Disciplina: Trabalho de Conclusão de Curso I – BCC

Caro orientando,

segue abaixo o Termo de Compromisso, as DUAS revisões do seu pré-projeto contendo a avaliação do professor "avaliador" e professor "TCC1", junto com as avaliações da defesa na banca de qualificação. É muito importante que revise com cuidado e discuta possíveis dúvidas decorrente das revisões com o seu professor orientador, e com o professor de TCC1. Sempre procure fazer todos os ajustes solicitados, até mesmo o menores detalhes, pois todos são importantes e irão refletir na sua nota nesta disciplina. Mas, caso o professor orientador julgue que algumas anotações das revisões não devam ser feitas, ou mesmo que sejam feitas de forma diferente a solicitada pelo revisor, anexe ao final do seu projeto a ficha "Projeto: Observações – Professor Orientador" disponível no material da disciplina, e justifique o motivo.
Lembrem que agora o limite de páginas do projeto é no máximo 12 (doze) páginas. E que a seção de "Revisão Bibliográfica" deve ser complementada.

Atenciosamente,

Digitally signed by Dalton Solano dos Reis:66568803987
Date: 2021.10.28 12:50:43 -03'00'

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

TERMO DE COMPROMISSO

I – IDENTIFICAÇÃO DO ALUNO	
Nome:	Jonathan Luiz de Lara
CV Lattes:	http://lattes.cnpq.br/0798742029436658
E-mail:	jolara@furb.br
Telefone:	(47) 99148-1401

II – IDENTIFICAÇÃO DO TRABALHO	
Título provisório:	FERRAMENTA PARA TESTES INTEGRADOS EM JAVA
Orientador:	Dalton Solano dos Reis
Coorientador (se houver):	
Linha de Pesquisa:	☐ Tecnologias aplicadas à informática na educação ☒ Tecnologias aplicadas ao desenvolvimento de sistemas

III – COMPROMISSO DE REALIZAÇÃO DO TCC

Eu (aluno), Jonathan Luiz de Lara
comprometo-me a realizar o trabalho proposto no semestre 2022/1 , de acordo com as normas e os prazos determinados pela FURB, conforme previsto na resolução nº.20/2016.
Assinatura: NÃO É NECESSÁRIO – Encaminhar por mail ao orientador

IV – COMPROMISSO DE ORIENTAÇÃO

Eu (orientador), Dalton Solano dos Reis
comprometo-me a orientar o trabalho proposto no semestre 2022/1 , de acordo com as normas e os prazos determinados pela FURB, conforme previsto na resolução nº.20/2016.
Assinatura: NÃO É NECESSÁRIO – Encaminhar por mail ao professor de TCC I

Blumenau, 21 de Setembro de 2021

CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2021/2

FERRAMENTA PARA TESTES INTEGRADOS EM JAVA

Jonathan Luiz de Lara

Dalton Solano dos Reis - Orientador

1 INTRODUÇÃO

Software é uma das construções humanas mais complexas. Portanto, é compreensível que sistemas de software estejam sujeitos aos mais variados tipos de erros e inconsistências. Para evitar que tais erros cheguem aos usuários finais e causem prejuízos de valor incalculável, é fundamental introduzir atividades de teste em projetos de desenvolvimento de software. De fato, teste é uma das práticas de programação mais valorizadas hoje em dia, em qualquer tipo de software. É também uma das práticas que sofreram mais transformações nos anos recentes (VALENTE, 2019).

O objetivo do teste de integração deixa de ser o teste de uma unidade pequena de código, como uma classe apenas. Em vez disso, testes de integração exercitam um serviço completo, isto é, uma funcionalidade de maior granularidade do sistema. Por isso, eles envolvem mais classes, às vezes de pacotes distintos. Também envolvem dependências e sistemas reais, como bancos de dados e serviços remotos. Além disso, quando se implementa testes de integração não faz mais sentido criar objetos que simulam o comportamento de objetos reais de forma controlada, conhecidos tecnicamente como *mocks* ou *stubs*. Como são testes maiores, eles levam mais tempo para executar e, consequentemente, são chamados com menor frequência (VALENTE, 2019). Teste de integração é o processo de verificar se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do software (LIMA; TRAVASSOS, 2004).

Um dos objetivos do teste de integração é detectar problemas junto às interfaces, ou seja, erros que aparecem com a junção das partes do sistema. Antes da junção são realizados testes nos pedaços individuais do sistema, os erros encontrados são corrigidos, outros erros aparentemente "menores", nem são notados. Mais tarde, com as partes do sistema integradas, aqueles erros "menores" podem se apresentar não tão pequenos assim, e causar falhas inaceitáveis no sistema (PRESSMAN, 1995).

Segundo Pressman (2011), o clamor por qualidade de software iniciou quando os softwares passaram a se tornar cada vez mais integrado nas atividades das nossas vidas. Mediante a evolução tecnológica que vai em direção a atender as crescentes inovações, e estas acabam criando maiores expectativas nas pessoas, as exigências por softwares confiáveis e que atendam fielmente ao que se promete se tornam requisitos obrigatórios, inegociáveis, demandando um bom projeto de software, capaz de garantir através de testes as aferições de qualidade do produto em suas partes. Sendo isto uma necessidade primária, o ambiente corporativo procura evoluir no processo de testes para maximizar a qualidade dos seus softwares.

As aplicações tendo cada vez mais complexidade trazem consigo um aumento do número de variáveis envolvidas. Segundo Bartié (2002), nesse cenário todas as variáveis envolvidas no processo de desenvolvimento têm um nível crescente de complexidade. Com isso os riscos de mau funcionamento aumentam proporcionalmente ao aumento dessas variáveis, tornando difícil construir softwares com o nível de qualidade desejado.

Diante do contexto apresentado, este trabalho consisti em oferecer ao profissional que está desenvolvendo o software alguns recursos para realizar testes de integração nas suas funcionalidades, construindo os casos de testes, informando neles os dados de entrada e os dados que devem ser produzidos pela funcionalidade. Com estas definições de análise feitas pelo profissional a ferramenta será capaz de aferir se para aquele conjunto de valores de entrada a funcionalidade está tendo o resultado que se espera.

1.1 OBJETIVOS

Este trabalho tem como objetivo oferecer uma ferramenta para auxiliar no processo de desenvolvimento dos testes de integração, contribuindo para aumentar a qualidade da entrega do software.

Os objetivos específicos são:

Comentado [LPdAK1]: Quem disse isso?

São duas frases fortes que precisam de referência.

Coloquei destacado em amarelo palavras que são fortes que fazem com que a frase precise de uma referência bibliográfica.

Comentado [LPdAK2]: Faltou uma ligação entre esses dois parágrafos.
Relacionar o teste ou o motivo dele antes.

Comentado [LPdAK3]: Achei muito coloquial "pedaços individuais" – o que isso significa?

Comentado [LPdAK4]: Reescrever, não ficou legal.

Comentado [LPdAK5]: Ficou muito coloquial, reescrever

Comentado [LPdAK6]: Deve ser impessoal. (não usar 1ª pessoa)

Comentado [LPdAK7]: Remover ,

Comentado [LPdAK8]: referenciar

Comentado [LPdAK9]: Novamente faltou um link com o parágrafo anterior.

Também falta referenciar.

Comentado [LPdAK10]: ...pelo profissional, a ferramenta será....

Comentado [LPdAK11]: Remover ENTERs no documento.

Comentado [LPdAK12]: Como você vai provar isso?

a) desenvolver uma ferramenta para facilitar a realização de testes de integração para as funcionalidades do software (módulos, classes e métodos);

b) apresentar ao desenvolvedor os resultados dos testes, sinalizando se a funcionalidade produziu o resultado esperado ou não;

c) permitir que funcionalidades que não foram projetadas para serem testadas por um outro software possam ser usadas por essa ferramenta de forma mais simples e segura;

d) oferecer para o projeto uma bateria de testes que pode ser utilizada recorrentemente, conforme o software vai sendo modificado.

Comentado [LPdAK13]: Qual a diferença desse para o objetivo geral?

Comentado [LPdAK14]: RF – o que você quer com essa funcionalidade? Este é o objetivo.

Comentado [LPdAK15]: Como garantir que é mais simples e segura? E mais simples e segura do que quem?

Comentado [LPdAK16]: Que projeto?

2 TRABALHOS CORRELATOS

Nesta seção serão apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto neste trabalho. Na seção 2.1 será apresentado o trabalho sobre um estudo de caso sobre automatização de testes de software na empresa de desenvolvimento Softpan (FERNANDES; FONSECA, 2019). Na seção 2.2 será apresentado o trabalho de um estudo de caso sobre automação de software utilizando um *framework* da IBM (FANTINATO, 2019). Por fim, na seção 2.3 será apresentado o trabalho sobre o estudo que aplica testes automatizados utilizando a ferramenta Rational Visual Test em um produto comercial desenvolvido por uma empresa privada de Blumenau (TOMELIN, 2001).

2.1 AUTOMAÇÃO DE TESTES DE SOFTWARE: ESTUDO DE CASO DA EMPRESA SOFTPLAN

O trabalho de Fernandes e Fonseca (2019) tem como objetivo geral apresentar um estudo de caso sobre automatização de testes de software na empresa de desenvolvimento Softplan. Este trabalho foi realizado avaliando pesquisas bibliográficas sobre qualidade, testes de software e automatização deste processo, estudo sobre como a automação de testes pode auxiliar à equipe quanto a redução de tempo e cobertura de testes, realizado comparativo entre os testes manuais e automatizados apresentando suas vantagens e desvantagens.

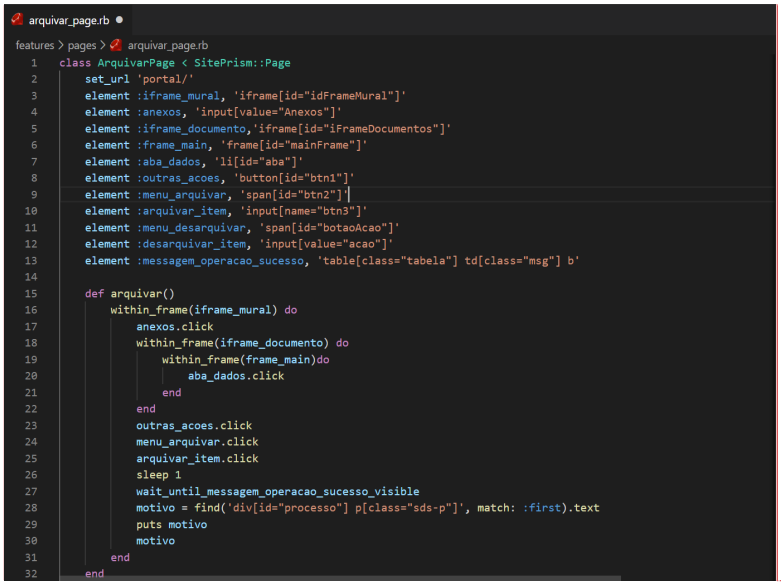
Comentado [LPdAK17]: Trocar “Este trabalho” por “O trabalho de Fernandes e Fonsceca (2019)”

Durante o desenvolvimento do trabalho foram analisados alguns *frameworks* para a realização da automação dos cenários de testes, como o framework Robot e a ferramenta comercial Ranorex. Porém o software que demonstrou maior aderência ao propósito do trabalho de Fernandes e Fonseca (2019) foi a ferramenta Cucumber. Essa escolha ocorreu com base em fatores técnicos e sua aderência ao propósito do trabalho. A Figura 1 demonstra como é feita a implementação dos elementos visuais que serão utilizados na implementação da automação de teste do trabalho de Fernandes e Fonseca (2019).

Comentado [LPdAK18]: Padronizar em itálico ao longo de todo o texto.

Comentado [LPdAK19]: Não inciiar frase com Porém

Figura 1 - Implementação das pages em Ruby



Fonte: Fernandes e Fonseca (2019).

Comentado [LPdAK20]: Formatar todas as legendas como TF-LEgenda

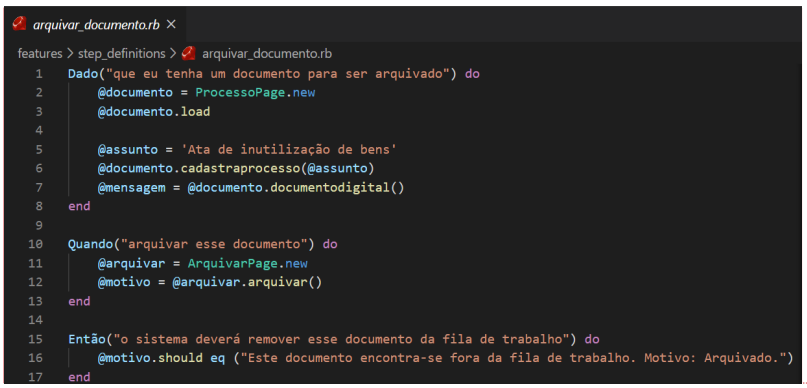
Comentado [LPdAK21]: Não é figura. É quadro. Rever na legenda e também no txto.

Explicar com detalhes o quadro. Não entendi olhando para ele como é feita a implementação da automação do teste...

A Figura 2 apresenta a estrutura dos passos definidos para a execução do caso de teste, estruturado segundo a prática Behavior Driven Development (BDD) e pronto para execução pela ferramenta Cucumber para aferir o resultado de cada um.

Comentado [LPdAK22]: Remover todos os ENTERs no texto!

Figura 2 - Step_definitions



Comentado [LPdAK23]: Mesmo comentário que figura anterior.
Não é figura é um quadro e deve ser melhor explicado.

Fonte: Fernandes e Fonseca (2019).

A Figura 3 apresenta um relatório separado por *feature* informando quantos casos de testes passaram e quantos falharam. E assim, informando a duração da execução dos testes e sua respectiva situação.

Comentado [LPdAK24]: Não iniciar frase com "E"

Figura 3 - Relatório de status dos testes

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
Arquivamento	24	0	0	0	0	24	6	0	6	3:16.290	Passed
Assinatura	32	0	0	0	0	32	8	0	8	4:32.534	Passed
Cadastros básicos	8	0	0	0	0	8	2	0	2	24.091	Passed

Fonte: Fernandes e Fonseca (2020).

2.2 AUTOTEST – UM FRAMEWORK REUTILIZÁVEL PARA A AUTOMAÇÃO DE TESTES FUNCIONAL DE SOFTWARE

Comentado [LPdAK25]: Esse trabalho está muito sucinto. Se não achasse mais informações sugiro rever o correlato.

O trabalho de Fantinato (2019) tem como objetivo central apresentar um estudo de caso sobre automatização de testes de software utilizando o framework AutoTest em um software que gerencia e controla faturamentos. Este trabalho foi realizado com uma metodologia de coleta de métricas, para comprovar que o investimento na realização da atividade de automação é plenamente vantajoso a médio e longo prazo, visto que o maior trabalho é para realizar a análise, identificando os cenários, as pré-condições, o conjunto de elementos de entrada e a identificação dos elementos que devem ser produzidos. Para isso foi elaborado um plano para realizar o trabalho de Fantinato (2019), que consiste em medir o esforço das atividades na realização da atividade de teste de forma manual e comparando com a forma de teste automatizada. Para isso todas as atividades dos dois modelos foram aferidas, com o objetivo de dar transparência sobre este comparativo. O objetivo do trabalho de Fantinato (2019) foi apresentar os resultados, dar clareza e segurança a respeito do investimento e dar detalhes sobre seus ganhos e resultados. A Figura 4 apresenta as métricas dos testes realizados em um ambiente manual e em um ambiente automatizado.

Comentado [LPdAK26]: Parece que alguém fez uma plano para realizar o trabalho de Fantinato... reescrever.

Figura 4 – Métricas coletadas

Tipo de Métrica	Teste Manual	Teste Automatizado
Número de Casos de Teste Executados	930	1644
Cobertura Funcional dos Casos de Teste[1]	65 %	88 %
Erros Detectados	174	+ 33
Tempo de Projeto de Teste	101 h	101 h
Tempo de uma Execução Completa dos Casos de Teste	123 h	14 h
Análise dos Resultados e Registro de Erros	34 h	28 h

Fonte: Fantinato (2019).

Comentado [LPdAK27]: É um quadro.

2.3 TESTE DE SOFTWARE A PARTIR DA FERRAMENTA VISUAL TEST

O trabalho de Tomelin (2001) tem como objetivo apresentar um estudo sobre a utilização da ferramenta Rational Visual Test. Apresentando sua camada de comunicação com o usuário, sua linguagem de elaboração de casos de testes e os resultados obtidos.

Comentado [LPdAK28]: Não tem algo mais recente?

Comentado [LPdAK29]: Não iniciar frase no gerúndio, pois ela parece que não tem início.

Este trabalho foi realizado avaliando pesquisas bibliográficas sobre o desenvolvimento da disciplina de testes defendida pela engenharia de software, abordando conceitualmente todas as atividades e separando os diferentes tipos de testes que essa disciplina possui. Desde a definição conceitual sobre os limites dos testes, as baterias de testes, o seu planejamento, seus planos e suas especificações.

Comentado [LPdAK30]: Substituir pelo nome do autor do trabalho, pois assim parece que é o seu trabalho

Comentado [LPdAK31]: Confuso. O que acontece desde a definição conceitual?

Para o desenvolvimento deste trabalho, ele tomou como ponto de partida os processos adotados nas atividades de testes da empresa WK Sistemas. Permitindo desta forma verificar deficiências em relação os conceitos empregados no trabalho com a prática adotada na empresa. Desta forma ele utilizou um software da empresa para analisar e iniciar a elaboração da implementação dos casos de testes, implementando um script de teste para cada caso de teste, incluindo em cada script além dos testes do software controles de operação no repositório de dados, visto a dependência do software com repositório, realizando a restauração do repositório no inicio do script e fazendo o backup ao final da execução do script.

Comentado [LPdAK32]: O seu?

Comentado [LPdAK33]: Quem?

Comentado [LPdAK34]: Não iniciar frase no gerúndio, pois ela parece que não tem início.

Comentado [LPdAK35]:

Comentado [LPdAK36]: início

A Figura 5 demonstra como os casos de testes são implementados na ferramenta Visual Test, demonstrando a estrutura de controle convencionada pela ferramenta e os passos que precisam ser implementados.

Figura 5 - Linguagem de criação dos testes

```
Scenario "Verifica Saldo"

    'VERIFICA SALDO DO ATIVO
    WEditSetText("#1211","17") 'Conta
    sleep 1
    Play "{TAB}"
    WEditSetText("#1212","01/01/99") 'Data Inicial
    sleep 1
    Play "{TAB}"
    WEditSetText("#1213","31/12/01") 'Data Final
    Sleep 1
    Play "{TAB}"
    WButtonClick("OK")
    Sleep 1
    IF ((StaticText("#1060") = "95.028,00") and (StaticText("#1061") = "107.724,00")) and (StaticText("#1062") = "237.304,00") then
            certo1 = 1
    else
        LogErros(Cabecalho,"O SALDO DO ATIVO NÃO CONFERE, algum lançamento não foi efetuado com  sucesso.", Cabecalho)
    end if
    'VERIFICA SALDO DO PASSIVO
    WEditSetText("#1211","939") 'Conta
    sleep 1
    Play "{TAB}"
    WEditSetText("#1212","01/01/99") 'Data Inicial
    sleep 1
    Play "{TAB}"
    WEditSetText("#1213","31/12/01") 'Data Final
    Sleep 1
    Play "{TAB}"
    WButtonClick("OK")
    Sleep 1
    IF ((StaticText("#1060") = "8.400,00") and (StaticText("#1061") = "340,00")) and (StaticText("#1062") = "241.940,00") then
            certo2 = 1
    else
        LogErros(Cabecalho,"O SALDO DO PASSIVO NÃO CONFERE, algum lançamento não foi efetuado com sucesso.",Cabecalho)
    end if
    Sleep 1
    tudocerto = (certo1 + certo2)
    IF tudocerto = 2 then
      LogErros(cabecalho,"Não foram encontrados erros",Cabecalho)
      MSGBOX "TUDO CERTO, os LANÇAMENTOS foram efetivados com Sucesso!!!",MB_ICONINFORMATION,"Teste de Lçtos
Normais"
    else
      VisualizaArquivo()
    End IF
    WButtonClick("Cancelar")
    WMenuSelect("&Arquivo\&Sair")
    Maximiza()
End Scenario
```

Fonte: Tomelin (2001).

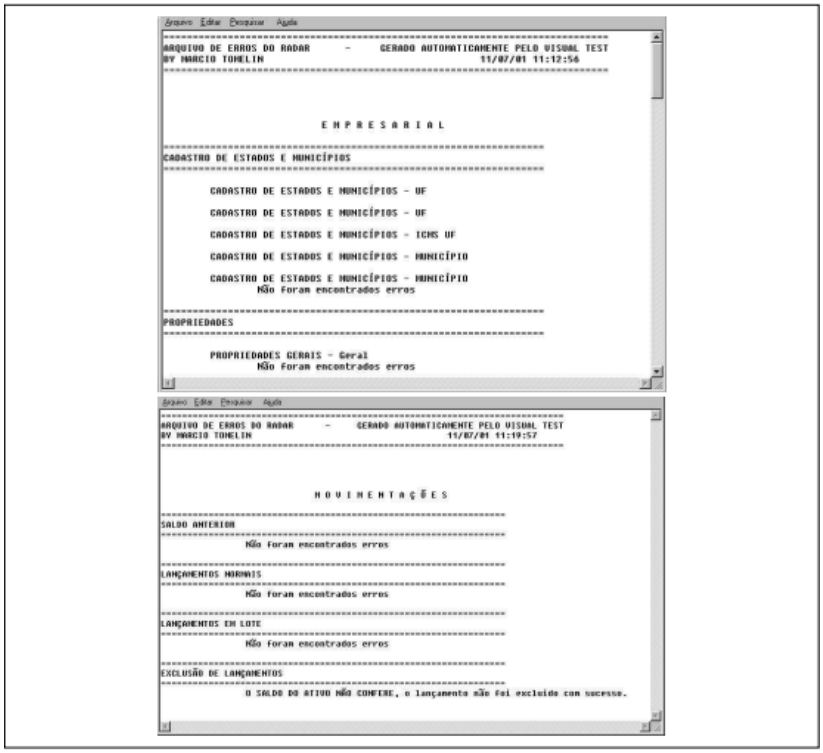
Comentado [LPdAK37]: é um quadro.

Explicar melhor.

A Figura 6 demonstra o resultado da execução dos casos de testes pela ferramenta Visual Test, precisando nesse momento do envolvimento de um profissional para interpretar os resultados e aferir com o resultado esperado.

Comentado [LPdAK38]: Aonde se encontra cada um na figura

Figura 6 - Linguagem de criação dos testes



Fonte: Tomelin (2001).

3 PROPOSTA DA FERRAMENTA

Neste capítulo será apresentado a justificativa para a elaboração deste trabalho, assim como os requisitos funcionais, não funcionais, a metodologia que será seguida em seu desenvolvimento e o seu cronograma de execução.

Comentado [LPdAK39]: apresentadA

3.1 JUSTIFICATIVA

No Quadro 1 é apresentado um comparativo das características mais relevantes entre os trabalhos correlatos. Nas linhas são descritas as características e nas colunas os trabalhos.

Comentado [LPdAK40]: Depois de mudar os quadros anteriores renumerar aqui

Quadro 1 - Comparativo dos trabalhos correlatos

Características \ Trabalhos Correlatos	Fernandes e Fonseca (2019)	Fantinato (2019)	Tomelin (2001)
Bateria de testes por funcionalidade	Sim	Sim	Sim
Registro das funcionalidades	Não	Sim	Sim
Registro dos casos de testes	Não	Sim	Sim
Técnica de record e playback	Não	Sim	Não
Técnica de teste guiado por BDD	Sim	Não	Não
Configuração para camada de pré-condição	Não	Não	Não
Consultas sobre os resultados dos testes	Sim	Sim	Sim
Relatórios sobre os resultados dos testes	Sim	Sim	Não
Gráficos sobre os resultados dos testes	Sim	Sim	Não

Fonte: elaborado pelo autor.

Conforme apresentado no Quadro 1, os trabalhos de Fernandes e Fonseca (2019) e Fantinato (2019) possuem muitas características parecidas ao que elas oferecem, porém, apresentam características distintas para o mecanismo de construir os testes, que é o núcleo de uma ferramenta de testes.

O trabalho de Fernandes e Fonseca (2019) demonstra o funcionamento da ferramenta Cucumber, sua estrutura e o seu princípio de funcionamento, destacando o diferencial da ferramenta que é o recurso que permite a aplicação da prática de desenvolvimento guiado pelo comportamento, conhecida em inglês como Behavior Driven Development (BDD), que é a realização de descrições funcionais em texto simples. Esse recurso traz algumas vantagens, como o fato de poder escrever os testes estruturado em torno do modelo de domínio, ou seja, em torno do contexto conceitual no qual os testes vão ser aplicados. Com essa característica os testes podem ser escritos pelos próprios *stakeholders* enquanto os mesmos definem as regras do domínio e o comportamento que o software deve respeitar. Dessa forma os testes vão evoluindo conforme as regras de negócio e o comportamento funcional são especificados, e é um trabalho que pode ocorrer conjuntamente.

O trabalho do Fantinato (2019) demonstra os resultados obtidos com a adoção de uma ferramenta de testes em um ambiente corporativo, com um software em operação em um ambiente complexo e crítico, demonstrando os benefícios da adoção da ferramenta de automação de testes na disciplina de testes do projeto.

A razão mais forte em investir em pesquisa neste trabalho é a falta de uma ferramenta de apoio na construção de testes para softwares que não foram projetados para testes, ou seja, que possuem uma estrutura de código com muita dependência, com nível de acoplamento alto, trazendo para os testes um trabalho muito grande para a criação da camada das pré-condições, obrigando o profissional a realizar a programação de toda essa fase de pré-condição. Outro ponto é a falta de qualidade dos softwares em operação somado a dificuldade de investimento e pela alta complexidade em reconceber ou refatorar o software.

Diante deste contexto, este trabalho se torna relevante pelo fato de ajudar os profissionais a construírem os casos de testes durante o projeto, diminuindo o esforço na criação da camada de pré-condição e na camada que avalia os resultados esperados. O trabalho proposto irá oferecer configurações para o profissional definir as atividades que devem ser realizadas na camada de pré-condição e informar qual o valor esperado que a ação sob teste deve produzir.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais deste trabalho são:

a) permitir o registro da funcionalidade e dos casos de testes abrangidos por ela;

b) permitir a configuração das pré-condições para cada cenário de teste;

c) permitir consultar as funcionalidades e seus casos de testes que estão sob cobertura de testes pela ferramenta;

d) a ferramenta deverá permitir criar os cenários previstos para a operação de cada classe/método;

e) a ferramenta deverá permitir a execução de uma funcionalidade específica assim como também de um cenário específico;

Comentado [LPdAK41]: Todas as características desse quadro devem estar descritas nos correlatos na seção anterior!

Comentado [LPdAK42]: Aqui deve ser relatado todas as características indicadas das linhas do seu quadro de correlatos. Não consigo ver essas características descritas aqui de forma evidente.

Comentado [LPdAK43]: Evitar parágrafo com uma só frase.

Comentado [LPdAK44]: Não fica legal essa frase.

Pesquisa?! O TCC não é pesquisa...

Comentado [LPdAK45]: Não entendi:

Há falta de ferramenta de apoio ao teste que não é projetada para o teste? Mas pq não usar uma ferramenta projetada para o teste?

Comentado [LPdAK46]: Achei muito confuso. Dividir em frases.

Comentado [LPdAK47]: Não está evidente como a ferramenta fará para executar o teste? Eu vou programar os casos de teste no próprio código-fonte ou é uma ferramenta a parte?

f) a ferramenta deverá apresentar o resultado da execução do cenário, podendo ser “OK” ou “NOK”.

Os requisitos não funcionais deste trabalho são:

a) deverá ser construído na plataforma de desenvolvimento Java versão 11;

b) deverá ser utilizado a prática de Test Driven Development (TDD), utilizando o *framework* Junit;

c) o trabalho deverá ter seus requisitos, seu modelo estrutural e comportamental na Unified Modeling Language (UML), utilizando a ferramenta Star UML.

Comentado [LPdAK48]: A versão 11 não é gratuita.. seria o OpenJDK?

Comentado [LPdAK49]: Tenho minhas duvidas se isso vai aqui

3.3 METODOLOGIA

A construção do trabalho respeitará a seguintes fases:

a) pesquisa bibliográfica: pesquisar as características de código com baixa coesão, código com alto acoplamento, código monolítico, avaliação e testes em estruturas de códigos que não são modulares;

b) elicitação dos requisitos: construir detalhadamente com o uso do diagrama de Caso de Uso da UML os requisitos de acordo com a proposta apresentada;

c) projeto do software: construir o modelo estrutural e comportamental da ferramenta com o uso do Diagrama de classes, objetos e atividades da UML, dando uma visão técnica clara da sua estrutura e do seu comportamento, facilitando estudos, evoluções ou até mesmo correções;

d) prova de conceito: visto a complexidade do tema será necessário realizar alguns ensaios técnicos para aferir se o que se pensa é de fato implementável;

e) construção: fase que será implementada a ferramenta, para todas as camadas, sendo camada de apresentação, aplicação, domínio e infraestrutura. A construção será dirigida por testes, usando a prática de análise de código TDD;

f) testes funcionais: fase que será liberada para o usuário realizar testes que garantam o atendimento do que o usuário espera.

Comentado [LPdAK50]: Acho um pouco coloquial para o texto

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2022									
	fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
pesquisa bibliográfica										
elicitação dos requisitos										
projeto de software										
prova de conceito (POC)										
construção/implementação										
testes										

Fonte: elaborado pelo autor.

Comentado [LPdAK51]: Não é pouco tempo??

Comentado [LPdAK52]: Pouco tempo para um TCC que aborda testes

4 REVISÃO BIBLIOGRÁFICA

Este capítulo descreve de forma breve o assunto que fundamentará o estudo a ser realizado para a construção deste trabalho: código com baixa coesão, alto acoplamento, arquitetura monolítica, nível de testes de integração.

Comentado [LPdAK53]: Não está abordando todos esses assuntos. Colocar aqui na ordem em que aparecem no texto.

Teste consiste na execução de um programa com um conjunto finito de casos, com o objetivo de verificar se ele possui o comportamento esperado. Existem diversos tipos de testes, como testes de unidade (quando se testa uma pequena unidade do código de forma isolada, como uma classe), testes de integração (quando se testa uma unidade com maior granularidade, como um conjunto de classes se comunicando), testes de performance (quando se submete o sistema a uma carga de processamento para verificar seu desempenho) (VALENTE, 2019).

Comentado [LPdAK54]: Somente uma fonte?

Deve falar mais sobre o teste de integração que é o seu foco.

A garantia da qualidade de software é atualmente reconhecida como um assunto que permeia todo o processo de desenvolvimento, os testes e a verificação dos problemas propriamente ditos são tópicos de grande importância no processo de qualidade. Já existe uma discussão técnica para verificar a correção de programas de forma matematicamente rigorosa, mas a conclusão é que a maioria dos sistemas é verificada por meio de testes, infelizmente, tais testes são, na melhor das hipóteses, inexatos. Não é possível garantir que uma unidade de software esteja correta por meio de testes, a menos que fôssemos capazes de executar testes que exaurissem todos os cenários possíveis, sabemos que humanamente isto é muito difícil. Mesmo programas simples podem existir bilhões de caminhos diferentes a serem percorridos. Então testar todos os caminhos possíveis dentro de um programa complexo é uma tarefa impossível (BROOKSHEAR, 2013).

Comentado [LPdAK55]: Referenciar – quem disse?

Comentado [LPdAK56]: verificadO

Comentado [LPdAK57]: Não ficou bom e também não entendi o motivo dessa frase no seu texto.

Comentado [LPdAK58]: Não usar a primeira pessoa

Comentado [LPdAK59]: Não usar a primeira pessoa

Comentado [LPdAK60]: Somente uma referência?

REFERÊNCIAS

BARTIÉ, Alexandre. **Garantia da qualidade de software**. Elsevier, 2002.

BROOKSHEAR, Glenn J. **Ciência da Computação:** Uma visão abrangente. 11. ed. Porto Alegre: Bookman, 2013. 573 p. Tradução Eduardo Kessler Piveta.

FANTINATO, Marcelo. AutoTest – **Um framework reutilizável para a automação de testes funcionais de software.** [2019] Disponível em: <https://www.researchgate.net/profile/Marcelo_Fantinato/publication/229004366_AutoTest-Um_Framework_Reutilizavel_para_a_Automacao_de_Teste_Funcional_de_Software/links/00b7d525a17636e087000000/AutoTest-Um-Framework-Reutilizavel-para-a-Automacao-de-Teste-Funcional-de-Software.pdf>. Acesso em 17 set. 2021.

FERNANDES, Matheus.; FONSECA, Samuel. **Automação de testes de software:** Estudo de caso da empresa softplan. [2019].
Disponível em: <https://www.riuni.unisul.br/bitstream/handle/12345/10127/AUTOMA%c3%87%c3%83O%20DE%20TESTES%20DE%20SOFTWARE-ESTUDO%20DE%20CASO%20DA%20EMPRESA%20SOFTPLAN-TCC_MATHEUS%20FERNANDES-SAMUEL%20TOMKELSKI%20FONSECA.pdf?sequence=1&isAllowed=y> Acesso em 14 set. 2021.

LIMA, G. M. P. S.; Travassos, G. H. **Testes de Integração Aplicados a Software Orientados a Objetos:** Heurísticas para Ordenação de Classes. Relatório Técnico ES632/04, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2004.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 1995.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 2011.

TOMELIN, Marcio. **Testes de software a partir da ferramenta visual test.** [2001]
Disponível em: < http://campeche.inf.furb.br/tccs/BCC/2001-I/2001-1marciotomelinvf.pdf> Acesso em 28 set. 2021.

VALENTE, Marco. **Engenharia de software moderna.** [2019]
Disponível em: <https://docplayer.com.br/178875236-Engenharia-de-software-moderna.html>
Acesso em 14 set. 2021.

Comentado [LPdAK61]: Não tem mais < >

FORMULÁRIO DE AVALIAÇÃO – PROFESSOR AVALIADOR

Avaliador(a): **Luciana Pereira de Araújo Kohler**

Atenção: quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

ASPECTOS AVALIADOS[1]			Atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1.	INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
		O problema está claramente formulado?	X		
	2.	OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?		X	
		Os objetivos específicos são coerentes com o objetivo principal?		X	
	3.	TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?		X	
	4.	JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?		X	
		São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?		X	
		São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?	X		
	5.	REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?		X	
	6.	METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?		X	
		Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?		X	
	7.	REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?		X	
		As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?		X	
ASPECTOS METODOLÓGICOS	8.	LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?		X	
		A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		

4.1.1

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DISCIPLINA: TRABALHO DE CONCLUSÃO DE CURSO I
CURSO: CIÊNCIA DA COMPUTAÇÃO - BCC

ATA DA DEFESA: BANCA DO PRÉ-PROJETO

Venho, por meio deste, manifestar minha avaliação sobre a **apresentação** do Pré-Projeto de TCC

realizado pelo(a) acadêmico(a), Jonathan Luiz de Lara no **SEGUNDO SEMESTRE DE 2021**, com o título FERRAMENTA PARA TESTES INTEGRADOS EM JAVA, sob orientação do prof(a). Dalton Solano dos Reis.

A referida apresentação obteve a seguinte nota:

Componente da Banca	**Nota (de 0 a 10)**
Professor(a) Avaliador(a): Luciana Pereira de Araújo Kohler	**8,0**

ATENÇÃO. A nota acima se refere somente a apresentação do pré-projeto e vai ser repassada para o aluno (orientando). Favor preencher os campos acima e enviar por e-mail ao professor de TCC1 (dalton@furb.br). Não passar o arquivo com as anotações da revisão já enviado ao professor de TCC1 para o orientando e nem para o professor orientador. Após o professor de TCC1 receber esta ata preenchida, o professor de TCC1 vai disponibilizar para o orientando/orientador os arquivos com as revisões. Caso julgue necessário fazer mais alguma consideração relacionada ao pré-projeto ou a defesa, favor usar o espaço abaixo.

Observações da apresentação:

Trabalhar os slides para se ter menos textos.

CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2021/2

FERRAMENTA PARA TESTES INTEGRADOS EM JAVA

Jonathan Luiz de Lara

Dalton Solano dos Reis - Orientador

5 INTRODUÇÃO

Software é uma das construções humanas mais complexas. Portanto, é compreensível que sistemas de software estejam sujeitos aos mais variados tipos de erros e inconsistências. Para evitar que tais erros cheguem aos usuários finais e causem prejuízos de valor incalculável, é fundamental introduzir atividades de teste em projetos de desenvolvimento de software. De fato, teste é uma das práticas de programação mais valorizadas hoje em dia, em qualquer tipo de software. É também uma das práticas que sofreram mais transformações nos anos recentes (VALENTE, 2019).

O objetivo do teste de integração deixa de ser o teste de uma unidade pequena de código, como uma classe apenas. Em vez disso, testes de integração exercitam um serviço completo, isto é, uma funcionalidade de maior granularidade do sistema. Por isso, eles envolvem mais classes, às vezes de pacotes distintos. Também envolvem dependências e sistemas reais, como bancos de dados e serviços remotos. Além disso, quando se implementa testes de integração não faz mais sentido criar objetos que simulam o comportamento de objetos reais de forma controlada, conhecidos tecnicamente como *mocks* ou *stubs*. Como são testes maiores, eles levam mais tempo para executar e, consequentemente, são chamados com menor frequência (VALENTE, 2019). Teste de integração é o processo de verificar se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do software (LIMA; TRAVASSOS, 2004).

Um dos objetivos do teste de integração é detectar problemas junto às interfaces, ou seja, erros que aparecem com a junção das partes do sistema. Antes da junção são realizados testes nos pedaços individuais do sistema, os erros encontrados são corrigidos, outros erros aparentemente “menores”, nem são notados. Mais tarde, com as partes do sistema integradas, aqueles erros “menores” podem se apresentar não tão pequenos assim, e causar falhas inaceitáveis no sistema (PRESSMAN, 1995).

Segundo Pressman (2011), o clamor por qualidade de software iniciou quando os softwares passaram a se tornar cada vez mais integrado nas atividades das nossas vidas. Mediante a evolução tecnológica que vai em direção a atender as crescentes inovações, e estas acabam criando maiores expectativas nas pessoas, as exigências por softwares confiáveis e que atendam fielmente ao que se promete se tornam requisitos obrigatórios, inegociáveis, demandando um bom projeto de software, capaz de garantir através de testes as aferições de qualidade do produto em suas partes. Sendo isto uma necessidade primária, o ambiente corporativo procura evoluir no processo de testes para maximizar a qualidade dos seus softwares.

As aplicações tendo cada vez mais complexidade trazem consigo um aumento do número de variáveis envolvidas. Segundo Bartié (2002), nesse cenário todas as variáveis envolvidas no processo de desenvolvimento têm um nível crescente de complexidade. Com isso os riscos de mau funcionamento aumentam proporcionalmente ao aumento dessas variáveis, tornando difícil construir softwares com o nível de qualidade desejado.

Diante do contexto apresentado, este trabalho consisti em oferecer ao profissional que está desenvolvendo o software alguns recursos para realizar testes de integração nas suas funcionalidades, construindo os casos de testes, informando neles os dados de entrada e os dados que devem ser produzidos pela funcionalidade. Com estas definições de análise feitas pelo profissional a ferramenta será capaz de aferir se para aquele conjunto de valores de entrada a funcionalidade está tendo o resultado que se espera.

5.1 OBJETIVOS

Este trabalho tem como objetivo oferecer uma ferramenta para auxiliar no processo de desenvolvimento dos testes de integração, contribuindo para aumentar a qualidade da entrega do software.

Os objetivos específicos são:

Comentado [MCL62]: Sugiro falar sobre diferentes tipos de teste antes de se fixar em um apenas.

Comentado [MCL63]: Plural. Vou assinalar em amarelo eventuais erros de redação.

Comentado [MCL64]: ,

Comentado [MCL65]: Essa parte não deve estar no objetivo. É um resultado esperado.

Comentado [MCL66]: Você não colocou objetivos mas sim requisitos da sua ferramenta. Pense em como vc vai validar a sua solução e não como o usuário irá utilizar.
Outra questão: use os estilos corretos em cada tipo de parágrafo do texto. Aqui há problemas de espaçamento por não ter usado o estilo TF-Alinea.

e) desenvolver uma ferramenta para facilitar a realização de testes de integração para as funcionalidades do software (módulos, classes e métodos);

f) apresentar ao desenvolvedor os resultados dos testes, sinalizando se a funcionalidade produziu o resultado esperado ou não;

g) permitir que funcionalidades que não foram projetadas para serem testadas por um outro software possam ser usadas por essa ferramenta de forma mais simples e segura;

h) oferecer para o projeto uma bateria de testes que pode ser utilizada recorrentemente, conforme o software vai sendo modificado.

6 TRABALHOS CORRELATOS

Nesta seção serão apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto neste trabalho. Na seção 2.1 será apresentado o trabalho sobre um estudo de caso sobre automatização de testes de software na empresa de desenvolvimento Softpan (FERNANDES; FONSECA, 2019). Na seção 2.2 será apresentado o trabalho de um estudo de caso sobre automação de software utilizando um *framework* da IBM (FANTINATO, 2019). Por fim, na seção 2.3 será apresentado o trabalho sobre o estudo que aplica testes automatizados utilizando a ferramenta Rational Visual Test em um produto comercial desenvolvido por uma empresa privada de Blumenau (TOMELIN, 2001).

Comentado [MCL67]: Não repetir palavras na mesma frase.

Comentado [MCL68]: Tem vários autores. Corrigir em todo o trabalho.

Comentado [MCL69]: Esse correlato tem 20 anos e outro que você colocou como sendo de 2019 é de 2004. Veja se consegue publicações mais atuais.

6.1 AUTOMAÇÃO DE TESTES DE SOFTWARE: ESTUDO DE CASO DA EMPRESA SOFTPLAN

O trabalho de Fernandes e Fonseca (2019) tem como objetivo geral apresentar um estudo de caso sobre automatização de testes de software na empresa de desenvolvimento Softplan. Este trabalho foi realizado avaliando pesquisas bibliográficas sobre qualidade, testes de software e automatização deste processo, estudo sobre como a automação de testes pode auxiliar à equipe quanto a redução de tempo e cobertura de testes, realizado comparativo entre os testes manuais e automatizados apresentando suas vantagens e desvantagens.

Durante o desenvolvimento do trabalho foram analisados alguns *frameworks* para a realização da automação dos cenários de testes, como o framework Robot e a ferramenta comercial Ranorex. Porém o software que demonstrou maior aderência ao propósito do trabalho de Fernandes e Fonseca (2019) foi a ferramenta Cucumber. Essa escolha ocorreu com base em fatores técnicos e sua aderência ao propósito do trabalho. A Figura 1 demonstra como é feita a implementação dos elementos visuais que serão utilizados na implementação da automação de teste do trabalho de Fernandes e Fonseca (2019).

Comentado [MCL70]: Vai ser feito ainda?

Comentado [MCL71]: Eu não consigo relacionar o que escreve aqui com a figura. Inclusive a legenda da figura fala di Ruby que não é explicado no texto.

Figura 1 - Implementação das pages em Ruby

arquivar_page.rb

features > pages > arquivar_page.rb

```ruby
class ArquivarPage < SitePrism::Page
    set_url 'portal/'
    element :iframe_mural, 'iframe[id="idFrameMural"]'
    element :anexos, 'input[value="Anexos"]'
    element :iframe_documento,'iframe[id="iFrameDocumentos"]'
    element :frame_main, 'frame[id="mainFrame"]'
    element :aba_dados, 'li[id="aba"]'
    element :outras_acoes, 'button[id="btn1"]'
    element :menu_arquivar, 'span[id="btn2"]'
    element :arquivar_item, 'input[name="btn3"]'
    element :menu_desarquivar, 'span[id="botaoAcao"]'
    element :desarquivar_item, 'input[value="acao"]'
    element :messagem_operacao_sucesso, 'table[class="tabela"] td[class="msg"] b'

    def arquivar()
        within_frame(iframe_mural) do
            anexos.click
            within_frame(iframe_documento) do
                within_frame(frame_main)do
                    aba_dados.click
                end
            end
            outras_acoes.click
            menu_arquivar.click
            arquivar_item.click
            sleep 1
            wait_until_messagem_operacao_sucesso_visible
            motivo = find('div[id="processo"] p[class="sds-p"]', match: :first).text
            puts motivo
            motivo
        end
    end
```

Fonte: Fernandes e Fonseca (2019).

Comentado [MCL72]: Trata-se de um quadro pois o conteúdo é apenas textual.

Comentado [MCL73]: itálico

Comentado [MCL74]: Utilize corretamente os estilos de parágrafo pois não há espaço entre legenda, figura e fonte. Corrigir todas.

A Figura 2 apresenta a estrutura dos passos definidos para a execução do caso de teste, estruturado segundo a prática Behavior Driven Development (BDD) e pronto para execução pela ferramenta Cucumber para aferir o resultado de cada um.

Figura 2 - Step_definitions

arquivar_documento.rb

features > step_definitions > arquivar_documento.rb

```
1  Dado("que eu tenha um documento para ser arquivado") do
2      @documento = ProcessoPage.new
3      @documento.load
4
5      @assunto = 'Ata de inutilização de bens'
6      @documento.cadastraprocesso(@assunto)
7      @mensagem = @documento.documentodigital()
8  end
9
10 Quando("arquivar esse documento") do
11     @arquivar = ArquivarPage.new
12     @motivo = @arquivar.arquivar()
13 end
14
15 Então("o sistema deverá remover esse documento da fila de trabalho") do
16     @motivo.should eq ("Este documento encontra-se fora da fila de trabalho. Motivo: Arquivado.")
17 end
```

Fonte: Fernandes e Fonseca (2019).

A Figura 3 apresenta um relatório separado por *feature* informando quantos casos de testes passaram e quantos falharam. E assim, informando a duração da execução dos testes e sua respectiva situação.

Figura 3 - Relatório de status dos testes

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
Arquivamento	24	0	0	0	0	24	6	0	6	3:16.290	Passed
Assinatura	32	0	0	0	0	32	8	0	8	4:32.534	Passed
Cadastros básicos	8	0	0	0	0	8	2	0	2	24.091	Passed

Fonte: Fernandes e Fonseca (2020).

6.2 AUTOTEST – UM FRAMEWORK REUTILIZÁVEL PARA A AUTOMAÇÃO DE TESTES FUNCIONAL DE SOFTWARE

O trabalho de Fantinato (2019) tem como objetivo central apresentar um estudo de caso sobre automatização de testes de software utilizando o framework AutoTest em um software que gerencia e controla faturamentos. Este trabalho foi realizado com uma metodologia de coleta de métricas, para comprovar que o investimento na realização da atividade de automação é plenamente vantajoso a médio e longo prazo, visto que o maior trabalho é para realizar a análise, identificando os cenários, as pré-condições, o conjunto de elementos de entrada e a identificação dos elementos que devem ser produzidos. Para isso foi elaborado um plano para realizar o trabalho de Fantinato (2019), que consiste em medir o esforço das atividades na realização da atividade de teste de forma manual e comparando com a forma de teste automatizada. Para isso todas as atividades dos dois modelos foram aferidas, com o objetivo de dar transparência sobre este comparativo. O objetivo do trabalho de Fantinato (2019) foi apresentar os resultados, dar clareza e segurança a respeito do investimento e dar detalhes sobre seus ganhos e resultados. A Figura 4 apresenta as métricas dos testes realizados em um ambiente manual e em um ambiente automatizado.

Comentado [MCL75]: Não existem linhas em branco no projeto.

Comentado [MCL76]: Não inicie frases com "E".

Comentado [MCL77]: Com o que está descrito aqui não é possível entender o que é o seu correlato. Duas telas e uma tabela. Para caracterizar um correlato você deve ver seus objetivos, funcionalidades, recursos técnicos e resultados alcançados.

Figura 4 – Métricas coletadas

Tipo de Métrica	Teste Manual	Teste Automatizado
Número de Casos de Teste Executados	930	1644
Cobertura Funcional dos Casos de Teste[1]	65 %	88 %
Erros Detectados	174	+ 33
Tempo de Projeto de Teste	101 h	101 h
Tempo de uma Execução Completa dos Casos de Teste	123 h	14 h
Análise dos Resultados e Registro de Erros	34 h	28 h

Fonte: Fantinato (2019).

Comentado [MCL78]: Também desse correlato se tira muito pouca coisa para entender a relação com seu trabalho.

6.3 TESTE DE SOFTWARE A PARTIR DA FERRAMENTA VISUAL TEST

O trabalho de Tomelin (2001) tem como objetivo apresentar um estudo sobre a utilização da ferramenta Rational Visual Test. Apresentando sua camada de comunicação com o usuário, sua linguagem de elaboração de casos de testes e os resultados obtidos.

Comentado [MCL79]: Frase incompleta.

Também evite parágrafos curtos.

Este trabalho foi realizado avaliando pesquisas bibliográficas sobre o desenvolvimento da disciplina de testes defendida pela engenharia de software, abordando conceitualmente todas as atividades e separando os diferentes tipos de testes que essa disciplina possui. Desde a definição conceitual sobre os limites dos testes, as baterias de testes, o seu planejamento, seus planos e suas especificações.

Comentado [MCL80]: De onde saiu esse texto? Referencie.

Para o desenvolvimento deste trabalho, ele tomou como ponto de partida os processos adotados nas atividades de testes da empresa WK Sistemas. Permitindo desta forma verificar deficiências em relação os conceitos empregados no trabalho com a prática adotada na empresa. Desta forma ele utilizou um software da empresa para analisar e iniciar a elaboração da implementação dos casos de testes, implementando um script de teste para cada caso de teste, incluindo em cada script além dos testes do software controles de operação no repositório de dados, visto a dependência do software com repositório, realizando a restauração do repositório no inicio do script e fazendo o backup ao final da execução do script.

Comentado [MCL81]: O seu?

Comentado [MCL82]: Frases iniciadas em gerúndio ou complementam ou são complementadas. Nunca estão assim sozinhas.

Comentado [MCL83]: Itálico. Corrigir todos.

Comentado [MCL84]: itálico

A Figura 5 demonstra como os casos de testes são implementados na ferramenta Visual Test, demonstrando a estrutura de controle convencionada pela ferramenta e os passos que precisam ser implementados.

Comentado [MCL85]: Melhorar construção da frase pois está difícil entender. Também precisa da referência pois o texto não é de um trabalho seu.

Figura 5 - Linguagem de criação dos testes

Comentado [MCL86]: Também é um quadro.

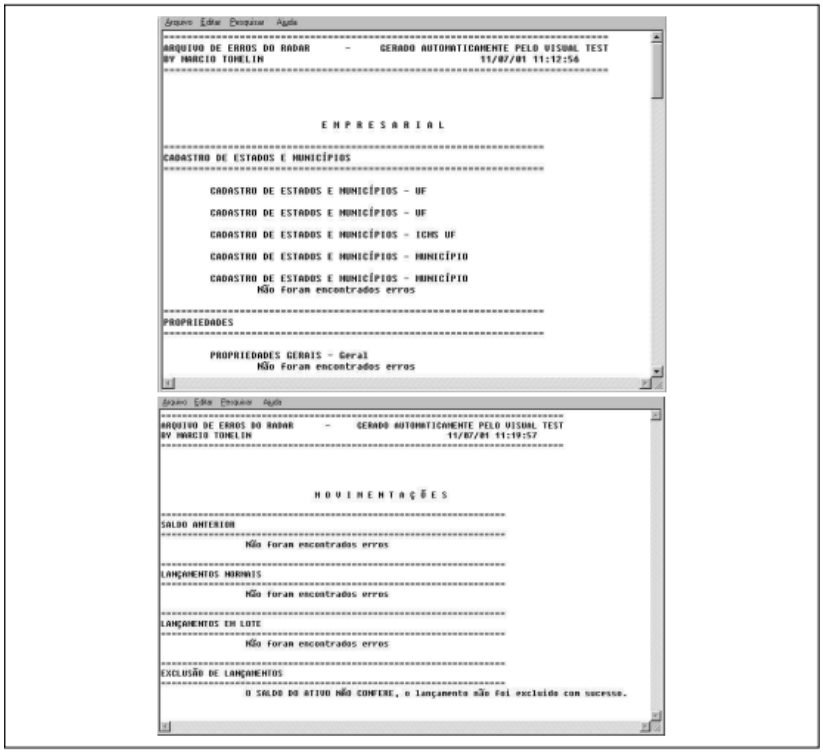
```
Scenario "Verifica Saldo"

    'VERIFICA SALDO DO ATIVO
    WEditSetText("#1211","17") 'Conta
    sleep 1
    Play "{TAB}"
    WEditSetText("#1212","01/01/99") 'Data Inicial
    sleep 1
    Play "{TAB}"
    WEditSetText("#1213","31/12/01") 'Data Final
    Sleep 1
    Play "{TAB}"
    WButtonClick("OK")
    Sleep 1
    IF ((StaticText("#1060") = "95.028,00") and (StaticText("#1061") = "107.724,00")) and (StaticText("#1062") = "237.304,00") then
            certo1 = 1
    else
        LogErros(Cabecalho,"O SALDO DO ATIVO NÃO CONFERE, algum lançamento não foi efetuado com  sucesso.", Cabecalho)
    end if
    'VERIFICA SALDO DO PASSIVO
    WEditSetText("#1211","939") 'Conta
    sleep 1
    Play "{TAB}"
    WEditSetText("#1212","01/01/99") 'Data Inicial
    sleep 1
    Play "{TAB}"
    WEditSetText("#1213","31/12/01") 'Data Final
    Sleep 1
    Play "{TAB}"
    WButtonClick("OK")
    Sleep 1
    IF ((StaticText("#1060") = "8.400,00") and (StaticText("#1061") = "340,00")) and (StaticText("#1062") = "241.940,00") then
            certo2 = 1
    else
        LogErros(Cabecalho,"O SALDO DO PASSIVO NÃO CONFERE, algum lançamento não foi efetuado com sucesso.",Cabecalho)
    end if
    Sleep 1
    tudocerto = (certo1 + certo2)
    IF tudocerto = 2 then
      LogErros(cabecalho,"Não foram encontrados erros",Cabecalho)
      MSGBOX "TUDO CERTO, os LANÇAMENTOS foram efetivados com Sucesso!!!",MB_ICONINFORMATION,"Teste de Lçtos
Normais"
    else
      VisualizaArquivo()
    End IF
    WButtonClick("Cancelar")
    WMenuSelect("&Arquivo\&Sair")
    Maximiza()
End Scenario
```

Fonte: Tomelin (2001).

A Figura 6 demonstra o resultado da execução dos casos de testes pela ferramenta Visual Test, precisando nesse momento do envolvimento de um profissional para interpretar os resultados e aferir com o resultado esperado.

Figura 6 - Linguagem de criação dos testes



Fonte: Tomelin (2001).

Comentado [MCL87]: Os seus correlatos precisam ser reescritos.

7 PROPOSTA DA FERRAMENTA

Neste capítulo será apresentado a justificativa para a elaboração deste trabalho, assim como os requisitos funcionais, não funcionais, a metodologia que será seguida em seu desenvolvimento e o seu cronograma de execução.

Comentado [MCL88]: tirar

7.1 JUSTIFICATIVA

No Quadro 1 é apresentado um comparativo das características mais relevantes entre os trabalhos correlatos. Nas linhas são descritas as características e nas colunas os trabalhos.

Quadro 2 - Comparativo dos trabalhos correlatos

Características \ Trabalhos Correlatos	Fernandes e Fonseca (2019)	Fantinato (2019)	Tomelin (2001)
Bateria de testes por funcionalidade	Sim	Sim	Sim
Registro das funcionalidades	Não	Sim	Sim
Registro dos casos de testes	Não	Sim	Sim
Técnica de record e playback	Não	Sim	Não
Técnica de teste guiado por BDD	Sim	Não	Não
Configuração para camada de pré-condição	Não	Não	Não
Consultas sobre os resultados dos testes	Sim	Sim	Sim
Relatórios sobre os resultados dos testes	Sim	Sim	Não
Gráficos sobre os resultados dos testes	Sim	Sim	Não

Fonte: elaborado pelo autor.

Conforme apresentado no Quadro 1, os trabalhos de Fernandes e Fonseca (2019) e Fantinato (2019) possuem muitas características parecidas ao que elas oferecem, porém, apresentam características distintas para o mecanismo de construir os testes, que é o núcleo de uma ferramenta de testes.

O trabalho de Fernandes e Fonseca (2019) demonstra o funcionamento da ferramenta Cucumber, sua estrutura e o seu princípio de funcionamento, destacando o diferencial da ferramenta que é o recurso que permite a aplicação da prática de desenvolvimento guiado pelo comportamento, conhecida em inglês como Behavior Driven Development (BDD), que é a realização de descrições funcionais em texto simples. Esse recurso traz algumas vantagens, como o fato de poder escrever os testes estruturado em torno do modelo de domínio, ou seja, em torno do contexto conceitual no qual os testes vão ser aplicados. Com essa característica os testes podem ser escritos pelos próprios *stakeholders* enquanto os mesmos definem as regras do domínio e o comportamento que o software deve respeitar. Dessa forma os testes vão evoluindo conforme as regras de negócio e o comportamento funcional são especificados, e é um trabalho que pode ocorrer conjuntamente.

O trabalho do Fantinato (2019) demonstra os resultados obtidos com a adoção de uma ferramenta de testes em um ambiente corporativo, com um software em operação em um ambiente complexo e crítico, demonstrando os benefícios da adoção da ferramenta de automação de testes na disciplina de testes do projeto.

A razão mais forte em investir em pesquisa neste trabalho é a falta de uma ferramenta de apoio na construção de testes para softwares que não foram projetados para testes, ou seja, que possuem uma estrutura de código com muita dependência, com nível de acoplamento alto, trazendo para os testes um trabalho muito grande para a criação da camada das pré-condições, obrigando o profissional a realizar a programação de toda essa fase de pré-condição. Outro ponto é a falta de qualidade dos softwares em operação somado a dificuldade de investimento e pela alta complexidade em reconceber ou refatorar o software.

Diante deste contexto, este trabalho se torna relevante pelo fato de ajudar os profissionais a construírem os casos de testes durante o projeto, diminuindo o esforço na criação da camada de pré-condição e na camada que avalia os resultados esperados. O trabalho proposto irá oferecer configurações para o profissional definir as atividades que devem ser realizadas na camada de pré-condição e informar qual o valor esperado que a ação sob teste deve produzir.

7.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais deste trabalho são:

g) permitir o registro da funcionalidade e dos casos de testes abrangidos por ela;

h) permitir a configuração das pré-condições para cada cenário de teste;

i) permitir consultar as funcionalidades e seus casos de testes que estão sob cobertura de testes pela ferramenta;

j) a ferramenta deverá permitir criar os cenários previstos para a operação de cada classe/método;

k) a ferramenta deverá permitir a execução de uma funcionalidade específica assim como também de um cenário específico;

Comentado [MCL89]: Praticamente não há informação sobre as características apresentadas aqui na descrição dos trabalhos.

Comentado [MCL90]: Itálico

Comentado [MCL91]: idem

Comentado [MCL92]: Como conseguimos observar isso pelo quadro?

Comentado [MCL93]: Essa descrição deveria estar lá no correlato. Aqui é para comparar.

Comentado [MCL94]: Também não faz sentido aqui.

Comentado [MCL95]: Até aqui nada disso havia sido abordado. Nem na introdução, nem nos objetivos e nem nos correlatos.

Comentado [MCL96]: Frase repetitiva. Tente destacar claramente o diferencial tecnológico e prático do seu TCC.

Comentado [MCL97]: Aplicar o estilo correto pois há erro no espaço entre parágrafos.

Comentado [MCL98]: Os outros requisitos não são da ferramenta?

l) a ferramenta deverá apresentar o resultado da execução do cenário, podendo ser “OK” ou “NOK”.

Os requisitos não funcionais deste trabalho são:

d) deverá ser construído na plataforma de desenvolvimento Java versão 11;

e) deverá ser utilizado a prática de Test Driven Development (TDD), utilizando o *framework* Junit;

f) o trabalho deverá ter seus requisitos, seu modelo estrutural e comportamental na Unified Modeling Language (UML), utilizando a ferramenta Star UML.

Comentado [MCL99]: tirar

Comentado [MCL100]: Não há requisitos de interface, de desempenho, de plataforma ou coisas assim?

7.3 METODOLOGIA

A construção do trabalho respeitará a seguintes fases:

g) pesquisa bibliográfica: pesquisar as características de código com baixa coesão, código com alto acoplamento, código monolítico, avaliação e testes em estruturas de códigos que não são modulares;

h) elicitação dos requisitos: construir detalhadamente com o uso do diagrama de Caso de Uso da UML os requisitos de acordo com a proposta apresentada;

i) projeto do software: construir o modelo estrutural e comportamental da ferramenta com o uso do Diagrama de classes, objetos e atividades da UML, dando uma visão técnica clara da sua estrutura e do seu comportamento, facilitando estudos, evoluções ou até mesmo correções;

j) prova de conceito: visto a complexidade do tema será necessário realizar alguns ensaios técnicos para aferir se o que se pensa é de fato implementável;

k) construção: fase que será implementada a ferramenta, para todas as camadas, sendo camada de apresentação, aplicação, domínio e infraestrutura. A construção será dirigida por testes, usando a prática de análise de código TDD;

l) testes funcionais: fase que será liberada para o usuário realizar testes que garantam o atendimento do que o usuário espera.

Comentado [MCL101]: Aplicar o estilo correto

Comentado [MCL102]: Não falou nada sobre isso antes. Qual o sentido aqui?

Comentado [MCL103]: Requisito e caso de uso não são exatamente a mesma coisa.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2022									
	fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
pesquisa bibliográfica										
elicitação dos requisitos										
projeto de software										
prova de conceito (POC)										
construção/implementação										
testes										

Fonte: elaborado pelo autor.

8 REVISÃO BIBLIOGRÁFICA

Este capítulo descreve de forma breve o assunto que fundamentará o estudo a ser realizado para a construção deste trabalho: código com baixa coesão, alto acoplamento, arquitetura monolítica, nível de testes de integração.

Teste consiste na execução de um programa com um conjunto finito de casos, com o objetivo de verificar se ele possui o comportamento esperado. Existem diversos tipos de testes, como testes de unidade (quando se testa uma pequena unidade do código de forma isolada, como uma classe), testes de integração (quando se testa uma unidade com maior granularidade, como um conjunto de classes se comunicando), testes de performance (quando se submete o sistema a uma carga de processamento para verificar seu desempenho) (VALENTE, 2019).

Comentado [MCL104]: Aqui vc fez o que eu pedi lá na introdução. Faça isso e diga que vai trabalhar com teste de integração. Na fundamentação, detalhe o que é e como funciona esse tipo de teste.

A garantia da qualidade de software é atualmente reconhecida como um assunto que permeia todo o processo de desenvolvimento, os testes e a verificação dos problemas propriamente ditos são tópicos de grande importância no processo de qualidade. Já existe uma discussão técnica para verificar a correção de programas de forma matematicamente rigorosa, mas a conclusão é que a maioria dos sistemas é verificada por meio de testes, infelizmente, tais testes são, na melhor das hipóteses, inexatos. Não é possível garantir que uma unidade de software esteja correta por meio de testes, a menos que fôssemos capazes de executar testes que exaurissem todos os cenários possíveis, sabemos que humanamente isto é muito difícil. Mesmo programas simples podem existir bilhões de caminhos diferentes a serem percorridos. Então testar todos os caminhos possíveis dentro de um programa complexo é uma tarefa impossível (BROOKSHEAR, 2013).

Comentado [MCL105]: Melhorar redação dessa frase.

Comentado [MCL106]: Usar impessoal

Comentado [MCL107]: Precisa ampliar bastante essa fundamentação. O que escreveu até aqui não ajuda nada a compreender o que será feito no trabalho.

REFERÊNCIAS

BARTIÉ, Alexandre. **Garantia da qualidade de software**. Elsevier, 2002.

BROOKSHEAR, Glenn J. **Ciência da Computação:** Uma visão abrangente. 11. ed. Porto Alegre: Bookman, 2013. 573 p. Tradução Eduardo Kessler Piveta.

FANTINATO, Marcelo. AutoTest – **Um framework reutilizável para a automação de testes funcionais de software.** [2019] Disponível em: <https://www.researchgate.net/profile/Marcelo_Fantinato/publication/229004366_AutoTest-Um_Framework_Reutilizavel_para_a_Automacao_de_Teste_Funcional_de_Software/links/00b7d525a17636e087000000/AutoTest-Um-Framework-Reutilizavel-para-a-Automacao-de-Teste-Funcional-de-Software.pdf>. Acesso em 17 set. 2021.

FERNANDES, Matheus.; FONSECA, Samuel. **Automação de testes de software:** Estudo de caso da empresa softplan. [2019].
Disponível em: <https://www.riuni.unisul.br/bitstream/handle/12345/10127/AUTOMA%c3%87%c3%83O%20DE%20TESTES%20DE%20SOFTWARE-ESTUDO%20DE%20CASO%20DA%20EMPRESA%20SOFTPLAN-TCC_MATHEUS%20FERNANDES-SAMUEL%20TOMKELSKI%20FONSECA.pdf?sequence=1&isAllowed=y> Acesso em 14 set. 2021.

LIMA, G. M. P. S.; Travassos, G. H. **Testes de Integração Aplicados a Software Orientados a Objetos:** Heurísticas para Ordenação de Classes. Relatório Técnico ES632/04, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 2004.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 1995.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 2011.

TOMELIN, Marcio. **Testes de software a partir da ferramenta visual test.** [2001]
Disponível em: < http://campeche.inf.furb.br/tccs/BCC/2001-I/2001-1marciotomelinvf.pdf> Acesso em 28 set. 2021.

VALENTE, Marco. **Engenharia de software moderna.** [2019]
Disponível em: <https://docplayer.com.br/178875236-Engenharia-de-software-moderna.html>
Acesso em 14 set. 2021.

Comentado [MCL108]: Referência incompleta

Comentado [MCL109]: Evite fazer referências do researchgate pois ele é apenas um indexador. A publicação original a que vc se refere é de 2004 e está aqui: https://sol.sbc.org.br/index.php/sbqs/article/view/16195. Ou seja, é um correlato de quase 20 anos atrás!

Comentado [MCL110]: Este símbolo de < e > caiu na ABNT. Retirar de todos.

Comentado [MCL111]: Eu não consegui localizar este arquivo pelo link. De qquer modo é um TCC e a ABNT define exatamente como fazer a referência a um TCC. Você sequer colocou o ano correto.

Comentado [MCL112]: ABNT não tem esse modelo. Em que se baseou?

Comentado [MCL113]: Se é o mesmo livro, devem ser edições diferentes. Diferenciar. Também não entendo pq pegar duas edições com tanta diferença no tempo.

Comentado [MCL114]: É um TCC. A ABNT define como referenciar.
Também é um correlato de 20 anos atrás!

Comentado [MCL115]: Será que não consegue substituir essa referência por algo cuja procedência não seja questionável?

FORMULÁRIO DE AVALIAÇÃO BCC – PROFESSOR TCC I

Avaliador(a): **Maurício Capobianco Lopes**

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	9. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?		x	
	O problema está claramente formulado?		x	
	10. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?		x	
	Os objetivos específicos são coerentes com o objetivo principal?		x	
	11. JUSTIFICATIVA São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?		x	
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?		x	
	12. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?		x	
	Os métodos, recursos e o cronograma estão devidamente apresentados?	x		
	13. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?		x	
ASPECTOS METODOLÓGICOS	14. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?		x	
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?		x	
	15. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?		x	
	16. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?		x	
	17. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?			x
	As citações obedecem às normas da ABNT?		x	
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?	x		

Comentado [MCL116]: Melhorar a contextualização

Comentado [MCL117]: O que está contextualizado não leva extamente ao problema.

Comentado [MCL118]: Simplificar

Comentado [MCL119]: Repensar

Comentado [MCL120]: Melhorar argumentação, mesmo pq até aquele ponto não se tem ideia do que será o trabalho.

Comentado [MCL121]: Idem

Comentado [MCL122]: Melhorar descrição

Comentado [MCL123]: Muito superficiais

Comentado [MCL124]: Vários erros.

Comentado [MCL125]: Pontos a serem corrigidos.

Comentado [MCL126]: Usar os estilos corretos.

Comentado [MCL127]: Idem

Comentado [MCL128]: Norma é para ser seguida. Referência não é para ser feita de qualquer maneira segundo o que é mais simples.

Comentado [MCL129]: Há erros em autores e datas.

8.1.1

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DISCIPLINA: TRABALHO DE CONCLUSÃO DE CURSO I
CURSO: CIÊNCIA DA COMPUTAÇÃO - BCC

ATA DA DEFESA: BANCA DO PRÉ-PROJETO

Venho, por meio deste, manifestar minha avaliação sobre a **apresentação** do Pré-Projeto de TCC realizado pelo(a) acadêmico(a), Jonathan Luiz de Lara no **SEGUNDO SEMESTRE DE 2021**, com o título FERRAMENTA PARA TESTES INTEGRADOS EM JAVA.

A referida apresentação obteve a seguinte nota:

Componente da Banca	**Nota (de 0 a 10)**
Professor(a) Orientador(a): Dalton Solano dos Reis	**8,0**

A apresentação aconteceu em 28/10/2021 na sala de reunião virtual do MS-Teams, tendo início às 11:33 hs e foi encerrada às 11:52 hs.

ATENÇÃO. A nota acima se refere somente a apresentação do pré-projeto e vai ser repassada para o aluno (orientando). Favor preencher os campos acima e enviar por e-mail ao professor de TCC1 (dalton@furb.br). Lembro que os arquivos com as anotações das revisões do professor de TCC1 e Avaliador serão enviados para o orientando e professor orientador após o professor de TCC1 receber esta ata preenchida. Caso julgue necessário fazer mais alguma consideração relacionada ao pré-projeto ou a defesa, favor usar o espaço abaixo.

Observações da apresentação: