop': <DesktopTag selectable={selectable} {...props} />,
        'Infra': <InfraTag selectable={selectable} {...props} />,
        'Bug': <BugTag selectable={selectable} {...props} />,
    }

    if (Object.keys(_componentMap).includes(name)) {
        return _componentMap[name]
    }

    return <></>
}

export const getTaskTagByNameComponent = (name) => {
    var _componentMap = {
        'Backend': BackendTag,
        'Frontend': FrontendTag,
        'Mobile': MobileTag,
        'Desktop': DesktopTag,
        'Infra': InfraTag,
        'Bug': BugTag,
    }

    if (Object.keys(_componentMap).includes(name)) {
        return _componentMap[name]
    }

    return <></>
}
```

frontend\src\components\shared\tags\team_tag.js

```
import { Chip } from "@mui/material"
import GroupsIcon from '@mui/icons-material/Groups';
import DeleteIcon from '@mui/icons-material/Delete';
import { MMBaseTag } from "./mm_base_tag";

export const TEAMS = ['Frontend', 'Backend', 'Embedded', 'Web team']

const _defaultProps = {
    icon: <GroupsIcon color="error" />
}

export const TeamTag = ({ teamName, backgroundColor, key }) => {
    return (
        <MMBaseTag label={teamName} key={teamName} onDelete={() => { }} variant="outline" sx={{ color: '#333', fontSize: '12px', padding: '8px' }} />
    )
}

export const RemoteTeamTag = (props) => {
    const tag = 'Remote'
    return <MMBaseTag tag={tag} key={tag} {...props} />
}

export const WebTeamTag = (props) => {
    const tag = 'Web team'
    return <MMBaseTag tag={tag} key={tag} {..._defaultProps} {...props} />
}

export const FrontendTeamTag = ({props}) => {
    const tag = 'Frontend'
    return <MMBaseTag tag={tag} key={tag} {..._defaultProps} {...props} />
}

export const BackendTeamTag = (props) => {
```

```
    const tag = 'Backend'
    return <MMBaseTag tag={tag}  {..._defaultProps} {...props} />
}

export const EmbeddedTeamTag = (props) => {
    const tag = 'Embedded'
    return <MMBaseTag tag={tag}  {..._defaultProps} {...props} />
}

export const QATeamTag = (props) => {
    const tag = 'QA'
    return <MMBaseTag tag={tag} {..._defaultProps} {...props} />
}

export const getTeamTagByName = (name, selectable = false, onSelectionChange = () => { }) => {
    var _componentMap = {
        'Web team': <WebTeamTag selectable={selectable} onSelectionChange={onSelectionChange} />,
        'Frontend': <FrontendTeamTag selectable={selectable} onSelectionChange={onSelectionChange} />,
        'Backend': <BackendTeamTag selectable={selectable} onSelectionChange={onSelectionChange} />,
        'Embedded': <EmbeddedTeamTag selectable={selectable} onSelectionChange={onSelectionChange} />,
        'QA': <QATeamTag />,
        'Time 1': <WebTeamTag />,
        'Time 2': <FrontendTeamTag />,
        'Time 3': <BackendTeamTag />,
    }

    if (Object.keys(_componentMap).includes(name)) {
        return _componentMap[name]
    }

    return <></>
}
```

frontend\src\components\shared\mm_circular_progress.js

```
import { Box, styled } from "@mui/material"
import { useEffect, useState } from "react"
import { PieChart, useDrawingArea } from "@mui/x-charts"

import { pieArcLabelClasses, pie } from "@mui/x-charts"


const StyledText = styled('text')(({ theme }) => ({
    // fill: theme.palette.text.primary,
    textAnchor: 'middle',
    dominantBaseline: 'central',
    fontSize: 48,
    fontWeight: "bold",
    fontFamily: 'Roboto',
    color: "red"
}));

function PieCenterLabel({ children }) {
    const { width, height, left, top } = useDrawingArea();
    return (
        <StyledText x={left + width / 2} y={top + height / 2}>
            {children}
        </StyledText>
    );
}

export const MMCircularProgressCentered = ({ charWidth, progress, color }) => {
    const { width, height, left, top } = useDrawingArea();
    return (
        <Box display="flex" boxSizing="border-box" width="600px">
            <Box width={`${600 - width / 2}px`} />
            <Box display="flex" alignContent="end" alignItems="center">
                <MMCircularProgress progress={progress} color={color} radius="30" width />
            </Box>
        </Box>
    )
}

export const MMCircularProgress = ({ radius, progress, color }) => {
    const [count, setCount] = useState(0)
    const [animationFinished, setAnimationFinished] = useState(false)
    useEffect(() => {
        const interval = setInterval(() => {
            if (count < progress) {
                setCount((count) => count + 1)
            } else {
                clearInterval(interval)
            }
```

```
        }, 10)

        return () => {
            clearInterval(interval)
            setAnimationFinished(true)
        }
    }, [count])

    return (
        <Box bgcolor="" display="flex" alignItems="center" justifyItems="center">
            <PieChart
                series={[
                    {
                        data: [{ id: 'key2', value: animationFinished ? progress : count, color: color }, { id: 'key', value: animationFinished ? 100 - progress : 100 - count, color: '#CFD8DC' }],
                        highlightScope: { fade: 'global', highlight: 'item' },
                        faded: { innerRadius: 120, additionalRadius: -10, color: 'gray' },
                        innerRadius: 120
                    },
                ]}
                slotProps={{
                    legend: {
                        hidden: false,
                        direction: 'column',
                        position: { vertical: "bottom", horizontal: "bottom" },
                        markGap: 0,
                        labelStyle: {
                            fontSize: 0,
                            display: 'none'
                        },
                        itemMarkWidth: 0,
                        itemMarkHeight: 0,
                        padding: 0,
                        itemGap: 0,


                    },
                }}
                sx={{
                    [`& .${pieArcLabelClasses.root}`]: {
                        fontWeight: 'bold',




                        fill: 'white',
                    },
                }}

                width={500}
                height={500}
            >
                <PieCenterLabel>{animationFinished ? progress : count}%</PieCenterLabel>
            </PieChart>

        </Box>
    );
}

```

frontend\src\components\shared\mm_date_range_picker.js

```
import { DatePicker } from '@mui/x-date-pickers-pro'
import * as React from 'react'
import { LocalizationProvider } from '@mui/x-date-pickers-pro/LocalizationProvider'
import { AdapterDayjs } from '@mui/x-date-pickers-pro/AdapterDayjs'
import { Box, Typography } from '@mui/material'
import dayjs, { Dayjs } from 'dayjs'
import { useState } from 'react'
import { useEffect } from 'react'

const _defaultBeginDate = dayjs('2024-01-01')
const _defaultEndDate = dayjs(Date.now())
const _slotProps = { textField: { size: 'small' } }
const Separator = () => <Box width={'16px'} />

export default function MMDateRangePicker({ updateHandler, direction = 'row' }) {

  const [millisecondsSinceEpochStart, setMillisecondsSinceEpochStart] = useState((new Date(2024, 0, 1)).getTime())
  const [millisecondsSinceEpochEnd, setMillisecondsSinceEpochEnd] = useState((new Date()).getTime())

  const onBeginRangeInputHandler = (value) => {
    const date = new Date(value.$d)
    if (date.getTime() <= Date.now()) {
      setMillisecondsSinceEpochStart(date.getTime())
    }
```

```
  }

  const onEndRangeInputHandler = (value) => {
    const date = new Date(value.$d)
    if (date.getTime() >= millisecondsSinceEpochStart) {
      setMillisecondsSinceEpochEnd(date.getTime())
    }
  }

  useEffect(() => {
    updateHandler(millisecondsSinceEpochStart, millisecondsSinceEpochEnd)
  }, [millisecondsSinceEpochStart, millisecondsSinceEpochEnd])

  return (
    <Box display="flex" alignItems={'center'} flexDirection={direction}>
      <Box display="flex" width="100%" flex="1" alignItems="center">
        <Box display="flex" flex="1">
          <Typography variant="body1">From:</Typography>
        </Box>
        <LocalizationProvider
          dateAdapter={AdapterDayjs} localeText={'From'}>
          <DatePicker onChange={onBeginRangeInputHandler} maxDate={dayjs(Date.now())} defaultValue={_defaultBeginDate} slotProps={_slotProps} />
        </LocalizationProvider>
      </Box>
      <Box height="16px" width="16px"/>
      <Box display="flex" width="100%" flex={1} alignItems="center">
        <Box display="flex" width="100%" flex={1} alignItems="center">
          <Box display="flex" flex="1">
            <Typography variant="body1">To:</Typography>
          </Box>
        </Box>
        <Separator />
        <LocalizationProvider dateAdapter={AdapterDayjs} localeText="From">
          <DatePicker onChange={onEndRangeInputHandler} minDate={dayjs(millisecondsSinceEpochStart)} defaultValue={_defaultEndDate} slotProps={_slotProps}
/>
        </LocalizationProvider>
      </Box>
    </Box>
  )
}
```

frontend\src\components\shared\mm_paper.js

```
import { Paper, styled } from "@mui/material";

export const MMPaper = styled(Paper, {
    name: "StyledPaper",
    slot: "Wrapper",
    shouldForwardProp: (prop) => {
        return true // return prop !== "elevation";
    }
})(({width = '100%', height = '100%', color = '#6B8068', padding = '0px', background = 'white'}) => ({
    color: color,
    margin: "auto",
    borderRadius: 2,
    height: height,
    width: width,
    display: "flex",
    padding: padding,
    background: background,
    // ".MuiButton-root": { color: "#FF0000" }
}));
```

frontend\src\components\shared\mm_select.js

```
import { FormControl, InputLabel, MenuItem, Select } from "@mui/material"

export const MMSelect = ({ list, onChangeValue, placeholder, value = null }) => {
    return (<FormControl fullWidth>
        <InputLabel id="time-input-label" size="small">{placeholder}</InputLabel>
        <Select
            size="small"
            labelId="time-label"
            id="time-select"
            // label="Sprints"
            // defaultValue={list[0]}
            // value="days"
            onChange={(event) => { onChangeValue(event.target.value) }}
        >
            {
                list.map((item) => <MenuItem value={value == null ? item.toLowerCase() : value}>{item}</MenuItem>)
            }
```

```
        </Select>
    </FormControl>)
}
```

frontend\src\components\shared\simple_lable.js

```
import { Chip } from "@mui/material"
import { useState } from "react"

export const FloatingLabel = ({ right, top, color, text }) => {
    return (
        <Chip size="medium" sx={{ position: 'absolute', top: top, right: right }} label={text} color={color} />
    )

}

export const Label = ({ color, text }) => {
    return (
        <Chip size="medium" label={text} color={color} />
    )
}

export const ToggleLabel = ({active = false, text, clickHandler}) => {
    const [isActive, setIsActive] = useState(active)
    const variant = isActive ? "filled" : "outlined"

    const _clickHandler = () => {
        clickHandler(!isActive)
        setIsActive(!isActive)
    }

    return <Chip key={Math.random()} size="medium" variant={variant} color="primary" label={text} clickable={true} onClick={_clickHandler} />
}


```

frontend\src\components\sprint_card\index.js

```
import { Alert, Box, Button, Card, CardActions, CardContent, Container, LinearProgress, Popover, Typography } from "@mui/material"
import { Link } from "react-router-dom";
import { DoneLabel, InProgressLabel } from "../shared/tags/sprint_status";




import { useState } from "react";

const sxLabel = { position: 'absolute', top: 16, right: 8 }

const ChangeFeatureNotAvailablePopover = ({ }) => {
  const [anchorEl, setAnchorEl] = useState(null);
  const open = Boolean(anchorEl);
  const id = open ? 'simple-popover' : undefined;

  return (
    <>
      <Button
        size="medium"
        onMouseOver={(e) => setAnchorEl(e.currentTarget)}
        onClick={(e) => e.preventDefault()}>
        <Link style={{ color: 'inherit', textDecoration: 'inherit' }}
          to={"133/metrics"}>See Metrics
        </Link>
      </Button>

      <Popover
        id={id}
        open={open}
        anchorEl={anchorEl}
        onClose={() => setAnchorEl(null)}
        transformOrigin={{
          vertical: 'bottom',
          horizontal: 'left',
        }}>
        <Alert severity="warning">This feature is not implemented yet.</Alert>
      </Popover>
    </>
  )
}

const SprintCard = ({
  title, code, beginDate, endDate, pointsDone, pointsTotal, description, descriptionMatchingIndexes, titleMatchingIndexes, teamName, done = false
}) => {

  const progress = Math.floor((pointsDone / pointsTotal) * 100)

```

```
  return (
    <Card sx={{ height: 300, paddingTop: '32px', position: 'relative' }}>
      {done ? <DoneLabel sx={sxLabel} /> : <InProgressLabel sx={sxLabel} />}
      <CardContent sx={{ height: '80%' }}>
        <Typography variant="h5" component="div">
          {
            titleMatchingIndexes.map((info) => {
              const [[indexBegin, indexEnd], shouldHighligth] = info
              const color = shouldHighligth ? 'darkGreen' : ''
              const fontWeigth = shouldHighligth ? 'bold' : 'normal'
              return <span style={{ color: color, fontWeight: fontWeigth }}>{title.substring(indexBegin, indexEnd)}</span>
            })
          }
        </Typography>
        <Typography sx={{ color: 'text.secondary', mb: 1.5 }}>{pointsDone}/50 points completed</Typography>
        <Typography variant="body2">

          {
            descriptionMatchingIndexes.map((info) => {
              const [[indexBegin, indexEnd], shouldHighligth] = info
              const color = shouldHighligth ? 'darkGreen' : ''
              return <span style={{ color: color }}>{description.substring(indexBegin, indexEnd)}</span>
            })
          }
        </Typography>

        <Container sx={{ flex: 1, height: '16px' }} />
        <LinearProgress variant="determinate" value={progress} />
      </CardContent>
      <Container sx={{ flex: 1, height: 'auto' }} />
      <CardActions>
        <ChangeFeatureNotAvailablePopover>

        </ChangeFeatureNotAvailablePopover>
        <Box flex={1} />
        <Button size="medium"><Link style={{ color: 'inherit', textDecoration: 'inherit' }} to={`${code}/tasks`}>See Tasks</Link></Button>
      </CardActions>
    </Card>
  )
}



export default SprintCard
```

frontend\src\pages\home\about\about_page.js

```
import { Box, Typography } from "@mui/material"

export const AboutPage = ({ }) => {


    return (
        <Box display="flex" flexDirection="column" sx={{height: `calc(100vh - ${80}px - 80px)`}}>
            <Typography variant="h5">About the Application</Typography>
            <Box height="32px" />
            <Box display="flex" flexDirection="column">
                <Typography variant="body1" sx={{ wordWrap: "break-word" }}>
                    Metrics Insight is a web application with focus on visualization and implementation of agile metrics and customized
                    ones in the context of agile teams in software companies.
                </Typography>
                <Box height="16px" />
                <Typography>
                    The aggregation of different metrics from different units of the organization, like teams, sprints or even individual metrics
                    allows to detect points of failure in different parts of the project/organization.
                </Typography>
                <Typography>
                    Management teams can greatly benefit from this since these indicators guide decisions at every aspect of software development.
                </Typography>
                <Box height="16px" />
                <Typography variant="body1">
                    This project was developed as part of the final thesis for the Bachelor's degree in Computer Science at the Federal University of
                    Santa Catarina (UFSC).
                </Typography>
            </Box>
            <Box flex={1} />
            <Box display={"flex"} flexDirection="column">
                <Typography>Autor: Wesley Mayk</Typography>
                <Typography>Versão: v0.0.1</Typography>
                <Typography>05/11/2024</Typography>
            </Box>
        </Box>
```

```
    )
}
```

frontend\src\pages\home\account\account_page.js

```
import { Alert, Box, Divider, Popover, Typography } from "@mui/material"
import { useContext, useEffect, useState } from "react"
import { HomeController } from "../controller/home_controller"
import { useAuth } from "../../../state/providers/auth_provider"
import DrawerContext from "../home/drawer_context"
import { NoPhotoEmployee } from "../../../components/shared/no_photo_employee/no_photo_user"
import { Link } from "react-router-dom"


const ChangeFeatureNotAvailablePopover = ({ }) => {
    const [anchorEl, setAnchorEl] = useState(null);
    const open = Boolean(anchorEl);
    const id = open ? 'simple-popover' : undefined;

    return (
        <>
            <Link href="/" underline="none" onMouseOver={(e) => setAnchorEl(e.currentTarget)} onClick={(e) => e.preventDefault()}>Change</Link>
            <Popover
                id={id}
                open={open}
                anchorEl={anchorEl}
                onClose={() => setAnchorEl(null)}
                transformOrigin={{
                    vertical: 'bottom',
                    horizontal: 'left',
                }}>
                    <Alert severity="warning">Feature not available due integration to Jira.</Alert>
            </Popover>
        </>
    )
}

export const AccountPage = () => {

    const [userInfo, setUserInfo] = useState(null)




    const [infoLoaded, setInfoLoaded] = useState(false)
    const isDrawerOpen = useContext(DrawerContext)
    const auth = useAuth()
    const homeController = new HomeController()

    useEffect(() => {
        const fetchBasicInfo = async () => {
            const info = await homeController.getEmployeeInfo(auth.authState.email)
            setUserInfo(info)
            setInfoLoaded(true)
        }
        fetchBasicInfo()
    }, [])

    const getMainBoxDisplacement = () => isDrawerOpen ? '32px' : '0px'

    return (
        <Box width="100%" height="100%" display="flex" flexDirection="column" paddingLeft={getMainBoxDisplacement()}>
            <Typography variant="h4">Your Account</Typography>
            <Box height="24px" />
            <Divider />
            <Box height="36px" />
            <Box display="flex">
                <Box display="flex" flexDirection={'column'} flex={1}>
                    <Typography variant="h5" fontWeight={'bold'} color="#666">Private Info</Typography>
                    <Box height={'24px'} />
                    <Typography variant="h6">Email</Typography>
                    <Box display="flex">
                        <Typography variant="body1">{infoLoaded ? userInfo.email : ''}</Typography>
                        <Box width={'8px'} />
                        <ChangeFeatureNotAvailablePopover />
                    </Box>
                    <Box height={'24px'} />
                    <Typography variant="h6">Password</Typography>
                    <Box display="flex">
                        <Typography variant="body1">{infoLoaded ? '**********' : ''}</Typography>
                        <Box width={'8px'} />
                        <ChangeFeatureNotAvailablePopover />
                    </Box>
                    <Box height={'18px'} />
                </Box>
```

```
                <Box display="flex" flexDirection={'column'} flex={1}>
                    <Typography variant="h5" fontWeight={'bold'} color="#666">Profile</Typography>
                    <Box height={'24px'} />
                    <Typography variant="h6">Name</Typography>
                    <Typography variant="body1">{infoLoaded ? userInfo.name : ''}</Typography>
                    <Box height={'24px'} />
                    <Typography variant="h6">Position</Typography>
                    <Typography variant="body1">{infoLoaded ? userInfo.position : ''}</Typography>
                    <Box height={'24px'} />
                    <Typography variant="h6">Team</Typography>
                    <Typography variant="body1">{infoLoaded ? userInfo.teamName : ''}</Typography>
                    <Box height={'24px'} />
                    <Typography variant="h6">Last Sprints</Typography>
                    <Typography variant="body1">{infoLoaded ? userInfo.lastSprints.map((sprint) => <Link to={`../sprints/${sprint}/tasks`} >{sprint}
</Link>) : ''}</Typography>
                    <Box height={'24px'} />
                </Box>
                <Box display="flex" alignItems={'flex-start'} flex={1}>
                    <NoPhotoEmployee />
                </Box>
            </Box>
        </Box>
    )
}
```

frontend\src\pages\home\controller\router\protected_router.js

```
import { Navigate } from "react-router-dom"
import { useAuth } from "../../../../state/providers/auth_provider"

export const ProtectedRouter = ({ children }) => {
    const { authState, setAuthState, ready } = useAuth()
    const { accessToken } = authState

    if (accessToken === null || accessToken === undefined || accessToken === '' && ready) {
        return <Navigate to={'/'} />
    }

    return (
        <>
            {children}




        </>
    )
}
```

frontend\src\pages\home\controller\home_controller.js

```
import axios from "axios"
axios.defaults.withCredentials = true;


export class HomeController {
    constructor(axios) {
        this.axios = axios
    }

    static apiURL = 'http://host.docker.internal:8080'

    async authenticate(email, password) {
        try {
            const response = await axios.post(`${HomeController.apiURL}/login`, {
                'user': email,
                'password': password
            })

            return [response.data['token'], response.data['user']]
        } catch (error) {
            return 'unknown'
        }

    }

    async getSprint(sprintCode) {

        try {
            const response = await axios.get(`${HomeController.apiURL}/sprints/${sprintCode}`, {})

            return response.data['sprint']
        } catch (error) {
            console.log(error)

            return []
```

```
        }
    }

    async getSprints() {

        try {
            const response = await axios.post(`${HomeController.apiURL}/sprints/`, {})

            return response.data['sprints']
        } catch (error) {
            console.log(error)

            return []
        }
    }

    async getSprintPointsHistory(sprintCode) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/sprint/${sprintCode}/story-points-history`)

            return response.data['data']
        } catch(error) {
            console.log(error)

            return []
        }
    }

    async getWorkLoadHistoryFromSprint(sprintCode) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/sprint/${sprintCode}/work-scope-history`)
            return response.data['workloadHistory']
        } catch(error) {
            console.log(error)

            return []
        }
    }

    async getTasksFromSprint(sprintCode) {
        try {




            const response = await axios.get(`${HomeController.apiURL}/sprint/${sprintCode}/tasks`)
            return response.data['tasks']
        } catch(error) {
            console.log(error)

            return []
        }
    }

    async getWorkInProgressHistoryFromSprint(sprintCode) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/sprint/${sprintCode}/work-in-progres-history`)
            const history = response.data['workInProgressHistory']

            return history
        } catch (error) {
            console.log(error)
            return []
        }
    }


    async getTeamsGeneralInfo() {
        try {
            const response = await axios.get(`${HomeController.apiURL}/teams`)
            const teams = response.data['teams']

            return teams
        } catch (error) {
            return []
        }
    }

    async getTeamInfo(teamName) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/teams/${teamName}`)
            const history = response.data['info']

            return history
        } catch (error) {
            console.log(error)
```

```
            return []
        }
    }

    async getAllEmployees() {
        try {
            const response = await axios.get(`${HomeController.apiURL}/employee`)
            const history = response.data

            return history
        } catch (error) {
            console.log(error)
            return []
        }
    }

    async getEmployeeInfo(email) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/employee/${email}/basic-info`)
            const history = response.data['info']

            return history
        } catch (error) {
            console.log(error)
            return []
        }
    }

    async getEmployeeParticipationInSprint(email, sprintCode) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/employee/${email}/participation-report/sprint/${sprintCode}`)
            return response.data['participation']
        } catch (error) {
            console.log(error)
            return {}
        }
    }

    async getEmployeePerformanceReport(email, projects, tags, range, granularity) {
        try {
            // const params = {
            //     'projects': projects,
            //     'tags': tags,
            //     'range':  { ...range },
            //     'granularity': granularity,
            // }
            // console.log(`params`)
            // console.log(params)
            // const response = await axios.post(`${HomeController.apiURL}/employee/${email}/performance-report`, {...params})
            // return response.data
            console.log(tags)
            // granularity: sprints
            if (tags.includes('Backend') && !tags.includes('Frontend')) {


                return {
                    "SP-001": 5,
                    "SP-002": 8,
                    "SP-003": 7,
                    "SP-004": 4,
                    "SP-005": 7,
                }
            }

            if (!tags.includes('Backend') && tags.includes('Frontend')) {

                return {
                    "SP-001": 1,
                    "SP-002": 3,
                    "SP-003": 2,
                    "SP-004": 5,
                    "SP-005": 7,
                }
            }

            if (tags.includes('Backend') && tags.includes('Frontend')) {
                return {
                    "SP-001": 6,
                    "SP-002": 11,
                    "SP-003": 9,
                    "SP-004": 9,
                    "SP-005": 14,
```

```
                }
            }

            return {}

        } catch (error) {
            console.log(error)
            return {}
        }
    }

    async getBugsPerClient(start, end) {
        try {
            const response = await axios.get(
                `${HomeController.apiURL}/clients/bugs-counts?millisecondsSinceEpochBegin=${start}&millisecondsSinceEpochEnd=${end}`)
            console.log(response.data)
            return response.data
        } catch (error) {
            console.log(error)
            return {}
        }
    }

    // Projects
    async getProjects() {
        try {
            const response = await axios.get(`${HomeController.apiURL}/projects`)
            return response.data
        } catch (error) {
            return {}
        }
    }

    async getTasksFromProject(projectId) {
        try {
            const response = await axios.get(`${HomeController.apiURL}/projects/${projectId}/tasks`)
            return response.data
        } catch (error) {
            return {}
        }
    }
```

```
}
```

frontend\src\pages\home\employee\charts\participation_pie_chart.js

```

import { pieArcLabelClasses, PieChart } from "@mui/x-charts"


export const ParticipationPieChart = ({ participationCategory, data, height, width }) => {

    const formatData = () => {
        if (data === null || data === undefined) return []
        const total = Object.values(data).reduce((accumulator, current) => accumulator + current, 0)

        return Object.keys(data).map((key) => { return { id: key, value: (data[key] * 100 / total).toFixed(2), label: key } })
    }

    const widthIncrease = () => {
        if (data === null || data === undefined) return 0

        const charsNumber = Object.keys(data).reduce((a, b) => {
            return a.length > b.length ? a : b;
        }).length;

        return charsNumber * 10
    }

    return (
        <PieChart
            series={[
                {
                    arcLabel: (item) => `${item.value}%`,
                    arcLabelMinAngle: 1,
                    arcLabelRadius: 150,
                    data: formatData(),
                    highlightScope: { fade: 'global', highlight: 'item' },
                    faded: { innerRadius: 50, additionalRadius: -10, color: 'gray' },
                },
            ]}
            slotProps={{
```

```
                legend: {
                    hidden: false,
                    direction: 'column',
                    position: { vertical: "", horizontal: 'right' },
                    markGap: 10,
                    labelStyle: {
                        fontSize: 12
                    },
                    itemMarkWidth: 20,
                    itemMarkHeight: 20,
                },
            }}
            sx={{
                [`& .${pieArcLabelClasses.root}`]: {
                    fontWeight: 'bold',
                    fill: 'white',
                },
            }}

            width={width+widthIncrease()}
            height={height}
        />
    )
}
```

frontend\src\pages\home\employee\charts\performance_chart.js

```
import { LineChart } from "@mui/x-charts"
import { defaultDateFormatter, sprintXAxisFormater } from "../../../../components/metrics/charts/date_formatters"
import { Box, Divider, FormControl, Grid2, Typography } from "@mui/material"
import { getTaskTagByName, TaskTag, TaskTags } from "../../../../components/shared/tags/task_tags"
import { MMPaper } from "../../../../components/shared/mm_paper"
import { TimeGranularitySelect, WorkloadGranularitySelect } from "../components/select"
import { ProjectTags } from "../../../../components/shared/tags/project_tags"
import MMDateRangePicker from "../../../../components/shared/mm_date_range_picker"
import { useEffect, useState } from "react"
import { HomeController } from "../../controller/home_controller"
import { useParams } from "react-router-dom"
import { MMBaseTag } from "../../../../components/shared/tags/mm_base_tag"

export const Chart = ({ dataX, dataY, time = 'sprint', workload = 'points' }) => {



    const getScaleType = () => {
        if (time === 'sprint' || time == 'sprints')
            return 'band'
        if (time === 'day' || time == 'days')
            return 'time'
    }

    const getValueFormatter = () => {
        if (time == 'day' || time == 'days')
            return defaultDateFormatter
        if (time == 'sprints' || time == 'sprint')
            return (code) => code
    }

    return (
        <LineChart
            xAxis={[{ data: dataX, valueFormatter: getValueFormatter(), scaleType:  getScaleType()}]}
            yAxis={[
                { scaleType: 'linear' },
            ]}
            series={[
                {
                    data: dataY,
                    color: 'blue',
                    label: workload
                },
            ]}
            // width={'100%'}
            sx={{ flexGrow: 1 }}
            // height={300}
        />
    )
}

export const PerformanceChart = () => {
    const homeController = new HomeController()
    const { email } = useParams('email')

    const [chartData, setChartData] = useState({
        'xData': [],
```

```
        'yData': []
    })
    const [chartSettings, setChartSettings] = useState({
        'range': {
            'begin': '',
            'end': ''
        },
        'tags': [],
        'projects': [],
        'workloadGranularity': '',
        'timeGranularity': '',
    })

    const updateRangeHandler = (millisecondsSinceEpochStart, millisecondsSinceEpochEnd) => {
        setChartSettings((prev) => ({
            ...prev,
            'range': {
                'begin': millisecondsSinceEpochStart,
                'end': millisecondsSinceEpochEnd,
            }
        }))
    }

    const updateTagHandler = (tag, selectionStatus) => {
        const tags = chartSettings['tags']

        setChartSettings((prev) => ({
            ...prev,
            'tags': selectionStatus ? [...tags, tag] : tags.filter((t) => t !== tag)
        }))
    }

    const updateProjectHandler = (tag, selectionStatus) => {
        const projects = chartSettings['projects']

        setChartSettings((prev) => ({
            ...prev,
            'projects': selectionStatus ? [...projects, tag] : projects.filter((project) => project !== tag)
        }))

    }




    const updateTimeGranularity = (selectedGranularity) => {
        setChartSettings((prev) => ({
            ...prev,
            'timeGranularity': selectedGranularity,
        }))
    }

    const updateWorkloadGranularity = (selectedGranularity) => {
        setChartSettings((prev) => ({
            ...prev,
            'workloadGranularity': selectedGranularity,
        }))
    }

    useEffect(() => {
        const fetchPerformance = async () => {
            const { timeGranularity } = chartSettings
            const ret = await homeController.getEmployeePerformanceReport(
                email,
                chartSettings['projects'],
                chartSettings['tags'],
                chartSettings['range'],
                timeGranularity,
            )
            if (timeGranularity == 'days') {
                const sortedKeys = Object.keys(ret).sort()
                setChartData({
                    xData: sortedKeys.map((ts) => new Date(parseInt(ts))),
                    yData: sortedKeys.map((key) => ret[key]),
                })
            }
            if (timeGranularity == 'sprints') {
                setChartData({
                    xData: Object.keys(ret),
                    yData: Object.values(ret),
                })
            }
        }
        fetchPerformance()
    }, [chartSettings])
```

```
    return (<MMPaper elevation={5}>
        <Box width={'100%'} display={'flex'} padding={'32px'} maxHeight={'800px'}>
            <Box flex={1} >
                <Chart dataX={chartData.xData} dataY={chartData.yData} workload={chartSettings['workloadGranularity']} time={chartSettings.timeGranularity} />
            </Box>
            <Box width={'16px'} />
            <Divider orientation="vertical" />
            <Box width={'16px'} />
            <Box width={'500px'} display={'flex'} flexDirection={'column'} padding={''}>
                <Box display={'flex'} flexDirection={'column'}>
                    <Typography variant="h6">Select Sprints</Typography>
                    <Box height={'16px'} />
                    <MMDateRangePicker updateHandler={updateRangeHandler} />
                </Box>
                <Box height={'24px'} />
                <Box display={'flex'} flexDirection={'column'}>
                    <Typography variant="h6">Select Tags</Typography>
                    <Grid2 direction={'row'} container spacing={0} columns={{ xs: 4, sm: 8, md: 12 }}>
                        {TaskTags.map(tag => <Grid2 padding={'8px'}>{getTaskTagByName(tag, true, {
                            onSelectionChange: (selection) => {
                                updateTagHandler(tag, selection)
                            }
                        })}</Grid2>)}
                    </Grid2>
                </Box>
                <Box height={'24px'} />
                <Box display={'flex'} flexDirection={'column'}>
                    <Typography variant="h6">Select Projects</Typography>
                    <Grid2 direction={'row'} container spacing={0} columns={{ xs: 4, sm: 8, md: 12 }}>
                        {ProjectTags.map(tag => <Grid2 padding={'8px'}><TaskTag tag={tag} selectable={true}
                            onSelectionChange={(tag, selection) => {
                                updateProjectHandler(tag, selection)
                            }}
                        /></Grid2>)}
                    </Grid2>
                </Box>
                <Box height={'24px'} />
                <Box display={'flex'} flexDirection={'column'}>
                    <Typography variant="h6">Select Granularity</Typography>
                    <Box height={'16px'} />
                    <Box display={'flex'} flexDirection={'row'}>
                        <FormControl id="form" fullWidth>
                            <TimeGranularitySelect onValueChange={updateTimeGranularity} />
                        </FormControl>
                        <Box width={'64px'} />
                        <FormControl id="form" fullWidth>
                            <WorkloadGranularitySelect onValueChange={updateWorkloadGranularity} />
                        </FormControl>
                    </Box>
                </Box>
            </Box>
        </Box>
    </MMPaper>)
}
```

frontend\src\pages\home\employee\components\select.js

```
import { InputLabel, MenuItem, Select } from "@mui/material"

export const TimeGranularitySelect = ({onValueChange = (value) => {}}) => {
    return (
        <>
            <InputLabel id="time-input-label" size="small">Time</InputLabel>
            <Select
                size="small"
                labelId="time-label"
                id="time-select"
                label="Time"
                defaultValue={''}
                // value="days"
                onChange={(event) => onValueChange(event.target.value)}
            >
                <MenuItem value={"days"}>Days</MenuItem>
                <MenuItem value={"sprints"}>Sprints</MenuItem>
            </Select>
        </>
    )
}
```

```
export const WorkloadGranularitySelect = ({onValueChange = (value) => {}}) => {
    return (
        <>
            <InputLabel id="workload-label" size="small">Workload</InputLabel>
            <Select
                size="small"
                labelId="workload-label"
                id="workload-select"
                label="Time"
                defaultValue={''}
                onChange={(event) => onValueChange(event.target.value)}
            >
                <MenuItem value={'points'}>Points</MenuItem>
                <MenuItem value={'hours'}>Hours</MenuItem>
            </Select>
        </>
    )
}
```

frontend\src\pages\home\employee\employee_full_info_page.js

```
import { useParams } from "react-router-dom"
import { useEffect, useState } from "react"
import { getTaskTagByName, TaskTags } from "../../../components/shared/tags/task_tags"
import { HomeController } from "../controller/home_controller"
import { ParticipationPieChart } from "./charts/participation_pie_chart"
import { PerformanceChart } from "./charts/performance_chart"
import { wesleyProfileImageURL } from "../../../images/wesley_pic"

const { Box, Divider, Typography, Link } = require("@mui/material");

export const EmployeeFullInfoPage = () => {
    const homeController = new HomeController()
    const { email: employeeEmail } = useParams('email')
    const [employee, setEmployee] = useState(null)
    const [participationData, setParticipationData] = useState({})
    const [lastSprintCode, setLastSprintCode] = useState(null)

    useEffect(() => {
        const fetchEmployeeInfo = async () => {
            const employeeInfo = await homeController.getEmployeeInfo(employeeEmail)
            setEmployee(employeeInfo)
        }

        fetchEmployeeInfo(employeeEmail);
    }, []);

    useEffect(() => {
        if (employee === null) return;
        const fetchParticipationData = async () => {
            let participationState = {}

            for (let sprintCode of employee['lastSprints']) {
                const participation = await homeController.getEmployeeParticipationInSprint(employeeEmail, sprintCode)
                participationState[sprintCode] = participation
            }
            if ((employee['lastSprints'] || []).length > 0) {
                const index = employee['lastSprints'].length - 1
                setLastSprintCode(employee['lastSprints'][index])
            }
            setParticipationData(participationState)
        }

        fetchParticipationData()
    }, [employee])

    const componentOrNothing = (data, dataLabel, Component) => {
        return <Component width={500} height={400} data={data[dataLabel]} />
    }

    return (
        <Box display={'flex'} flexDirection={'column'} flex={1} marginTop="100px">
            <Box display={'flex'} flexDirection={'row'} width={'100%'}>
                <Box display={'flex'} flexDirection={'column'} flex={1} alignItems={'center'}>
                    <div style=
                        {{
                            height: '150px',
                            width: '150px',
                            borderRadius: '50%',
                            backgroundColor: 'red',
                            backgroundSize: 'cover',
                            backgroundImage: `url(${wesleyProfileImageURL})`
```

```
                    }}>
                </div>
                <Box height={'16px'} />
                <Typography variant="h6">{employee !== null ? employee.name : ''}</Typography>
            </Box>
            <Divider orientation="vertical" />
            <Box display={'flex'} flex={2} flexDirection={'column'} padding={'0 0px 0px 16px'} boxSizing={'border-box'} >
                <Box display={'flex'}>
                    <Box>
                        <Box height={'48px'}>
                            <Typography>Role: </Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography>Email: </Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography>Tags: </Typography>
                        </Box>
                    </Box>
                    <Box width={'64px'} />
                    <Box>
                        <Box height={'48px'}>
                            <Typography>{employee !== null ? employee.name : ''} </Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography>{employee !== null ? employee.email : ''}</Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography>{getTaskTagByName('Backend', false)} </Typography>
                        </Box>
                    </Box>
                    <Box width={'128px'} />
                    <Box>
                        <Box height={'48px'}>
                            <Typography>Team: </Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography>Last Sprints: </Typography>
                        </Box>
                    </Box>
                    <Box width={'64px'} />
```

```
                    <Box>
                        <Box height={'48px'}>
                            <Typography>Backend </Typography>
                        </Box>
                        <Box height={'48px'}>
                            <Typography><Link>FES-01</Link></Typography>
                        </Box>
                    </Box>
                </Box>
            </Box>
        </Box>
        <Box height="16px" />
        <Typography variant="h5" marginTop="24px">Participation Report:</Typography>
        <Box height="16px" />
        <Box display="flex" flexDirection="column" padding="24px">
            <Box display="flex" flexDirection="row">
                <Box display={'flex'} flexDirection="column" flex="1">
                    <Typography variant="h6">By Client: </Typography>
                    <Box display="flex" alignItems="center">
                        {componentOrNothing(participationData[lastSprintCode] || {}, 'participationByClient', ParticipationPieChart)}
                    </Box>
                </Box>
                <Box display={'flex'} flexDirection={'column'} flex={1}>
                    <Typography variant="h6">By Project: </Typography>
                    <Box display="flex" alignItems="end" justifyContent="center">
                        {componentOrNothing(participationData[lastSprintCode] || {}, 'participationByProject', ParticipationPieChart)}
                    </Box>
                </Box>
            </Box>
            <Box height={'64px'} />
            <Box flex={1} display={'flex'} flexDirection={'column'}>
                <Typography variant="h6">By Tags: </Typography>
                <Box display="flex" justifySelf="right" justifyContent="start" alignSelf="self-start" paddingRight="24px" >
                    {componentOrNothing(participationData[lastSprintCode] || {}, 'participationByTag', ParticipationPieChart)}
                </Box>
            </Box>
            <Box height={'64px'} />
            <Typography variant="h5">{employee !== null ? employee.name : ''}'s Performance</Typography>
            <Box height={'32px'} />
            <Box display={'flex'} flexDirection={'column'}>
                <PerformanceChart />
```

```
                </Box>
            </Box>
        </Box>
    );
}

```

frontend\src\pages\home\employee\employee_list_page.js

```
import { useNavigate, useParams } from "react-router-dom"
import { useEffect, useState } from "react"
import { getTaskTagByName, TaskTags } from "../../../components/shared/tags/task_tags"
import { HomeController } from "../controller/home_controller"
import { ParticipationPieChart } from "./charts/participation_pie_chart"
import { PerformanceChart } from "./charts/performance_chart"
import { getTeamTagByName } from "../../../components/shared/tags/team_tag"
import { wesleyProfileImageURL } from "../../../images/wesley_pic"

const { Box, Divider, Typography, Link, Grid2, Button } = require("@mui/material");

export const EmployeeListPage = () => {
    const homeController = new HomeController()
    const [employees, setEmployees] = useState([])

    useEffect(() => {
        const fetchEmployees = async () => {
            const employeeInfo = await homeController.getAllEmployees()
            setEmployees(employeeInfo)
            console.log(employeeInfo)
        }

        fetchEmployees();
    }, []);



    return (
        <Box width="100vw">

            <Grid2 container direction="row" spacing={1}>




                {employees.map(e => <Grid2><EmployeeCard name={e.name} email={e.email} teamName={e.teamName} position={e.position} lastSprints={e.lastSprints} /></Grid2>)}

            </Grid2>
        </Box>
    );
}

const EmployeeCard = ({ name, email, teamName, position, lastSprints }) => {
    const navigate = useNavigate()
    return (
        <Box height={'400px'} width="400px" display={'flex'} flexDirection={'column'} padding={'16px'} bgcolor="#efefefe0" borderRadius="8px">
            <Box display={'flex'} height={'50px'} alignItems={'center'} justifyContent={'space-between'} >
                <Typography variant="h6">{name}</Typography>
                <div style=
                    {{
                        height: '50px',
                        width: '50px',
                        borderRadius: '50%',
                        backgroundColor: 'red',
                        backgroundSize: 'cover',
                        backgroundImage: `url(${wesleyProfileImageURL})`
                    }}>

                </div>
            </Box>
            <Box height={'32px'} />
            <Box display={'flex'}>
                <Box display={'flex'} flexDirection={'column'} flex={1}>
                    <Box height={'48px'}>
                        <Typography>Role: </Typography>
                    </Box>
                    <Box height={'48px'}>
                        <Typography>Email: </Typography>
                    </Box>
                    <Box height={'48px'}>
                        <Typography>Team: </Typography>
                    </Box>
                    <Box height={'48px'}>
                        <Typography>Last Sprints: </Typography>
                    </Box>
```

```
                </Box>
                <Box display={'flex'} flexDirection={'column'} flex={1}>
                    <Box height={'48px'}>
                        <Typography>{position}</Typography>
                    </Box>
                    <Box height={'48px'}>
                        <Link><Typography>{email}</Typography></Link>
                    </Box>
                    <Box height={'48px'}>
                        {getTeamTagByName(teamName)}
                    </Box>
                    <Box height={'48px'}>
                        {lastSprints ? lastSprints.map((code) => <Link underline="hover">{code}</Link>) : <></>}
                    </Box>
                </Box>
            </Box>
            <Box >
                <Link>
                    <Button onClick={() => navigate(`${email}/full-metrics`)}>See metrics on {name}</Button>
                </Link>
            </Box>
        </Box>
    )
}
```

frontend\src\pages\home\metrics\bpc\bugs_per_client.js

```
import { Box, Container, Divider, FormControl, InputLabel, MenuItem, Select, Typography } from "@mui/material"
import { MMPaper } from "../../../../components/shared/mm_paper"

import { PieChart, pieArcLabelClasses } from "@mui/x-charts"

import MMDateRangePicker from "../../../../components/shared/mm_date_range_picker"
import { useEffect, useState } from "react"
import { HomeController } from "../../controller/home_controller"

export const WorkGranularitySelect = ({ onValueChange = (value) => { } }) => {
    return (
        <>
            <InputLabel id="time-input-label" size="small">Time</InputLabel>
            <Select
                size="small"
                labelId="time-label"
                id="time-select"
                label="Workload"
                defaultValue={''}
                onChange={(event) => onValueChange(event.target.value)}
            >
                <MenuItem value={"days"}>Points</MenuItem>
                <MenuItem value={"sprints"}>Hours</MenuItem>
            </Select>
        </>
    )
}

export const BugPieChart = ({ data }) => {
    const total = Object.values(data).map(e => e.value).reduce((a, b) => a + b, 0)
    return (
        <PieChart
            series={[
                {
                    arcLabel: (item) => `${item.value * 100 / total}%`,
                    data: data,
                    highlightScope: { fade: 'global', highlight: 'item' },
                    faded: { innerRadius: 50, additionalRadius: -10, color: 'gray' },
                },
            ]}
            slotProps={{
                legend: {
                    hidden: false,
                    direction: 'column',
                    position: { vertical: "top", horizontal: 'right' },
                    padding: 0,
                    labelStyle: {
                        fontSize: 16
                    },
                    itemMarkWidth: 20,
                    itemMarkHeight: 20,
                },
            }}
            sx={{
                [`& .${pieArcLabelClasses.root}`]: {
```

```
                    fontWeight: 'bold',
                    fill: 'white',
                },
            }}
            width={500}
            height={500}
        />
    )
}

export const BugsPerClient = () => {

    const homeController = new HomeController()
    const [data, setData] = useState([])
    const [range, setRange] = useState({ 'start': 1719792000000, 'end': 1733011200000 })
    const [workGranularity, setWorkGranularity] = useState('points')

    const transformer = (data) => {
        return Object.keys(data).map(key => ({ "label": key, "value": data[key], "id": key }))
    }

    useEffect(() => {
        const fetchData = async () => {
            const { start, end } = range
            const response = await homeController.getBugsPerClient(start, end)

            setData(transformer(response))
        }
        fetchData()
    }, [workGranularity, range])

    const dateRangeHandler = (milliEpochStart, milliEpochEnd) => setRange({ 'start': milliEpochStart, 'end': milliEpochEnd })
    const total = Object.values(data).map(e => e.value).reduce((a, b) => a + b, 0)


    return (
        <MMPaper width="90%" elevation={5}>
            <Box display="flex" flexDirection="row" width="100%" padding="32px">
                <Box display="flex" flexDirection="column" flexGrow={1}>
                    <Typography variant="h5">Bugs Per Client</Typography>
                    <Box height="16px"/>
```

```
                    <Box display="flex" alignItems="center">
                        <BugPieChart data={data} />
                    </Box>
                </Box>
                <Divider orientation="vertical" sx={{ margin: '0 16px 0px 8px' }} />
                <Box width={'400px'} display={'flex'} flexDirection={'column'}>
                    <Box display="flex" flexDirection="row" height={'fit-content'}>
                        <Box flex={1}>
                            <Typography variant="body1" fontSize={'16px'}>Total: </Typography>
                        </Box>
                        <Box flex={1}>
                            <Typography variant="body1" fontSize={'16px'}>{total} </Typography>
                        </Box>
                    </Box>
                    <Box height="16px" />
                    <FormControl fullWidth>
                        <WorkGranularitySelect onValueChange={(granularity) => setWorkGranularity(granularity)} />
                    </FormControl>
                    <Box height="16px" />
                    <MMDateRangePicker direction="column" updateHandler={dateRangeHandler} />
                </Box>
            </Box>
        </MMPaper>
    )
}
```

frontend\src\pages\home\metrics\burndown\burndown_page.js

```
import { BurndownChart } from "../../../../components/metrics/charts/burndown_chart"
import { useEffect, useState } from "react"
import { HomeController } from "../../controller/home_controller"
import { MetricBase } from "../metric_base_single_sprint"

const _transformer = (data) => {


    const {pointsHistory, totalPoints} = data
    const pointsHistoryList = Object.values(pointsHistory)
    const pointsDaysList = Object.keys(pointsHistory).map((ts) => new Date(parseInt(ts)))

    let currentPointsDone = 0
```

```
    let culmulativePointsDonePerDay = []
    for (let i = 0; i < pointsHistoryList.length; i++) {
        currentPointsDone += pointsHistoryList[i]
        culmulativePointsDonePerDay.push(currentPointsDone)
    }


    return {
        'culmulativePointsDonePerDay': culmulativePointsDonePerDay,
        'days': Object.keys(pointsHistory).map((ts) => new Date(parseInt(ts))),
        'totalPoints': totalPoints,
    }
}

export const BurndownPage = ({ sprintCode = 'FES-01' }) => {
    // const points = useState([3, 10, 20, 22, 24, 30, 35, 40, 44, 47, 49, 54, 57, 60])
    const [points, setSprintPoints] = useState([])
    const [days, setDays] = useState([])
    const sprintPoints = 65
    const progress = (points[points.length - 1] / sprintPoints) * 100
    const homeController = new HomeController()

    useEffect(() => {
        // async function loadHistory() {
        //     const pointsHistory = await homeController.getSprintPointsHistory(sprintCode)
        //     const pointsHistoryList = Object.values(pointsHistory)
        //     const pointsDaysList = Object.keys(pointsHistory).map((ts) => new Date(parseInt(ts)))

        //     let currentPointsDone = 0
        //     let temp = []
        //     for (let i = 0; i < pointsHistoryList.length; i++) {
        //         currentPointsDone += pointsHistoryList[i]
        //         temp.push(currentPointsDone)
        //     }

        //     setSprintPoints(temp)
        //     setDays(pointsDaysList)
        // }

        // loadHistory()
    }, [])




    async function loadHistory(sprintCode) {
        const pointsHistory = await homeController.getSprintPointsHistory(sprintCode)

        return pointsHistory
    }


    return (
        <MetricBase
            sprint={null}
            sprintCode={'FES-01'}
            metricName={'Burndown'}
            loadSelecteSprintInfo={loadHistory}
            transformer={_transformer}
            MetricComponent={BurndownChart}
        />
    )
}
```

frontend\src\pages\home\metrics\burndown\burnup_page.js

```
import { useEffect, useState } from "react"
import { HomeController } from "../../controller/home_controller"
import { MetricBase } from "../metric_base_single_sprint"
import { BurnUpChart } from "../../../../../components/metrics/charts/burnup_chart"
import { deltaListToCummulativeList } from "../../../../../components/metrics/charts/utils"
import { useParams } from "react-router-dom"




const _transformer = ([workloadHistory, pointsHistory]) => {
    return {
        'culmulativePointsDonePerDay': deltaListToCummulativeList(Object.values(workloadHistory).map((v) => parseInt(v))),
        'workloadHistory': deltaListToCummulativeList(Object.values(pointsHistory)),
        'days': Object.keys(workloadHistory).map((ts) => new Date(parseInt(ts)))
    }
}

```

```
export const BurnUpPage = () => {
    const homeController = new HomeController()

    async function loadHistory(sprintCode) {
        const workloadDiffs = (await homeController.getWorkLoadHistoryFromSprint(sprintCode))
        const points = (await homeController.getSprintPointsHistory(sprintCode))

        return [workloadDiffs, points['pointsHistory']]
    }

    return (
        <MetricBase
            sprint={null}
            sprintCode={'FES-01'}
            metricName={'Burnup'}
            loadSelecteSprintInfo={loadHistory}
            transformer={_transformer}
            MetricComponent={BurnUpChart}
        />
    )
}
```

frontend\src\pages\home\metrics\moving_average_velocity\moving_average_velocity_page.js

```
import { MovingAverageVelocityChart } from "../../../../components/metrics/customized/charts/moving_average_velocity_chart"
import { MetricBaseManyStrints } from "../metric_base_many_sprints"

const _fromSprintsToChartProps = (sprints) => {
    const commitedPointsHistory = sprints.map((sprint) => sprint.pointsTotal)
    const realizedPointsHistory = sprints.map((sprint) => sprint.pointsDone)

    return {
        'commitedPointsHistory': commitedPointsHistory,
        'realizedPointsHistory': realizedPointsHistory,
        'K': 5,
    }
}

export const MovingAverageVelocityPage = ({ }) => {

    return <MetricBaseManyStrints




        metricName={'Moving Average Velocity'}
        transformer={_fromSprintsToChartProps}
        MetricComponent ={MovingAverageVelocityChart}
    />
}
```

frontend\src\pages\home\metrics\velocity\velocity_page.js

```
import { Velocity } from "../../../../components/metrics/charts/velocity";
import { MetricBaseManyStrints } from "../metric_base_many_sprints";

const transformer = (sprints) => {
    const commitedPointsHistory = sprints.map((sprint) => sprint.pointsTotal)
    const realizedPointsHistory = sprints.map((sprint) => sprint.pointsDone)

    return {
        'commitedPointsFromLastKSprints': commitedPointsHistory,
        'realizedPointsFromLastKSprints': realizedPointsHistory,
        'sprintsNames': sprints.map((sprint) => sprint.title),
        'config': { width: 400, height: 600 }
    }
}
export const VelocityPage = () => {

    return (
        <MetricBaseManyStrints
            metricName={"velocity"}
            transformer={transformer}
            MetricComponent={Velocity}
        />
    )
}
```

frontend\src\pages\home\metrics\wip\wip_page.js

```
import { useParams } from "react-router-dom"
import { useEffect, useState } from "react"
import { HomeController } from "../../controller/home_controller"
import { LineChart } from "@mui/x-charts"
import { WIPChart } from "../../../../components/metrics/charts/work_in_progress_chart"
import { MetricBase } from "../metric_base_single_sprint"
```

```

const _transformer = (history) => {
    return {
        'workInProgressPointHistory': Object.values(history),
        'days': Object.keys(history).map((ts) => new Date(parseInt(ts)))
    }
}

export const WIPPage = ({ }) => {

    const { code } = useParams('code')
    const homeController = new HomeController()

    const fetchHistory = async (sprintCode) => {
        const history = await homeController.getWorkInProgressHistoryFromSprint(sprintCode)
        return history
    }


    return <MetricBase
        metricName={"Work In Progress"}
        sprintCode={code}
        loadSelecteSprintInfo={fetchHistory}
        transformer={_transformer}
        MetricComponent={WIPChart}
    />
}
```

frontend\src\pages\home\metrics\metric_base_many_sprints.js

```
import { Box, Container, Divider, Typography } from "@mui/material"
import { HomeController } from "../controller/home_controller"
import { useEffect, useState } from "react"
import MMDateRangePicker from "../../../components/shared/mm_date_range_picker"
import { MMPaper } from "./../../../components/shared/mm_paper";
import { MMBaseTag, MMTagGroup } from "../../../components/shared/tags/mm_base_tag"


export const MetricBaseManyStrints = ({ sprint, sprintCode, metricName, MetricComponent, transformer }) => {

    const [hasDataLoaded, setHasDataLoaded] = useState(false)




    const homeController = new HomeController()
    const [sprintsInfo, setSprintsInfo] = useState([])
    const [teams, setTeams] = useState([])
    const [filteredSprints, setFilteredSprints] = useState([])
    const [filterSettings, setFilterSettings] = useState({
        'dateRange': {
            'start': new Date(2024, 0, 1),
            'end': Date.now(),
        },
        'teams': []
    })

    const hasData = () => sprintsInfo !== null && sprintsInfo !== undefined && sprintsInfo.length > 0

    const onTeamSelectionChange = (tag, selectionStatus) => {

        setFilterSettings((prev) => ({
            ...prev,
            'teams': [tag]
        }))
    }

    const updateDateRangeHandler = (beginDateTimestamp, endDateTimestamp) => {
        setFilterSettings((prev) => ({
            ...prev,
            'dateRange': {
                'start': beginDateTimestamp,
                'end': endDateTimestamp,
            },
        }))
    }

    useEffect(() => {
        const fetchSprints = async () => {
            setSprintsInfo(await homeController.getSprints())
        }

        const fetchTeams = async () => {
            setTeams(await homeController.getTeamsGeneralInfo())
        }

```

```
        fetchSprints()
        fetchTeams()
    }, [])

    useEffect(() => {

        const filterByDateRange = (sprints) => {
            const { start, end } = filterSettings['dateRange']

            return sprints.filter((sprint) => sprint.beginDate >= start && sprint.endDate <= end)
        }

        const filterBySelectedTeams = (sprints) => {
            const { teams } = filterSettings
            const selectAll = teams.length === 0
            if (selectAll) return sprints

            return sprints.filter((sprint) => teams.includes(sprint.teamName))
        }

        let filtered = filterByDateRange(sprintsInfo);
        filtered = filterBySelectedTeams(filtered)

        setFilteredSprints(filtered)

    }, [filterSettings])

    console.log(filteredSprints)

    return (
        <>
            <MMPaper elevation={5}>
                <Box display={'flex'} flexDirection={'row'} width={'100%'} padding={'32px'}>
                    <Box flexGrow={1}>
                        {/* <Typography variant="h5">{metricName} - {hasData() ? sprintInfo.code : ''}</Typography> */}
                        <Container sx={{ height: '24px' }} />
                        <MetricComponent {...transformer(filteredSprints)} />
                    </Box>
                    <Divider orientation="vertical" sx={{ margin: '0 16px 0px 8px' }} />
                    <Box width={'400px'} display={'flex'} flexDirection={'column'}>


                        {/* header  */}
                        <Box width={'100%'} height={'50px'} padding={'8px 0 8px 0'} display={'flex'} alignItems={'center'} flexDirection={'row'}>
                            {/* <Typography variant="h4">{hasData() ? sprintInfo.code : ''}</Typography> */}
                            <Box flex={1}></Box>
                            <Divider orientation="vertical" sx={{ margin: '0 6px 0 6px', height: '90%', width: '10px' }} />
                            {/* {hasData() ? getTeamTagByName(sprintInfo.teamName) : <></>} */}
                        </Box>
                        <Box height={'32px'} />
                        <Box display={'flex'} flexDirection={'row'} height={'fit-content'}>
                            <Box flex={1}>
                                <Typography variant="body1" fontSize={'16px'}>Started: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Ended: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Status: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Commited Points: </Typography>
                            </Box>
                            <Box width="50px"></Box>
                            <Box flex={1} >
                                <Typography variant="body1">
                                    {hasData() ? new Date(sprintsInfo[0].beginDate).toLocaleDateString() : ''}
                                </Typography>
                                <Container sx={{ height: '8px' }} />

                                <Typography variant="body1">
                                    {hasData() ? new Date(sprintsInfo[sprintsInfo.length - 1].endDate).toLocaleDateString() : ''}
                                </Typography>
                                <Container sx={{ height: '8px' }} />

                                {/* <Chip label={'Done'} color={'success'} variant="outline" /> */}
                                Done
                                <Container sx={{ height: '8px' }} />
                                {/* <Typography variant="body1">{hasData() ? sprintInfo.pointsTotal : ''}</Typography> */}


                            </Box>
                        </Box>
                        {/* <Box height="50px"></Box>
                        <Typography color="textSecondary">
                            {hasData() ? `${sprintInfo.pointsDone}: points out of ${sprintInfo.pointsTotal} Completed` : ''}
```

```
                        </Typography> */}
                        {/* <Box height="10px"></Box>
                        <LinearProgress variant="determinate" value={formatPercentage()} /> */}
                        <Box height={'32px'} />
                        <Box display={'flex'} flexDirection={'column'}>
                            {/* <Grid2 direction={'row'} container spacing={0} columns={{ xs: 4, sm: 8, md: 12 }}>
                                {TEAMS.map(tag => <Grid2 padding={'8px'}>{getTeamTagByName(tag, true, onTeamSelectionChange)}</Grid2>)}
                            </Grid2> */}
                            <MMTagGroup onSelectionChange={onTeamSelectionChange}>
                                {teams.map(team => <MMBaseTag selectable tag={team.teamName} key={team.teamName} />)}
                            </MMTagGroup>
                            <Box height={'32px'} />
                            <MMDateRangePicker
                                updateHandler={updateDateRangeHandler}
                            >
                            </MMDateRangePicker>
                        </Box>

                    </Box>
                </Box>
            </MMPaper>
        </>
    )
}
```

frontend\src\pages\home\metrics\metric_base_single_sprint.js

```
import { Box, Chip, Container, Divider, FormControl, InputLabel, LinearProgress, MenuItem, Select, Typography } from "@mui/material"
import { getTeamTagByName, TEAMS } from "../../../components/shared/tags/team_tag"
import { HomeController } from "../controller/home_controller"
import { useEffect, useState } from "react"
import { MMPaper } from "../../../components/shared/mm_paper"
import { MMBaseTag, MMTagGroup } from "../../../components/shared/tags/mm_base_tag"

const ChipInfra = () => {
    return (
        <Chip label={'Infra'} variant="filled" sx={{ bgcolor: 'darkcyan', color: 'white' }} />
    )
}

const SprintSelect = ({ sprints, onChangeValue }) => {




    return (<>
        <InputLabel id="time-input-label" size="small">Sprints</InputLabel>
        <Select
            size="small"
            labelId="time-label"
            id="time-select"
            label="Sprints"
            defaultValue={''}
            // value="days"
            onChange={(event) => { onChangeValue(event.target.value) }}
        >
            {
                sprints.map((sprint) => <MenuItem value={sprint.code}>{sprint.code}</MenuItem>)
            }
        </Select>
    </>)
}

/**
 * relevant information: sprint code, pointsCompleted/pointsCommited and responsable team
 *
 */

export const MetricBase = ({ sprint, sprintCode, metricName, teamLabel, MetricComponent, loadSelecteSprintInfo, transformer }) => {

    const homeController = new HomeController()
    const [sprintInfo, setSprintInfo] = useState(null)
    const [teams, setTeams] = useState([])
    const [selectedSprintCode, setSelectedSprintCode] = useState(null)
    const [sprints, setSprints] = useState([])
    const [sprintsFiltered, setSprintsFiltered] = useState([])

    const [filterSetting, setFilterSetting] = useState({
        selectedTeams: []
    })

    const hasData = () => selectedSprintCode !== null && selectedSprintCode !== undefined
    const hasSprintData = () => sprintInfo !== null && sprintInfo !== undefined

    const formatPercentage = () => {
        if (!hasData()) {
```

```
            return 0
        }
        return (_selectedSprint().pointsDone * 100 / _selectedSprint().pointsTotal)
    }

    useEffect(() => {
        const fetchSprints = async () => {
            setSprints(await homeController.getSprints())
        }

        const fetchTeams = async () => {
            setTeams(await homeController.getTeamsGeneralInfo())
        }

        fetchSprints()
        fetchTeams()
    }, [])

    useEffect(() => {
        setSprintsFiltered(sprints.filter((sprint) => filterSetting.selectedTeams.includes(sprint.teamName)))
    }, [filterSetting])


    const onTeamSelectionChange = (tag, selectionStatus) => {
        setFilterSetting((prev) => ({
            ...prev,
            // 'selectedTeams': selectionStatus ? [...prev.selectedTeams, tag] : [...prev.selectedTeams.filter((team) => team !== tag)]
            selectedTeams: [tag]
        }))
    }

    const onSprintSelectionHandler = (sprintCode) => {
        const fetchSprint = async () => {
            const dataSprint = await loadSelecteSprintInfo(sprintCode)
            setSprintInfo(await loadSelecteSprintInfo(sprintCode))
            setSelectedSprintCode(sprintCode)
        }

        fetchSprint()
    }
```

```
    const _selectedSprint = () => sprintsFiltered.filter((sprint) => sprint.code == selectedSprintCode)[0]

    return (
        <>
            <MMPaper width={'90%'} elevation={5}>
                <Box display={'flex'} flexDirection={'row'} width={'100%'} padding={'32px'}>
                    <Box flexGrow={1} bgcolor="" maxHeight="700px">
                        <Typography variant="h5">{metricName} - {hasData() ? _selectedSprint().code : ''}</Typography>
                        <Container sx={{ height: '24px' }} />
                            {hasSprintData() ? <MetricComponent{...transformer(sprintInfo)} /> : <></>}
                    </Box>
                    <Divider orientation="vertical" sx={{ margin: '0 16px 0px 8px' }} />
                    <Box width={'400px'} display={'flex'} flexDirection={'column'}>
                        <Box width={'100%'} height={'50px'} padding={'8px 0 8px 0'} display={'flex'} alignItems={'center'} flexDirection={'row'}>
                            <Typography variant="h4">{hasData() ? _selectedSprint().code : ''}</Typography>
                            <Box flex={1}></Box>
                            <ChipInfra />
                            <Divider orientation="vertical" sx={{ margin: '0 6px 0 6px', height: '90%', width: '10px' }} />
                            {hasData() ? getTeamTagByName(_selectedSprint().teamName) : <></>}
                        </Box>
                        <Box height="32px" />
                        <Box display="flex" flexDirection="row" height={'fit-content'}>
                            <Box flex={1}>
                                <Typography variant="body1" fontSize={'16px'}>Started: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Ended: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Status: </Typography>
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1" fontSize={'16px'}>Commited Points: </Typography>

                            </Box>
                            <Box width="50px"></Box>
                            <Box flex={1} >
                                <Typography variant="body1">{hasData() ? new Date(_selectedSprint().beginDate).toLocaleDateString() : ''}</Typography>
                                <Container sx={{ height: '8px' }} />

                                <Typography variant="body1">{hasData() ? new Date(_selectedSprint().endDate).toLocaleDateString() : ''}</Typography>
                                <Container sx={{ height: '8px' }} />

                                {/* <Chip label={'Done'} color={'success'} variant="outline" /> */}
```

```
                                Done
                                <Container sx={{ height: '8px' }} />
                                <Typography variant="body1">{hasData() ? _selectedSprint().pointsTotal : ''}</Typography>


                            </Box>
                        </Box>
                        <Box height="64px" />
                        <Typography color="textSecondary">
                            {hasData() ? `${_selectedSprint().pointsDone}: points out of ${_selectedSprint().pointsTotal} Completed` : ''}
                        </Typography>
                        <Box height={'16px'} />
                        <LinearProgress variant="determinate" value={formatPercentage()} />
                        <Box height={'32px'} />
                        <MMTagGroup onSelectionChange={onTeamSelectionChange}>
                            {teams.map(team => <MMBaseTag selectable tag={team.teamName} key={team.teamName} />)}
                        </MMTagGroup>
                        <Box height="32px" />
                        <FormControl fullWidth>
                            <SprintSelect sprints={sprintsFiltered} onChangeValue={onSprintSelectionHandler} />
                        </FormControl>
                    </Box>
                </Box>
            </MMPaper>
        </>
    )
}

```

frontend\src\pages\home\project\project_card.js

```
import { Box, Button, Card, CardActions, CardContent, Container, Typography } from "@mui/material"
import { Link } from "react-router-dom";

const sxLabel = { position: 'absolute', top: 16, right: 8 }

const ProjectCard = ({ project }) => {

    const { title, id } = project
    console.log(project)




    return (
        <Card sx={{ height: 300, paddingTop: '32px', position: 'relative' }}>
            {/* {done ? <DoneLabel sx={sxLabel} /> : <InProgressLabel sx={sxLabel} />} */}
            <CardContent sx={{ height: '80%' }}>
                <Typography variant="h5" component="div">
                    {title}
                </Typography>
                <Typography variant="body2">
                    {/* ? */}
                </Typography>

                <Container sx={{ flex: 1, height: '16px' }} />
                {/* <LinearProgress variant="determinate" value={progress} /> */}
            </CardContent>
            <Container sx={{ flex: 1, height: 'auto' }} />
            <CardActions>
                <Button size="medium"><Link style={{ color: 'inherit', textDecoration: 'inherit' }} to={`${id}/tasks`}>See Tasks</Link></Button>
                <Box flex={1} />
            </CardActions>
        </Card>
    )
}


export default ProjectCard

```

frontend\src\pages\home\project\projects.js

```
import { Box, Grid2, Typography } from "@mui/material"
import { useEffect } from "react"
import ProjectCard from "./project_card"

export const Projects = ({ projects }) => {

    return (
        <>
            <Box sx={{ width: '100%' }} boxSizing={'border-box'} display={'flex'} flexDirection={'column'}>
                <Typography variant="h5">{Projects}</Typography>
                <Box height={'32px'} />
                <Grid2 boxShadow={'5px 5px 5px rgba(25, 118, 210, 0.1);'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }} direction={'row'} sx={{
```

```
                    marginTop: '32px', background: '', boxSizing: 'border-box'
                }}>

                    {
                        projects.map(project => <Grid2 size={3}><ProjectCard project={project} /></Grid2>)
                    }
                </Grid2>
            </Box>
        </>
    )
}
```

frontend\src\pages\home\project\task_list_page.js

```
/**
 *
 * Task - title, created_by, assigned_to, points,
 */

import { Box, Typography } from "@mui/material"
import { useParams } from "react-router-dom";
import { useEffect, useState } from "react";
import { HomeController } from "../controller/home_controller";
import { TaskAccordion } from "../sprints/tasks/tasks_list/task_accordion/task_accordion";

export const TaskListFromProject = ({ title }) => {

    const { projectId } = useParams('projectId')
    const [tasks, setTasks] = useState([])
    const homeController = new HomeController()

    useEffect(() => {
        const fetchTasks = async () => {
            const tasksList = await homeController.getTasksFromProject(projectId)
            setTasks(tasksList)
        }

        fetchTasks()
    }, [])

    return (




        <Box sx={{ width: '100%' }} boxSizing={'border-box'} display={'flex'} flexDirection={'column'}>
            <Typography variant="h5">Tasks</Typography>
            <Box height={'32px'} />
            {
                tasks.map(task => <TaskAccordion task={task} />)
            }
        </Box>
    )
}
```

frontend\src\pages\home\sprints\metrics\agile\agile_metrics.js

```
import { Box, Button, Card, CardActions, CardContent, Collapse, Fade, Grid2, IconButton, Modal, Typography, Zoom } from "@mui/material"
import { BurndownChart } from "../../../../../components/metrics/charts/burndown_chart"
import { useState } from "react"
import { BurnUpChart } from "../../../../../components/metrics/charts/burnup_chart"
import { Velocity } from "../../../../../components/metrics/charts/velocity"
import { MetricCard } from "../components/metric_card"
import { MovingAverageVelocityPage } from "../../../../../metrics/moving_average_velocity/moving_average_velocity_page"
import { MovingAverageVelocityChart } from "../../../../../components/metrics/customized/charts/moving_average_velocity_chart"
import { MockedBurndownChart, MockedBurnupChart, MockedVelocityChart } from "./mocked_charts/mocked_charts"
import { MockedWIPChart } from "../costume/charts/mocked"


const burndownChartDescription = 'A burndown chart is a graph that represents the work left to do versus the time it takes to complete it. It can be especially useful for teams working in sprints, as it can effectively show whether your deadlines are able to be met along the way.'
const burnupChartDescription = 'A burn up chart is a roadmap that plots your work on two lines along a vertical axis. One line indicates the entire workload for the project. The other depicts the work completed thus far. When you finish the project, the two lines meet.'
const velocityChartDescription = 'The velocity chart displays the average amount of work a scrum team completes during a sprint. Teams can use velocity to predict how quickly they can work through the backlog because the report tracks the forecasted and completed work over several sprints.'

export const Metrics = () => {
    return (
        <Box padding={'16px'} sx={{ width: '100%' }} boxSizing={'border-box'} >
            <Box display={'flex'} flexDirection={'column'}>
                <Typography variant="h4">Agile Metrics</Typography>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Sprint</Typography>
                    <Box height={'32px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
```

```
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Burndown Chart"}
                                description={burndownChartDescription}
                                chartMock={<MockedBurndownChart />}
                                metricURL={"../sprints/FES-01/metrics/burndown"}
                            />
                        </Grid2>
                        <Grid2 size={5}>
                            <MetricCard name={"Burnup Chart"}
                                description={burnupChartDescription}
                                chartMock={<MockedBurnupChart />}
                                metricURL={"../sprints/FES-01/metrics/burnup"}
                            />
                        </Grid2>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"WIP"}
                                description={velocityChartDescription}
                                chartMock={<MockedWIPChart />}
                                metricURL={"../sprints/FES-01/metrics/wip"}
                            />
                        </Grid2>
                    </Grid2>
                </Box>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Sprints</Typography>
                    <Box height={'32px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Velocity Chart"}
                                description={velocityChartDescription}
                                chartMock={<MockedVelocityChart />}
                                metricURL={"../sprints/FES-01/metrics/velocity"}
                            />
                        </Grid2>
                    </Grid2>
                </Box>
            </Box>




        </Box>
    )
}

```

frontend\src\pages\home\sprints\metrics\components\metric_card.js

```
import { Box, Button, Card, CardActions, CardContent, Collapse, Typography } from "@mui/material"
import { useState } from "react"
import { Link } from "react-router-dom"

export const MetricCard = ({ name, description, chartMock, metricURL }) => {
    const [isExpanded, setIsExpanded] = useState(false)
    return (
        <Card sx={{ background: '' }}>
            <CardContent>
                <Typography gutterBottom variant="h5" component="div" color="textSecondary">
                    {name}
                </Typography>
                <Typography gutterBottom variant="body1" component="div">
                    {description}
                </Typography>
                <CardActions>
                    <Button
                        size="medium"
                        color="primary"
                        aria-label="More Information"
                        onClick={() => setIsExpanded(!isExpanded)}
                    >
                        {/* <InfoIcon /> */}
                        See more
                    </Button>
                </CardActions>
                <Collapse orientation="vertical" in={isExpanded} collapsedSize={40}>
                    <Box height="48px" />
                    <Box display="flex" alignItems="center" justifyContent="center" height="500px" width="600px">
                        {isExpanded ? chartMock : <></>}
                    </Box>
                    <Button variant="contained">
                        <Link
                            style={{ color: 'inherit', textDecoration: 'inherit' }} to={metricURL}>
```

```
                        See Metric
                    </Link>
                </Button>
            </Collapse>
        </CardContent>
    </Card>
  )
}
```

frontend\src\pages\home\sprints\metrics\costume\costume_metrics.js

```
import { Box, Button, Card, CardActions, CardContent, Collapse, Grid2, Typography } from "@mui/material"
import { useState } from "react"
import { MovingAverageVelocityChart } from "../../../../../components/metrics/customized/charts/moving_average_velocity_chart"
import { Chart, PerformanceChart } from "../../../employee/charts/performance_chart"
import { MockBugsPerClient, MockedMovingAverageVelocityChart, MockedPerformanceChart } from "./charts/mocked"
import { MetricCard } from "../components/metric_card"
import { useAuth } from "../../../../../state/providers/auth_provider"


const velocityChartDescription = 'The velocity chart displays the average amount of work a scrum team completes during a sprint. Teams can use velocity to predict how quickly they can work through the backlog because the report tracks the forecasted and completed work over several sprints.'

export const CostumeMetrics = () => {
    const { authState } = useAuth()
    const { email } = authState

    return (
        <Box sx={{ width: '100%' }} boxSizing={'border-box'} >
            <Box display={'flex'} flexDirection={'column'}>
                <Typography variant="h4">Costume Metrics</Typography>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Sprint</Typography>
                    <Box height={'16px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                    </Grid2>
                </Box>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Sprints</Typography>
                    <Box height={'32px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Moving Average Velocity"}
                                description={velocityChartDescription}
                                chartMock={<MockedMovingAverageVelocityChart />}
                                metricURL={"../teams/infra/metrics/moving-average-velocity"}
                            />
                        </Grid2>
                    </Grid2>
                </Box>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Individual</Typography>
                    <Box height={'32px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Performance Chart"}
                                description={velocityChartDescription}
                                chartMock={<MockedPerformanceChart />}
                                metricURL={`../employees/${email}/full-metrics`}
                            />
                        </Grid2>
                    </Grid2>
                </Box>
                <Box height={'32px'} />
                <Box>
                    <Typography variant="h4">By Clients</Typography>
                    <Box height={'32px'} />
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Bugs Per Client"}
                                description={velocityChartDescription}
                                chartMock={<MockBugsPerClient />}
                                metricURL={`../clients/metrics/bpc`}
                            />
                        </Grid2>
                    </Grid2>
```

```
                </Box>
            </Box>
        </Box>
    )
}

```

frontend\src\pages\home\sprints\metrics\okr\ok_metric_base.js

```
import { Box, Divider, FormControl, InputLabel, MenuItem, Select, styled, Typography } from "@mui/material"
import { MMPaper } from "../../../../../components/shared/mm_paper"
import { MMSelect } from "../../../../../components/shared/mm_select"
import { useEffect, useState } from "react"
import { MMBaseTag, MMTagGroup } from "../../../../../components/shared/tags/mm_base_tag"

const TimeWindowSelect = ({ timeWindowList, onChangeValue }) => {
    return (<FormControl fullWidth>
        <InputLabel id="time-input-label" size="small">Sprints</InputLabel>
        <Select
            size="small"
            labelId="time-label"
            id="time-select"
            label="Sprints"
            defaultValue={''}
            // value="days"
            onChange={(event) => { onChangeValue(event.target.value) }}
        >
            {
                timeWindowList.map((timeWindow) => <MenuItem value={timeWindow}>{timeWindow}</MenuItem>)
            }
        </Select>
    </FormControl>)
}
const numberElementsPerTimeWindow = {

    'month': 12,
    'bimonth': 6,
    'quarter': 3,
    'semester': 2,
    'yearly': 1
}



const StyledText = styled('text')(({ theme }) => ({
    fill: theme.palette.text.primary,
    textAnchor: 'middle',
    dominantBaseline: 'central',
    fontSize: 48,
    // fontWeight: "bold"
    fontFamily: 'Roboto',
    color: "#333"
}));

export const OKRMetricBase = ({ metricTitle, chart, data, updateHandler }) => {

    const [year, setYear] = useState('2024')
    const [timeGranularity, setTimeGranularity] = useState('month')
    const [timeItem, setTimeItem] = useState(0)

    const getData = () => {
        return data[year][timeGranularity][timeItem]
    }

    useEffect(() => {
        updateHandler(year, timeGranularity, timeItem)
    }, [year, timeGranularity, timeItem])

    const onTimeWindowChangeHandler = (timeWindow) => {
        setTimeGranularity(timeWindow)
    }

    const onYearChangeHandler = (year) => {
        setYear(year)
    }

    const onTimeWindowItemChangeHandler = (item, _) => {
        setTimeItem(item)
    }

    const numOfLabels = numberElementsPerTimeWindow[timeGranularity]

    const generateTimeWindowLabel = () => {
        return (
```

```
            <MMTagGroup onSelectionChange={onTimeWindowItemChangeHandler}>
                {
                    Array.from({ length: numOfLabels }, (_, i) => i)
                        .map(i => <MMBaseTag
                            selectable
                            tag={`${i + 1} ${timeGranularity}`}
                            valueTag={i}
                            onSelectionChange={onTimeWindowItemChangeHandler}
                        />)

                }
            </MMTagGroup>
        )
    }

    return (
        <MMPaper width={'90%'} elevation={5}>
            <Box display={'flex'} flexDirection={'row'} width={'100%'} padding={'32px'} height="100%">
                <Box display="flex" alignItems="center" justifyContent="center" flex={1}>
                    {chart}
                </Box>
                <Box width="16px" />
                <Divider orientation="vertical" />
                <Box width="16px" />
                <Box display="flex" flexDirection="column" width="400px">
                    <Typography variant="h6">{metricTitle}</Typography>
                    <Box height="32px" />
                    <Box display="flex" flexDirection="row">
                        <MMSelect onChangeValue={onTimeWindowChangeHandler} list={['Month', 'Bimonth', 'Quarter', 'Semester', 'Yearly']} placeholder="Granularity" />
                        <Box width="32px" />
                        <MMSelect onChangeValue={onYearChangeHandler} list={["2024"]} placeholder="Year" />
                    </Box>
                    <Box>
                        {generateTimeWindowLabel()}
                    </Box>
                </Box>

            </Box>
        </MMPaper >

    )
}
```

frontend\src\pages\home\sprints\metrics\okr\okr_metrics.js

```
import { Box, Grid2, Typography } from "@mui/material"

import { MetricCard } from "../components/metric_card"
import { MMCircularProgressCentered } from "../../../../../components/shared/mm_circular_progress"

const totalSprintDeliveriesDescription =
`
    The OKR Total Sprint Deliveries refers to the goal of achieving an average completion rate of 80% of planned tasks in each
    sprint within an established evaluation period. The objective of this OKR is to encourage teams to maintain a high average
    of completed deliveries, fostering consistency in outcomes.
`

const bugResolutionTimeDescription =
`
    The OKR Bug Resolution Time, in turn, measures the time taken to resolve tasks tagged as “Bug”
    immediately after issues are reported in the production environment. The set objective is to correct
    75% of reported bugs within three days of identification.
`
const bugTrackerCompletedProjects =
`
    The OKR Bug Tracker for Completed Projects monitors the percentage of completed projects
    (i.e., projects that have concluded all associated tasks) that generate bug records in production.
    The established goal for this OKR is to ensure that a maximum of 20% of completed projects
    result in at least one task identified as a “Bug” after being deployed in the production environment.
`
export const OKRMetrics = () => {
    return (
        <Box padding={'16px'} sx={{ width: '100%' }} boxSizing={'border-box'} >
            <Box display={'flex'} flexDirection={'column'}>
                <Typography variant="h4">OKRs Metrics</Typography>
                <Box height={'32px'} />
                <Box>
                    <Grid2 direction={'row'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }}>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Total Sprint Deliveries"}
```

```
                                description={totalSprintDeliveriesDescription}
                                chartMock={<MMCircularProgressCentered progress="65" radius="30" color="#a10d47"/>}
                                metricURL={"../metrics/okr/completed-tasks-of-all-sprints"}
                            />
                        </Grid2>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Bug Resolution Time"}
                                description={bugResolutionTimeDescription}
                                chartMock={<MMCircularProgressCentered progress="79" radius="30" color="#0D47A1"/>}
                                metricURL={"../metrics/okr/bugs-solved"}
                            />
                        </Grid2>
                        <Grid2 size={5}>
                            <MetricCard
                                name={"Bug Tracker for Completed Projects"}
                                description={bugTrackerCompletedProjects}
                                chartMock={<MMCircularProgressCentered progress="95" radius="30" color="#47a10d"/>}
                                metricURL={"../metrics/okr/completed-bugs-tracker"}
                            />
                        </Grid2>
                    </Grid2>
                </Box>
            </Box>
        </Box>
    )
}

```

frontend\src\pages\home\sprints\metrics\types\metrics_card_base.js

```
import { Box, Card, CardActions, CardContent, Typography } from "@mui/material"
import { Link, useLocation } from "react-router-dom"

export const MetricsCardBase = ({ name, description, path }) => {
    const location = useLocation()
    const currentURL = location.pathname
    return (
        <Card sx={{ background: '' }}>
            <CardContent>
                <Box height="150px">




                    <Typography gutterBottom variant="h5" component="div" color="textSecondary">
                        {name}
                    </Typography>
                    <Typography gutterBottom variant="body1" component="div">
                        {description}
                    </Typography>

                </Box>
                <CardActions>
                    <Link
                        to={`${currentURL}/${path}`}
                    >
                        See more
                    </Link>
                </CardActions>
            </CardContent>
        </Card>
    )
}
```

frontend\src\pages\home\sprints\metrics\metrics_page.js

```
import { Grid2, Stack } from "@mui/material";
import { MetricsCardBase } from "./types/metrics_card_base";

export const MetricsPage = () => {
    return (

        <Grid2 boxShadow={'5px 5px 5px rgba(25, 118, 210, 0.1);'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }} direction={'row'} sx={{
            marginTop: '32px', background: '', boxSizing: 'border-box'
        }}>
            <Grid2 size={3}>
                <MetricsCardBase
                    name="Agile Metrics"
                    description="Metrics widely discussed in the literature and available for team tracking."
                    path="agile"
                />
            </Grid2>
            <Grid2 size={3}>
                <MetricsCardBase
                    name="Custom Metrics"
```

```
                    description="Metrics tailored to the client’s needs. Built on demand."
                    path="custom"
                />
            </Grid2>
            <Grid2 size={3}>
                <MetricsCardBase
                    name="OKRs"
                    description="Real-time tracking of Objective Key Results."
                    path="okr"
                />
            </Grid2>
        </Grid2>
    );
}
```

frontend\src\pages\home\sprints\search\utils.js

```
const isInRage = ([begin, end], value) => begin <= value && value < end;
const assembleRanges = (matchRanges, string) => {
    let info = [];
    let scanIndex = 0;
    let matchesIndex = 0;
    const lastIndex = string.length - 1;

    if (matchRanges.length === 0) return [[[0, string.length - 1], false]]

    while (scanIndex <= lastIndex && matchesIndex < matchRanges.length) {
        const currentMatch = matchRanges[matchesIndex];
        const [mBegin, mEnd] = currentMatch

        if (!isInRage(currentMatch, scanIndex)) {
            info.push([[scanIndex, mBegin], false])
        }

        info.push([currentMatch, true]);

        if (matchesIndex === matchRanges.length - 1) {
            if (mEnd <= lastIndex) {
                info.push([[mEnd, lastIndex + 1], false])
            }
        }




        scanIndex = mEnd
        matchesIndex++
    }

    return info
};


export const getMatchingRangesFromString = (string, pattern) => {
    const regex = RegExp(`${pattern}`, 'g')
    const matches = [...string.matchAll(regex)]
    const matchRanges = matches.map((match) => [match.index, match.index + match[0].length])
    const ranges = assembleRanges(matchRanges, string)
    return ranges
}
```

frontend\src\pages\home\sprints\tasks\tasks_list\task_accordion\employee_info_popover.js

```
import { Box, Button, Popover, Typography } from "@mui/material";
import { useState } from "react";
import { getTeamTagByName } from "../../../../../../components/shared/tags/team_tag";
import { HomeController } from "../../../../../controller/home_controller";
import { Link } from "react-router-dom";
import { wesleyProfileImageURL } from "../../../../../../images/wesley_pic";


export const EmployeeInfoPopover = ({ employeeName, employeeEmail }) => {
    const [anchorEl, setAnchorEl] = useState(null);
    const [employeeInfo, setEmployeeInfo] = useState({})
    const homeController = new HomeController()

    const handleClick = (event) => {
        const fetchEmployeeInfo = async () => {
            const info = await homeController.getEmployeeInfo(employeeEmail)
            setEmployeeInfo(info)
        }
        fetchEmployeeInfo()
        setAnchorEl(event.currentTarget);
    };

```

```
    const handleClose = () => {
        setAnchorEl(null);
    };

    const open = Boolean(anchorEl);
    const id = open ? 'simple-popover' : undefined;

    const { name, email, position, teamName, lastSprints } = employeeInfo

    return (
        <div>
            <Link href="/" onMouseOver={handleClick} underline="none" onClick={(e) => e.preventDefault()}>
                {employeeName}
            </Link>
            <Popover
                id={id}
                open={open}
                anchorEl={anchorEl}
                onClose={handleClose}
                transformOrigin={{
                    vertical: 'bottom',
                    horizontal: 'right',
                }}>
                <Box height={'400px'} width={'400px'} display={'flex'} flexDirection={'column'} padding={'16px'}>
                    <Box display={'flex'} height={'50px'} alignItems={'center'} justifyContent={'space-between'} >
                        <Typography variant="h6">{ name }</Typography>
                        <div style=
                            {{
                                height: '50px',
                                width: '50px',
                                borderRadius: '50%',
                                backgroundColor: 'red',
                                backgroundSize: 'cover',
                                backgroundImage: `url(${wesleyProfileImageURL})`
                            }}>

                        </div>
                    </Box>
                    <Box height={'32px'} />
                    <Box display={'flex'}>
                        <Box display={'flex'} flexDirection={'column'} flex={1}>
                            <Box height={'48px'}>
                                <Typography>Role: </Typography>
                            </Box>
                            <Box height={'48px'}>
                                <Typography>Email: </Typography>
                            </Box>
                            <Box height={'48px'}>
                                <Typography>Team: </Typography>
                            </Box>
                            <Box height={'48px'}>
                                <Typography>Last Sprints: </Typography>
                            </Box>
                        </Box>
                        <Box display={'flex'} flexDirection={'column'} flex={1}>
                            <Box height={'48px'}>
                                <Typography>{position}</Typography>
                            </Box>
                            <Box height={'48px'}>
                                <Link><Typography>{email}</Typography></Link>
                            </Box>
                            <Box height={'48px'}>
                                {getTeamTagByName(teamName)}
                            </Box>
                            <Box height={'48px'}>
                                {lastSprints ? lastSprints.map((code) => <Link underline="hover">{code}</Link>) : <></>}
                            </Box>
                        </Box>
                    </Box>
                    <Box >
                        <Link to={`../employees/${employeeEmail}/full-metrics`}>
                            <Button>See metrics on {name}</Button>
                        </Link>
                    </Box>

                </Box>
            </Popover>
        </div>
    );
}
```

frontend\src\pages\home\sprints\tasks\tasks_list\task_accordion\task_accordion.js

```jsx
import { Accordion, AccordionDetails, AccordionSummary, Box, Divider, Paper, styled, Typography } from "@mui/material";
import { EmployeeInfoPopover } from "./employee_info_popover";
import { getPriorityChip } from "../../../../../../components/shared/tags/priority_tags";
import { getTaskTagByName } from "../../../../../../components/shared/tags/task_tags";
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';
import { MMPaper } from "../../../../../../components/shared/mm_paper";
import { toReadbleDate } from "../../../../../presenters/utility"
import { Link } from "react-router-dom";
import { wesleyProfileImageURL } from "../../../../../../images/wesley_pic";

const TaskPointsChip = ({ points }) => {
    return (
        <Box width={'30px'} marginRight={'8px'} >
            <Paper elevation={1} sx={{ height: '100%', textAlign: 'center' }}>{points}</Paper>
        </Box>
    )
}

export const TaskAccordion = ({ task }) => {
    const {
        title,
        sprintCode,
        points,
        devEmail,
        devName,
        qaEmail,
        qaName,
        tags,
        description,
        linkedTs,
        beginTs,
        endTs
    } = task;
    const tagsComponents = tags.map((tag) => getTaskTagByName(tag))
    return (
        <Accordion>
            <AccordionSummary
                expandIcon={<ExpandMoreIcon />}
                aria-controls="panel1-content"
                id="panel1-header"
            >
                <Typography>{title}</Typography>
                <Box flex={1}></Box>
                {tags.map((tag) => <Box margin="0 8px 0 8px">{getTaskTagByName(tag)}</Box>)}

                <div style=
                    {{
                        height: '30px',
                        width: '30px',
                        borderRadius: '50%',
                        backgroundColor: 'red',
                        backgroundSize: 'cover',
                        backgroundImage: `url(${wesleyProfileImageURL})`
                    }}>

                </div>
                <Box width={'8px'} />
                <TaskPointsChip points={points} />
            </AccordionSummary>
            <AccordionDetails>
                <Box>
                    <Divider orientation="horizontal" sx={{ width: '100%' }} />
                    <Box height={'32px'} />
                    <Box flex={1} display={'flex'} flexDirection={'row'}>
                        <Box flex={1}>
                            <MMPaper padding={'16px'} height="100%">
                                <Box display={'flex'} flexDirection={'column'}>
                                    <Typography variant="h6">Details:</Typography>
                                    <Box height={'16px'} />
                                    <Box flexDirection={'column'} >
                                        <Box display={'flex'} flexDirection={'row'} paddingLeft={'32px'}>
                                            <Box display={'flex'} flexDirection={'column'} width={'fit-content'} height={''}>
                                                <Box height={'64px'}>
                                                    <Typography variant="body1">Priority:</Typography>
                                                </Box>
                                                <Box>
                                                    <Typography variant="body1">Tags:</Typography>
                                                </Box>
                                            </Box>
                                            <Box width={'32px'} />
                                            <Box display={'flex'} flexDirection={'column'} width={'fit-content'} height={'200px'} >
                                                <Box height={'64px'}>
```

```
                                        <Typography variant="body1">{getPriorityChip('high')}</Typography>
                                    </Box>
                                    <Box display={'flex'} alignItems={'center'} >

                                        {
                                            tagsComponents[0]
                                        }
                                        {
                                            tagsComponents.slice(1, tagsComponents.length).map((component) => <><Divider orientation="vertical" sx={{ margin: '0 4px 0 4px', height: '60%' }} />{component}</>)
                                        }
                                    </Box>
                                </Box>
                                <Box width={'128px'} />
                                <Box display={'flex'} flexDirection={'column'} width={'fit-content'} height={'200px'} >
                                    <Box height={'64px'}>
                                        <Typography variant="body1">Task Status: </Typography>
                                    </Box>
                                    <Box height={'64px'}>
                                        <Typography variant="body1">Sprint: </Typography>
                                    </Box>
                                </Box>
                                <Box width={'128px'} />
                                <Box display={'flex'} flexDirection={'column'} width={'fit-content'} height={'200px'} >
                                    <Box height={'64px'}>
                                        <Typography variant="body1">Resolved</Typography>
                                    </Box>
                                    <Box height={'64px'}>
                                        <Typography><Link>{sprintCode}</Link></Typography>
                                    </Box>
                                </Box>
                            </Box>
                            <Typography variant="h6">Description:</Typography>
                            <Box flexDirection={'row'} paddingLeft={'32px'}>
                                <p>{description}</p>
                            </Box>
                        </Box>
                    </Box>
                </MMPaper>
            </Box>
            <Box width={'16px'} />
```

```
            <Divider orientation="vertical" flexItem />
            <Box width={'16px'} />
            <Box flex={1}>
                <MMPaper padding={'16px'} height="100%">
                    <Box display={'flex'} flex={1} flexDirection={'column'}>
                        <Typography variant="h6">People: </Typography>
                        <Box display={'flex'}>
                            <Box display={'flex'} flexDirection={'column'} flex={1}>
                                <Box height={'64px'}>
                                    <Typography>Asignee: </Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>Tester: </Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>Added At: </Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>Started At: </Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>Finished At: </Typography>
                                </Box>
                            </Box>
                            <Box display={'flex'} flexDirection={'column'} flex={1}>
                                <Box height={'64px'}>
                                    <EmployeeInfoPopover employeeName={devName} employeeEmail={devEmail} />
                                </Box>
                                <Box height={'64px'}>
                                    <EmployeeInfoPopover employeeName={qaName} employeeEmail={qaEmail} />
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>{toReadbleDate(linkedTs)}</Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>{toReadbleDate(beginTs)}</Typography>
                                </Box>
                                <Box height={'64px'}>
                                    <Typography>{toReadbleDate(endTs)}</Typography>
                                </Box>
                            </Box>
```

```
                                    </Box>
                                </Box>
                            </MMPaper>
                        </Box>
                    </Box>
                </Box>
            </AccordionDetails>
        </Accordion>
    )
}
```

frontend\src\pages\home\sprints\tasks\tasks_list\tasks_list.js

```
/**
 *
 * Task - title, created_by, assigned_to, points,
 */

import { Box, Typography } from "@mui/material"
import { useParams } from "react-router-dom";
import { useEffect, useState } from "react";
import { HomeController } from "../../../controller/home_controller";
import { TaskAccordion } from "./task_accordion/task_accordion";

export const TaskList = ({ }) => {

    const { code } = useParams('code')
    const [tasks, setTasks] = useState([])
    const homeController = new HomeController()


    useEffect(() => {
        const fetchTasks = async () => {
            const tasksList = await homeController.getTasksFromSprint(code)
            setTasks(tasksList)
        }


        fetchTasks()
    }, [])




    return (
        <Box sx={{ width: '100%' }} boxSizing={'border-box'} display={'flex'} flexDirection={'column'}>
            <Typography variant="h5">{code}</Typography>
            <Box height={'32px'} />
            {
                tasks.map(task => <TaskAccordion task={task} />)
            }
        </Box>
    )
}
```

frontend\src\pages\home\sprints\index.js

```
import { Alert, Box, Container, Grid2, Popover } from "@mui/material"
import { FilterArea } from "../filter_area"
import SprintCard from "../../../components/sprint_card"
import { useEffect, useState } from "react"
import { createContext } from "react";
import { getMatchingRangesFromString } from "./search/utils";
import { Link } from "react-router-dom";

const SearchSettingContext = createContext({
    teams: [],
    projects: [],
    rangeDate: [null, null],
    textSearch: ''
});

export const Sprints = ({ sprints, teams, headerHeight = 80 }) => {
    const [sprintsInfo, setSprintsInfo] = useState([])
    const [filterSettings, setFilterSettings] = useState({
        teams: [],
        projects: [],
        rangeDate: [null, null],
        textSearch: ''
    })

    useEffect(() => {
        const temp = sprints.map(sprint => {
            return {
                'sprint': sprint,
```

```
                'isSelected': true,
                'matchingIndexesDescription': [
                    {
                        'indexPair': [0, sprint.description.length],
                        'isMatch': false
                    }
                ],
                'matchingIndexesTitle': [
                    {
                        'indexPair': [0, sprint.description.length],
                        'isMatch': false
                    }
                ]

            }
        })

        setSprintsInfo(temp)
    }, [])

    const getSearchInfo = (text, textPattern) => {
        const ranges = getMatchingRangesFromString(text, textPattern)

        return {
            'ranges': ranges,
            'isMatch': ranges.length > 1 || text === textPattern
        }
    }

    useEffect(() => {
        const { textSearch, teams: filteredTeams } = filterSettings
        const updatedSprintsInfo = sprints.map((sprint) => {
            const { title, description, teamName } = sprint
            const { ranges: rangesDescription, isMatch: isMatchDescription } = getSearchInfo(description, textSearch)
            const { ranges: rangesTitle, isMatch: isMatchTitle } = getSearchInfo(title, textSearch)

            const teamSelected = filteredTeams.length === 0 || filteredTeams.includes(teamName)

            return {
                'sprint': { ...sprint },
                'isSelected': (isMatchTitle || isMatchDescription || textSearch.length === 0) && teamSelected,
```

```
                'matchingIndexesDescription': rangesDescription.map((range) => {
                    return {
                        'indexPair': range[0],
                        'isMatch': range[1],
                    }
                }),
                'matchingIndexesTitle': rangesTitle.map((range) => {
                    return {
                        'indexPair': range[0],
                        'isMatch': range[1],
                    }
                })
            }
        })
        setSprintsInfo(updatedSprintsInfo)
    }, [filterSettings])


    const handleTypeText = (text) => {
        setFilterSettings({
            ...filterSettings,
            'textSearch': text
        })
    }

    const labelOnChangeSelectionHandler = (team, isActive) => {
        let activeLabels = filterSettings['teams']

        if (isActive) activeLabels = [...activeLabels, team]
        else activeLabels = activeLabels.filter((label) => label != team)

        setFilterSettings({
            ...filterSettings,
            'teams': activeLabels
        })
    }

    return (
        <Box width="100%">
        <Box boxSizing="border-box" width={"100%"} boxShadow={'5px 5px 5px rgba(25, 118, 210, 0.1);'} alignSelf={"begin"}>
```

```
            <Container maxWidth={false} sx={{ height: '164px', gap: '16px', alignItems: 'start', alignItems: 'center', display: 'flex', flex: 1, boxSizing: 'border-box', width: '100%' }} >
                <SearchSettingContext.Provider>
                    <FilterArea teams={teams} textInputCallback={handleTypeText} teamLabelClickHandler={labelOnChangeSelectionHandler} />
                </SearchSettingContext.Provider>
            </Container>
        </Box>

        <Grid2 boxShadow={'5px 5px 5px rgba(25, 118, 210, 0.1);'} container spacing={5} columns={{ xs: 4, sm: 8, md: 12 }} direction={'row'} sx={{
            marginTop: '32px', background: '', boxSizing: 'border-box'
        }}>

            {
                sprintsInfo.map(sprintInfo => {
                    const { sprint, isSelected, matchingIndexesDescription, matchingIndexesTitle } = sprintInfo

                    const descriptionMatchingIndexes = matchingIndexesDescription.map(matchingIndex => {
                        return [matchingIndex['indexPair'], matchingIndex['isMatch']]
                    })
                    const titleMatchingRanges = matchingIndexesTitle.map(matchingIndex => {
                        return [matchingIndex['indexPair'], matchingIndex['isMatch']]
                    })
                    if (!isSelected) return <></>

                    return (<Grid2 size={3}>
                        <SprintCard
                            title={sprint.title}
                            teamName={sprint.teamName}
                            code={sprint.code}
                            description={sprint.description}
                            pointsDone={sprint.pointsDone}
                            pointsTotal={sprint.pointsTotal}
                            beginDate={"11-12-2024"}
                            endDate={"11-12-2024"}
                            descriptionMatchingIndexes={descriptionMatchingIndexes}
                            titleMatchingIndexes={titleMatchingRanges}
                            done={sprint.description}
                        />
                    </Grid2>)
                }
                )
```

```
            }
        </Grid2>
    </Box>)
}
```

frontend\src\pages\home\teams\team_card.js

```
import { Button, Card, CardContent, CardActions, Box, Container, Typography, Collapse, IconButton, styled } from "@mui/material"
import { useState } from "react";
import { Link } from "react-router-dom"
import ExpandMoreIcon from '@mui/icons-material/ExpandMore'
import { EmployeeInfoPopover } from "../sprints/tasks/tasks_list/task_accordion/employee_info_popover";

const ExpandMore = styled((props) => {
  const { expand, ...other } = props;
  return <IconButton {...other} />;
})(({ theme }) => ({
  marginLeft: 'auto',
  transition: theme.transitions.create('transform', {
    duration: theme.transitions.duration.shortest,
  }),
  variants: [
    {
      props: ({ expand }) => !expand,
      style: {
        transform: 'rotate(0deg)',
      },
    },
    {
      props: ({ expand }) => !!expand,
      style: {
        transform: 'rotate(180deg)',
      },
    },
  ],
}));

export const TeamCard = ({ team }) => {
  const { teamName, memberEmails, membersNames, lastSprints } = team
  const [emailListExpanded, setEmailListExpanded] = useState(false)

```

```
  function handleExpandClick() {
    setEmailListExpanded(!emailListExpanded)
  }

  return (
    <Card sx={{ minHeight: 300, paddingTop: '32px', position: 'relative' }}>
      <CardContent sx={{ height: '80%' }}>
        <Typography variant="h6">{teamName}</Typography>
        <Box display={'flex'} flexDirection={'row'} flex={1}>
          <Box display={'flex'} flexDirection={'column'} flex={1}>
            <Typography>Tech Lead: </Typography>

            <Box display={'flex'} flexDirection={'row'} alignItems={'center'} height={emailListExpanded ? '100px' : '50px'} sx={{ transitionDuration: '0.2s' }}>
              <Typography>Members </Typography>
              <ExpandMore
                expand={emailListExpanded}
                onClick={handleExpandClick}
                aria-expanded={emailListExpanded}
                aria-label="show more"
              >
                <ExpandMoreIcon enableBackground={true} />
              </ExpandMore>
            </Box>
          </Box>
          <Box display={'flex'} flexDirection={'column'} flex={1} >
            <Typography>{membersNames[0]}</Typography>
            <Collapse in={emailListExpanded}>
              <Box display={'flex'} flexDirection={'column'} height={'fit-content'} justifyContent={'center'}>
                {membersNames.map((name, index) => <Typography>
                  <EmployeeInfoPopover
                    employeeEmail={memberEmails[index]}
                    employeeName={name}
                  >{name}
                  </EmployeeInfoPopover>
                </Typography>)}
              </Box>
            </Collapse>
          </Box>
        </Box>
      </CardContent>
```

```
      <CardActions>
        <Button size="medium">
          <Link style={{ color: 'inherit', textDecoration: 'inherit' }} to={""}>
            See Metrics on {teamName}
          </Link>
        </Button>
        <Box flex={1} />
      </CardActions>
    </Card>
  )
}

/**
 * mostrar nome do time - ok
 * mostrar o tech lead se houver
 * mostrar últimas sprints
 * mostrar uma lista de membros - ok
*/
```

frontend\src\pages\home\teams\teams_page.js

```
import { Box, Grid2, Paper, styled } from "@mui/material"
import { HomeController } from "../controller/home_controller"
import { useEffect, useState } from "react"
import { TeamCard } from "./team_card";

export const TeamsPage = ({ }) => {
    const homeController = new HomeController()
    const [teams, setTeams] = useState([])

    useEffect(() => {
        const fetchTeams = async () => {
            const teams = await homeController.getTeamsGeneralInfo()
            setTeams(teams)
        }
        fetchTeams()
    }, [])

    return (
        <Box width={'100%'} boxSizing="border-box" height="100vh">
            <Grid2 container spacing={2} alignItems="flex-start" width="100%">
```

```
            {
                teams.map(team => <Grid2 xs={4}>
                    <TeamCard team={team} />
                </Grid2>)
            }
            </Grid2>
        </Box>
    )
}

```

frontend\src\pages\home\filter_area.js

```
import { Container, TextField } from "@mui/material"
import { Label, ToggleLabel } from "../../components/shared/simple_lable"

export const FilterArea = ({ teams, textInputCallback, teamLabelClickHandler }) => {
    return (<>
        {teams.map((team) => <ToggleLabel text={team} color={"success"} clickHandler={(isActive) => teamLabelClickHandler(team, isActive)} />)}
        <TextField id="standard-basic" label="Search" fullWidth={true} variant="standard" onChange={(e) => textInputCallback(e.target.value)}/>
    </>)
}

```

frontend\src\pages\login\login_page.js

```
import { Alert, Box, Button, FormControl, IconButton, InputAdornment, InputLabel, OutlinedInput, TextField, Typography } from "@mui/material";
import { MMPaper } from "../../components/shared/mm_paper";
import { HomeController } from "../home/controller/home_controller";
import { useRef, useState } from "react";
import { useAuth } from "../../state/providers/auth_provider";
import { json, useNavigate } from "react-router-dom";
import Visibility from '@mui/icons-material/Visibility';
import VisibilityOff from '@mui/icons-material/VisibilityOff';

import background from './../../images/login/background2.jpg'
import logo from './../../images/logo.png'

const LoginTextField = ({ text, onValueChange, error = false }) => {
    const textRef = useRef()
    return (




        <FormControl variant="outlined">
                <InputLabel htmlFor="outlined-adornment-password">Email</InputLabel>

            <OutlinedInput

                id="standard-basic"
                label={text}
                error={error}
                variant="outlined"
                onChange={(_) => {
                    onValueChange(textRef.current.value)
                }}
                inputRef={textRef}
                sx={{ background: 'white' }}
            />
        </FormControl>
    )
}

const PasswordTextField = ({ onValueChange, error = false }) => {
    const [showPassword, setShowPassword] = useState(false)
    const textRef = useRef()

    const handleClickShowPassword = () => setShowPassword((show) => !show);

    const handleMouseDownPassword = (event) => {
        event.preventDefault();
    };

    const handleMouseUpPassword = (event) => {
        event.preventDefault();
    };

    return (
        <>
            <FormControl variant="outlined">
                <InputLabel htmlFor="outlined-adornment-password">Password</InputLabel>
                <OutlinedInput
                    id="outlined-adornment-password"
                    type={showPassword ? 'text' : 'password'}
                    inputRef={textRef}
```

```
                    onChange={(_) => {
                        console.log(textRef.current.value)
                        onValueChange(textRef.current.value)
                    }}
                    error={error}
                    endAdornment={
                        <InputAdornment position="end">
                            <IconButton
                                aria-label={
                                    showPassword ? 'hide the password' : 'display the password'
                                }
                                onClick={handleClickShowPassword}
                                onMouseDown={handleMouseDownPassword}
                                onMouseUp={handleMouseUpPassword}
                                edge="end"
                            >
                                {showPassword ? <VisibilityOff /> : <Visibility />}
                            </IconButton>
                        </InputAdornment>
                    }
                    label="Password"
                    sx={{ background: 'white' }}
                />
            </FormControl>
        </>
    )
}

const emptyFieldError = {
    error: true,
    message: 'Please fill email and password.'
}

const notPossibleToAuthenticateError = {
    error: true,
    message: 'It was not possible to continue. Please check your credentials.'
}

export const LoginPage = ({ }) => {
    const [loginInfo, setLoginInfo] = useState({ 'email': '', 'password': '' })
    const { authState, setAuthState } = useAuth()
```

```
    const [errorInfo, setErrorInfo] = useState({
        error: false,
        message: '',
    })

    const navigate = useNavigate()
    const homeController = new HomeController(null)

    const authenticationHandler = async () => {
        const authenticate = async () => {
            const { email, password } = loginInfo
            console.log(email, password)

            if (email === '' || password === '') {
                setErrorInfo(emptyFieldError)
                return
            }

            const resp = await homeController.authenticate(email, password)

            if (resp === 'unknown') {
                setErrorInfo(notPossibleToAuthenticateError)
                return
            }
            console.log(resp)
            const [ token, user ] = resp

            setAuthState((prev) => ({
                'accessToken': token,
                'refreshToken': token,
                // ...prev.user,
                'name': user.name,
                'email': user.email,

            }))

            navigate("/home");
        }

        await authenticate()
    }
```

```
    return (
        <Box sx={{ background: `url(${background})`, backgroundSize: 'cover', backgroundRepeat: 'no-repeat' }} width={'100vw'} height={'100vh'} bgcolor={'#b84761'} display={'flex'} alignItems={'center'} justifyContent={'center'}>
            <MMPaper width={'500px'} height={'500px'} sx={{ borderRadius: '32px', background: 'rgba(255, 255, 255, .75)' }}  >
                <Box height={'100%'} width={'100%'} display={'flex'} flexDirection={'column'} alignItems={'center'} justifyContent={'center'} padding={'16px'}>

                    <Box flex={1} bgcolor="red" width="200px" height="200px" sx={{ background: `url(${logo})`, backgroundSize: 'cover', backgroundRepeat: 'no-repeat' }} />

                    <Box height="16px" />
                    <Box display={'flex'} flex={2} flexDirection={'column'} width={'70%'}>
                        <LoginTextField text={"Email"} onValueChange={(value) => setLoginInfo({ 'email': value, 'password': loginInfo.password })} />
                        <Box height={'32px'} />
                        <PasswordTextField text={"Password"} onValueChange={(value) => setLoginInfo({ 'email': loginInfo.email, 'password': value })} />
                        <Box height={'32px'} />
                        <Button
                            onClick={async () => await authenticationHandler()}
                            variant="contained">Login</Button>
                        <Box height="32px" />
                        <Box height="32px" display="flex" alignItems="center" justifyContent="center">
                            { errorInfo.error ? <Alert severity="warning">{errorInfo.message}</Alert> : ''}
                        </Box>
                        <Box height="64px" />
                    </Box>
                </Box>
            </MMPaper>
        </Box>
    )
}
```

frontend\src\state\providers\auth_provider.js

```
import React, { createContext, useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { useLocalStorage } from '../../components/shared/hooks/use_local_storage';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
```

```
  const [authState, setAuthState] = useState({
    'accessToken': '',
    'refreshToken': '',
    'name': '',
    'email': ''
  })

  const [ready, setReady] = useState(false)

  useEffect(() => {
    setReady(false)
    if (authState.accessToken !== '' && authState.accessToken !== null && authState.accessToken !== undefined) {
      localStorage.setItem('authInfo', JSON.stringify(authState))
    }
    setReady(true)

  }, [authState])


  useEffect(() => {
    const authInfoString = localStorage.getItem('authInfo')
    if (authInfoString === undefined || authInfoString === null) {
      return
    }

    const { accessToken, name, email } = JSON.parse(authInfoString)
    setAuthState((_) => ({
      'name': name,
      'email': email,
      'accessToken': accessToken,
      'refreshToken': accessToken,
    }))

    setReady(true)

  }, [])

  const logout = () => {
    localStorage.removeItem('authInfo')
    setAuthState({
      'accessToken': null,
```

```
      'refreshToken': null,
      'name': null,
      'email': null
    })
  }

  return (
    <AuthContext.Provider value={{ authState, setAuthState, logout, ready }}>
      { ready ? children : <></>}
    </AuthContext.Provider>
  )
}

export const useAuth = () => useContext(AuthContext)

```

frontend\src\App.css

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #838b9c;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```

frontend\src\App.js

```
import logo from './logo.svg';
import './App.css';
import Home from './home';
import { ThemeProvider, createTheme } from '@mui/material/styles';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import { LoginPage } from './pages/login/login_page';
import { AuthProvider } from './state/providers/auth_provider';
import { ProtectedRouter } from './pages/home/controller/router/protected_router';


const darkTheme = createTheme({
    palette: {
        mode: 'dark',
        primary: {
            main: '#ff0000'
        }
    },
});

function App() {
    return (
        <AuthProvider>
```

```
            <BrowserRouter>
                <Routes>
                    <Route path="/" element={<LoginPage />} />
                    <Route path="/home/*" element={<ProtectedRouter><Home /></ProtectedRouter>} />
                </Routes>
            </BrowserRouter>
        </AuthProvider>
    );
}

export default App;

```

frontend\src\App.test.js

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});

```

frontend\src\home_drawer.js

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Drawer from '@mui/material/Drawer';
import List from '@mui/material/List';
import Divider from '@mui/material/Divider';
import ListItem from '@mui/material/ListItem';
import ListItemButton from '@mui/material/ListItemButton';
import ListItemIcon from '@mui/material/ListItemIcon';
import ListItemText from '@mui/material/ListItemText';
import Collapse from "@mui/material/Collapse";

import Container from "@mui/material/Container";
import DataObjectIcon from '@mui/icons-material/DataObject';



import AccountCircleIcon from '@mui/icons-material/AccountCircleRounded';
import BarChartIcon from '@mui/icons-material/BarChart';

import PeopleOutlineIcon from '@mui/icons-material/PeopleOutline';

import { IconButton } from '@mui/material';
import { Link, useLocation, useNavigate } from 'react-router-dom';

import ChevronLeftIcon from '@mui/icons-material/ChevronLeft';
import { useContext } from 'react';
import DrawerContext from './pages/home/home/drawer_context';
import logo from './images/logo.png'
import StorageIcon from '@mui/icons-material/Storage';
import InfoIcon from '@mui/icons-material/Info';
import PeopleAltIcon from '@mui/icons-material/PeopleAlt';

const selectedMenuItemBackground = '#e9e9e9'

const MenuItem = ({ text, Icon, path, submenu = [], selected = false }) => {
    const navigate = useNavigate()
    return (
        <>
            <ListItem sx={{ width: "100%" }}>
                <ListItemButton onClick={() => navigate(path)} sx={selected ? { background: selectedMenuItemBackground } : {}}>
                    <ListItemIcon>
                        {Icon}
                    </ListItemIcon>
                    <ListItemText primary={text} />
                </ListItemButton>
            </ListItem >
            <Collapse in={selected} timeout="auto"
                unmountOnExit>
                <Container sx={{
                    lineHeight: 2
                }}>
                    {
                        submenu.map(menu => {
                            const { text, path } = menu
                            return (<ListItem disablePadding>
                                <ListItemButton>
                                    <Link
```

```
                                    style={{ color: 'inherit', textDecoration: 'inherit' }} to={path}>
                                    <ListItemText primary={text} />
                                </Link>
                            </ListItemButton>
                        </ListItem>)
                    })
                }
            </Container>
        </Collapse>
    </>
  )
}

export const HomeDrawer = ({ width, headerHeight, drawerCallback }) => {

    const isDrawerOpen = useContext(DrawerContext)
    const location = useLocation();

    const selectedItem = location.pathname.split('/')[2]

    const menuInfo = {
        'about': {
            text: 'About',
            icon: <InfoIcon />,
            subMenu: []
        },
        'account': {
            text: 'YourAccount',
            icon: <AccountCircleIcon />,
            subMenu: []
        },
        'teams': {
            text: 'Teams',
            icon: <PeopleOutlineIcon />,
            subMenu: []
        },
        'sprints': {
            text: 'Sprints',
            icon: <StorageIcon />,
            subMenu: []
        },
        'projects': {
            text: 'Projects',
            icon: <DataObjectIcon />,
            subMenu: []
        },
        'metrics': {
            text: 'Metrics',
            icon: <BarChartIcon />,
            subMenu: [
                {
                    text: 'Agile Metrics',
                    path: 'metrics/agile',
                },
                {
                    text: 'Custom Metrics',
                    path: 'metrics/custom',
                },
                {
                    text: 'OKRs',
                    path: 'metrics/okr',
                },
            ]
        },
        'employees': {
            text: 'Employees',
            icon: <PeopleAltIcon/>,
            subMenu: []
        }
    }

    return (
        <Drawer
            variant="persistent"
            open={isDrawerOpen}
            sx={{
                // width: drawerWidth,
                flexShrink: 1,
                [`& .MuiDrawer-paper`]: { width: width, boxSizing: 'border-box' },
            }}
        >
            <Box height={`${headerHeight}px`} boxSizing={'border-box'} display={'flex'} justifyContent={'flex-end'}>
```

```
                <Box flex={1} display="flex" alignItems="center" justifyContent="center">
                    <Box sx={{ background: `url(${logo})`, backgroundSize: 'contain', backgroundRepeat: 'no-repeat' }} width="100px" height="100px">
</Box>
                </Box>
                <IconButton onClick={drawerCallback}>
                    <ChevronLeftIcon />
                </IconButton>
            </Box>
            <Divider orientation='horizontal' sx={{ width: '100%', alignSelf: 'center' }} />
            <Box>
                <List>
                    {
                        Object.keys(menuInfo).map((key) => {
                            const { text, icon } = menuInfo[key]
                            return <MenuItem text={text} Icon={icon} submenu={menuInfo[key]['subMenu']} path={key} selected={selectedItem == key} />
                        })
                    }
                </List>
            </Box>
        </Drawer>
    )
}
```

frontend\src\home.js

```
import * as React from 'react';
import Box from '@mui/material/Box';
import AppBar from '@mui/material/AppBar';
import CssBaseline from '@mui/material/CssBaseline';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import { useState } from "react";


import AccountCircleIcon from '@mui/icons-material/AccountCircleRounded';


import { BottomNavigation, BottomNavigationAction, IconButton, Menu, MenuItem, useTheme } from '@mui/material';
import { Route, Routes } from 'react-router-dom';
import { Sprints } from './pages/home/sprints';
import { BurndownPage } from './pages/home/metrics/burndown/burndown_page';
```

```
import { HomeController } from './pages/home/controller/home_controller';
import { VelocityPage } from './pages/home/metrics/velocity/velocity_page';
import { BurnUpPage } from './pages/home/metrics/burndown/burnup_page';
import MenuIcon from '@mui/icons-material/Menu';
import { WIPPage } from './pages/home/metrics/wip/wip_page';
import { MovingAverageVelocityPage } from './pages/home/metrics/moving_average_velocity/moving_average_velocity_page';
import { TaskList } from './pages/home/sprints/tasks/tasks_list/tasks_list';
import { HomeDrawer } from './home_drawer';
import DrawerContext from './pages/home/home/drawer_context';
import { useEffect } from 'react';
import { EmployeeFullInfoPage } from './pages/home/employee/employee_full_info_page';
import { TeamsPage } from './pages/home/teams/teams_page';
import { Metrics } from './pages/home/sprints/metrics/agile/agile_metrics';
import { useAuth } from './state/providers/auth_provider';
import { CostumeMetrics } from './pages/home/sprints/metrics/costume/costume_metrics';
import { AccountPage } from './pages/home/account/account_page';
import { DrawerAware } from './components/shared/drawer_aware/drawer_aware';
import { BugsPerClient } from './pages/home/metrics/bpc/bugs_per_client';
import { AboutPage } from './pages/home/about/about_page';
import { OKRMetrics } from './pages/home/sprints/metrics/okr/okr_metrics';
import { Projects } from './pages/home/project/projects';
import { TaskListFromProject } from './pages/home/project/task_list_page';
import { ClosedBugsOKR } from './pages/home/sprints/metrics/okr/closed_bugs';
import { CompletedProjectsBugTrackerOKR } from './pages/home/sprints/metrics/okr/completed_projects_bug_tracker_okr';
import { SprintsDeliveryOKR } from './pages/home/sprints/metrics/okr/sprints_delivery_okr';
import { MetricsPage } from './pages/home/sprints/metrics/metrics_page';
import { EmployeeListPage } from './pages/home/employee/employee_list_page';


const drawerWidth = 240;
const headerHeight = 80;

const BasicMenu = ({ }) => {
    const [anchorEl, setAnchorEl] = React.useState(null)
    const { logout } = useAuth()

    const open = Boolean(anchorEl)
    const handleClick = (event) => {
        setAnchorEl(event.currentTarget)
    };
    const handleClose = () => {
```

```
        setAnchorEl(null)
    };

    const handleLogout = () => { logout() }

    return (
        <div>
            <AccountCircleIcon sx={{ fontSize: '36px', '&:hover': { cursor: 'pointer' } }}
                id="basic-button"
                aria-controls={open ? 'basic-menu' : undefined}
                aria-haspopup="true"
                aria-expanded={open ? 'true' : undefined}
                onClick={handleClick}
            />
            <Menu
                id="basic-menu"
                anchorEl={anchorEl}
                open={open}
                onClose={handleClose}
                MenuListProps={{
                    'aria-labelledby': 'basic-button',
                }}
            >
                <MenuItem onClick={handleClose}>Profile</MenuItem>
                <MenuItem onClick={handleClose}>Settings</MenuItem>
                <MenuItem onClick={handleLogout}>Logout</MenuItem>
            </Menu>
        </div>
    );
}

export default function Home() {

    const [isDrawerOpen, setIsDrawerOpen] = useState(true)

    const [sprints, setSprints] = useState([])
    const [projects, setProjects] = useState([])
    const [teams, setTeams] = useState([])

    const theme = useTheme();
    const primaryColor = theme.palette.primary.main
```

```
    const homeController = new HomeController()

    // funcoes que são executadas ao carregar da view
    useEffect(() => {
        const fetchSprints = async () => {
            const sprintList = await homeController.getSprints()
            setSprints(sprintList)
        }

        const fetchTeams = async () => {
            const teamsList = await homeController.getTeamsGeneralInfo()

            setTeams(teamsList.map(team => team.teamName))
        }

        const fetchProjects = async () => {
            const projectsList = await homeController.getProjects()

            setProjects(projectsList)
        }

        fetchSprints()
        fetchProjects()
        fetchTeams()
    }, [])


    return (
        <DrawerContext.Provider value={isDrawerOpen} drawerCallback={() => setIsDrawerOpen(!isDrawerOpen)}>
            <AppBar position="fixed" sx={{ zIndex: (theme) => theme.zIndex.drawer, height: `${headerHeight}px`, justifyContent: 'center' }}>
                <Toolbar>
                    <IconButton
                        color="inherit"
                        aria-label="open drawer"
                        edge="end"
                        onClick={() => setIsDrawerOpen(!isDrawerOpen)}
                        sx={[isDrawerOpen && { display: 'none' }]}
                    >
                        <MenuIcon />
                    </IconButton>
                    <Box width={'16px'} />
```

```
                    <Box display={'flex'} width={isDrawerOpen ? '380px' : 'fit-content'} sx={{ transitionDuration: '1s' }} justifyContent={'flex-end'}>
                        <Typography variant="h5" noWrap component="div">
                            Metrics Insight
                        </Typography>
                    </Box>
                    <HomeDrawer width={drawerWidth} headerHeight={headerHeight} drawerCallback={() => setIsDrawerOpen(!isDrawerOpen)} />
                    <Box flexGrow={1} />
                    <BasicMenu />
                </Toolbar>
            </AppBar>
            <CssBaseline />
            {/* <Box boxSizing="border-box" display="flex" flexDirection="column" width="100%" sx={{ height: `calc(100vh - ${100}px)`, overflowY: "hidden" }}> */}
                <Box padding={`${headerHeight + 32}px 32px 0 32px`} height="100%" sx={{overflow: 'hidden'}}>
                    <DrawerAware>
                        <Routes>
                            <Route>
                                <Route path="about" element={<AboutPage />} />
                                <Route path="account" element={<AccountPage />} />

                                {/* projects routes */}
                                <Route path="projects" element={<Projects projects={projects} />} />
                                <Route path="projects/:projectId/tasks" element={<TaskListFromProject title="s"/>} />

                                <Route path="clients/metrics/bpc" element={<BugsPerClient />} />

                                {/* metrics routes */}

                                <Route path="metrics" element={<MetricsPage />} />
                                <Route path="metrics/agile" element={<Metrics />} />
                                <Route path="metrics/custom" element={<CostumeMetrics />} />

                                {/* sprints routes */}
                                <Route path="sprints" element={<Sprints sprints={sprints} teams={teams} />} />
                                <Route path="sprints/:number/metrics/burndown" element={<BurndownPage />} />
                                <Route path="sprints/:code/metrics/burnup" element={<BurnUpPage />} />
                                <Route path="sprints/:code/metrics/wip" element={<WIPPage />} />
                                <Route path="sprints/:number/metrics/velocity" element={<VelocityPage />} />
                                <Route path="sprints/:code/tasks" element={<TaskList />} />

                                {/* teams routes  */}

                                <Route path="teams" element={<TeamsPage teams={teams}/>} />
                                <Route path="teams/:teamName/metrics/moving-average-velocity" element={<MovingAverageVelocityPage />} />

                                {/* employees routes */}

                                <Route path="employees" element={<EmployeeListPage />} />
                                <Route path="employees/:email/full-metrics" element={<EmployeeFullInfoPage />} />

                                {/* OKRs routes */}
                                <Route path="metrics/okr" element={<OKRMetrics />} />
                                <Route path="metrics/okr/bugs-solved" element={<ClosedBugsOKR />} />
                                <Route path="metrics/okr/completed-bugs-tracker" element={<CompletedProjectsBugTrackerOKR />} />
                                <Route path="metrics/okr/completed-tasks-of-all-sprints" element={<SprintsDeliveryOKR />} />
                            </Route>
                        </Routes>
                    </DrawerAware>
                </Box>
            {/* </Box> */}
            <BottomNavigation sx={{
                position: 'fixed',
                bottom: 0,
                width: '100%',
                height: '100px',
                background: `${primaryColor}`,
                display: 'flex',
                // flexDirection: 'column',
                // alignItems: 'flex-end'
            }} showLabels >
                <BottomNavigationAction sx={{height: '80px'}} label={
                    <Box display="flex" flexDirection="column" alignItems="flex-end" flex="1" width="100vw" paddingRight="16px">
                        <Typography color="white" variant="body2">Metric Insight App v0.0.1</Typography>
                        <Typography color="white" variant="body2">© Copyright 2024 - Wesley Corporation LTDA</Typography>
                        <Typography color="white" variant="body2"> Metric Insight App v0.0.1</Typography>
                    </Box>}
                />
            </BottomNavigation>
        </DrawerContext.Provider>
    );
}
```

frontend\src\index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

```

frontend\src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();

```

docker-compose.yml

```
version: '3'



services:
  backend:
    depends_on:
      database:
        condition: service_healthy
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - ${BACKEND_PORT}:${BACKEND_PORT}
    container_name: backend_container
    env_file: .env
    # networks:
    #   - app-network

  database:
    image: postgres
    volumes:
      - ./backend/db:/docker-entrypoint-initdb.d
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "Postgres2022!"
    ports:
      - "5432:5432"
    healthcheck:
      test: pg_isready -U $$POSTGRES_USER

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    container_name: frontend
    working_dir: /usr/src/app
    volumes:
      # - ./react:/usr/src/app
      - /usr/src/app/node_modules
    tty: true
    ports:
      - "3000:3000"
    command: npm run start
```

```
    # networks:
    #   - app-network

# networks:
#   app-network:
#     driver: bridge
```