Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Colegiado de Sistemas de Informação

Uma Ferramenta de Apoio ao Teste de Software no Suporte ao Ensino de Programação de Computadores

Marcus Vinícius Nunes Calisto

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Euler Horta Marinho

Agosto, 2016
João Monlevade/MG

Marcus Vinícius Nunes Calisto

Uma Ferramenta de Apoio ao Teste de Software no Suporte ao Ensino de Programação de Computadores

Orientador: Euler Horta Marinho

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Universidade Federal de Ouro Preto

João Monlevade

Agosto de 2016

Marcus Vinícius Nunes Calisto
Uma Ferramenta de Apoio ao Teste de Software no Suporte ao Ensino de Programação de Computadores/ Marcus Vinícius Nunes Calisto. – João Monlevade, 11 de agosto de 2016-
87 p. : il. (algumas color.) ; 30 cm.

Orientador: Euler Horta Marinho

Monografia (graduação) – Universidade Federal de Ouro Preto, 11 de agosto de 2016.
1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

ATA DE DEFESA

Aos 11 dias do mês de agosto de 2016, às 09 horas e 00 minutos, na sala C203 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Marcus Vinicius Nunes Calisto,** sendo a Comissão Examinadora constituída pelos professores: Prof. Me. Euler Horta Marinho, Prof. Dr. Diego Zuquim Guimarães Garcia e Prof. Dr. Fernando Bernardes de Oliveira.

O candidato apresentou a monografia intitulada: *"Uma Ferramenta de Apoio ao Teste de Software no Suporte ao Ensino de Programação de Computadores"*. A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 9,0 (Nove Vírgula Zero), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

João Monlevade, 11 de Agosto de 2016.

Prof. Me. Euler Horta Marinho
Professor Orientador/Presidente

Prof. Dr. Diego Zuquim Guimarães Garcia
Professor Convidado

Prof. Dr. Fernando Bernardes de Oliveira
Professor Convidado

Marcus Vinicius Nunes Calisto
Graduando

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Meus agradecimentos aos amigos Hugo Leocádio Mansur, Adriano Romero Francisco, Arthur Gervásio Galvão, João Pedro Santos de Moura, Oto Braz Assunção, companheiros de trabalho e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida com certeza.

Obrigado meus irmãos Talles Nunes Calisto e Gisele Nunes Calisto, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente.

À Raquel Nayara Cota Silva, pessoa com quem amo partilhar a vida. Com você tenho me sentido mais vivo de verdade. Obrigado pelo carinho, a paciência e por sua capacidade de me trazer paz.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Aos meus amigos, pelas alegrias, tristezas e dores compartilhadas.
Com vocês, as pausas entre um parágrafo e outro
de produção melhora tudo o que tenho produzido na vida.

Resumo

O ensino de programação de computadores se torna complicado para principiantes, devido, em parte, ao fato que existem muitas linguagens de programação (JAVA, C, Python, PHP etc) e diferentes paradigmas. Dessa maneira, a definição de uma linguagem para ser tomada como ponto de partida nesse processo pode se tornar difícil. Alunos apresentam diversas dificuldades ao longo da aprendizagem de uma nova linguagem, principalmente nos primeiros passos para assimilação da lógica de programação. Por outro lado, nem sempre são utilizadas ferramentas de apoio ao desenvolvimento adequadas, o que pode levar a um desinteresse para o planejamento e execução do teste de software. Os professores estão diretamente ligados a esses problemas, pois é necessário saber como conduzir o ensino de modo a atender as necessidades de todos os alunos. Então, é necessário que professores recebam retorno com o propósito de analisar os pontos fracos e fortes do grupo de alunos. Este trabalho tem como objetivo a criação de uma aplicação Web denominada **TESTAÍ**, que permita ao professor realizar os testes de programas em Java de uma maneira simplificada, porém completa em atividades propostas aos alunos. A utilização da ferramenta de testes EvoSuite permite a integração de testes de software na correção de trabalhos de programação de computadores.

Palavras-chave: Engenharia de Software, Teste de Software, Aplicações Web, Programação de Computadores.

Abstract

The teaching of computer programming becomes complicated for beginners due in part to the fact that there are several programming languages (JAVA, Python, C, PHP etc) and paradigms. Thus, the choice of the initial language can be difficult. Students have many difficulties throughout the learning of a new language, mainly during the first steps for assimilating programming logic. In the other hand, the most appropriate development tools are not always used, which can lead to a disinterest for the planning and execution of the software testing. The professors are directly linked to these problems because there is a need to know how to conduct the teaching in order to satisfy the needs of all students. So, professors must receive feedback for the purpose of analyze the weak and strong points of the class. This work has as goal the development of a Web Application named **TESTAÍ** that allows users to execute tests of Java programs in a simplified and at the same time comprehensive manner. The use of the EvoSuite tool enables the integration of software testing in the correction of practical assignments.

Keywords: Software Engineering, Software Testing, Web Applications, Computer Programming.

Lista de ilustrações

Lista de abreviaturas e siglas

TI	Tecnologia da Informação
SBST	Search-Based Software Testing
AG	Algoritmo Genético
MVC	Model-View-Controller

Sumário

Introdução

A Engenharia de Software é uma das áreas da Tecnologia de Informação (TI) que possivelmente tem maior abrangência, visto que pode ser aplicada em diversas áreas, se não todas, em processos de TI e também possui grande impacto na realização de um projeto.

Segundo Pressman (2010), a Engenharia de Software pode ser definida como um processo que visa aplicar técnicas de engenharia ao desenvolvimento de software de alta qualidade com baixo custo.

A partir disso, a Engenharia de Software torna-se essencial tanto para o desenvolvimento de software, quanto para a busca da qualidade do produto final. Uma das áreas de maior importância da Engenharia de Software é a que se refere aos testes.

Segundo Myers (2004) o teste de software possui algumas diretrizes, tais como:

- Testar é um processo que envolve a busca por erros em um programa;
- É considerado um bom conjunto de testes aquele que tem a maior chance de descobrir um erro;
- Um teste bem sucedido é aquele que encontra erros.

Essas regras vão de encontro à definição antes empregada de que um teste bem sucedido seria um teste que não encontrasse erros. O objetivo da atividade de teste é criar testes que cubram todo o sistema, de maneira a revelar todas as possíveis classes de erros.

A Programação de Computadores é uma disciplina fundamental para diversas áreas. Porém, existem diversos fatores que podem dificultar o seu ensino, como por exemplo, o ritmo de aprendizagem de cada aluno e o modo de condução das atividades envolvidas Rocha et al. (2010). Diversos estudos investigam as razões envolvendo os altos índices de evasão em disciplinas de programação, principalmente as dificuldades encontradas pelos alunos no início do aprendizado para assimilação dos conceitos mais fundamentais necessários para o progresso em conceitos mais avançados Rocha et al. (2010).

Trabalhos práticos são um tipo de avaliação normalmente existente em disciplinas de programação. Entretanto, o professor pode ter dificuldades na sua correção e também na identificação dos conceitos nos quais os alunos apresentam maior nível de dificuldade, por exemplo, em razão da quantidade de alunos em sala Rocha et al. (2010). Nesse sentido, o desenvolvimento de ferramentas que auxiliem o professor na correção de trabalhos práticos e a consequente orientação dos alunos é de grande importância.

Este trabalho está organizado como segue. A introdução apresenta os objetivos deste trabalho. No Capítulo 1, serão apresentados os conceitos relevantes para o entendimento deste trabalho e conceitos básicos em relação ao teste de software. Ainda no Capítulo 1, a ferramenta *EvoSuite*, utilizada de maneira integrada à ferramenta desenvolvida, é detalhada. No Capítulo 2, serão abordados os fatores que levam ao desenvolvimento e estudo de novas técnicas e ferramentas para o auxilio do ensino à programação e também os trabalhos relacionados a este assunto que serviram de incentivo para a criação deste trabalho. No Capítulo 3, a ferramenta **TESTAÍ** será detalhada. No Capítulo 4, será apresentada a validação da ferramenta por meio de testes. E, por fim, o Capítulo 5 apresentará a conclusão obtida após a realização deste trabalho.

Este trabalho tem os objetivos geral e específicos detalhados a seguir.

0.1 Objetivo geral

Desenvolvimento de uma ferramenta para suporte ao ensino da programação de computadores na linguagem JAVA. Ou seja, uma ferramenta que de auxilio tanto a professores quanto à alunos para a melhoria do aprendizado.

0.2 Objetivos específicos

- Desenvolvimento de uma aplicação Web denominada**TESTAÍ** para amplo acesso de alunos e professores. A interface gráfica deverá ser simples e intuitiva.
- Mecanismo de submissão de código desenvolvido pelo aluno e o teste automatizado do mesmo.
- Disponibilização de mecanismo de *feedback* para a interação entre o professor e alunos.
- Controle do acesso a usuários previamente cadastrados.
- Validação da aplicação Web desenvolvida por meio de testes automatizados.

Parte I

Revisão Bibliográfica

1 Teste de Software

O teste de software é uma parte importante do processo de desenvolvimento, porém, ainda hoje, uma parte significativa do tempo do desenvolvimento é gasta nele, o que o torna muito custoso. Por esse motivo, muitos deixam de fazê-lo e apenas dedicam a essa atividade um quantia insuficiente de tempo.

Sendo assim, devido ao custo das atividades de teste, muitos programas podem apresentar ***defeitos***, que são passos, processos ou definições de dados incorretos. Defeitos são decorrentes de ***enganos*** causados por ações humanas. A existência desses defeitos no produto final ou em qualquer parte do processo, pode gerar um ***erro***, que é caracterizado por um estado inconsistente ou inesperado alcançado durante a execução. Por fim, esse estado, seja ele qual for, pode levar a uma ***falha***, ou seja, o resultado produzido é diferente do resultado esperado para aquela execução Delamaro (2007).

Segundo Myers (2004), hoje o processo de teste de software se tornou mais importante em um processo de desenvolvimento de software, e isso estimulou diversos estudos mais aprofundados na área.

Há diversas classificações para o teste. Como exemplo, há a classificação do teste em função do objetivo do teste Bernal e Hira (2014):

- Teste Funcional;
- Teste Não-Funcional;
- Teste Estrutural e
- Teste de Regressão.

Segundo Bernal e Hira (2014), o Teste Funcional é planejado de acordo com a especificação de requisitos. São testes de caixa-preta, no qual é considerado o comportamento externo do software.

O Teste Não-Funcional aborda diferentes requisitos não-funcionais. Inclui o Teste de Carga, Teste de *Stress*, Teste de Desempenho, dentre outros.

Já o Teste Estrutural, é considerado um teste de caixa-branca, observando o comportamento interno do software. São projetados levando em consideração o código do programa.

O Teste de Regressão é aplicado na versão mais recente do software. Assim é possível verificar se há novos defeitos em partes já testadas.

1.1 Teste de Unidade

Segundo Myers (2004) o Teste de Unidade (ou Teste de Módulo) é o processo de testar subprogramas, sub-rotinas e procedimentos, de maneira individual. Ou seja, ao invés de testar o software como um todo, o foco está em pequenos blocos de códigos. Assim, se um erro for descoberto durante esse procedimento, é mais fácil identificar de onde ele vem.

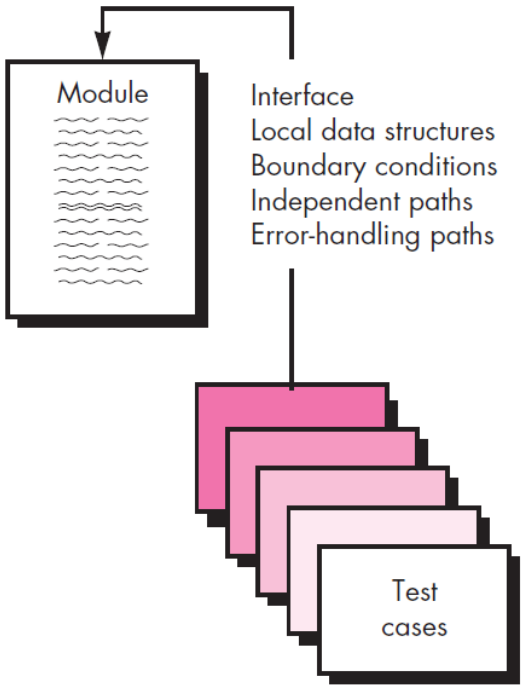

Figura 1 – Teste de Unidade Pressman (2010)

A figura acima ilustra a estrutura do Teste de Unidade. A interface é testada para garantir que a entrada e saída das informações fluam de maneira correta na parte do software em teste. A estrutura dos dados locais é examinada para ter certeza que os dados mantenham, temporariamente, sua integridade durante a execução do código. Todos os caminhos independentes por meio da estrutura de controle são exercitados para garantir que todos as instruções sejam executadas ao menos uma vez. As condições são testadas para garantir que os módulos atendam corretamente às condições impostas para limitar ou restringir o processo. Por fim, todas as as estruturas para tratamento de erros são testadas Pressman (2010).

Uma tarefa essencial durante o teste de unidade é a seleção dos caminhos de execução a serem seguidos. Casos de teste devem ser projetados para revelar diferentes tipos de erros tais como comparações incorretas, computação errada ou um controle de fluxo impróprio Pressman (2010).

Softwares tendem a falhar frequentemente quando, por exemplo, a n-ésima posição de um *array* com n-dimensões é acessada, quando um *loop* faz a sua n-ésima repetição,

ou mesmo quando os limites de valores, negativos ou positivos, são acessados, dentre outros casos. Por isso, testar os limites das variáveis e estruturas de dados é uma tarefa extremamente importante Pressman (2010).

1.1.1 Procedimentos de Teste de Unidade

Teste de Unidade é normalmente realizado junto com a codificação. Mas ele pode acontecer antes ou após a escrita do código da aplicação. Em geral, cada conjunto de testes deve ter incluído as saídas esperadas, para o veredito dos testes (passou/falhou) Pressman (2010).

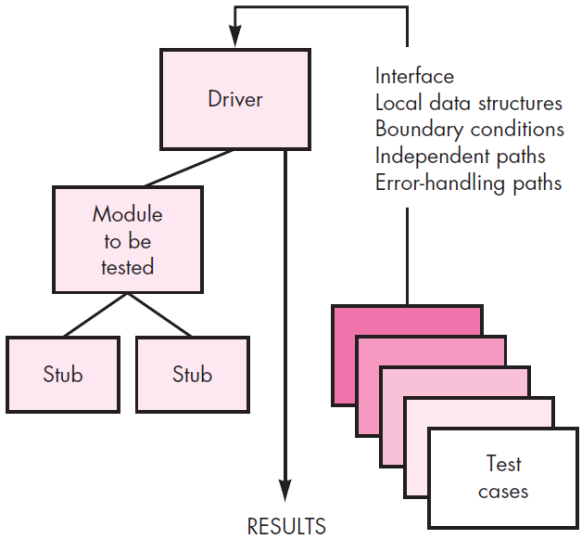

Figura 2 – Ambiente de Teste de Unidade Pressman (2010)

Testes de unidade podem exigir a implementação de *Drivers* e *Stubs*. O *Driver* é um componente que invoca o componente testado, passando-lhe os dados e recebendo a saída. Um *Stub* serve para substituir um método subordinado, quando este não estiver disponível. O *Stub* possui uma lógica simples, retornando a saída ao componente que o chamou. O desenvolvimento de *Drivers* e *Stubs* aumenta o tempo e o custo das atividades de teste, não acrescentando funcionalidades ao produto final Pressman (2010).

1.1.2 Critérios de Teste

São necessários dois tipos de informação para a criação de um teste de unidade: a especificação da unidade a ser testada e o código fonte desta unidade. Como dito anteriormente, o Teste de Unidade é do tipo caixa-branca. Entretanto, esse tipo de teste se torna menos praticável para se abordar grandes quantidades de código, sendo aplicado em métodos ou funções de menor porte.

É possível dividir os critérios de teste estrutural em três: baseados na complexidade, baseados em fluxos de controle e baseados em fluxo de dados.

1.1.2.1 Critérios baseados na complexidade

Os critérios baseados na complexidade utilizam informações do software em teste para derivar os requisitos de teste. Um critério bastante comum é o teste de caminho básico, ou critério de McCabe, que utiliza a complexidade ciclomática, que por sua vez, é uma métrica de software que permite analisar uma medida quantitativa da complexidade lógica do software Delamaro (2007).

1.1.2.2 Critérios baseados em fluxo de controle

Utilizam apenas características de controle de execução do programa, como desvios ou comandos, a fim de tentar determinar quais estruturas são realmente necessárias. Exemplos de comandos podem ser *ifs, whiles* e *fors* Delamaro (2007).

1.1.2.3 Critérios baseados em fluxo de dados

Os critérios baseados em fluxo de dados utilizam a análise do fluxo de dados para derivar os requisitos de teste. Eles têm como características básicas, a necessidade de testar as interações que envolvam definições de variáveis e subsequentes referências a elas (DELAMARO, 2007).

1.2 Teste de Mutação

Proposto por DeMillo, baseia-se na hipótese do *programador competente*, na qual se assume que os programadores com experiência escrevem códigos sempre corretos ou o mais próximo possível do correto e na ideia de *efeito de acoplamento*, na qual é sugerido que um conjunto de casos de testes para encontrar determinados erros, podem também encontrar erros de maior escala. Estudos envolvendo os critérios de mutação vem se desenvolvendo ao longo dos anos para que casos de testes sejam cada vez mais relevantes e que o tempo gasto seja cada vez menor para execução dos testes Delamaro (2007).

O Teste de Mutação ou Análise de Mutantes, como também é chamado, é um critério de teste baseado em defeitos. São utilizados defeitos comuns do processo de implementação de software nessa técnica, para que os requisitos de testes sejam determinados. No caso do Teste de Mutação, são criados vários outros programas derivados de um programa principal, com diversas modificações, sendo esses novos programas chamados de mutantes Delamaro (2007).

1.2.1 Processo de Análise de Mutação

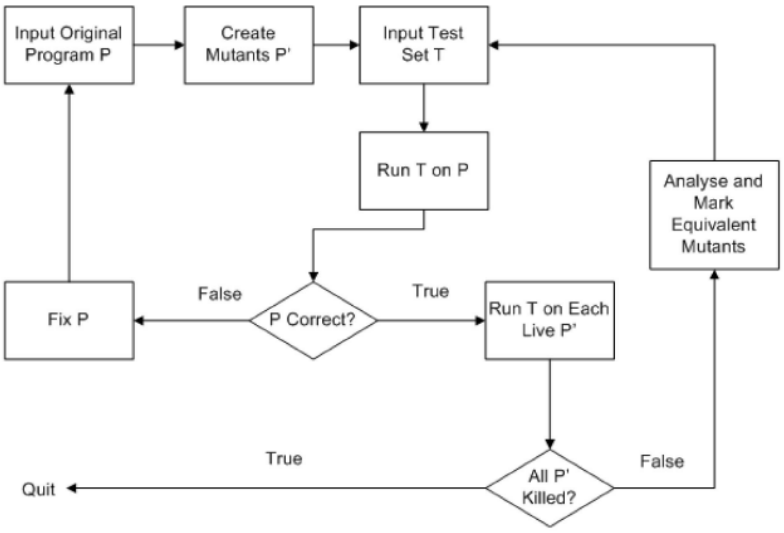

Figura 3 – Processo geral de análise de mutação Jia e Harman (2011)

Como mostrado no fluxograma da Figura 3, o processo de análise de mutantes cria, a partir de um programa original **P**, um conjunto de outros programas incorretos **P'**, denominados mutantes, contendo algumas mudanças sintáticas em relação ao original. Esses programas incorretos são programas com alterações sintáticas em relação ao programa original Jia e Harman (2011).

Uma vez escolhido um conjunto de testes para o programa em teste **P**, este deve ser executado em **P**, e a saída obtida é comparada à saída esperada. Caso sejam descobertos erros nesse processo, **P** deve ser corrigido. Por outro lado, se esse conjunto de testes estiver correto, ou seja, não houver retorno de nenhum erro quando testado em **P**, esse conjunto passa a ser executado em cada um dos mutantes, **P'**.

Após a execução dos testes em cada mutante **P'**, é realizada uma análise para filtragem dos mutantes. Caso algum erro tenha sido detectado, o mutante correspondente é considerado 'morto' e é eliminado do conjunto de mutantes. Caso não tenham sido identificados erros, deve-se determinar se os mutantes **P'** são equivalentes ao programa **P**, para retirá-los do conjunto original de mutantes. Ao final, calcula-se o escore de mutação, ou seja, a razão entre os mutantes mortos e o total de mutantes, excluindo-se os mutantes equivalentes. Quando todos os mutantes **P'** forem 'mortos' ou quando o *score* de mutação estiver dentro do desejado, o teste de mutação estará finalizado.

Como é possível ver na Figura 4, um exemplo simples de mudança em um programa derivado de P, seria a simples troca de operadores. Neste exemplo o operador $E(\&)$ é substituído por um operador OU $(||)$. Esse passo de transformação de um programa original para um mutante é denominado operador de mutação.

Program p	Mutant p'
...	...
if ($a > 0$ && $b > 0$)	if ($a > 0$ || $b > 0$)
return 1;	return 1;
...	...

Figura 4 – Exemplo de uma Operação de Mutação Jia e Harman (2011)

1.2.2 Desafios para o Teste de Mutação

Desde o primeiro estudo envolvendo o Teste de Mutação, diversos problemas foram identificados para a melhoria da técnica. Isso gerou novas ideias e métodos para incrementá-la.

Um dos problemas que impede o Teste de Mutação de ser uma das técnicas mais vantajosas a ser usada é o alto custo de execução computacional. Esse custo pode ser muito alto, levando em consideração o número de testes a serem feitos com um conjunto muito grande de casos de teste Jia e Harman (2011).

Outros problemas que podem ser levados em consideração são o problema do oráculo humano e o problema dos mutantes equivalentes. Ambos, devido ao enorme esforço feito por humanos para realizar cada ação. O problema do oráculo, se refere à ação de checar a saída do programa original a cada caso de teste. Já o problema dos mutantes equivalentes, é um problema indecidível, o que acarreta um maior esforço humano para a determinação de tais mutantes Jia e Harman (2011).

Há diversos estudos na área para o crescimento da eficiência do teste de mutação. Um outro problema é a quantidade de linguagens que existem e estão surgindo, e cada uma delas tem sua peculiaridade, sendo necessários ajustes para criação de casos de teste, execução e criação do conjunto de mutantes Jia e Harman (2011).

1.3 A Ferramenta EvoSuite

A ferramenta de testes utilizada para este trabalho é a EvoSuite. A EvoSuite é uma ferramenta que aperfeiçoa o processo de criação de casos de testes, buscando a maior cobertura possível de código.

EvoSuite utiliza técnicas evolucionárias que cobrem o máximo de possibilidades possíveis, baseadas em um critério geral. Otimiza a geração de casos de teste à medida em que o processo é executado. Assim, é possível fazer a cobertura dos testes em relação a um critério geral, o que é mais produtivo do que fazer os testes em relação a cobertura de critérios individuais. Outro ponto é que a EvoSuite utiliza Teste de Mutação para criar

conjuntos de testes menores possíveis, mas que garantam o número máximo de defeitos descobertos no programa em teste (FRASER; ARCURI, 2011).

1.3.1 Histórico

Desenvolvida por Gordon Fraser e Andrea Arcuri, a EvoSuite tem como objetivo simplificar e melhorar a técnica de teste de software. A ferramenta começou seu percurso de criação e implementação em 2010, a fim de participar de algumas provas de testes, que analisavam o desempenho.

A partir daí, como os resultados foram animadores, decidiram então fazer melhorias na ferramenta, dando a ela maiores opções de testes, como análise de cobertura, testes de mutação, regressão, entre outros parâmetros que foram adicionados durante o processo de aprimoramento da ferramenta.

De fato, os primeiros artigos e informações sobre a EvoSuite, foram em relação a algumas comparações entre ela e outras ferramentas de testes existentes, como eToc, Randoop e TestFul.

Apesar de todas elas utilizarem JUnit para criação de casos de testes, a diferença entre a EvoSuite e as demais citadas, exceto a Randoop, é a automação completa do processo teste. A EvoSuite está em constante processo de melhorias, enquanto algumas dessas ferramentas não sofrem atualizações há anos, e não utilizam teste de mutação para a geração de conjuntos mais enxutos de entradas Fraser e Arcuri (2011).

Ao longo dos anos, desde 2011, quando o primeiro artigo que detalhava a ferramenta foi publicado Fraser e Arcuri (2011), ela sofreu diversas mudanças, permitindo sua utilização em diferentes plataformas (Linux, Mac OS e Windows) e de maneiras diferentes (Linha de comando, *Plugin* para o Eclipse, através da Web). Desse modo, o uso seria ampliado, pois ficou mais simples executar os comandos necessários para realização de um teste.

Em 2013, em uma competição entre ferramentas de teste, *International Workshop on Search-Based Software Testing (SBST)*, a EvoSuite conseguiu o primeiro lugar com excelência, causando uma boa impressão aos criadores. Desde então, ela participa sempre de competições para que seu rendimento possa ser medido e as atualizações analisadas.

De acordo com o site da competição, *SBST Contest* SBST (2013), os critérios de pontuação em 2013 eram: tempo de execução, cobertura de código e ponto de mutação.

Abaixo o resultado, com as quatro primeiras ferramentas na competição de 2013:

- 1. evosuite com 156.9459 pontos de média;
- 2. randoop com 101.8129 pontos de média;
- 3. t2 com 50.4938 pontos de média;

- 4. dsc com 43.3208 pontos de média.

É possível analisar que a EvoSuite teve um desempenho muito superior às demais ferramentas. Portanto, a competição contribuiu e ainda contribui para o desenvolvimento da ferramenta. Em 2015, a ferramenta conseguiu o segundo lugar e em 2016 conseguiu o maior número de pontos de toda competição e atingiu o primeiro lugar novamente.

Em 2015, a EvoSuite teve seu código disponibilizado no GitHub, e atualmente é possível encontrar todas as publicações em relação a ferramenta em seu site[1], assim como a versão mais nova da ferramenta e tutoriais sobre execução e instalação. Todo desenvolvimento é suportado pela Google (Google Focused Research Award).

1.3.2 Execução

EvoSuite é disponibilizada como um arquivo executável com extensão *.jar* e pode ser executada com uma chamada por meio de linha de comando, *plugin* para Eclipse, Maven e Intellij Idea.

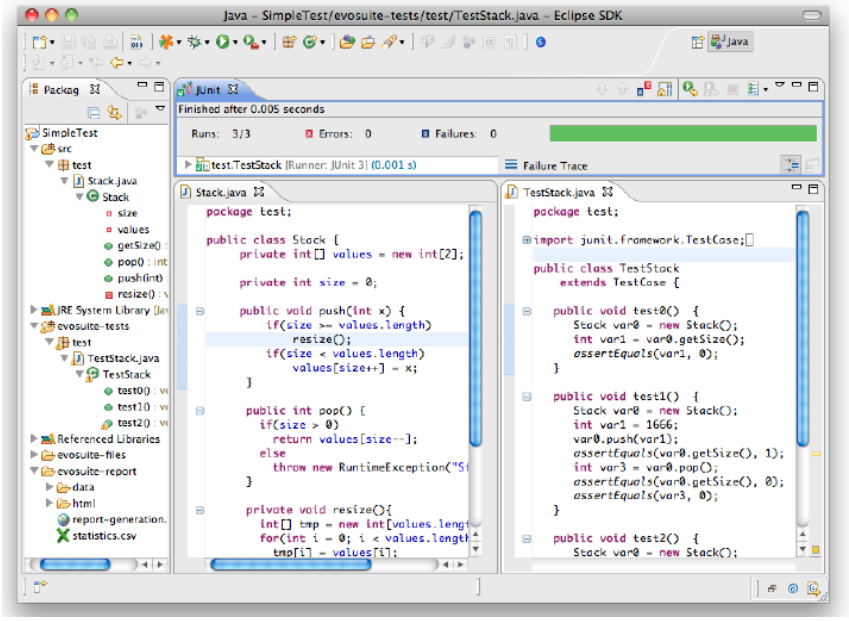

Figura 5 – Pluguin para o Eclipse Fraser e Arcuri (2011)

Neste trabalho, o método de chamada utilizado é via linha comando. Assim, é possível fazer a chamada da ferramenta a partir de uma implementação *PHP*, passando os parâmetros necessários para sua execução.

Se utilizado o Eclipse, basta fazer a instalação do *plugin*, seguindo as instruções que estão no site, e quando desejar, acionar a ferramenta pelo Eclipse.

[1] Disponível em: www.evosuite.org

Quando usada a linha de comando, o processo de execução da ferramenta segue algumas regras para que funcione corretamente. É necessário ter o JAVA instalado no computador e o acesso a ferramenta tem de ser feito direto do diretório onde ela se encontra, ou seja, o comando no terminal tem que ser de dentro do diretório onde a ferramenta está.

O comando básico para iniciar a ferramenta é o seguinte:

```
java -jar evosuite.jar <opções>
```

O campo de opções pode ser substituído por diversas ações como *-help, - target, -projectCP, listParameter*, dentre outros. Cada opção serve para um caminho de testes. Por padrão, a ferramenta usa uma configuração que permite ao usuário utilizar com o comando padrão e essa configuração é o suficiente para fazer toda a cobertura de código e retornar informações relevantes.

```
java -jar evosuite.jar -class <Nome da Classe> -projectCP <Caminho para a Classe>
```

Caso alguma dúvida apareça, o comando *-help* imprime todos os comandos existentes para a execução de um teste.

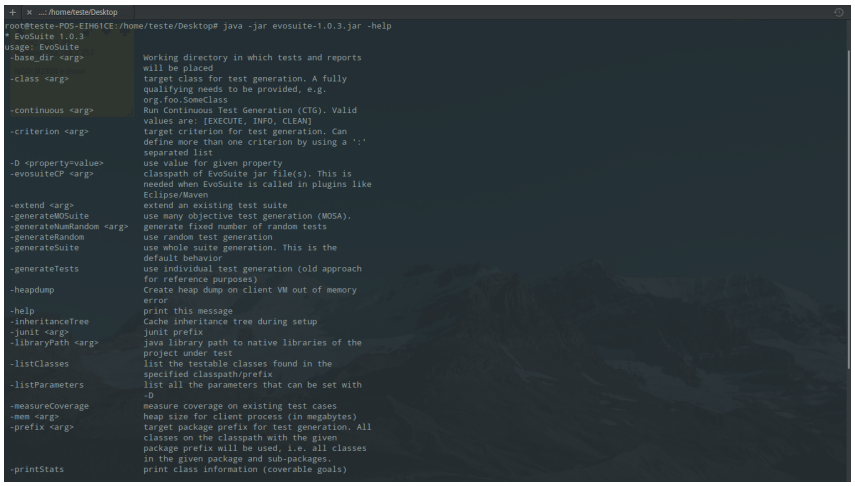

Figura 6 – Comando *-help*

1.3.3 Funcionamento

EvoSuite é uma ferramenta que gera casos de testes em JUnit automaticamente para uma determinada classe JAVA. É baseada na técnica *search-based test* e utiliza Algoritmo Genético (GA) para desenvolver uma população de casos de testes possíveis. O critério padrão de cobertura da EvoSuite é a cobertura de desvios, porém, há também suporte para testes de mutação e fluxo de dados, e outros critérios podem ser implementados baseados em uma função *fitness* Fraser e Arcuri (2013).

É necessário apenas o arquivo *.class*, ou seja o ***bytecode*** JAVA da classe em teste. O ***bytecode*** é analisado e instrumentado e ao final da busca a EvoSuite produz um conjunto de teste JUnit para a classe. É possível testar pacotes inteiros, e nesse modo, a ferramenta testa uma classe por vez, encontrada no local determinado. Para produzir instâncias das classes, a ferramenta considera todos os métodos e construtores possíveis que geram uma instância de um tipo, e recursivamente, tenta satisfazer todas as dependências.

2 O Suporte ao Ensino de Programação de Computadores

2.1 Juízes Online

Uma ferramenta encontrada para a execução de testes de software via aplicação web foi a Juízes Online. Juízes Online são sistemas que compilam, executam e testam códigos-fonte para julgar se estão corretos ou não. O processo de execução é, basicamente: existe um problema a ser resolvido, uma codificação é feita para aquele problema, essa codificação é submetida ao sistema do Juiz Online, ele executa o código, obtém as saídas, analisa de acordo com as saídas esperadas e retorna um parecer (certo, errado, erro de compilação ou erro de execução) CHAVES (2014).

Diversos sistemas de Juízes Online podem ser encontrados na Internet, como por exemplo o URI Juiz Online, SPOJ Brasil e o UVA Online Judge. A utilização destes sistemas de avaliação automática é muito comum em competições de programação, tanto no Brasil, quanto no mundo. Exemplos desses concursos são o TopCoder e o ACM *Collegiate Programming Contest International*. Juízes Online ainda são bons incentivadores aos alunos iniciantes em programação, pois estes podem assimilar novos conhecimentos, passando por diversos níveis de desafios CHAVES (2014).

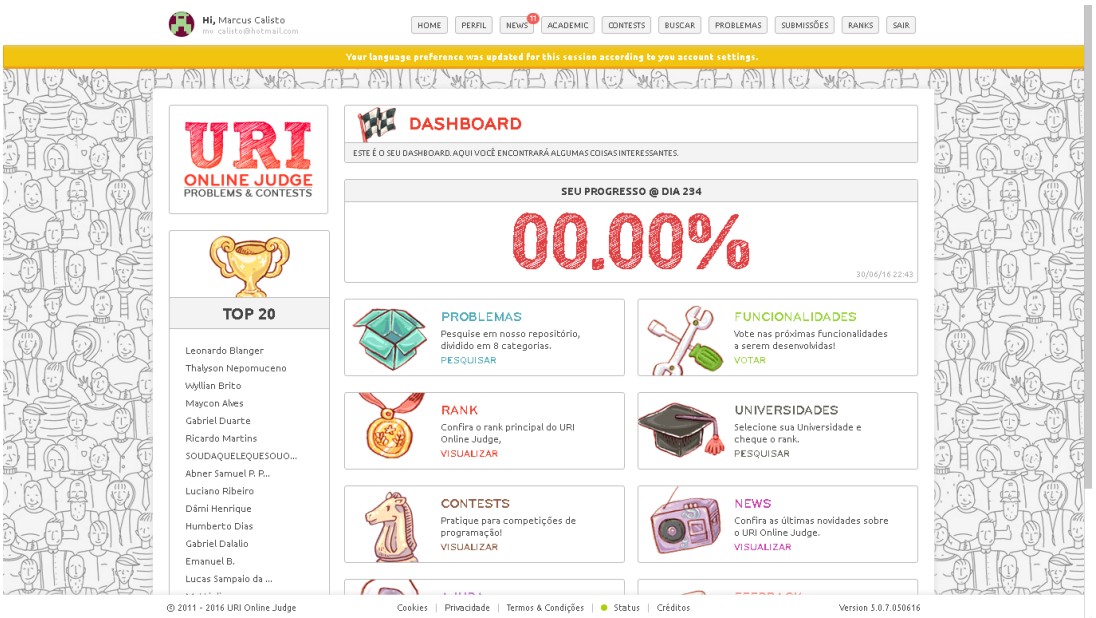

Figura 7 – URI Juiz Online

A figura acima mostra a página inicial do sistema de juiz online URI Online

Judge (URI Juiz Online)[2]. Ele conta com uma versão em Português, e suas principais características são:

- Interface amigável;
- Disponibilização de fóruns, *ranking* de classificação, tutoriais e materiais extras;
- Possui exercícios divididos por categorias e
- Suporte para diversas linguagens de programação (JAVA, C++, Python, Ruby) Chaves et al. (2013).

2.2 Trabalhos Relacionados ao Ensino

Estudos envolvendo a melhoria do ensino de programação foram encontrados durante o recolhimento de informações para a implementação desse trabalho. Os esforços para um ambiente simplificado e encorajador aos novos aspirantes a programadores é visível.

O Cafezinho é uma linguagem que tenta facilitar o entendimento do usuário com relação à programação. Ela foi projetada com base na Linguagem C, derivada de uma outra linguagem também criada com este propósito. O Cafezinho não tem todas as funcionalidades das linguagens C e Java, mas possui o básico para programadores principiantes LEBTAG (2014).

O MOJO é um Módulo que pode ser instalado na Plataforma *Moodle* e traz o auxilio de Juiz Online para o ensino. O MOJO faz a integração total das atividades com os Juízes Online envolvidos no projeto, facilitando o acesso dos usuários CHAVES (2014).

Como não foi encontrado nenhum outro trabalho semelhante ao **TESTAÍ**, que integre uma ferramenta de testes à uma aplicação web, a pesquisa para o desenvolvimento e as ideias criadas para cada passo de implementação foram importantes e bem calculadas para que o resultado fosse bem sucedido.

Embora estes trabalhos citados acima não integrem exatamente uma ferramenta à uma aplicação Web como é caso da **TESTAÍ**, que utiliza a EvoSuite, foi de extrema importância entender como é o processo de criação de um ambiente que priorize a simplicidade no ensino e manuseio. É importante que os alunos iniciantes em programação consigam obter um retorno concreto sobre suas dúvidas.

[2] Disponível em: www.urionlinejudge.com.br/judge/pt/

Parte II

Desenvolvimento e Resultados

3 Descrição da Ferramenta

A ferramenta **TESTAÍ** consiste em uma aplicação WEB, desenvolvida com o auxílio do *framework* Laravel[3], integrando a ferramenta EvoSuite[4], para realizar testes em classes JAVA. O *framework* Laravel se baseia em uma arquitetura *Model-View-Controller* (MVC).

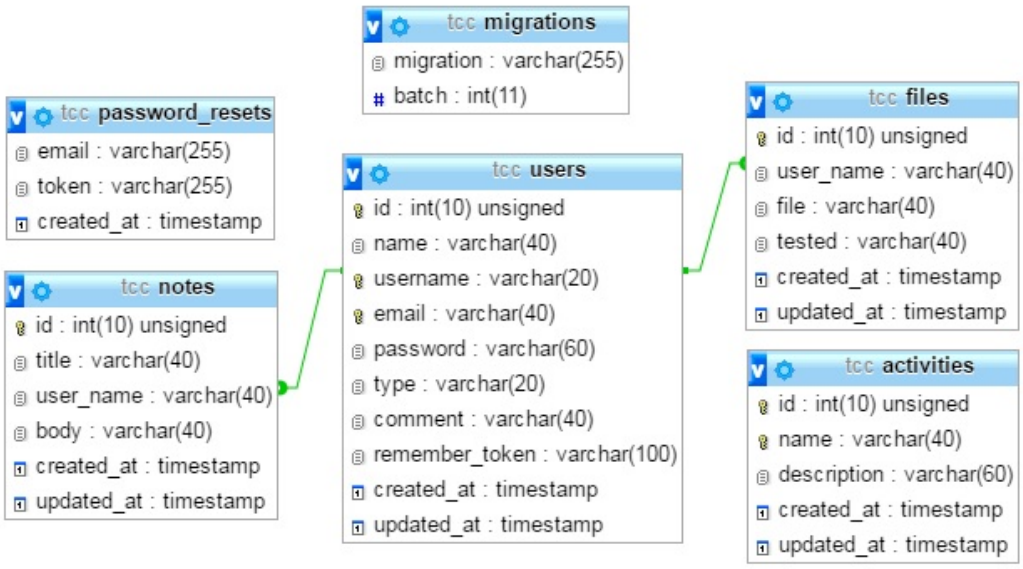

Figura 8 – Diagrama Entidade-Relacionamento

A Figura 8 apresenta o diagrama Entidade-Relacionamento da aplicação. Como pode ser observado este é composto de 6 entidades : *users*, *notas*, *files*, *atividades*, *password_resets*, *migrations*.

A entidade ***users***, a entidade principal, é composta de atributos que identificam o usuário como aluno ou professor/administrador. O tipo de usuário(*user*) determina qual área o usuário pode acessar e quais ações ele pode realizar. Desta maneira, é possível limitar o acesso a usuários previamente cadastrados e prover uma interface gráfica personalizada. Outras entidades dependem dela para preencher alguns atributos.

A entidade ***files*** é responsável por armazenar as informações necessárias do arquivo enviado pelo aluno. Ela contém o nome do arquivo, o nome do usuário que enviou o arquivo e a data que ele foi enviado, mantendo assim um controle de prazos. Além disso, um atributo chamado 'testado' fornece informação ao professor indicando se o arquivo em questão foi ou não testado e se passou ou falhou nos testes da EvoSuite.

[3] Disponível em: www.laravel.com/
[4] Disponível em: www.evosuite.org/

A entidade ***notes***, também chamada de *comentários* na maior parte do desenvolvimento, é utilizada para adicionar algum comentário para o aluno referente à alguma atividade feita por ele. Desta maneira, o professor pode direcionar para um aluno algo referente aos resultados dos testes, dando a ele uma orientação do que está errado, indicando ao aluno a fazer da maneira correta em uma próxima vez ou simplesmente fazer um comentário elogiando o desempenho.

Ambas entidades, ***notes*** e ***files***, recebem uma chave estrangeira da entidade **users** como mostrado no diagrama, representadas pela ligação de uma linha verde entre as classes. A relação entre elas é de um para muitos, ou seja, um usuário pode enviar diversos arquivos e ter diversos arquivos atribuídos ao seu perfil, assim como vários comentários podem ser atribuídos a ele pelo professor.

Por fim, a entidade ***activities*** é a responsável pela criação das atividades propostas pelo professor. Quando o professor cria o nome da atividade, automaticamente são salvas as informações no banco de dados e um diretório é criado no servidor para receber os arquivos enviados pelos alunos.

As entidades *password_reset* e *migrations* são padrões do *framework* Laravel[5] e podem ser exploradas para trabalhos futuros para melhoria e implementação de funções e também para a expansão da ferramenta.

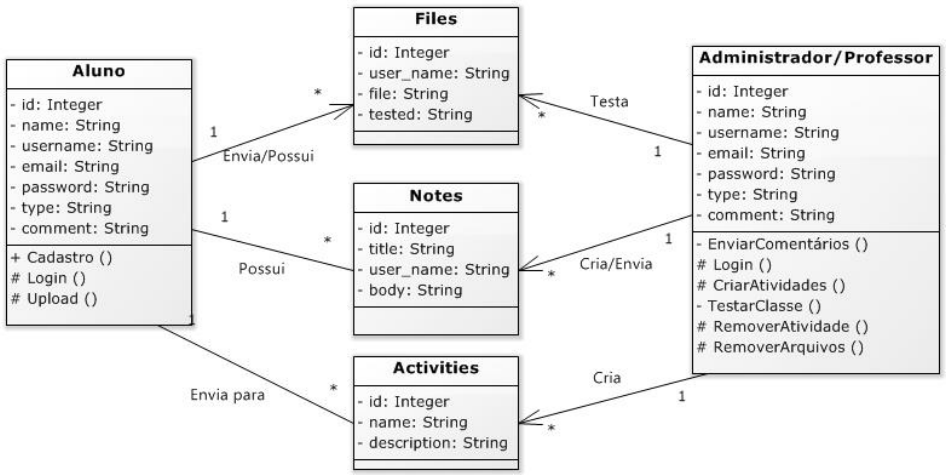

Figura 9 – Diagrama de Classes

A Figura 9 apresenta um diagrama com associações entre as classes da ferramenta **TESTAÍ**. As classes denominadas como **Aluno** e **Administrador/Professores** se referem aos tipos de usuários existentes no sistema, portanto fazem parte da classe ***user***. Mais detalhes são apresentados no apêndice onde é exibido o código fonte da aplicação.

[5] Disponível em : www.laravel.com

3.1 Funcionalidades

A ferramenta pretende oferecer aos usuários, alunos e professores, um ambiente intuitivo para realizar determinadas ações. Para isso, ela disponibiliza um conjunto de funcionalidades, de acordo com o diagrama apresentado na Figura 10.

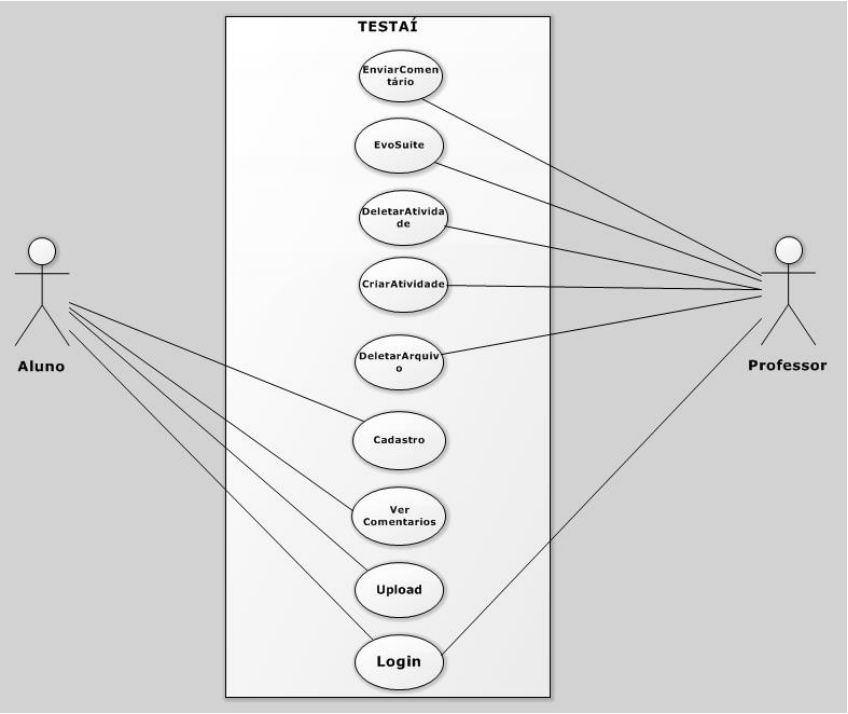

Figura 10 – Diagrama de Casos de Uso

A página inicial da ferramenta é ilustrada na 11. A partir dela o usuário pode escolher se deseja fazer o Login ou Cadastro. O Administrador tem permissão para entrar em qualquer uma das áreas, mas o aluno apenas na área do aluno.

Uma vez que o aluno faz o login, é apresentada a área do aluno onde pode ser feito o *upload* do arquivo *.class* relacionado à atividade proposta pelo professor. Ao fazer o *upload* do arquivo, é exibida uma tela onde aparece a confirmação de *upload*, o nome do arquivo e uma tabela exibindo o nome do aluno, o nome do arquivo e a data de envio.

Figura 11 – Página Inicial

A seguir, são discutidas as demais funcionalidades.

- **Cadastro:** área de cadastro para alunos, campos específicos utilizados para distinguir o tipo de usuário, sendo o acesso realizado de acordo com o tipo cadastrado. Alunos possuem tipo 2 e professores, que também são Administradores, possuem tipo 1. Os professores são cadastrados manualmente, pois é importante que a ferramenta permita Administradores para monitorarem suas atividades. O cadastro é composto por nome, nome de usuário (*username*), *email* e senha.
- **Área do Aluno:** uma vez cadastrado, o aluno tem acesso à Área do Aluno que permitirá que ele visualize os trabalhos propostos e as suas respectivas datas de entrega, além de poder fazer o envio de suas classes Java para o professor. Ele tem acesso aos comentários enviados pelos professor referentes à alguma atividade ou trabalho. As opções do aluno são detalhadas a seguir:
  - ***Upload*** : É necessário que o aluno envie o arquivo de **bytecode**, ou seja o arquivo *.class*, que será usado pelo professor para testes com a ferramenta EvoSuite. Quando o aluno entra na tela de *Upload*, ele tem que escolher um caminho para esse arquivo, que no caso será uma atividade criada pelo professor, e então selecionar o arquivo que deseja enviar e fazer o *upload*. Caso tudo ocorra de acordo, uma tela de 'sucesso' (Figura 12) é exibida e uma tabela listando os arquivos anteriores enviados pelo aluno também. O arquivo é salvo no diretório escolhido, mas o nome salvo no banco de dados desconsidera a extensão do arquivo. Exemplo: um arquivo teste *.class* sera salvo no banco de dados apenas como teste.

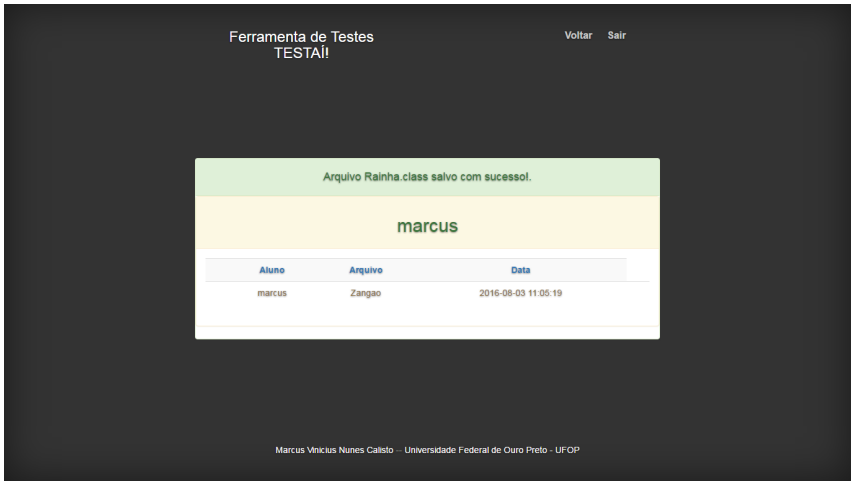

Figura 12 – *Upload* confirmado

– **Ver Comentários**: A cada atividade, o professor poderá enviar comentários aos alunos referentes aos resultados dos testes realizados. Nessa tela (Figura 13), serão exibidos os comentários referentes ao nome de usuário do aluno. Assim ele pode ver todos os comentários feitos pelo professor com data e referência à atividade.

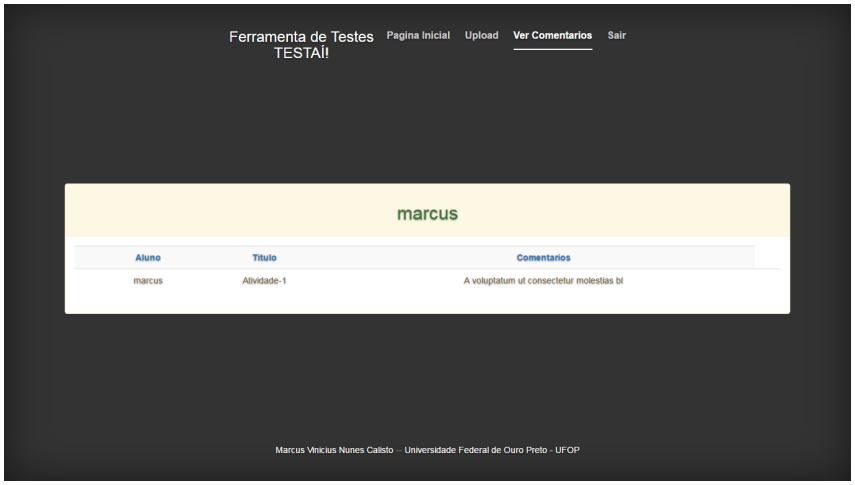

Figura 13 – Tela de comentários

- **Área do Professor:** a Área do Professor permite o acesso e visualização dos arquivos enviados pelos alunos. Dessa maneira, ele poderá acessar a ferramenta de testes EvoSuite (2016) para realizar os testes nas classes e analisar os resultados obtidos. O professor pode também enviar comentários aos alunos, criar as atividades e o diretório para onde serão enviados os arquivos dos alunos. Ele tem autonomia para excluir arquivos e atividades. As opções do professor são detalhadas a seguir:

  – **Painel de Controle:** Na tela direcionada após o login (Figura 14), o professor consegue visualizar as tabelas com o nome dos alunos, atividades criadas e

arquivos enviados. Caso ele queria adicionar uma nova atividade, basta clicar no botão 'Adicionar Atividade' e um formulário será exibido para ele.

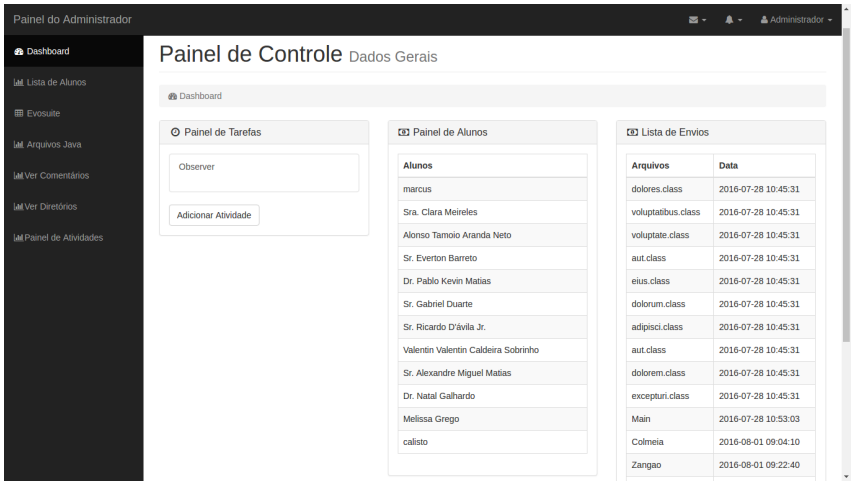

Figura 14 – Painel de Controle

– **Lista de Alunos**: A lista de alunos cadastrados aparece em uma tabela contendo todas as informações necessárias ao professor: nome, nome de usuário e *email*. É nessa tela (Figura 15) que o professor pode adicionar um comentário clicando no botão "Adicionar Nota".

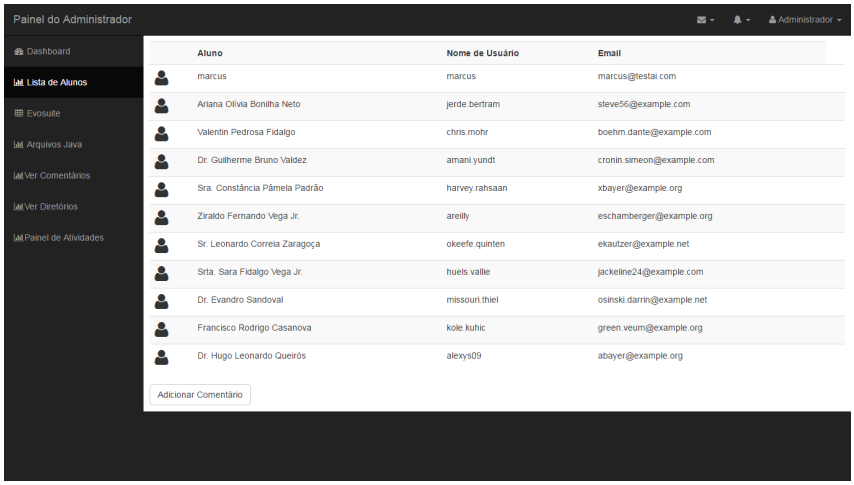

Figura 15 – Lista de Alunos

– **EvoSuite:** Dentre os objetivos da ferramenta, encontra-se a integração com a ferramenta EvoSuite (EVOSUITE, 2016) para realização dos testes de classes JAVA. Essa integração se dá por meio da execução de comandos em **PHP** que permitem a execução da EvoSuite em linha de comando. Nessa tela (Figura 16), é exibido um formulário onde o professor escolhe um arquivo referente a

um aluno e o submete a teste. O código por trás da execução da ferramenta gira em torno da função *shell_exec*, que executa uma chamada em linha de comando e retorna sua execução completa em uma variável.

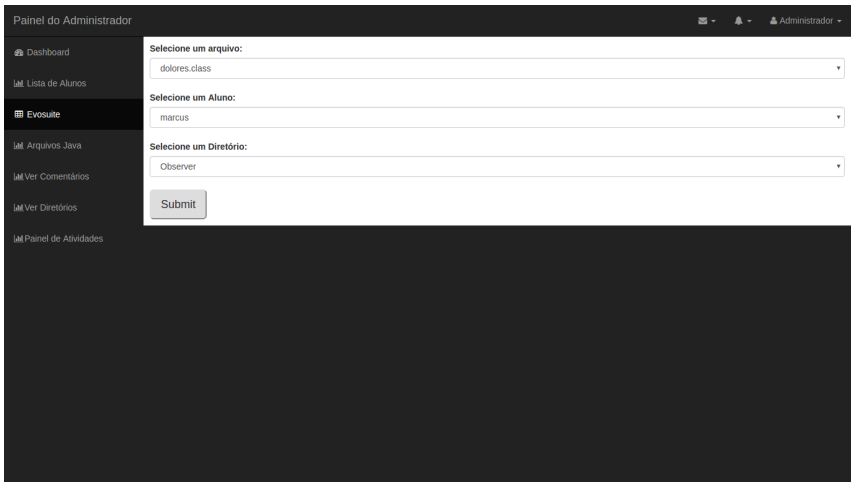

Figura 16 – Formulário de execução da ferramenta

– **Arquivos Java:** Todos os arquivos enviados pelos alunos podem ser vistos nessa tela (Figura 17). Uma tabela é exibida contendo o nome do usuário que enviou o arquivo, o nome do arquivo *.class* e a data de envio. O nome do arquivo é exibido sem a extensão.

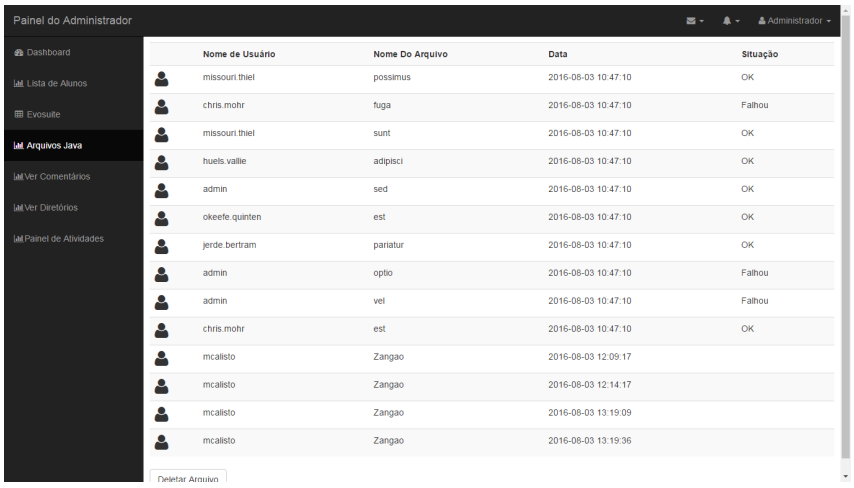

Figura 17 – Lista de arquivos Java enviados

– **Ver Comentários:** Cada comentário enviado para um aluno por um professor, é gravado no banco de dados e caso o professor deseje ver todos os comentários enviados, uma tela (Figura 18) apresenta uma tabela mostrando o nome do aluno, o título da atividade, o conteúdo da mensagem e a data de postagem daquele comentário.

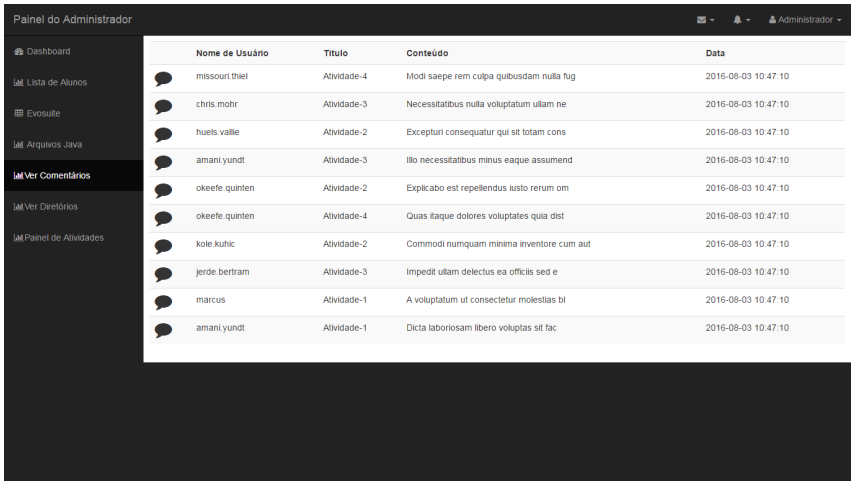

Figura 18 – Tabela de comentários

– **Painel de Atividades:** Nessa tela (Figura 19) estão listadas todas as atividades criadas pelo professor. Ele tem autonomia para removê-las e criá-las quando bem entender. A tabela exibe o nome da atividade, sua breve descrição e a data de criação.

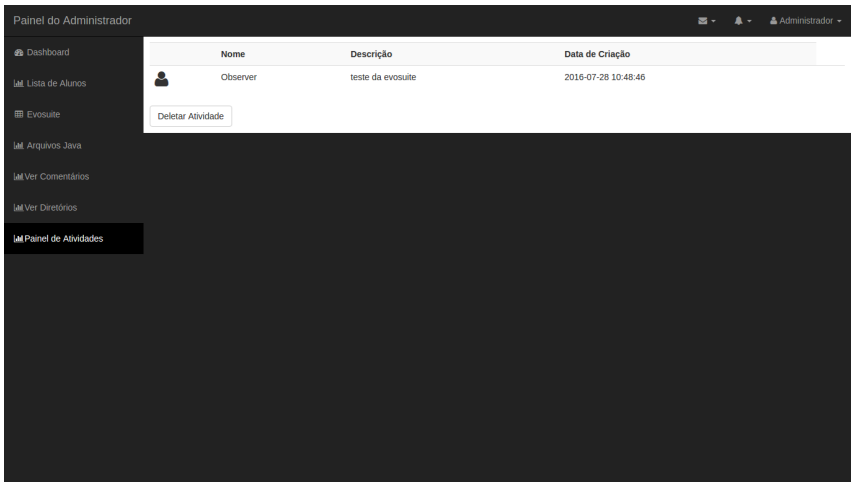

Figura 19 – Lista de Atividades

De maneira simples, porém eficiente, a ideia é que a **TESTAÍ** seja uma plataforma de auxílio ao ensino de programação a alunos iniciantes e que ajude no trabalho dos professores, principalmente na identificação dos pontos de maior dificuldade.

4 Validação da Ferramenta TESTAÍ

A validação foi realizada utilizando a ferramenta de testes Selenium IDE [6], executada como um complemento do navegador Mozilla Firefox[7]. Cada funcionalidade foi validada através de uma série de ações que deveriam chegar a um resultado esperado, como por exemplo o cadastro de um aluno ou o envio de um arquivo.

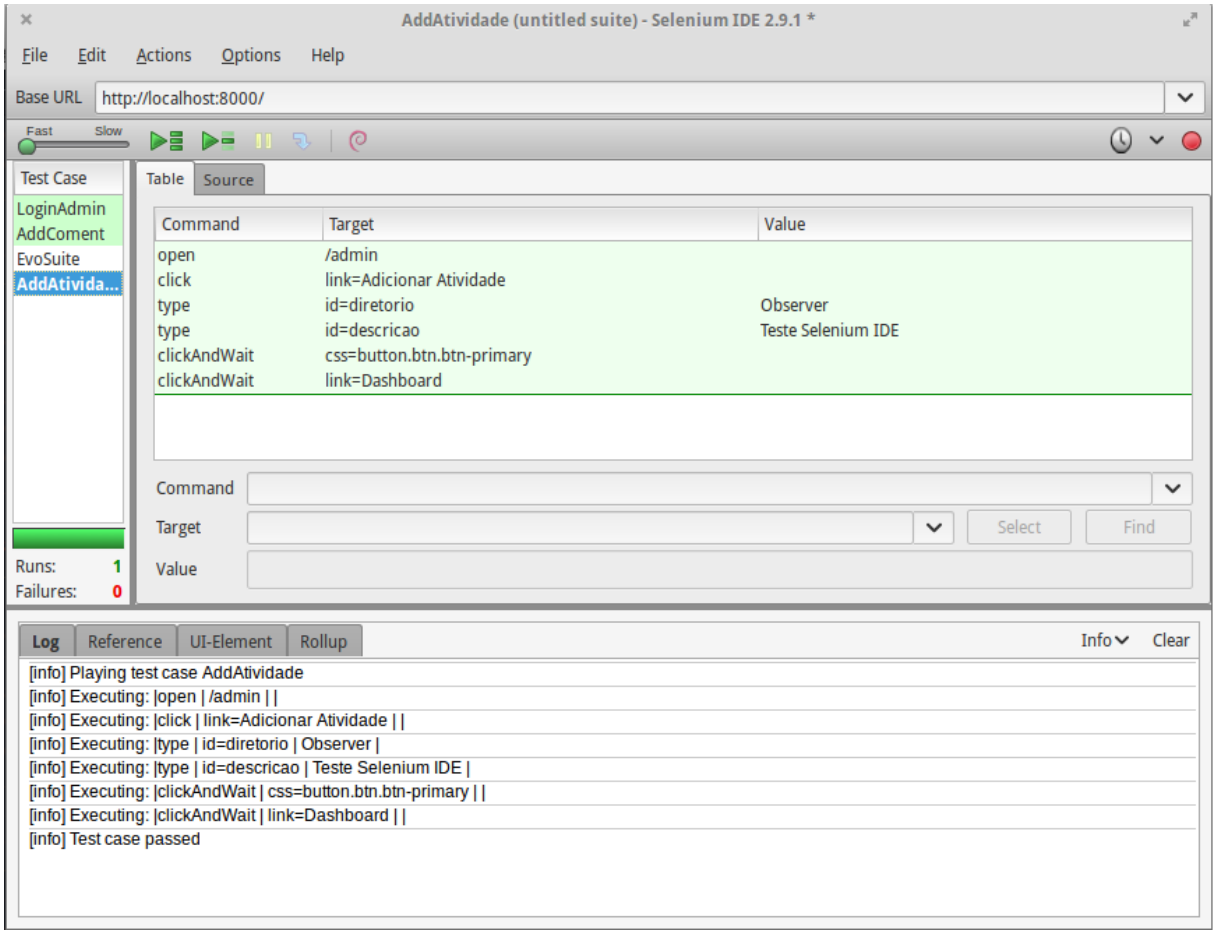

Figura 20 – Selenium IDE

A primeira funcionalidade validada foi o cadastro. A ação comum de um aluno iniciante, deveria ser, primeiramente, o cadastro, para depois ter acesso a área do aluno e poder enviar seus arquivos e verificar os comentários enviados pelo professor. A Figura 21 mostra o conjunto de ações para a realização de um cadastro.

6 Selenium... (2016)
7 (MOZILLA..., 2016)

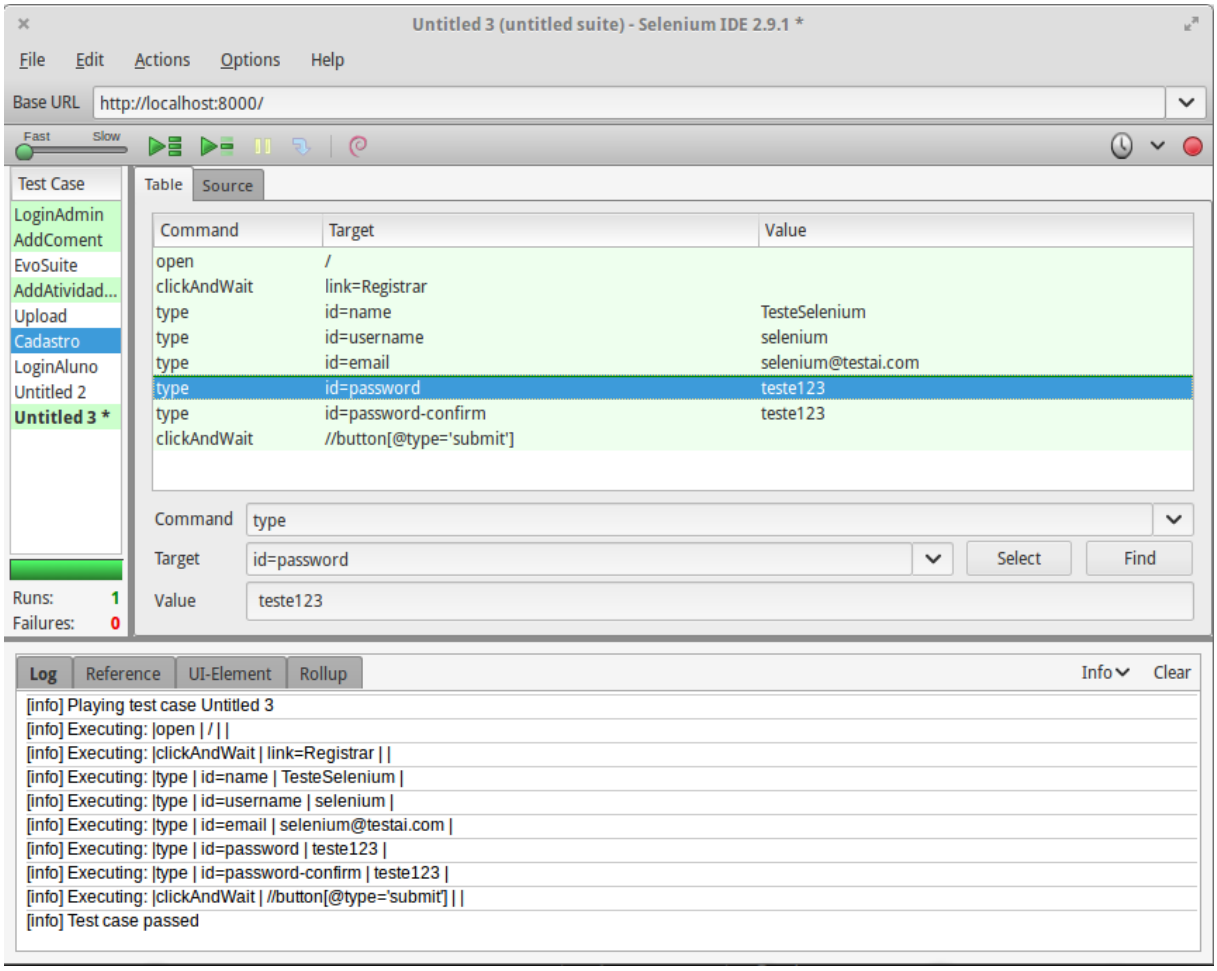

Figura 21 – Entradas e Ações para realização de um cadastro

Como mostrado na Figura 21, as entradas para o cadastro devem seguir as restrições necessárias definidas no sistema. Uma das restrições, por exemplo, é que a senha deve ter no mínimo seis dígitos.

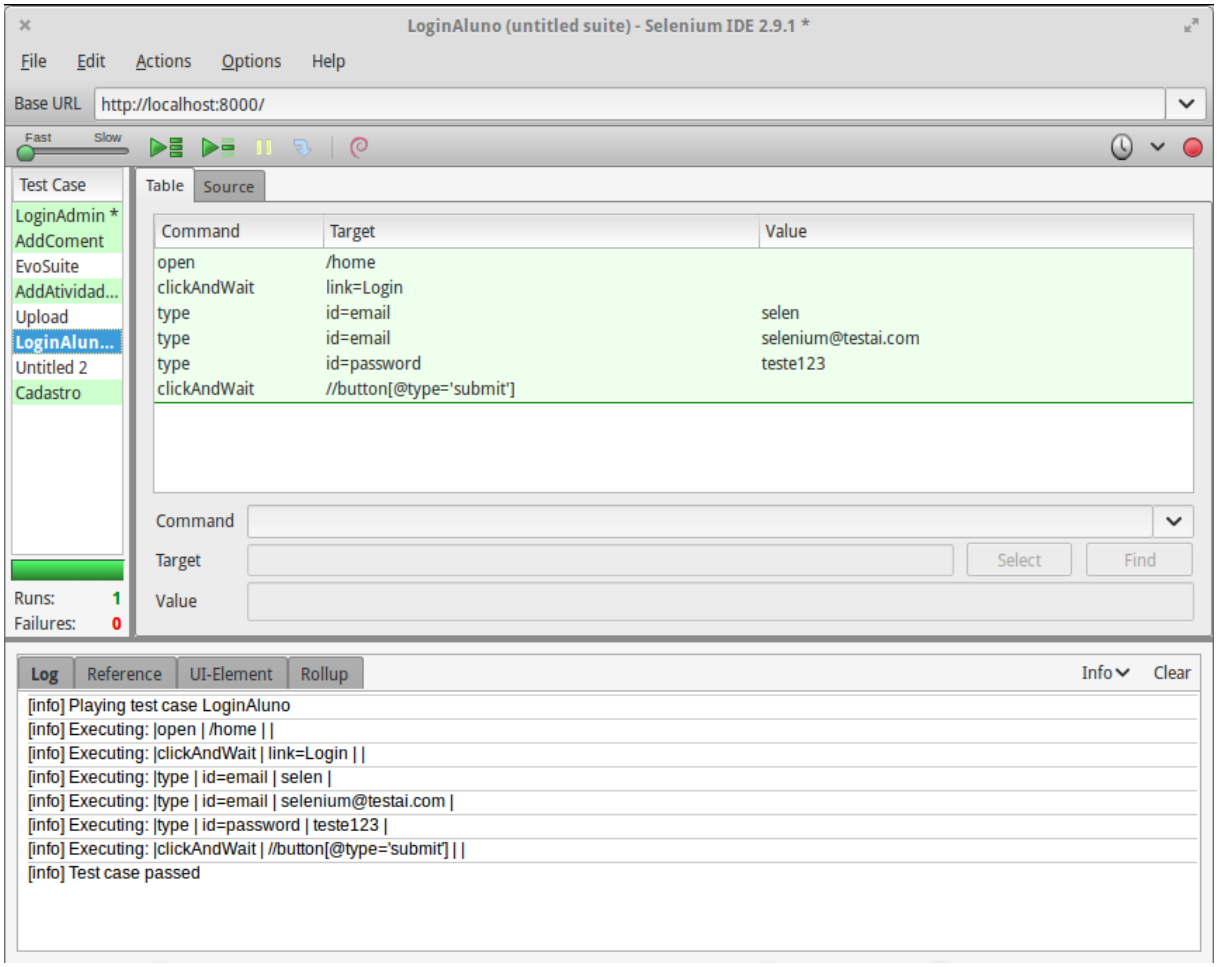

Figura 22 – Entradas e Ações para realização do login do aluno

Para verificar se o cadastro funcionou, basta conferir no banco de dados se os dados inseridos foram salvos com sucesso. Uma outra maneira de verificar é fazendo o login do aluno na ferramenta (Figura 22).

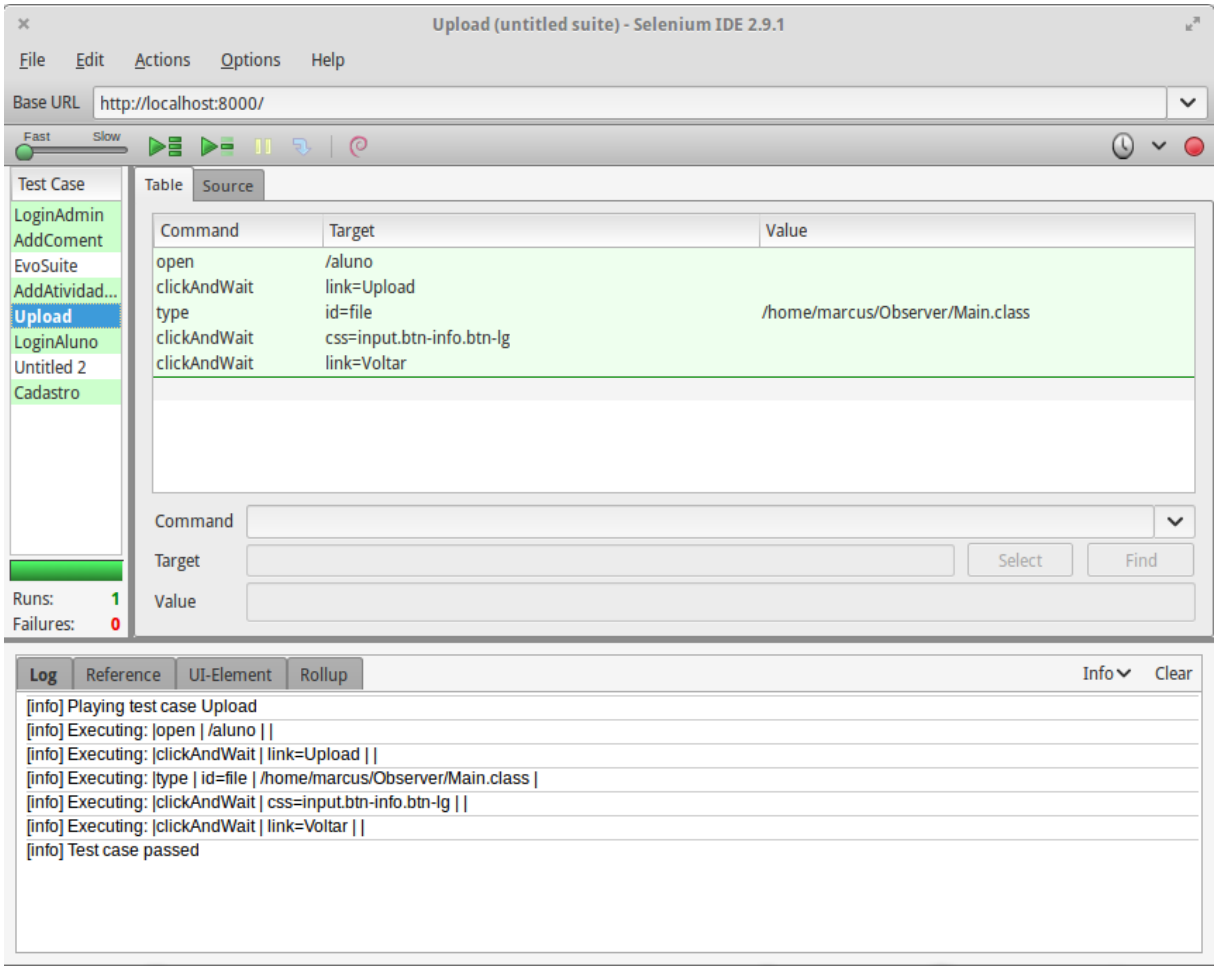

Figura 23 – Entradas e Ações para realização do envio do arquivo

Outra tarefa importante a ser validada para garantir o funcionamento correto, é a ação de fazer o *upload* do arquivo a ser testado pelo professor (Figura 23). É necessário que as informações sejam gravadas corretamente e que não ocorram falhas em nenhuma das etapas de envio.

O *upload* do arquivo deve ser feito para o diretório correto e apenas um arquivo *.class* deve ser selecionado. Após o envio do arquivo, o aluno será redirecionado para uma tela informando que o arquivo foi enviado com sucesso e uma tabela é apresentada com todos os arquivos enviados.

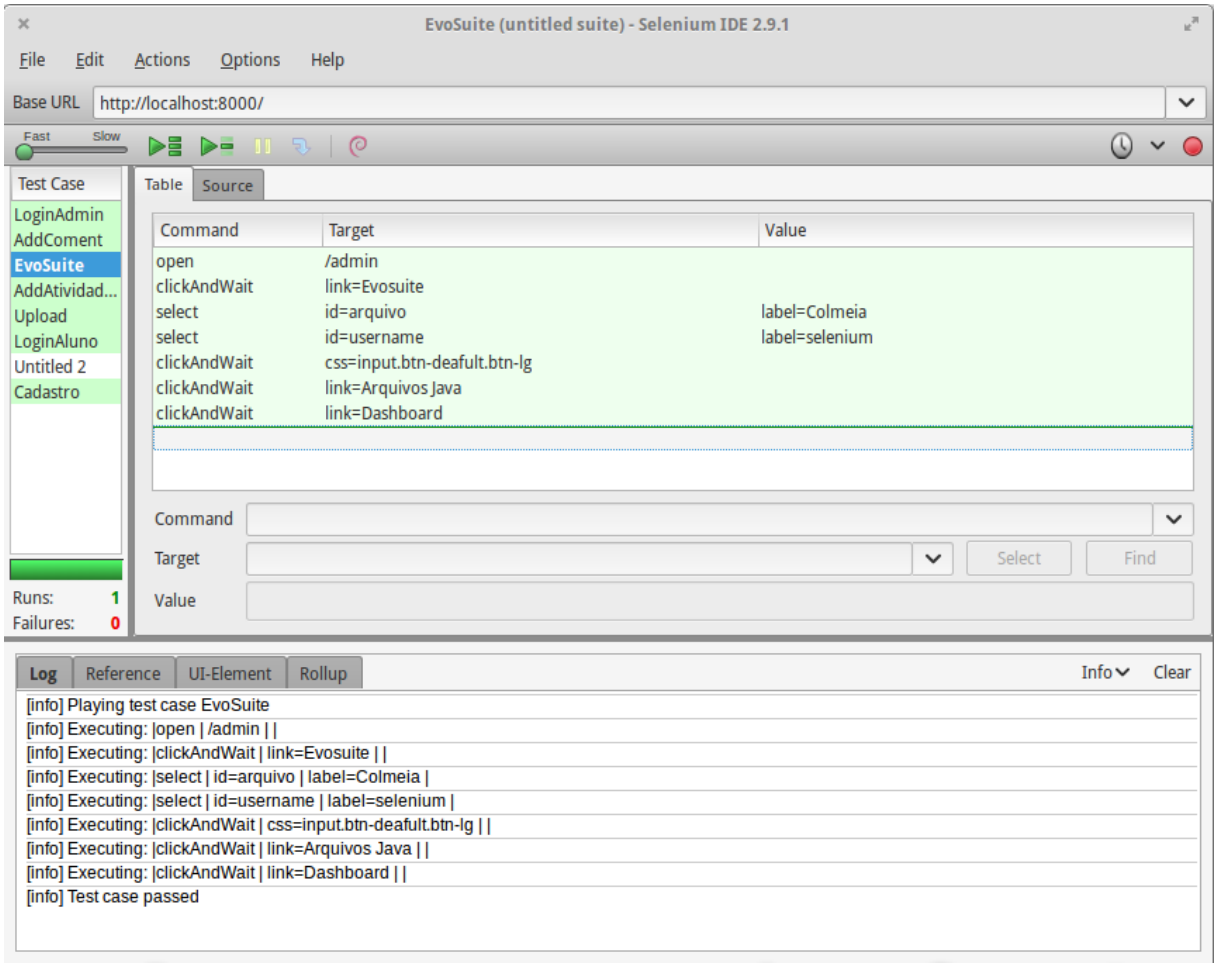

Figura 24 – Execução da ferramenta EvoSuite

A principal ação a ser executada pela ferramenta TESTAÍ é o acesso à ferramenta EvoSuite, que necessita receber os parâmetros corretos sobre as classes e o caminho correto para encontrar o arquivo *.class* e assim fazer o teste da classe. Como mostrado na Figura 24, a execução passou nos testes e retornou corretamente a finalização do teste da classe com sucesso.

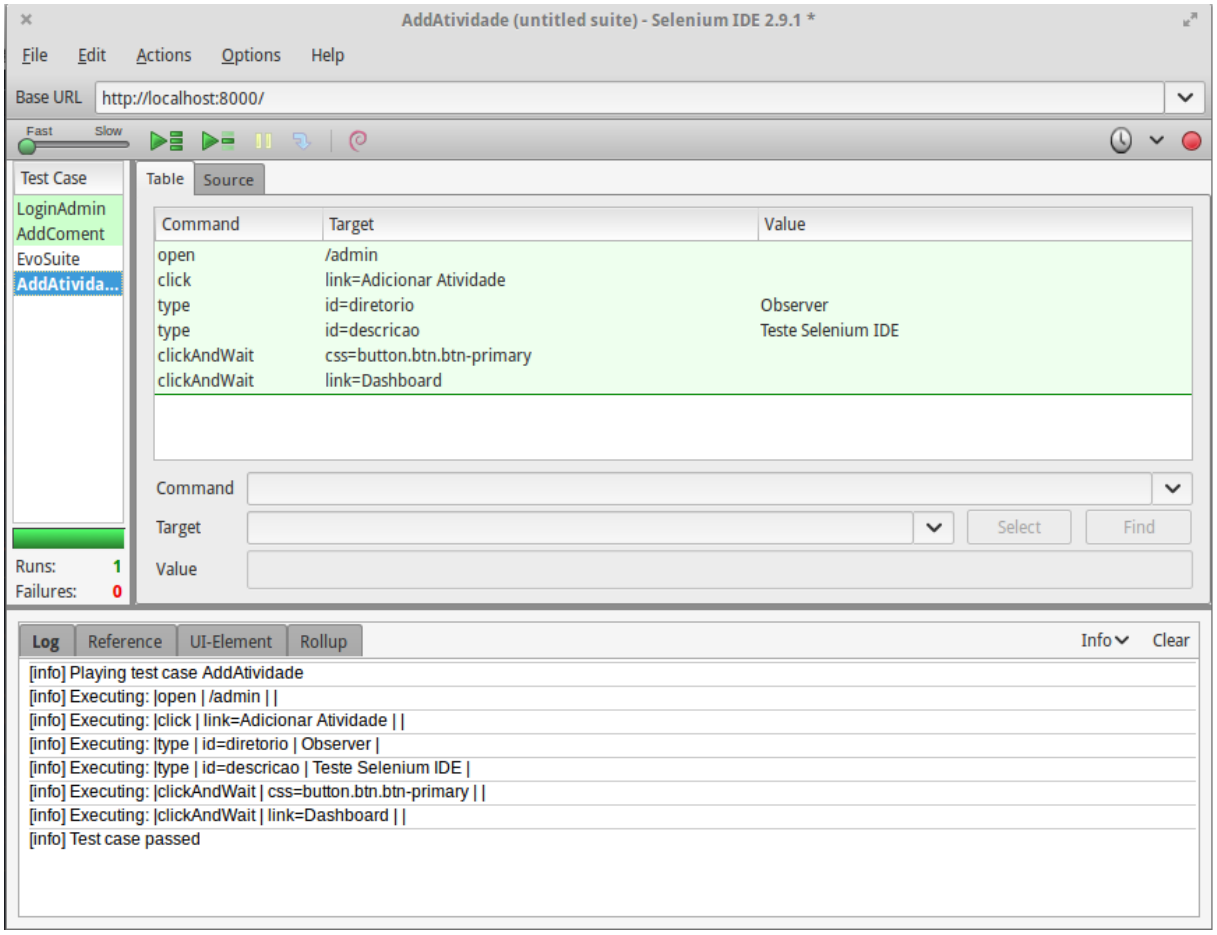

Figura 25 – Entradas e Ações para criação de uma Atividade

A criação de atividades é uma tarefa simples, e apenas aloca um diretório para receber os arquivos enviados pelos alunos.

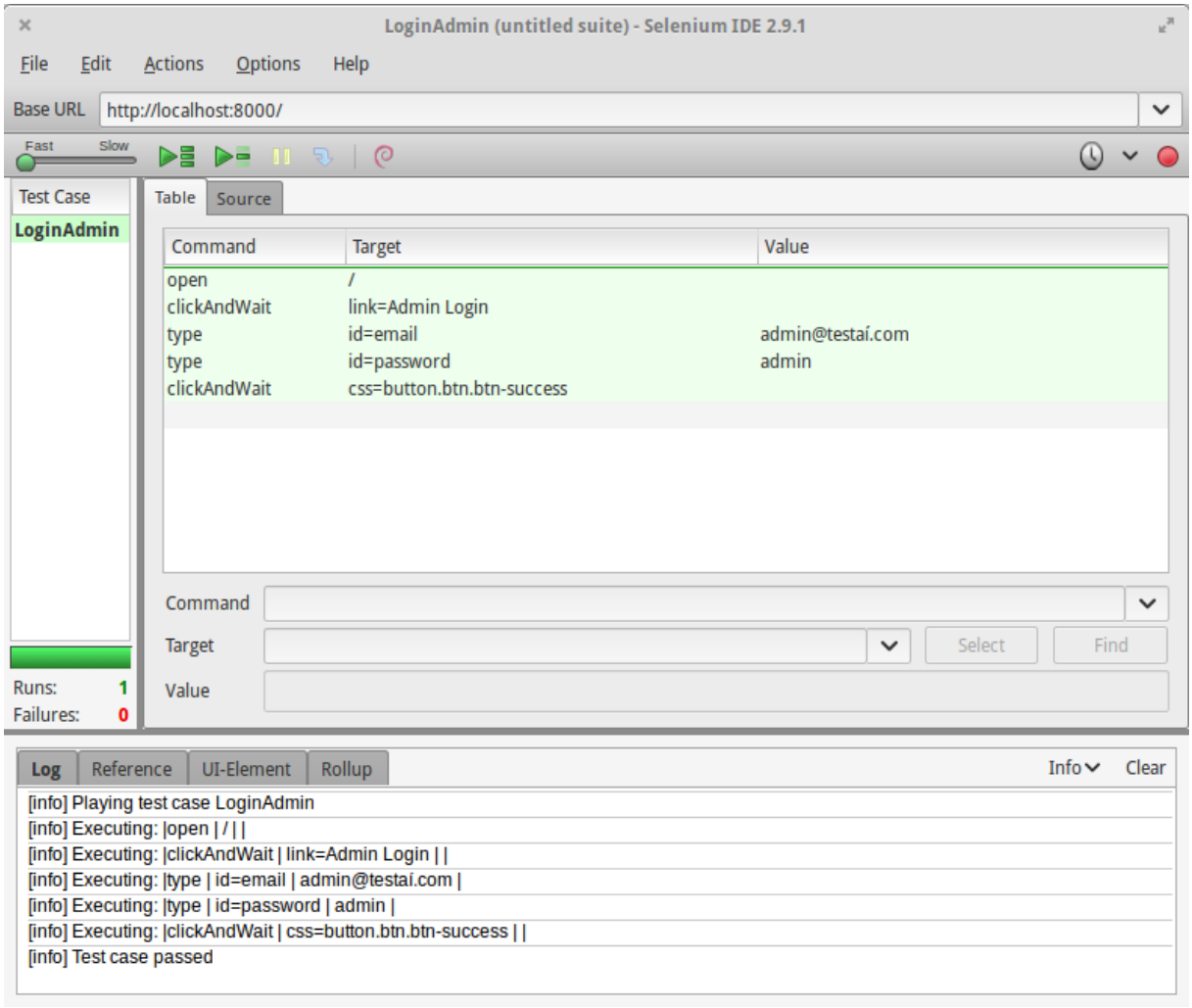

Figura 26 – Entradas e Ações para realização do login do professor/administrador

Da mesma forma que o aluno, o administrador (professor) também tem que fazer login na ferramenta (Figura 27). Nessa parte, entra o *middleware* que faz a proteção para que apenas administrador/professor acessem a área do administrador.

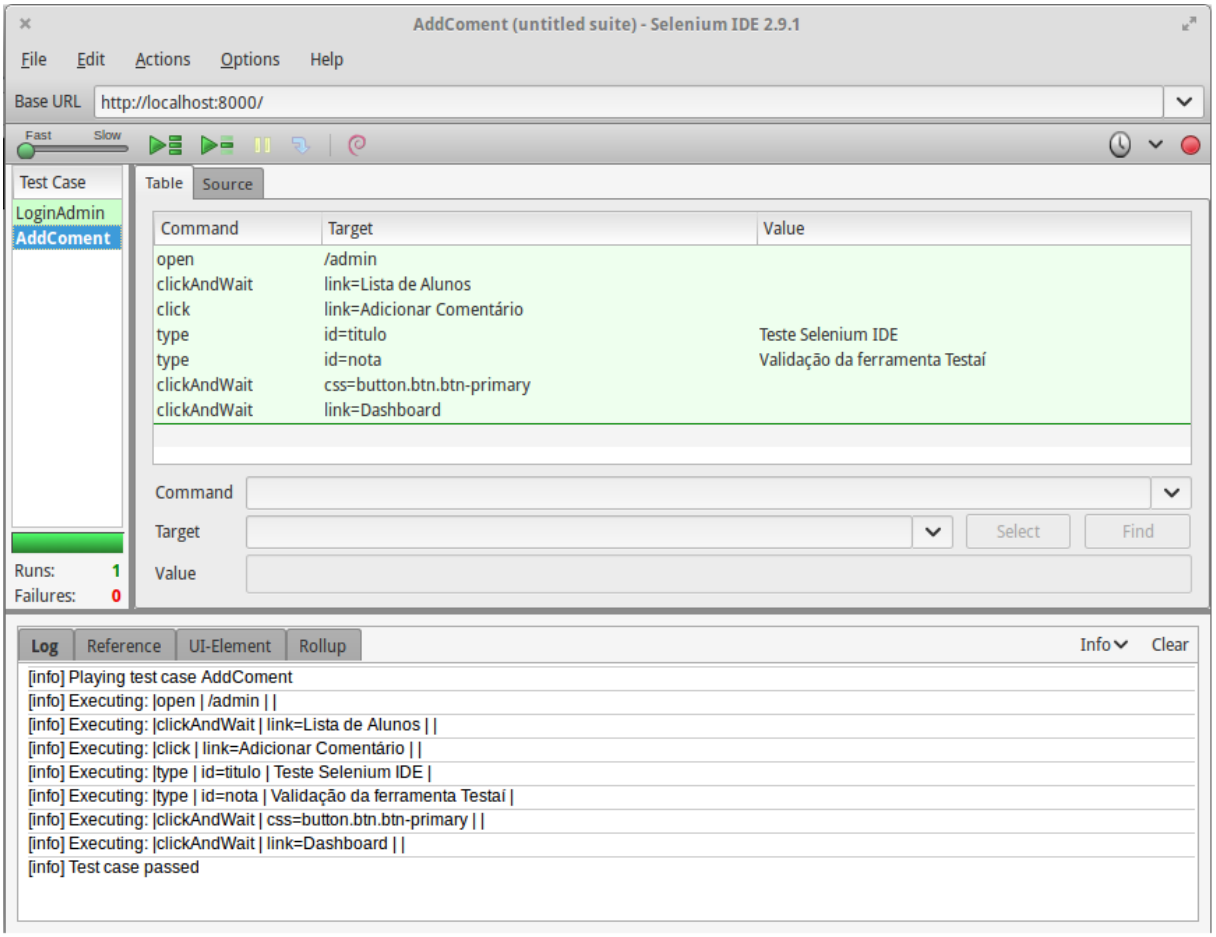

Figura 27 – Entradas e Ações para adicionar um comentário

É possível a adição de um comentário após o teste de uma classe JAVA, caso o professor deseje (Figura 27). Basta direcionar a um aluno e escrever o que deseja, e assim o aluno poderá ver todos os comentários a ele destinados em sua área.

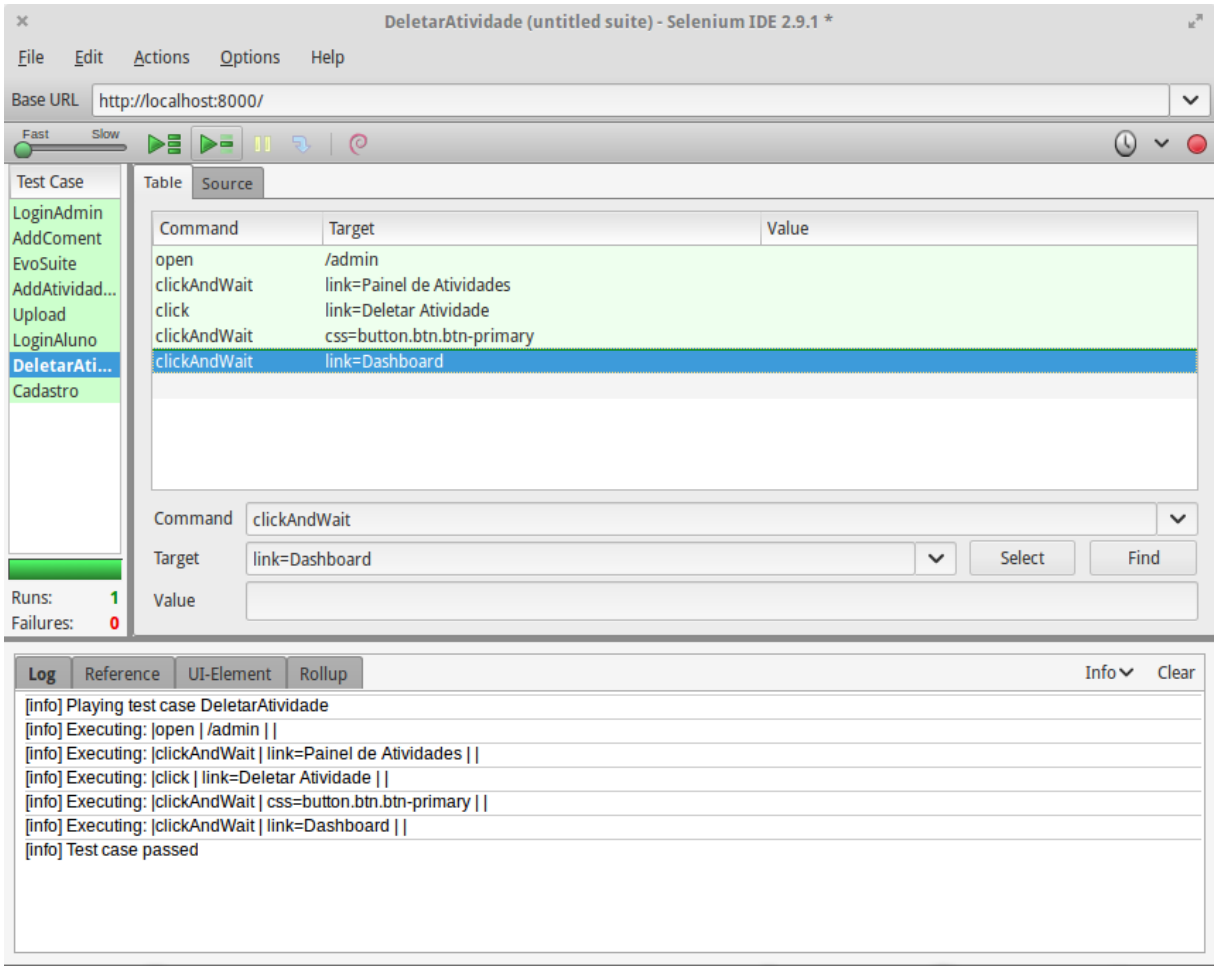

Figura 28 – Entradas e Ações para deletar uma atividade

A Figura 28 apresenta a ação para a remoção de uma atividade após a data limite de entrega.

Como mostrado pelas figuras, os testes da Selenium foram completados com sucesso, e as funcionalidades retornaram os resultados esperados. Sendo assim, a ferramenta **TESTAÍ** faz o que lhe é proposto, de maneira simples porém funcional.

Conclusão

A implementação da ferramenta **TESTAÍ** foi muito satisfatória no quesito aprendizagem. Foi possível analisar de maneira detalhada os avanços no ambiente de ensino a programação e ver que essa é uma preocupação real de todos que de alguma forma estão ligados ao processo de ensino.

As dificuldades encontradas durante o desenvolvimento da ferramenta se mostraram presentes e instigantes, sendo necessário muito empenho e pesquisa para solucionar certos impasses em relação a lógica utilizada e o caminho a ser tomado para a execução de cada passo.

A ferramenta utilizada para os testes de classes JAVA, EvoSuite EvoSuite (2016), se mostrou de grande ajuda e muito eficiente. A princípio, foi necessária uma análise mais detalhada de seu funcionamento, visto que o intuito do trabalho era usar a sua execução via linha de comando e não como um *plugin* do Eclipse. Porém, os resultados obtidos a partir de ensaios simples, foram satisfatórios e incentivaram a realização de análises mais detalhadas sobre as funcionalidades e possibilidades que a ferramenta oferece.

De fato, muito ainda pode ser feito, de acordo com o interesse do usuário. Muitas funcionalidades podem ser implementadas e até mesmo melhorias naquelas implementadas nesse trabalho. Mas como um primeiro passo para a melhoria do ensino, de maneira a ajudar tanto aluno quanto professor à identificarem problemas e melhorarem seus respectivos pontos fracos. Foi animador ver que o resultado, ao final da implementação e da validação, foi satisfatório, pois o funcionamento da ferramenta se mostrou correta.

Ficou claro que ainda é necessário mais esforço para a criação de um ambiente mais propício ao aluno iniciante na área da programação e que o acompanhamento do professor é imprescindível. Sendo assim, é importante proporcionar ao professor também, um ambiente capaz de auxiliá-lo nas correções e facilitar a identificação de pontos fracos dos alunos.

Como trabalhos futuros, o pensamento é de ampliação da ferramenta **TESTAÍ**. Uma ideia é a integração de outras ferramentas que testem programas escritos em diferentes linguagens (C, Python, PHP, Haskell etc). Dado que a ferramenta lida apenas com programas escritos em JAVA, restringindo o escopo de disciplinas onde pode ser utilizada, a ideia de suporte para outras linguagens abre possibilidade de utilização em outras disciplinas. Assim, será possível que diversos professores criem seu próprio espaço e gerenciem um conjunto diversificado de disciplinas. Outro ponto a ser explorado é a possibilidade de execução da EvoSuite em um ambiente de processamento distribuído, visto que a análise de mutantes realizada é um processo que pode ser bastante custoso do

ponto de vista computacional.

Referências

BERNAL, V. B.; HIRA, A. *Tipos de teste de software*. 2014. Disponível em: <http://disciplinas.stoa.usp.br/pluginfile.php/384739/mod_resource/content/1/Aula%205_2014_Tipos-de-teste-software-v2.pdf>. Citado na página 16.

CHAVES, J. O. et al. Uma ferramenta baseada em juízes online para apoio às atividades de programação de computadores no moodle. *Novas Tecnologias na Educação*, v. 11, n. 3, dezembro 2013. Citado na página 27.

CHAVES, J. O. M. Uma ferramenta de apoio ao processo de ensino-aprendizagem em disciplinas de programação de computadores por meio da integração dos juízes online ao moodle. 2014. Citado 2 vezes nas páginas 26 e 27.

DELAMARO, J. C. M. M. J. M. E. *Introdução ao Teste de Software*. [S.l.]: Elsevier, 2007. Citado 2 vezes nas páginas 16 e 19.

EVOSUITE. 2016. Disponível em: <http://www.evosuite.org/evosuite/>. Citado 3 vezes nas páginas 33, 34 e 46.

FRASER, G.; ARCURI, A. Evosuite: automatic test suite generation for object-oriented software. In: *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. New York, NY, USA: ACM, 2011. (ESEC/FSE '11), p. 416–419. ISBN 978-1-4503-0443-6. Disponível em: <http://doi.acm.org/10.1145/2025113.2025179>. Citado 3 vezes nas páginas 9, 22 e 23.

FRASER, G.; ARCURI, A. Evosuite at the sbst 2013 tool competition. In: IEEE. *6th International Workshop on Search-Based Software Testing (SBST'13) at ICST'13*. [S.l.], 2013. p. 406–409. Citado na página 24.

JIA, Y.; HARMAN, M. An analysis and survey of the development of mutation testing. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, v. 37, n. 5, p. 649–678, Setembro/Outubro 2011. Citado 3 vezes nas páginas 9, 20 e 21.

LARAVEL. 2016. Disponível em: <https://laravel.com/>. Citado na página 51.

LEBTAG, B. G. A. Cafezinho - um ambiente de apoio aoensino de programação. 2014. Citado na página 27.

MOZILLA Firefox. 2016. Disponível em: <https://www.mozilla.org/pt-BR/firefox/products/>. Citado na página 37.

MYERS, G. J. *The Art of Software Testing*. [S.l.]: John Wiley, 2004. Citado 3 vezes nas páginas 13, 16 e 17.

PRESSMAN, R. S. *Software Engineering A Practitioner's Approach*. 7. ed. [S.l.]: McGraw-Hill, 2010. v. 1. Citado 4 vezes nas páginas 9, 13, 17 e 18.

ROCHA, P. S. et al. Ensino e aprendizagem de programação: Análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino. *Novas Tecnologias na Educação*, v. 8, n. 3, Dezembro 2010. Citado na página 13.

SBST. 2013. Disponível em: <http://sbstcontest.dsic.upv.es/>. Citado na página 22.

SELENIUM IDE. 2016. Disponível em: <http://www.seleniumhq.org/projects/ide/>. Citado na página 37.

Apêndices

APÊNDICE A – Código fonte TESTAÍ

Como mencionado anteriormente, o *framework* utilizado para desenvolvimento deste projeto, Laravel Laravel (2016), utiliza a arquitetura MVC. Nesta seção, serão apresentados os *Models*, *Views* e *Controllers*.

A.1 Models

Os *Models* são as partes chaves do sistema. Eles gerenciam um ou mais elementos de dados e respondem a perguntas para mudança ou sobre seu estado. É o *Model* que modela o problema a ser resolvido.

Abaixo o código do ***Model* de Administradores** do sistema, que no caso poderá ser um professor.

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Foundation\Auth\User as Authenticatable;
7
8  class Admin extends Authenticatable
9  {
10   protected $guard = "admins";
11   /**
12    * The attributes that are mass assignable.
13    *
14    * @var array
15    */
16   protected $fillable = [
17       'name', 'username', 'email', 'password',
18   ];
19
20   /**
21    * The attributes excluded from the model's JSON form.
22    *
23    * @var array
24    */
25   protected $hidden = [
```

```
26        'password', 'remember_token',
27    ];
28 }
```

Model de Comentários

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Notas extends Model
8  {
9    public function notas(){
10
11     return $this->belongsTo(User::class);
12
13   }
14 }
```

Model de Usuários

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Foundation\Auth\User as Authenticatable;
6
7  class User extends Authenticatable
8  {
9
10     public function notas(){
11
12       return $this->hasMany(Notas::class);
13
14     }
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array
19      */
20     protected $fillable = [
21         'name', 'username', 'email', 'password',
```

```
22      ];
23
24      /**
25       * The attributes that should be hidden for arrays.
26       *
27       * @var array
28       */
29      protected $hidden = [
30          'password', 'remember_token',
31      ];
32  }
```

Model de Atividades

```
1   <?php
2
3   namespace App;
4
5   use Illuminate\Database\Eloquent\Model;
6
7   class Atividades extends Model
8   {
9     protected $table = 'atividades';
10    public $timestamps  = false;
11
12  protected $fillable = array('nome', 'descricao');
13  }
```

Model de Arquivos

```
1   <?php
2
3   namespace App;
4
5   use Illuminate\Database\Eloquent\Model;
6
7   class File extends Model
8   {
9     protected $table = 'files';
10    public $timestamps  = false;
11
12  protected $fillable = array('nome', 'tipo', 'file');
13  }
```

A.2 Controllers

Os *Controllers* fazem o papel de controle das operações. As funções são declaradas nos *Controllers* e executadas.

***Controller* de Administradores**

Todas as funções do Administrador estão presentes nesta parte do código. As funções de destaque nesta seção são a ***evoSuite*** e ***addNotas*** que fazem ações muito importantes para o resultado final do sistema.

A função ***evoSuite*** faz a chamada de execução da EvoSuite via linha de comando, verifica se o teste foi completado com sucesso utilizando a função *strstr* para verificar se há uma dada palavra no conteúdo da execução e preenche o campo "testado"na tabela de arquivos com "OK", caso falhe, o campo é preenchido com "Falhou". Há uma variável chamada *projectCP* que é um parâmetro a ser passado para a EvoSuite, indicando o local onde se encontra o arquivo *.class*.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\User;
7  use App\Http\Requests;
8  use App\File;
9  use Illuminate\Support\Facades\Input;
10 use DB;
11 use Carbon\Carbon;
12 use View;
13 use App\Notas;
14 use App\Atividades;
15
16 class AdminsController extends Controller
17 {
18     public function adminIndex(){
19
20       $users = User::where('tipo', '!=', 1)->get();
21       $files = File::all();
22       $atividades = Atividades::all();
23
24
25       return view('admins/admin', ['users'=> $users, 'files'=>
            $files, 'atividades' => $atividades]);
```

```
26      }
27
28      public function criarDiretorio(){
29
30        $dir = Input::get('diretorio');
31        mkdir("JavaFiles/$dir", 0777);
32        $descricao = Input::get('descricao');
33        $data = Carbon::now();
34        DB::table('atividades')->insertGetID(['nome' => $dir, '
            descricao' => $descricao, 'created_at' => $data]);
35
36        $atividades = Atividades::all();
37        return view('admins.atividades', ['atividades' =>
            $atividades]);
38
39      }
40
41      public function verFolders(){
42
43        $atividades = Atividades::all();
44        return view('admins.folders', ['atividades' => $atividades
            ]);
45      }
46
47      public function verAtividades(){
48        $atividades = Atividades::all();
49
50        return view('admins.atividades', ['atividades' =>
            $atividades]);
51      }
52
53      public function evoSuite(){
54
55        $arquivo = Input::get('arquivo');
56        $username = Input::get('username');
57        $dir = Input::get('dirName');
58
59        //$arquivo = ('/home/teste/TCC1-25-27-v1.0/public/JavaFiles
            /');
60        $projectCP = ('/home/teste/TCC1-25-27-v2.0-final/public/
            JavaFiles/');
61        $a = shell_exec("java -jar evosuite-1.0.3.jar -class $dir.
```

```
        $arquivo  -projectCP $projectCP ");
62
63
64      if(strstr($a, 'Done') == true) {
65        //echo 'Teste completado com sucesso';
66        DB::table('files')->where('file', '=', $arquivo)->where('
            user_name', '=', $username)->update(array('testado' =>
            'OK'));
67        $files = File::all();
68        $users = User::where('tipo', '!=', 1)->get();
69
70        echo "<pre>$a</pre>";
71
72
73      }else {
74        return 'Deu B2';
75      }
76
77
78
79  }
80
81  public function lista(){
82    $users = User::where('tipo', '!=', 1)->get();
83    return view('admins.lista_alunos', ['users'=> $users]);
84  }
85  public function listaArquivos(){
86
87    $files = File::all();
88    $users = User::where('tipo', '!=', 1)->get();
89    return view('admins.uploads_folder', ['files'=> $files, '
        users'=> $users]);
90
91  }
92
93  public function evoSuiteForm(){
94
95    $files= File::all();
96    $users = User::where('tipo', '!=', 1)->get();
97    $atividades = Atividades::all();
98
99    return view('admins.evosuiteForm', ['files' => $files, 'users
```

```
        '=> $users, 'atividades' => $atividades]);
100
101   }
102
103
104   public function addNotas(){
105
106     $user = Input::get('name');
107     $nota = Input::get('nota');
108     $titulo = Input::get('titulo');
109     $data = Carbon::now();
110     //DB::table('notas')->('name',$user)->update(['created_at' =>
          $data,  'notas' => $nota]);
111     DB::table('notas')->insertGetID(['user_name' => $user, 'body'
          => $nota, 'title' => $titulo, 'created_at' => $data]);
112     $users = User::where('tipo', '!=', 1)->get();
113
114
115     return view('admins.lista_alunos', ['users'=> $users]);
116   }
117
118   public function verNotas(){
119
120     //DB::table('notas')->insertGetID(['user_name' => $user, '
          body' => $nota, 'title' => $titulo, 'created_at' => $data])
          ;
121     $notas = Notas::all();
122     return view('admins.ver_notas', ['notas'=> $notas]);
123
124
125   }
126   public function folders(){
127
128     $dir = Input::get('dirName');
129    //$files1 = scandir($dir, 0);
130
131
132     return view('admins.folders', ['dirName' => $dir]);
133 }
134 }
```

Controller de Acesso

Uma função importante aqui é *__construct* que é uma função padrão do Laravel no início de cada projeto. Nela é possível escolher o tipo de autenticação que será aplicada, utilizando *Middleware*. No caso da **TESTAÍ**, foi implementado um *middleware* para evitar que alunos acessem a área de Admin.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Http\Requests;
6  use Illuminate\Http\Request;
7
8  class HomeController extends Controller
9  {
10     /**
11      * Create a new controller instance.
12      *
13      * @return void
14      */
15     public function __construct()
16     {
17         $this->middleware('admins');
18     }
19
20     /**
21      * Show the application dashboard.
22      *
23      * @return \Illuminate\Http\Response
24      */
25     public function index()
26     {
27         return view('welcome');
28     }
29
30     public function aluno(){
31
32       return view('alunos/aluno');
33     }
34
35 }
```

Controller de Alunos

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;
use App\User;
use App\File;
use Illuminate\Support\Facades\Input;
use DB;
use Carbon\Carbon;
use View;
use App\Notas;
use Auth;
use App\Atividades;

class AlunosController extends Controller
{
  public function upload(){
    $atividades = Atividades::all();
    return view('alunos/uploads', ['atividades' => $atividades]);


  }

  public function aluno(){



    return view('alunos/aluno');
  }

  public function showNotas(){


    $users = Auth::user()->name;
    $notas = Notas::where('user_name', '=', $users)->get();

    //$users = User::where('tipo', '!=', 1)->get();

```

```
42      return view('alunos/aluno_notas',['users'=> $users, 'notas'
            => $notas]);
43    }
44
45    public function uploadSuccess(){
46
47
48      $users = Auth::user()->name;
49      $files = File::where('user_name', '=', $users)->get();
50
51      //$users = User::where('tipo', '!=', 1)->get();
52
53      return view('alunos/uploadSuccess',['users'=> $users, 'files'
            => $notas]);
54    }
55
56
57
58
59
60  }
```

Controller de Arquivos

O controle dos arquivos é de extrema importância, principalmente pelo fato de que o *upload* é uma ação trabalhosa. Como é possível ver na função ***uploadFile***, o processo de envio do arquivo é custoso e passa por várias etapas de verificação.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\Input;
7  use App\Http\Requests;
8  use DB;
9  use Auth;
10 use App\User;
11 use Carbon\Carbon;
12 use App\File;
13 use App\Atividades;
14
15 class FilesController extends Controller
16 {
17   public function uploadFile(){
18
19     $dir = Input::get('dirName');
20
21     $target_dir = "JavaFiles/$dir/";
22     $target_file = $target_dir . basename($_FILES["file"]["name"
         ]);
23     $target_file1 = basename($_FILES["file"]["name"], ".class");
24     $uploadOk = 1;
25     $FileType = pathinfo($target_file,PATHINFO_EXTENSION);
26
27     $name = Auth::user()->username;
28     $data = Carbon::now();
29     $files = File::where('user_name', '=', $name)->get();
30     DB::table('files')->insertGetId(['user_name' => $name, 'file'
           => $target_file1, 'updated_at' => $data]);
31     // Check if image file is a actual image or fake image
32     if(isset($_POST["btn-upload"])) {
33
34         $check = pathinfo($_FILES["file"]["tmp_name"]);
35         if($check !== false) {
```

```
36
37              $uploadOk = 1;
38          } else {
39              return view('alunos.uploadError');
40              $uploadOk = 0;
41          }
42      }
43      // Check if file already exists
44      if (file_exists($target_file)) {
45
46          return view('alunos.uploadError');
47          $uploadOk = 0;
48      }
49      // Check file size
50      if ($_FILES["file"]["size"] > 500000) {
51
52          return view('alunos.uploadError');
53          $uploadOk = 0;
54      }
55      // Allow certain file formats
56      if($FileType != ("class" || "java")) {
57
58          return view('alunos.uploadError');
59          $uploadOk = 0;
60      }
61
62      // Check if $uploadOk is set to 0 by an error
63      if ($uploadOk == 0) {
64
65          return view('alunos.uploadError');
66      // if everything is ok, try to upload file
67      } else {
68
69          if (move_uploaded_file($_FILES["file"]["tmp_name"],
                $target_file)) {
70
71
72          } else {
73
74              return view('alunos.uploadError');
75          }
76      }
```

```
77          return view('alunos/uploadSuccess', ['files' => $files]);
78      }
79
80      public function deleteFile(){
81
82
83          $name = Input::get('username');
84          $file = Input::get('fileName');
85          DB::table('files')->where('file', '=', $file)->where('
                user_name', '=', $name)->delete();
86
87          $files = File::all();
88          $users = User::where('tipo', '!=', 1)->get();
89
90
91
92          return view('admins.uploads_folder', ['files'=> $files, '
                users' => $users]);
93
94
95      }
96      public function deleteAtividade(){
97
98
99          $name = Input::get('nome');
100         //$file = Input::get('fileName');
101         DB::table('atividades')->where('nome', '=', $name)->delete();
102
103         $atividades = Atividades::all();
104         //$users = User::where('tipo', '!=', 1)->get();
105
106
107
108         return view('admins.atividades', ['atividades'=> $atividades
                ]);
109
110
111     }
112
113 }
```

A.3 *Views*

A.3.1 *Views* do Administrador

***View* da página inicial do Administrador**

```
1  @extends('admins.admin_base')
2
3  @section('headerside')
4  <li class="active">
5      <a href="{{url ('admin')}}"><i class="fa fa-fw fa-dashboard"
          ></i> Dashboard</a>
6  </li>
7  <li>
8      <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw fa-bar-
          chart-o"></i> Lista de Alunos</a>
9  </li>
10 <li>
11     <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw fa-
          table"></i> Evosuite</a>
12 </li>
13 <li>
14 <a href="{{url ('uploads_folder')}}"><i class="fa fa-fw fa-bar-
      chart-o"></i> Arquivos Java</a>
15 </li>
16 <li>
17 <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar-chart-o
      "></i>Ver Comentários</a>
18 </li>
19 <li>
20 <a href="{{url ('folders')}}"><i class="fa fa-fw fa-bar-chart-o"
      ></i>Ver Diretórios</a>
21 </li>
22 <li>
23 <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-bar-chart-
      o"></i>Painel de Atividades</a>
24 </li>
25
26 @endsection
27
28
29 @section('content')
30 <div class="container-fluid">
```

```
31
32      <!-- Page Heading -->
33      <div class="row">
34          <div class="col-lg-12">
35              <h1 class="page-header">
36                  Painel de Controle <small>Dados Gerais</small>
37              </h1>
38              <ol class="breadcrumb">
39                  <li class="active">
40                      <i class="fa fa-dashboard"></i> Dashboard
41
42
43
44
45                  </li>
46              </ol>
47          </div>
48      </div>
49      <!-- /.row -->
50
51      <div class="row">
52          <div class="col-lg-12">
53
54          </div>
55      </div>
56      <!-- /.row -->
57
58
59
60
61      <div class="row">
62
63          <div class="col-lg-4">
64              <div class="panel panel-default">
65                  <div class="panel-heading">
66                      <h3 class="panel-title"><i class="fa fa-clock
                           -o fa-fw"></i> Painel de Tarefas</h3>
67                  </div>
68                  <div class="panel-body">
69                      <div class="list-group">
70
71                          @foreach ($atividades as $atividade)
```

```
72    <a href="{{url ('atividades')}}" class="list-group-item">
73                                <tr>
74                                  <div class="list-group">
75                                  <td>{{$atividade->nome}}</td>
76                                    </div>
77                                </tr>
78  </a>
79                                @endforeach
80
81
82
83
84                        </div>
85                        <a type="button" class="button btn btn-
                             default" data-toggle="modal" data-target="#
                             addAtv">
86                          Adicionar Atividade
87                        </a>
88                    </div>
89                </div>
90            </div>
91            <div class="col-lg-4">
92                <div class="panel panel-default">
93                    <div class="panel-heading">
94                        <h3 class="panel-title"><i class="fa fa-money
                              fa-fw"></i> Painel de Alunos</h3>
95                    </div>
96                    <div class="panel-body">
97                        <div class="table-responsive">
98                            <table class="table table-bordered table-
                                 hover table-striped">
99                              <thead>
100                                 <tr>
101                                     <th>Alunos</th>
102
103                                 </tr>
104                             </thead>
105                             <tbody>
106                                 @foreach ($users as $user)
107                                 <tr>
108                                   <td>{{$user->name}}</td>
109                                 </tr>
```

```
110                                 @endforeach
111                             </tbody>
112                         </table>
113                     </div>
114
115                 </div>
116             </div>
117         </div>
118         <div class="col-lg-4">
119             <div class="panel panel-default">
120                 <div class="panel-heading">
121                     <h3 class="panel-title"><i class="fa fa-money
                            fa-fw"></i> Lista de Envios</h3>
122                 </div>
123                 <div class="panel-body">
124                     <div class="table-responsive">
125                         <table class="table table-bordered table-
                                hover table-striped">
126                             <thead>
127                                 <tr>
128                                     <th>Arquivos</th>
129                                     <th>Data</th>
130
131                                 </tr>
132                             </thead>
133                             <tbody>
134                                 @foreach ($files as $file)
135                                 <tr>
136                                   <td>{{$file->file}}</td>
137                                   <td>{{$file->updated_at}}</td>
138                                 </tr>
139                                 @endforeach
140                             </tbody>
141                         </table>
142                     </div>
143
144                 </div>
145             </div>
146         </div>
147
148     </div>
149     <!-- /.row -->
```

```
150
151 </div>
152 <!-- /.container-fluid -->
153
154 <!-- Modal de ADD ATIVIDADE-->
155 <div class="modal fade" id="addAtv" tabindex="-1" role="dialog"
       aria-labelledby="addAtv">
156   <div class="modal-dialog" role="document">
157     <div class="modal-content">
158       <div class="modal-header">
159         <button type="button" class="close" data-dismiss="modal"
              aria-label="Close"><span aria-hidden="true">&times;</
              span></button>
160         <h4 class="modal-title" id="addAtv">Adicionar Atividade</
              h4>
161       </div>
162       <div class="modal-body">
163         <form class="form" action="{{url('criarDiretorio')}}"
              method="post">
164               {{ csrf_field() }}
165
166
167               <div class="form-group">
168                 <label for="name"> Nome da Atividade:</label>
169
170
171                 <input type="text" class="form-control" id="
                      diretorio" name="diretorio" placeholder="
                      Titulo" required>
172
173
174               </div>
175
176           <div class="form-group">
177               <label for="name"> Descrição:</label>
178               <input type="text" class="form-control" id="
                     descricao" name="descricao" placeholder="Titulo"
                     required>
179           </div>
180
181
```

```
182           <button type="button" class="btn btn-danger" data-
                 dismiss="modal">Cancelar</button>
183           <button type="submit" class="btn btn-primary">Criar</
                 button>
184
185         </form>
186       </div>
187     </div>
188   </div>
189 </div><!--./modal -->
190 @endsection
```

View para visualizar atividades

```
1  @extends('admins.admin_base')
2
3  @section('headerside')
4  <!-- Sidebar Menu Items - These collapse to the responsive
      navigation menu on small screens -->
5
6          <li >
7              <a href="{{url ('admin')}}"><i class="fa fa-fw fa-
                   dashboard"></i> Dashboard</a>
8          </li>
9          <li >
10             <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw
                    fa-bar-chart-o"></i> Lista de Alunos</a>
11         </li>
12         <li>
13             <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw
                    fa-table"></i> Evosuite</a>
14         </li>
15         <li>
16         <a href="{{url ('uploads_folder')}}"><i class="fa fa-fw
              fa-bar-chart-o"></i> Arquivos Java</a>
17         </li>
18         <li>
19         <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar
              -chart-o"></i>Ver Comentários</a>
20         </li>
21         <li>
22         <a href="{{url ('verFolders')}}"><i class="fa fa-fw fa-
              bar-chart-o"></i>Ver Diretórios</a>
```

```
23          </li>
24          <li class="active">
25          <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-
               bar-chart-o"></i>Painel de Atividades</a>
26          </li>
27
28
29  @endsection
30
31  @section('content')
32
33  <table class="table table-striped table-hover">
34        <tr>
35          <th></th>
36          <th>Nome</th>
37          <th>Descrição</th>
38          <th>Data de Criação</th>
39
40        </tr>
41        @foreach ($atividades as $atividade)
42          <tr height="30">
43            <td><i class="fa fa-2x fa-user"></i></td>
44            <td>{{$atividade->nome}}</td>
45            <td>{{$atividade->descricao}}</td>
46            <td>{{$atividade->created_at}}</td>
47
48            <td ></td>
49          </tr>
50        @endforeach
51      </table>
52
53      <a type="button" class="button btn btn-default" data-toggle="
           modal" data-target="#DeleteFile">
54        Deletar Atividade
55      </a>
56
57
58      <!-- Modal -->
59      <div class="modal fade" id="DeleteFile" tabindex="-1" role="
           dialog" aria-labelledby="DeleteFileLabel">
60        <div class="modal-dialog" role="document">
61          <div class="modal-content">
```

```
62          <div class="modal-header">
63            <button type="button" class="close" data-dismiss="
                 modal" aria-label="Close"><span aria-hidden="true"
                 >&times;</span></button>
64            <h4 class="modal-title" id="DeleteFileLabel">Deletar
                 Atividade</h4>
65          </div>
66          <div class="modal-body">
67            <form class="form" action="{{url('deleteAtividade')}}
                 " method="post">
68                 {{ csrf_field() }}
69
70
71                 <div class="form-group">
72                   <label for="nome"> Nome da Atividade:</
                        label>
73
74                   <select class="form-control" id="nome" name
                        ="nome" placeholder="Nome" required>
75                     @foreach ($atividades as $atividade)
76                       <option>{{$atividade->nome}}</option>
77                       @endforeach
78                   </select>
79
80                 </div>
81
82
83              <button type="button" class="btn btn-danger" data-
                   dismiss="modal">Cancelar</button>
84              <button type="submit" class="btn btn-primary">
                   Deletar</button>
85
86            </form>
87          </div>
88        </div>
89      </div>
90    </div><!--./modal -->
91
92 @endsection
```

View com formulário para chamada da EvoSuite

```
1 @extends('admins.admin_base')
```

```
2
3  @section('headerside')
4  <!-- Sidebar Menu Items - These collapse to the responsive
       navigation menu on small screens -->
5
6          <li >
7              <a href="{{url ('admin')}}"><i class="fa fa-fw fa-
                   dashboard"></i> Dashboard</a>
8          </li>
9          <li >
10             <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw
                    fa-bar-chart-o"></i> Lista de Alunos</a>
11         </li>
12         <li class="active">
13             <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw
                    fa-table"></i> Evosuite</a>
14         </li>
15         <li>
16         <a href="{{url ('uploads_folder')}}"><i class="fa fa-fw
              fa-bar-chart-o"></i> Arquivos Java</a>
17         </li>
18         <li>
19         <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar
              -chart-o"></i>Ver Comentários</a>
20         </li>
21         <li>
22         <a href="{{url ('verFolders')}}"><i class="fa fa-fw fa-
              bar-chart-o"></i>Ver Diretórios</a>
23         </li>
24         <li>
25         <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-
              bar-chart-o"></i>Painel de Atividades</a>
26         </li>
27
28
29
30 @endsection
31
32 @section('content')
33
34
35
```

```
36
37 <div class="evotest">
38       <form id="evotest" action="{{ url('/evoSuite') }}" method="
            post">
39
40             <label>Selecione um arquivo:</label>
41             <select class="form-control" name="arquivo" id="
                  arquivo">
42             @foreach ($files as $file)
43             <option>{{$file->file}}</option>
44             @endforeach
45             </select>
46
47             <br>
48
49             <label>Selecione um Aluno:</label>
50             <select class="form-control" id="username" name="
                  username" placeholder="Nome" required>
51               @foreach ($users as $user)
52                 <option>{{$user->username}}</option>
53                 @endforeach
54             </select>
55
56   <br>
57             <label>Selecione um Diretório:</label>
58             <select class="form-control" id="dirName" name="
                  dirName" placeholder="Diretório" required>
59               @foreach ($atividades as $atividade)
60                 <option>{{$atividade->nome}}</option>
61                 @endforeach
62             </select>
63
64
65
66
67       </br><input type="submit"  class="btn-deafult btn-lg "></br
            >
68       <input type="hidden" value="{{ csrf_token() }}" name="
            _token">
69       </form>
70 </div>
71
```

```
72
73 @endsection
```

View com a lista de alunos

```
1  @extends('admins.admin_base')
2
3  @section('headerside')
4  <!-- Sidebar Menu Items - These collapse to the responsive
       navigation menu on small screens -->
5
6          <li >
7              <a href="{{url ('admin')}}"><i class="fa fa-fw fa-
                   dashboard"></i> Dashboard</a>
8          </li>
9          <li class="active">
10             <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw
                    fa-bar-chart-o"></i> Lista de Alunos</a>
11         </li>
12         <li>
13             <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw
                    fa-table"></i> Evosuite</a>
14         </li>
15         <li>
16         <a href="{{url ('uploads_folder')}}"><i class="fa fa-fw
               fa-bar-chart-o"></i> Arquivos Java</a>
17         </li>
18         <li>
19         <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar
               -chart-o"></i>Ver Comentários</a>
20         </li>
21         <li>
22         <a href="{{url ('verFolders')}}"><i class="fa fa-fw fa-
               bar-chart-o"></i>Ver Diretórios</a>
23         </li>
24         <li>
25         <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-
               bar-chart-o"></i>Painel de Atividades</a>
26         </li>
27
28
29 @endsection
30
```

```
@section('content')

<table class="table table-striped table-hover">
      <tr>
        <th></th>
        <th>Aluno</th>
        <th>Nome de Usuário</th>
        <th>Email</th>

      </tr>
      @foreach ($users as $user)
        <tr height="30">
          <td><i class="fa fa-2x fa-user"></i></td>
          <td>{{$user->name}}</td>
          <td>{{$user->username}}</td>
          <td>{{$user->email}}</td>
          <td ></td>
        </tr>
      @endforeach
    </table>

    <a type="button" class="button btn btn-default" data-toggle="
       modal" data-target="#addNota">
      Adicionar Comentário
    </a>

  <!-- Modal -->
  <div class="modal fade" id="addNota" tabindex="-1" role="dialog
     " aria-labelledby="addNotatLabel">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal
             " aria-label="Close"><span aria-hidden="true">&times
             ;</span></button>
          <h4 class="modal-title" id="addNotaLabel">Adicionar
             Comentários</h4>
        </div>
        <div class="modal-body">
          <form class="form" action="{{url('add_notas')}}" method
             ="post">
                {{ csrf_field() }}
```

```
67
68
69                  <div class="form-group">
70                    <label for="name"> Nome do Aluno:</label>
71
72                    <select class="form-control" id="name" name="
                         name" placeholder="Nome" required>
73                      @foreach ($users as $user)
74                        <option>{{$user->name}}</option>
75                        @endforeach
76                    </select>
77
78                  </div>
79
80              <div class="form-group">
81                  <label for="name"> Título:</label>
82                  <input type="text" class="form-control" id="
                       titulo" name="titulo" placeholder="Titulo"
                       required>
83              </div>
84
85
86              <div class="form-group">
87                <label for="name"> Comentários:</label>
88                <input type="text" class="form-control" id="nota"
                     name="nota" placeholder="Nota" required>
89              </div>
90
91
92              <button type="button" class="btn btn-danger" data-
                   dismiss="modal">Cancelar</button>
93              <button type="submit" class="btn btn-primary">Salvar
                   </button>
94
95            </form>
96          </div>
97        </div>
98      </div>
99    </div><!--./modal -->
100 @endsection
```

View com a lista de arquivos

```
@extends('admins.admin_base')

@section('headerside')
<!-- Sidebar Menu Items - These collapse to the responsive
   navigation menu on small screens -->

        <li >
            <a href="{{url ('admin')}}"><i class="fa fa-fw fa-
               dashboard"></i> Dashboard</a>
        </li>
        <li >
            <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw
                fa-bar-chart-o"></i> Lista de Alunos</a>
        </li>
        <li>
            <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw
                fa-table"></i> Evosuite</a>
        </li>
        <li class="active">
            <a href="{{url ('uploads_folder')}}"><i class="fa fa-
               fw fa-bar-chart-o"></i> Arquivos Java</a>
        </li>
        <li>
        <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar
           -chart-o"></i>Ver Comentários</a>
        </li>
        <li>
        <a href="{{url ('verFolders')}}"><i class="fa fa-fw fa-
           bar-chart-o"></i>Ver Diretórios</a>
        </li>
        <li>
        <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-
           bar-chart-o"></i>Painel de Atividades</a>
        </li>


@endsection

@section('content')

<table class="table table-striped table-hover">
      <tr>
```

```
35          <th></th>
36          <th>Aluno</th>
37          <th>Nome Do Arquivo</th>
38          <th>Data</th>
39          <th>Situação</th>
40
41        </tr>
42        @foreach ($files as $file)
43          <tr height="30">
44            <td><i class="fa fa-2x fa-user"></i></td>
45            <td>{{$file->user_name}}</td>
46            <td>{{$file->file}}</td>
47            <td>{{$file->updated_at}}</td>
48            <td>{{$file->testado}}</td>
49
50          </tr>
51        @endforeach
52      </table>
53
54      <a type="button" class="button btn btn-default" data-toggle="
           modal" data-target="#DeleteFile">
55        Deletar Arquivo
56      </a>
57
58
59      <!-- Modal -->
60      <div class="modal fade" id="DeleteFile" tabindex="-1" role="
           dialog" aria-labelledby="DeleteFileLabel">
61        <div class="modal-dialog" role="document">
62          <div class="modal-content">
63            <div class="modal-header">
64              <button type="button" class="close" data-dismiss="
                   modal" aria-label="Close"><span aria-hidden="true"
                   >&times;</span></button>
65              <h4 class="modal-title" id="DeleteFileLabel">Deletar
                   Arquivo</h4>
66            </div>
67            <div class="modal-body">
68              <form class="form" action="{{url('deleteFile')}}"
                   method="post">
69                    {{ csrf_field() }}
70
```

```
71
72                      <div class="form-group">
73                        <label for="name"> Nome do Aluno:</label>
74
75                        <select class="form-control" id="username"
                             name="username" placeholder="Nome"
                             required>
76                          @foreach ($users as $user)
77                            <option>{{$user->username}}</option>
78                            @endforeach
79                        </select>
80
81                      </div>
82
83                      <div class="form-group">
84                        <label for="name">Nome do Arquivo:</label>
85
86                        <select class="form-control" id="fileName"
                             name="fileName" placeholder="Arquivo"
                             required>
87                          @foreach ($files as $file)
88                            <option>{{$file->file}}</option>
89                            @endforeach
90                        </select>
91
92                      </div>
93
94
95                <button type="button" class="btn btn-danger" data-
                     dismiss="modal">Cancelar</button>
96                <button type="submit" class="btn btn-primary">
                     Deletar</button>
97
98              </form>
99            </div>
100         </div>
101       </div>
102     </div><!--./modal -->
103 @endsection
```

View com a lista de comentários

```
1 @extends('admins.admin_base')
```

```
2
3  @section('headerside')
4  <!-- Sidebar Menu Items - These collapse to the responsive
       navigation menu on small screens -->
5
6          <li >
7              <a href="{{url ('admin')}}"><i class="fa fa-fw fa-
                   dashboard"></i> Dashboard</a>
8          </li>
9          <li >
10             <a href="{{url ('lista_alunos')}}"><i class="fa fa-fw
                    fa-bar-chart-o"></i> Lista de Alunos</a>
11         </li>
12         <li>
13             <a href="{{url ('evosuiteForm')}}"><i class="fa fa-fw
                    fa-table"></i> Evosuite</a>
14         </li>
15         <li >
16             <a href="{{url ('uploads_folder')}}"><i class="fa fa-
                   fw fa-bar-chart-o"></i> Arquivos Java</a>
17         </li>
18         <li class="active">
19         <a href="{{url ('ver_notas')}}"><i class="fa fa-fw fa-bar
              -chart-o"></i>Ver Comentários</a>
20         </li>
21         <li>
22         <a href="{{url ('verFolders')}}"><i class="fa fa-fw fa-
              bar-chart-o"></i>Ver Diretórios</a>
23         </li>
24         <li>
25         <a href="{{url ('atividades')}}"><i class="fa fa-fw fa-
              bar-chart-o"></i>Painel de Atividades</a>
26         </li>
27
28
29 @endsection
30
31 @section('content')
32
33 <table class="table table-striped table-hover">
34       <tr>
35         <th></th>
```

```
36          <th>Aluno</th>
37          <th>Titulo</th>
38          <th>Conteúdo</th>
39          <th>Data</th>
40
41        </tr>
42        @foreach ($notas as $nota)
43          <tr height="30">
44            <td><i class="fa fa-2x fa-comment"></i></td>
45            <td>{{$nota->user_name}}</td>
46            <td>{{$nota->title}}</td>
47            <td>{{$nota->body}}</td>
48            <td>{{$nota->created_at}}</td>
49
50          </tr>
51        @endforeach
52      </table>
53
54
55
56  @endsection
```

A.3.2 Views do Aluno

Página inicial do Aluno

```
1  @extends('layouts.app')
2
3  @section('header')
4
5  <li class="active"><a href="{{ url ('/aluno') }}">Pagina Inicial
       </a></li>
6  <li><a href="{{ url ('/uploads') }}">Upload </a></li>
7  <li><a href="{{ url ('/aluno_notas') }}">Ver Comentarios </a></li
       >
8  <li><a href="{{ url ('/logout') }}">Sair </a></li>
9
10
11
12 @stop
13
14 @section('content')
```

```
<div class="container">
    <div class="row">
        <div class=" ">
            <div class="panel panel-default">
                <div class="panel-heading"><h1 class="text-info">
                   Pagina Inicial</h2></div>

                <div class="panel-body">
                  @if( Auth::check() )
                  <h2 class="text-success"> Bem vindo {{ Auth::
                     user()->name}} !</h2>
                  @endif

                </div>
            </div>
        </div>
    </div>
</div>

@endsection
```

***View* com os comentários enviados ao aluno**

```
@extends('layouts.app')

@section('header')
<li ><a href="{{ url ('/aluno') }}">Pagina Inicial </a></li>
<li><a href="{{ url ('alunos/uploads') }}">Upload </a></li>
<li class="active"><a href="{{ url ('/aluno_notas') }}">Ver
   Comentarios </a></li>
<li><a href="{{ url ('/logout') }}">Sair </a></li>

@endsection

@section('content')
<div class="container">
    <div class="row">
        <div class=" ">
            <div class="panel panel-warning">
                <div class="panel-heading">
                  @if( Auth::check() )
                  <h2 class="text-success"> {{ Auth::user()->name
                     }}</h2>
```

```
19                    @endif
20
21                </div>
22                <div class="panel-body">
23                <table class="table table-striped table-hover">
24                      <tr>
25                        <th></th>
26                        <th class="text-primary" style="text-
                             align:center;">Aluno</th>
27                        <th class="text-primary" style="text-
                             align:center;">Titulo</th>
28                        <th class="text-primary" style="text-
                             align:center;">Comentarios</th>
29
30                      </tr>
31                      @foreach ($notas as $nota)
32                        <tr height="30">
33                          <td ><i class="fa fa-2x fa-user"></i></
                               td>
34                          <td class="text-warning">{{$nota->
                               user_name}}</td>
35                          <td class="text-warning">{{$nota->title
                               }}</td>
36                          <td class="text-warning">{{$nota->body
                               }}</td>
37                          <td ></td>
38                        </tr>
39                      @endforeach
40                    </table>
41
42              </div>
43          </div>
44      </div>
45  </div>
46
47  @endsection
```

View com mensagem de erro ao fazer Upload

```
1  @extends('layouts.app')
2
3  @section('header')
4
```

```
<li ><a href="{{ url ('aluno') }}">Voltar </a></li>
<li><a href="{{ url ('/logout') }}">Sair </a></li>

@endsection

@section('content')
<?php
echo "<pre>File is not an java file or</pre>";

echo "<pre>Sorry, file already exists or</pre>";

echo "<pre>Sorry, your file is too large or</pre>";

echo "<pre>Sorry, only java files are allowed or</pre>";

  echo "<pre>Sorry, your file was not uploaded or</pre>";

    echo "<pre>Sorry, there was an error uploading your file.</
       pre>";

?>
@endsection
```

View com o formulário de envio de arquivo

```
@extends('layouts.app')

@section('header')

<li><a href="{{ url ('/aluno') }}">Pagina Inicial </a></li>
<li class="active"><a href="{{ url ('alunos/uploads') }}">Upload
   </a></li>
<li ><a href="{{ url ('/aluno_notas') }}">Ver Comentarios </a></
   li>
<li><a href="{{ url ('/logout') }}">Sair </a></li>



@stop


@section('content')

```

```
17
18 <div class="upload">
19       <form id="upload" action="{{ url('/uploadFile') }}" method=
            "post" enctype="multipart/form-data">
20         <select class="btn btn-info" name="dirName" id="dirName">
21         @foreach ($atividades as $atividade)
22         <option>{{$atividade->nome}}</option>
23         @endforeach
24         </select>
25
26       </br></br>
27
28       <input type="file" name="file" id="file" class="btn btn-
            warning btn-lg center-block" required>
29       </br><input type="submit"  class="btn-info btn-lg "></br>
30       <input type="hidden" value="{{ csrf_token() }}" name="
            _token">
31       </form>
32 </div>
33
34
35
36
37 @endsection
```

***View* de sucesso ao enviar um arquivo**

```
1  @extends('layouts.app')
2
3
4  @section('header')
5
6  <li ><a href="{{ url ('aluno') }}">Voltar </a></li>
7  <li><a href="{{ url ('/logout') }}">Sair </a></li>
8
9  @endsection
10
11 @section('content')
12
13  <div class="container">
14      <div class="row">
15          <div class=" col-md-8 col-md-offset-2">
16              <div class="panel panel-success">
```

```
17                  <div class="panel-heading"><h4>
18                    <?php echo  "Arquivo ". basename( $_FILES["
                         file"]["name"]). " salvo com sucesso!.";?>
19 </h4>
20              </div>
21              <div class="panel panel-warning">
22                  <div class="panel-heading">
23                    @if( Auth::check() )
24                    <h2 class="text-success"> {{ Auth::user()->
                         name}}</h2>
25                    @endif
26
27                  </div>
28                  <div class="panel-body">
29                  <table class="table table-striped table-hover">
30                        <tr>
31                          <th></th>
32                          <th class="text-primary" style="text-
                               align:center;">Aluno</th>
33                          <th class="text-primary" style="text-
                               align:center;">Arquivo</th>
34                          <th class="text-primary" style="text-
                               align:center;">Data</th>
35
36                        </tr>
37                        @foreach ($files as $file)
38                          <tr height="30">
39                            <td ><i class="fa fa-2x fa-user"></i
                                 ></td>
40                            <td class="text-warning">{{$file->
                                 user_name}}</td>
41                            <td class="text-warning">{{$file->file
                                 }}</td>
42                            <td class="text-warning">{{$file->
                                 updated_at}}</td>
43                            <td ></td>
44                          </tr>
45                        @endforeach
46                      </table>
47
48              </div>
49          </div>
```

```
50              </div>
51          </div>
52      </div>
53  @endsection
```