5 Estudo de caso

Neste capítulo serão apresentadas as aplicações web utilizadas na aplicação da abordagem proposta, bem como a tecnologia em que foram desenvolvidas, o escopo do experimento e a situação atual da documentação e dos testes. Também serão apresentados os resultados obtidos na aplicação do estudo de caso.

5.1.Visão geral das aplicações utilizadas nos testes

Para realizar o estudo de caso foram utilizadas duas aplicações web: o *Assine Já* e a *Central de Relacionamentos*.

O *Assine Já* é um web site desenvolvido para suportar o comércio eletrônico dos produtos e serviços de um grande provedor da Internet. Em seu funcionamento, um usuário da internet acessa o site do *Assine Já*, adiciona os produtos e/ou serviços de seu interesse em um carrinho de compras, lê e aceita os termos contratuais, informa os dados cadastrais, os dados de pagamento e finaliza o cadastro. Ao finalizar o procedimento, o sistema cadastra os dados do usuário que, a partir desse momento, passa a ser um assinante do provedor e ter direito a usar os produtos e serviços comprados.

A *Central de Relacionamentos* é outro web site desenvolvido para permitir que o usuário ao se cadastrar pelo *Assine Já* e se tornar um assinante do provedor, possa visualizar os produtos e serviços comprados, visualizar e editar seus dados cadastrais, suas informações de pagamento, visualizar suas cobranças, regularizar sua situação financeira, administrar seus dependentes, visualizar o contrato dos produtos, visualizar o histórico das suas conexões na internet através do provedor e entrar em contato para tirar dúvidas, relatar problemas e realizar reclamações, através da opção de envio de e-mail ou através de mensagens instantâneas.

As duas aplicações fazem parte da camada visão dentro da infra-estrutura de projetos da plataforma de vendas do provedor e foram desenvolvidas utilizando as seguintes tecnologias:

- ***Java***: linguagem de programação orientada a objetos;

- ***JSP***: *JavaServer Pages*, tecnologia utilizada no desenvolvimento de aplicações para Web;
- ***Struts***: *framework* para auxiliar no desenvolvimento da camada controladora numa estrutura seguindo o padrão Modelo 2;
- **Modelo 2**: arquitetura web que utiliza o projeto MVC (*Model-View-Controller*) para separar a lógica de negócio da lógica de apresentação do conteúdo, permitindo o desenvolvimento, teste e manutenção isolado de ambos;
- ***Ajax***: *Asynchronous JavaScript and XML*, desenvolvimento de aplicações em *JavaScript* com *XML*, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações;
- ***CSS***: *Cascading Style Sheets,* linguagem de estilo que define a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento;
- ***JavaScript***: linguagem de programação baseada na linguagem de programação *ECMAScript* padronizada pela Ecma *international* nas especificações **ECMA-262** e ISO/IEC 16262 e é atualmente a principal linguagem para programação *Client-Side* em navegadores web;
- ***Apache Tomcat***: servidor web Java, mais especificamente, um container de *servlets*. Possui algumas características próprias de um servidor de aplicação, porém não pode ser considerado um servidor de aplicação por não preencher todos os requisitos necessários.

O *Assine Já* e a *Central de Relacionamentos* não fazem comunicação direta com o banco de dados para armazenar e consultar informações, conforme ilustrado na figura 49 abaixo. Existe entre as aplicações e o banco de dados outra aplicação mediadora chamada *Kernel* para encapsular as regras de negócio do provedor. Construído com a tecnologia EJB - *Enterprise JavaBeans* um dos principais componentes da plataforma J2EE - *Java 2 Enterprise Edition*.

O *kernel* é uma aplicação do tipo servidor que executa no container do servidor de aplicação e seu principal objetivo é fornecer um rápido e simplificado

desenvolvimento de aplicações Java baseado em componentes distribuídos, transacionais, seguros e portáveis.

Ao utilizar uma das aplicações, os usuários de um dispositivo com conexão a internet e através de um navegador web, interage com o *Assine Já* ou com a *Central de Relacionamentos*, que valida e formata os dados informados e as envia para o *Kernel*. O *Kernel* recebe a requisição, processa as validações de negócio, armazena e obtêm informações no banco de dados para retornar as aplicações clientes, o *Assine Já* ou e a *Central de Relacionamentos*, que tratam as informações e as apresentam ao usuário.

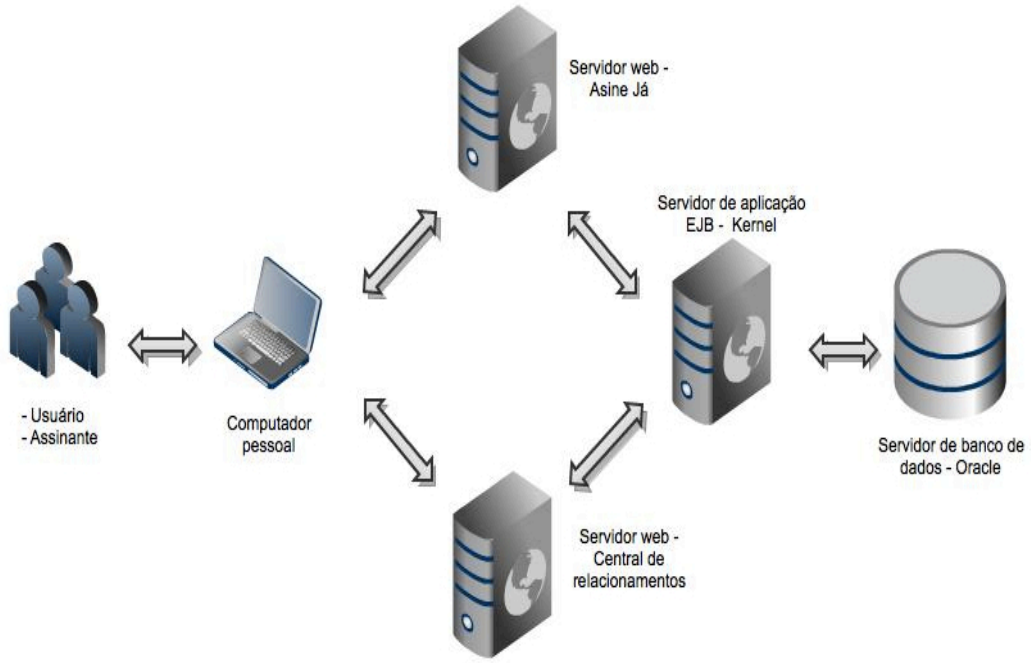

Figura 49 – Visão geral das aplicações utilizadas no experimento.

As aplicações têm forte dependência de negócio e são maduras em produção e consequentemente sofrem constantes manutenções para melhoria e otimização do software desenvolvido, como também reparo de defeitos. Pelo *Assine Já* são cadastradas aproximadamente três mil assinaturas diárias, apesar de suportar números bem superiores. A *Central de Relacionamentos* prove suporte para que aproximadamente um milhão de assinantes possa interagir com as informações da sua assinatura no provedor.

5.2.Escopo do experimento

Dado o elevado número de elementos que podem ser cadastrados no sistema e o tempo disponível para a realização dos testes, apenas um subconjunto de elementos foi selecionado pela equipe de desenvolvimento para passar pelo processo de geração de testes proposto e para comparar com outras formas de teste. Cada funcionalidade abaixo pode ser expressa em um caso de uso e poderão também ser referenciadas como tal a partir de agora.

Funcionalidade do *Assine Já* e seus principais fluxos de eventos utilizados no estudo de caso:

- UC1 - Efetuar compra
  - Efetuar primeira compra - produto Banda Larga - Cartão de Crédito;
  - Efetuar primeira compra - produto Banda Larga - Débito Instantâneo;
  - Efetuar segunda compra - produto Banda Larga - cartão de Crédito;
  - Efetuar segunda compra - produto Banda Larga - Débito Instantâneo;

Funcionalidades da *Central de Relacionamentos* e seus principais fluxos de eventos utilizados no estudo de caso:

- UC2 – Realizar autenticação
  - Autenticar com sucesso;
  - Autenticar com usuário incorreto;
  - Autenticar com senha incorreta;
- UC3 - Alterar dados cadastrais
  - Alterar informações de cadastro;
  - Alterar senha;
  - Alterar informações de localização;
  - Alterar contatos telefônicos;
- UC4 - Administrar dependentes
  - Adicionar dependente;
  - Alterar dependente;

- Remover dependente;
- UC5 - Alterar informações de pagamento;
  - Alterar dados do cartão de crédito;
  - Alterar para débito instantâneo;
  - Alterar dados do Débito instantâneo;

O caso de uso "UC1 - Efetuar compra" do *Assine Já,* cobre os principais fluxos de compra da aplicação, que é o principal canal de entrada de novas assinaturas no provedor, aproximadamente 80% dos assinantes cadastram suas assinaturas utilizando esses fluxos. Os quatro casos de uso da *Central de Relacionamentos* foram considerados pela equipe como as funcionalidades mais relevantes para os assinantes do provedor.

A indisponibilidade ou o mau funcionamento em umas das funcionalidades citadas acima, tanto do *Assine Já* quanto da *Central de Relacionamentos*, pode aumentar o número de chamadas telefônicas no *Call Center,* aumentando o TMA (Tempo Médio de Atendimento) dos atendentes, o que além de aumentar o custo da operacional, impacta o cadastro de novas assinaturas, já que o *Call Center*, assim como o *Assine Já*, é outro canal de venda.

Ao acionar a execução dos testes automáticos e verificar que as funcionalidades dos casos de uso executaram corretamente, a equipe estará segura de que as manutenções efetuadas no *Assine Já,* na *Central de Relacionamentos* ou no *Kernel* não causaram impacto nas funcionalidades e que elas estão em conformidade com a documentação descrita nos fluxos de evento do caso de uso.

5.3.Situação atual

Atualmente, as aplicações acima descritas não dispõem de nenhuma documentação formal de requisitos, em consequência disto, não existem documentos de testes extraídos de documentos funcionais. As aplicações não são cobertas por testes funcionais automatizados e os testes são executados de forma manual. A cada desenvolvimento de uma manutenção em uma das aplicações, seja ela corretiva ou evolutiva, existe a necessidade de realizar testes de regressão nas funcionalidades das aplicações alteradas, uma tarefa que exige muito tempo e esforço da equipe de desenvolvimento. Os testes manuais são executados de maneira desordenada, o que não garante um resultado único a cada execução.

5.4.Resultados obtidos

Para avaliar os resultados obtidos através da aplicação da abordagem proposta em um cenário real na fábrica de software do um provedor, foram realizados as seguintes medições e métodos:

5.4.1.Esforço para redigir os casos de uso

A primeira atividade realizada na aplicação do experimento foi medir o esforço para redigir cada caso de uso, listado no item "5.2.Escopo do experimento". Os casos de uso foram cadastrados pelos integrantes técnicos da equipe de desenvolvimento. Essa medição servirá como base para calcular o esforço de redigir todos os fluxos identificados e para comparar com outros métodos de teste. O esforço para cadastrar cada caso de uso engloba os dois passos seguintes:

- **Identificar os elementos do HTML** – tempo em minutos para identificar os elementos do HTML. Dado que o experimento foi executado utilizando aplicações já desenvolvidas e a ferramenta proposta gera os scripts para integrar ao *Selenium* e executar as "ações" e "verificações" nos formulários web, foi necessário fazer um levantamento dos elementos HTML de cada formulário web utilizado no campo "alvo" de cada passo dos fluxos descritos nos casos de uso. A forma mais ágil encontrada para identificar os elementos foi utilizar o *Selenium IDE,* navegar por cada fluxo do caso de uso e coletar as informações para documentar os passos.
- **Redigir os casos de uso** - tempo em minutos para redigir o caso de uso e seus fluxos de eventos (principal e alternativos).

O esforço para realizar cada atividade foi cronometrado e depois somado para chegar ao tempo total de cadastro do caso de uso, conforme tabela 4 abaixo. As medições da tabela 4 foram utilizadas como base para a estimativa do item 5.4.2 e para as comparações realizadas nos itens 5.4.3 e 5.4.4.

Casos de uso	Quantidade de Fluxos	Identificar HTML (minutos)	Redigir os casos de uso (minutos)	Total (minutos)
UC1	4	4.92	38.52	43.44
UC2	3	1.50	6.96	8.46
UC3	4	4.80	12.60	17.40
UC4	3	5.65	14.91	20.56
UC5	3	2.57	10.58	13.15
Total (minutos)	17	19.44	83.57	103.01

Tabela 4 - Esforço para redigir os casos de uso.

5.4.2.Esforço para assegurar que tenham sido criados suficientes scripts de teste

Para medir o esforço de cadastrar uma documentação mais abrangente e assegurar que os scripts cubram a maior parte das funcionalidades existentes, as aplicações foram mapeadas em casos de uso e, para cada um deles, seus fluxos de eventos:

1. **Identificar o número máximo de fluxos** – para cada caso de uso, identificar todos os possíveis fluxos das duas aplicações que não foram utilizados no escopo do experimento, item 5.4.1. Abaixo, os outros fluxos identificados para cada caso de uso:
   - Novos fluxos identificados no *Assine Já*:
     - UC1 - Efetuar compra
       - Efetuar primeira compra - produto Acesso Discado - Cartão de Crédito;
       - Efetuar primeira compra - produto Acesso Discado - Débito Instantâneo;
       - Efetuar segunda compra - produto Acesso Discado - cartão de Crédito;
       - Efetuar segunda compra - produto Acesso Discado - Débito Instantâneo;
       - Efetuar primeira compra - produto Avulso - Cartão de Crédito;
       - Efetuar primeira compra - produto Avulso - Débito Instantâneo;
       - Efetuar segunda compra - produto Avulso - cartão de Crédito;

- Efetuar segunda compra - produto Avulso - Débito Instantâneo;
- Efetuar primeira compra – CPF Inválido;
- Efetuar primeira compra – data de nascimento inválida;
- Efetuar primeira compra – usuário inválido;
- Efetuar primeira compra – senha inválida;
- Efetuar primeira compra – confirmação de senha inválida;
- Efetuar primeira compra – e-mail inválido;
- Efetuar primeira compra - Cartão de Crédito – numero do cartão inválido;
- Efetuar primeira compra - Cartão de Crédito – data vencimento do cartão inválida;
- Efetuar primeira compra - Cartão de Crédito – CPF outro pagante inválido;
- Efetuar primeira compra - Cartão de Crédito – data nascimento do pagante inválida;
- Efetuar primeira compra - Débito instantâneo – dados bancários inválidos;
- Efetuar primeira compra - Débito instantâneo – CPF outro pagante inválido;
- Efetuar primeira compra - Débito instantâneo – data nascimento do outro pagante inválida;
- Efetuar primeira compra – palavra de segurança não confere;

- Novos fluxos identificados na *Central de Relacionamentos*:
  - UC2 – Realizar autenticação
    - Autenticar com senha e usuário inválido;
    - Autenticar com usuário inadimplente;
    - Autenticar com usuário bloqueado;
    - Autenticar com usuário com data do cartão de crédito expirada;
  - UC3 - Alterar dados cadastrais

- Alterar informações de cadastro – CPF inválido;
- Alterar informações de cadastro – e-mail inválido;
- Alterar informações de cadastro – data nascimento inválida;
- Alterar senha – senha atual errada;
- Alterar senha – nova senha inválida;
- Alterar senha – confirmação de senha inválida;
- Alterar informações de localização – CEP inválido;
- Alterar contatos telefônicos – telefone contato inválido;
- Alterar contatos telefônicos – celular inválido;

- UC4 - Administrar dependentes
  - Adicionar dependente – dependente existente;
  - Adicionar dependente – senha inválida;
  - Adicionar dependente – senha não confere;
  - Adicionar dependente – e-mail inválido;
  - Adicionar dependente – celular inválido;
- UC5 - Alterar informações de pagamento
  - Alterar dados do cartão de crédito – numero cartão inválido;
  - Alterar dados do cartão de crédito – data de validade do cartão inválida;
  - Alterar dados do cartão de crédito – CPF do outro pagante inválido;
  - Alterar dados do cartão de crédito – data nascimento do outro pagante inválida;
  - Alterar dados do Débito instantâneo – dados bancários inválidos;
  - Alterar dados do Débito instantâneo – CPF outro pagante inválido;
  - Alterar dados do Débito instantâneo – data nascimento do outro pagante inválida;
- UC6 – Visualizar contratos
  - Visualizar contrato de um produto;

- Visualizar contratos de mais de um produto;
- Visualizar impressão;

- UC7 – Visualizar histórico de conexões
  - Visualizar histórico acesso banda larga;
  - Visualizar histórico acesso discado;
  - Visualizar impressão;
- UC7 – Baixar discador
  - Pesquisar cidades;
  - Baixar discador;
  - Acessar Central de Ajuda;
- UC8 – Entrar em contato
  - Entrar em contato via chat;
  - Entrar em contato por e-mail;
  - Entrar em contato por e-mail para cancelamento;
  - Entrar em contato por telefone;
  - Acessar Central de Ajuda;

2. **Calcular o tempo médio para cadastrar cada caso de teste** – calcular o tempo médio para redigir cada caso de teste na ferramenta proposta, tomando com base o tempo total para cadastrar os casos de uso utilizados no escopo do experimento, dividido pelo número de fluxos, ambos resultados da tabela 4. Ou seja, 103.01 minutos (tempo total) / 17 (número de fluxos cadastrados) = 6.06 minutos (tempo médio encontrado para cadastrar cada fluxo).
3. **Calcular o esforço para redigir cada caso de uso** - Calcular o tempo para redigir todos os fluxos e cada caso de uso, tanto os utilizados no experimento, quanto os identificados. Como exemplo, podemos utilizar o cálculo realizado para o caso de uso UC1. Foram redigidos na ferramenta proposta 4 fluxos que somados aos outros 22 fluxos identificados, totalizam 26 fluxos. Esses 26 fluxos foram multiplicados por 6.06 minutos (tempo médio para cadastrar um fluxo) e o resultado para cadastrar todos os fluxos do UC1 foi de 157.54 minutos.
4. **Calcular o esforço para redigir todos os fluxos identificados –** foram identificados um total de 74 fluxos que multiplicados por 6.06

minutos (tempo médio para cadastrar cada fluxo) totalizam 448.40 minutos.

Casos de uso	Principais fluxos cadastrados	Outros fluxos identificados	Total de fluxos	Tempo médio de cadastro do fluxo (minutos)	Total (minutos)
UC1	4	22	26	6.06	157.54
UC2	3	4	7	6.06	42.42
UC3	4	9	13	6.06	78.77
UC4	3	5	8	6.06	48.48
UC5	3	7	10	6.06	60.59
UC6	0	3	3	6.06	18.18
UC7	0	3	3	6.06	18.18
UC8	0	4	4	6.06	24.24
Total	17	57	74		448.40

Tabela 5 - Estimativa para cadastrar os casos de teste identificados.

O valor aproximado de 448.40 minutos é tempo total estimado para redigir todos os 74 fluxos identificados como necessários para assegurar que a aplicação gere e execute scripts de testes suficientes para dar confiança aos envolvidos com a aplicação que seu comportamento está conforme esperado. Para cadastrar os 17 fluxos utilizados no escopo deste experimento foram gastos 103.01 minutos (ver tabela 4), ou seja, foram redigidos apenas 23% do total de fluxos identificados.

5.4.3.Comparação com testes produzidos manualmente

Diante do fato da inexistência de testes funcionais automatizados nas aplicações utilizadas no experimento, do conhecimento adquirido no *framework* de testes *Selenium* e da tecnologia que as aplicações foram desenvolvidas, foi utilizado *JUnit* (JUnit, 2011) e *Selenium* para codificar manualmente os testes dos casos de uso. É importante ressaltar que a escrita manual do teste não teve o auxílio de funcionalidades de criação de código presentes em algumas *IDE's* e nem uso de "copiar" e "colar".

Para a criação dos testes com codificação manual foi utilizada a IDE Eclipse (Eclipse, 2011). Cada caso de uso codificado manualmente consistiu em um arquivo Java (Ex.: EfetuarCompraTest.java) de execução automatizada através de JUnit e a interação com o navegador automatizada a partir da API do *Selenium* para Java (Selenium, 2011). Foram criados manualmente 5 classes com 17

métodos para simular o mesmo cadastro realizado através da ferramenta proposta, conforme tabela 4.

A figura 50, a figura 51 e a figura 52 abaixo ilustram a estrutura criada para suportar a codificação manual e um exemplo do código criado para testar o fluxo principal do "UC1 - Efetuar Compra".

Figura 50 – Testes manuais criados na Eclipse IDE (Eclipse, 2011).

```
package com.globo.assine.selenium;

import junit.framework.Test;
import junit.framework.TestSuite;

import org.openqa.selenium.server.RemoteControlConfiguration;
import org.openqa.selenium.server.SeleniumServer;
public class AllTests {

    public static Test suite() {

        SeleniumServer server = null;
        try {
            RemoteControlConfiguration remoteControlConfiguration = new RemoteControlConfiguration();
            remoteControlConfiguration.setPort(1234);
            server = new SeleniumServer(remoteControlConfiguration);
            server.start();
        } catch (Exception e) {
            e.printStackTrace();
        }

        TestSuite suite = new TestSuite("Testes");
        suite.addTest(new TestSuite(UC1_EfetuarPrimeiraCompra.class));
        suite.addTest(new TestSuite(UC2_RealizarAutenticacao.class));
        suite.addTest(new TestSuite(UC3_AlterarDadosCadastrais.class));
        suite.addTest(new TestSuite(UC4_AdministrarDependentes.class));
        suite.addTest(new TestSuite(UC5_AlterarInformacoesPagamento.class));
        return suite;
    }

    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
}
```

Figura 51 – Código *JUnit* para invocar os testes manuais.

```
public class UC1EfetuarPrimeiraCompraTest extends AssineSeleneseTestCase {
    public void UC1EfetuarPrimeiraCompraComCartao() throws Exception {
        selenium.open("http://assine.globo.com/globocom/globomail-pro");
        verifyEquals("bt-adicionar-ao-carrinho", selenium.getAttribute("bt-adicionar-ao-carrinho@class"));
        assertTrue(selenium.isTextPresent("SuperProteção Premium"));
        assertTrue(selenium.isTextPresent("Antivírus globo.com"));
        selenium.click("bt-adicionar-ao-carrinho");
        selenium.isElementPresent("btn-prosseguir");
        selenium.click("btn-prosseguir");
        selenium.isElementPresent("//div[@id='glb-lateral']/div[2]/div/a/img");
        selenium.click("//div[@id='glb-lateral']/div[2]/div/a/img");
        selenium.waitForPageToLoad("30000");
        selenium.type("nome", "teste" + (Math.random() + "").substring(2, 9));
        selenium.type("cpf", "88844477718");
        selenium.type("dia-nasc", "11");
        selenium.type("mes-nasc", "12");
        selenium.type("ano-nasc", "1976");
        selenium.click("feminino");
        selenium.click("salvar-dados-usuario");
        selenium.waitForPageToLoad("30000");
        String loginGerado = "teste" + (Math.random() + "").substring(2, 9);
        selenium.type("loginEmail", loginGerado+ "@teste.com");
        selenium.typeKeys("nova-senha", "11q1q21q2w");
        selenium.type("confirmarSenha", "11q1q21q2w");
        selenium.select("perguntaSecretaId", "label=Como se chama seu bicho de estimação?");
        selenium.type("respostaSecreta", "cachorro");
        selenium.type("email-alternativo", "teste@tewte.com");
        selenium.type("cep", "22061060");
        selenium.fireEvent("cep", "keyup");
        selenium.focus("numero");
        selenium.type("numero", "9999999999");
        selenium.type("codigoTelRes", "21");
        selenium.type("telefoneRes", "24445555");
        selenium.type("codigoCel", "21");
        selenium.type("celular", "98989898");
        selenium.type("nome-cartao", "teste teste teste");
        selenium.type("numero-cartao", "4225375667238221");
        selenium.type("mes-cartao", "11");
        selenium.type("ano-cartao", "2010");
        selenium.type("palavra", "ispdois");
        selenium.click("salvar-dados-usuario");
        assertTrue(selenium.isTextPresent("sucesso"));
```

Figura 52 – Exemplo de um *script* utilizando Java com *Selenium API*.

O tempo gasto na codificação manual foi comparado com o tempo gasto para escrever o caso de uso apoiado pela ferramenta. O tempo para escrita dos oráculos não foi levado em conta em nenhum dos casos, pois, em ambos os testes, teriam que ser escritos os mesmos oráculos.

Geração manual	UC1 (minutos)	UC2 (minutos)	UC3 (minutos)	UC4 (minutos)	UC5 (minutos)	Total
Criar cada caso de uso	74.50	13.67	28.86	56.89	18.15	192.70
Refatorar o teste	21.40	5.79	9.78	15.78	7.65	60.40
Tempo total (minutos)	95.9	19.46	38.64	72.67	25.8	252.47

Tabela 6 – Esforço para criar os casos de teste manualmente.

Enquanto que para o cadastro de 5 casos de uso com 17 fluxos foram gastos apenas 103.01 minutos utilizando a ferramenta proposta (conforme tabela 4), a codificação manual consumiu 252.47 minutos, conforme tabela 6 acima. Assim, a geração do script de teste com o uso da ferramenta proposta neste trabalho exigiu, aproximadamente, apenas 40.8% do tempo gasto na geração manual do script.

5.4.4.Comparação com testes produzidos através de "*capture and replay*"

A abordagem para a geração e execução automática de testes apresentada neste trabalho e a abordagem "*capture and replay*" foram comparadas quanto ao tempo gasto para a geração dos testes. Essa comparação deu-se através dos testes gerados para cinco casos de uso, um do *Assine Já* e quatro da *Central de Relacionamentos.* A ferramenta escolhida para gerar os casos de teste a partir de "*capture and replay*" foi o *Selenium IDE,* uma extensão do navegador *Mozilla Firefox 3*.6.

A geração dos scripts de teste com *Selenium IDE* engloba três passos:

1. **Gravação:** gravar as ações do usuário para cada caso de uso.
2. **Refatoração do código:** todos os passos do caso de uso foram gravados em uma única execução do *Selenium IDE*, o que necessitou que o código gerado fosse refatorado para atribuir uma função a cada caso de teste e uma função inicial que prepara o ambiente para cada caso de teste (Todos estes passos encontravam-se em apenas uma única função no código oriundo da gravação). Ao executar os testes gravados pelo próprio *Selenium IDE*, foram encontrados problemas na execução de alguns passos, principalmente quando o formulário web utiliza *AJAX* (***A**synchronous **J**avascript **a**nd **X**ML*) e esses também foram refatorados para que o testes executassem corretamente.
1. **Inserção de oráculos**: mesmo com "*capture and replay*" ainda é necessário inserir as chamadas para as funções que exercem o papel de oráculos (ou assertivas).

Selenium IDE (capture and replay)	UC1 (minutos)	UC2 (minutos)	UC3 (minutos)	UC4 (minutos)	UC5 (minutos)	Total (minutos)
Gravação do fluxo	17.74	4.25	7.69	7.23	5.78	42.69
Refatoração do código	23.34	3.34	9.35	11.88	7.5	55.41
Inserção de oráculos	11.89	2.89	6.3	8.5	4.7	34.28
Tempo total em minutos	52.97	10.48	23.34	27.61	17.98	132.38

Tabela 7 – Esforço criar os cenários através de "*capture and replay*".

Como podemos ver a partir dos dados da tabela 7 acima comparada com o resultado da tabela 4, o ganho de tempo na geração de testes com a metodologia

presente neste trabalho em relação ao tempo gasto para a geração de testes utilizando "*capture and replay*" foi de, aproximadamente, 28,4% para os mesmos 5 casos de uso e 17 fluxos.

5.4.5.Dificuldades encontradas

Um problema encontrado diz respeito a identificação dos elementos do *HTML* das aplicações testadas na hora de redigir o caso de uso. Uma vez que a ferramenta proposta utiliza a integração com o *framework Selenium* para executar os scripts de teste na interface web, ele precisa conhecer o elemento "alvo" no formulário *HTML* para executar uma "ação" ou "verificação". Então, antes de redigir cada caso de uso, foi necessário identificar cada elemento do *HTML* do formulário web para que fosse possível redigir cada passo do tipo "ação" e "verificação" do fluxos de eventos. Do tempo total de 103.01 minutos gastos no cadastro dos 17 fluxos utilizados no experimento, 19.44 minutos foram utilizados para identificar os elementos *HTML*. Ou seja, aproximadamente 20% do tempo total para redigir os casos de uso foram utilizados na atividade. Atividade que não seria necessária se o experimento estivesse sido realizado em aplicações ainda não implementadas, uma vez que ao documentar o caso de uso e redigir cada passo do tipo "ação" e "verificação", os campos "alvo" seriam sugeridos pelo redator e posteriormente utilizados pelo desenvolvedor na implementação do formulário web da aplicação.

Outra dificuldade encontrada foi que o formulário de cadastro dos casos de uso se mostrou muito extenso e cansativo de ler. O problema pode ser exemplificado no caso de uso "UC1", onde foram identificados um total de 26 fluxos e eventos e todos são semelhantes ao fluxo principal ilustrado na figura 53 abaixo. Cadastrar os 26 fluxos tornaria o formulário demasiadamente grande e difícil de ler. Tal problema poderia ser minimizado se fosse implementado uma forma de reaproveitar fluxos ou passos semelhantes entre os fluxos e assim diminuir o número de passos repetidos do fluxo de eventos e consequentemente o seu tamanho.

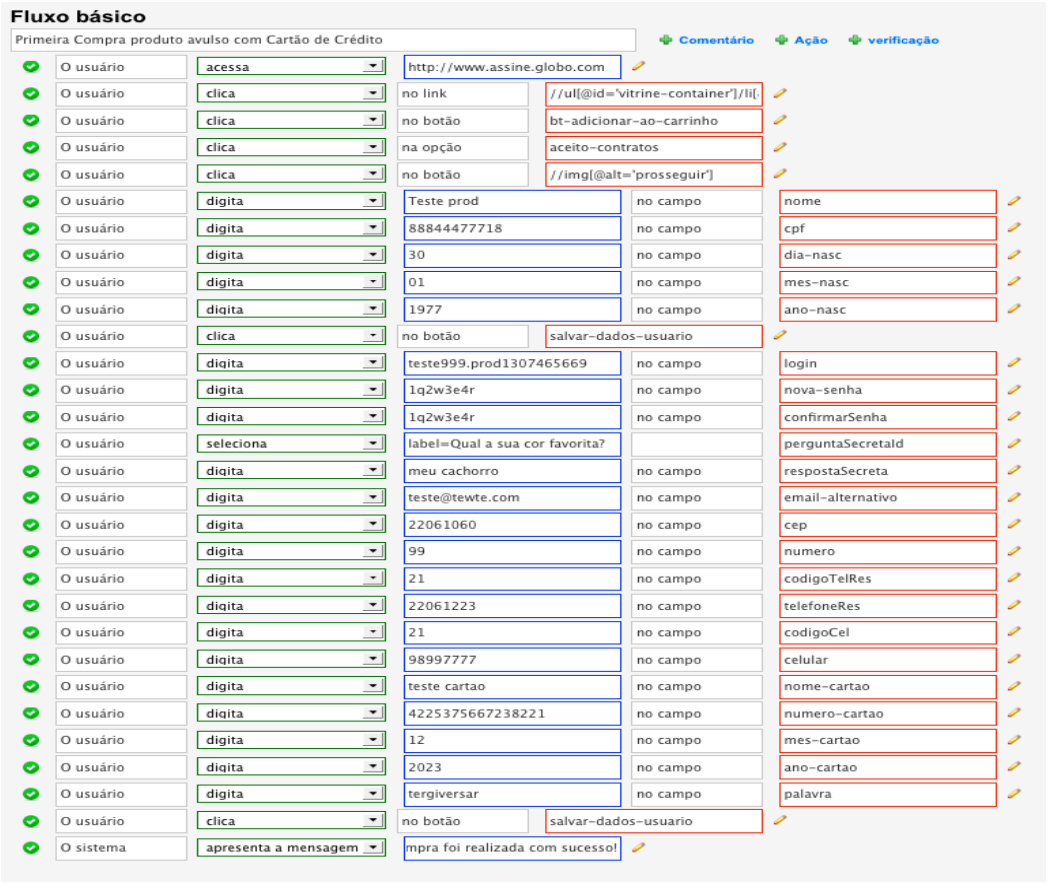

Figura 53 – Fluxo básico de eventos do caso de uso UC1.