# 4 Desenvolvimento da ferramenta

Visando facilitar a tarefa de documentar requisitos funcionais e de gerar testes automáticos em uma única ferramenta para proporcionar mais agilidade ao desenvolvimento de software, pesquisamos e verificamos na literatura que o caso de uso é uma boa forma para documentar as funcionalidades do software, é fácil de escrever e de ser entendido por pessoas com pouco conhecimento técnico, além de ser uma boa fonte para gerar casos de teste.

Pensamos então em especificar, projetar, desenvolver uma ferramenta para facilitar a documentação dos casos de uso, utilizando uma abordagem direcionada por comportamentos e um português restrito para descrever os fluxos de eventos (principal e secundários) para que seja possível integrar com um *framework* de testes e executar automaticamente testes funcionais em aplicações web e verificar se o comportamento está conforme descrito.

Neste capítulo serão apresentados os requisitos e a arquitetura necessários para que a ferramenta possa ajudar a minimizar o problema da falta de documentação e a dificuldade em automatizar testes, ambos observados e já discutidos anteriormente em capítulos anteriores neste documento.



## 4.1.Requisitos funcionais

O processo pensado para documentar casos de uso e gerar e executar testes automaticamente com informações dessa documentação através da ferramenta consiste em implementar três funcionalidades principais: O cadastro das informações do caso de uso, a geração e execução dos scripts de testes e a visualização do resultado. Abaixo a descrição dos requisitos desejados para que a ferramenta possa auxiliar na execução dessas atividades.

### 4.1.1.Cadastrar projetos

- O usuário informa que deseja cadastrar um projeto;
- A ferramenta deve exibir o formulário para cadastrar o projeto;

- O usuário preenche o formulário com os dados para criar o projeto e confirma o cadastro;
- A ferramenta deve cadastrar os dados do projeto, retornar para lista de projetos, conforme o requisito "visualizar projetos" e exibir uma mensagem de sucesso.

### 4.1.2.Visualizar projetos

- O usuário informa que deseja visualizar os projetos cadastrados;
- O ferramenta deve exibir uma lista dos projetos cadastrados e para cada projeto as seguintes informações: nome, descrição, quantidade de casos, data de criação, data de alteração e se o projeto está ativo ou inativo para gerar e executar automaticamente os scripts de teste.

### 4.1.3.Filtrar projetos

- Na lista de projetos descrita no requisito acima, "visualizar projetos", o usuário informa o nome ou parte do nome do projeto e solicita a pesquisa;
- A ferramenta deve pesquisar e apresentar uma lista dos projetos encontrados no mesmo formato da lista descrita no requisito funcional "visualizar projetos".

### 4.1.4.Alterar projetos

- O usuário informa o projeto que deseja alterar;
- A ferramenta deve apresentar o formulário com os dados do projeto;
- O usuário altera os dados do projeto e confirma a alteração;
- A ferramenta deve salvar as alterações realizadas nos dados no projeto, retornar para lista de projetos, conforme descrito no requisito funcional "visualizar projetos", e exibir uma mensagem de sucesso.

### 4.1.5.Cadastrar casos de uso

- O usuário informa que deseja cadastrar um caso de uso para um determinado projeto;
- A ferramenta deve exibir o formulário para cadastrar o caso de uso;
- O usuário informa os dados e confirma o cadastro;
- A ferramenta deve cadastrar o caso de uso para o projeto informado, retornar para lista de casos de uso, conforme o requisito "visualizar casos de uso" e exibir uma mensagem de sucesso.

### 4.1.6.Visualizar casos de uso

- O usuário informa que deseja visualizar os casos de uso cadastrados para um determinado projeto;
- O ferramenta deve exibir uma lista dos casos de uso cadastrados para o projeto e para cada caso de uso apresentar as seguintes informações: nome, quantidade de fluxos de eventos, quantidade de fluxos que geraram e executaram corretamente os scripts de teste, quantidade de fluxos que geraram e executaram incorretamente os scripts de teste, data de alteração e se o caso de uso esta ativo ou inativo para gerar e executar automaticamente os scripts de teste.

### 4.1.7.Filtrar casos de uso

- Na lista de casos de uso descrita no requisito acima, "visualizar casos de uso", o usuário informa o projeto e o nome ou parte do nome do caso de uso e solicita a pesquisa;
- A ferramenta deve pesquisar e apresentar uma lista dos casos de uso encontrados para o projeto informado e nome informados no mesmo formato da lista descrita no requisito funcional "visualizar casos de uso".

### 4.1.8.Alterar casos de uso

- O usuário informa o caso de uso do projeto que deseja alterar;

- A ferramenta deve apresentar o formulário com os dados do caso de uso;
- O usuário altera os dados do caso de uso e confirma a alteração;
- A ferramenta deve salvar as alterações nos dados no caso de uso para o projeto selecionado, retornar para lista de casos de uso, conforme descrito no requisito funcional "visualizar casos de uso", e exibir uma mensagem de sucesso.

### 4.1.9.Gerar e executar automaticamente scripts de teste para um determinado projeto

- O usuário solicita a geração e execução automática dos scripts de teste de um determinado projeto.
- A ferramenta deve iniciar o servidor (*Selenium-Server*) e realizar as seguintes operações para cada caso de uso cadastrado no projeto com status "ativo":
    - buscar o fluxo básico e o(s) fluxo(s) s alternativo(s);
    - Para cada fluxo, buscar os passos;
    - Para cada passo, gerar um script de teste, integrar ao framework de testes (*Selenium-Client*) e executar a interação com a interface web;
    - Armazenar o resultado da execução de cada script;
    - Finalizar o servidor (*Selenium-Server*).

.

### 4.1.10.Gerar e executar automaticamente scripts de teste para todos os projetos

- O usuário solicita a geração e execução automática dos scripts de teste para todos os casos de uso de todos os projetos cadastrados.
- A ferramenta deve iniciar o servidor (*Selenium-Server*) e realizar as seguintes operações para cada projeto cadastrado com status "ativo:
    - buscar os casos de uso também com status "ativo" para o processo de geração e execução automática de scripts de teste;

- Para cada caso de uso, buscar o fluxo básico e os(s) fluxo(s) alternativo(s);
- Para cada fluxo, buscar os passos;
- Para cada passo, gerar um script de teste, integrar ao framework de testes (*Selenium-Client*) e executar a interação com a interface web;
- Armazenar o resultado da execução de cada script;
- Finalizar o servidor (*Selenium-Server*).

### 4.1.11. Visualizar o resultado da geração e execução dos testes

- O usuário solicita visualizar o resultado da geração e execução dos scripts de teste executados através de um dos requisitos funcionais "gerar e executar automaticamente scripts de teste para um determinado projeto" ou "gerar e executar automaticamente scripts de teste para todos os projetos";
- A ferramenta deve apresentar o resultado da geração e execução dos scripts de teste através de 3 visões, afim de facilitar a identificação de possíveis falhas ocorridas durante o processo:
  - **Visão do geral** – listar todos os projetos cadastrados e as seguintes informações para cada projeto: nome, quantidade de casos de uso, quantidade de fluxos de eventos, quantidade de fluxos que geraram e executaram corretamente os scripts de teste, quantidade de fluxos que geraram e executaram incorretamente os scripts de teste e data da última execução.
  - **Visão do projeto** – listar os casos de uso de um projeto, conforme descrito no requisito funcional "visualizar casos de uso".
  - **Visão do caso de uso** – mesma visão descrita no requisito funcional "alterar casos de uso" com a situação da execução de cada passo.

## 4.2.Requisitos não funcionais

Abaixo a lista dos requisitos não funcionais pensados para o desenvolvimento da ferramenta:

- A ferramenta deve ser uma aplicação web para facilitar a distribuição e ser disponibilizada na intranet da empresa durante a utilização do experimento.
- Deve ter uma interface simples e de fácil uso para permitir agilidade no cadastro da documentação.
- Deve suportar o acesso simultâneo de pelo menos três usuários dos seis usuários que vão interagir com a ferramenta, dentre eles: quatro desenvolvedores, um coordenador e um cliente. Ambos fazem parte de uma equipe de desenvolvimento ágil e têm conhecimento avançado de internet;
- Ser compatível com os seguintes sistemas:
  - *Linux* - distribuição *UBUNTU* – versão 9 ou superior;
  - *Windows* - versão VISTA ou superior;
  - *Mac OS X* – versão 10 ou superior;
- Ser compatível com os seguintes navegadores e:
  - *Mozilla Firefox* – versão 3 ou superior;
  - *Safari* – versão 4 ou superior;
  - *Internet Explorer* – Versão 7 ou superior;
- Deve utilizar apenas tecnologias de softwares livre, tais como:
  - Banco de dados;
  - Linguagens e *frameworks* de programação;
  - Servidores de aplicação;
  - Ferramentas de desenvolvimento;
  - Gerenciadores de versão;
  - Ferramentas de *build* e *deploy*;

## 4.3.Diagrama de casos de uso

A figura 14 abaixo ilustra a interação do usuário com os casos de uso e relação entre os casos de uso. Como todos os atores interessados podem interagir

com todos os casos de uso do sistema, esses atores foram generalizados em um ator “Usuário”.

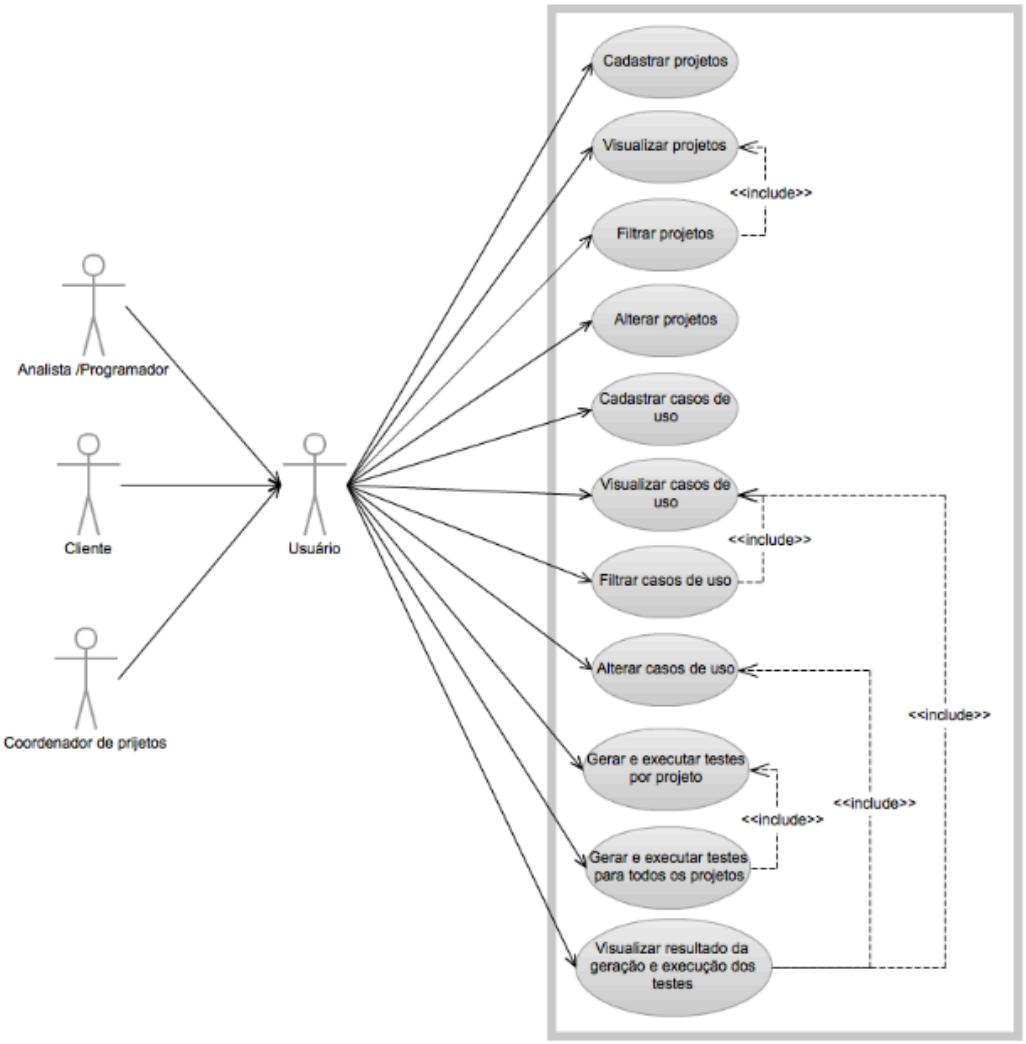


**Figura 14 – Diagrama dos casos de uso da ferramenta.**

Na relação entre os casos de uso, temos:

- “filtrar projetos” utiliza o caso de caso de uso “visualizar projetos” para apresentar o resultado do filtro realizado através do nome do projeto.
- “Filtrar casos de uso” utiliza o caso de uso “visualizar casos de uso” para apresentar o resultado do filtro realizado através do nome do caso de uso.
- “Gerar e executar testes para todos os projetos” utiliza o caso de uso “Gerar e executar testes por projeto” para realizar a gerar e executar os testes em todos os projetos cadastrados.

- "Visualizar resultado da geração e execução dos testes" utiliza o caso de uso "visualizar casos de uso" para apresentar o resultado dos testes por casos de uso do projeto e também utiliza o caso de uso "alterar casos de uso" para apresentar o resultado de cada passo dos fluxos do caso de uso.

## 4.4.Modelo de dados

A figura 15 abaixo apresenta o modelo de dados desenhado para suportar a ferramenta no armazenamento das informações do caso de uso e na extração das informações para montar os scripts de teste.

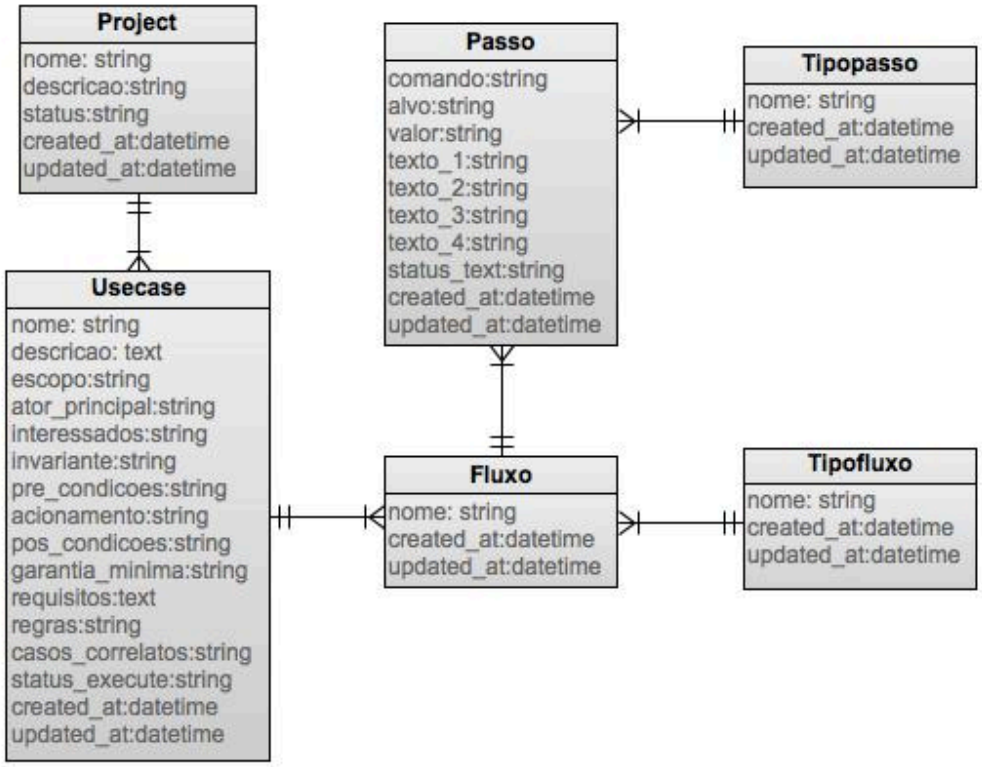


**Figura 15 – Modelo de dados da ferramenta.**

## 4.5.Persistência dos dados

Conforme mencionado anteriormente, no capítulo 3, no item 3.2, a ferramenta utiliza a arquitetura *ActiveRecord* (Flower, 2003), para armazenar os dados em um banco de dados relacional. Nesta arquitetura, uma tabela do banco

de dados é representada por uma classe e uma instância dessa classe (objeto) está vinculada a uma única linha na tabela. Após a criação de um objeto, uma nova linha é adicionada à tabela e qualquer objeto carregado obtém suas informações do banco de dados. Quando um objeto é atualizado a linha correspondente na tabela também é atualizada.

O fluxograma da figura 16 abaixo ilustra os passos para cadastrar um caso de uso através da ferramenta. O primeiro passo é cadastrar o projeto no qual o caso de uso será inserido, isso porque a organização dos casos de uso é feita através dos projetos.

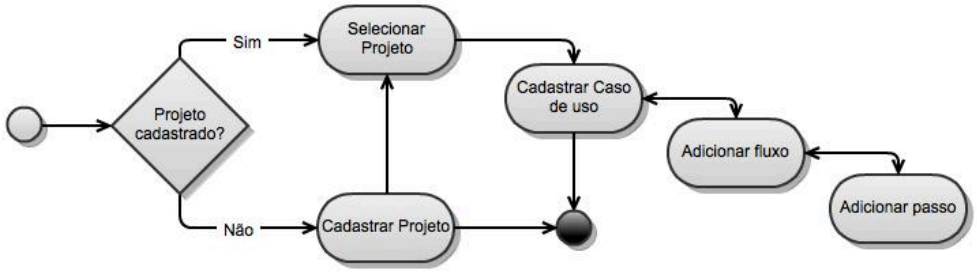


**Figura 16 – Fluxo para cadastrar um caso de uso.**

O cadastro do projeto deve ser realizado pela ferramenta através da funcionalidade “cadastrar projetos”, conforme descrita nos requisitos funcionais. Ao finalizar o cadastro, os dados são armazenados na tabela “Project” apresentada no modelo de dados da figura 15. Vale ressaltar que o atributo “status” do projeto, pode ter apenas os dois valores: “S” ou “N“. Esse atributo diz respeito ao status do projeto no processo de execução dos testes. Caso tenha o valor “S”, o projeto será utilizado na geração e execução de testes. Caso contrário, não será utilizado no processo. A figura 17 abaixo apresenta um exemplo do formulário para cadastrar projeto de um projeto já cadastrado.

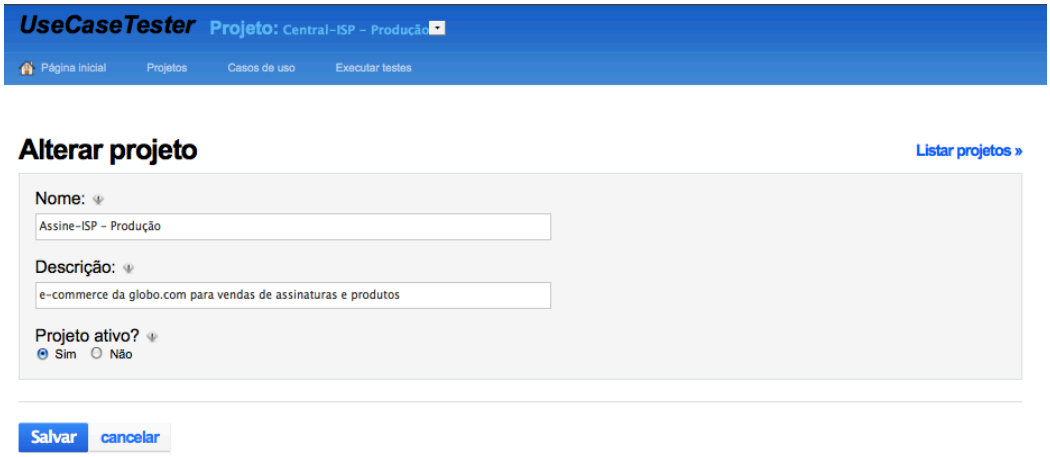


**Figura 17 – Formulário para cadastrar um projeto na ferramenta.**

Após cadastrar o projeto pode ser iniciado o cadastro dos casos de uso. O cadastro do caso de uso deve ser realizado através da funcionalidade "cadastrar casos de uso", descrita como requerimento funcional neste capítulo. O formulário para cadastrar caso de uso é responsável por manipular os dados em três tabelas: "Usecase", "Fluxos" e "Passos", ou seja, ao cadastrar um caso de uso, os dados são distribuídos nessas três tabelas. As figura 18 e figura 19 abaixo apresentam um exemplo do formulário de um caso de uso já cadastrado para o projeto "Central-ISP – Produção", selecionado no menu superior do formulário, e como os dados foram distribuídos nas tabelas do modelo.

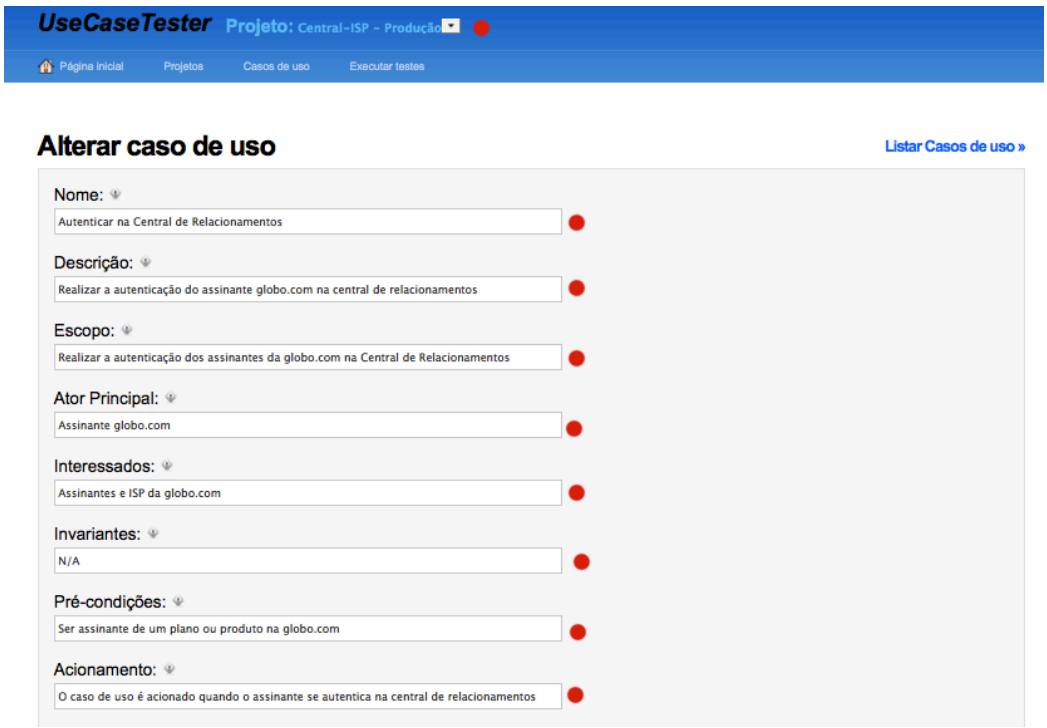


**Figura 18 – Formulário para redação do caso de uso – 1 de 2.**

As informações assinaladas com um círculo na cor vermelho estão armazenadas na tabela “Usecase”, as assinaladas com a cor amarelo estão armazenadas na tabela de “Fluxos” e as assinaladas na cor preto estão armazenadas na tabela de “Passos”. Assim como o projeto, o caso de uso tem uma propriedade “status_execute” que pode ter apenas dois valores: “S” ou “N“. Esse atributo diz respeito ao status do caso de uso para execução dos testes. Caso tenha o valor “S”, o caso de uso será utilizado na geração e execução de testes. Caso contrário, não será utilizado no processo.

Os fluxos do caso de uso podem ser de dois tipos: Fluxo principal e Fluxo alternativo. Um caso de uso deve ter um fluxo principal e pode ter quantos fluxos alternativos forem necessários. Os fluxos de eventos podem receber o cadastro de três tipos de passos para descrever seu comportamento, são eles: “comentário”, “ação” e “verificação”.

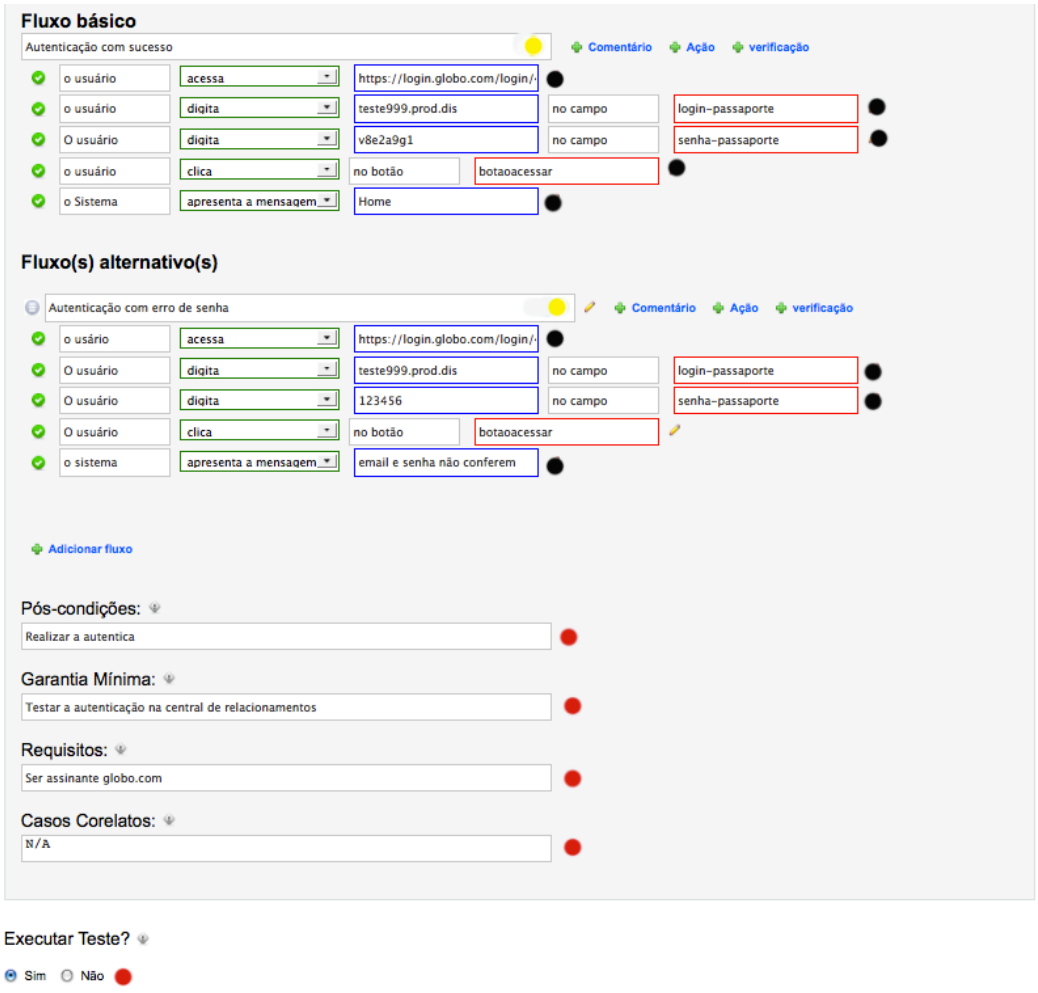


**Figura 19 – Formulário para redação do caso de uso – 1 de 2.**

**Comentário** – o passo comentário, conforme ilustrado abaixo na figura 20 é

um campo texto livre para digitação e pode ser utilizado para enriquecer a documentação do fluxo. Passos desse tipo não tem utilidade no processo de geração dos testes, apenas complementam a documentação.


**Figura 20 – Passo do tipo comentário.**

**Ação** – cada passo do tipo ação cadastrado no fluxo de eventos será responsável por realizar uma interação com o formulário web através da integração com o *framework* de testes *Selenium*. A figura 21 abaixo ilustra a inclusão desse tipo de passo dentro do fluxo e também exibe a lista de opções com os tipos de ações disponíveis.

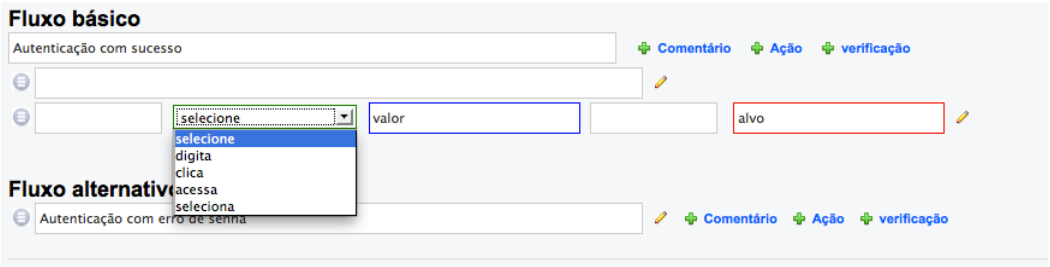


**Figura 21 – Passo do tipo "ação".**

A configuração dos campos do passo do tipo ação pode ser alterada dependendo do tipo de ação escolhida para o passo na lista de seleção. O exemplo da figura 22 abaixo apresenta a configuração do passo para cada tipo de ação disponível na lista de seleção apresentada na figura 21.


**Figura 22 – Diferentes opções do passo "ação".**

**Verificação** – esse tipo de passo é responsável por verificar, através da integração com o *framework* de testes *Selenium*, se o texto cadastrado no passo existe no formulário web da aplicação. Isso possibilita que a ferramenta verifique as mensagens apresentadas pela aplicação web e consiga verificar se o fluxo atingiu o objetivo esperado. Abaixo na figura 23 um exemplo da inclusão desse

tipo de passo em um fluxo.

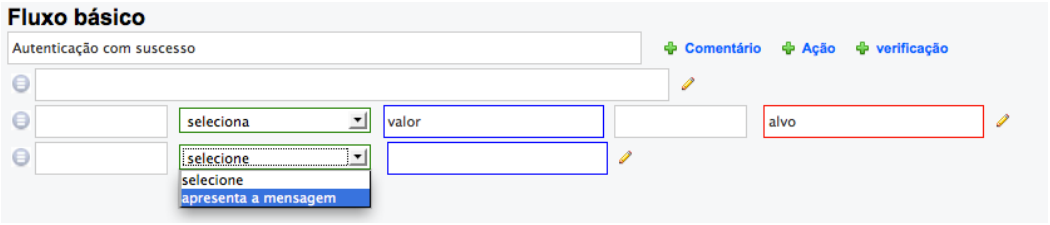


**Figura 23 – Passo do tipo "verificação".**

Para os três tipos de passos apresentados (documentação, ação e verificação), assim como na manutenção de fluxos, os passos incluídos no fluxo podem ser excluídos e também reordenados dentro do próprio fluxo através do recurso *drag-and-drop* (arrastar e largar) implementado no formulário.

A figura 24, a figura 25, a figura 26 e a figura 27 abaixo ilustram os dados armazenados nas tabelas do banco de dados após realizar o fluxo completo de cadastro de um caso de uso. Na figura 24 estão as informações do projeto, na figura 25 as informações do caso de uso, na figura 26 dos fluxos e na figura 27 as informações dos passos.

| id | nome | descricao | status | created_at | updated_at |
|---|---|---|---|---|---|
| 3 | Central-ISP - Produção | Central de relacionamentos para o assinante globo.com em Produção | S | 2011-03-23... | 2011-06-10... |

**Figura 24 – Projeto armazenado na tabela "Projects".**

| id | project_id | nome | descricao | escopo | ator_principal | interessados | invariante | pre_condicoes | acionamento |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 3 | Autenticar na Central de Relacionamentos | Realizar a autenti... | Realizar a... | Assinante glob... | Assinantes e IS... | N/A | Ser assinante... | O caso de uso é... |

**Figura 25 – Caso de uso armazenados na tabela "Usecase".**

| id | nome | usecase_id | tipofluxo_id | created_at | updated_at |
|---|---|---|---|---|---|
| 1068 | Autenticação com sucesso | 16 | 1 | 2011-06-13... | 2011-06-13... |
| 1069 | Autenticação com erro de senha | 16 | 2 | 2011-06-13... | 2011-06-13... |

**Figura 26 – Fluxos armazenados na tabela "Fluxos"**

| id | fluxo_id | tipopasso_id | texto_1 | comando | valor | texto_2 | alvo | status_test | created_at | updated_at | texto_3 | texto_4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9888 | 1086 | 2 | o usuário | acessa | https://login.globo.com/login/464 | | | S | 2011-06-29... | 2011-06-29... | | |
| 9889 | 1086 | 2 | o usuário | type | teste999.prod.dis | no campo | login-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9890 | 1086 | 2 | O usuário | type | v8e2a9g1 | no campo | senha-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9891 | 1086 | 2 | o usuário | click | | no botão | botaoacessar | S | 2011-06-29... | 2011-06-29... | | |
| 9892 | 1086 | 3 | o Sistema | verifica | Home | | | S | 2011-06-29... | 2011-06-29... | | |
| 9893 | 1087 | 2 | o usário | acessa | https://login.globo.com/login/464 | | | S | 2011-06-29... | 2011-06-29... | | |
| 9894 | 1087 | 2 | O usuário | type | teste999.prod.dis | no campo | login-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9895 | 1087 | 2 | O usuário | type | 123456 | no campo | senha-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9896 | 1087 | 2 | O usuário | click | | no botão | botaoacessar | S | 2011-06-29... | 2011-06-29... | | |
| 9897 | 1087 | 3 | o sistema | verifica | email e senha não conferem | | | S | 2011-06-29... | 2011-06-29... | | |

**Figura 27 – Passos armazenados na tabela "Passos" .**

## 4.6.Gerador e executor de scripts de teste

Após realizar o cadastro dos casos de uso a ferramenta pode gerar os scripts e integrar ao *framework* de testes *Selenium* para executar a interação com a interface web e assim verificar o comportamento da aplicação web. O fluxograma

da figura 28 abaixo é um retrato do código implementado para gerar e executar os scripts de testes. As atividades em branco são as atividades que a ferramenta faz a integração com o *framework Selenium* para manipular o *Selenium-Server* e o *Selenium-Client*.

Quando inicia a geração e execução dos testes a primeira atividade executada pela ferramenta é iniciar *Selenium-Server*, essa atividade inicializa a parte servidora do *framework* para receber as futuras requisições *HTTP's* provindas dos scripts que serão gerados pela ferramenta e executados através dos *Selenium-Client*. O *Selenium-Server* é mantido ativo até que todos os testes sejam executados, após terminar é encerrado.

Após inicializar o *Selenium-Server* a ferramenta verifica se a solicitação para gerar e executar os testes foi feita para um projeto ou para todos os projetos. Quando a solicitação é feita para executar testes em todos os projetos a ferramenta faz a busca no banco de dados pelos projetos cadastrados com atributo "status=S" e então realiza o mesmo fluxo para cada projeto.

Em seguida a ferramenta busca pelos casos de uso do projeto com atributo "status_execute=S". Para cada caso de uso é iniciada uma instância do navegador através do *Selenium-Client* e a instância é mantida até que todos os fluxos do caso de uso sejam executados.

O próximo passo é buscar os fluxos do caso de uso e para cada fluxo criar uma nova sessão do navegador que foi iniciado no passo anterior. A sessão é mantida até que todos os passos do fluxo sejam executados. Criar uma nova sessão do navegador para cada fluxo garante que não haja interferência de dados armazenados na sessão do navegador pelo fluxo anterior.

Por fim, a ferramenta busca os passos do fluxo e para cada passo do fluxo monta um script de teste e integra ao *selenium-client* para que o mesmo possa executar uma interação com a interface web.

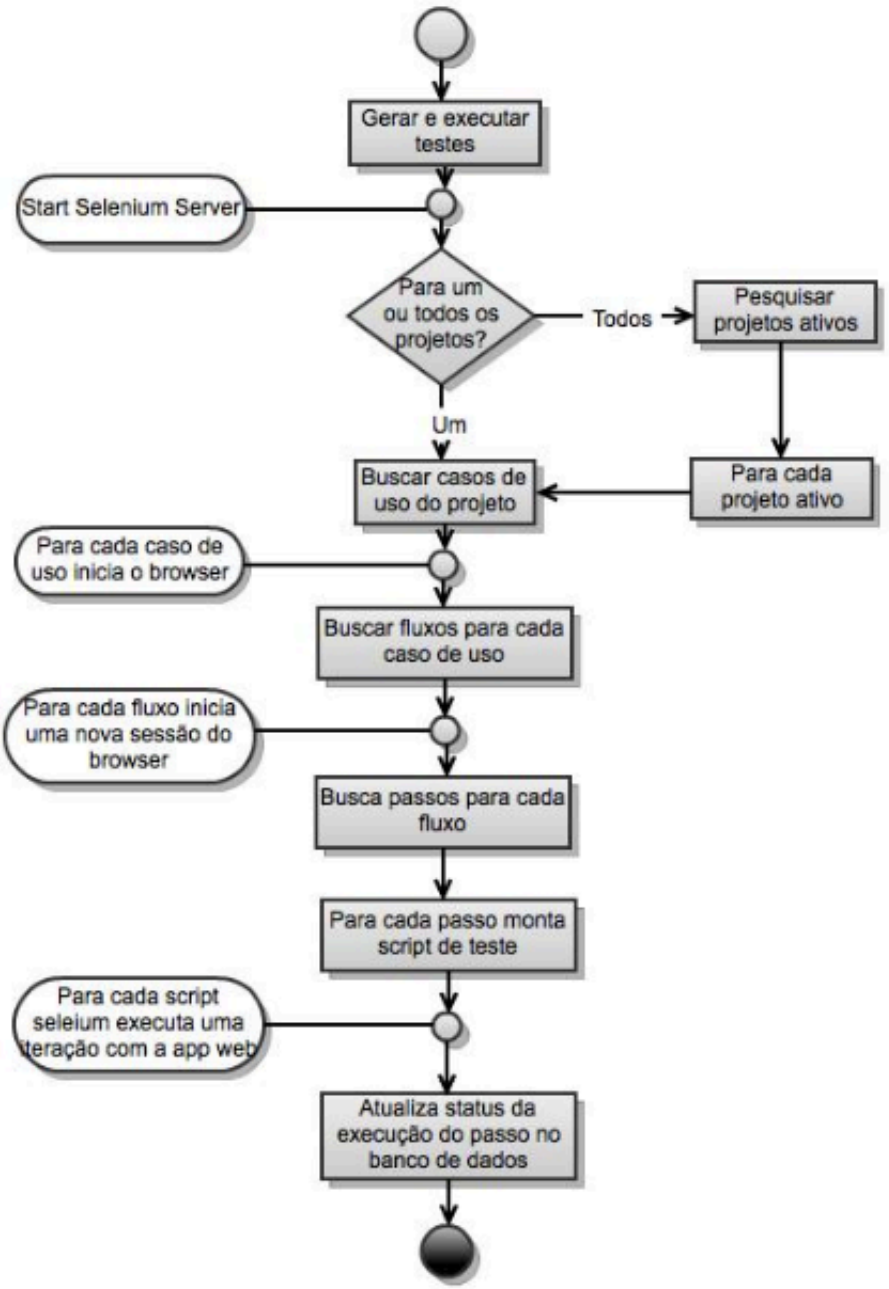


**Figura 28 – Fluxo para gerar e executar automaticamente os testes.**

Abaixo um exemplo de como a ferramenta utiliza as informações da tabela "passos", ilustrados na figura 29 para montar os scripts no padrão do *Selenium-Client*.

| id | fluxo_id | tipopasso_id | texto_1 | comando | valor | texto_2 | alvo | status_test | created_at | updated_at | texto_3 | texto_4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9888 | 1086 | 2 | o usuário | acessa | https://login.globo.com/login/464 | | | S | 2011-06-29... | 2011-06-29... | | |
| 9889 | 1086 | 2 | o usuário | type | teste999.prod.dis | no campo | login-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9890 | 1086 | 2 | O usuário | type | v8e2a9g1 | no campo | senha-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9891 | 1086 | 2 | o usuário | click | | no botão | botaoacessar | S | 2011-06-29... | 2011-06-29... | | |
| 9892 | 1086 | 3 | o Sistema | verifica | Home | | | S | 2011-06-29... | 2011-06-29... | | |
| 9893 | 1087 | 2 | o usário | acessa | https://login.globo.com/login/464 | | | S | 2011-06-29... | 2011-06-29... | | |
| 9894 | 1087 | 2 | O usuário | type | teste999.prod.dis | no campo | login-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9895 | 1087 | 2 | O usuário | type | 123456 | no campo | senha-passaporte | S | 2011-06-29... | 2011-06-29... | | |
| 9896 | 1087 | 2 | O usuário | click | | no botão | botaoacessar | S | 2011-06-29... | 2011-06-29... | | |
| 9897 | 1087 | 3 | o sistema | verifica | email e senha não conferem | | | S | 2011-06-29... | 2011-06-29... | | |

**Figura 29 – Tabela "Passos" com os registros utilizados no teste.**

Scripts gerados para o fluxo – Autenticar com sucesso:

- *Selenium.open(“https://login.globo.com/login/464”);*
- *Selenium.type(“login-passaporte”, “teste999.prod.dis”);*
- *Selenium.type(“senha-passaporte”, “123456”);*
- *Selenium.click(“botaoacessar”);*
- *Selenium.isTextPresent?(“Home”);*

Scripts gerados para o fluxo – Autenticar com erro de senha:

- *Selenium.open(“https://login.globo.com/login/464”);*
- *Selenium.type(“login-passaporte”, “teste999.prod.dis”);*
- *Selenium.type(“senha-passaporte”, “654321”);*
- *Selenium.click(“botaoacessar”);*
- *Selenium.isTextPresent?(“email ou senha não conferem”);*

Ao executar cada passo a ferramenta recebe um retorno do *Selenium-Client* indicando se o *script* gerado foi executado com sucesso ou não, caso o script tenha sido executado com sucesso a ferramenta atualiza a propriedade “status_test” do passo no banco de dados para “S”, caso contrário, para “N”. É através deste propriedade do passo que a ferramenta apresenta o resultado dos testes.

As figuras abaixo, entre a figura 30 e a figura 44 apresentam a sequência das telas e dos passos utilizados em cada tela para realizar execução da compra de um produto no e-commerce de um provedor de internet. A execução inicia com passo ilustrado na figura 30, utilizado para acessar a vitrine de produtos do e-commerce.

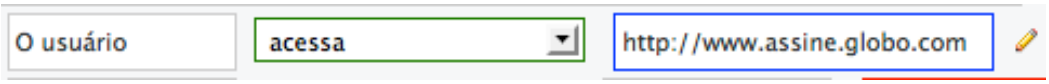


**Figura 30 – Passo do tipo “ação” para acessar a vitrine de produtos.**

Script gerado e utilizado na integração com o *Selenium*:

1. *Selenium.open*(“http://assine.globo.com”);

A figura 31 abaixo apresenta a vitrine de produtos carregada.


**Figura 31 – Tela da vitrine de produtos e serviços do e-commerce.**


**Figura 32 – Passo "ação" para selecionar o produto.**

Script gerado e utilizado na integração com o *Selenium*:

1. *Selenium.click*("//ul[@id='vitrine-container']/li[8]/div/div/a/img");

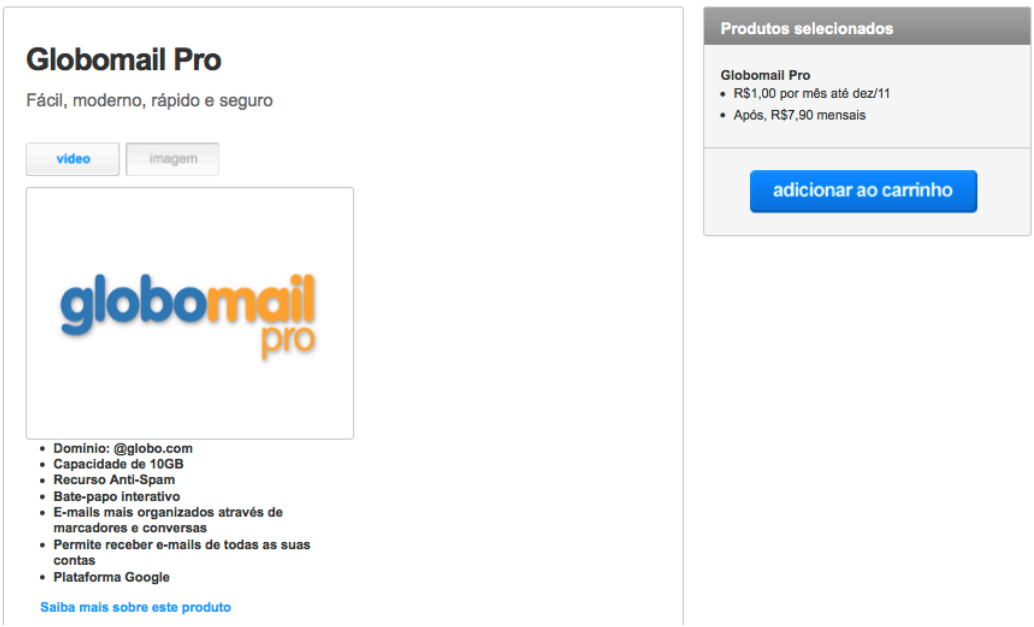

**Figura 33 – Tela de detalhe do produto selecionado na vitrine.**

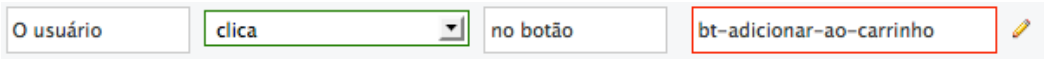


**Figura 34 – Passo "ação" para adicionar o produto no carrinho.**

Script gerado e utilizado na integração com o *Selenium*:

- *Selenium.click("bt-adicionar-ao-carrinho");*


**Figura 35 – Tela do carrinho de compras.**

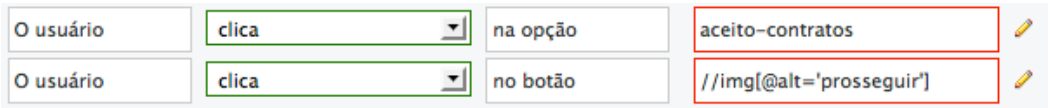


**Figura 36 – Passos do tipo "ação" utilizados no carrinho de compras.**

Scripts gerados e utilizados na integração com o *Selenium*:

- *Selenium.click("aceito-contratos");*
- *Selenium.click("//img[@alt='prosseguir']");*

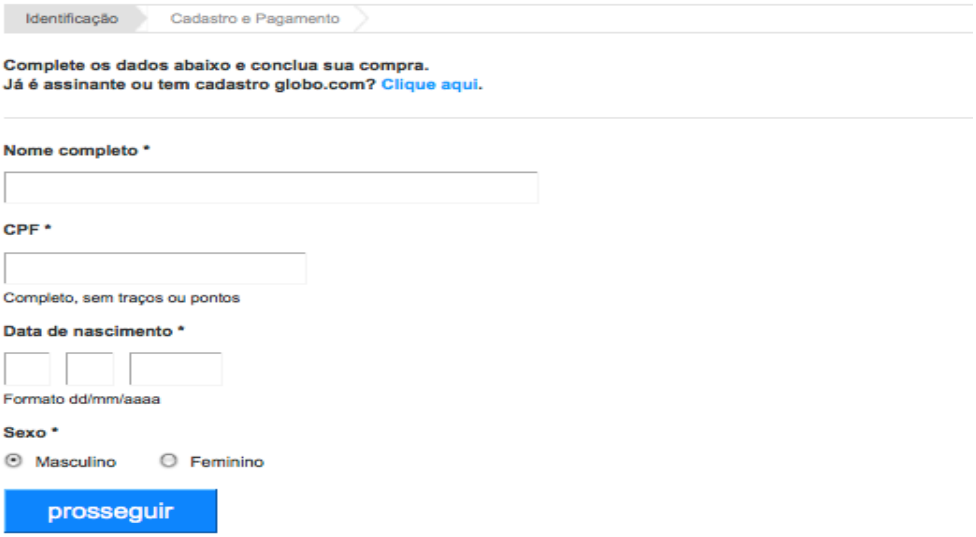


**Figura 37 – Tela para informar os dados de identificação.**

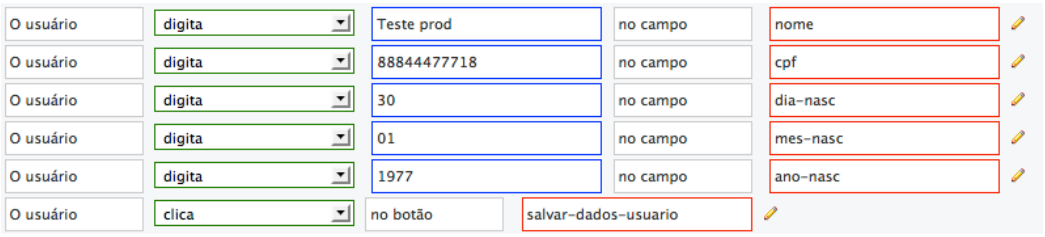


**Figura 38 – Passos do tipo "ação" utilizados na tela de identificação.**

Scripts gerados e utilizados na integração com o *Selenium*:

- *Selenium.type("nome", "Teste prod");*
- *Selenium.type("cpf", "88844477718");*
- *Selenium.type("dia-nasc","30");*
- *Selenium.type("mes-nasc", "01");*
- *Selenium.type("ano-nasc", "1977");*
- *Selenium.click("salvar-dados-usuario");*

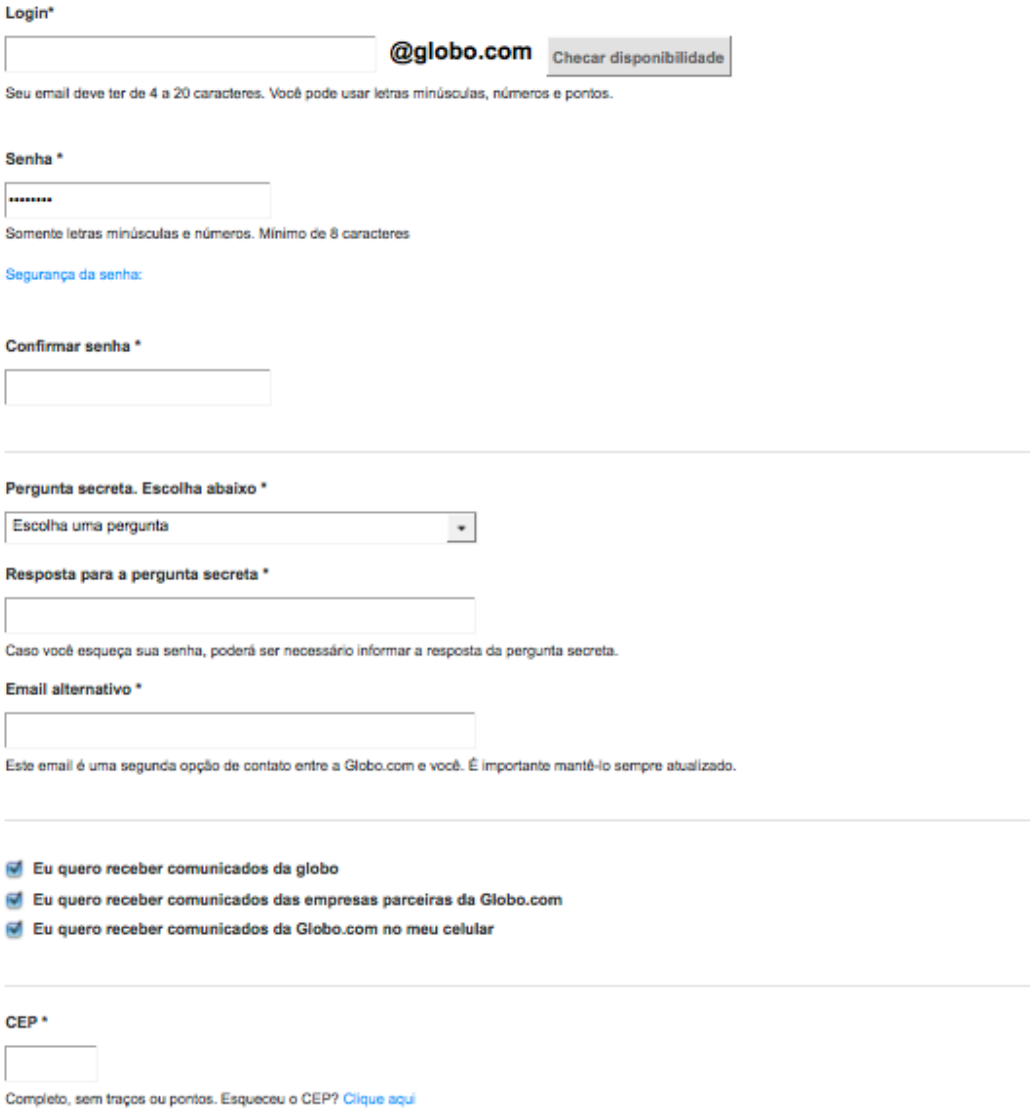


**Figura 39 – Tela para informar os dados de cadastro – 1 de 2.**


**Figura 40 – Passos tipo "ação" utilizados no tela de cadastro 1 de 2.**

Scripts gerados e utilizados na integração com o *Selenium*:

- *Selenium.type("login", "teste999.prod1307465669");*
- *Selenium.type("nova-senha", "1q2w3e4r");*
- *Selenium.type("confirmarSenha", "1q2w3e4r");*

- *Selenium.select("perguntaSecretaId", "label=Qual a sua cor favorita?");*
- *Selenium.type("respostaSecreta", "meu cachorro");*
- *Selenium.type("email-alternativo", "teste@tewte.com");*
- *Selenium.type("cep", "22061060");*

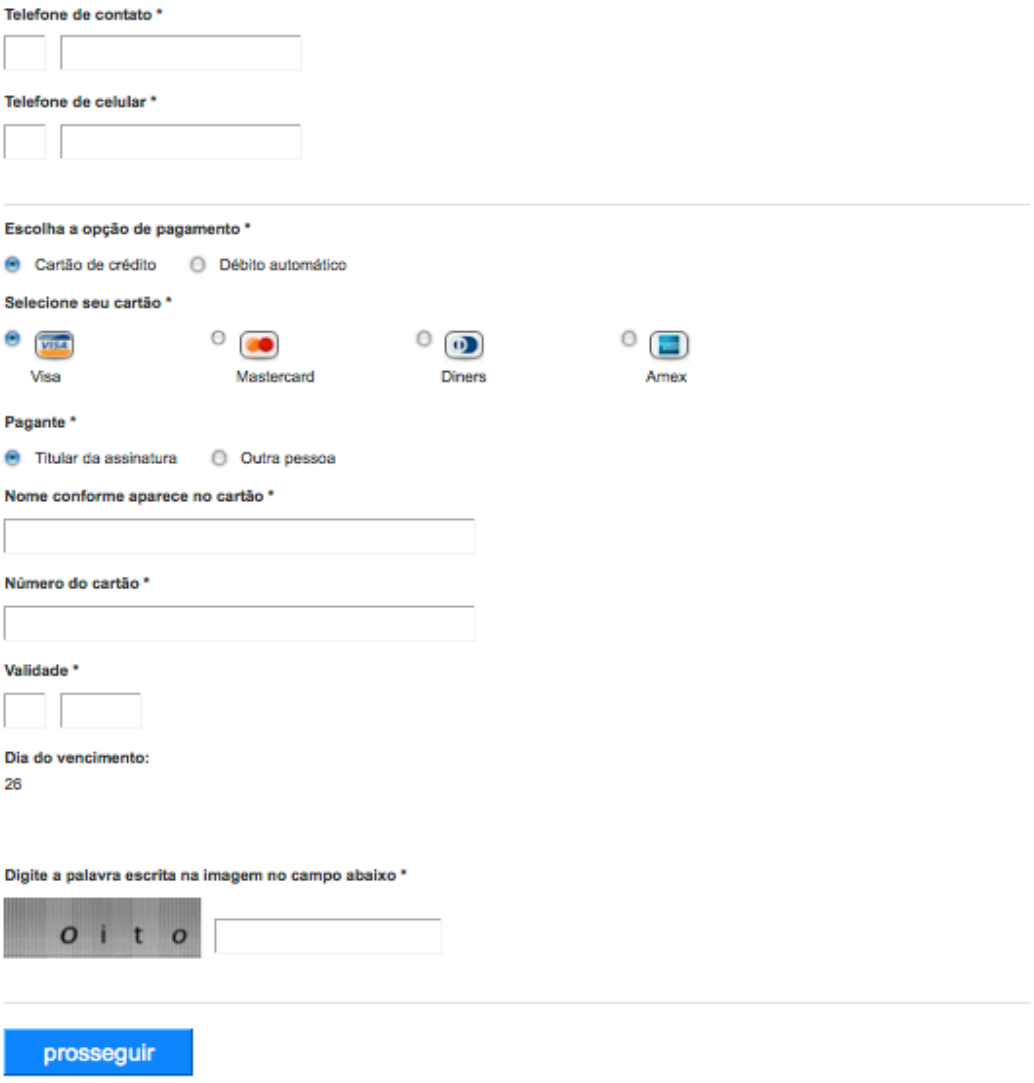


**Figura 41 – Tela para informar os dados de cadastro – 2 de 2.**

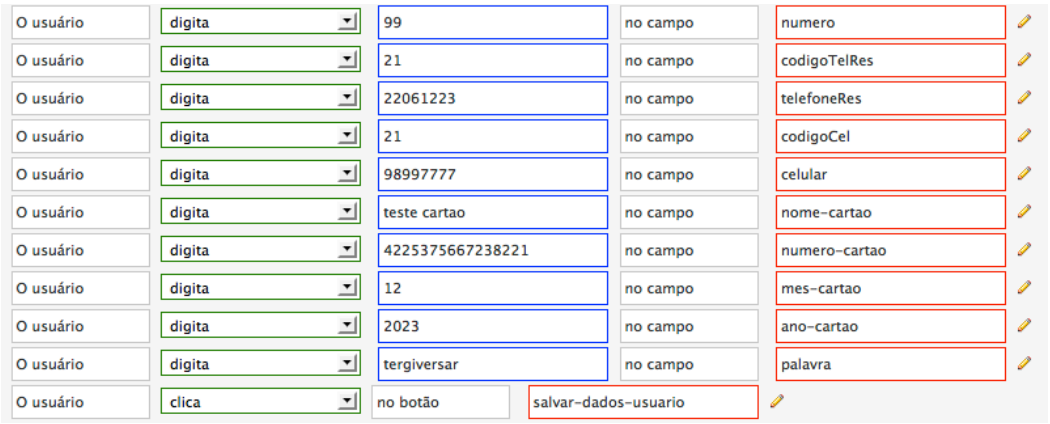


**Figura 42 – Passos tipo "ação" utilizados no tela de cadastro 1 de 2.**

Scripts gerados e utilizados na integração com o *Selenium*:

- *Selenium.type("numero", "99");*
- *Selenium.type("codigoTelRes", "21");*
- *Selenium.type("telefoneRes", "22061223");*
- *Selenium.type("codigoCel", "21");*
- *Selenium.type("telefoneCel", "98997777");*
- *Selenium.type("nome-cartao", "4225375667238221");*
- *Selenium.type("numero-cartao", "4225375667238221");*
- *Selenium.type("mes-cartao", "12");*
- *Selenium.type("ano-cartao", "2023");*
- *Selenium.click("salvar-dados-usuario");*

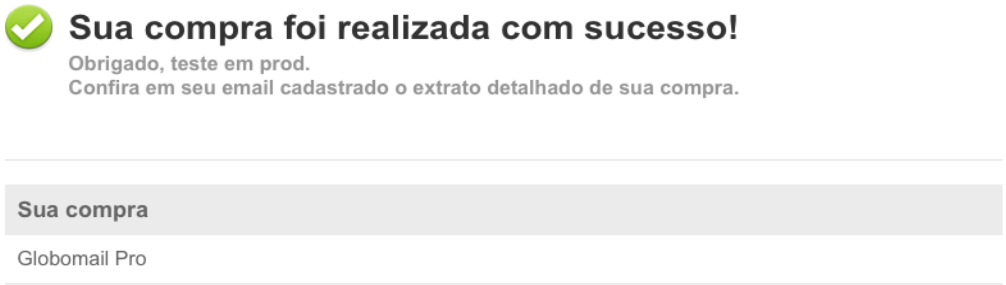


**Figura 43 – Tela de sucesso na realização da compra.**

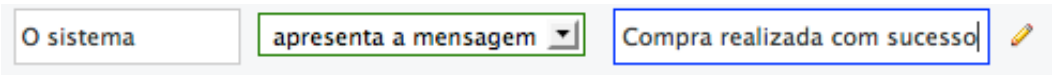


**Figura 44 – Passo do tipo "verificação" utilizado na tela de sucesso.**

Script gerado e utilizado na integração com o *Selenium*:

- *Selenium.isTextPresent?("Compra realizada com sucesso")*

## 4.7. Resultado da execução dos testes

O resultado da geração e execução dos testes é extraído da informação do atributo “status_test” de cada passo e foi organizado para ser analisado através de três diferentes visões: visão geral, visão do projeto e visão do caso de uso.

Através da visão geral, localizada na página inicial da ferramenta, é possível verificar rapidamente pela lista de projetos, informações sobre a última execução dos testes para cada projeto da lista. A visão exibe as seguintes informações:

**Projeto**: nome do projeto.

**Caso(s) de uso**: quantidade de casos de uso do projeto.

**Qtde. de fluxos**: quantidade de fluxos do projeto.

**Fluxos válidos**: quantidade de fluxos do projeto que executaram os testes corretamente.

**Fluxos Inválidos**: quantidade de fluxo que não executaram os testes corretamente. Neste caso, se a quantidade de fluxos for maior que zero, será apresentado um ícone alertando que neste projeto existem fluxos que não executaram corretamente.

**Última execução**: data da última execução do dos testes.

Conforme apresentado no exemplo da figura 45, na última execução dos testes a ferramenta indicou falha no comportamento de dois fluxos de um projeto.

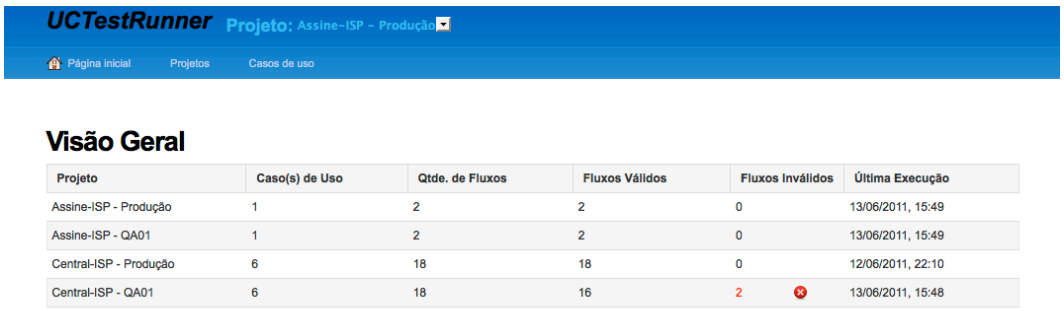


**Figura 45 – Visão geral do resultado por projetos.**

Na visão do projeto, localizada na lista de casos de uso do projeto é possível localizar os casos de uso que apresentaram falha. Conforme exemplo da figura 46 abaixo, dois fluxos de um caso de uso estão indicados como inválidos.

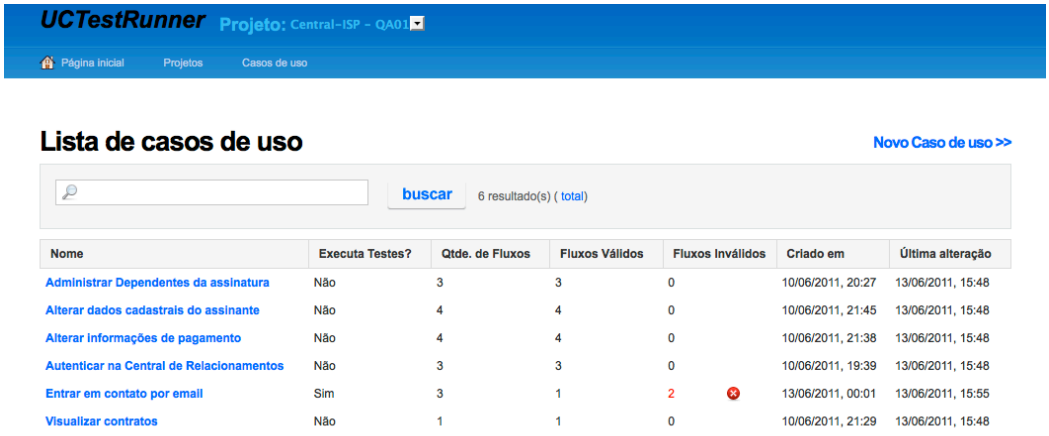


**Figura 46 – Visão do resultado por casos de uso do projeto.**

Na visão do caso de uso, localizada no formulário de cadastro e manutenção de casos de uso, é possível identificar os fluxos e o passo de cada fluxo com comportamento indesejado, conforme exemplo da figura 46 abaixo.

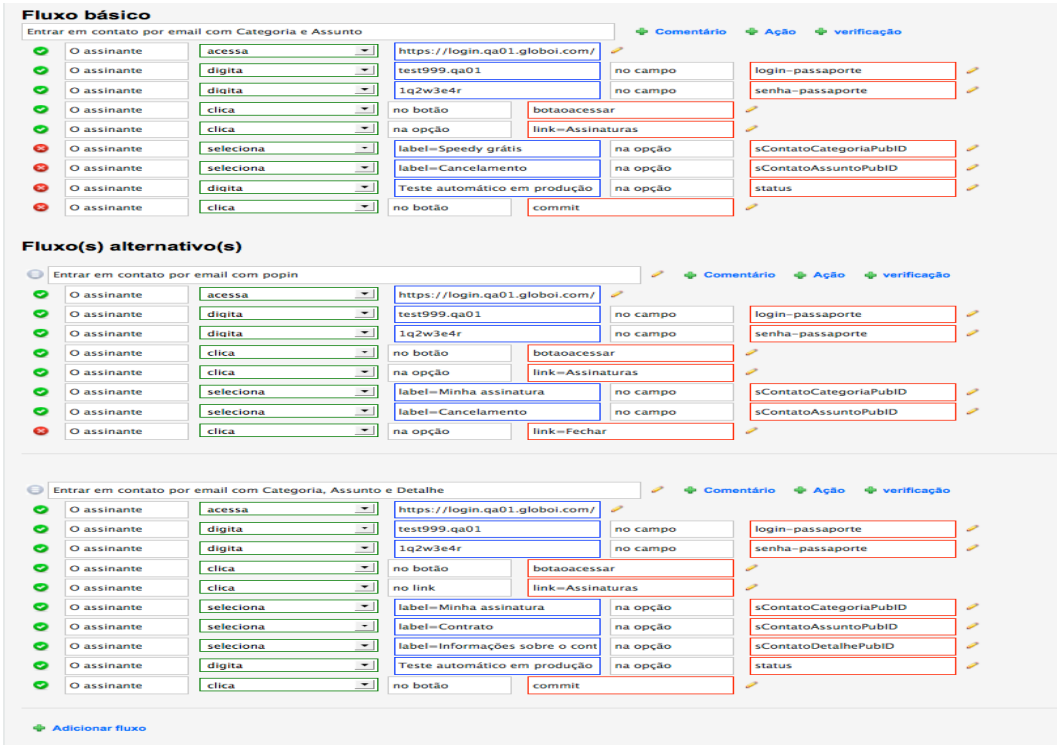


**Figura 47 – Visão do resultado no caso de uso.**

Ou seja, para verificar qual passo falhou na geração e execução automática dos testes, é preciso identificar o projeto na página inicial da ferramenta, acessar a lista de casos de uso desse projeto e abrir o formulário do caso de uso. Essas três visões facilitam a identificação e correção do problema.

## 4.8.Diagrama de implantação

O diagrama da figura 48 abaixo ilustra como a ferramenta será distribuída através de nós de hardware e as suas devidas relações de comunicação.

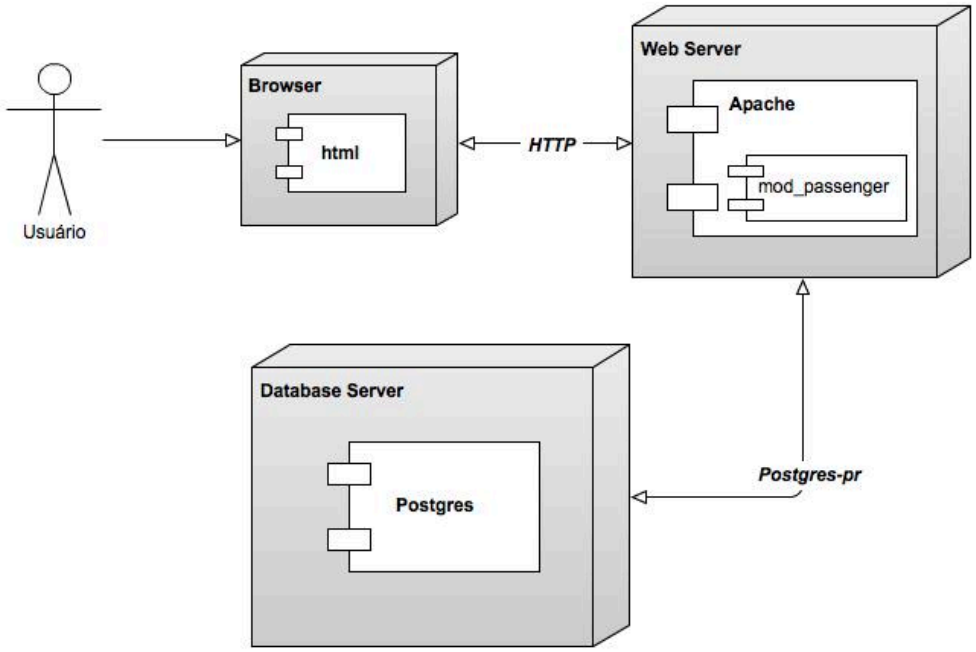


**Figura 48 – Diagrama de implantação da ferramenta.**

A ferramenta é uma aplicação web servida pelo ***mod_passenger***, um módulo do Web Server **Apache**, que recebe requisições HTTP através do navegador do usuário e verifica se o pedido deve ser tratado por uma aplicação ***Ruby on Rails***. Se sim, então o ***mod_passenger*** vai encaminhar o pedido para a aplicação que utiliza o driver ***postgres-pr*** para fazer a comunicação com o banco de dados ***Postgres*** para armazenar e/ou extrair dados e retornar com a resposta para que o navegador possa exibir as informações para o usuário.

## 4.9.Decisões de projeto

As tecnologias descritas abaixo foram utilizadas para projetar, desenvolver e suportar a ferramenta porque são tecnologias de código aberto, gratuitas, do conhecimento técnico do autor e por se adequarem as práticas utilizadas no contexto da empresa de onde será aplicado o experimento.

***Ruby on Rails*****:** *framework* de código aberto escrito na linguagem de

programação *Ruby* que promete aumentar velocidade e facilidade no desenvolvimento de sites orientados a banco de dados (*database-driven* web sites), uma vez que é possível criar aplicações com base em estruturas pré-definidas. As aplicações criadas utilizando o framework *Rails* são desenvolvidas com base no padrão de projeto MVC (*Model-View-Controller*).

***Aptana Studio*** – ambiente integrado de desenvolvimento de código aberto e gratuito desenvolvido em Java e suporta as linguagens CSS, HTML, pode ser configurado para suportar *Ruby on Rails* através do plug-in *Aptana RedRails*. É baseado no Eclipse, programa similar que por sua vez desenvolve linguagens de programação. É distribuído em múltiplas plataformas ou seja, pode ser utilizado no Windows, Linux e Mac OS X.

**CVS:** CVS ou *Concurrent Version System* (Sistema de Versões Concorrentes) é um sistema de controle de versão de código aberto e gratuito que permite que se trabalhe com diversas versões de arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos. É especialmente útil para se controlar versões de um software durante seu desenvolvimento, ou para composição colaborativa de um documento.

***PostgreSQL*:** é um sistema de gerenciamento de banco de dados (SGBD) que utiliza a linguagem SQL como interface. Desenvolvido como projeto de código aberto.

***Capistrano*:** ferramenta de código aberto e gratuita que permite rodar scripts em múltiplos servidores. Seu principal uso é para realização de *deploy* de aplicações WEB. Com ele é possível automatizar o processo de subida para produção de uma aplicação permitindo a execução de várias tarefas em múltiplos servidores.

***Selenium Remote Control* (RC):** é uma ferramenta que permite automatizar testes em aplicações web. É dividido em duas partes:

1. Um servidor que automaticamente inicia e encerra o navegador web, e atua como um *proxy HTTP* para solicitações web provindas desses navegadores.
2. Biblioteca cliente para montar e executar os scripts de teste. Disponível para diferentes linguagens de programação.