Desenvolvimento e Análise de uma ferramenta de testes automatizados em contexto de transações financeiras

Caio H Fernandes, Daniel F L Souza

Departamento de Ciências Exatas – Universidade Federal da Paraíba (UFPB)
Rio Tinto – PB – Brazil

Abstract. *The demand for an automated testing tool within the context of financial transactions and also in the Phoebus context is something that is notorious, with this the need for the Phast-Test-Tool arises, in order to reduce the time spent during the testing phase and increase the its effectiveness performing tests automatically based on the specifications and reality of each scenario, taking shape through meetings and prerequisites, being developed using technologies and specifications already used within Phoebus, was validated and placed in a production environment with tests in gigantic scale, obtained expressive results.*

Resumo. *A demanda por uma ferramenta de testes automatizados dentro do contexto de transações financeiras e também no contexto Phoebus é algo notório, com isso surge a necessidade da Phast-Test-Tool, com o objetivo de diminuir o tempo gasto durante a fase de testes e aumentar a sua eficácia realizando testes de forma automática com base nas especificações e realidade de cada cenário, tomando forma através de reuniões e pré-requisitos, sendo desenvolvida utilizando tecnologias e especificações já utilizadas dentro da Phoebus, foi validade e colocada em ambiente de produção com testes em escala gigantesca, obteve resultados expressivos.*

1. Introdução

Uma demanda cada vez mais crescente por software de qualidade, associada à uma necessidade crescente de entrega em um prazo mais curto fazem com que a fase de testes seja o ponto crucial para a entrega de software à luz dessas diretrizes. Em especial, quando falamos de softwares relacionados à área de transações financeiras, esse desafio se torna ainda mais complexo, dada a natureza do domínio de aplicação, a alta demanda, prazos curtos e complexidade das regras de negócio.

Transação financeira é toda e qualquer movimentação monetária. Uma classe especial de transações dessa natureza, são as que envolvem cartões de crédito. Estas podem ser realizadas na forma de débito ou crédito e são compostas por um conjunto de processos que envolvem pagamento, confirmação, anulação e estorno. A complexidade envolvida em sistemas de software que cuidam de transações financeiras é bem relevante, pois além de exigir uma boa quantidade de testes exige também alta atenção e qualidade. Para que um software desta natureza possa ser considerado em versão final é necessário uma certificação em várias camadas começando pelos testes internos da

empresa e indo para testes externos por empresas terceirizadas por conta da função do software que envolve valores e transações entre empresas e pessoas.

A disciplina de testes no contexto de transações financeiras têm grande demanda devido a complexidade e quantidade de testes necessários para validar cada cenário nas várias prestadoras de serviços(adquirentes), como Cielo e Prisma, não existindo nenhuma ferramenta que consiga testar de forma automática os logs. Isso se dá ao fato de cada empresa ligada ao universo de operações com cartões, utilizar sua própria linguagem, processos e protocolos para aumentar a segurança.

A Phoebus[1] é uma empresa paraibana que atua no mercado de processamento de operações financeiras baseadas em cartão de crédito. A empresa está no mercado desde 1997. Um dos desafios da empresa é entregar soluções financeiras com alto nível de qualidade e para tanto a disciplina de testes é primordial. Assim como outras empresas do segmento, o grande desafio da Phoebus é trabalhar na construção de ferramentas que possam permitir a melhoria do seu processo de testes. Nessa linha, uma das demandas existentes atualmente na empresa é a necessidade de uma ferramenta para automação de testes para transações financeiras baseadas em cartão.

Ressaltando que todo o processo de *sprints* e testes foram realizados dentro do contexto do quad Server dentro da empresa Phoebus, onde constatamos o modelo ágil chamado Apollo que é feito para empresas melhorarem seus processos e tambem seu desempenho.

Este trabalho tem como objetivo a análise e validação de uma ferramenta para automatização de testes no contexto da comunicação entre terminais clientes e serviços de processamento de operações financeiras. Tais operações têm como complexidade a diversidade de elementos envolvidos na comunicação, quais sejam: diferentes clientes de pontos de venda e diferentes formas de comunicação com adquirentes. Desta forma, o conjunto de casos de testes e nuances a serem tratadas tornam um processo de teste manual demorado e passível de falhas.

Esse artigo está organizado da seguinte forma: (a) a seção 2 apresenta uma breve fundamentação teórica com algumas definições sobre temas necessários para o entendimento do trabalho; (b) a seção 3 exibe a definição do problema e a solução de forma arquitetural e logo depois a forma que está sendo executada a ferramenta; © a seção 4 demonstra a expectativa na execução da ferramenta, futuras melhorias; (d) a seção 5 apresenta o produto final e seus resultados são vistos de forma objetiva assim ,verificando o seu rendimento.

[1] www.phoebus.com.br

2. Fundamentação teórica

2.1 Qualidade e Teste de software

A definição de qualidade feita por Mello Cordeiro(2004) diz que a qualidade é a adequação do uso, ou seja, é necessário considerar as expectativas e requisitos dos clientes uma vez que são os mesmos que vão usar o produto, segundo Kan (2002) o tema de qualificação de qualidade é considerado um termo ambíguo dependendo do interesse e atributos de qualidade das entidades envolvidas. O termo qualidade faz parte do nosso cotidiano e tem significados diferentes em usos populares e profissionais (Kan, 2002). A qualidade de um software tem como uma de suas principais disciplinas o teste de software.

A fase de testes é uma das mais importantes para garantir o padrão elevado e assim garantir o bom funcionamento de todas as partes do sistema, o teste de software está intimamente ligado ao conceito de qualidade de software e ele sempre foi um dos principais métodos de garantia de qualidade na indústria de software(BERTOLINO, 2007).

Os testes de software numa visão ampla são notados como atividades para analisar e encontrar erros, processos feitos para procurar em uma parte do software alguma inconsistência ou incoerência nesta parte assim averiguando a qualidade do software(Myers, Sandler et al., 2004), podendo ter variados tipos cada um se adequando aos requisitos especificados no produto como visto na Figura 1.

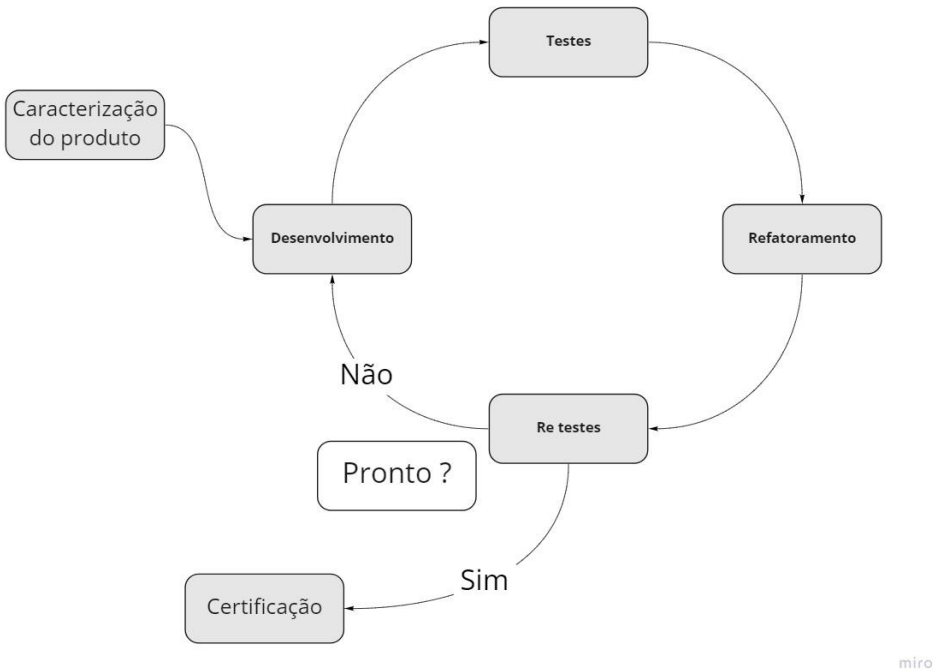


Figura 1. Fluxograma de testes . Fonte : Autoria própria

O fluxo de testes em metodologias ágeis geralmente tem o mesmo padrão, sempre ocorrendo a caracterização do produto entre cliente e o *Product Owner*(Dono do produto) para definir os requisitos necessários durante as *sprints*(ciclos de entrega), assim dando início ao desenvolvimento, logo em seguida os testes onde é definidos quais tipos de testes vão ser usados e executados verificando que não existe nenhum erro, próximo passo é a refatoração onde é analisado as melhorias técnicas na parte de programação depois vem re-teste onde se analisa mais uma vez para saber se existe algum erro, caso não é finalizada a sprint e ocorre a entrega.

A etapa de testes é considerada a parte mais crítica do ciclo de vida do desenvolvimento, uma vez que define a próxima sequência de atividades executadas(Dubey, 2017). O ciclo de vida de um teste é determinado pela fase de análise de requisitos, durante esta fase são analisados os requisitos que o software deve fazer e os que não devem fazer, determinando a segurança e o desempenho do sistema(Hood & Chhillar, 2015), Logo em seguida o Design, nesta fase os objetivos são transformados em condições de teste(Graham et al., 2008), Implementação e Execução é a próxima e última fase onde ocorre a implementação e a execução dos testes em cenários reais e também são retornados os resultados para a avaliação dos testes.

Há vários tipos de testes, a exemplo do teste funcional, teste estrutural e também os testes de condições. O conceito de testes funcionais é utilizar as especificações do produto e gerar os testes sem a necessidade de conhecer o código do programa, os testes se baseiam apenas nos resultados obtidos e nos valores especificados previamente durante a caracterização do produto, os testes de condições tem como objetivo testar todos os valores através de estruturas de decisão e condicionamento ao menos uma vez, o estrutural é chamado também de teste de “caixa-branca” porque ele tem acesso ao código fonte e o testador analisa cada etapa do software através do fluxo de dados,

Também temos o teste regressivo que consiste na aplicação dos testes em componentes antigos para garantir que mesmo em versões novas eles continuem em seu funcionamento normal, assim testando todo o código de uma só vez os componentes novos e os antigos.

2.2 Transações Financeiras Baseadas em Cartão de Crédito

A execução de transações financeiras ocorre em sequência como mostra a figura 2, passando pela requisição de compra feita em uma maquineta de cartão, sendo enviada até o servidor onde ocorre o processamento das informações de cartão e da compra, retornando a resposta na maquineta e assim relatando a resposta do servidor de confirmação da compra.

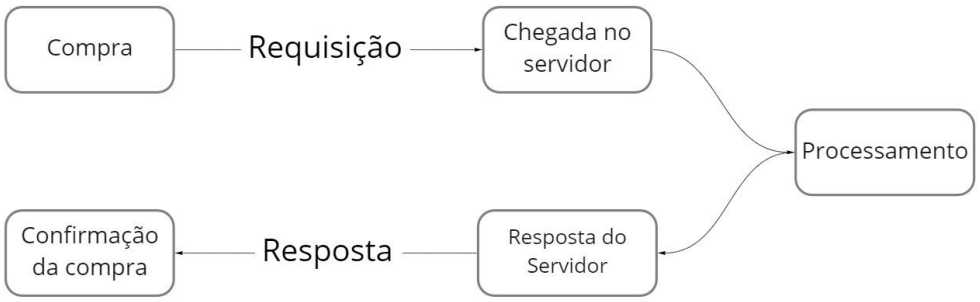


Figura 2. Processo de transação financeira via Cartão . Fonte : Autoria própria

A operação ocorre da mesma forma, mas existem possibilidades de não ter essa sequência por conta das especificações que são únicas para cada adquirente, sendo elas a forma que o adquirente especifica o processamento da confirmação de pagamento ou o processamento de negação ou recusa.

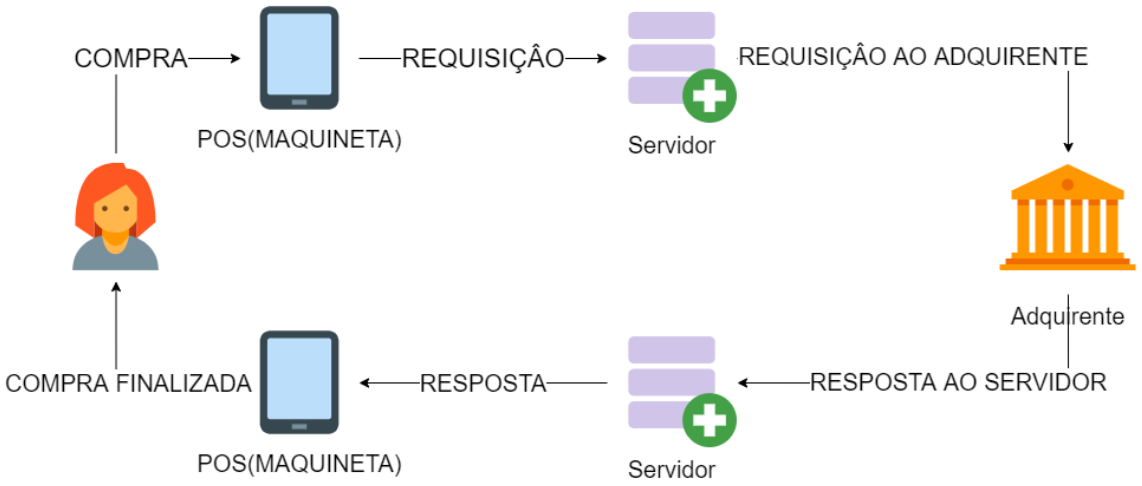


Figura 3. Fluxograma de transação financeira . Fonte : Autoria própria

A Transação financeira do ponto de vista físico tem uma ordenação bem estabelecida como visto na figura 3, apresentando a requisição do cliente onde vão todas as informações sobre o terminal, loja, cartão, cliente e também a forma de pagamento.

Logo em seguida ocorre a requisição do servidor que envia as informações contidas da requisição do cliente e espera a resposta do adquirente. Em seguida, ocorre a confirmação através da resposta do adquirente que é chamada de resposta ao servidor que contém o dado da aprovação e caso necessário dados sobre o cartão e conta do cliente. Na resposta a máquina do cartão, também chamada de resposta ao cliente o servidor, envia informações sobre o terminal e o dado de confirmação. Esse fluxo de forma física geralmente ocorre desta maneira, mas ocorrem algumas outras mudanças dependendo da especificação da adquirente. Uma mudança, por exemplo, é a confirmação do POS(máquina do cartão) que recebeu a resposta do cliente, essa mudança é realizada para finalizar as transações pendentes.

3. Problemática

3.1 Definição do problema

Como já explicado acima, a problemática que envolve esse assunto está relacionada à complexidade e à demanda de tempo para o teste completo de todos os cenários possíveis e existentes nesse tipo de demanda, assim esgotando todas as eventualidades, garantindo, dessa forma, um resultado promissor na otimização de tempo.

Entendendo que o tempo é um ponto principal na área empresarial de desenvolvimento de software devido às cobranças feitas pelo clientes e também a complexidade da análise dos documentos retornados pelo servidor e adquirente, pelo fato de ter uma grande quantidade de campos e também alguns documentos que vêm criptografados, geralmente em hexadecimal assim impossibilitando a leitura de forma natural.

O tempo é o principal problema em atrasos de entrega, na etapa de testes atraso é um motivo para ter gargalos no time, assim atrasando o cumprimento de prazos. Esse tempo demandado pela equipe de testes é bem longo por conta da análise dos campos ser feita atualmente de maneira manual, uma análise individual dos campos, assim prolongando o tempo necessário para cada teste e cenário, além do tempo temos a complexidade de cada teste existindo um arquivo com dados(log) que são enviados de forma criptografadas assim exigindo do testador um conhecimento sobre a criptografia e também uma forma para descriptografar o log, aumentando a demanda de tempo, por esses motivos a necessidade de existir uma ferramenta capaz de fazer essa análise de forma precisa e eficaz.

A problemática sobre o tempo está relacionada também à quantidade de testes existentes para cada adquirente. Atualmente em apenas uma adquirente existem mais de 4 mil testes e são executados de forma manual com a análise sendo feita de forma individual para cada campo, essa quantidade de testes tende a aumentar a cada adquirente nova ou a cada atualização nos sistemas crescendo a necessidade da ferramenta de testes.

3.2. Solução Proposta

Pensou-se em uma ferramenta de automatização à luz do conjunto de características internas ao time e que estão associadas à forma como a solução de pagamentos funciona, adaptada à esteira de desenvolvimento da empresa, as adquirentes suportadas, aos protocolos.

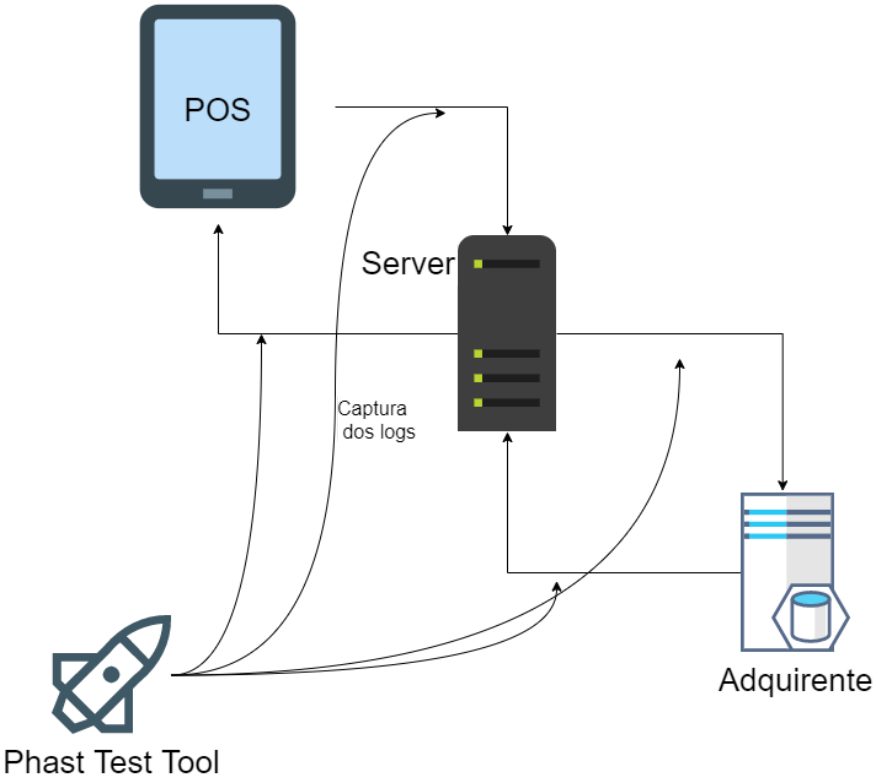


Figura 4. Solução arquitetural da ferramenta. Fonte : Autoria própria

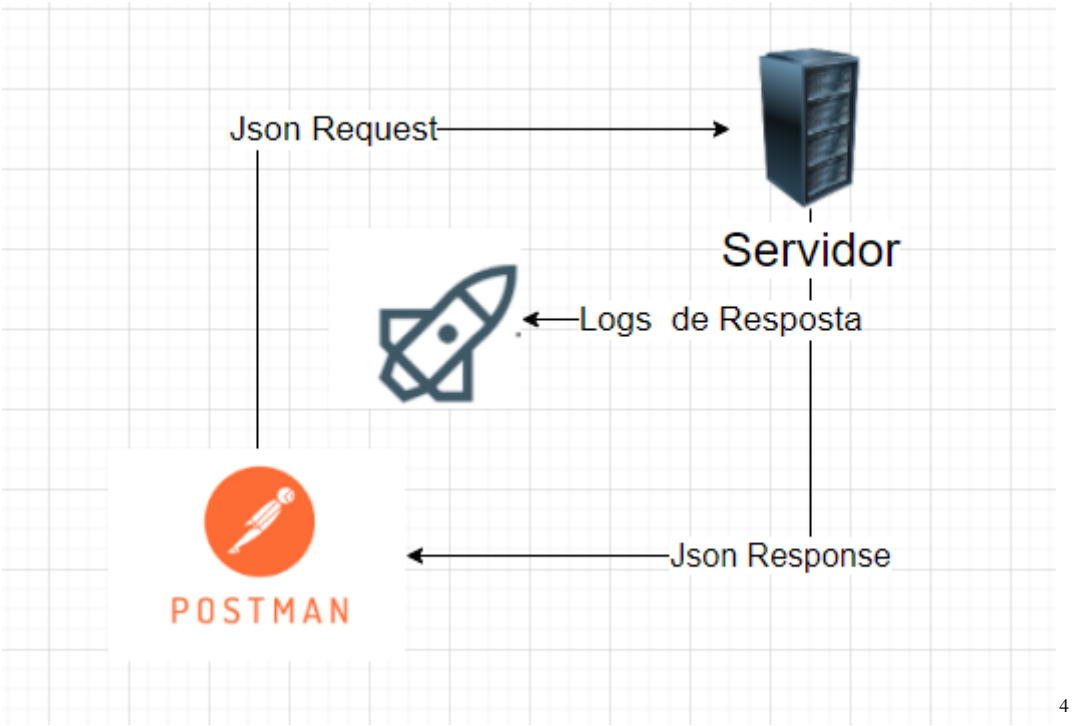
A solução proposta foi a ferramenta Phast-Test-Tool, que consiste em um mecanismo de testes automatizados feito sob medida para as necessidades encontradas na empresa Phoebus tratando de cada característica necessária para a validação dos cenários e dos campos existentes. A Figura 4 mostra como a ferramenta Phast-Test-Tool é utilizada para capturar os logs e quais logs ela captura, sendo eles JSON e ISO, conseguindo pegar todos os logs que são passados pelo servidor e assim testando cada log. Desta forma, é possível validar todos os campos existentes em um teste automatizado.

Phast-Test-Tool foi dividida em três camadas, sendo a primeira camada responsável por recolher e categorizar todas as requisições e a segunda camada encarregada por processar todas as informações e decodificar todos os campos que estão em hexadecimal ou criptografados, bem como a terceira que se desenvolve os testes para verificação dos campos de requisições e de respostas analisando e atestando a validade do código.

A Phast-Test-Tool utiliza o [2]Spring Boot como base para seu desenvolvimento e também utiliza o [3]Kafka Listener para a leitura dos Logs que são enviados pelo servidor através de requisições feitas pelo Postman, assim facilitando todo o processo de testes

[2]https://spring.io
[3]https://kafka.apache.org

que ocorre na segunda camada onde é feito o processamento que consiste em separação dos logs através de campos específicos.



[4]

Figura 5. Fluxo de dados Phast Test Tool. Fonte : Autoria própria

O Fluxo de dados utilizado durante o processo de testes foi organizado como ilustrado na Figura 5, de forma que todos os JSON e ISO logs que são recebidos ou enviados pelo servidor e capturado pela Phast Test Tool, essa captura é feita em ordem de chegada através do gerenciador de logs Kafka.

Os logs JSON são responsáveis por toda a comunicação entre o POS e o servidor, como demonstrado nas Listagens 1 e 2. as informações sobre terminal, mercador, cartao, forma de pagamento, valor e parcelas são todas incluídas neste log, os logs ISO são responsáveis pela transmissão dos dados entre servidor e adquirente, transmitindo as informações de uma forma mais segura utilizando de criptografia para guardar os dados sensíveis .

Listagem 1. Log da requisição enviada ao servidor pelo Postman

```
1. Json Request
2. merchantId: "16416"
3. terminalID: "3800145"
4. transactionInfo:
5. tipo: "02"
6. productId: 3
7. numeroTicket: 4769
8. dataTransaçao: "2021-10-25T18:12:46.785Z"
```

[4]

```
9.    valor: 100
10.   cardInfo:
11.           formaEntrada: 2
12.           nomeCartao: "LUCAS CRUZ NEVES"
13.           marcaCartao:"VISA"
```

Listagem 2. Log da resposta retornada pelo servidor ao Postman

```
1.  Json Response
2.  transactionAuthorization:
3.  dataTransaçao: "2021-10-25T18:14:03.806Z"
4.  valor: 100
5.  providerId: "33"
6.  merchantId: "016416"
7.  terminalId: "38000145"
8.  idLigação: "4ba7-3881-903a-4c52f29f61da"
```

O Processo de inicialização do Phast-Test-Tool é feito a partir da disponibilização de uma versão no ambiente de desenvolvimento, onde o pipeline é iniciada e gera a versão do servidor iniciado em uma máquina virtual para que se possa executar os testes. Logo após o início do servidor a pipeline inicializa a ferramenta através do Spring Boot e também inicializa o Kafka Listener. A partir daí as requisições são feitas pelo Postman enviando cada cenário e cada teste para que a ferramenta possa testar os logs enviados e recebidos pelo servidor.

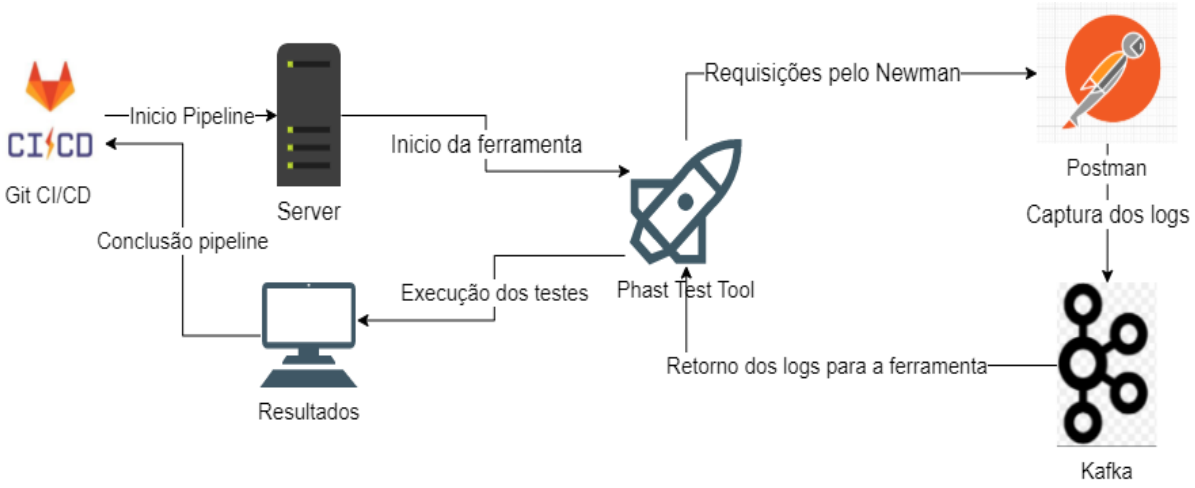


Figura 6. Diagrama de funcionamento da ferramenta. Fonte : Autoria própria

Conforme exposto durante a definição do problema, a maior necessidade na elaboração da ferramenta de testes está na questão do tempo. A Ferramenta por questão de

necessidade vai ter acesso a cada log enviado ou recebido pelo servidor para que possa de forma eficiente analisar e testar todos os valores retornados, como vemos na Figura 6. Desta forma conseguindo agregar todos os logs de forma organizada e fácil de ser acessada até mesmo a análise se torna prática, tornando mais eficiente todo o processo de testes onde a ferramenta só responde apenas os erros ou o sucesso de cada teste sem precisar de nenhuma interferência humana .

4. Resultados

Os resultados aqui discutidos foram extraídos a partir da análise de 100 testes completos de cenários que foram validados. Em uma estimativa de cem unidades de caso de testes, a ferramenta conseguiu uma média de tempo de execução de 20 à 30 segundos por cenário, em comparação com os testes realizados de forma manual que levariam em torno de 36 horas de trabalho de um analista de testes com experiência e conhecimento nessa área, na parte de transações financeiras e análise de logs JSON e ISO.

Nesse sentido, a melhora se deu tanto em relação ao tempo quanto no quesito eficácia, pois durante o processo de testes a própria ferramenta Phast-Test-Tool pôde ser validada por completo, notadamente no que diz respeito a todas as nuances que envolvem a estrutura dos arquivos de log.

Visto isso, observamos que a partir dessa ferramenta foi possível a validação de 2226 casos, que já foram analisados e criados os testes necessários para verificar que em cada campo está sendo retornado o valor previsto em sua especificação, Graças a alta adaptabilidade da ferramenta, os casos de testes podem ser adaptados em outros tipos de cenários e/ou adquirentes.

Na visão dos Analistas de testes já presentes na equipe, os resultados são nítidos, A melhoria foi significativa em relação ao tempo tendo execuções de testes na ferramenta até 6.000(seis mil) vezes mais rápido que a execução do teste de forma manual, resolvendo a incapacidade da equipe de não conseguir fazer o reteste em todos os cenários possíveis de uma adquirente ou de todas as adquirentes. Outra melhoria citada por um dos analistas foi “a praticidade por precisar implementar o teste apenas uma vez e a ferramenta continuar validando os cenários sempre que preciso”, assim facilitando todo o cronograma de testes e retestes que forem precisos.

5. Considerações Finais e Trabalhos Futuros

Esse trabalho tratou sobre a problemática de automatização de testes no contexto de desenvolvimento de software na área de operações financeiras, citando as necessidades e as motivações para que a ferramenta fosse desenvolvida. Como visto, a problemática é a falta de uma forma eficiente para cumprir os prazos, por conta da falta de testes automatizados impactando no aumento dos custos e também o consumo de tempo da equipe, deixando explícito mais uma vez a necessidade de testes automatizados.

No contexto da Phoebus a solução proposta foi a elaboração da ferramenta de testes automatizados com o recolhimento de todos os logs e processamento de cada um dos campos de forma individual para que seja confirmado o retorno e a comparação

feita com retorno do erro ou do sucesso. Assim foi desenvolvida a ferramenta Phast-Test-Tool, suprindo a necessidade durante a fase de teste proposta no cenário atual.

Como trabalhos futuros pretende-se realizar melhorias na ferramenta é a customização para a Phast-Test-Tool se adequar aos demais times presentes na Phoebus, essa customização foi definida após reunião com a diretoria onde ela será executada em toda a esteira de testes da empresa, adição dos demais adquirentes. As customizações futuras consistem em habilitar uma variável de leitura de logs específicos ou de forma resumida apenas testando partes necessárias para cada equipe.

6. Referências

Bertolino, A. (2007). Software Testing Research: Achievements, Challenges, Dreams. 2007 Future of Software Engineering, IEEE Computer Society.

Graham, D., Van Veenendaal, E., & Evans, I. (2008). Foundations of software testing: ISTQB certification: Cengage Learning EMEA

Hooda, I., & Chhillar, R. S. (2015). Software test process, testing types and techniques. International Journal of Computer Applications, 111(13).

Kan, S. H. (2002). Metrics and models in software quality engineering: Addison-Wesley Longman Publishing Co., Inc

Mayers, G. et al (2004). The Art of Software Testing. Wiley. ISBN 0471469122

Prof . Swati Dubey, P. S. T., Prof .Dipti Dighe. (2017). STUDY ON VARIOUS PHASES OF SOFTWARE