Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500
Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

SISTEMA INTEGRADO AO JIRA SOFTWARE PARA AUTOMAÇÃO DE CRIAÇÃO DE RELATÓRIOS E ENVIO DE E-MAILS[1]

Otávio Pohren[2]

Débora Cristina Engelmann[3]

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta de automação para criação e envio de relatórios do escopo de trabalho de um time de desenvolvimento *web.* Este time tem comunicação recorrente com o cliente utilizando relatórios, e estes são enviados semanalmente. Para a criação destes relatórios, é demandado tempo do profissional além de conhecimento técnico em ferramentas como o Jira Software e Excel. O tempo gasto com a criação e envio de relatórios poderia ser utilizado em uma tarefa de maior valor para o cliente. Estes são problemas levados em conta ao desenvolver a ferramenta. O sistema desenvolvido contém integração com agentes externos como Jira Software e QuickChart, e é construído com computação em nuvem utilizando o paradigma *serverless.* Também são demonstrados os resultados da aplicação deste trabalho em uma empresa de desenvolvimento de software.

Palavras-chave: automação, Jira Software, AWS, serverless.

ABSTRACT

This paper presents the development of an automation tool for creating and sending reports on the scope of work of a web development team. This team frequently communicates with the client using reports, which are sent weekly. It takes valuable time for the workers to come up with these reports, as well as technical knowledge in tools such as Jira Software and Excel. The time spent creating and submitting reports could be spent on a task of greater value to the customer. Those were issues considered when developing the tool. The developed system contains integration with external agents such as Jira Software and QuickChart, and it was built with cloud computing using the serverless paradigm. The results of using this system in a real software development company are also shown in the article.

Keywords: automation, Jira Software, AWS, serverless.

[1] Trabalho de Conclusão de Curso. Data da submissão e aprovação: 12 nov. 2022.
[2] Acadêmico do curso de Sistemas de Informação das Faculdades Integradas de Taquara – Faccat/RS. *E-mail*: otaviopohren@sou.faccat.br.
[3] Professora Orientadora do curso de Sistemas de Informação das Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil. E-mail: deboraengelmann@faccat.br

1 INTRODUÇÃO

A pandemia de COVID-19 obrigou milhares de pessoas a permanecerem em suas residências, já que segundo a Fiocruz (2020), uma forma de conter o avanço da doença é praticando o isolamento domiciliar. Com estas restrições, o trabalho remoto foi uma possível solução aplicada no mercado de trabalho. Porém, segundo Abarca et al. (2020), para que times virtuais tenham sucesso, é necessária a utilização colaborativa de tecnologia para se comunicar já que a separação geográfica existe.

Além disso, em um cenário virtual em que times de desenvolvimento de software aplicam uma metodologia ágil como Scrum, os processos se repetem devido à natureza iterativa e incremental da metodologia. Segundo Schawaber e Sutherland (2020), esta abordagem visa prever de forma mais efetiva e controlar possíveis riscos. Segundo Beck et al. (2001), o Manifesto Ágil define princípios, como: "Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo". Devido a isso, o processo de comunicação com o cliente acaba tendo uma recorrência alta.

A empresa na qual este trabalho está envolvido faz uso de relatórios gráficos para auxiliar na comunicação entre desenvolvedores e partes interessadas, como outros desenvolvedores e clientes. Tais relatórios apresentam o escopo de trabalho realizado ou prestes a se realizar, sendo assim importante para que a empresa possa monitorar o trabalho de uma equipe.

Em conversas informais com profissionais responsáveis pelos relatórios, se descobriu que a criação dos mesmos é um processo repetitivo e lento, que demanda atenção plena do responsável. Para criar os relatórios, é necessário exportar dados manualmente da ferramenta Jira Software, utilizada na empresa. Depois são criados gráficos e uma tabela para demonstrar o trabalho realizado pelo time de desenvolvimento. Só depois, estas informações são unificadas em um documento e o mesmo é encaminhado via e-mail para as partes interessadas. Todo este processo toma muito tempo do profissional que poderia ser utilizado na geração de valor. Além disso, é requerido um certo nível de conhecimento das ferramentas utilizadas para gerar o relatório.

Levando em conta o cenário atual de modelo de trabalho virtual, a necessidade recorrente de comunicação entre cliente e desenvolvedores na empresa em questão, e a dificuldade para geração destes relatórios, o presente trabalho demonstra o desenvolvimento de uma ferramenta para automatizar o processo de criação e envio

de relatórios digitais. Podendo assim, remover a responsabilidade do profissional de fazer essa comunicação. Além disso, a natureza dos relatórios é estática ao ponto de não necessitar de um trabalho criativo na produção, sendo repetitivo ao ponto de uma automatização trazer muito mais benefícios do que riscos.

O presente artigo apresenta, em ordem: referencial teórico ilustrando conceitos utilizados no sistema na Seção 2; A metodologia de desenvolvimento do software assim como as tecnologias utilizadas são apresentadas na Seção 3; os resultados obtidos com o uso do sistema na Seção 4; as considerações finais sobre o presente trabalho na Seção 5.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção são descritas as diferentes tecnologias e ferramentas utilizadas no sistema, como Jira Software, AWS (Amazon Web Service), QuickChart.io e Node.js.

2.1 Jira Software

Jira software é uma plataforma que traz um conjunto de soluções ágeis de gerenciamento de trabalho, muito útil para a colaboração entre profissionais (ATLASSIAN, 2022a). Entre suas ferramentas, o Jira possibilita a customização da ferramenta oferecendo diversos templates para trabalho, como Scrum, Kanban, DevOps entre vários outros. Kanban é um método ágil, de origem japonesa e seu nome significa "cartão de sinalização" (ANDERSON, 2010). Ainda segundo Anderson (2010), estes cartões são utilizados em um quadro kanban para sinalizar o andamento de uma tarefa. O quadro kanban é aplicado nos 3 templates citados acima, mas com variações e sucetível a customização.

O template Scrum entrega um ambiente de projeto pronto para aplicar iterações de trabalho utilizando períodos de tempo fixos, nomeados Sprints (ATLASSIAN, 2022b). Já o template de DevOps oferece integrações com diversas ferramentas, como o Opsgenie, ferramenta utilizada para monitorar a saúde de um sistema (ATLASSIAN, 2022c).

Além de fornecer ferramentas de forma gráfica, o Atlassian também oferece o Jira Agile, ferramenta que contém um conjunto de APIs (Application Programming Interface - Interface de programação de aplicativos) e que possibilita a integração

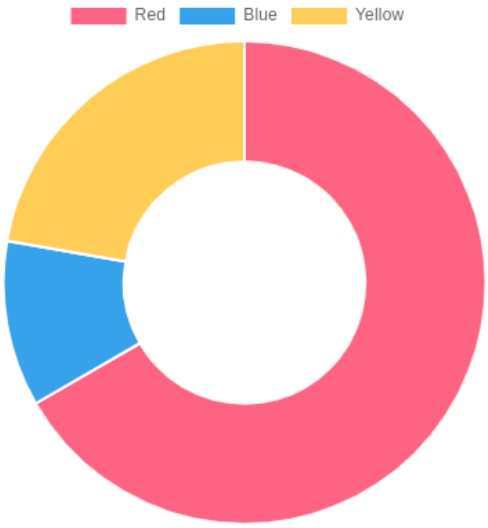
HTTP (Hypertext Transfer Protocol - Protocolo de Transferência de Hipertexto) com aplicações web. Com esta ferramenta, é possível criar e manter tarefas, épicos, sprints, quadros de trabalho, etc. (ATLASSIAN, 2022d).

O Jira ainda conta com processos de automação nos quais é possível configurar regras de execução baseado em diversos critérios de disparo, execução e condições para as regras (ATLASSIAN, 2022e).

2.2 QuickChart

QuickChart é um serviço web que gera imagens de gráficos em tempo real que é baseado na biblioteca gráfica de código aberto, Chart.js (QuickChart, 2022). O QuickChart disponibiliza uma API para realizar integrações via requisições HTTP.

Figura 1 – Gráfico gerado com quickChart



Fonte: **Chart.Js (2022).**

O gráfico de rosca apresentado na figura 1 é um exemplo de gráfico que pode ser gerado realizando uma requisição a API do quickchart. Nesta requisição, é possível definir o tipo de gráfico, cores, tamanhos, textos embutidos, etc. O QuickChart também pode gerar outros tipos de gráficos, como de linha, barras, etc.

2.3 AWS

AWS é uma plataforma de nuvem que oferece mais de 200 serviços para construção de soluções em tecnologia, serviços como bases de dados, computação

e *machine learning* (AWS, 2022a). Além disso, é oferecido um nível gratuito de uso para mais de 100 serviços, é oferecida a oportunidade de explorar e aprender diversas ferramentas da plataforma, desde que se não ultrapasse limites específicos de uso (AWS, 2022b).

Uma maneira de realizar integrações de sistemas com os serviços da AWS é utilizando SDKs ou Software Development Kit. Este tipo de ferramenta entrega um conjunto de bibliotecas que atuam como uma interface entre o desenvolvedor e a AWS e permite gerenciar credencias, reprocessamento de requisições, serialização e deserialização de dados, etc (SDK, 2022).

O S3 (Simples Storage Service ou Serviço de Armazenamento Simples) é um serviço AWS que oferece armazenamento de objetos variados com a capacidade de escalabilidade (S3, 2022). Segundo a documentação do S3, é possível armazenar qualquer volume de dado para utilizar em aplicações como sites, aplicações *mobile*, análise de dados, etc (S3, 2022). S3 funciona utilizando containers de objetos, estes containers são nomeados buckets. Dentro de cada bucket é possível armazenar objetos, sendo objeto um arquivo e qualquer metadado que descreva este arquivo (S3, 2022). O metadado de um arquivo pode ser a sua data de modificação, por exemplo.

DynamoDB é um banco de dados noSQL disponível no ecossistema AWS que entrega escalabilidade, mas sem necessitar de provisionamento, replicação e configuração de hardware (DYNAMODB, 2022).

Função Lambda é um serviço AWS que permite executar código com escalabilidade e disponibilidade sem a necessidade de provisionar ou manter servidores. Além disso, o Lambda possibilita o registro de log da execução (LAMBDA. 2022). É possível construir funções utilizando linguagens como Java, Javascript, .NET,

2.4 Backend

Esta seção apresenta conceitos necessários para a compreensão do processo interno da aplicação, da qual o usuário não tem acesso direto. Os conceitos vão desde comunicação entre sistemas até a arquitetura da aplicação.

2.4.1 Node

Segundo Cantelon et al. (2014), Node.js é um ambiente de execução de código baseado no motor javascript V8 de código aberto, pertencente a Google. Com a utilização do motor V8, se tem um ganho de performance ao ignorar o processo de interpretação de código e partir direto para a compilação (CANTELON, 2014), porém, fora do navegador. Segundo a documentação de Node.js (2022), este ambiente de execução foi idealizado para construir projetos escaláveis, já que não cria novas *threads* para lidar com novas requisições. Devido a isso, Node.js trabalha de forma assíncrona, não utilizando processamento enquanto aguarda a resolução de uma consulta em banco ou um sistema de arquivos (NODE, 2022). Só quando a requisição obtém resposta é que o Node retoma o processamento. Devido a isso, são utilizados menos ciclos de processamento, facilitando o escalonamento (NODE, 2022).

2.4.2 Typescript

TypeScript oferece todas as habilidades de Javascript com uma camada adicional que é o sistema de tipagem forte. Javascript fornece tipagens primitivas como string e number, mas que não são verificadas pela linguagem. O Typescript é capaz de realizar estas verificações (TYPESCRIPT. 2020). A documentação do Typescript (2020) também afirma que o principal benefício desta linguagem é ressaltar comportamentos inesperados no código, diminuindo a incidência de bugs.

2.4.3 Comunicação e integração

JSON (JavaScript Object Notation) é um padrão de sintaxe baseado em JavaScript. É utilizado para definir o formato de um dado para que seja possível a comunicação entre diferentes sistemas e, portanto, não depende de um sistema baseado em JavaScript para ser utilizado (BASSETT. 2015).

Para realizar a comunicação entre sistemas, é utilizado o protocolo HTTP. Segundo Fielding et al. (1999) HTTP é um protocolo de nível de aplicação utilizado por sistemas distribuídos e colaborativos. Ele define regras gerais para comunicação, nomenclatura, validação, autenticação e entre outros.

2.4.4 Serverless

Segundo a documentação de Serverless Framework (2022a), esta é uma ferramenta de linha de comando de código aberto que facilita o desenvolvimento, implementação e solução de problemas com aplicações que utilizam o paradigma sem servidor. É possível utilizar a ferramenta para implantar sistemas serverless em diversas nuvens de processamento no mercado, entre elas, a AWS (SERVERLESS, 2022a).

Com Serverless Framework, é possível manter tanto o código do sistema como também a sua infraestrutura. Podendo criar, por exemplo, *buckets* no S3, funções Lambda e tabelas no DynamoDB. Para isso, é necessário especificar tais estruturas em um arquivo de texto com sintaxe YAML (YAML Ain't Markup Language) (SERVERLESS, 2022b). YAML é uma sintaxe que possibilita a leitura humana e é independente de linguagem de programação (FILEINFO, 2022).

2.5 Trabalhos Relacionados

Ao realizar a pesquisa para a execução deste projeto, foram encontradas ferramentas semelhantes à do trabalho, mas nenhuma que contemplasse todas as necessidades da empresa.

Um exemplo é a Chart Macro, produto Atlassian oferecido sem custo adicional, que tem a capacidade de criar diversos gráficos, similares aos desenvolvidos neste trabalho. Porém, não é capaz de gerar outros gráficos mais complexos como os desenvolvidos neste trabalho, além de não ser capaz de exportar estes dados via e-mail (CONFLUENCE, 2022).

Já a ferramenta Custom Charts for Jira Reports and Jira Dashboard Filter é mais robusta para a criação de imagens, mas também não faz envio de informação por e-mail. Esta ferramenta é disponibilizada no *Marketplace* da Atlassian, e tem custo a partir de 10 usuários (ATLASSIAN, 2022f).

Outra ferramenta disponível no Marketplace da Atlassian é a Email this Page. Esta ferramenta adiciona um novo botão ao Confluence e possibilita enviar uma página de confluence por e-mail. Essa solução não gera gráficos, portanto, necessitaria de uma combinação de demais ferramentas para ter uma solução próxima do ideal (ATLASSIAN, 2022g).

A tabela 1 demonstra um comparativo entre as ferramentas citadas acima. É possível visualizar que nenhuma ferramenta atende todas as necessidades que o

projeto deste trabalho atende. Além disso, duas ferramentas têm custo para serem utilizadas.

Tabela 1 – Comparativo entre soluções similares

Funcionalidade	Presente trabalho	Chart Macro	Custom Charts for Jira Reports and Jira Dashboard Filter	Email this Page
Criação de gráficos	**X**	**X**	**X**	
Criação de tabelas	**X**		**X**	
Envio de e-mail	**X**			**X**
Gratuita	**X**	**X**		

Fonte: **Elaborada pelo autor (2022).**

3 METODOLOGIA

Esta seção discorre sobre as metodologias utilizadas na análise, desenvolvimento e entrega do sistema. Além disso, são citadas as tecnologias utilizadas para auxiliar nestas atividades.

3.1 Metodologia de pesquisa e desenvolvimento

A metodologia escolhida para pesquisa e construção do sistema é o Scrum Solo. Esta metodologia é uma adaptação que une as boas práticas do Scrum e do PSP para criar um método utilizado em desenvolvimento individual (Pagotto, 2016). Com Sprints semanais, foi possível realizar entregas menores de análise e desenvolvimento, com uma frequência saudável para acompanhar a evolução do sistema e definir ações para responder a mudanças de escopo.

Foi criado um repositório de projeto para armazenar os artefatos do projeto. O primeiro artefato criado na interação entre cliente e desenvolvedor é o escopo de projeto (Pagotto, 2016). Toda documentação referente ao sistema ficou armazenada neste repositório em nuvem.

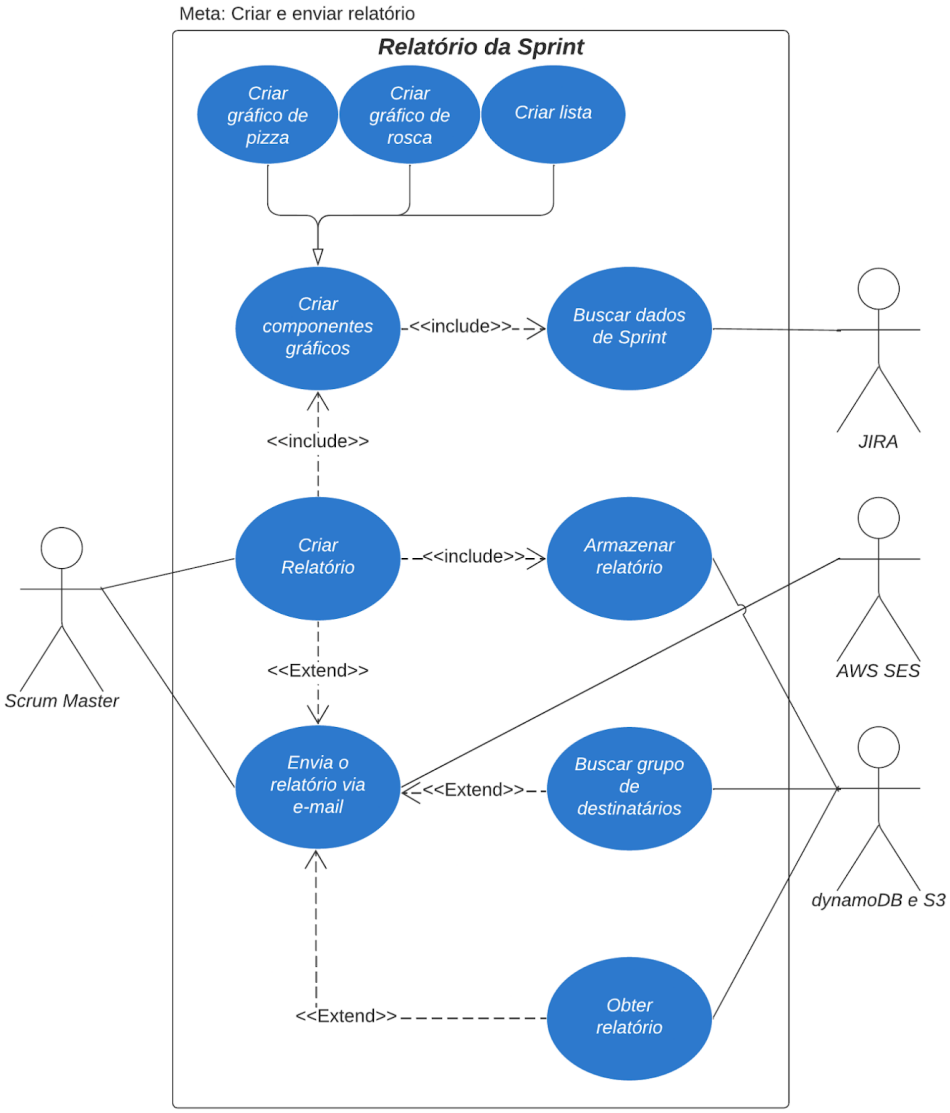
Também foi criado um backlog de produto, um armazém descrevendo as funcionalidades do sistema que devem ser implementadas. O product backlog faz parte do repositório de projeto.

Além disso, foi criado um cronograma em forma de gráfico de Gantt para demonstrar as atividades necessárias desde a análise até desenvolvimento e escrita do trabalho. Tais atividades são descritas de forma sequencial, demonstrando a relação de dependência entre elas. Foi utilizado o método Kanban para organizar as atividades de forma gráfica em um quadro.

3.2 Levantamento de casos de usos e modelagem do sistema

A figura 2 representa o principal caso de uso do sistema. O Scrum Master ou responsável pelos relatórios solicita a criação de um relatório e também o envio do mesmo via e-mail. Um relatório é composto por texto e componentes gráficos, sendo estes os gráficos de pizza e rosca e uma lista de todas as tarefas da sprint. Também é possível ver os agentes externos do sistema, como o Jira, o AWS SES (Simple Email Service) e o DyamoDB e S3.

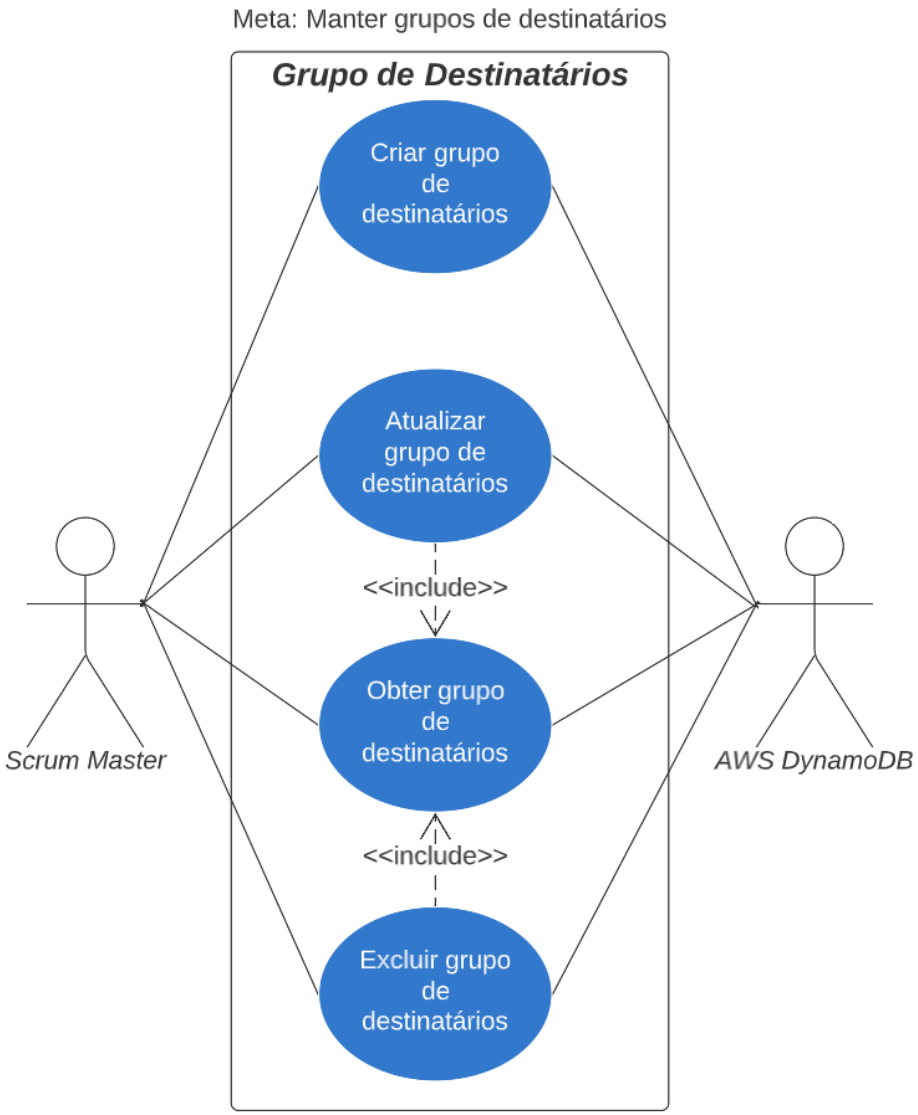
Figura 2 – Caso de Uso geral



Fonte: **Elaborada pelo autor (2022).**

O envio de relatórios pode ser dependente de uma busca por um grupo de destinatários. É possível criar este grupo, informando quais endereços de e-mail irão receber o relatório. Assim, esta configuração precisa ser feita apenas uma vez, sendo um fluxo de suporte e auxílio para o caso de uso principal. Este fluxo de suporte pode ser visto na figura 3.

Figura 3 – Caso de uso manter grupos de destinatários



Fonte: **Elaborada pelo autor (2022).**

A criação assim como a manutenção de grupos de destinatários depende de 4 funcionalidades: A criação, atualização, busca e exclusão de grupos. Todos estes dados devem ser armazenados pelo agente externo AWS DynamoDB, responsável por manter os dados.

3.3 Entrega

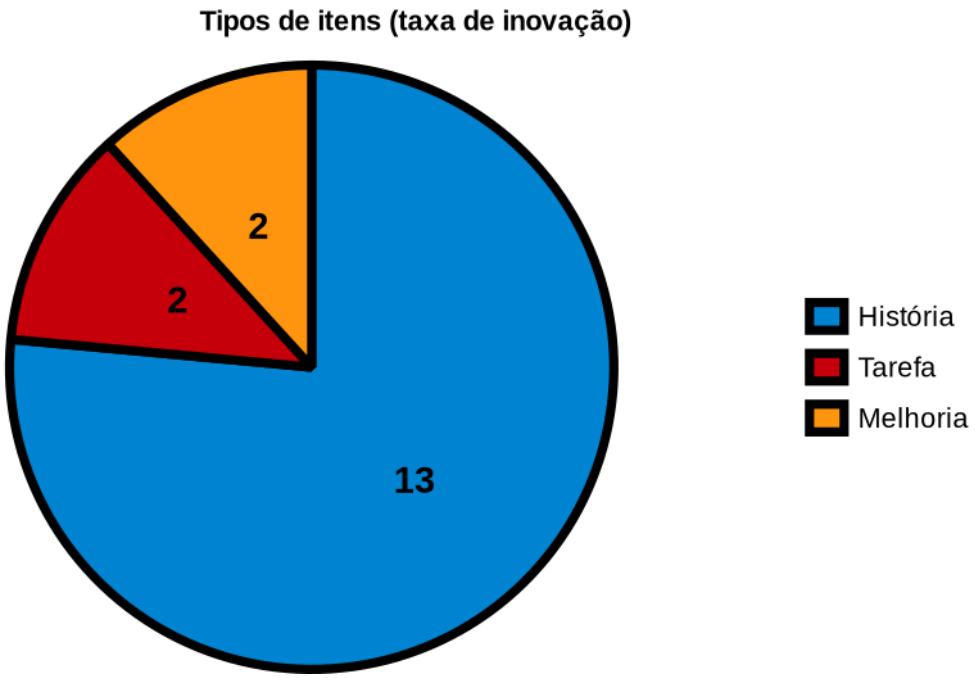
Após realizada a análise do sistema, foi definido como entrega um software capaz de integrar ao Jira Software e obter dados de desenvolvimento de software de um time em específico. Com estes dados, gerar gráficos e uma tabela como

componentes de um relatório de escopo de trabalho. Posteriormente, armazenar este relatório e fazer o envio do mesmo via e-mail para as a partes interessadas.

Para compor o relatório de início e fim de *sprint*, são utilizados gráficos de pizza e de rosca. Além disso, é criada uma tabela detalhando melhor cada tarefa da sprint.

O gráfico de pizza retratado na figura 4 tem o objetivo de demonstrar os tipos de tarefas desempenhadas por um time de desenvolvimento. Esta informação é necessária para que as partes interessadas tenham ciência do quanto de uma entrega são novas funcionalidades, correções de bugs, refatoração de código, etc. Além da imagem, é calculada a porcentagem de tarefas do tipo história na *sprint*.

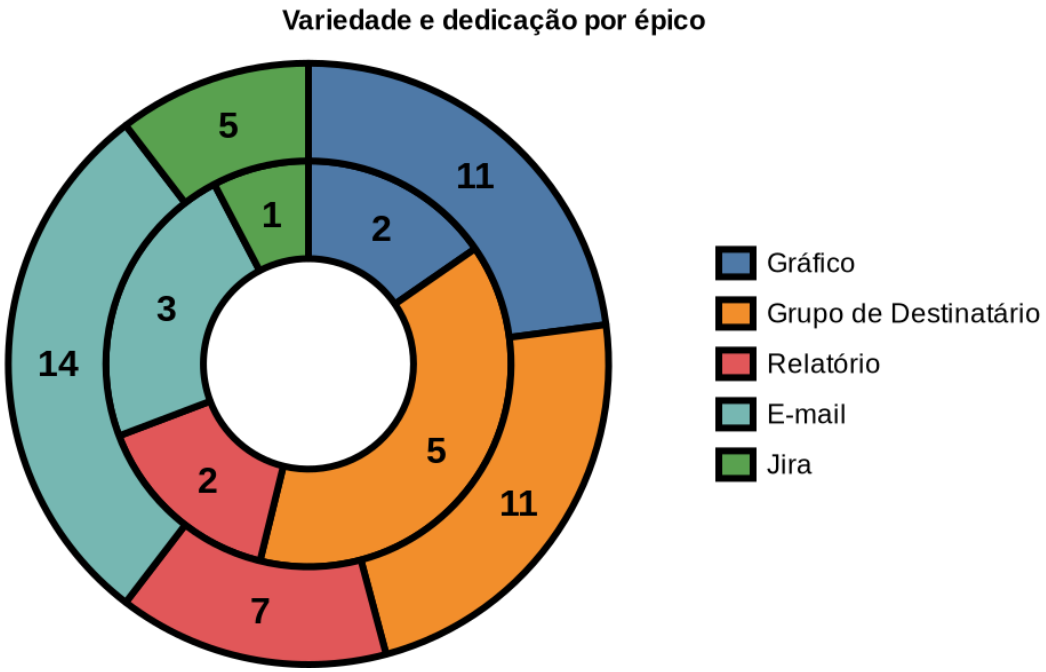
Figura 4 – Gráfico de pizza gerado com quickChart



Fonte: **Elaborada pelo autor (2022).**

Já o gráfico na figura 3 visa demonstrar a qual épico ou domínio cada tarefa pertence (círculo interno), além de demonstrar a dedicação por épico (círculo externo). As unidades do círculo interno são a soma de números de tarefas por épico e as unidades do círculo externo são a soma de pontos de esforço por épico. O esforço de cada tarefa é estimado a partir de pontos de história. Segundo Coelho e Basu (2012), ponto de história é uma unidade utilizada para definir o tamanho de uma tarefa levando em conta a complexidade de desenvolvimento, o esforço e o risco envolvido na tarefa.

Figura 5 – Gráfico de rosca gerado com quickChart



Fonte: **Elaborada pelo autor (2022).**

Uma tabela como a tabela 1 é inclusa no relatório e seu objetivo trazer demais informações sobre todas as tarefas incluídas em uma sprint, como o título, tipo, a qual épico pertente, etc.

Tabela 2 – Tabela listando todas as tarefas de uma sprint

Chave	Resumo	Tipo de Item	Epic	Status	Pontos de história
TCC-3	Novo endpoint criar relatório	História	Gráfico	DONE (Produção)	9
TCC-23	Novo endpoint: Criar grupo de destinatários	História	Grupo de Destinatário	DONE (Produção)	4
TCC-20	Criar Tabela tbDestination	História	Grupo de Destinatário	DONE (Produção)	2
TCC-19	Criar tabela tbReports	História	Relatório	DONE (Produção)	2
TCC-11	Criar bucket no S3	História	Gráfico	DONE (Produção)	2
TCC-29	Criar Documentação Postman	Tarefa	-	DONE (Produção)	1
TCC-8	Novo endpoint: Enviar e-mail	História	E-mail	DONE (Produção)	8

TCC-4	Criar e-mail dinâmico com imagens e lista	História	E-mail	DONE (Produção)	3
TCC-7	criar template de e-mail html/css	História	E-mail	DONE (Produção)	3
TCC-5	novo endpoint Listar sprints	História	Jira	DONE (Produção)	5
TCC-9	criar projeto api-report	História	Relatório	DONE (Produção)	5
TCC-27	Criar Casos de uso	Tarefa	-	DONE (Produção)	2

Fonte: **Elaborada pelo autor (2022).**

Tanto os gráficos como a tabela citados acima, foram gerados utilizando o sistema, com dados obtidos de um quadro Kanban no Jira utilizado para desenvolver a aplicação.

3.5 Tecnologias utilizadas

Nas fases de análise se fez uso do Lucidchart para criar e documentar os casos de uso do projeto, armazenar em nuvem e utilizar como um repositório de projeto. LucidChart é uma ferramenta multiplataforma focada na criação de diagramas que podem ser construídos de forma colaborativa (LUCIDCHART, 2022).

Foi utilizado o editor de código fonte Visual Studio Code para a codificação do sistema. Este editor é leve, mas poderoso, apresentando suporte nativo para Javascript, Typescript e o ambiente de execução Node.js, além de oferecer um amplo sistema de extensões para demais linguagens (VISUAL STUDIO, 2022).

Para construir, testar e utilizar o sistema foi utilizado o Postman, uma ferramenta capaz de armazenar e gerenciar requisições, documentação, casos de testes, etc (POSTMAN, 2022). Neste trabalho, o Postman foi utilizado para realizar requisições HTTP e armazenar coleções de requisições, podendo assim guardar diferentes fluxos de uso e agilizar a construção do sistema.

Para garantir o versionamento e *back-up* da aplicação, foi utilizada a ferramenta de repositório em nuvem Bitbucket. O Bitbucket utiliza Git (GIT, 2022) e sendo um produto Atlassian, tem integração com o Jira de forma nativa (BITBUCKET, 2022).

4 RESULTADOS

Este sistema foi aplicado em uma empresa de desenvolvimento *web*, em uma estrutura de times funcionais dependentes de um time de *backend*. O relatório traz valor de informação tanto para outros times de tecnologia, quanto para o setor executivo da empresa.

Conversas foram realizadas com ambas as partes a fim de coletar resultados, o time usuário da automação e os leitores interessados no relatório. Foram apontados benefícios diferentes, de acordo com os interesses. A primeira parte apontou resultados como economia de tempo, menos erros e menor responsabilidade. Já os leitores do relatório valorizaram a assiduidade de envio e também a pontualidade.

4.1 Benefícios

Com o uso do sistema, a criação dos relatórios é praticamente instantânea, já que não necessita de nenhum envolvimento humano para gerar gráficos e tabelas ou sequer um processo de escrita. Sendo assim, o profissional tem uma economia de tempo e pode trabalhar com outras tarefas de maior valor para o cliente.

Por mais que tenha ocorrido uma automação, não é necessário conhecimento técnico sobre a mesma para utilizar o sistema desenvolvido, já que o próprio Jira dá início ao processo. A única necessidade é configurar um evento de automação no Jira para criar um evento que consuma o sistema. Tendo este evento criado, é necessário apenas criar e finalizar sprints e esta atividade já é suficiente para criar os relatórios.

Com este acionamento automático, não há a necessidade de um responsável para criação e envio do relatório. Portanto, não é necessário aprender conhecimentos específicos para gerar o relatório e a empresa se protege da perda de conhecimento devido rotatividade de profissionais.

Com a automação aplicada, o envio de relatórios é assíduo além de pontual, já que é enviado em toda abertura e fechamento de *sprint*. O time de desenvolvimento comentou que nem sempre é possível criar e enviar um relatório, pois em algumas ocasiões há demandas mais urgentes para atender e que com a automação, este envio é garantido. De acordo com os leitores, a pontualidade de envio do relatório é valorizada pois cria um processo previsível com envios sempre no mesmo dia da semana.

5 CONCLUSÃO

Visto a dificuldade de um time de desenvolvimento para criar relatórios de *sprints* e comunicar a demais times o seu escopo de trabalho, foi idealizado um sistema no qual seria possível automatizar esta criação de relatórios com o intuito de tornar a comunicação do time mais assídua e recorrente.

Com a utilização deste sistema, é possível criar um relatório com gráficos, textos e tabela, contendo informações sobre o trabalho realizado na *sprint* passada, ou prestes a se realizar na *sprint* atual. Assim, há a oportunidade de exercer a transparência entre diferentes setores da empresa.

Além disso, esta automação remove a responsabilidade de um integrante do time de criar os relatórios, visto que o processo se torna independente de usuários, necessitando apenas de um gatilho vindo do Jira. Sendo assim benéfico para a empresa, já que com a rotatividade de profissionais, um trabalho como a criação de relatórios pode ser negligenciado ou feito de forma incorreta.

Também foi obtido um resultado positivo de acordo com os profissionais de operação e de executivo da empresa, que alegaram valorizar a assiduidade e pontualidade dos relatórios.

Sugere adicionar novos recursos ao sistema, como a possibilidade de inserir o *link* de cada tarefa embutido na coluna de chave da tabela. Assim, o leitor pode acessar um item em específico direto no Jira. Além disso, adicionar um trecho de texto para listar as tarefas prioritárias de cada sprint. Para isto, pode ser utilizado o recurso de categorias no Jira para designar tarefas prioritárias. Também sugere-se realizar uma melhoria gráfica para gerar gráficos com o fundo de imagem transparente, para que fique uniforme ao utilizar um cliente de e-mail com tema escuro.

Esta ferramenta poderá ser disponibilizada para que outros times possam utilizar. Idealmente uma única função pode atender diversos quadros Jira. Entretanto, para que isso seja possível, todos os quadros devem ter a mesma configuração. Devido a isso, o ideal é que sejam criadas novas funções específicas para cada quadro, evitando erros e comportamentos inesperados. Também é sugerido adicionar uma camada de autenticação no sistema, para que a segurança seja reforçada e nenhum dado vazado.

Levando em conta os benefícios de aplicação da automação, a possibilidade de expansão e melhorias para o sistema, a mesma continuará sendo aplicada na empresa em questão.

REFERÊNCIAS

ABARCA, Victor Garro; SANCHEZ, Pedro Palos; CAMACHO, Mariano Aguayo. **Virtual Teams in Times of Pandemic: Factors That Influence Performance.** Disponível em: https://www.frontiersin.org/articles/10.3389/fpsyg.2021.624637. Acesso em: 9 nov. 2022.

ANDERSON, David J.; **Kanban: Successful Evolutionary Change for Your Technology Business**. 1d. Sequim, Washington. Blue Hole Press. 2010.

ATLASSIAN. **Visão geral do Jira.** Disponível em: https://www.atlassian.com/br/software/jira/guides/getting-started/overview. Acesso em: 02 set. 2022a.

ATLASSIAN. **Scrum template.** Disponível em: https://www.atlassian.com/software/jira/templates/scrum. Acesso em: 09 nov. 2022b.

ATLASSIAN. **DevOps template** Disponível em: https://www.atlassian.com/software/jira/templates/devops. Acesso em: 09 nov. 2022c.

ATLASSIAN. **Rest API**. Disponível em https://developer.atlassian.com/cloud/jira/software/rest/intro. Acesso em: 09 nov. 2022d.

ATLASSIAN. **Automação: noções básicas**. Disponível em: https://www.atlassian.com/br/software/jira/guides/expand-jira/automation. Acesso em: 02 set. 2022e.

ATLASSIAN. **Custom Charts for Jira Reports and Jira Dashboard Filter.** Disponível em: https://marketplace.atlassian.com/apps/1220925/custom-charts-for-jira-reports-and-jira-dashboard-filter. Acesso em: 09 nov. 2022f.

ATLASSIAN. **Email this Page.** Disponível em: https://marketplace.atlassian.com/apps/1217983/email-this-page. Acesso em: 9 nov. 2022g.

AWS. **O que é AWS? Como funciona Amazon Web Services.** Disponível em: https://aws.amazon.com/pt/what-is-aws. Acesso em: 9 out. 2022a.

AWS. **Nivel gratuito da AWS.** Disponível em: https://aws.amazon.com/pt/free. Acesso em: 9 out. 2022b.

BASSETT, Lindsay. **Introduction to JavaScript Object Notation.** O'Reilly. 2015. P.2.

BECK, Kent. et al. **Princípios por trás do Manifesto Ágil**. Disponível em: https://agilemanifesto.org/iso/ptbr/principles.html. Acesso em: 16 de nov. 2021.

BITBUCKET. **A brief overview of Bitbucket**. Disponível em: https://bitbucket.org/product/guides/getting-started/overview. Acesso em: 11 out. 2022.

CANTELON, Mike; HARTER, Marc; HOLOWAYCHUK, T. J.; RAYLICH, Nathan. Welcome to Node.js. **Node.js in action**. Manning. 2014. P.4.

COELHO, Evita; BASU, Anirban. **Effort Estimation in Agile Software Development using Story Points.** Disponível em: https://www.researchgate.net/publication/306391610_Effort_Estimation_in_Agile_Software_Development_using_Story_Points. Acesso em: 02 nov. 2022.

CONFLUENCE. **Chart Macro**. Disponível em: https://confluence.atlassian.com/doc/chart-macro-163415075.html. Acesso em: 9 nov. 2022.

CHART.JS. **Doughnut and Pie.** Disponível em: https://www.chartjs.org/docs/2.8.0/charts/doughnut.html. Acesso em: 12 set. 2022.

DYNAMODB. **O que é o Amazon DyanmoDB?** Disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/Introduction.html. Acesso em: 9 out. 2022.

FILEINFO. **.YML File Extension**. Disponível em: https://fileinfo.com/extension/yml. Acesso em: 9 nov. 2022.

FIELDING, R. et al. **Hypertext Transfer Protocol -- HTTP/1.1**. Disponível em: https://www.hjp.at/doc/rfc/rfc2616.html#sec_9. Acesso em: 18 out. 2022.

FIOCRUZ. **Plano de contigência da Fiocruz diante da pandemia da doença pelo SARS-CoV2 (COVID-19)**. Disponível em: https://portal.fiocruz.br/sites/portal.fiocruz.br/files/documentos/plano_de_contingencia_fiocruz_covid19_2020-03-13_v1-1.pdf. Acesso em: 9 nov. 2022.

GIT. **Git.** Disponível em: https://git-scm.com/. Acesso em: 11 out. 2022.

LAMBDA. **O que é o AWS Lambda?** Disponível em: https://docs.aws.amazon.com/pt_br/lambda/latest/dg/welcome.html. Acesso em: 22 set. 2022.

LUCIDCHART. **Diagram. Collaborate. Visualize data. All in one platform**. Disponível em: https://www.lucidchart.com/pages/product. Acesso em: 31 out. 2022.

NODE. **Introduction to Node.js.** Disponível em: https://nodejs.dev/en/learn/. Acesso em: 11 nov. 2022.

PAGOTTO, Tiago; FABRI, José Augusto; L´Erario. **Scrum solo**. 2016 Disponível em: https://engenhariasoftware.files.wordpress.com/2016/04/scrum-solo.pdf. Acesso em: 31 out. 2022.

POSTMAN. **What is Postman?** https://www.postman.com/product/what-is-postman/. Acesso em: 22 set. 2022.

QUICKCHART. QuickChart Documentation. Disponível em https://quickchart.io/documentation/. Acesso em: 12 set. 2022.

SDK. **AWS SDK for Javascript**. Disponível em: https://aws.amazon.com/pt/sdk-for-javascript/. Acesso em: 10 out. 2022.

SERVERLESS. **Serverless Framework Documentation**. Disponível em: https://www.serverless.com/framework/docs. Acesso em: 10 out. 2022a.

SERVERLESS. **Serverless Framework Concepts**. Disponível em: https://www.serverless.com/framework/docs/providers/aws/guide/intro. Acesso em: 10 out. 2022b.

SCHWABER, Ken; SUTHERLAND, Jeff. **O Guia do Scrum**: O Guia Definitivo para o Scrum: As Regras do Jogo. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf. Acesso em: 17 de nov. 2021. p.11.

S3. **O que é o Amazon S3?** Disponível em https://docs.aws.amazon.com/pt_br/AmazonS3/latest/userguide/Welcome.html. Acesso em: 26 de set. 2022.

TYPESCRIPT. **TypeScript for JavaScript Programmers.** Disponível em: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Acesso em: 10 out. 2022.

VISUAL STUDIO. **Documentation for Visual Studio Code.** Disponível em: https://code.visualstudio.com/docs. Acesso em: 31 out. 2022.

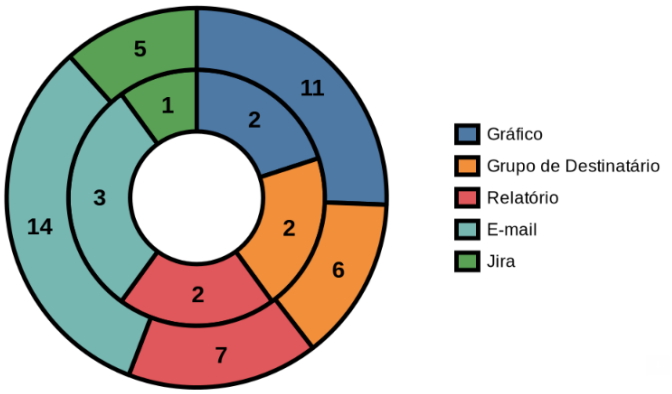
APÊNDICE A

Figura 6 – Exemplo de relatório completo.

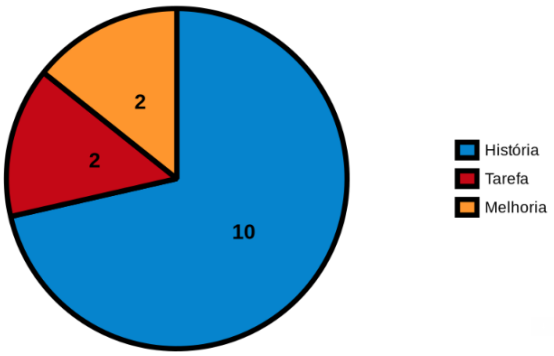
Prezados(as),

a review da Sprint Nome da Sprint ocorreu na última sexta-feira. Este e-mail contém um resumo do que foi apresentado na review e entregue na Sprint:

Variedade e dedicação por épico



Tipos de itens (taxa de inovação)



A sprint conteve um percentual de 83% de cards de história, sendo demais cards para bugs, melhorias ou tarefas (scripts, levantamento de dados, etc).

Lista completa de cards entregues:

Chave	Resumo	Tipo de Item	Epic	Status	Pontos de história
TCC-3	Novo endpoint criar relatório	História	Gráfico	DONE (Produção)	9
TCC-23	Novo endpoint: Criar grupo de destinatários	História	Grupo de Destinatário	DONE (Produção)	4
TCC-20	Criar Tabela tbDestination	História	Grupo de Destinatário	DONE (Produção)	2
TCC-19	Criar tabela tbReports	História	Relatório	DONE (Produção)	2
TCC-11	Criar bucket no S3	História	Gráfico	DONE (Produção)	2
TCC-29	Criar Documentação Postman	Tarefa	-	DONE (Produção)	1
TCC-8	Novo endpoint: Enviar e-mail	História	E-mail	DONE (Produção)	8
TCC-4	Criar e-mail dinâmico com imagens e lista	História	E-mail	DONE (Produção)	3
TCC-7	criar template de e-mail html/css	História	E-mail	DONE (Produção)	3
TCC-5	novo endpoint Listar sprints	História	Jira	DONE (Produção)	5
TCC-9	criar projeto api-report	História	Relatório	DONE (Produção)	5
TCC-27	Criar Casos de uso	Tarefa	-	DONE (Produção)	2

Este e-mail foi gerado de forma automatizada. Entre em contato para sugerir melhorias ou informar possiveis erros.

Fonte: **Elaborada pelo autor (2022).**