IZABELLE RAMOS TOMÉ

ANÁLISE DE QUALIDADE DE SOFTWARE EM UM ECOSSISTEMA DE PROCESSAMENTOS DE MOVIMENTAÇÕES FINANCEIRAS

LAVRAS - MG

2023

IZABELLE RAMOS TOMÉ

ANÁLISE DE QUALIDADE DE SOFTWARE EM UM ECOSSISTEMA DE PROCESSAMENTOS DE MOVIMENTAÇÕES FINANCEIRAS

Relatório de Estágio apresentado à Universidade Federal de Lavras como parte das exigências do curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Prof. Dr. Ramon Gomes Costa
Orientador

LAVRAS - MG

2023

IZABELLE RAMOS TOMÉ

ANÁLISE DE QUALIDADE DE SOFTWARE EM UM ECOSSISTEMA DE PROCESSAMENTOS DE MOVIMENTAÇÕES FINANCEIRAS

Relatório de Estágio apresentado à Universidade Federal de Lavras como parte das exigências do curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADA em 21 de Novembro de 2023.

Prof. Dr. Ramon Gomes Costa	UFLA
Prof. Dra. Renata Teles Moreira	UFLA
Prof. Dr. Paulo Afonso Parreira Junior	UFLA

Prof. Dr. Ramon Gomes Costa
Orientador

LAVRAS - MG
2023

AGRADECIMENTOS

Agradeço a minha família, em especial minha mãe Neuza e minha avó Ana por serem grandes referências de força e resiliência. Agradeço aos colegas que conheci nesses anos de graduação que me ajudaram a superar os desafios. Agradeço aos professores que lecionaram e me instruíram durante os anos de gradução, em especial, ao meu orientador Ramon. E por fim, agradeço aos meus colegas da empresa Equals, que me ensinaram a ser uma profissional mais preparada.

RESUMO

O presente trabalho tem como objetivo descrever algumas das atividades realizadas durante um período de estágio supervisionado na empresa *Equals*. Nesse contexto, a estagiária desempenhou funções relacionadas à análise de qualidade de *software*, com foco especial no projeto Venda Interna, responsável pela geração de relatórios bancários. No decorrer deste relatório, serão detalhadas as atividades executadas, as tecnologias e procedimentos empregados, bem como a relação dessas experiências com os conhecimentos adquiridos ao longo do curso de Sistemas de Informação.

Palavras-chave: Estágio; Análise de Qualidade; Dados Bancários; Relatórios; Padrão de Testes.

SUMÁRIO

1 Introdução 5
1.1 Sobre a empresa 5
2 Tecnologias utilizadas 7
2.1 Scrum 7
2.2 Jira Software 8
2.3 API REST 8
2.4 API SOAP 8
2.5 Swagger 9
2.6 JavaScript 9
2.7 Puppeteer 9
2.8 Postman 10
2.9 Testes de Software 10
3 Atividades Desenvolvidas 12
3.1 Integração ao local de estágio 12
3.2 Ritos do Scrum 12
3.3 Controle de versões 13
3.4 Planos de teste 13
3.5 Venda Interna 14
3.5.1 Tecnologias e atividades realizadas 16
3.6 Conciliador 19
3.6.1 Tecnologias e atividades realizadas 19
3.7 Banking 21
3.7.1 Tecnologias e atividades realizadas 21
3.8 Relação teoria e prática 27
4 Considerações Finais 28
REFERÊNCIAS 29

1 INTRODUÇÃO

Com o avanço das tecnologias, o acesso das pessoas à Internet tem se tornado cada vez maior e constante (GOUVEA, 2022). Essa massa de usuários movimenta o mercado de *software*, que para proporcionar sistemas capazes de suportar o tráfego de dados necessita de grande quantidade de profissinais de Tencologia da Informação (TI) (CNN, 2021). Os sistemas desenvolvidos por esses especialistas podem atender diversas necessidades e possuem inúmeras formas de serem implementados e disponibilizados ao usuário.

O curso de Sistemas de Informação tem como função preparar os estudantes para adentrar no mercado da tecnologia com conhecimentos sobre desenvolvimento e qualidade de *software* (ICET, 2023). Apesar de o conteúdo do curso oferecer grande conhecimento, ele deve ser considerado um pilar para que o profissional a ser construído aprimore suas habilidades e progrida nesse ramo após a conclusão do seu bacharelado.

Ao escolher por um estágio, a estudante pôde transformar a teoria discutida em sala de aula em prática, com os imprevistos e particularidades que a participação no mercado da tecnologia pode oferecer. Ao realizar as missões propostas, o conhecimento adquirido foi notável e a rotina com profissionais da área gerou frutos sociais e intelectuais.

Este relatório de estágio tem como finalidade descrever a vivência em um estágio supervisionado na área de análise de qualidade de *software*, que ocorreu durante o período 09/03/2020 a 07/12/2021. A estagiária exerceu papel de analista de qualidade, fazendo testes nas demandas, buscando melhorias, otimizando processos, gerando relatórios de teste e de *deploys*, além de participar de ritos e gestão dos projetos e tarefas.

1.1 Sobre a empresa

O estágio foi realizado na empresa *Equals* - Gestão Financeira Inteligente [1]. A empresa atua no ramo de *softwares* para conciliação bancária e gestão financeira de empresas. Atualmente, a empresa conta com cerca de 100 colaboradores e possui três escritórios, dois em São Paulo - SP

[1] https://www.equals.com.br/

e um na cidade de Lavras - MG. Sua organização empresarial utiliza de métodos ágeis para dirigir os projetos. Os principais produtos fornecidos pela empresa são: Raio-X, *Equals Core* e *Banking*.

A empresa se divide em diversos times que podem variar entre 5 a 8 membros. Suas divisões de cargo são: *Product Owner (PO)*, responsável por capturar as necessidades do cliente e aprovar a versão final de evoluções e melhorias; Analista de Negócios, responsável por capturar e especificar projetos futuros; Desenvolvedor, responsável por construir e dar manutenção ao *software*; Analista de Qualidade, responsável por testar a aplicação, encontrando *bugs* e melhorias. Os times contam com a utilização do método *Scrum* para o prosseguimento das atividades. Faz parte desse método, o planejamento do projeto, discussões sobre as demandas, desenvolvimento e testes, entrega e retorno do cliente.

Este relatório está organizado da seguinte forma: no Capítulo 2, serão apresentadas as tecnologias que a estagiária utilizou no seu cotidiano e que foram apoio para a realização das atividades propostas; no Capítulo 3, serão apresentadas as atividades desenvolvidas durante o estágio, os ritos comuns a todos os projetos da empresa e os sistemas nos quais a aluna atuou; e no Capítulo 4, será apresentada a relação com o curso de Sistemas de Informação, assim como as conclusões que a estagiária alcançou após refletir sobre o período do estágio.

2 TECNOLOGIAS UTILIZADAS

Neste capítulo, serão apresentadas e detalhadas as tecnologias utilizadas durante o período do estágio, como *frameworks*, linguagens de programação e ferramentas que apoiaram os ciclos de teste diários. Ainda que algumas das tecnologias mencionada não tenham sido tópico em aulas durante o curso, elas foram de ajuda para o ganho de conhecimento que tive durante os meses atuando como estagiária.

2.1 Scrum

O *Scrum* [1] (SCHWABER, 2020) é um método cuja proposta baseia-se no processo de desenvolvimento e manutenção de *software* em pequenos ciclos, com reuniões constantes para alinhamento de toda a equipe da evolução das tarefas (PAULA, 2016). Em parceria com um quadro *Kanban*[2], ele possibilita que toda a equipe envolvida no desenvolvimento da aplicação acompanhe o projeto em tempo real. Ele é um método cuja característica principal são os seus ciclos ou etapas, que são: as *sprints*, que são o período onde as tarefas serão feitas; as *plannings*, ou reuniões de planjamento, que se caracterizam por uma reunião do time todo para que as tarefas futuras sejam apresentadas e discutidas; *daily*, ou reunião diária, que é uma reunião de poucos minutos todos os dias, onde os membros do time falam de forma sucinta sobre as tarefas trabalhadas e problemas não planejados que ocorreram nos ambientes de homologação ou produção, além de ter um espaço para alinhamentos gerais; retrô, ou retrospectiva, uma conversa informal entre os integrantes da *sprint* a fim de levantar os pontos positivos e negativos do período, propondo melhorias -quando possível-. A empresa possui forte cultura de *Scrum*, seguindo os ritos propostos pelo método.

A empresa utiliza de dois papeis do scrum (FONSECA, 2023), que são: o *PO*, na posição de representante dos *stakeholders*, decidindo quais tarefas serão executadas e o time de *Developers*, que capturam os requisitos, constroem o projeto e o disponibilizam para os usuários. Não há uma figura responsável pelo papel de *Scrum Master* na organização.

[1] https://www.treasy.com.br/blog/scrum/
[2] https://www.alura.com.br/artigos/metodo-kanban

Como artefatos da metodologia (HARRIS, 2023), há: o *Backlog* do produto, uma listagem com a descrição dos próximos projetos e seus respectivos requisitos; e o *Backlog* da *sprint*, uma listagem das tarefas que integram o próximo pacote funções disponibilizadas para os usuários.

2.2 Jira Software

O Jira [3] é um *software* cujo objetivo é acompanhar as tarefas que fazem parte dos projetos da empresa (ATLASSIAN, 2023). Ele se relaciona com o *Scrum*, disponibilizando os artefatos da metodologia, os *Backlogs*, em formato de quadro. Além disso, proporciona à equipe uma visão abrangente da situação da *sprint* atual e das próximas. A tecnologia é utilizada na organização por meio de um *board* para acompanhamento das tarefas.

2.3 API REST

A *Application Programming Interface (API) REST* [4] é uma aplicação que possui um conjunto de funções específicas para acesso de dados ao servidor, como cadastro (POST), busca (GET) ou deleção (DELETE) (REDHAT, 2023a). Ela se caracteriza por enviar uma requisição para um *endpoint* (*URL* específica utilizada para realizar o acesso a um recurso fornecido por aplicações Web a partir de uma *API*) via *HTTP*, podendo ser de diversos formatos e para diversos fins. A tecnologia é utilizada na organização para disparar as funcionalidades dos sistemas, como criação, edição e deleção de cadastros ou configurações e também para disparo das ferramentas de geração de relatórios e cancelamento de vendas.

2.4 API SOAP

A *API SOAP* [5] é uma aplicação, que assim como a *API REST*, possui um conjunto de funções específicas para acesso de dados ao servidor (IBM, 2022). Ela se caracteriza por enviar

[3] https://www.atlassian.com/software/jira
[4] https://aws.amazon.com/pt/what-is/api/
[5] https://www.ibm.com/docs/pt-br/taddm/7.3.0?topic=overview-using-soap-api

uma requisição para a *API Java*, podendo ser de diversos formatos e para diversos fins. A tecnologia é utilizada na organização para disparar a geração de relatórios para integração entre as vendas e pagamentos realizados no sistemas da conciliadora *Linx* [6].

2.5 Swagger

O *Swagger* [7] é um *framework* voltado para documentação de *APIs REST* (REDHAT, 2023a). Com ele, é possível nomear e especificar os parâmetros que devem ser passados em cada requisição, realizar testes manuais nas *APIs* e disponibilizar exemplos de requisição e retornos para cada rota (DIAS, 2023). A tecnologia é utilizada na organização como documentação das *APIs* de Venda Interna.

2.6 JavaScript

O *JavaScript*[8] é uma linguagem de programação amplamente utilizada em aplicações Web (DIO, 2023). Ela é dinâmica, orientada a objetos e focada em aplicações que são executadas utilizando navegadores. A tecnologia é empregada na organização como a linguagem de programação escolhida para a criação do sistema *Banking* e das automações feitas para ele.

2.7 Puppeteer

O *Puppeteer* [9] é um *framework* que fornece uma *API* de alto nível para controlar aplicação *Chromium*. Após sua instalação ele permite o controle completo do *browser* (AJEWOLE, 2020). A tecnologia é utilizada na organização para a criação dos testes automatizados do sistema *Banking*.

[6] https://www.linx.com.br/
[7] https://swagger.io/
[8] https://blog.betrybe.com/javascript/
[9] https://raullesteves.medium.com/puppeteer-navegando-no-chrome-via-nodejs-420484bf7012

2.8 Postman

O *Postman* [10] é uma ferramenta que dá suporte para as requisições em *APIs*. Nela é possível executar as rotas das *APIs*, realizar automação de testes e gerar relatórios (URBANO, 2022). A tecnologia é utilizada na organização para realização dos testes em *API* (RODRIGUES, 2023).

2.9 Testes de Software

O teste de *software* é um momento do ciclo de desenvolvimento em que o Analista de Qualidade executa uma ou mais funcionalidades para validar seu comportamento e seus retornos (OBJECTIVE, 2022). Para que um ciclo de testes seja bem-sucedido, é necessário organizar o ambiente, criando ou disponibilizando uma quantidade de insumos para serem utilizados; escrever um passo a passo a ser seguido; e ter os requisitos mapeados, a fim de validar se a aplicação está de acordo com o esperado. A estagiária passou por diversos tipos de teste durante o estágio (PITTET, 2022). São eles:

- Integração: nesse tipo de teste, é validado se as diversas funções do sistema coexistem sem conflitos internos, tornando a aplicação unitária e coerente. Eram realizados após o encerramento da *sprint*, a fim de validar a integridade do sistema.

- Funcionais: nesse tipo de teste, é analisado se os requisitos do projeto foram cumpridos corretamente. Eram realizados em todas as tarefas da *sprint*, com o objetivo de validar o cumprimento das especificações do projeto.

- Usabilidade: nesse tipo de teste, o analista simula os comportamentos de um usuário para validar a lógica da navegação pelo sistema, o recebimento de confirmações via SMS ou *e-mail*, e requisitos não-funcionais, como o tempo de carregamento de uma tela do sistema ou de geração de um relatório, entre outros. Eram realizados periodicamente no sistema e quando uma nova tela ou funcionalidade era disponibilizada, impactando outras funções.

[10] https://enotas.com.br/blog/postman/

- Aceitação: nesse tipo de teste, é validado se todo o sistema está de acordo com a regra de negócio e se está sendo aprovado pelos usuários. Eram realizados periodicamente, juntamente com os testes de usabilidade, para proporcionar uma visão abrangente das opiniões dos usuários para a equipe de desenvolvimento.

- Desempenho: nesse tipo de teste, é analisado como o sistema se comporta em situações de estresse, como processamento de arquivos grandes ou diversos acessos simultâneos na plataforma. Eram realizados periodicamente para identificar gargalos ou falhas na escalabilidade da aplicação.

- Regressão: nesse tipo de teste, o analista executa alguma funcionalidade do sistema a fim de garantir que ela está funcionando corretamente. Eram realizados para identificar falhas não reportadas ou para certificar que as funcionalidades estavam executando conforme o esperado.

3 ATIVIDADES DESENVOLVIDAS

Neste capítulo, serão apresentadas as atividades comuns a todos os projetos e as atividades específicas de cada projeto nas quais a aluna participou durante o período de estágio na *Equals*.

A estagiária desempenhou o papel de Analista de Qualidade ao longo de todo o estágio e realizou testes nos sistemas: Venda Interna, *Equals Core* e *Banking*.

3.1 Integração ao local de estágio

O estágio teve início no dia 09/03/2020 na sede da empresa, em São Paulo. A primeira semana teve como foco instruir a estagiária sobre a empresa e seus processos, com diversas reuniões e palestras sobre a cultura empresarial da mesma. Foi uma semana de imersão cultural.

A semana seguinte, já em Lavras, foi de imersão nas tecnologias utilizadas. Após realizar os cursos necessários, a estagiária foi integrada a um time composto por 5 membros. O time já possuía demandas ativas e projetos em vista; sua rotina era consistente, e a estagiária passou a participar dos ritos comuns do time, como a *daily*, onde eram discutidos os resultados das últimas tarefas trabalhadas e o planejamento para o dia.

3.2 Ritos do Scrum

Após alguns dias, o time iniciou o ciclo de disponibilização do *software* na metodologia *Scrum* (DRUMOND, 2023). A *planning* foi realizada, na qual foram apresentadas as tarefas a serem realizadas. Durante a reunião, foram discutidas as implicações das tarefas e realizada a pontuação de cada uma delas. Após o término do prazo da *sprint*, é realizada uma retrospectiva para discutir sobre os pontos altos e baixos da mesma, e há um comprometimento por parte do time em resolver as questões problemáticas nas próximas *sprints*. Em seguida, é realizada uma nova reunião de *planning*.

3.3 Controle de versões

A estagiária criou uma documentação que estabelecia a relação entre as tarefas da *sprint* e seus testes e *bugs*. Cada projeto possuía sua própria documentação, que era atualizada ao final de cada *sprint*. Nestes documentos, também constava quem foi o responsável por implementar as correções no ambiente de produção, a taxa de *bugs* encontrados por tarefa e o quão precisa foram as estimativas feitas na reunião de *planning*.

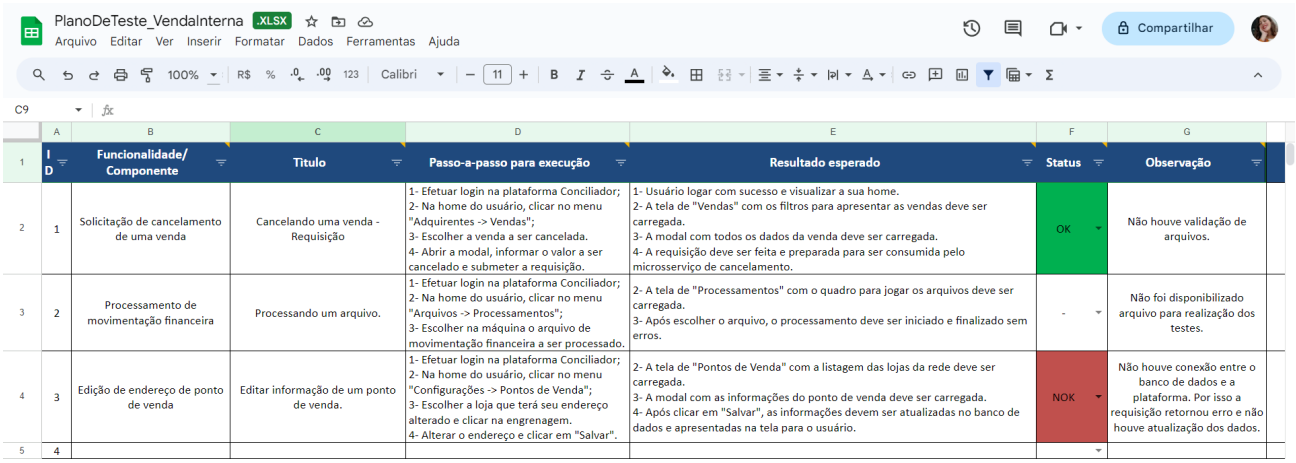
3.4 Planos de teste

A estagiária foi responsável pela criação e manutenção de um plano de testes aprofundado sobre algumas funcionalidades escolhidas. O plano cobria as funcionalidades de: cancelamento, processamento de arquivos, relatório de integração entre vendas e pagamentos e *APIs* do sistema Venda Interna. Nessa documentação, continha a descrição da função e quais testes deveriam ser executados após uma correção que impactasse nela, com alguns exemplos e passo-a-passo de testes mais complexos, incluindo execução de *queries* e *logs* internos.

A estagiária utilizava dos planos de teste criados para executar validações nos sistemas Conciliador, *Banking* e Venda Interna. Os ciclos de teste executados deveriam ter o retorno esperado, caso contrário uma análise de *logs* deveria ser feita, a fim de encontrar em qual ponto do processo houve a tratativa que resultou no erro. Após encontrar a raiz do problema, a estagiária a informava para a equipe de desenvolvimento por meio de uma tarefa no quadro *Kanban* do time e os ritos de planejamento, pontuação e priorização da tarefa são feitos para que ela seja inserida na próxima *sprint*.

O plano de testes era feito via Jira ou planilha. Nele era descrito: o passo-a-passo do teste, o resultado esperado após cada etapa e alguma informação ou observação sobre a execução do teste, como apresentado na Figura 3.1.

Figura 3.1 – Plano de Teste



ID	Funcionalidade/ Componente	Titulo	Passo-a-passo para execução	Resultado esperado	Status	Observação
1	Solicitação de cancelamento de uma venda	Cancelando uma venda - Requisição	1- Efetuar login na plataforma Conciliador; 2- Na home do usuário, clicar no menu "Adquirentes -> Vendas"; 3- Escolher a venda a ser cancelada. 4- Abrir a modal, informar o valor a ser cancelado e submeter a requisição.	1- Usuário logar com sucesso e visualizar a sua home. 2- A tela de "Vendas" com os filtros para apresentar as vendas deve ser carregada. 3- A modal com todos os dados da venda deve ser carregada. 4- A requisição deve ser feita e preparada para ser consumida pelo microsserviço de cancelamento.	OK	Não houve validação de arquivos.
2	Processamento de movimentação financeira	Processando um arquivo.	1- Efetuar login na plataforma Conciliador; 2- Na home do usuário, clicar no menu "Arquivos -> Processamentos"; 3- Escolher na máquina o arquivo de movimentação financeira a ser processado.	2- A tela de "Processamentos" com o quadro para jogar os arquivos deve ser carregada. 3- Após escolher o arquivo, o processamento deve ser iniciado e finalizado sem erros.	-	Não foi disponibilizado arquivo para realização dos testes.
3	Edição de endereço de ponto de venda	Editar informação de um ponto de venda.	1- Efetuar login na plataforma Conciliador; 2- Na home do usuário, clicar no menu "Configurações -> Pontos de Venda"; 3- Escolher a loja que terá seu endereço alterado e clicar na engrenagem. 4- Alterar o endereço e clicar em "Salvar".	2- A tela de "Pontos de Venda" com a listagem das lojas da rede deve ser carregada. 3- A modal com as informações do ponto de venda deve ser carregada. 4- Após clicar em "Salvar", as informações devem ser atualizadas no banco de dados e apresentadas na tela para o usuário.	NOK	Não houve conexão entre o banco de dados e a plataforma. Por isso a requisição retornou erro e não houve atualização dos dados.
4						

Fonte: Autora

3.5 Venda Interna

O sistema Venda Interna é um conjunto de microsserviços que não possuem uma tela para visualização. Sua proposta é fazer as transações e disponibilizar os dados nas plataformas da empresa. Ele é responsável por processar os arquivos bancários, realizar a conciliação entre as vendas e os pagamentos, gerar relatórios de integração entre as transações, cancelar parte ou toda venda feita por um estabelecimento e calcular taxas e impostos para que o comerciante saiba o lucro de cada produto vendido. As atividades executadas neste projeto consistiam em validar se os processamentos estavam ocorrendo da maneira correta e em tempo hábil, se as vendas canceladas eram removidas dos relatórios de vendas realizadas, se os microsserviços de processamento e integração eram capazes de ter bom desempenho sob pressão.

Quando é identificado um fluxo já estruturado que ainda não possui um plano de testes específico para ele, a estagiária criava a documentação. As documentações públicas da organização são salvas na plataforma *Confluence*. Para criar o plano de teste, bastava acessar o sistema, buscar pelo local ideal para salvar o documento -de acordo com o time e sistema que era contemplado pelo plano de testes- e escrever sobre a funcionalidade e sobre os casos de teste mais importantes que deveriam ser validados nos testes de regressão ou de funcionalidade, quando forem executados. Em

um plano de teste, é necessário citar as saídas esperadas para os dados de entrada, informar onde os *logs* estão disponíveis durante a execução da ação e quais tabelas do banco de dados receberão as alterações.

A documentação sobre cancelamento de vendas (ECOMMIT, 2022) informa ao leitor a diferença entre cancelamento e *chargeback* (MERENNA, 2020). No cancelamento, o valor da compra é devolvido como limite no cartão de quem a fez, enquanto o *chargeback*, ou estorno (SOUZA, 2022), depende da compra ter sido aprovada e constar na fatura do cartão de crédito ou no extrato bancário para que a devolução do valor seja realizada. O cancelameto de uma compra pode ser feito de maneira total, quando todo o valor da compra é cancelado, ou parcial, quando uma parte do valor total é cancelado. A funcionalidade de cancelamento foi disponibilizada para testes e a estagiária os realizou. Os ciclos de teste consistiam em realizar o cancelamento de uma venda de forma manual, onde uma ou mais vendas eram selecionadas e canceladas de forma total ou parcial, em seguida os microsserviços de cancelamentos capturam a mensagem e a consomem, retirando o valor a ser cancelado do total da venda, gerando o arquivo de remessa (arquivo com as informações da venda e do cancelamento, que deve ser enviado para a adquirente). A adquirente é responsável por analisar a solicitação e enviar um arquivo resposta para o usuário, o arquivo de retorno (que aprova ou rejeita o cancelamento). Caso o cancelamento seja rejeitado, a venda retorna ao seu valor total na plataforma. Caso o cancelamento seja aprovado, o valor é retirado do total da venda, que pode ficar zerada ou sobrar algum valor para o comerciante receber posteriormente. A documentação descrevendo o processo de cancelamento possui exemplos da estrutura dos arquivos e as posições alteradas por cada adquirente no arquivo de retorno, proporcionando reprodução semelhante ao processo realizado pelas empresas responsáveis. O cancelamento pode, também, ser feito de forma automática, onde a adquirente envia para o estabelecimento os arquivos de remessa e retorno, o usuário os processa e o cancelamento é realizado.

Os *frameworks Cypress* [1] e *Robot* [2] foram estudados no intuito de encontrar o mais adequedado para realizar as automações de cancelamento de vendas. A estagiária os estudou a partir de

[1] https://testingcompany.com.br/blog/conheca-o-cypress-e-suas-vantagens-para-automacao-de-testes
[2] https://www.devmedia.com.br/automacao-de-testes-com-o-robot-framework/32032

cursos, vídeo-aulas e documentações para sanar suas dúvidas e avaliar a viabilidade de sua utilização. Entretanto, foi reavaliada a necessidade de automatizar o fluxo de cancelamento e a estagiária foi informada pelo *PO* que era para pausar por um tempo com os estudos, pois haviam demandas prioritárias sendo definidas e planejadas para as próximas *sprints*.

O sistema Venda Interna utiliza de diversas *APIs* (REDHAT, 2023b) para consumir os dados, e cada *API* possui uma gama de rotas. Devido a quantidade, foi necessária a criação de uma documentação para concentrá-las em um mesmo lugar. A plataforma escolhida para armazenar esses dados foi o *Swagger*. As rotas foram enviadas para ele e a estagiária executava uma por vez na aplicação e na *API*, validando a consistência entre os retornos e os dados salvos no banco de dados.

A finalidade dessas documentações é ser um exemplo de testes que são importantes serem feitos no caso de alteração das funcionalidades, mostrar os resultados esperados e apoiar os testes de regressão, que devem ser realizados periodicamente no sistema.

3.5.1 Tecnologias e atividades realizadas

A equipe Venda Interna realizou diversas atividades, destacando uma reformulação completa no sistema de processamento de arquivos. Foi identificado um gargalo ao processar uma grande quantidade de dados, o que causou um problema no banco de dados e nos servidores do ambiente de produção, resultando no travamento da funcionalidade de processamento de arquivos para os clientes da *Equals*. A equipe se mobilizou para restabelecer o sistema de processamento, apagando todos os arquivos na fila de processamento, restaurando o servidor e informando aos clientes afetados para enviarem seus arquivos via *email*, permitindo que a equipe de suporte os processasse manualmente ao longo da semana.

Após a retomada do fluxo de processamento, foi decidido criar uma equipe para reformular o sistema de processamento de arquivos. Essa equipe era composta por um *Product Owner* (*PO*), dois desenvolvedores (um deles liderando o time) e a estagiária, atuando como Analista de Qualidade de *Software*. O processamento, anteriormente realizado por meio do microsserviço conhecido

como *Enterprise Service Bus* (ESB), que gerenciava o fluxo de processamento de vendas, pagamentos e suas inter-relações, era robusto e complexo de ser alterado. A decisão de reformular o processamento envolveu dividir o ESB em três microsserviços menores e independentes, devido à sua complexidade.

Foi decidido como seria feita a divisão e o primeiro microsserviço captaria a mensagem no Conciliador após o envio do arquivo no sistema e ficaria com o pré-processamento do mesmo, atualizando a tabela de processamento com as informações do arquivo e liberando uma mensagem para ser capturada pelo próximo microsserviço, para apenas após essa captura, o processamento ser continuado. O segundo microsserviço é o responsável por ler os dados das vendas e pagamentos, salvar essas informações no banco de dados e disparar uma mensagem para o último microsserviço, que por sua vez, após capturar a mensagem faz a conciliação entre vendas e pagamentos e finaliza o processo, atualizando a tabela de processamentos com a data e hora do final do processamento do arquivo. Como os microsserviços são independentes, é possível processar centenas de arquivos ao mesmo tempo sem haver gargalo, já que as mensagens são salvas na *Amazon Web Services* (*AWS*) por ordem de chegada e o serviço dispara uma mensagem por vez para cada microsserviço.

O processo de criação e testes dos microsserviços de processamento durou cerca de dois meses e meio e se iniciou com uma série de reuniões entre os membros do time para discutir as melhores soluções e propor o projeto. Após a estruturação do projeto, a *PO* reuniu os requisitos e os apresentou aos *stakeholders* para a aprovação e início do desenvolvimento do mesmo.

A aprovação do projeto foi concedida dias após sua apresentação, e a equipe iniciou imediatamente o desenvolvimento do primeiro microsserviço. Sua função era capturar a mensagem sobre um arquivo enviado no Conciliador, inserir suas informações no banco de dados e criar uma mensagem para o próximo microsserviço. Durante os testes, a estagiária validou as informações do arquivo e verificou se o *status* do arquivo no banco de dados era atualizado.

Enquanto a estagiária conduzia os testes, o segundo microsserviço estava em processo de criação. Ele era responsável por capturar a mensagem em sua fila, realizar a leitura e inserção das informações no banco de dados, além de criar a mensagem para o próximo microsserviço.

Novamente, a estagiária validou se os dados salvos nas tabelas de vendas e pagamentos estavam corretos e se o *status* do arquivo no banco de dados era atualizado.

Após a conclusão dos testes no segundo microsserviço, o terceiro já estava disponível para validação. Esse último microsserviço tinha a responsabilidade de capturar a mensagem enviada pelo segundo microsserviço, encerrar o processamento e fazer a atualização final do *status* do processamento. A estagiária validou se o processamento era encerrado corretamente e se, após sua finalização, os dados eram exibidos de forma adequada na plataforma.

Após encerramento dos testes e aprovação dos *stakeholders*, os três microsserviços foram disponibilizados para o público.

Foi realizada uma comparação entre o tempo de processamento de arquivos do maior cliente da *Equals*, que tinha uma média de quarenta e duas horas, e o tempo de processamento após a liberação dos três microsserviços. Como resultado, foi constatado que o tempo de processamento reduziu para oito horas. Após alguns dias, diversos clientes expressaram satisfação com a rapidez na disponibilização de seus dados, gerando valor para o projeto e deixando a equipe contente.

Além da reestruturação do processamento, a estagiária foi escolhida para ser a Analista de Qualidade responsável pela absorção da empresa *Linx*, que passou a integrar o portifólio de empresas pertencetes à *StoneCo* e teve seu sistema aprimorado, reformulando a regra de negócio e criando um novo produto a partir do já utilizado pela empresa.

A *Linx* é uma conciliadora que usa da *API SOAP* (SOAPUI, 2023) para conciliar os dados de vendas e pagamentos. A sua absorção foi feita por uma equipe composta por dois desenvolvedores e a estagiária.

A aluna teve uma imersão na regra de negócio da empresa e nas modificações que seriam feitas para que alguns microsserviços já utilizados pelo Conciliador pudessem ser reaproveitados. A *Linx* trabalha com o processamento de um arquivo de remessa gerado pelas adquirentes e enviado via *FileZilla*[3], que é um *software* de transferência de arquivos via protocolo (*FTP*) (ABREU, 2021).

[3] https://filezilla-project.org/

Os clientes acessam o *FileZilla*, enviam o arquivo da adquirente, que é então consumido pela *API*, gerando como resultado um arquivo de retorno para a conciliação.

As *APIs* foram reformuladas para apenas disparar a geração de retorno pelos microsserviços utilizados pelo Conciliador, e a estagiária realizou testes de integridade para validar se os arquivos gerados pelos microsserviços da *Equals* possuíam todos os dados necessários e não alteravam a estrutura final do arquivo de retorno. Como a *Linx* possuía um sistema pequeno, não houve complicações no processo de testes, e dentro de alguns dias a *Equals* absorveu o processo da *Linx*.

3.6 Conciliador

O *Equals Core* se trata de um *software* de conciliação bancária para comércios e *marketplaces* e é o principal sistema da *Equals*. A plataforma contém a relação entre a venda e o pagamento dos itens comercializados, estatísticas sobre o número de vendas por período, informações detalhadas dos pagamentos feitos, como o método de pagamento e número de parcelas.

Desde o seu lançamento, o projeto tem proporcionado uma visão ampla da situação financeira das empresas contratantes, apoiando na sua organização e tomadas de decisão para o futuro dos seus negócios.

No *Core*, é possível ter acesso completo às vendas realizadas por diversas formas de pagamento, cancelar uma venda inteira ou apenas parte dela, obter informações sobre as bandeiras de cartões e adquirentes utilizados em cada venda, bem como a porcentagem do valor da venda destinada a essas empresas. Além disso, é possível gerar relatórios abrangentes sobre a integração entre vendas e pagamentos, e realizar o processamento de arquivos bancários.

Sua proposta é ser o centro dos dados financeiros das empresas, podendo ser utilizado para tomada de decisões importantes, consultas e auditorias.

3.6.1 Tecnologias e atividades realizadas

O *Core* se relaciona com todos os outros sistemas da *Equals* e é alimentado por eles, principalmente pelo sistema Venda Interna.

A Venda Interna alimenta e movimenta os dados para disponibilizá-los ao usuário na tela do Conciliador. Destaca-se o cancelamento de vendas, que pode ser total ou parcial, gerando um relatório específico para registrar o montante cancelado, quem realizou o cancelamento, quando e o motivo para tal ação. As vendas canceladas são armazenadas em um gráfico separado e não são consideradas na receita da empresa, a menos que seja efetuado um cancelamento parcial. Nesse caso, apenas a parte cancelada é removida das previsões. Os testes realizados na seção de cancelamento de vendas consistiam em cancelar uma venda e validar a retirada do valor cancelado nos relatórios (encontrados na tela de geração de retornos) e nos gráficos do cliente (apresentados na tela inicial do Conciliador).

A tela de remessas disponibiliza os arquivos de retorno gerados após alterações na receita do cliente e é alimentada pelas *APIs* de cancelamento e de conciliação, onde ambas alteram os dados da tela após a execução de suas funcionalidades. A estagiária realizava testes de cancelamento e validava a redução no valor total que o cliente iria receber no relatório de retorno. Também realizava a conciliação de vendas e pagamentos e validava o aumento do valor total que o cliente iria receber no mesmo relatório.

A plataforma era utilizada pela estagiária para visualizar os resultados dos testes realizados pelos microsserviços e *APIs* de Venda Interna.

O *Core* possui uma ferramenta de antecipação, na qual a empresa pode, a partir de uma relação de todos os futuros recebimentos de pagamentos, solicitar a antecipação desses pagamentos e ter o dinheiro que iria ser recebido no futuro disponível em suas contas bancárias em até quarenta e oito horas. Os testes de antecipação consistiam em acessar a plataforma, na tela inicial, visualizar o gráfico de recebimentos futuros, solicitar a antecipação de um valor para uma data próxima, verificar os custos da antecipação, realizar a solicitação e verificar se o valor é removido do gráfico de valores a receber e inserido no gráfico de valores recebidos, que apresenta os depósitos realizados nas contas bancárias do cliente na semana vigente.

3.7 Banking

O Banking é um sistema bancário que pode ser integrado ao Equals Core, mas também pode ser utilizado de forma isolada. Ele é um sistema menos robusto e focado em microempreendedores que não desejam um relatório completo de sua organização financeira. O sistema disponibiliza a relação entre vendas e pagamentos, gráficos que mostram o crescimento ou decréscimo da receita da instituição contratante. Sua proposta é consolidar as informações das contas bancárias do contratante em um único lugar, proporcionando uma visão completa das transações financeiras da organização.

3.7.1 Tecnologias e atividades realizadas

O *Banking* é um sistema simples, que não demanda muitos recursos. Sua função principal é ser um extrato bancário de todas as contas do cliente, sendo altamente visual, apresentando várias telas com dados separados por categorias e gráficos personalizados. As principais telas incluem:

- Tela de Extrato Bancário: disponibiliza ao cliente o extrato completo de todas as contas que possui em diversos bancos. Detalha se o valor entrou como pagamento por uma venda realizada ou se houve retirada para o pagamento de alguma despesa.
- Tela de Movimentações Financeiras: apresenta ao cliente a relação de todas as movimentações realizadas em uma conta específica, selecionável antes da geração do relatório.
- Tela de Estabelecimentos: agrupa informações de extrato bancário, movimentações financeiras e detalhes de endereço e responsável pelo ponto de venda.
- Tela de Contas Bancárias: exibe uma relação de todas as contas bancárias e bancos dos quais o usuário é cliente.
- Tela de Conciliação Manual: destinada a registros não capturados pela *API* de conciliação após o processamento de arquivos, permitindo a conciliação manual de vendas e pagamentos, mesmo que de valores distintos.

- Tela de Desfazer Conciliação: apresenta uma relação de todas as vendas conciliadas, permitindo ao usuário desfazer qualquer conciliação, manual ou automática. Após desfazer a conciliação, os dados são carregados na tela de conciliação manual para futuras conciliações.
- Tela de Conciliação: apresenta os dados conciliados, mostrando a relação entre vendas e pagamentos, juntamente com informações detalhadas das vendas.

O *Banking* utiliza *APIs* para o cadastro de lojas, usuários e contas bancárias. No entanto, o processamento de extratos bancários e movimentações financeiras é realizado por um sistema separado, a fim de evitar complexidade no projeto.

O sistema responsável pelo processamento no *Banking* é chamado de Retaguarda. Quando um arquivo é enviado para a tela de processamento de arquivo no *Banking*, a *API* de processamento o captura e o encaminha para a Retaguarda. Neste ponto, a Retaguarda processa o arquivo, armazena os dados no banco de dados, realiza a conciliação automática entre vendas e pagamentos e, por fim, disponibiliza os dados para o usuário no *Banking*.

As atividades executadas neste projeto consistiam em validar se os dados estavam sendo relacionados de forma correta, por exemplo, uma venda para o seu respectivo pagamento, se os processamentos estavam acontecendo em tempo hábil e se o sistema era capaz de processar grande quantidade de dados de maneira simultânea.

A estagiária fez prova de conceito (POC) para o *Banking*, que é um protótipo para praticar o conceito de alguma área. O ramo escolhido foi automação de testes (XAVIER, 2023). Essas automações faziam o *login* no portal, recuperavam senha, acessavam a tela principal com os gráficos financeiros, tiravam um *print* e o salvava com a data e hora da execução do teste. As automações foram feitas utilizando o *framework Puppteer* (FERREIRA, 2021). A finalidade dessas automações eram fazer testes de regressão, para assegurar que apesar das atualizações que estavam sendo disponibilizadas, o sistema ainda estava em pleno funcionamento.

As tarefas realizadas no *Banking* consistiam em acessar o *board* do time, ler e entender as tarefas que estavam com *status* de liberadas para teste, selecionar a de prioridade maior dentre elas, criar o plano de teste com base no critério de aceite e executar os testes de maneira sequencial.

Para os casos de teste que retornavam resultado conforme o esperado, é necessário coletar as evidências e mudar o *status* do caso de teste para aprovado. Para os casos de teste que retornavam resultado diferente do esperado, é necessário reproduzir o erro coletando as evidências, abrir uma tarefa de *bug* com a explicação detalhada do ocorrido, mudar o *status* do caso de teste para reprovado e vincular a tarefa de *bug* ao caso de teste que foi encontrado o erro.

A estagiária detectou uma ausência de documentações sobre o sistema e como realizado em Venda Interna, as criou e disponibilizou no *Confluence*.

A *POC* da tela de login do *Banking* foi feita na linguagem *JavaScript* e valida as funcionalidades de alteração do idioma do sistema, recuperação de senha e login.

A automação foi desenvolvida seguindo o padrão *Triple A*, que estabelece que os testes feitos com essa metodologia devem passar por três etapas: *Arrange*, onde ocorre a preparação do ambiente e do banco de dados, inicialização de variáveis, entre outras configurações; *Act*, onde ocorre a execução do teste; e *Assert*, onde ocorre a validação dos resultados e a atribuição do *status* para aprovado ou reprovado (DARDE, 2020).

O teste inicia sua execução na etapa de *Arrange*, inicializando o *browser*, conforme apresentado na Figura 3.2, e acessando a *URL* do sistema, conforme apresentado na Figura 3.3.

Figura 3.2 – Função de requisição do *framework*

```
1  const puppeteer = require('puppeteer');
2  (async () => {
3   const browser = await puppeteer.launch({
4     //headless: true -> ele não abre o browser, apenas executa o teste
5     //headless: false -> executa o teste com o browser aberto
6     headless: false
7   });
```

Fonte: Autora

Figura 3.3 – Função de acesso ao sistema

```
17  //Uma nova aba com a URL será carregada:
18  const page = await browser.newPage();
19  await page.goto('http://192.168.121.222:9090/banking');
```

Fonte: Autora

Para realizar a troca de idioma do sistema, foi necessário identificar os botões. Alguns deles possuem identificadores (ID); no entanto, os botões responsáveis pela troca do idioma não tinham IDs atribuídos no momento da criação da função de troca de idiomas. Portanto, a estagiária os identificou pelo nome do elemento, obtido por meio da ferramenta "Inspecionar"do navegador, conforme ilustrado na Figura 3.4. Após a seleção, o sistema recarregava a página, exibindo o idioma escolhido.

O *Assert* era realizado de maneira visual, onde a estagiária verificava se houve a troca do idioma. Não existia uma função específica para realizar essa verificação.

Figura 3.4 – Função de troca de idioma do sistema para Inglês, Espanhol e Português

```
29  //Alteração de Idiomas ---------------------------------------------
30
31  await page.click('button.btn-idioma');
32
33  //Inglês:
34  await page.click('.btn-group>.btn:not(:first-child):not(:last-child):not(.dropdown-toggle)');
35  await page.waitFor(3000);
36
37  //Espanhol:
38  await page.click('.btn-group>.btn:last-child:not(:first-child), .btn-group>.dropdown-toggle:not(:first-child)');
39  await page.waitFor(3000);
40
41  //Português:
42  await page.click('.btn-group>.btn:first-child:not(:last-child):not(.dropdown-toggle)');
43  await page.waitFor(3000);
44
45  //Para fechar a Box de Idiomas ------------------------------------
46  await page.click('button.btn-fechar-escolher-idioma');
```

Fonte: Autora

A função de recuperação de senha consiste em trocar da página de *login* para a página de recuperação da senha, fazer a inserção do *email*, seleção do *reCAPTCHA*, que é uma tecnologia para inibir o tráfego de robôs e *bots* em sites (GARRETT, 2020) e disparar a requisição, que após ser consumida pela *API*, enviará para o *email* informado o formulário de redefinição de senha. Porém, como a função do *reCAPTCHA* é inibir tráfego artificial, não foi possível concluir a automação da funcionalidade, sendo necessário após a digitação dos dados, clicar para voltar para a tela de login, como apresentado na Figura 3.5.

O *Act* é conduzido de modo que a execução do teste continue apenas se o *Assert* for bem-sucedido. Caso contrário, a execução do teste é interrompida, e o *browser* permanece travado no momento do erro.

Figura 3.5 – Função de recuperação de senha

```
50  //Para Recuperar a Senha ------------------------------------------
51
52  //Clicar no botão Esqueceu sua senha?:
53  await page.click('.recuperarSenha');
54
55  await page.waitFor(1000);
56
57  //Preenchimento do campo de e-mail:
58  await page.type('#loginInputRec', 'nome.usuario@email.com');
59
60  //Clicar no botão Recuperar Senha:
61  const element = (await page.$$('button.button-login'))[1];
62  await element.click('button.button-login');
63
64  await page.waitFor(6000);
65
66  //Clicar no botão Voltar:
67  await page.click('button.button2');
```

Fonte: Autora

Para realizar o *login* no sistema, é necessário preencher os campos de *email*, senha e clicar em "Entrar". Após o acesso ao sistema, uma listagem de organizações que o usuário tem acesso é carregada e, entre elas, basta selecionar a desejada que o acesso será feito e a tela inicial do sistema, com os gráficos de extrato bancário, conciliação e receita serão carregados.

A automação é concluída capturando uma imagem da tela com os gráficos e salvando-a no computador pessoal do executor com o nome e dimensões da imagem fornecidos por ele, conforme ilustrado na Figura 3.6.

O *Act*, assim como na função de recuperação de senha, é conduzido de modo que a execução do teste continue apenas se o *Assert* for bem-sucedido.

Figura 3.6 – Função de login

```
71  //Para fazer o LOGIN ------------------------------------------------
72
73  //Preenchimento dos campos digitáveis da tela:
74  await page.type('#loginInputLog', 'nome.usuario@email.com');
75  await page.type('#loginSenha', 'SenhaUsuario123');
76   //Clique no botão Entrar:
77  await page.click('button.button-login');
78
79  await page.waitFor(3000);
80
81  //Entrando em Clientes ----------------------------------------------
82  await page.click('.select-client-table tbody > tr label');
83
84  //Delay de 6 segundos:
85  await page.waitFor(6000);
86
87  //Screenshot da tela ------------------------------------------------
88  await page.screenshot({path: 'example.png', fullPage: true});
89 })();
```

Fonte: Autora

A automação da tela de login do *Banking* foi proposta como um dos objetivos semestrais da avaliação de desempenho (AVD), a fim de avaliar a viabilidade de criar uma automação para o sistema completo. Isso se dava pelo fato de a empresa não possuir um teste de regressão automatizado para o sistema, o que impossibilitava garantir a integridade de todas as funcionalidades, especialmente quando alguma tarefa crítica ou que alterasse funções críticas era disponibilizada para os usuários.

Após a entrega da automação, a empresa optou por estudar outros *frameworks* mais robustos, e o projeto foi congelado devido a uma nova tela que estava sendo desenvolvida para o *Banking*, exigindo total dedicação da estagiária.

3.8 Relação teoria e prática

Durante o estágio foi possível aplicar diversos conhecimentos aprendidos durante o bacharelado em Sistemas de Informação. Algumas disciplinas se destacaram, são elas:

- **GCC224 - Introdução aos Algoritmos:** foi nesta disciplina que a aluna teve seu primeiro contato com programação. Ao longo do curso foram realizadas diversas atividades que resultaram em uma criação de pensamento lógico, que posteriormente foi muito importante profissionalmente.

- **GCC214 - Introdução a Sistemas de Banco de Dados:** foi nesta disciplina que a aluna aprendeu como funciona um banco de dados e a fazer *queries* utilizando a linguagem *SQL*.

- **GCC219 - Interação Humano-Computador:** foi nesta displina que a aluna entendeu e aprendeu a ver o *software* com o olhar do cliente, se tornando parte das suas análises e rotina de testes validar a usabilidade dos sistemas, para que eles sejam o mais acessível para todos os possíveis usuários.

- **GCC178 - Práticas de Programação Orientada a Objetos:** foi nesta disciplina que a aluna se aprofundou na linguagem de programação *Java*, que posteriormente seria cotada para escrever as automações.

- **GCC188 - Engenharia de *Software*:** foi nesta disciplina que a aluna teve seu primeiro contato formal com o ciclo de vida de um projeto de desenvolvimento e com a qualidade de *software*, que posteriormente se tornaria sua área de atuação na *Equals*.

- **GCC244 - Processos de *Software*:** foi nesta disciplina que a aluna se aprofundou em metodologias ágeis e entendeu como funciona o *Scrum*, facilitando sua adaptação aos ritos quando entrou na empresa.

4 CONSIDERAÇÕES FINAIS

O período do estágio foi de suma importância para o desenvolvimento pessoal e profissional da estagiária. A integração no mercado de trabalho, a convivência com os desafios e as pessoas que o envolvem - clientes e colegas - não apenas reforçaram o que foi aprendido durante os anos de graduação, mas também foram além, complementando com vivências diversas questões abordadas durante o curso.

A *Equals* se mostrou flexível em relação ao cumprimento da carga horária e ofereceu suporte para que a graduação fosse continuada sem problemas. Durante o estágio, a aluna identificou oportunidades de melhoria na empresa, como a criação de um cargo para o papel de *Scrum Master*. Essa medida visa garantir que os ritos do *Scrum* sejam realizados corretamente e que as dúvidas dos colaboradores sobre a metodologia sejam direcionadas a uma pessoa com conhecimento e treinamento para resolvê-las. Além disso, foi percebida a necessidade de uma relação mais próxima entre a equipe de desenvolvimento e a liderança do time. A aluna notou um certo distanciamento do líder em relação à equipe, especialmente na área de qualidade, o que dificultava a análise do progresso do time além das entregas realizadas.

O tempo da aluna na empresa não se limitou ao estágio; ao término do contrato, a estagiária recebeu uma proposta de efetivação, que foi aceita. A parceria entre a *Equals* e a aluna se manteve, porém na posição de Analista de Qualidade Júnior.

O curso de Sistemas de Informação poderia incluir uma nova disciplina em sua grade, que aplicasse os conceitos abordados nas disciplinas de Engenharia de *Software* e Qualidade de *Software*, com o objetivo de proporcionar aos estudantes uma experiência semelhante ao ambiente de trabalho e desenvolver o pensamento de qualidade nos futuros profissionais.

O estágio foi de grande importância na construção da carreira da estagiária. Foi uma base sólida para construir uma profissional que tem paixão pelo que faz, que vê a qualidade de *software* como fundamental e consegue se colocar no lugar do seu usuário final.

REFERÊNCIAS

ABREU, L. **Filezilla: entenda para que serve e como instalar**. 2021. (Acessado em 01/12/2023). Disponível em: <https://rockcontent.com/br/blog/filezilla/>.

AJEWOLE, B. **Introduction to web scraping with Puppeteer**. 2020. (Acessado em 10/07/2023). Disponível em: <https://rexben.medium.com/introduction-to-web-scraping-with-puppeteer-1465b89fcf0b>.

ATLASSIAN. **Jira Software para equipes**. 2023. (Acessado em 01/12/2023). Disponível em: <https://www.atlassian.com/br/software/jira/guides/getting-started/who-uses-jira#for-project-management-teams>.

CNN, B. **Procura por profissionais de tecnologia cresce 671pandemia**. 2021. (Acessado em 01/12/2023). Disponível em: <https://www.cnnbrasil.com.br/economia/procura-por-profissionais-de-tecnologia-cresce-671-durante-a-pandemia/>.

DARDE, P. R. **O Padrão Triple A (Arrange, Act, Assert)**. 2020. (Acessado em 02/12/2023). Disponível em: <https://medium.com/@pablodarde/o-padr%C3%A3o-triple-a-arrange-act-assert-741e2a94cf88>.

DIAS, W. **Documentando sua API Rest com Swagger**. 2023. (Acessado em 18/09/2023). Disponível em: <http://www2.decom.ufop.br/terralab/documentando-sua-api-rest-com-swagger/>.

DIO. **JavaScript: O que é, Onde é usado e Porque aprender essa linguagem**. 2023. (Acessado em 02/12/2023). Disponível em: <https://www.dio.me/technologies/javascript>.

DRUMOND, C. **O que é scrum e como começar**. 2023. (Acessado em 15/06/2023). Disponível em: <https://www.atlassian.com/br/agile/scrum>.

ECOMMIT. **Cancelamento de vendas: o que é e como funcionam?** 2022. (Acessado em 18/09/2023). Disponível em: <https://www.techtudo.com.br/listas/2020/07/o-que-e-recaptcha-entenda-como-funciona-recurso-de-seguranca-do-google.ghtml>.

FERREIRA, B. E. de M. **Web Scraping com NodeJS e Puppeteer**. 2021. (Acessado em 06/07/2023). Disponível em: <https://github.com/brunoemferreira/NodeJS-WebScraping-Puppeteer>.

FONSECA, V. **Scrum: Entenda os 3 papéis fundamentais para o sucesso do seu projeto**. 2023. (Acessado em 29/11/2023). Disponível em: <https://awari.com.br/scrum-entenda-os-3-papeis-fundamentais-para-o-sucesso-do-seu-projeto/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Scrum:%20Entenda%20os%203%20pap%C3%A9is%20fundamentais%20para%20o%20sucesso%20do%20seu%20projeto>.

GARRETT, F. **O que é reCAPTCHA? Entenda como funciona recurso de segurança do Google**. 2020. (Acessado em 17/09/2023). Disponível em: <https://www.techtudo.com.br/listas/2020/07/o-que-e-recaptcha-entenda-como-funciona-recurso-de-seguranca-do-google.ghtml>.

GOUVEA, F. P. by. **Pessoas na internet e internet nas pessoas: a verdadeira transformação digital de resultados**. 2022. (Acessado em 01/12/2023). Disponível em: <https://friedman.com.br/pessoas_na_internet/>.

HARRIS, C. **Artefatos do Scrum ágil**. 2023. (Acessado em 29/11/2023). Disponível em: <https://www.atlassian.com/br/agile/scrum/artifacts#:~:text=Resumo%3A%20artefatos%20de%20Scrum%20%C3%A1gil,do%20sprint%20e%20os%20incrementos.>

IBM. **Usando a API SOAP**. 2022. (Acessado em 21/06/2023). Disponível em: <https://www.ibm.com/docs/pt-br/taddm/7.3.0?topic=overview-using-soap-api>.

ICET, U. **Curso de Graduação em Sistemas de Informação (Bacharelado)**. 2023. (Acessado em 01/12/2023). Disponível em: <https://icet.ufla.br/graduacao/sistemas-informacao-bacharelado>.

MERENNA, I. **O que é chargeback?** 2020. (Acessado em 18/09/2023). Disponível em: <https://raiox.com.br/o-que-e-chargeback/#:~:text=Qual%20a%20diferen%C3%A7a%20entre%20Cancelamento,ta%20que%20realizou%20a%20venda.>

OBJECTIVE. **Testes de Software: Definição, Conceitos e Exemplos**. 2022. (Acessado em 29/11/2023). Disponível em: <https://www.objective.com.br/insights/testes-de-software/>.

PAULA, G. B. de. **Tudo sobre Metodologia Scrum: o que é e como essa ferramenta pode te ajudar a poupar tempo e gerir melhor seus projetos**. 2016. (Acessado em 15/06/2023). Disponível em: <https://www.treasy.com.br/blog/scrum/>.

PITTET, S. **Diferentes tipos de testes de software**. 2022. (Acessado em 29/11/2023). Disponível em: <https://www.atlassian.com/br/continuous-delivery/software-testing/types-of-software-testing>.

REDHAT. **API REST**. 2023. (Acessado em 20/06/2023). Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>.

REDHAT. **O que é API?** 2023. (Acessado em 20/06/2023). Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>.

RODRIGUES, J. **Testando serviços Web API com Postman**. 2023. (Acessado em 25/07/2023). Disponível em: <http://www.linhadecodigo.com.br/artigo/3712/testando-servicos-web-api-com-postman.aspx>.

SCHWABER, J. S. K. **O Guia do Scrum: O Guia Definitivo para o Scrum: As Regras do Jogo**. 2020. (Acessado em 29/11/2023). Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>.

SOAPUI. **SOAP vs REST 101: Understand The Differences**. 2023. (Acessado em 20/06/2023). Disponível em: <https://www.soapui.org/learn/api/soap-vs-rest-api/>.

SOUZA, T. **Estorno e cancelamento de compras: entenda a diferença**. 2022. (Acessado em 18/09/2023). Disponível em: <https://www.foregon.com/blog/estorno-e-cancelamento-de-compras-entenda-a-diferenca/>.

URBANO, L. **Postman: saiba como instalar e dar seus primeiros passos**. 2022. (Acessado em 26/07/2023). Disponível em: <https://www.alura.com.br/artigos/postman-como-instalar-dar-seus-primeiros-passos>.

XAVIER, N. V. G. E. J. **Automação de testes para backend nodejs utilizando o framework jest**. 2023. (Acessado em 03/07/2023). Disponível em: <http://www2.decom.ufop.br/terralab/automacao-de-testes-para-backend-nodejs-utilizando-o-framework-jest/>.