UNIVERSIDADE DO SUL DE SANTA CATARINA

ARTUR TODESCHINI CRESTANI

DOUBLEDAY KARLO FRANCOTTI

ESTUDO DA VIABILIDADE PARA MODELAGEM E IMPLEMENTAÇÃO DE APLICAÇÕES DESKTOP, WEB E WEB COM AJAX USANDO MODELOS COMPARTILHADOS

Palhoça
2007

ARTUR TODESCHINI CRESTANI
DOUBLEDAY KARLO FRANCOTTI

ESTUDO DA VIABILIDADE PARA MODELAGEM E IMPLEMENTAÇÃO DE APLICAÇÕES DESKTOP, WEB E WEB COM AJAX USANDO MODELOS COMPARTILHADOS

Trabalho de conclusão de curso apresentado no curso de Sistemas de Informação na Universidade do Sul de Santa Catarina, como requisito parcial à obtenção dos títulos de bacharel em Sistema de Informação.

Orientador: Prof. Osmar de Oliveira Braz Júnior M.Eng.

Palhoça
2007

ARTUR TODESCHINI CRESTANI

DOUBLEDAY KARLO FRANCOTTI

ESTUDO DA VIABILIDADE PARA MODELAGEM E IMPLEMENTAÇÃO DE APLICAÇÕES DESKTOP, WEB E WEB COM AJAX USANDO MODELOS COMPARTILHADOS

Este projeto foi julgado adequado à obtenção do título de bacharel em Sistemas de Informação e aprovado em sua forma final pelo curso de Sistemas de Informação da Universidade do Sul de Santa Catarina.

Palhoça, 22 de novembro de 2007.

_____

Profº. Osmar de Oliveira Braz Júnior M.Eng.
Universidade do Sul de Santa Catarina

_____

Profº Aran Bey Tcholakian Morales Dr.

Universidade do Sul de Santa Catarina

_____

Profª. Vera Rejane Niedersberg Schuhmacher M.Eng.
Universidade do Sul de Santa Catarina

_____

Ricardo Limonta
Ciências Econômicas

AGRADECIMENTOS

Agradecemos ao nosso orientador Prof. Osmar de Oliveira Braz Junior pela paciência dispensada em nossa orientação. À professora Maira Inés Castañeira. À professoras Vera Rejane Niedersberg Schuhmacher pela ajuda nas modelagens e aconselhamentos na realização desta. À Laci Todeschini pela ajuda na formatação desta. À Liria Todeschini e Catia Silveira pela revisão gramatical e ortográfica desta.

À minha namorada Catia Silveira pelo apoio e a tolerância durante todo o processo de elaboração da mesma. À minha tia, que colaborou muito na elaboração do capítulo 1 do presente documento. À minha mãe que exaustivamente corrigiu os erros ortográficos e gramaticais presente em versões anteriores do presente documento. Pela ajuda dos colegas Joel Curtarelli, Matheus Meira, Alexandre Souza Vieira que ajudaram e tiram várias dúvidas tornando possível o desenvolvimento das aplicações envolvidas no documento. Ao meu amigo e instrutor Ricardo Limonta pelas idéias e auxílio com as dúvidas no desenvolvimento. Ao professor Osmar, nosso orientador em especial, pois sem ele não iríamos ter chego até aqui.

(Artur Todeschini Crestani)

À minha família em especial à minha mãe Maria Uciliadora Buzzo e meu irmão Wealth Karlo Francotti pelo amor e carinho que tenho pelos dois, pela compreensão das dificuldades enfrentadas durante o trabalho de criação da mesma. À minha noiva Pricilla Padaratz pelo carinho, apoio e a tolerância durante todo o processo de elaboração da mesma. Aos meus amigos e colegas de faculdade.

(Doubleday Karlo Francotti)

RESUMO

A proposta do presente projeto é realizar um estudo da viabilidade para modelagem e implementação de aplicações Desktop, Web e Web com AJAX usando modelos compartilhados, onde AJAX realizará a interatividade existente nas aplicações Desktop. Na busca de informações sobre a viabilização de transferir aplicações Desktop para Web com o uso da técnica AJAX, não encontramos muitas informações com fundamentos; foi quando surgiu a intenção de realizar a pesquisa. Este trabalho visa ajudar e orientar futuros projetos com a intenção de demonstrar a viabilidade das aplicações Desktop para Web tradicional e com o uso do AJAX. O projeto consiste em três protótipos com suas modelagens: Desktop, Web tradicional e Web com AJAX. Através das modelagens será demonstrada a viabilidade de transferir a aplicação Desktop para Web com o uso do AJAX. O objetivo do projeto e tornar possível a migração de um protótipo Desktop para Web mantendo a mesma interatividade e funcionalidades usando modelos compartilhados. Os objetivos específicos são: Discutir e apresentar as mudanças da modelagem de aplicações Desktop para Web; Identificar e comparar as diferenças entre a implementação de aplicações usando AJAX com Aplicações Web tradicional; Avaliar a perda de elementos interativos de interfaces gráficas sem o uso do AJAX para Web; Estudar os componentes funcionais do AJAX. A metodologia da pesquisa desenvolvida neste projeto sob o ponto de vista de sua natureza é considerada científica, contendo uma pesquisa aplicada na elaboração de protótipos e suas modelagens. O projeto irá apresentar uma análise das mudanças das modelagens das implementações de três protótipos Desktop, Web e Web com AJAX. O documento contém um estudo dos componentes funcionais da técnica AJAX, um breve resumo de engenharia de software, com metodologias e linguagens adotadas para a modelagem dos protótipos. Para o desenvolvimento dos protótipos utilizamos a linguagem JAVA, banco de dados Oracle 10g XE, como container Web utilizamos o Apache Tomcat 6x. A IDE utilizada para desenvolvimento foi o Eclipse com os plugins Visual Editor e Web Tools Platform. Para testes usamos a plataforma Windows XP para o protótipo Desktop e a dos protótipos para Web utilizamos os navegadores Mozilla Firefox 2x, Internet Explorer 6x e7x da Microsoft e Opera 9x. No desenvolvimento foi utilizado o paradigma de orientação a objetos e o padrão MVC de arquitetura de software.

Palavras chave: Modelagem, Compartilhados, AJAX.

ABSTRACT

The proposal of this project is a study of the feasibility for modeling and implementation of applications Desktop, Web and Web with AJAX using models shared, where AJAX held interactivity existing applications on Desktop. In search of information on the feasibility of transferring Desktop for Web applications using the AJAX technique, we found much information with pleas; was when the intention to carry out the search. This work aims to assist and guide future projects with the intention of demonstrating the viability of traditional Web applications for Desktop and with the use of AJAX. The project consists of three prototypes with their modelagens: Desktop, Web and traditional Web with AJAX. Through modelagens will be demonstrated the feasibility of transferring the application to Web Desktop with the use of AJAX. The purpose of the project and made possible the migration of a prototype Web Desktop for maintaining the same functionality and interactivity using models shared. The specific objectives are: Discuss and make changes in modeling of applications for Desktop Web; identify and compare the differences between the implementation of applications using AJAX with traditional Web Applications; Assessing the loss of interactive elements of graphical interfaces without the use of AJAX to Web; Studying the functional components of AJAX. The methodology of the research on this project from the point of view of its nature is considered scientific, containing an applied research in the development of prototypes and their modelagens. The project will present an analysis of the changes of modelagens of implementations of three prototypes Desktop, Web and Web with AJAX. The document contains a study of the functional components of the art AJAX, a brief summary of software engineering, languages and methodologies adopted for the modeling of prototypes. For the development of prototypes using the Java language, the database Oracle 10g XE, as the container Web use Apache Tomcat 6x. The IDE used for development was the Eclipse plugins with the Visual Editor and Web Tools Platform. For testing we used the Windows XP platform prototype for the Desktop and the prototypes to use the Web browser Mozilla Firefox 2x, Internet Explorer 6x e7x of Microsoft and Opera 9x. In developing the model was used to guide objects and the standard MVC architecture of the software.

Key works: Modeling,.Share, AJAX.

LISTA DE SIGLAS

AJAX – Asynchronous JavaScript And XML

API – Application Programming Interface

ASP – Active Server Pages

CGI – Common Gateway Interface

CMMI – Capability Maturity Model Integration

CSS – Cascading Style Sheets

DAO – Data Access Object

DHTML – Dynamic Hyper Text Transfer Protocol

DOM – Document Object Model

DTD – Document Type Definition

DWR – Direct Web Remoting

GWT – Google Web Toolkit

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

IETF – Internet Engineering Task Force

ISO – International Organization for Standardization

J2SE – Java 2 Standard Edition

JDBC – Java Data Base Connectivity

JSON – JavaScript Object Notation

JRE – Java Runtime Environment

JSP – JavaServer Pages

JVM – Java Virtual Machine

MVC – Model *View* Controller

OMG – Object Management Group

PHP – PHP Hypertext Preprocessor

POJO – Plain Old Java Object

RFC – Request for Comments

RUP – Rational Unified Process

SAJAX – Simple AJAX Toolkit

SGML – Standard Generalized Markup Language

UI – User Inteface

UML – Unified Modeling Language

URL – Uniform Resource Locator

W3C – World Wide Web Consortium

WWW – World Wide Web

XHR – XML HTTP Request

XP – eXtreme Programming

XML – Extensible Markup Language

YFT – Yellow Fade Technique

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A presente monografia demonstra o estudo da viabilidade para modelagem e implementação de aplicações Desktop, Web e Web com *Asynchronous JavaScript And XML* (AJAX) usando modelos compartilhados. Onde a técnica AJAX realizará a interatividade existente nas aplicações Desktop.

> No inicio do milênio o destaque era para aplicações Desktop com acesso ao banco, que desenvolvíamos com tecnologia de multicamadas. Então surgiu a necessidade de oferecer uma interface Web para acessar as informações geradas pelos sistemas tradicionais. As primeiras telas para internet geralmente eram para formulários de busca que desenvolviam resultados e HTML. Basicamente tudo era realizado pelos servidores escritos em JAVA e o JavaScript era utilizado para validação dos campos. O desenvolvimento das novas aplicações direcionara-se à Web. (BORBA, 2006, p.9).

Segundo Borba (2006, p. 9) a Web tornou-se uma nova alternativa interessante para os desenvolvedores de software, porém os aplicativos desenvolvidos para Web eram sites dinâmicos, com pouca interatividade, que comparado com as aplicações Desktop. Cunha (2006, p. 11) comenta a necessidade de trazer para as aplicações Web uma interatividade semelhante às utilizadas nas aplicações Desktop.

A popularização do uso de AJAX possibilita ao usuário da Web uma experiência de usabilidade mais rica, realizando a mesma interatividade das aplicações Desktop como processador de texto, planilhas de cálculo e programas de apresentação de slides para Desktop.

Oliveira e Rigo (2006, p. 7) ressaltam que AJAX e seus elementos internos em uma página Web, proporcionam interatividade às páginas *Hypertext Markup Language* (HTML).

> [...] a identificação de elementos internos a uma página Web, que fazem parte de sua composição com finalidades distintas e cujo acesso pode ter um significado importante. Tecnologias como AJAX estão sendo testadas para prover o tratamento dos elementos das páginas Web com maior nível de detalhe e com maior interatividade. (OLIVEIRA; RIGO, 2006, p.7).

Apresenta-se um estudo da viabilidade da migração de aplicações Desktop para Web, através de uma análise comparativa as modelagens desenvolvidas, onde utilizaremos modelos compartilhados.

Para essa análise, desenvolveremos três protótipos:

Protótipo Desktop, Protótipo Web tradicional e Protótipo Web com AJAX mostrando as igualdades, semelhanças e diferenças entre as três modelagens e suas implementações.

1.1 PROBLEMÁTICA

O problema para desenvolver aplicações para Web, realizando as mesmas interatividades das aplicações Desktop, é que os servidores Web foram projetados para fornecer respostas sincronizadas e para elementos interativos de interfaces são necessárias respostas assíncronas. Com o uso da técnica AJAX é possível reproduzir essas respostas assíncronas.

1.2 OBJETIVOS GERAIS

Realizar um estudo da viabilidade pra a modelagem e implementação de aplicações Desktop, Web e Web com AJAX usando modelos compartilhados.

1.3 OBJETIVOS ESPECÍFICOS

A seguir estão descritos os objetivos específicos deste projeto:

a) Discutir e apresentar as igualdades, semelhanças e diferenças entre as três modelagens desenvolvidas, usando modelos compartilhados;
b) Identificar e comparar as diferenças entre a implementação de aplicações usando AJAX com Aplicações Web tradicional;
c) Mostrar a implementação das aplicações desenvolvidas (protótipos Desktop, Web e AJAX);
d) Avaliar a perda de elementos interativos de interfaces gráficas sem o uso do AJAX para Web;

e) Estudar os componentes funcionais do AJAX.
f) Tornar possível a migração de um protótipo Desktop para a Web mantendo a sua mesma interatividade e funcionalidade presente no protótipo Desktop.

1.4 JUSTIFICATIVA

Realizar um projeto que demonstre um estudo da viabilidade para modelagem e implementação de aplicações Desktop, Web e Web com AJAX usando modelos compartilhados.

Demonstrar que é possível migrar uma aplicação Desktop, para Web mantendo suas funcionalidades e interatividade com o uso da técnica AJAX.

Analisar as igualdades, semelhanças e diferenças na modelagem e nas implementações realizadas pelos autores.

Gerar um material com qualidade, para que esse projeto possa servir de referência não só para colegas, mas também para profissionais, no dia a dia de seu trabalho, bem como para outras pessoas, pois foi encontrada pouca documentação em periódicos, meios eletrônicos ou mesmo em livros de como funcionam os processos de manipulação de dados com uso da técnica AJAX.

Adquirir conhecimento mais conciso das metodologias e técnicas estudadas, para buscando além do conhecimento gerar um material de qualidade para futuras pesquisas.

1.5 PROPOSTA DE SOLUÇÃO

Desenvolvimento de três protótipos em duas plataformas diferentes e seus respectivos modelos com acesso a banco de dados e tabelas compartilhadas.

Protótipo Desktop;

Protótipo Web tradicional;

Protótipo Web com AJAX.

Na figura 1 demonstrada a proposta de solução do projeto. No qual terá três desenvolvimentos com suas receptivas modelagens.

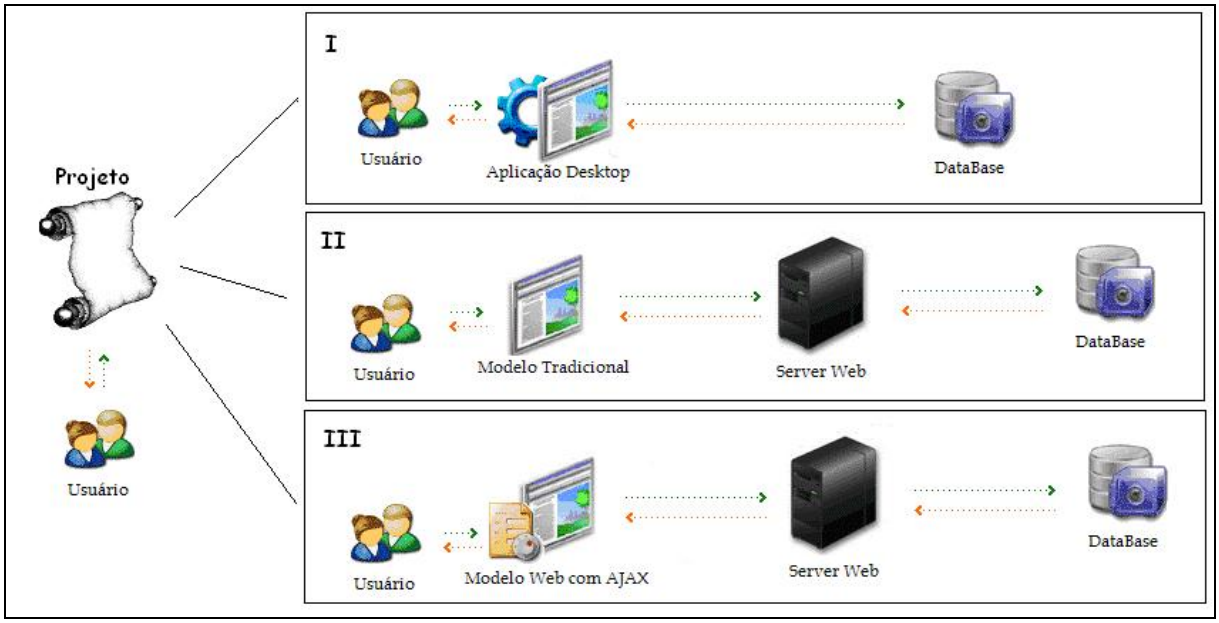


Figura 1 - Proposta de solução
Fonte: Os autores

1.5.1 Descrição da proposta

Demonstrar que aplicações Desktop e aplicações Web com AJAX podem ser modeladas e implementadas de maneiras semelhantes sem restrições de interatividade com relação a aplicações Web sem AJAX.

1.5.2 Elementos contidos na proposta

Serão desenvolvidos três pequenos protótipos com as funcionalidades de cadastro, exclusão e pesquisa em base de dados Oracle 10g XE. Contendo para cada protótipo o desenvolvimento das modelagens e seus requisitos funcionais e não funcionais (caso de uso, diagrama de realização e diagrama de atividade).
Como demonstrada na figura 1:

O item I será realizado o desenvolvimento do protótipo Desktop na linguagem Java usando a API *Swing*;

O item II será realizado o desenvolvimento do protótipo Web também na linguagem Java usando *Servlets* e *Java Server Pages* (JSP);

O item III será realizado o desenvolvimento do protótipo Web com AJAX também utilizando a linguagem Java usando *Servlets* e JSP.

Onde os três protótipos usarão as mesmas classes de objetos de domínio e a mesma DAO com um driver JDBC.

1.6 DELIMITAÇÃO DO TRABALHO

A proposta deste trabalho não compreende o desenvolvimento de softwares completos. Apenas a implementação de três protótipos simples em suas respectivas plataformas, Desktop utilizando a tecnologia JAVA, Web e Web com AJAX utilizando a tecnologia Java para Web com *Servlet* e *Java Server Pages* (JSP).

Não será levado em consideração requisitos de usabilidade nos desenvolvimentos dos protótipos.

Não será levado em consideração o custo para a migração de um projeto Desktop para Web para uma empresa ou organização.

Não será levado em consideração o uso do processo de segurança nos protótipos.

Estes protótipos envolveram duas tabelas, um relacionamento e algumas validações em tela.

Comprovando, desta maneira, que a migração das aplicações Desktop para aplicações Web com o uso do AJAX traz a mesma interatividade presente nas aplicações Desktop.

1.7 METODOLOGIA

Segundo Bello (2004) a metodologia é a explicação minuciosa, detalhada, rigorosa e exata de toda ação desenvolvida no método (caminho) do trabalho de pesquisa.

A pesquisa desenvolvida neste projeto sob o ponto de vista de sua natureza é considerada científica, contendo uma pesquisa aplicada na elaboração de protótipos e suas modelagens.

1.8 ESTRUTURA DA MONOGRAFIA

O trabalho está estruturado da seguinte forma:

Capítulo 1: neste capítulo é apresentada à contextualização de pesquisa deste trabalho, são definidos os objetivos propostos e a justificativa.

Capítulo 2: aborda uma revisão sobre engenharia de software, ciclo de vida de software, um breve histórico da Web, as principais tecnologias de apresentação de conteúdo, comentário sobre o paradigma da orientação a objetos, um comentário sobre a *Unified Modeling Linguage* (UML) e uma revisão sobre ICONIX.

Capítulo 3: neste capítulo é realizada a modelagem dos três protótipos no formato de diagramas de caso de uso, diagrama seqüência, diagrama de robustez e diagrama de atividade e análise da modelagem.

Capítulo 4: neste capitulo é demonstrado as ferramentas computacionais utilizadas para o desenvolvimento do projeto.

Capítulo 5: neste capítulo apresentamos um histórico do desenvolvimento do projeto.

Capítulo 6: neste capítulo é realizada a análise dos resultados.

Capítulo 7: neste capítulo apresentamos uma proposta para trabalhos futuros.

Capítulo 8: neste capítulo encontram-se as conclusões do projeto.

2 REVISÃO BIBLIOGRÁFICA

Abordar-se-á nesse capítulo um resumo histórico da Web, interatividade, tecnologias de apresentação de conteúdo para o cliente, AJAX e um breve resumo de engenharia de software. Metodologias e linguagens adotadas para a modelagem dos protótipos.

2.1 HISTÓRICO DA WEB

Os Web sites, portais como são conhecidas as páginas desenvolvidas para *World Wide Web* (WWW), são colocados à disposição do mundo inteiro através da Internet. Os sites podem conter qualquer tipo de informações: dados de uma determinada empresa, um currículo de um indivíduo, fotografias, clips de vídeo e outras tantas informações. A idéia principal é disponibilizar as informações de maneira rápida e ágil aos clientes ou interessados no conteúdo.

> A WWW é, em termos gerais, a interface gráfica da Internet. Ela é um sistema de informações organizado de maneira a englobar todos os outros sistemas de informação disponíveis na Internet. Sua idéia básica é criar um mundo de informações sem fronteiras, prevendo as seguintes características: Interface consistente; Incorporação de um vasto conjunto de tecnologias e tipos de documentos; 'leitura universal'. (CASTRO, 2007).

Um site seria monótono se ele tivesse apenas texto e imagens estáticas. No começo da internet as páginas HTML eram estáticas. Isso era considerado maravilhoso, mas o ser humano é insaciável pela sua própria natureza, quer sempre mais. O dinamismo nas páginas tornou-se imprescindível, entretanto os servidores Web não foram projetados para serem dinâmicos. Surgiu um problema que não havia até então, pois o protocolo HTTP foi feito para ter arquivos estáticos e não dinâmicos. Foi possível certa dose de dinamismo na Web com *Common Gateway Interface* (CGI), resultando num retardo, pois as solicitações precisam ser enviadas ao servidor, um aplicativo ser executado e os resultados interpretados pelo navegador. Sendo assim, nasceu à necessidade do uso de linguagens no lado servidor (*client-side*). (SHARMA, 2000, p. 8 - 14).

Claro que o CGI podia ser aperfeiçoado. Em 1995, foi anunciado por John Gage o nascimento da linguagem de programação Java. Surge então um novo caminho para as

páginas dinâmicas. Assim aparece uma nova proposta para resolver um outro problema, o problema da falta da interatividade na Web. Para trazer o atrativo da interatividade para os sites, os usuários necessitavam a instalação de plugins de terceiros que são tecnologias proprietárias, tecnologias essas no lado cliente (*client-side*). Isso faz com que o usuário que deseja acessar um site desenvolvido com essa tecnologia, tenha instalado o plugin no seu navegador. No decorrer desse *capítulo* falaremos mais sobre plugins.

> [...] os clientes querem um aplicativo com mais recursos, e ao mesmo tempo desenvolvedores querem evitar a instalação de arquivos executáveis em milhares de estações de trabalho. (ASLESON; SCHUTTA, 2006, p.13).

Ressalta-se nesse ponto da monografia que a técnica de desenvolvimento AJAX para aplicações Web possibilita a mesma interatividade das aplicações Desktop, sem o uso de plugins no lado cliente.

2.2 INTERATIVIDADE

"Interatividade. 'Sabemos o que é quando a vemos', mas o que ela é? Quando pedimos para definir o termo, em meios de comunicação, muitos indivíduos ou até mesmo estudiosos podem achar difícil de responder". (MCMILLAN, 2000, p. 2, tradução nossa).

No dicionário Novo Aurélio, interatividade é: "[de interativo + (i)dade] S.t 1. Caráter ou condição de interativo. 2. Capacidade (de um equipamento, sistema de computação ou de computador, etc.) de interagir ou permitir interação." (FERREIRA, 1999, p. 1123).

No Oxford Dictionary of Computing. Interatividade: adjetivo (usuário operador / sistema operacional) conexão com dois sentidos de caminho de informação e dados. Entre um sistema e um usuário, com o sistema respondendo para o usuário suas requisições. (PYNE; TUCK, 1996, p. 195, tradução nossa).

A Interatividade é um atributo natural da conversação face a face, mas propôs-se ocorrer também em ajustes mediados de uma comunicação. Por exemplo, a interatividade é também uma das características definidas de sistemas em dois caminhos, - ida e volta, com uma resposta gerada pelo indivíduo. Isso pode ocorrer em sistemas de texto eletrônico, em

algum trabalho de programação, em jogos, vídeos interativos. A Interatividade está presente na operação dos meios tradicionais de notícias. Os fenômenos das letras no editor (interagindo como aplicativo), as mostras da conversa no rádio e televisão, com a participação do ouvinte nos programas, são caracterizados como interatividade. (MCMILLAN, 2000, p. 2, tradução nossa).

AJAX possibilita o uso de atributos interessantes a aplicativos Web, tornando-os mais interativos, ágeis, espertos, lúdicos e com jeito de software. Permite não tirar o usuário do seu contexto e não direcioná-lo para fluxos alternativos. O AJAX faz os desenvolvedores repensarem se faz sentido desenvolver os *client-side* tão distantes dos *server-side*. (GLUZ. 2006, p. 48 - 53).

2.3 TECNOLOGIAS DE APRESENTAÇÃO DE CONTEÚDO

Serão apresentadas nesse tópico as principais tecnologias de apresentação de conteúdo envolvidas nesta monografia.

2.3.1 HTML

Segundo a *Internet Engineering Task Force* (IETF), na RFC1866 (*Request for Comments*).

HTML é uma linguagem simples de marcação usada para criar documento de *hypertext* que são independentes de plataforma. Os documentos HTML são *Standard Generalized Markup Language* (SGML) com semântica genérica que são apropriados para apresentar a informação para a Web com uma larga escala dos domínios. A marcação HTML pode representar *hypertext* notícia, correio, documentação, e *hypermedia;* menus para opções; resultados da pergunta da base de dados; documentos estruturados simples com gráficos alinhados; e vistas do *hypertext* de corpos existentes de informação. O HTML esteve no uso da WWW em atividade na informação global desde 1990. O HTML é um requerimento da *International Organization for Standardization* (ISO) *Standard* 8879:1986 organizador das

informações texto e de sistemas de escritório SGML. (LEE; CONNOLLY, 1995, tradução nossa).

Exemplo de código de html: exemplo.html

```
<html>
  <head>
        <title>Exemplo Html</title>
   </head>
   <body>
        <p>
      Exemplo de códgio Html
        </p>
   </body>
</html>
```

A figura 2, demonstra a execução do código exemplo.html acima descrito.

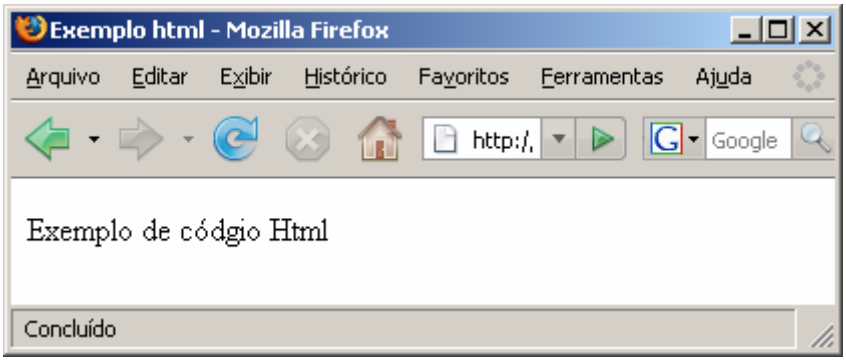


Figura 2 - Exemplo de html
Fonte: Os autores.

2.3.2 DHTML

Dynamic HTML (DHTML), ou HTML dinâmico é o termo do marketing aplicado a uma mistura dos padrões incluindo o HTML, as folhas do estilo do *Document Object Model* (DOM1) e scripting. Entretanto, não há nenhuma especificação do *World Wide Web Consortium* (W3C) que define formalmente DHTML. A maioria das normas de procedimento podem ser aplicáveis, usando DHTML. (CHISHOLN; VANDERHEIDEN; JACOBS, 1999, tradução nossa).

Exemplo de código dhtml: exemplo_dhtml.html

```
<html>
<head>
<script type="text/JavaScript">
```

```
msgfield=null
txtsize=0
maxsize=60

function writemsg()
{
   if(msgfield==null)
      msgfield=document.getElementById("msg")
  if (txtsize<maxsize)
    {
      msgfield.style.fontSize=txtsize
      txtsize++
      timer=setTimeout("writemsg()",10)
    }
}
function stoptimer()
{
   clearTimeout(timer)
}
</script>
<title>Exemplo DHTML</title>
</head>
<body onload="writemsg()" onunload="stoptimer()">
    <p id="msg" style="FONT-SIZE: 2px">
   Estou a crescer</p>
 </body>
</html>
```

A figura 3, demonstra a execução do código exemplo_dhtml.html acima descrito.

Figura 3 - Exemplo de dhtml
Fonte: Os autores.

2.3.3 DOM

Documnet Object Model (DOM) é uma recomendação do W3C, para um API de programação padrão, baseada em árvore, para documentos XML. Criada inicialmente como forma de implementar programas Java e JavaScript coerentemente, de forma que não importe

o browser que é utilizado pelo cliente. Evoluiu para uma API XML de uso geral para qualquer Protótipo, sejam eles editores ou até mesmo sistemas de gerenciamento de arquivos. (RAY, 2001, p. 321).

Segundo o W3C, O DOM é uma plataforma de imagem e linguagem-neutra que permite que os programas e os scripts acessem e façam *update* do contexto dinamicamente. O documento processado é o resultado do processo que pode ser incorporado por trás da página apresentada. (World Wide Web Consortium, 2007b, tradução nossa).

É um conjunto de interfaces Java (e JavaScript) que declaram métodos que o desenvolvedor deve desenvolver, onde as interfaces não definem *call-backs* para eventos, utilizam assim métodos os quais permitem a modificação dos objetos. O DOM faz para os programas o que o XML faz para os documentos.

O DOM descreve os contêineres para elementos, atributos e outros tipos de nós básicos, também inclui módulos que lhe acrescentam funcionalidades do tratamento especializado da HTML para eventos do usuário e folhas de estilo. (RAY, 2001, p 321 - 323).

É uma representação da página totalmente orientada a objeto. Pode ser alterada, percorrer o conteúdo da página com o uso de linguagens de scripts como JavaScript e VBScript, possibilitando páginas altamente dinâmicas. (ASLESON; SCHUTTA, 2006, p. 6-7).

O DOM foi baseado nas especificações do *Object Management Group* (OMG), permitindo usar qualquer linguagem de programação. Define os objetos a representação e alteração de documentos, o comportamento e atributos desses objetos e seus relacionamentos. Sem essa especificação muitos aspectos interessantes do AJAX não seriam possíveis. Representa uma árvore dos dados e estrutura de uma página, mas não pode ser implementado dessa forma. (ASLESON; SCHUTTA, 2006, p. 36).

Exemplo do código DOM:

O exemplo mostra tags HTML que são percorridas pelo DOM e que pode ser editadas por ele.

```
<table>
        <tbody>
          <tr>
            <td> Foo</td>
            <td>Bar</td>
          </tr>
        </tbody>
</table>
```

2.3.4 CSS

Cascading Style Sheets, ou CSS, segundo a W3C é um mecanismo simples para adicionar estilo (por exemplo, fontes, cores, espaçamento) aos documentos. (World Wide Web Consortium, 2007a, tradução nossa).

As CSS suportam um número considerável de unidades, inclusive medidas padrão, percentagens e sistemas de notações de cores e de *Uniform Resource Locator* (URL) específicos. Permitindo dessa maneira aos desenvolvedores atribuir unidades apropriadas. As medidas contidas dentro das CSS são tanto absolutas como relativas. (KIRK; MOULTIS, 2000, p. 300).

Exemplo de código CSS: exemplo_css.css

```
a:link {
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size: 11px;
text-align: left;
text-decoration: underline;
text-indent: 20px;
color: #0000CC;
}

a:visited {
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size: 11px;
text-align: left;
text-decoration: underline;
text-indent: 20px;
color: #993366;
}

.text1 {
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size: 11px;
font-weight: bold;
text-align: left;
text-decoration: none;
text-transform: inherit;
text-indent: 20px;
color: Black;
background-color: #FFFFCC;
}

a:hover{
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size: 12px;
font-weight: bold;
text-align: left;
text-decoration: underline;
background-color: #6699FF;
```

```
}
```

Exemplo do código exemplo.html utilizando o CSS exemplo_css

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

	<title>Exemplo CSS</title>

	<link href="exemplo_css.css" rel="stylesheet" type="text/css" />
</head>

<body>
	Exemplo de efeitos obtidos com CSS<br>
	<div class="text1">Exemplo de texto com o CSS</div>
	Exemplo de texto sem o uso do CSS<br>
	Exemplo obtidos sobre links usando como parâmetros os e-mails dos autores da monografia<br>
	<a href="mailto:artodeschini@gmail.com">Artur Todeschini Crestani</a><br>

	<a href="mailto:doubleday.fracotti@gmail.com">Doubleday K. Francotti</a><br>
</body>
</html>
```

A figura 4, demonstra a execução do código exemplo_css.html acima descrito.

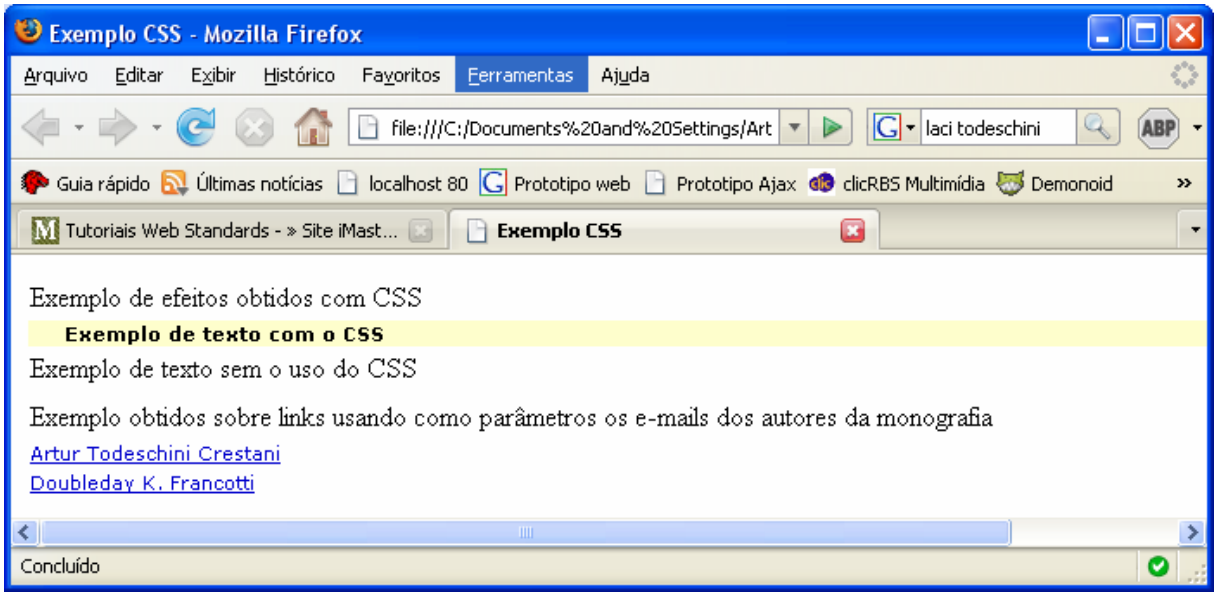


Figura 4 - Exemplo de CSS
Fonte: Os autores.

2.3.5 JavaScript

Linguagem de script desenvolvida pela Netscape teve várias nomenclaturas antes de concorrentes como da Microsoft VBScripts.

Originalmente criada para auxiliar os desenvolvedores a alterar dinamicamente as tags de suas páginas. Percebeu-se que seria possível então tratar as páginas como objetos nascendo assim o DOM. (ASLESON; SCHUTTA, 2006, p. 6).

Com JavaScript é possível criar pequenos programas que são executados a partir do browser tecnologia (*client/side*). Na realidade, os scripts são um conjunto de instruções, ou seja, quando uma página é carregada você pode fazer o script executar algum determinado procedimento. Pode ser usado para gerenciar frames do browser, criando dessa maneira sumários interativos. Enfim, pode-se acrescentar recursos a uma página Web com as suas funcionalidades.

Entre algumas das funcionalidades possíveis com o uso do JavaScript é possível colocar uma data de modificação em suas páginas, indicando a última vez que a página foi modificada. Permite abrir janelas secundárias, criar botões de navegação que tenham a mesma função dos botões encontrados na barra de navegação do browser, dar uma mensagem de boas vindas personalizada, mensagens de partidas da página e notícias que mudem freqüentemente (KENT; KENT 1997, p 1 – 41).

JavaScript Object Notation (JSON) ou notação do objeto do JavaScript é um link texto baseado, os dados da linguagem independente fazem um intercâmbio do formato. É derivado da linguagem de programação ECMAScript. JSON define um jogo pequeno de regras do formato para a representação portátil de dados estruturados. JSON pode representar quatro tipos primitivos (*strings*, *numbers*, *booleans e null*) e dois tipos estruturados (objetos e vetores). Uma string é uma seqüência de zero ou mais caracteres *Unicode*. Um objeto é uma coleção da união de zero ou mais pares do nome/valor, onde um nome é uma string e um valor é uma string, número, *boolean*, nulo, objeto ou vetor. Uma disposição é uma seqüência requisitada de zero ou mais valores. Os termos ‘objeto’ e ‘vetor’ vindo das convenções de JavaScript. Os objetivos do projeto JSON eram para ele ser mínimo, portátil, textual, e um subconjunto do JavaScript. (CROCKFORD, 2006, tradução nossa).

Exemplo do código JavaScript: destino.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Exemplo JavaScript </title>
                        <script language="JavaScript">
                                setTimeout("location='destino.html'", 2000);
                        </script>
        </head>
        <body>
                <h1> Exemplo JavaScript uma página que chama outra página
                <p>chama a página destino.html</h1>
        </body>
</html>
```

A figura 5, demonstra a execução do código destino.html acima descrito. Chamando-se a si mesma recursivamente.

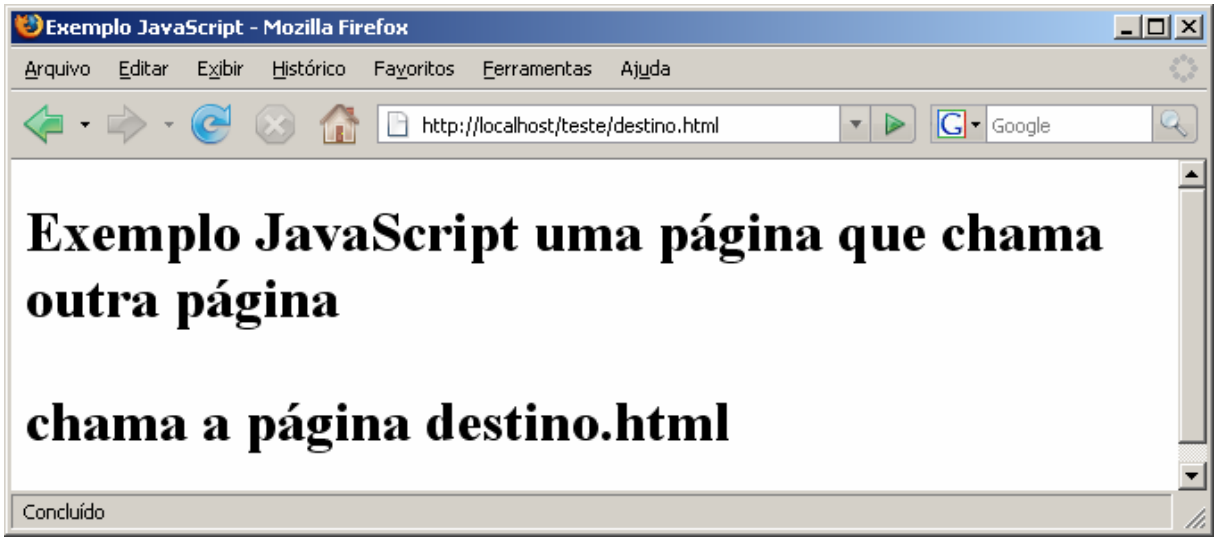


Figura 5 - Exemplo de JavaScript.
Fonte: Os autores.

2.3.6 Plugin

No Oxford Dictionary of Computing (PYNE; TUCK, 1996, p. 269), Plug: adjetivo (de software e hardware que é especificamente feito de maneira, que pode ser usado com diferentes computadores ou sistemas sem a necessidade de ser mudado.

Plugins para navegadores são: tecnologias que permitem as aplicações Web ter os mesmos recursos ricos com detalhes como os das aplicações Desktop. (JEVEAUX, 2005, p. 24).

Tecnologias do lado cliente, que requerem a instalação de plugins de navegadores e *softwares* patenteados. A *Rich Internet Application* (RIA) ou aplicações ricas para internet, são tecnologias que geram um novo tipo de experiência na Web que acoplando plugins ao browser geram interatividade as páginas Web. RIA's oferece a flexibilidade e a facilidade na utilização de um Protótipo Desktop inteligente, adicionando ao alcance de aplicações Web. Com RIA pode se alcançar uma audiência ampla das necessidades dos clientes, dos sócios, e dos empregados com rico índice e interatividade. (ADOBE, 2007, tradução nossa).

2.3.7 Java plug-in

O *Java Runtime Environment* (JRE) fornece as bibliotecas, a máquina virtual Java e outros componentes para executar os *applet* e as aplicações escritos na linguagem de programação Java. Em adição duas tecnologias chaves de desenvolvimento, que são parte do JRE: *Java Plug-in*, que permite *applet* de funcionar em browsers populares; *Java Web Start*, que organiza e prepara aplicações autônomas sobre a rede. (Sun Microsystem, 2007b, tradução nossa).

Java Plug-in tecnologia desenvolvida pela Sun Microsystem, incluída como parte do JRE e do *Java 2 Standard Edition* (J2SE). Tecnologia essa que permite estabelecer uma conexão entre browsers populares e a plataforma de Java. Esta conexão permite o *applet* ser executado dentro de um browser. (Sun Microsystem, 2007b, 2007c, tradução nossa).

Um *applet* é um programa escrito na linguagem de programação Java que pode ser incluído em uma página HTML. Quando você usa a tecnologia Java, que permite um browser ver uma página que contenha um *applet*, o código do *applet* é transferido a seu sistema operacional e executado pela *Java Virtual Machine* (JVM). (Sun Microsystem, 2007a, tradução nossa).

Exemplo de um applet. ExemploApplet.class, que pode ser inserido num HTML.

```
import Java.awt.*;
public class ExemploApplet extends Java.applet.Applet {
   public void init() {
   }
   public void paint(Graphics g) {
      super.paint(g);
      g.drawString("Exemplo de applet",25,25);
   }
}
```

Código html invocando o exemplo_Applet.class do applet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
	<head>
	</head>
	<body bgcolor="000000">
		<center>
			<applet
				code     = "ExemploApplet.class"
				width    = "300"
				height= "50"
				>
			</applet>
		</center>
	</body>
</html>
```

A figura 6, demonstra a execução do código exemplo_Applet.html acima descrito.

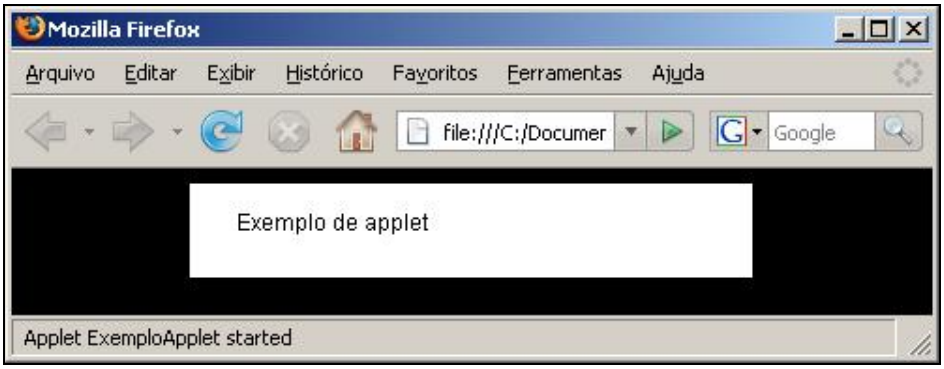


Figura 6 - Exemplo de Applet.
Fonte: Os autores.

Para usar os *applets* como recurso para trazer interatividade às páginas Web é necessário a instalação do JRE, na data do desenvolvimento dessa monografia o JRE 6 *Update* 1, era a versão estável oferecida pela Sun Microsystem. Tecnologia que é instalada no lado cliente.

2.3.8 Flash

O flash é uma aplicação de multimídia que oferece aos *designers* e desenvolvedores a liberdade para criar, apresentações, animações e sites Web ricos e atraentes. Essas aplicações podem ser criadas integrando imagens, desenhos, áudio, vídeo, textos. Arquivos flash podem ser vistos por visitantes on-line de um site Web, desde que o visitante tenha o flash player instalado.

> Flash Player é um pequeno software usado para executar arquivos SWF gerados pelo flash e uma aplicação ‘visualizador’, típica. Os arquivos SWF são executados em uma janela de navegador ou em player independentes; E o player garante que o conteúdo do flash pode ser visto independente da plataforma em que está sendo executado. (DEHAAN, 2004, p. 8).

Código html está invocando o arquivo ExemploFlash.swf dentro do código html: exemplo_flash.html

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
	<title>ExemploFlash</title>
	</head>
	<body bgcolor="#ffffff">
		<!--url's used in the movie-->
		<!--text used in the movie-->
		<!-- saved from url=(0013)about:internet -->
	<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,
0" width="300" height="300" id="ExemploFlash" align="middle">
		<param name="allowScriptAccess" value="sameDomain" />
		<param name="movie" value="ExemploFlash.swf" /><param name="quality"
value="high" /><param name="bgcolor" value="#ffffff" /><embed src="ExemploFlash.swf"
quality="high" bgcolor="#ffffff" width="300" height="300" name="ExemploFlash" align="middle"
allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />
	</object>
	</body>
</html>
```

A figura 7, demonstra a execução do código exemplo_flash.html acima descrito.

Figura 7 - Exemplo de Flash
Fonte: Os autores.

2.3.9 XML

Extensive Markup Language (XML) ou linguagem de marcação extensível é um kit de ferramentas para armazenamento de dados, um veículo configurável para qualquer tipo de informação e um padrão aberto. Permite armazenar e organizar praticamente qualquer tipo de informação em um formato adequado às necessidades (ou regras do negócio de uma empresa). Por ser um padrão aberto não é ligada a uma empresa que a desenvolva de forma isolada e nem acoplada a qualquer software em particular. Oferece maneiras de se verificar a qualidade de um documento, com regras para sintaxe, verificação de vínculo interno, comparação com modelos de documentos e tipo de dados.

Document Type Definition (DTD) declara todos os elementos e atributos de um documento XML, um DTD define quais elementos e atributos são válidos e em que contexto.

A XML foi criada para lidar com o crescente volume de informações geradas pelas empresas, que vem competindo para ter espaço na Web. O objetivo dessas empresas é a troca de dados, mas enfrentam uma barreira as incompatibilidades de seus sistemas de dados.

Como já vimos XML é uma linguagem para conter e gerenciar informações, uma família de tecnologias que pode fazer de tudo, formatar documentos, filtrar dados. XML busca o máximo de utilidade e flexibilidade para os dados, refinando-os a uma forma mais pura e mais estruturada. (RAY. 2001, p. 1 - 2).

Desenvolvedores de sites começaram a forçar os limites do HTML, além da sua capacidade, isso fez tornar-se evidente a necessidade de uma linguagem mais abrangente. O HTML que fornece apenas um conjunto de marcas limitadas para estruturar um documento, sendo assim relativamente fácil utilizá-lo. Para fazer algo mais avançado sem depender de usar marcas específicas de um navegador, os desenvolvedores teriam de usar uma linguagem de script separada, como JavaScript, ou alguma linguagem de script CGI, como PERL, porém essas opções não dão o controle total a estrutura real do documento.

O XML é um subconjunto do SGML é não um Protótipo do SGML como é o HTML, assim ele pode oferecer muitos dos mesmos recursos complexos, mas de forma controlável. O XML usa apenas recursos específicos do SGML que são necessários para publicar informações na internet ou na intranet. Documentos em XML são relativamente fáceis de serem criados e usados na Web, diferente dos documentos SGML. (KIRK; MOULTIS, 2000, p. 15).

Exemplo de um DTD pessoa.dtd

```
<!ELEMENT pessoa (nome, Web?, telefone+)>
<!ELEMENT nome (prenome, inicial*, sobrenome)>
<!ELEMENT Web (email|Website)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT prenome (#PCDATA)>
<!ELEMENT inicial (#PCDATA)>
<!ELEMENT sobrenome (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT Website (#PCDATA)>
<!ELEMENT telefone (#PCDATA)>
```

Exemplo de um xml código pessoa com seus atributos e o arquivo XML pessoa.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pessoa PUBLIC "pessoa.dtd">
<pessoa>
	<nome>
		<prenome>Giordano</prenome>
		<sobrenome>Bruno</sobrenome>
	</nome>
	<telefone>1199343232</telefone>
</pessoa>

<pessoa>
	<nome>
		<prenome>Giordano</prenome>
		<sobrenome>Bruno</sobrenome>
	</nome>
	<Web>
		<Website>www.site.com</Website>
	</Web>
	<telefone>1199343232</telefone>
</pessoa>

<pessoa>
	<nome>
		<prenome>Giordano</prenome>
		<inicial>F</inicial>
		<inicial>R</inicial>
		<sobrenome>Bruno</sobrenome>
	</nome>
	<Web>
		<email>giordano@Web.net</email>
	</Web>
	<telefone>1199343232</telefone>
	<telefone>1134999992</telefone>
</pessoa>
```

2.4 TECNOLOGIAS DE GERAÇÃO DE CONTEÚDO DINÂMICO

Tecnologias de geração de conteúdo dinâmico são linguagens de scripts executadas no próprio servidor. O processo de colocar código em uma página Web para o servidor executar é freqüentemente chamado de *server-side scripting*. Esse termo se refere ao processo em que o servidor Web interpreta a informação ou o código da página Web. O servidor então envia o resultado dessa operação ao navegador Web. O código do script não é enviado ao navegador, apenas o resultado de sua execução e transmitido em forma HTML. (BUYENS, 2002. p. 89).

2.4.1 ASP

O *Active Server Page* (ASP) é uma tecnologia desenvolvida pela Microsoft. Seu objetivo é permitir a criação de páginas e aplicações Web dinâmicas, utilizando script do lado do servidor. (WEISSINGER, 2000. p. 3 - 11).

Exemplo de utilização do ASP. exemplo_asp.asp

```
<%@ LANGUAGE= VBscript %>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <title>Exemplo de página em ASP</title>
    </head>
    <body>
        <h1>Exemplo ASP</h1>
        <p>     A data de hoje: <%= Date() %> hora     <%= time() %>
    </body>
</html>
```

A figura 8, demonstra a execução do código exemplo_asp.asp acima descrito.

Figura 8 - Exemplo de ASP
Fonte: Os autores.

2.4.2 PHP

PHP Hitertext Processor (PHP) é uma linguagem de criação de scripts do lado servidor projetada especificamente para Web. Foi concebido em 1994 com o trabalho de Rasmus Lerdorf, que inicialmente o denominou *Personal Home Page*. É um produto *Open Source*.

PHP possibilita a interação com o usuário através de formulários, parâmetros da URL e links. A diferença entre PHP e outras linguagens semelhantes como JavaScript, é que o código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. (BUYENS, 2002. p. 89).

Exemplo de utilização do PHP. Exemplo_php.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
                <title>Exemplo PHP</title>
        </head>
        <body>
        <h1> Exemplo PHP </h1>
        <?php
                $imprime_data = date("l dS of F Y h:i:s A");
                echo $imprime_data;
        ?>
        </body>
```

```
</html>
```

A figura 9, demonstra a execução do código exemplo_php.php acima descrito.

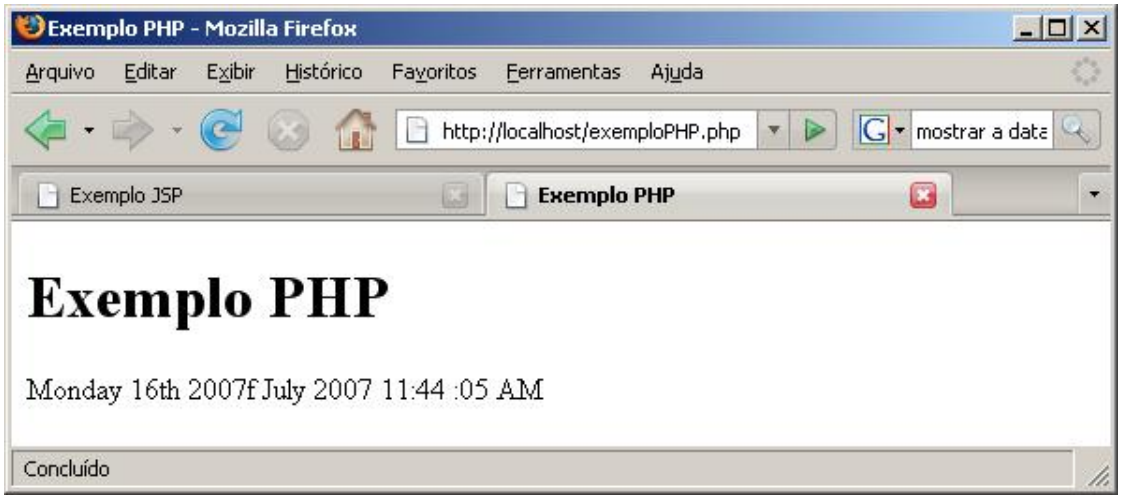


Figura 9 - Exemplo de PHP
Fonte: Os autores.

2.4.3 JSP

JavaServer Pages (JSP) é um a tecnologia desenvolvida pela Sun Microsystems, que faz parte da família Java. JSP fornece uma maneira simplificada, rápida de criar páginas com conteúdo dinâmico. A tecnologia JSP permite o desenvolvimento de aplicações Web básicas, que são servidas independentes de plataforma. A tecnologia traz facilidade de manter as páginas ricas em informações, com o uso de páginas dinâmicas que dão vantagem competitiva em sistemas existentes de negócio. JSP separa as interfaces do usuário da geração do conteúdo satisfatoriamente, permitindo aos *designers* mudar a disposição da página por toda parte sem alterar o índice dinâmico subjacente. (Sun Microsystem, 2007d, 2007e, tradução nossa).

Exemplo de utilização do JSP. Exemplo_jsp.jsp

```
<%@ page language="Java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<%@ page import = "Java.util.Date" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
            <title>Exemplo JSP</title>
        </head>
        <body>
            <h1>Exemplo JSP </h1>
            <p> <%= new Date() %></p>
        </body>
</html>
```

A figura10, demonstrada a execução do código exemplo_jsp.jsp acima descrito.

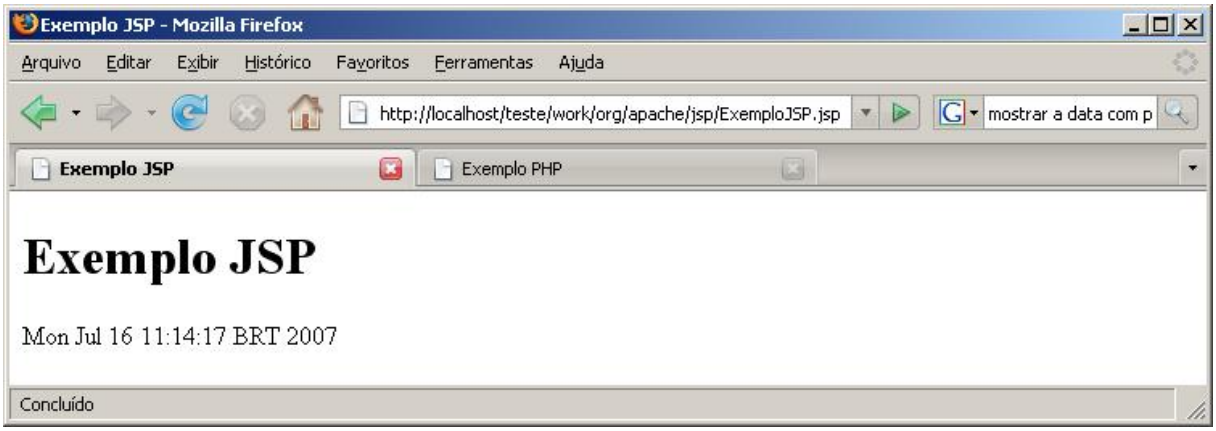


Figura 10 - Exemplo de JSP
Fonte: Os autores.

2.4.4 AJAX

AJAX é referenciado erroneamente em alguns periódicos como uma tecnologia, mas na verdade é uma técnica. Onde um dos seus principais componentes é o JavaScript e XML. Na verdade a técnica não é nenhuma novidade, a tecnologia 'mais recente' relacionado ao tema *XMLHttpRequest* (XHR) o IE 5 (*Internet Explore*) de 1999. Com o controle do Active X.

O AJAX funciona na maioria dos *browsers* modernos e não requer instalação de nenhum software ou hardware patenteado. Sendo está uma das suas vantagens. Outra vantagem seria o fato de ser uma abordagem do lado cliente que pode interagir com outras inúmeras tecnologias de geração de conteúdo dinâmico. Como exemplo as já citadas nesse capítulo JSP, PHP, ASP e outros.

A figura 11 demonstra as requisições de uma página sem o uso do AJAX. Onde o navegador cria uma requisição ao servidor para cada interação do usuário e o servidor retorna todo o conteúdo da página Web para o navegador.

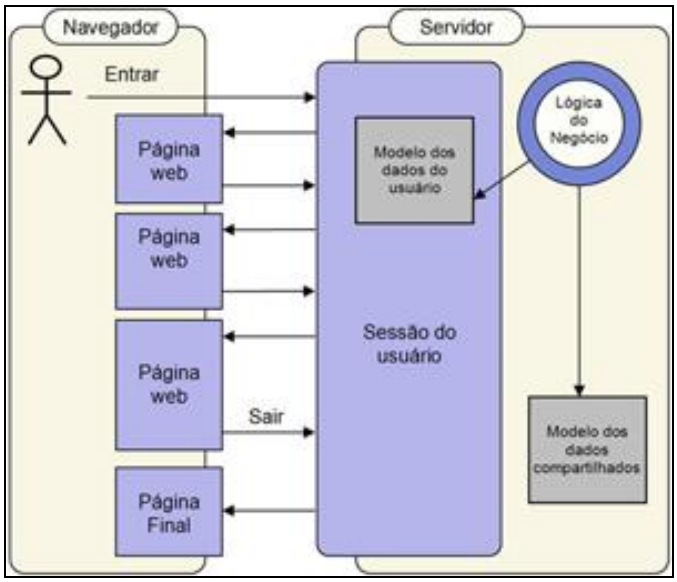


Figura 11 - Fluxo da atualização da Web sem AJAX
Fonte: (CRANE; PASCARELLO, 2006 . p. 18)

A figura 12 ilustra uma aplicação Web com AJAX, onde o navegador cria uma nova requisição ao servidor para cada interação como usuário. O servidor retorna somente os dados solicitados pelo navegador, que com o uso de JavaScript faz o tratamento dos dados recebidos, sem a necessidade de recarregar a página inteira.

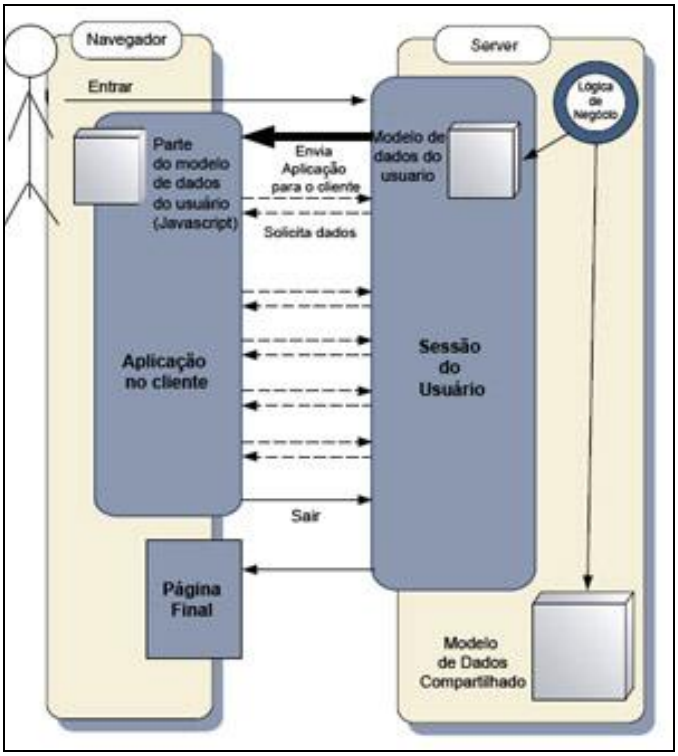


Figura 12 - Fluxo da atualização da Web com AJAX
Fonte: (CRANE; PASCARELLO, 2006 . p. 19)

2.4.4.1 Origem do termo

A origem do termo AJAX é incerta. No entanto Jesse James Garrett foi o primeiro de cunhou o termo em fevereiro de 2005, no ensaio ‘AJAX: uma nova abordagem para aplicativos Web. Originalmente considerado um acrônimo para *Asynchronous JavaScript* e XML, hoje o termo e usado simplesmente para englobar todas as tecnologias que permitem que o browser se comunique com o servidor sem atualizar a página atual. (ASLESON; SCHUTTA, 2006, p.12 - 14).

2.4.4.2 Usabilidade do AJAX

Antes de falarmos de usabilidade do AJAX vamos definir o que é usabilidade. Pela definição da ISO, usabilidade é a extensão na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico (International Organization for Standardization 9241-11, p. 1, tradução nossa).

Sobre a questão da usabilidade do AJAX em aplicações Web. Ele permite operações assíncronas que é outra de suas vantagens, mas pode se tornar seu maior problema. Aplicativos Web são executados com o paradigma solicitação resposta, mas com AJAX não temos essa limitação, porém com a atualização de parte da página sem a percepção do usuário podem deixá-los confusos e sem a possibilidade de retornar ao passo anterior.

Uma técnica que tem se popularizado com o recurso do AJAX é a *Yellow Fade Techinique* (YFT). Que nada mais, é tornar a parte da página que mudou em amarelo, indicando o que mudou ao usuário. (ASLESON; SCHUTTA, 2006, p.17 - 19).

AJAX torna-se útil na validação de dados no lado cliente por permitir que as regras fiquem apenas em um lugar, no servidor, o que evita duplicação de dados e não expondo-os. (STEIL; CAMARGO, 2005, p.36 - 48).

2.4.4.3 Objeto XMLHttlRequest

Originalmente o objeto XHR (*XMLHttpRequest*) foi implementado no IE 5 como componente do ActiveX pela Microsoft. Atualmente ele está presente na maioria dos navegadores populares. É importante ressaltar que o objeto XHR não é um padrão do W3C, embora grande parte da funcionalidade seja abordada em uma nova proposta o DOM *level* 3 *load and save Specifucation.* Por não se tratar de um padrão, seu comportamento pode deferir de um navegador para o outro.

Usa-se JavaStript para criar uma instância desse objeto. Como já vimos o IE implementa XHR como um objeto do ActiveX e os demais navegadores com JavaScript nativo. O código a seguir mostra como criar uma instância compatível com a maioria dos navegadores existentes. (ASLESON; SCHUTTA, 2006, p.23 - 38).

Exemplo: Criando uma instância do XHR.

```
var xmlHttp;
function createXMLHttpRequest() {
        if (window.ActiveXObject) {
                xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        else if (window.XMLHttpRequest) {
                xmlHttp = new XMLHttpRequest();
        }
}
```

A figura 13 demonstra uma página com AJAX onde um evento DOM é acionado. O evento cria o objeto XHR, que chama um Java *Servlet*. O *Servlet* realiza a interação com a aplicação lógica serializando a resposta com XML, que devolve a resposta para a função *callback* atualizando a página Web.

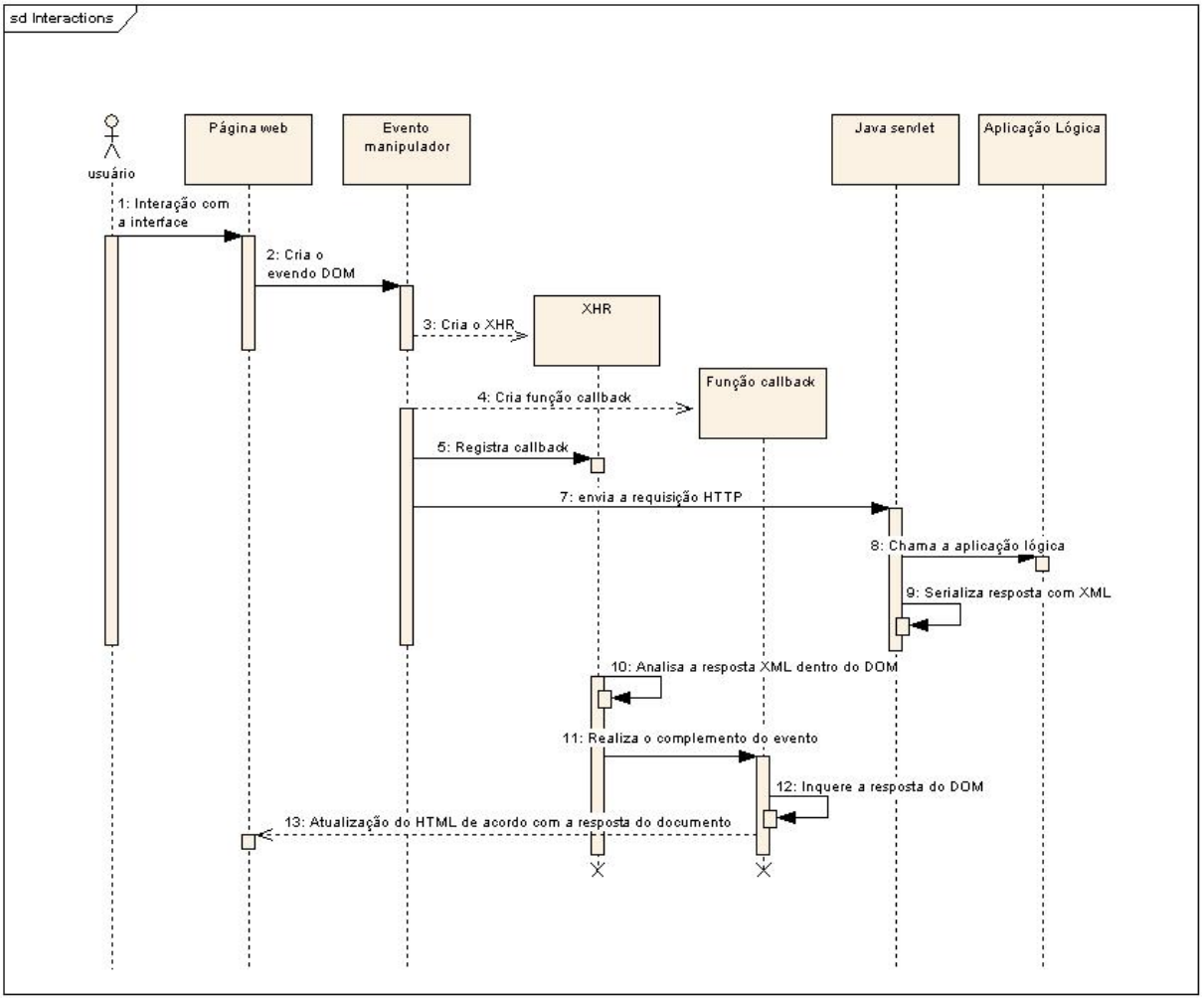


Figura 13 - Diagrama de seqüência do processo de utilização do AJAX e XmlHttpRequest
Fonte: MCCARTHY, 2005. Adaptado.

Graças a tipificação dinâmica do JavaScript e as implementações de XHR serem compatíveis com vários navegadores, pode-se acessar as propriedades e métodos da instância XHR da mesma forma, independente do método usado na criação da instância.

2.4.4.4 Métodos do objeto XHR

O quadro 1 apresentará alguns métodos típicos do objeto XHR.

Método	Descrição
abort( )	Aborta a solicitação atual.
getAllResponseHeaders( )	Retorna todos os cabeçalhos de resposta da solicitação http como pares chave/valor.
getResponseHeader(“header”)	Retorna o valor string do cabeçalho especificado

open(“method”,”url”)	Configura o estágio de uma chamada ao servidor. O argumento de método pode ser GET, POST ou PUT. O argumento de url pode ser relativo ao absoluto. Esse método inclui três argumentos opcionais.
Send(content)	Envia a solicitação para o servidor
setRequestHeader(“header”,”value”)	Configura o cabeçalho especificado com o valor fornecido, open( ) deve ser chamado antes da tentativa de configuração de qualquer cabeçalho.

Quadro 1 - Operações padrão de XHR
Fonte: ASLESON; SCHUTTA, 2006, p. 25.

2.4.4.5 Propriedades do objeto XHR

O quadro 2 apresentará algumas propriedades do objeto XHR.

Método	Descrição
onreadystatechange	O manipulador de eventos que é acionado a cada mudança de estado, normalmente uma chamada a uma função JavaStript.
readyState	O estado da solicitação. Os cinco valores possíveis são 0 = não inicializada, 1 = carregando, 2 = carregada, 3 = interativa e 4 = concluída.
responseText	A resposta do servidor na forma de uma String
responseXML	A resposta do servidor em formato XML. Esse objeto pode ser analisado e examinado como um objeto DOM.
Status	O código de status http do servidor (isso é, 200 para ok, 404 para não encontrado).
statusText	A versão em texto do código de status http (isso é, ok ou Not found).

Quadro 2 - Propriedades do objeto XHR
Fonte: ASLESON; SCHUTTA, 2006, p. 25.

2.4.4.6 Interação com AJAX

Anteriormente se falou sobre interação, aqui se demonstra a interatividade obtida com AJAX. Diferente da abordagem de solicitação/resposta padrão encontrada em cliente Web, comum, o aplicativo AJAX faz as coisas de maneiras um pouco distintas. Quando se cria e configura um objeto XHR que faz mudanças no visual do navegador trazendo ao mesmo a

interatividade perdida anteriormente sem o uso de AJAX. (ASLESON; SCHUTTA, 2006, p. 27 - 28).

> Devido a essa característica, trafego de informações entre o navegador e o servidor Web é menor, pois diferentemente do que acontece numa aplicação sem o uso de AJAX, o conteúdo HTML inteiro não é descarregado a cada nova reaquisição, mais sim apenas os dados necessários a cada interação. SENGER, VILLELA, 2007 . p. 45).

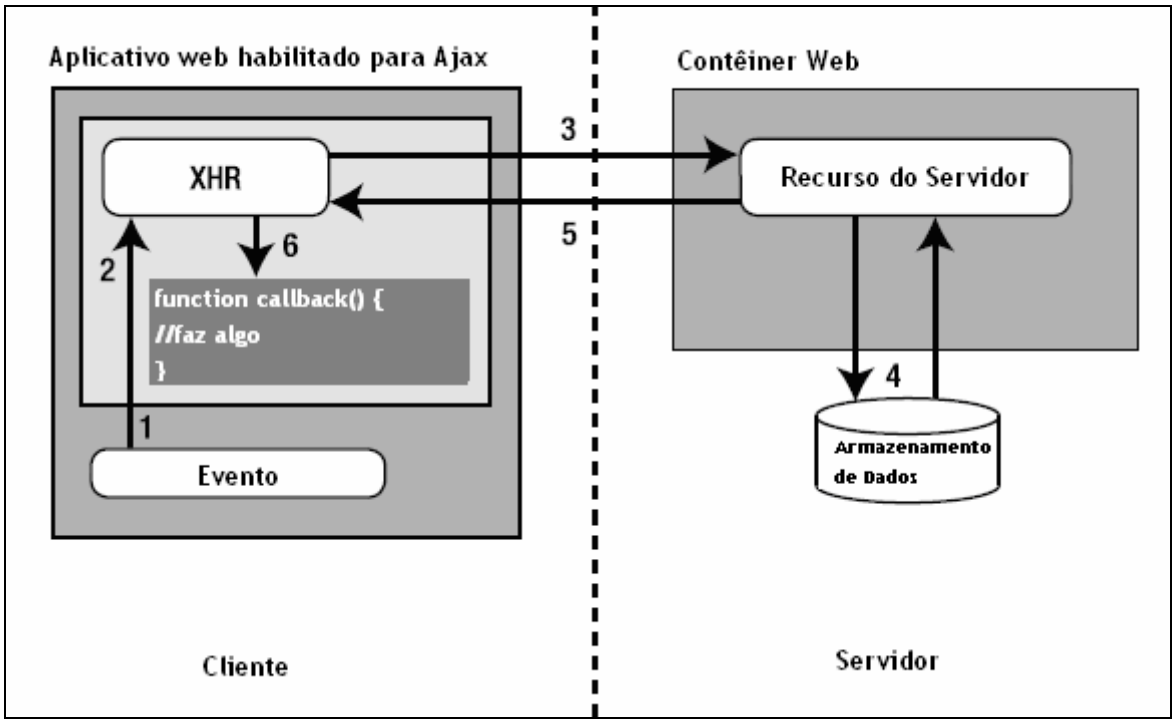


Figura 14 - Interação com AJAX.
Fonte: ASLESON; SCHUTTA, 2006, p. 27.

Detalhamento da figura 14.

1 Um evento qualquer no lado cliente dispara o evento AJAX, por exemplo alguma ação específica do usuário;

2 É criado uma instanciado do objeto XHR;

3 Ocorre uma solicitação ao servidor, por exemplo um *Servlet* ou script CGI.

4 Servidor acessa um Banco de dados.

5 A solicitação é retornada para o navegador. O tipo de conteúdo é texto/xml

6 No exemplo da figura 14 o objeto chama a função *callback( )* quando o processamento do retornar do servidor para o cliente.

Formas de interação de uma aplicação AJAX:

Uma aplicação AJAX possui várias formas de interagir com o servidor. A forma mais comum, inclusive e a mais utilizada por inúmeros *frameworks*, é a troca de dados. Outra forma comum de interação é a troca de conteúdo HTML com o servidor. A forma não tão

comum, mas também possível, é a troca de código em JavaScript para ser dinamicamente estruturado. As aplicações AJAX que trocam dados com o servidor costumam disponibilizar, através de uma URL, o acesso a uma função que recebe os parâmetros da requisição HTTP e retorna as informações em um formato específico (normalmente XML ou JSON). (GUERRA, 2007, p 36 – 43).

2.4.4.7 Framework

Frameworks são aceleradores de código que implementam soluções genéricas para um determinado problema. Baseiam-se em componentes reutilizáveis e flexíveis, oferecem um ponto de partida para a implementação do projeto, afirma (ROCHA, 2003).

Um *framework* fornece aos desenvolvedores um conjunto de componentes estruturais que têm as seguintes características:

- São conhecidos por funcionarem bem em outras aplicações.
- Estão prontos para serem usados com o próximo projeto.
- Pode também ser usados por outras equipes na organização.

Afirmações feitas pelo (HUSTED, 2004).

A seguir apresentam-se alguns dos *frameworks* utilizados para desenvolver aplicações com AJAX.

DWR

O *Direct Web Remoting* (DWR) é um *framework* que facilita a utilização de AJAX em sistemas Java para Web. O DWR se destaca por focar no acesso a recursos remotos Java no servidor Web. Ele permite que o navegador faça uso de código Java sendo executado remotamente no servidor de uma maneira rápida e prática como se estivesse no próprio navegador.

Para o desenvolvedor criar classes Java que são acessadas remotamente, torna-se transparente e simples como programar uma classe *Plain Old Java Object* (POJO), onde o DWR através de arquivos de configurações se encarrega de oferecer os recursos necessários para isso.

DWR divide-se em duas partes principais:

- ✓ Um *Servlet* Java executado no servidor que processa todas as requisições que irão se comunicar com o DWR.

- ✓ Código JavaScript executado pelo navegador que envia requisições e altera as páginas com o resultado recebido das requisições.

Atualmente, a versão de desenvolvimento na data do DWR é a 2.0RC2, mas a versão estável é a 1.1.4 do DWR. (MORO, 2007, p 26- 35).

> O DWR é um dos mais conceituados *frameworks* AJAX disponíveis para a plataforma Java. Tudo funciona de maneira simples e de fácil aprendizado. A estrutura do código fonte Java fica acessível ao cliente via JavaScript, não existindo uma distinção entre lado cliente e servidor do ponto de vista do desenvolvedor. Tudo é feito de forma transparente, pois o DWR cuida de todo processo de criação e manipulação do XMLHttpRequest. Esse produto possui código aberto. (LIMEIRA, 2007, p. 25).

GWT

O *Google Web Toolkit* (GWT) é um *framework* para AJAX. O seu objetivo é esconder do programador a implementação de código JavaScript, considerando que essa tarefa é repetitiva e propensa a erros e repleta de tarefas tediosas, como o tratamento de incompatibilidades entre navegadores. O GWT abstraí a linguagem JavaScript a partir de uma biblioteca de classes Java, na qual está disponível uma coleção de componentes visuais AJAX já conhecidos dos que usam as aplicações AJAX do Google, a exemplo o Gmail e Google Maps.

A principal novidade apresentada pelo GWT é o uso de classes Java para representar os componentes visuais da aplicação. Essas classes são convertidas em código JavaScript por um compilador distribuído no pacote do GWT. O uso de código JavaScript é vantajoso, por permitir que desenvolvedores utilizem a IDE Java de sua escolha, uma vez que o GWT é independente do ambiente de desenvolvimento.

O *framework* tem um modelo de arquitetura em duas grandes camadas. A camada de bibliotecas contém uma biblioteca de componentes visuais e uma biblioteca de emulação Java, que traz a implementação de algumas classes dos pacotes java.lang e java.util. Essa duas bibliotecas compõem a infra-estrutura básica necessária para representar os componentes

AJAX utilizando código Java. A outra camada é composta por duas ferramentas. A primeira é o complicador Java para JavaScript, que é o responsável por gerar o código JavaScript representando os componentes visuais utilizados. A segunda ferramenta é um navegador Web local, que evita ao desenvolvedor tenha de converter o seu código Java em JavaScript cada vez que deseje realizar teste. (RODRIGUES, 2006, 64 - 67).
Existem versões do GWT para Linux e Windows, a versão disponível na data é a 1.3.

ECHO2

Echo2 um *framework* para AJAX. Trata-se de uma plataforma para desenvolver as aplicações Web-baseadas que aproximam as potencialidades de clientes ricos. A versão 2.0, a estável na data tem as preensões de fornecer o desempenho, a potencialidade, e realces dramáticos ao usuário-experiência através do *engine* AJAX-baseado. Echo2 remove o colaborador de ter que pensar nos termos de aplicações "página-baseadas" e permite ao desenvolvedor desenvolver aplicações usando o paradigma Orientado a Objetos convencional para o desenvolvimento da relação de usuário. O conhecimento do HTML, do HTTP, e do JavaScript não é requerido. As aplicações podem ser hospedadas usando todo o recipiente do *Servlet* de Java. Echo2, como seu predecessor, é software da livre distribuído sob os termos da licença pública GNU LGPL. O Echo2 é baseado em *templates* HTML, não necessitando que o desenvolvedor saiba HTML ou JavaScript, apenas tenha conhecimento em Java. O desenvolvimento fica muito parecido com o feito utilizando o pacote *Swing* do Java. (NEXTAPP, 2007).
A versão estável na data era a 2.0 do Echo2.

ATLAS

A Microsoft está há muito tempo envolvida com o AJAX, afinal inventou o objeto XHR e o vem usando no seu desenvolvimento de aplicativos para Web. A Microsoft está dedicando-se a tornar as coisas mais fáceis para os desenvolvedores fornecendo o um ambiente de desenvolvimento robusto, ela quer distribuir muitos recursos, inclusive uma estrutura de criação de scripts no lado cliente, controles ASP .NET e a integração de serviços Web. (ASLESON, 2006, p.259). Para isso lançou o projeto Atlas que é um *framework* da Microsoft para ser integrado ao ASP.NET. O Atlas possui uma biblioteca que contém vários componentes prontos. Ele permite ao desenvolvedor facilmente incorporar no site funções de

drag-and-drop, menus dinâmicos, *popup* com informações temporárias como exemplo: calendário, controle de painéis na página, blocos sempre visíveis mesmo com rolagem da página, controle de campos *dropdown* em cascata (alterando um campo atualiza os demais), controle de animações. (LIMEIRA, 2006, p. 24).

AJAXLIB

É uma classe escrita em JavaScript e que pode ser utilizada em conjunto com várias Linguagens para Web, como PHP, PERL e JSP. É uma ferramenta simples e indicada para aplicações que não requerem muitos recursos. AJAXLib fornece uma maneira fácil executar o objeto XHR tornado assim fácil aplicar os recursos providos pelo AJAX. Esse produto possui código aberto. (SOURCEFORGE, 2007).

DOJO

O DOJO é o um *framework* mais antigos, seu desenvolvimento foi iniciado em setembro de 2004 e hoje é considerado um dos produtos mais maduros. Para desenvolvimento com AJAX. O DOJO é independente de plataforma. Seu foco está no desempenho, por isso é recomendado se o uso de JavaScript for intenso. Possui uma API complexa e sua documentação não é muito completa. Um recurso que o diferencia dos demais *frameworks* é seu suporte aos botões Voltar e Avançar, possibilitando registrar um método de retorno de chamada que será acionado se o usuário clicar no botão Voltar ou Avançar. Esse produto possui código aberto. (ASLESON, 2006, p.254).

libXmlRequest

O libXmlRequest é um *framework* para AJAX. Assim como o DOJO, ele está entre os mais antigos, foi laçado originalmente em 2003. Composto de um único arquivo JavaScript que age como encapsulador do objeto XHR expondo duas funções de solicitação sobrecarregadas *getXml* e *postXml*. Os seus atributos lidam com a formação de *pools*, armazenamento em cachê e várias funções utilitárias manipulando tarefas comuns como a análise do XML do servidor e a alteração do DOM. (ASLESON, 2006, p.254).

SAJAX

O *Simple AJAX Toolkit* (SAJAX) é um *framework* para AJAX, que permitirá ao desenvolvedor chame diretamente códigos do lado do servidor a partir de seu código JavaScript. Ele da suporte a várias linguagens como PHP, ASP, ColdFusion, Perl, Python, Ruby entre outros, mas não tem suporte a Java. (ASLESON, 2006, p.258)

2.4.4.8 MVC com AJAX

Model-View-Control este é o método que a maioria dos desenvolvedores utiliza em seus projetos. (CRANE; PASCARELLO, 2006, p. 91).

Com a utilização do método MVC, tornou-se mais refinado o desenvolvimento do código fonte das aplicações Web.

As camadas no processo de MVC na Web:

- A camada *View*

  Em uma aplicação JavaScript o *browser* é a *View*, a página visível, de acordo com a MVC. Permitir uma interface ao usuário, permitindo a realização de eventos, comunicando com o Controlador. Precisa também se comunicar (solicitação/resposta) com a *Model* por meio da *Control*.

  A *View* seria a fonte de trabalho dos artistas gráficos e programadores, onde um ficaria dando opinião em cima do trabalho do outro.

  Por isso, a utilização do modelo MVC na estruturação dos projetos, torna-se mais fácil a elaboração de páginas com CSS, HTML e JavaScript definidos em arquivos separados. (CRANE; PASCARELLO, Eric. 2006, p. 124 - 128).

- A camada *Control*

  O trabalho da *Control* claro seria a intermediação entre o *Model* e a *View*, encaixando essas camadas uma nas outras. A Camada *Control* é formado pelos manipuladores de eventos. Como no Desktop. Com a evolução das técnicas os navegadores suportam dois modelos de eventos diferentes. O modelo clássico que é relativamente simples, que está em processo de substituição pela recente especificação da W3C para manipulação de eventos. O problema vem na hora de escrever, a

implementação do novo modelo de manipulação de eventos varia de *browser* para *browser*, tornando um pouco problemática.

O exemplo da problemática tem o desenvolvimento para o *browser* Mozilla o evento *callbacks* fica assim: *addEventListener()* e para remover *removeEventListenet()* e no *Internet Explore* fica: *attachEvent()* e *detachEvent()*. (CRANE; PASCARELLO, 2006, p. 134 - 137).

- A camada *Model*

  É responsável por representar a camada de negócio da aplicação. Um contexto no mundo real com que a aplicação está inserida. O DOM não é o modelo utilizado na escala da aplicação agora. O modelo aqui é representado por uma coleção de códigos escritos em JavaScript. Como na maioria dos projetos em MVC ele é baseado na orientação a objeto. (CRANE; PASCARELLO, 2006, p. 143).

2.4.4.9 Segurança AJAX

A funcionalidade e interatividade em aplicações Web trazidas com o uso de AJAX aumentaram bastante a demanda pelo uso dessa técnica no mercado. Infelizmente, o uso da mesma aumenta a exposição de uma aplicação Web, aumentando dessa maneira a possibilidade de ataques. Aplicações Web com AJAX têm sido cada vez mais desejadas por clientes ávidos por ver interatividade e funcionalidade em um ambiente Web. O que esses clientes esquecem é que a segurança dos dados que estão por trás da aplicação é imprescindível independentemente de quão interativa a mesma seja a aplicação. A grande diferença de uma aplicação Web tradicional para uma aplicação Web com AJAX é que enquanto a Aplicação Web tradicional acessa o servidor só para buscar conteúdo, enquanto uma aplicação Web com AJAX pode buscar dados e scripts para a execução. Essa diferença faz com que as aplicações AJAX estejam mais expostas a ataques do que as aplicações Web tradicionais. Os ataques a uma aplicação se da de diferentes maneiras dependendo da forma na qual a aplicação AJAX faz a interação com o servidor.

Na forma de troca de dados o maior risco é superexposição de métodos de negócios que podem ser explorados de forma maliciosa por atacantes. Na de troca de conteúdo HTML os riscos são os mesmos de uma aplicação Web tradicional. Na de troca com código

JavaScript sendo executado dinamicamente, a principal vulnerabilidade é à injeção de fragmentos de script nos parâmetros.

No quadro 3, mostraremos mais detalhes sobre os possíveis tipos de interação e as principais vulnerabilidades de cada um. (GUERRA, 2007, p. 36 – 37).

Interação	Descrição	Possíveis ataques
Dados	A aplicação AJAX busca no servidor dados em formato XML, JSON ou texto plano e utiliza os mesmos para modificar dinamicamente a página HTML utilizado JavaScript	* O atacante acessa a URL utilizada para obter dados da aplicação AJAX passando diferentes parâmetros visando obter informações cujo acesso não era autorizado. *O atacante, de posse dos scripts que irão processar a informação vinda do servidor, envia dados para o servidor que fará que o processamento dos dados tenha algum efeito malicioso em outros usuários.
Conteúdo	A aplicação AJAX acessa uma URL que retorna um fragmento de conteúdo em formato HTML, e esse conteúdo é colocado dentro de uma tag ou .	* São enviados parâmetros maliciosos de forma a interferir no conteúdo que será gerado dinamicamente. Isso pode ser utilizado para roubar a sessão de um usuário e passar pela autenticação da aplicação. * Quando o conteúdo vem acompanhado de algum script para ser executado dinamicamente, se aplicam os mesmos ataques de interação via script.
Script	É gerado dinamicamente um script para ser executado pela aplicação AJAX quando esse for recebido do servidor. A execução é feita pela função eval().	* Juntamente com os parâmetros, são enviados fragmentos de script, para que o código para ser executado no browser seja gerado de forma alternada, permitindo a inserção de comandos maliciosos visando, por exemplo, seqüestrar a sessão do usuário.

Quadro 3 - Ataques possíveis em uma aplicação AJAX.
Fonte: (GUERRA, 2007, p. 37).

Um fator que favorece o atacante é a posse de todo código que é executado no cliente. O acesso a todo JavaScript da aplicação ajuda bastante na procura de falhas a serem exploradas, a fim de obter informações não autorizadas. Uma regra interessante é o uso de JavaScript apenas para acessar as regras de negócio que estão no servidor. Qualquer regra de negócio que estiver totalmente no cliente pode ser facilmente burlada. Uma forma de proteger o código em JavaScript contra leitura é ofuscar, mas o atacante tem acesso ao código mesmo que o esteja embaralhado. Mesmo o atacante tendo acesso é melhor do que deixa-lo aberto.

As aplicações AJAX que trocam dados com o servidor costumam disponibilizar, através de uma URL, o acesso a uma função que recebe os parâmetros da requisição HTTP e retorna as informações em formato XML ou JSON. O problema é que essa URL não estará disponível só para a aplicação, mas para também para quem desejar acessá-la indiscriminadamente. O que faz necessário que as aplicações AJAX terem um controle de

acesso robusto, para que os atacantes não façam uso dessas funcionalidades para obterem acesso a informações não autorizadas.

Na parte da validação de dados, essas precisam ser rápidas e práticas ao usuário, mas não pode ser burlada pela aplicação. Fazer uma validação de dados somente no navegador através do JavaScript, faz com que o tempo de resposta diminua, mas trata-se de uma prática perigosa que fere os requisitos de segurança e consistência dos dados. No quadro 4, segue a demonstração das opções que um desenvolvedor possui para fazer a validação de formulários em uma aplicação Web. (GUERRA, 2007, p. 37).

Interação	Descrição	Produtivo	Responsivo	Seguro
Cliente	O formulário HTML passa por uma validação em JavaScript e nem envia os dados para o servidor quando acontecer algum problema.	✓	✓	✗
Servidor	Os dados são enviados para o servidor e antes da requisição ser processada os dados passam por uma validação.	✓	✗	✓
Servidor e Cliente	O formulário recebe a validação em JavaScript antes de sair do cliente e outra validação ao chegar ao servidor.	✗	✓	✓
AJAX	O AJAX é utilizado para enviar as informações do formulário para serem validadas no servidor, sem que isso fique aparente para o usuário.	✓	✓	✓

Quadro 4 - Opções de validação em uma aplicação Web.
Fonte: (GUERRA, 2007, p. 38).

A validação para ser segura precisa acontecer no lado servidor, surge o problema que para validar os dados seria necessário recarregar a página. Para cada tentativa de envio de dados inválidos, a aplicação vai ao servidor e não apenas valida os dados, mas também recarrega do banco de dados todas as informações e gera uma nova página HTML. O processo faz com que a aplicação não pareça responsiva ao usuário, e gera uma carga adicional desnecessária ao servidor. O uso de AJAX resolve o problema de responsabilidade e segurança com a validação acontecendo no servidor a partir de uma requisição assíncrona.

Permite dessa forma uma validação implementada apensas uma vez, não havendo perda em produtividade devido à duplicação de lógica no cliente e no servidor. A performance também é otimizada, pois não existe mais a necessidade de recarregar as informações ou gerar o HTML das páginas novamente. (GUERRA, 2007, p. 37).

2.5 APLICAÇÕES DESKTOP X WEB X WEB COM AJAX

A diferença entre o protótipo Desktop, para o protótipo Web é que para o protótipo Web a cada interação com o usuário o mesmo faz requisições ao servidor. O protótipo Desktop trata esses eventos de maneira local pelo sistema operacional.

Desktop x Web

Páginas feitas para rodar em um *browser* podem ser bastante sofisticadas por si próprias, com o uso de tecnologias de geração de conteúdo dinâmico podemos criar protótipos de Desktop para Web, porém ocorre perda de boa parte da elegância e praticidade presentes no protótipo Desktop, pois as mesmas precisam ser recarregadas e buscar informações no servidor, para serem atualizadas. AJAX torna possível, que as páginas sejam mais naturais e pessoais, onde o JavaScript busca no servidor as informações para atualizar os dados que são modificados, como exemplo: Texto de um próxima página de um artigo, as informações complementares de um formulário ou algum processo de salvar informações cadastrais. (STEIL; CAMARGO, 2005, p.36 - 48).

> Uma das diferenças entre programar um aplicativo para Desktop, que fica instalado localmente na máquina do usuário, e um aplicativo Web, é em relação à maneira em como os eventos são tratados. Em um aplicativo Web, toda e qualquer ação do usuário implica em requisições ao servidor, enquanto em aplicativo instalado localmente o retorno é quase instantâneo. (STEIL; CAMARGO, 2005, p.36 - 48).

Web x Web com AJAX

Páginas feitas para rodar em um *browser* podem ser bastante sofisticadas por si próprias, mas perdem boa parte da elegância e praticidade quando as mesmas precisam ser recarregadas para buscar informações no servidor. Com AJAX é possível tornar as páginas mais naturais e pessoais, com JavaScript, que busca no servidor para atualizar os dados que são modificados, como por exemplo: Texto de um próxima página de um artigo, as informações complementares de um formulário ou algum processo de salvar informações cadastrais.

Comparar as aplicações Web com AJAX e sem AJAX observa-se um principio de revolução, surge com o AJAX um grau de interatividade com o usuário o que é uma novidade para Web, diferencia-se a técnica AJAX das tecnologias de geração de conteúdo dinâmico é a maneira com que a técnica realiza as tarefas e interagir com o usuário. Os sites dessa maneiras quebram paradigmas e introduzem novos padrões. (STEIL; CAMARGO, 2005, p.36 - 48).

Nas figuras 15 e 16, temos demonstrado a transações das aplicações Web tradicionais e aplicações Web com AJAX.

> Aplicações Web clássicas funcionam de uma maneira na qual o usuário dispara alguma ação, como clicar em um link ou enviar dados via formulário, cujo requisição é enviada via HTTP ao sistema no servidor, o qual pega e processa os dados, calcular, gravando ou alterando ao e, no final, o resultado da ação – a nova página solicitada ou a confirmação de envio dos dados do formulário volta ao usuário.
> Nesse meio tempo, o usuário não tem o que fazer a não ser esperar alguma resposta do servidor, o que geralmente leva bons segundos e tudo navegação fica com um ar muito pessoal, distante do cliente.
> Pro outro lado, as aplicações AJAX precisão apenas buscar ou processar o pedaço de informação essencial, eliminando todo o *overhead* de reprocessar a página inteira novamente. Pela natureza assíncrona do JavaScript, na maior parte dos casos o usuário não precisa ficar esperando pelo retorno do servidor para continuar a navegação. (STEIL; CAMARGO, 2005, p.36 - 48).

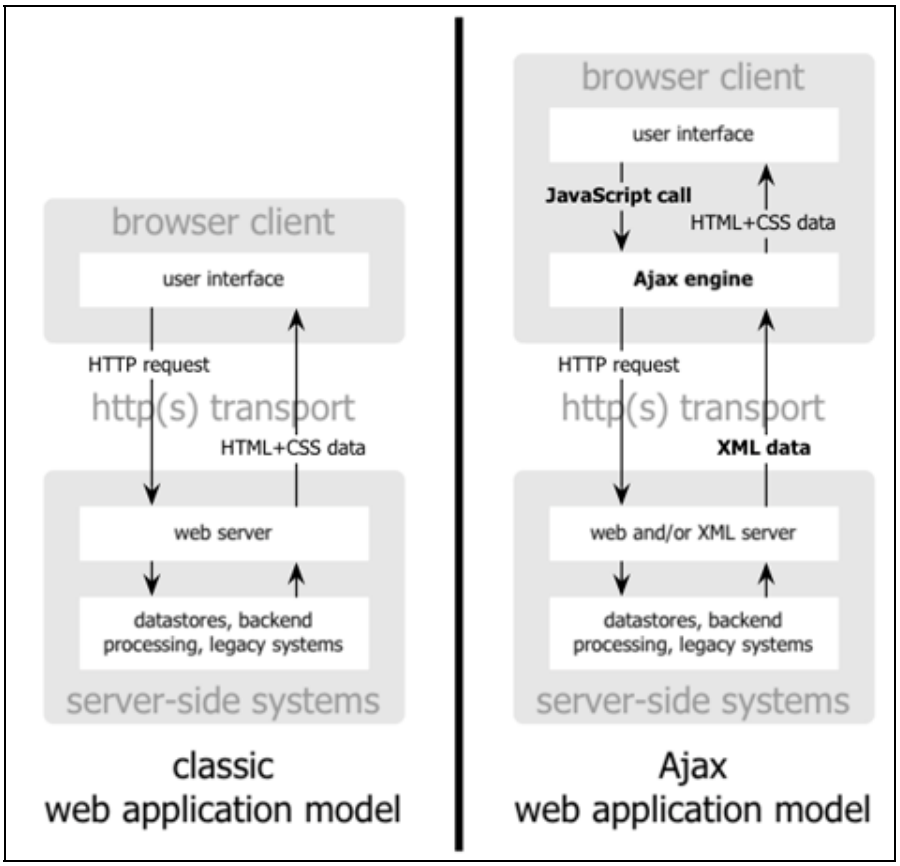


Figura 15 - Diferença da Web clássica para Web com AJAX
Fonte: AJAX: A New Approach to Web Applications, 2005.

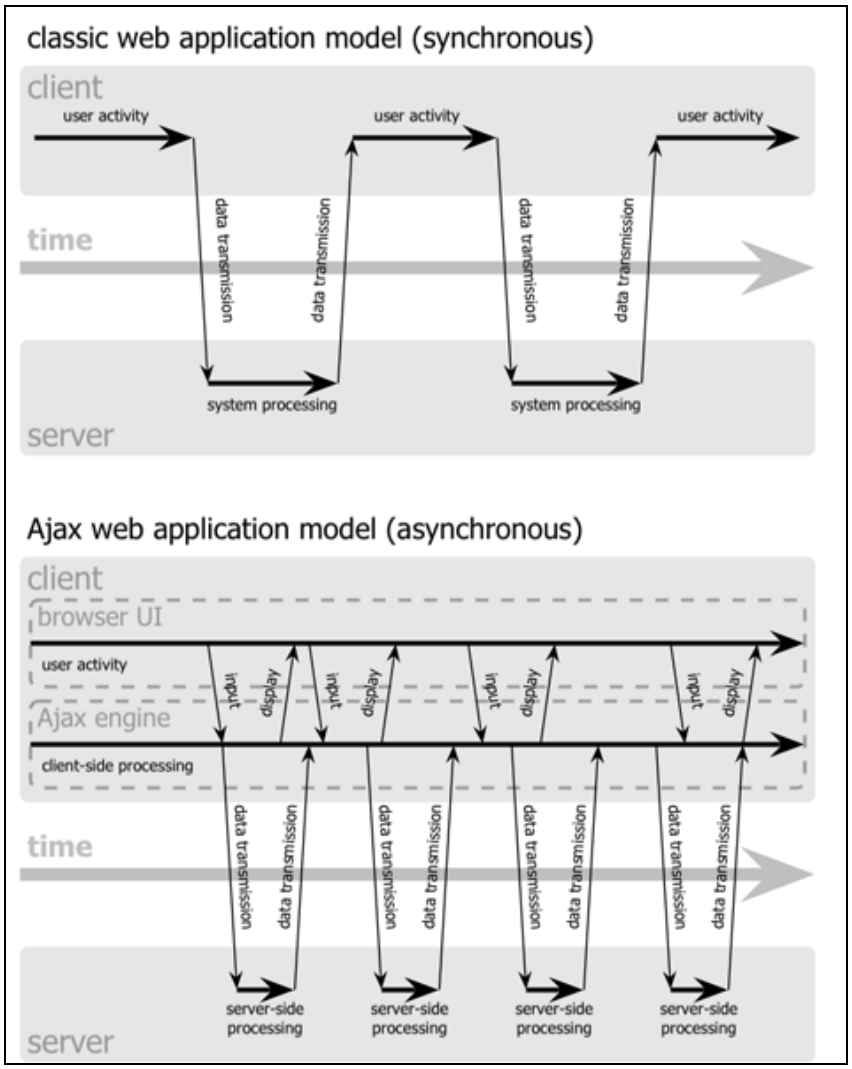


Figura 16 - AJAX evita o overheard
Fonte: AJAX: A New Approach to Web Applications, 2005.

2.6 PARADIGMA DA ORIENTAÇÃO A OBJETO (POO)

Atualmente, um grande número de linguagens incorpora características do POO, como Smalltalk, Perl, Python, Ruby, PHP 5, ColdFusion, C++, Object Pascal, Java, JavaScript, ActionScript e C#.

O paradigma da orientação a objetos (POO) é uma maneira de construir programas de computador que espelham o modo como os objetos são montados no mundo físico, permitindo o desenvolvimento de programas mais flexíveis, reutilizáveis e inteligentes. O POO é baseado no modo como, no mundo físico os objetos são constituídos de muitos objetos menores. Com o POO parte da criação de classes com um modelo abstrato, que instanciado,

dão novas características ao objeto. Ao escrevem um projeto, constrói se um conjunto de classes. Quando o programa é executado, os objetos são instanciados e a partir dessas classes e usados de acordo com as necessidades do projeto. (GONÇALVES, 2006, p. 53 – 54).

Para um melhor entendimento veremos o significado de paradigma. Segundo o dicionário Aurélio a definição de paradigma é:

> [Do gr. parádeigma, pelo lat. tard. paradigma.] S. m. 1. Modelo, padrão, estalão: 2. Termo com o qual Thomas Kuhn (v. kuhniano) designou as realizações científicas (p. ex., a dinâmica de Newton ou a química de Lavoisier) que geram modelos que, por período mais ou menos longo e de modo mais ou menos explícito, orientam o desenvolvimento posterior das pesquisas exclusivamente na busca da solução para os problemas por elas suscitados. (FERREIRA, 1999, p. 1494).

Essencial ao desenvolvimento atual de sistema de software o paradigma da orientação a objetos. É uma forma de se abordar um problema. Alan Kay, um dos pais do POO, formulou a chamada ‘analogia biológica’. Analogia Biológica seria como desenvolver um sistema de software que funciona como um ser vivo. No sistema cada ‘célula’, é uma unidade autônoma, interage com ouras células através de mensagens realizando dessa maneira um objetivo comum.

Alan Kay estabeleceu os seguintes princípios para o Paradigma de Orientação a Objetos (OO). Qualquer coisa é um objeto. Objetos realizam tarefas através de requisições de serviços a outros objetos. Cada objeto pertence a uma determinada classe. Uma classe agrupa objetos similares. A classe é um repositório para comportamento associado ao objeto.
Classes são organizadas em hierarquias.

O POO visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas especificas. É Através da interação entre objetos que uma tarefa computacional é realizada.

Um sistema de software OO consiste de objetos em colaboração com o objetivo de realizar as funcionalidades desse sistema. Cada objeto é responsável por tarefas especificas. É Através da cooperação entre objetos que a computação do sistema desenvolve. (BEZERRA, 2003, p. 4 - 7).

2.6.1 Descrevendo alguns conceitos do POO

2.6.1.1 Objetos

Para começar a falar em objetos vamos dar uma definição do que é um objeto. Segundo o dicionário Aurélio a definição de objeto é:

> [Do lat. objectu, part. de objicere, 'pôr, lançar diante', 'expor'.] S. m. 1. Tudo que é apreendido pelo conhecimento, que não é o sujeito do conhecimento. 2. Tudo que é manipulável e/ou manufaturável. 3. Tudo que é perceptível por qualquer dos sentidos. 4. Coisa, peça, artigo de compra e venda: 2 5. Matéria, assunto: 2 6. Agente; motivo, causa: 2 7. O ponto de convergência duma atividade; mira, desígnio: 2 8. Mira, fim, propósito, intento, intuito, desígnio; objetivo. 9. Filos. Na relação de conhecimento, o correlato do sujeito, isto é, o que é conhecido, em oposição ao que conhece. [Cf., nesta acepç., sujeito (13), conhecimento (10) e teoria do conhecimento.] 10. Filos. O que é real ou realizável e se torna motor da ação de um sujeito.[Cf., nesta acepç., sujeito (11).] 11. Inform. Em programação orientada a objetos (q. v.), qualquer módulo que contém rotinas e estruturas de dados, e capaz de interagir com outros módulos similares, trocando mensagens. 12. Inform. V. elemento de interface. 13. Inform. Programa-objeto (q. v.). 14. Jur. Aquilo sobre que incide um direito, obrigação, faculdade, norma de procedimento, proibição, etc. 15. Ópt. Fonte de luz ou corpo iluminado cuja imagem se pode formar num sistema óptico. (FERREIRA, 1999, p. 1427).

Definição de objeto: Uma coisa apresentada ou capaz de ser apresentada, aos sentidos. Em outras palavras, um objeto é quase qualquer coisa. Um objeto consiste de um conjunto de operações e atributos. (PAGE-JONES, 2001, p. 1 - 11).

Portanto, objetos são coisas que existem no mundo real, como pessoas, animais, que descobrimos observando as suas características (atributos). Por exemplo, um objeto pessoa se pode colocar como atributos a altura, tamanho, cor e seu comportamento (ações), como falar, dormir, andar. Objetos podem ser animados ou inanimados, por exemplo, uma bola, que possui atributos, como diâmetro, cor, peso e ações, como rolar encher, chutar. (LIMA, 2005, p. 20).

2.6.1.2 Abstração

Para começarmos a falar em abstração vamos definir primeiro o que é Abstração. Segundo o dicionário Aurélio a definição de abstração é:

> [Do lat. tard. abstractione.] S. f. 1. Ato de abstrair(-se); abstraimento. 2. Filos. Ato de separar mentalmente um ou mais elementos de uma totalidade complexa (coisa, representação, fato), os quais só mentalmente podem subsistir fora dessa totalidade. [Cf. determinação (6 e 7) e generalização (5).] 3. Filos. O resultado de abstrações (termo, conceito, idéia, elemento de classe, etc.); abstrato. 4. Estado de alheamento do espírito; enleio, devaneio, abstraimento. 5. P. ext. Falta de atenção; distração,

> alheamento; abstraimento. 6. Art. Plást. Obra de arte abstrata. (FERREIRA, 1999, p. 18).

Quando observa se a realidade, separando mentalmente os objetos que nos interessam para o estudo, estamos realizando uma abstração. Exemplo se nós temos em um cenário um ônibus, um carro, um caminhão, uma pessoa, um inseto e uma borboleta. Se o propósito fosse um estudo para desenvolvimento de um sistema de transporte o inseto e a borboleta não seriam relevantes. O papel da pessoa poderia ser de um motorista. (LIMA, 2005, p. 21).
Segundo o dicionário Aurélio a definição de modelar é:

> Modelar 1 [De modelo + -ar1.] Adj. 2 g. 1. Que serve de modelo (7); exemplar.
> Modelar 2 [De modelo + -ar2.] V. t. d. 1. Fazer o modelo (2 e 3) de; representar por meio de modelo. 2. Assinalar os contornos de; ajustar-se a; contornar: 2 3.Dar forma a; afeiçoar: 2 4. Moldar (4). 5. Delinear intelectualmente; traçar, delinear: 2 6. Pint. Reproduzir exatamente os contornos ou o relevo de. V. t. d. e i. 7. Conformar, moldar: 2 2 V. p. 8. Tomar-se por modelo. 9. Moldar-se (8). (FERREIRA, 1999, p. 1350).

Assim, o desenvolvimento com POO modela objetos do mundo real, estudando-os e criando classes a partir de suas características, como nome, cor, altura da pessoa. (LIMA, 2005, p. 21).

2.6.1.3 Classes

A definição de classe para o dicionário Aurélio é:

> [...] 21. Inform. Em programação ou modelagem orientada a objetos (v. orientação a objetos), categoria descritiva geral, que abrange o conjunto de objetos que compartilham uma ou mais características quanto a seus itens de dados e procedimentos associados. [...] (FERREIRA, 1999, p. 484).

Classe é uma instância a partir do qual são criados ou gerados os objetos. Cada objeto tem a mesma estrutura e comportamento da classe na qual ele teve origem. As diferenças entre os objetos originados de uma mesma classe são o identificador e o estado do objeto, significa que objetos têm ‘valores’ diferentes armazenados em suas variáreis. (PAGE-JONES, 2001, p. 27 -28).

> Ao comparar diferentes objetos, notamos que eles possuem atributos e comportamentos semelhantes. Usando abstração, podemos agrupar objetos observando suas características e comportamentos comuns. [...] Uma classe é a definição dos atributos e as funções de um tipo de objeto; ela descreve um conjunto de objetos individuais em qualquer contexto. Ela é obtida pela classificação de objetos com a mesma estrutura de dados e mesmo comportamento. (LIMA, 2005, p. 22).

A figura 17 representa um exemplo de uma classe com seu nome atributos e as suas operações.

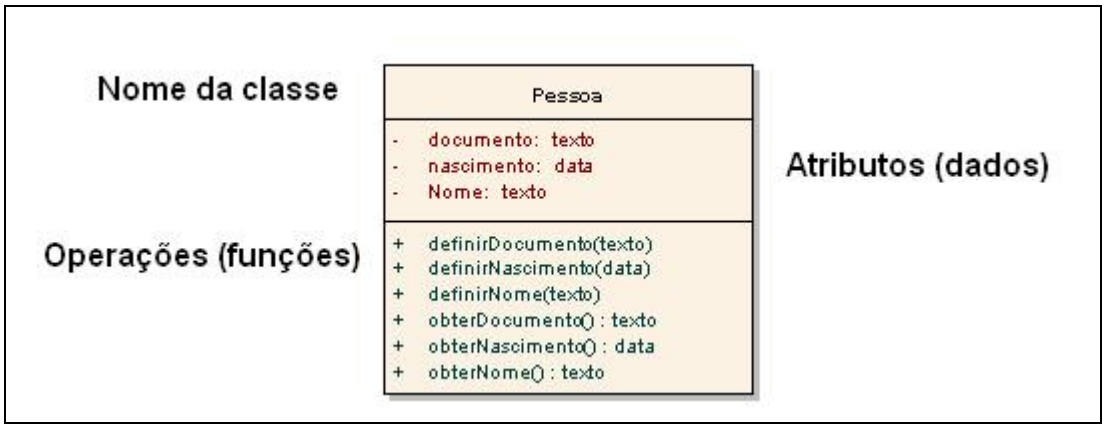


Figura 17 - Representação de classe
Fonte: (LIMA, 2005, p. 22).

Um objeto pertence a uma classe é denominado de instância de classe. Se pegarmos à classe pessoa pode se falar que Artur, Doubleday, Osmar são instancias da classe Pessoa. Observamos que os todos os objetos criados tem características em comum. Os atributos que são: documento, nascimento e nome. A classe Pessoa é um modelo para a criação dos objetos Artur, Doubleday e Osmar criados. (LIMA, 2004, p. 23).

2.6.1.4 Atributos de uma classe

Já falamos de atributo, mas antes de continuar falando em atributos vamos definir primeiro o que é atributo. Segundo o dicionário Aurélio a definição de atributo é:

> [Do lat. attributu.] S. m. Aquilo que é próprio de um ser: & 2. Emblema distintivo; símbolo. 3. Estat. Característica, qualitativa ou quantitativa, que identifica um membro de um conjunto observado. 4. Estat. Atributo homógrado. 5. Filos. Caráter essencial de uma substância. 6. Inform. Na modelagem conceitual (q. v.), cada uma das propriedades que definem um objeto ou entidade. [Em um banco de dados relacional (q. v.), corresponde a cada um dos campos [v. campo (19)] de um registro (20).] 7. Inform. Item de informação indivisível, em arquivo, banco de dados, ou na modelagem conceitual. 8. E. Ling. Termo que caracteriza o significado de uma palavra. 9. Lóg. Caráter afirmado ou negado de um sujeito; predicado, categorema, nota, característica. [Cf., nesta acepç., qualidade (9).] [...] (FERRIERA, 1999, p. 229).

Os atributos são recursos de um classe ou qualquer outro elemento que represente propriedades ou elementos de dados. Os atributos têm algumas características importantes, como visibilidade, nome, tipo de dado e valor inicial.

Quanto à visibilidade, um atributo pode ser: Público: Significa que o atributo é acessível por outras classes é representada pelo sinal +. Privada: Significa que o atributo somente é acessível pela própria classe. Representado pelo sinal -. Protegida: Significa que o atributo é acessível somente pela própria classe e suas subclasses. É representado pelo sinal #.

O tipo de dado e o valor inicial dependem da linguagem de programação utilizada. No exemplo da figura 17 utilizamos dados genéricos, como texto, data e texto. O valor inicial refere-se a um valor definido automaticamente quando o objeto da classe é instanciado (criado). (LIMA, 2004, p. 23).

2.6.1.5 Conceitos Fundamentais ao POO

Existem alguns conceitos de software que fundamentais ao POO são eles: O conceito de classe que acabamos de ver. Veremos outros como: O Encapsulamento, ocultação de informações e implementações, retenção de estado, identidade de objeto, mensagens, herança, polimorfismo e generalização. Estes conceitos independem da linguagem utilizada para implementação dos programas. No POO a ênfase é dada em cima dos dados. (PAGE-JONES, 2001, p. 3).

Encapsulamento: é o agrupamento de idéias afins em uma unidade, conceito esse que pode então ser informado em uma só palavra. (PAGE-JONES, 2001, p. 9). O encapsulamento orientado a objeto é um pacote de operações e atributos o qual o estado em um tipo de objeto, de tal forma que o estado é acessível ou modificável somente pela interface provida pelo encapsulamento. (PAGE-JONES, 2001, p. 11).

A figura 18 demonstra uma classe encapsulada onde ela recebe mensagens.

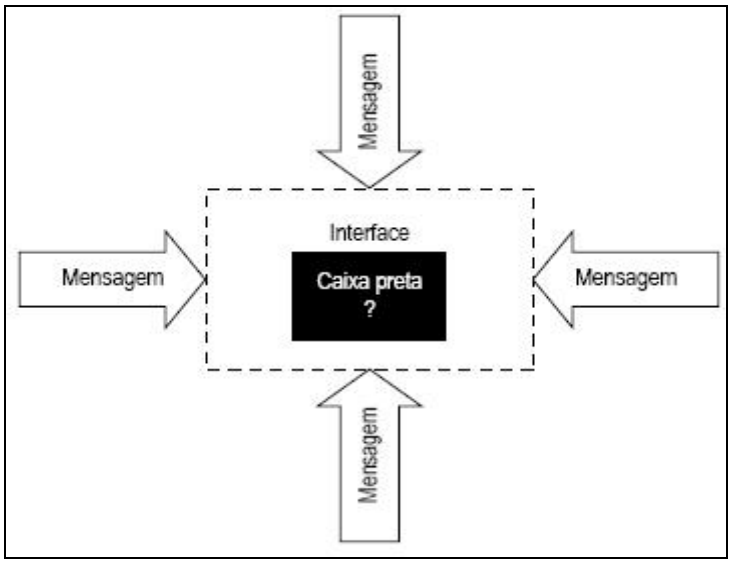


Figura 18 - Exemplo de Encapsulamento
Fonte: (PAGE-JONES, 2001, p. 13).

Ocultação de informações e implementações: é a utilização do encapsulamento para restringir a visibilidade externa de certos detalhes de informações ou implementações, os quais são internos à estrutura de encapsulamento. (PAGE-JONES, 2001, p. 13). Ocultar do usuário o funcionamento interno de uma classe. Desta forma, os atributos de um objeto não serão acessíveis, ou seja, só será possível acessar ou modificar o valor de um atributo de um objeto através de seus métodos.

Retenção de estado: Essa abstração é à habilidade de um objeto reter seu estado. Quando uma função retorna ao seu chamador ou *caller*, só o seu resultado do seu trabalho, após isso a função morre. Quando chamado novamente essa função ele refará a sua função novamente. Estado significa na prática, o conjunto de valores que um objeto consegue manter consigo. (PAGE-JONES, 2001, p. 15).

Identidade de objeto: Essa é a propriedade pela qual cada objeto (independentemente de sua classe ou estado) pode ser identificado e tratado como uma entidade distinta de software. Todo objeto tem sua própria identidade, o mesmo identificador permanece com o objeto por toda a sua vida, independentemente do que aconteça com ele durante esse período. Dois objetos nunca poderão ter o mesmo identificador. (PAGE-JONES, 2001, p. 16 -17).

Mensagens: Objetos solicitam uns ao outros que executem uma atividade via uma mensagem, ou podem também transmitir algumas informações para outro. Uma mensagem é

o veículo pelo qual um objeto remetente transmite a um outro objeto destinatário um pedido para aplicar um dos métodos do objeto destinatário. (PAGE-JONES, 2001, p. 20).

Herança: é a forma de abstração utilizada no POO.

Segundo o dicionário Aurélio herança é:

> [Do lat. haerentia, subst. do neutro pl. de haerens, tis, part. pres. de haerere (2ª conj.), 'estar pegado'; 'aderir'; a infl. de herdar explica a passagem de -ença a -ança.] S. f. 1. Aquilo que se herda. 2. Aquilo que se transmite por hereditariedade (2). 3. Jur. Bem, direito ou obrigação transmitidos por via de sucessão ou por disposição testamentária. 4. Fig. Aquilo que se recebeu dos pais, das gerações anteriores, da tradição; legado. 5. Inform. Em modelagem (3) orientada a objetos (v. orientação a objetos), transmissão automática de determinadas propriedades de uma classe (21) a outra(s) classe(s) dela derivada(s). [Cf. hierarquia (4).] 6. Inform. O conjunto das propriedades e especificações assim transmitidas de uma classe para outra(s). (FERRIRA, 1999, p. 1036).

A herança pode ser vista como um nível de abstração em uma classe entre objetos. Classes semelhantes são agrupadas em hierarquias. Cada nível pode ser visto como um nível de abstração. Esse mecanismo é o responsável direto pelo compartilhamento de comportamento comum entre um conjunto de classes semelhantes. (BEZERRA, 2003, p. 10). Uma classe mãe denominada como superclasse (ou classe mãe) deriva outras classes (classes derivadas ou filhas). (LIMA, 2005, p. 24 – 25). A figura 19 demonstra a classe Pessoa com seus atributos comuns. Que são herdados as classes filhas Aluno e Professor.

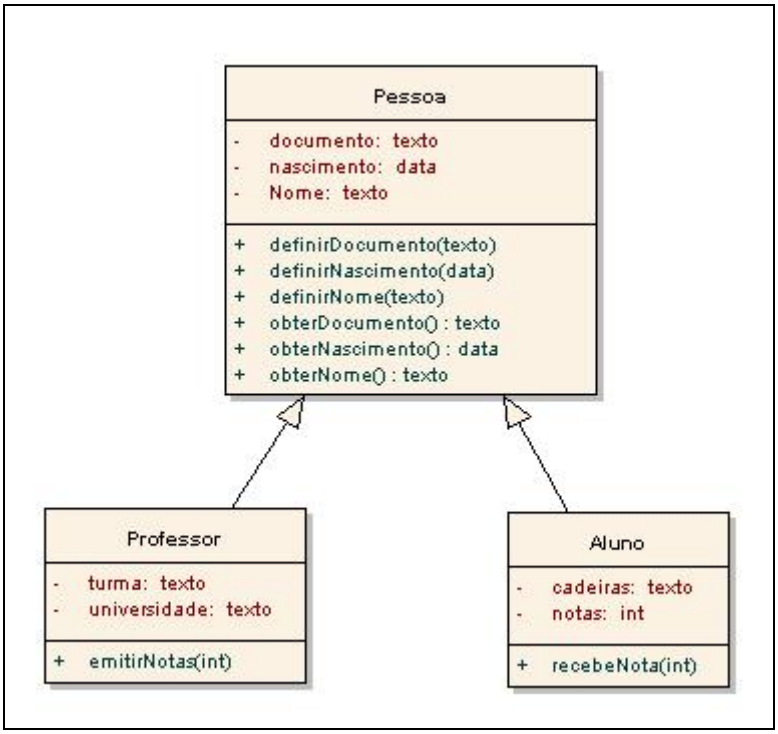


Figura 19 - Exemplo de Herança
Fonte: Os autores.

Polimorfismo: A palavra polimorfismo origina-se de duas palavras gregas que significam respectivamente, muitas e forma. (PAGE-JONES, 2001, p. 38).

Segundo o dicionário Aurélio polimorfismo é:

> [De polimorfo + -ismo.] S. m. 1. Bot. Existência de órgãos ou plantas com diversas formas. [O polimorfismo foliar significa que um vegetal apresenta folhas de vários tipos morfológicos.] 2. Quím. Fenômeno apresentado por substâncias que cristalizam em diferentes sistemas [v. sistema (16)]. 3. Genét. Ocorrência simultânea, na população, de genomas que apresentam variações nos alelos de um mesmo lócus, resultando em diferentes fenótipos, cada um com uma freqüência determinada. 4. Zool. Propriedade que têm certas espécies de animais de apresentarem formas diferentes de acordo com a função a ser desempenhada.[Ocorre, p. ex., nos insetos himenópteros onde há rainha, operárias, soldados, etc.] (FERREIRA, 1999, p. 1598).

Polimorfismo é a habilidade que um método ou atributo pode assumir muitas formas e que a forma adequada é executada dependendo do contexto em que é invocada. Polimorfismo é também o meio que um atributo ou variável pode apontar para objetos de diferentes classes em horas diferentes. (PAGE-JONES, 2001, p. 39).

A figura 20 descrever o exemplo de polimorfismo.

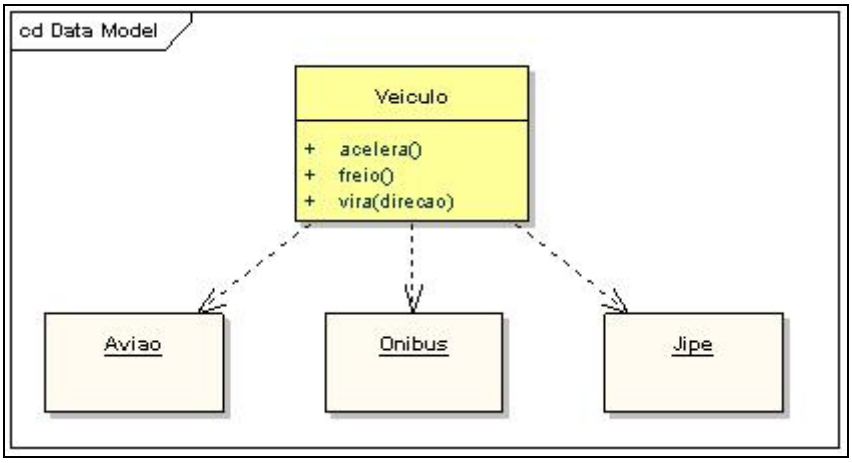


Figura 20 - Exemplo de Polimorfismo
Fonte: Os autores.

Generalização: é a construção de uma classe de forma que uma ou mais classes que ela utiliza internamente é fornecida somente no tempo de execução no qual o objeto é gerado. (PAGE-JONES, 2001, p. 44).

2.7 ENGENHARIA DE SOFTWARE

Segundo Pressman, a Engenharia de Software possui um grupo de três elementos: métodos, ferramentas e procedimentos, os quais tornam possível ao gerente o controle do processo e o desenvolvimento do software, e propõe ao profissional um preparo para a construção de sistemas de alta qualidade. Os métodos detalham "como fazer" para se construir o software, as ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos, e os procedimentos constituem o elo que mantém juntos os métodos e as suas ferramentas, e possibilita um processo de desenvolvimento claro, eficiente, visando garantir ao desenvolvedor e seus clientes, a produção de um software de qualidade. (PRESSMAN, 1995).

2.8 MODELO ENTIDADE DE RELACIONAMENTO

O modelo Entidade Relacionamento (ER) foi definido por Peter Chen em 1976.
O modelo ER tem por base a percepção de que o mundo real é formado por um conjunto de objetos chamados de entidades e pelo conjunto dos relacionamentos entre esses objetos. O modelo ER facilita o projeto do banco de dados, permitindo a especificação do esquema da empresa, que representa a estrutura lógica do banco de dados. O ER é um modelo com a maior capacidade semântica; os aspectos semânticos do modelo se referem à tentativa de representar o significado dos dados. O ER torna-se útil para mapear, sobre um esquema conceitual, o significado e interações das empresas reais. (SILBERSCHATZ, 2004, p. 21).

Conceitos básicos do ER

Existem três noções básicas empregadas pelo modelo ER: conjunto de entidades, conjunto de relacionamentos e os atributos. Iremos ver cada um deles.

Conjunto de entidades

Entidade: pode ser definida como um objeto do mundo real, concreto ou abstrato e que possui existência independente. Uma entidade possui um conjunto particular de propriedades que a descritivas chamadas de atributos. A designação de um atributo para um conjunto de

entidades expressa que o banco de dados mantém informações similares de cada uma das entidades expressa que o banco de dados mantém informações similares de cada uma das entidades do conjunto de entidades; entretanto cada entidade pode ter seu próprio valor em cada atributo. (SILBERSCHATZ, 2004, p. 21 - 22).

Conjuntos de entidades Fracas: um conjunto de entidades pode não ter atributos suficiente para formar uma chave primária. Esse tipo de conjunto de entidades é denominado de entidades fracas. Um conjunto de entidade que tem chave primaria de entidades fortes. (SILBERSCHATZ, 2004, p. 38).

Atributo: Os valores dos atributos que descrevem uma entidade consistem numa porção significativa dos dados que será armazenado no banco de dados, um atributo, como é usado no modelo ER, pode ser caracterizado por:

Atributo simples ou composto: Um atributo é simples quando não são divididos em partes podemos dar como exemplo sexo de uma pessoa. Por outro lado os atributos compostos são divididos em várias partes. Por exemplo, um endereço que seria formado por rua, número, complemento, cidade, cep, estado e país.

Atributos monovalorados ou multivalorados: Os atributos monovalorados são aqueles que se referem a apenas a um específico. Exemplo o atributo numero_do_empréstimo que se refere a somente um empréstimo. Por outro lado os multivalorados são aqueles que qualquer um pode ter mais de um. Imagine um cadastro, por exemplo, com o atributo numero_de_dependentes.

Atributos nulos: São usados quando uma entidade não possui valor para determinado atributo. No exemplo anterior imagine se uma pessoa não tem dependente então o campo numero_de_dependentes é *null* ou nulo. Pode significar que o valor atributo é desconhecido. Um valor pode se caracterizar por omissão. O valor existe, mas não o conhecemos e nem temos a informação deste.

Atributos derivado: O valor desse tipo de atributo pode ser dividido de outros atributos ou entidades a ele relacionado. Por exemplo, uma entidade cliente possui um atributo empréstimo_tomados, representando o número de empréstimos tomados do cliente ao banco.

Podemos derivar o valor desses empréstimos tomados do banco por um cliente. (SILBERSCHATZ, 2004, p. 23 - 24).

Relacionamento: é um conjunto de associações entre entidades, onde a associação inclui exatamente uma entidade de cada tipo participante no relacionamento. O grau de um tipo relacionamento é o número de tipos entidade que participam do tipo relacionamento. O grau de um relacionamento é ilimitado, porém, a partir do grau 3 (ternário), a compreensão e a dificuldade de se desenvolver a relação corretamente se tornam extremamente complexas. O mapeamento das cardinalidades, ou tateio de cardinalidades, expressa o número de entidades às quais uma outra entidade pode estar associada via um conjunto de relacionamentos. O mapeamento é útil na descrição dos conjuntos de relacionamentos, pois podem descrever os conjuntos de relacionamentos que envolvam mais de dois conjuntos de entidades. (SILBERSCHATZ, 2004, p. 24 - 29).

Pode haver para um conjunto de relacionamento R binário um mapeamento de cardinalidade pode ser:

Um para um: Uma entidade em A está associada no máximo a uma entidade em B e uma entidade em B está associada a no máximo uma entidade em A. (conforme a figura 21, entidade A).

Um para muitos: Uma entidade em A está associada a várias entidades em B. Uma entidade B, entretanto, deve estar associada no máximo a uma entidade em A. (conforme a figura entidade B).

Muitos para um: Uma entidade em A está associada à no máximo uma entidade em B. Uma entidade em B, entretanto, pode estar associada a um número qualquer de entidades em A (conforme a figura 21, entidade A).

Muitos para muitos: Uma entidade em A está associada a qualquer número de entidades em B e uma entidade em B está associada a um número qualquer de entidade em A (conforma a figura 21, entidade A). (SILBERSCHATZ, 2004, p. 29).

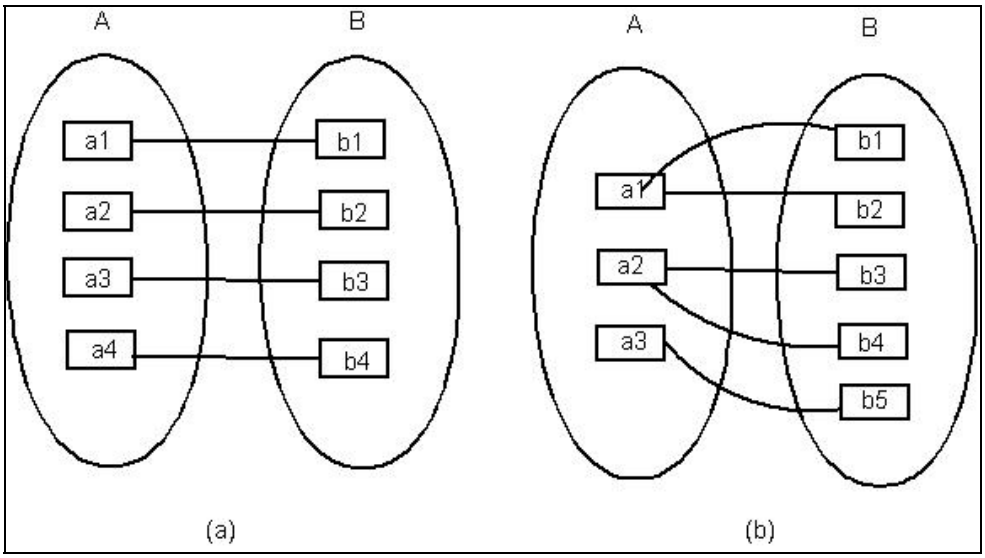


Figura 21 – Mapeamento das cardinalidades.
Fonte: (SILBERSCHATZ, 2004, p. 29).

Chaves

É importante especificar como as entidades dentro de um Banco de dados relacional que tem dentro de si um conjunto de relacionamentos podem ser identificadas. Conceitualmente, entidades e relacionamento individuais são distintos, entretanto, na perspectiva do banco de dados a diferença entre ambos deve ser estabelecida em termos de seus atributos. Onde o conceito de chave permite-nos fazer tais distinções. (SILBERSCHATZ, 2004, p. 32).

Diagrama Entidade Relacionamento

O diagrama Entidade Relacionamento é composto por um conjunto de objetos gráficos que visa representar todos os objetos do modelo Entidade Relacionamento tais como entidades, atributos, atributos chaves, relacionamentos, restrições estruturais, etc. Ele fornece uma visão lógica do banco de dados, fornecendo um conceito mais generalizado de como estão estruturados os dados de um sistema.

A seguir são apresentados seus principais componentes:

Retângulos: representam os conjuntos de entidades.

Elipses: representam os atributos.

Losangos: representam os conjuntos de relacionamentos.

Linhas: que ligam os atributos aos conjuntos de entidades e os conjuntos de entidades aos conjuntos de relacionamentos.

Elipses duplas: que representam atributos multivalorados.

Linhas duplas: que indicam participação total de uma entidade em um conjunto de relacionamento. (SILBERSCHATZ, 2004, p. 35).

A figura 22 mostra um esquema básico de um diagrama ER e seus principais componentes gráficos. Descritos a cima.

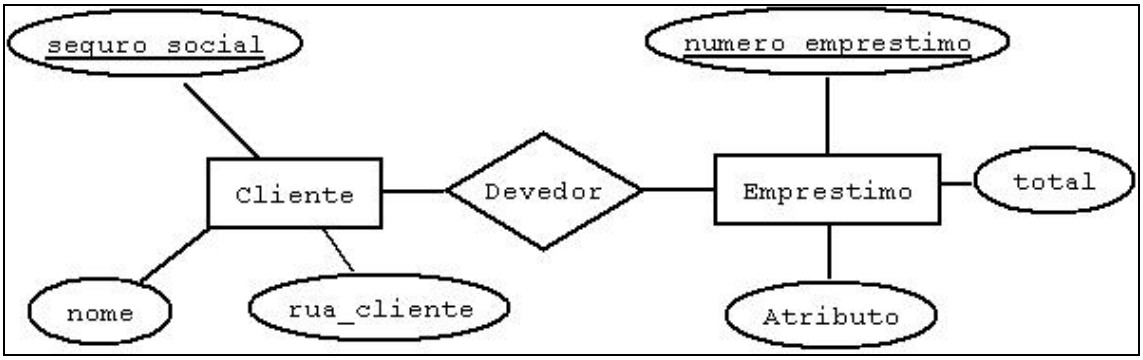


Figura 22 – Diagrama ER
Fonte: Os autores

2.9 LINGUAGEM DE MODELAGEM UNIFICADA - UML

A *Unified Modeling Linguage* (UML) é a padronização da linguagem de modelagem de desenvolvimento orientado a objetos para visualização, especificação, construção e documentação de sistemas. (BOOCH, 2000).

Um modelo é uma representação externa e explícita de parte da realidade vista pela pessoa que deseja usar aquele modelo para entender, mudar, gerenciar e controlar parte daquela realidade. (BEZERRA, 2002).

Com a o desenvolvimento da modelagem, alcançamos quatro objetivos:

a) Os modelos ajudam a visualizar o sistema como ele é ou como desejamos que seja;
b) Os modelos permitem especificar a estrutura ou o comportamento de um sistema;
c) Os modelos proporcionam um guia para a construção do sistema;
d) Os modelos documentam as decisões tomadas.

(BOOCH, 2000).

2.9.1 Diagrama na UML

Um diagrama é a apresentação gráfica de um conjunto de elementos. Permitindo a visualização de um sistema sob várias perspectivas. Na UML identificamos incluso nove desses diagramas divididos em dois grupos, diagramas estáticos e dinâmicos. (BOOCH, 2000).

Diagramas Estáticos:

* Diagrama de Classes

  Demonstra um conjunto de classes, interfaces e colaborações.

* Diagrama de Objetos

  Demonstra um conjunto de objetos e seus relacionamentos.

* Diagrama de Componentes

  Demonstra um conjunto de componentes e seus relacionamentos.

* Diagrama de Implementação

  Demonstra um conjunto de nós e seus relacionamentos.

Diagramas Dinâmicos:

* Diagrama de Caso de Uso.

  Organiza os comportamentos do sistema.

* Diagrama de Seqüência.

  Tem como foco, a ordem temporal das mensagens.

* Diagrama de Colaboração.

  Tem como foco, a organização estrutural de objetos que enviam e recebem mensagens.

* Diagrama de Gráfico de Estados.

  Tem como foco, o estudo de mudança de um sistema orientado por eventos.

* Diagrama de Atividades.

  Tem como foco, o fluxo de controle de uma atividade para outra.

(BOOCH, 2000).

2.10 MÉTODOLOGIAS DE DESENVOLVIMENTO

Uma linguagem de modelagem não é suficiente para dizer como um sistema de software deve ser feito.

Somente uma metodologia que defina quais das diretrizes e quando devem ser construídos.

Entre várias metodologia disponível no mercado, exemplos como *Rational Unifield Process* RUP, *Capability Maturity Model Integration* CMMI, Scrum, Crystal, *eXtreme Programming* (XP), decidimos nos basear na metodologia ICONIX.

2.10.1 ICONIX

ICONIX considerado uma metodologia pura, prática e simples, mas também poderosa e com um componente de análise e representação dos problemas sólido e eficaz. A metodologia ICONIX é caracterizada como um Processo de Desenvolvimento de Software desenvolvido pela ICONIX Software Engineering.O ICONIX é composto pelas seguintes principais fazes:

Diagrama de casos de uso.

Diagrama de robustez.

Diagrama de seqüência.

Diagrama de domínio.

Diagrama de classes.

Trata-se de uma metodologia de desenvolvimento de software que ajuda a planejar, projetar e avaliar o software de forma mais simples. É considerada uma metodologia simples e prática, porém poderosa, e com um componente de análise e representação dos problemas sólido e eficaz, caracterizando-a como um processo de desenvolvimento de software (MAIA, 2005). Na figura 26 demonstra o ciclo de desenvolvimento de software com a utilização da metodologia ICONIX.

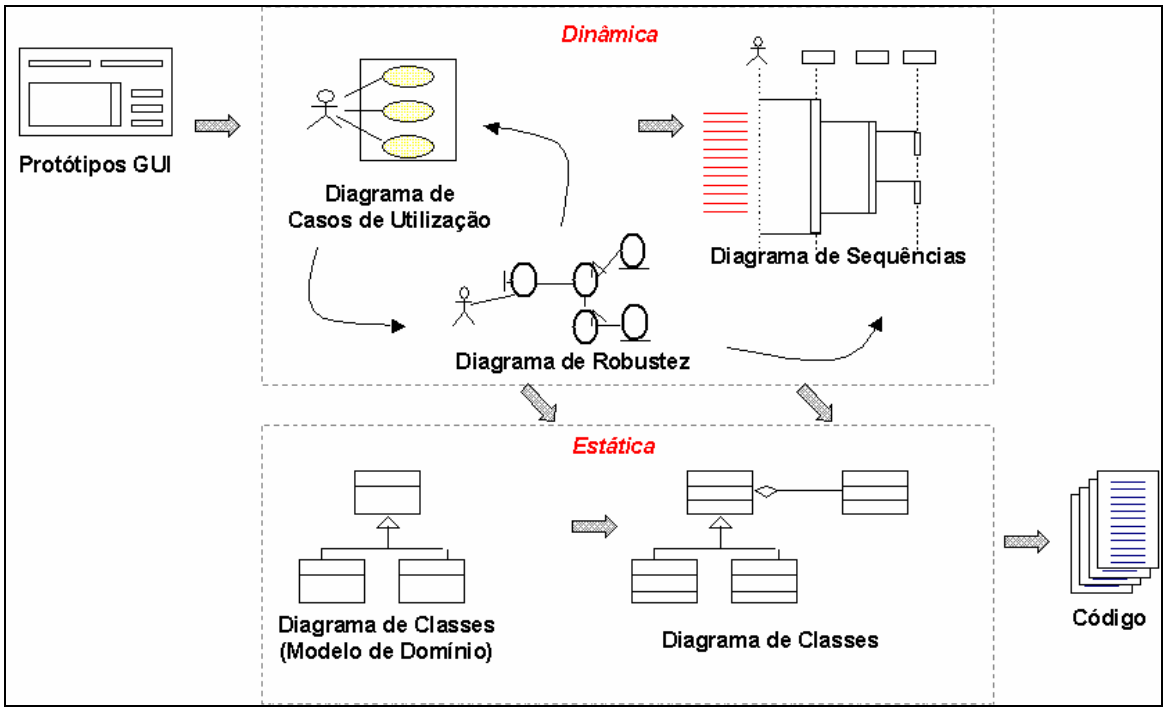


Figura 23 – Ciclo de desenvolvimento de software
Fonte: Doug, Rosenberg; Matt, Stephens. 2007

2.10.2 Modelo de domínio

É o processo pelo qual a informação usada para o desenvolvimento de software é identificada, capturada e organizada para que seja reutilizável quando da criação de novos sistemas.

Requisitos Não Funcionais

Este modelo diz respeito às funções que não são especificadas pelo sistema. Eles podem estar relacionados a propriedades de sistema emergentes, como confiabilidade, tempo de resposta e espaço em disco. Muitos requisitos não funcionais dizem respeito ao sistema como um todo, e não a características individuais dos sistemas. (SOMMERVILLE, 2003)

Requisitos Funcionais

Para um sistema descrevem a funcionalidade ou os serviços que se espera de um sistema forneça. Eles dependem do tipo de software desenvolvido. O levantamento do requisito funcional deve ser completo e consistente. Isto significa que todo o processo realizado pelo usuário deve ser descrito e documentado. (SOMMERVILLE, 2003)

Modelo de Caso de Uso

O modelo de caso de uso corresponde a uma visão externa do sistema e representa graficamente os atores, casos de uso e relacionamentos entre esses elementos. Com essa representação gráfica dos elementos externos interagindo com as funcionalidades do sistema, o sentido a finalidade é representar um tipo de ‘diagrama de contexto’. (BEZERRA, 2002).

Diagrama de Robustez

Os objetos são classificados em três tipos:

Objetos Limite ou interface (*Boundary Objects*) – são usados pelos atores (por exemplo, os usuários) para se comunicarem com o sistema.

Objetos de Controle (*Control Objects*) – são objetos que controlam a lógica de negócio, eles fazem a conexão entre os objetos interface e objetos entidade.

Objetos de Entidade (*Entity Objects*) – são responsáveis para realizar algum tipo de persistência. Geralmente eles vêm do modelo de domínio.

2.11 DIAGRAMA DE CLASSE DAO

Diagrama de Classes demonstra um conjunto de classes, interfaces e colaborações que compõem o sistema e as relações entre elas (por exemplo, a herança). Trata de um aspecto estático e estrutural do sistema.

Será demonstrado o diagrama de classe da DAO. Para as DAO, as classes serão as mesmas para as três modelagens.

Na figura 24 temos a demonstração da utilização da classe da DAO, que será a mesma para os três protótipos.

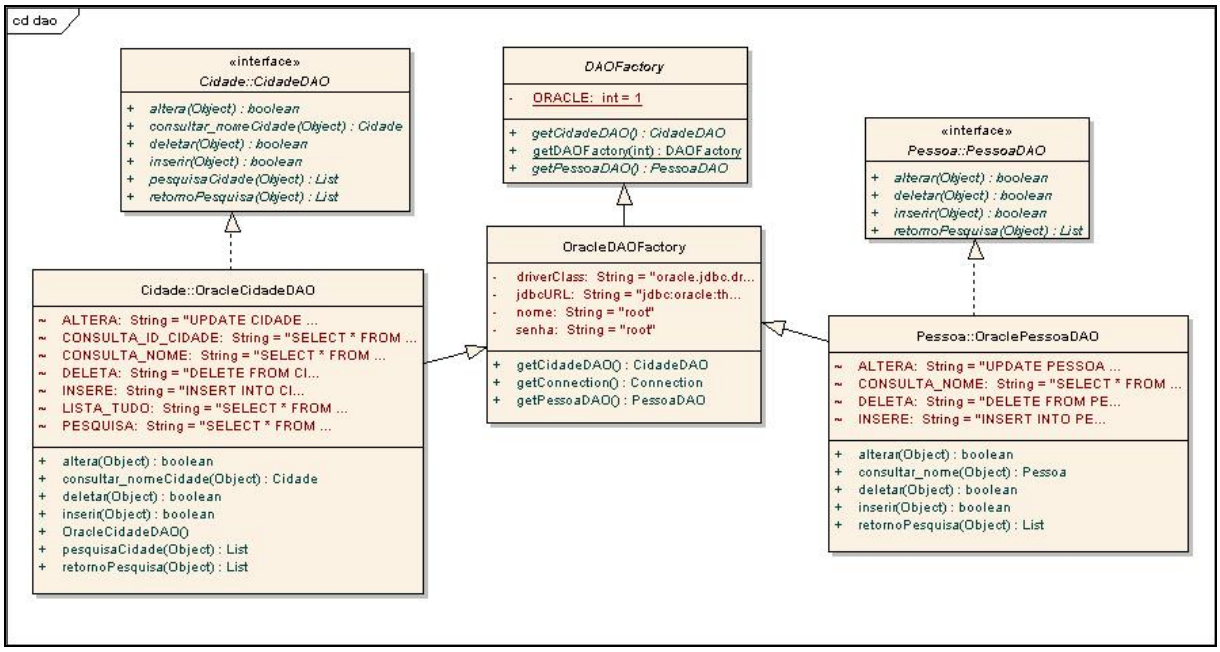


Figura 24 – Diagrama de classes da DAO
Fonte: Os autores.

3 MODELAGENS DOS PROTÓTIPOS

Nesse capítulo será detalhado o processo de desenvolvimento dos protótipos. Os requisitos funcionais, não funcionais, os diagramas de robustez, diagramas de casos de uso, diagramas de atividades e códigos fontes.

Baseamos-nos nas melhores praticas da metodologia ICONIX para gerar a documentação das modelagens.

Prototipamos as interfaces, geramos os casos de usos onde validamos com os diagramas de robustez. Descrevemos então os diagramas seqüência, para fechar a etapa dinâmica do ICONIX. Dando inicio na etapa estática iniciamos com os diagramas de classes de domino e os diagramas de classes.

Começamos pelo protótipo Desktop, onde desenvolvemos a camada View na API Swing da Linguagem JAVA, a camada Control, que foram únicas para esse protótipo. A camada Model e de persistência juntamente com as tabelas do banco de dados foi desenvolvida no primeiro protótipo e foram reutilizadas ou compartilhadas com os outros dois protótipos.

Reaproveitado as tabelas do banco de dados juntamente com as camadas Model e de persistência, desenvolvemos o protótipo Web utilizando JSP na camada View e Servlet na camada Control.

Para o terceiro protótipo, Web com AJAX, foi reutilizado todas as camadas do segundo, onde aproveitamos todas as camadas fazendo algumas pequenas modificações que não tiveram muito impactos no desenvolvimento, adicionando a esse protótipo o Framework DWR para que o protótipo AJAX pudesse realizar as mesmas funcionalidades assíncronas como o protótipo Desktop.

3.1 MODELO ENTIDADE DE RELACIONAMENTO DOS PROTÓTIPOS

Foram utilizas duas tabelas Pessoa e Cidade, com relacionamento de *n* pra *um*.
A figura 25 demonstra a estrutura das tabelas Pessoas e Cidade.

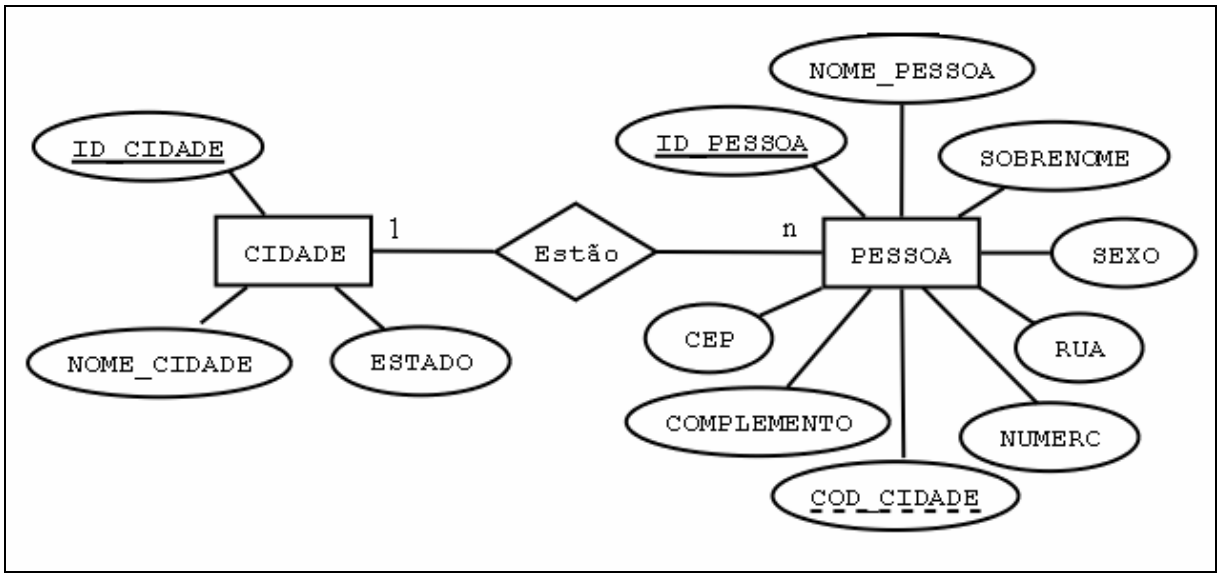


Figura 25 - Modelo de ER Utilizado
Fonte: Os autores.

Na figura 26 demonstramos as tabelas utilizadas nos protótipos no formato de classes para maior visualização e entendimento.

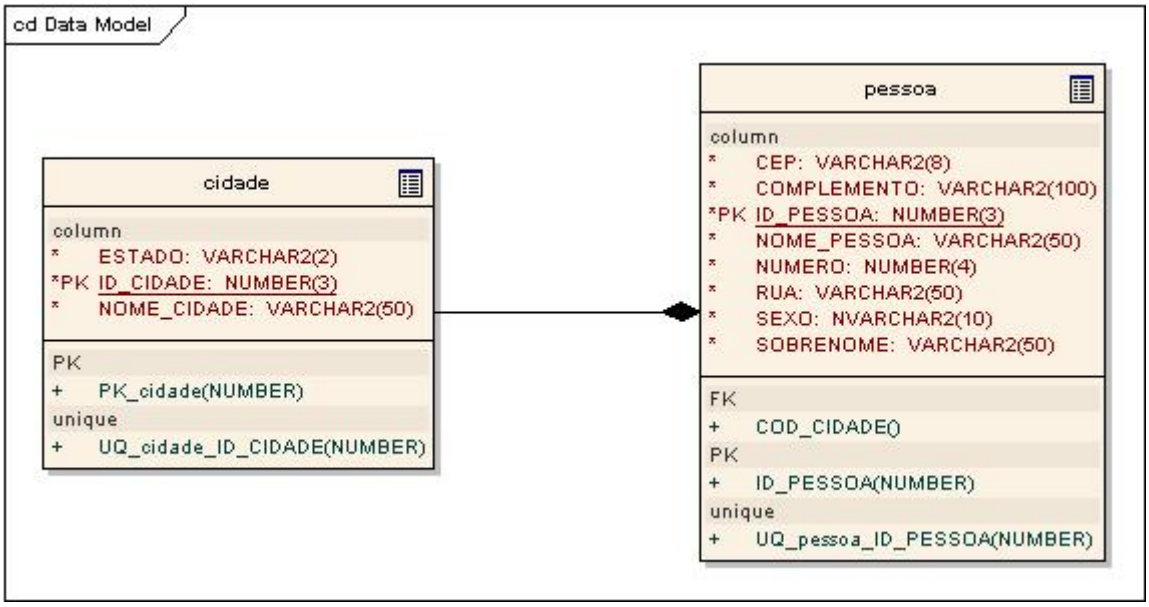


Figura 26 – Diagrama de classes das tabelas
Fonte: Os autores.

Os códigos SQL descritos a baixo foram produzidos na página de aplicação do Oracle 10g XE.
SQL da tabela CIDADE:

```
CREATE TABLE  "CIDADE"
   (    "ID_CIDADE" NUMBER,
        "NOME_CIDADE" VARCHAR2(20) NOT NULL ENABLE,
        "ESTADO" VARCHAR2(2),
         CONSTRAINT "CIDADE_PK" PRIMARY KEY ("ID_CIDADE") ENABLE
   )
```

```
/
```

SQL da tabela PESSOA:

```
CREATE TABLE  "PESSOA"
   (    "ID_PESSOA" NUMBER,
        "NOME_PESSOA" VARCHAR2(20),
        "SOBRENOME" VARCHAR2(20),
        "SEXO" VARCHAR2(4) NOT NULL ENABLE,
        "RUA" VARCHAR2(10),
        "NUMERO" NUMBER,
        "COMPLEMENTO" VARCHAR2(30),
        "COD_CIDADE" NUMBER,
        "CEP" VARCHAR2(9),
         CONSTRAINT "PESSOA_FK" FOREIGN KEY ("COD_CIDADE")
          REFERENCES  "CIDADE" ("ID_CIDADE") ENABLE
   )
/
```

Na figura 27 temos as classes das entidades utilizadas para os três protótipos (Desktop, Web tradicional e Web com AJAX).

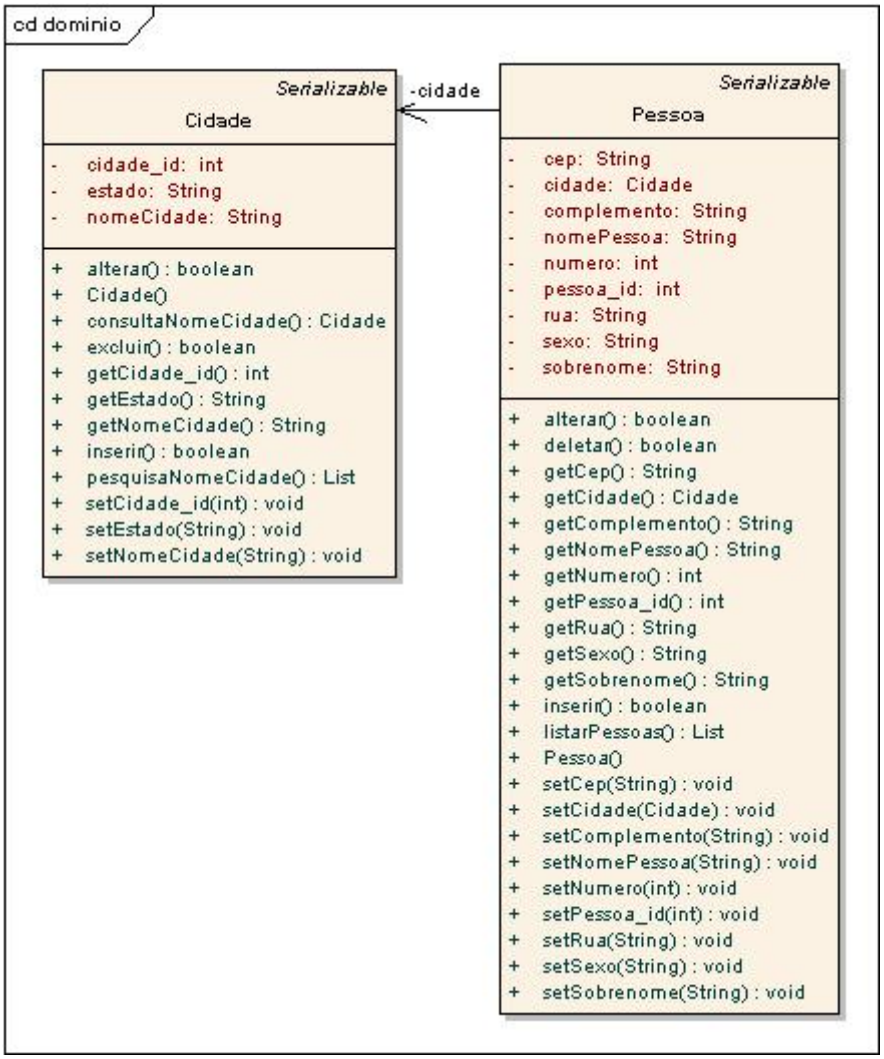


Figura 27 – Diagrama de classes do domínio

Fonte: Os autores.

3.2 DIAGRAMA DE CASO DE USO

Nesse diagrama demonstramos o caso de uso que será utilizado para os três protótipos: Protótipo Desktop, Protótipo Web tradicional e Protótipo Web com AJAX.
A figura 28 demonstra o diagrama de caso de uso para os três protótipos.

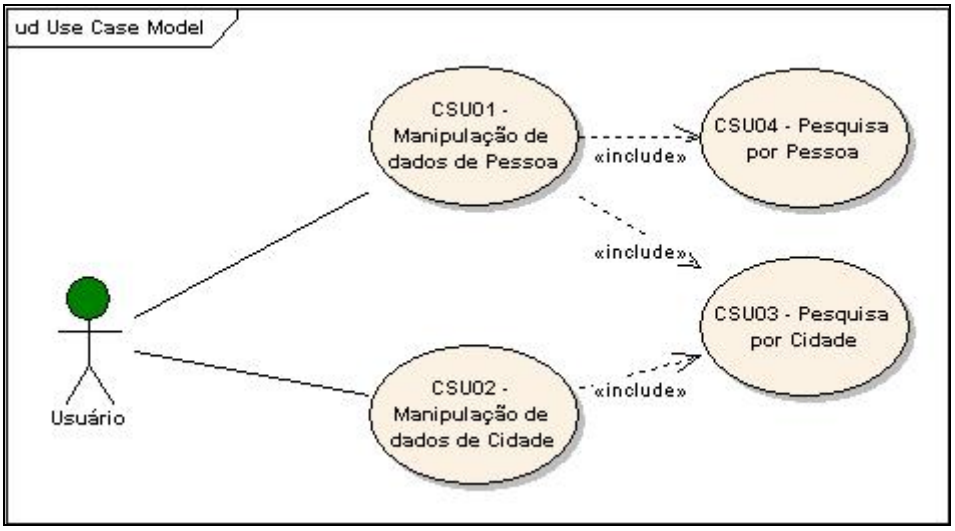


Figura 28 – Diagrama de caso de uso
Fonte: Os autores.

Na figura 29 demonstra a realização do caso do uso apresentado na figura 30.

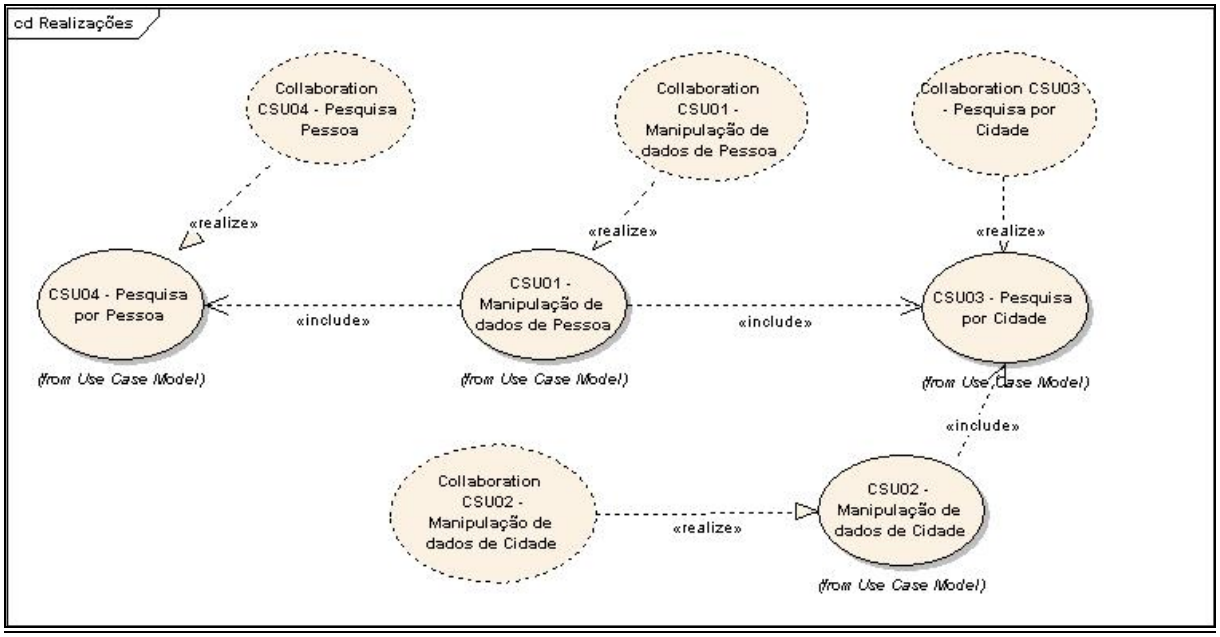


Figura 29 – Diagrama de realização do caso de uso
Fonte: Os autores.

3.3 REQUISITOS NÃO FUNCIONAIS PARA OS TRÊS PROTÓTIPOS

Para os três protótipos esses são os requisitos não funcionais que serão comuns entre eles.

RNF01 - Banco de Dados Utilizado

Descrição: O banco de dados a ser utilizado será o Oracle 10g XE.

RNF02 - Desenvolvimento em três camadas

Descrição: Desenvolvimento com o uso de classes no processo de MVC:
Model;
Control;
View.

RNF03 - Desenvolvimento orientado ao padrão DAO e a API JDBC

Descrição: Desenvolvimento orientado ao padrão DAO e utilizando a API JDBC para conexão como banco de dados.

RNF04 - Desenvolvimento orientado a objetos

Descrição: Desenvolvimento em orientação a objetos (OO) para reaproveitar dos objetos nos protótipos.

RNF05 - Portabilidade do protótipo

Descrição: Os protótipos serão desenvolvidos na plataforma Windows, mas por serem escritos em Java é possível que eles sejam executados em qualquer plataforma que tenha suporte a maquina virtual Java.

RNF06 - Portabilidade de Conexão ao banco

Descrição: A conexão por ser desenvolvida utilizando um *driver* JDBC, permitir a portabilidade dos protótipos em diversas plataformas.

RNF07 – Manutenibilidade

Descrição: O protótipo poderá ser utilizado em qualquer plataforma que ofereça suporte para Java SE 5.0 ou superior.

RNF08 – Tela principal de acesso

Descrição: Está funcionalidade permitir o acesso às funcionalidades dos protótipos.

RNF09 – Resolução de visualização

Descrição: Protótipo será desenvolvido na resolução de tela de 800 por 600 *pixels*.

3.4 REQUISITOS FUNCIONAIS PARA OS TRÊS PROTÓTIPOS

Define a funcionalidade que o software deve prover a fim de capacitar o usuário a realizar suas tarefas, satisfazendo os requisitos.

Esses são os requisitos funcionais que serão comuns para as três modelagens.

RNF01 - Manipulação de dados de pessoa

Descrição: Está funcionalidade deve permitir inclusão, exclusão e alteração de pessoas.

Os atributos manipuláveis do objeto pessoa são os descritos a baixo:

- nome;
- sobrenome;
- sexo;
- rua;
- número;
- complemento;
- cidade;
- CEP;
- estado.

RF02 - Manipulação de dados cidade

Descrição: Está funcionalidade deve permitir inclusão, exclusão e alteração de cidades.

Os atributos manipuláveis do objeto cidade são os descritos a baixo:

- nome cidade;
- estado.

RF03 - Consulta de Cidade

Descrição: Está funcionalidade deve permitir a consulta de cidade.
- nome cidade;

RF04 – Consulta de Pessoa

Descrição: Está funcionalidade deve permitir a consulta de pessoa.
- nome pessoa;

3.5 PROTÓTIPO DESKTOP

O protótipo Desktop foi desenvolvido para demonstrar os processos de cadastro, exclusão, alterações e pesquisas de informações em base de dados.

Figura 30 demonstra a tela principal do protótipo Desktop. A tela principal do protótipo não tem referencia a nenhum caso de uso.

Figura 30 – Interface Desktop – tela inicial
Fonte: Os autores.

3.5.1 Requisitos Não Funcionais

RNF01 - Portabilidade do protótipo Desktop

Descrição: O protótipo foi desenvolvido utilizando a API *Swing* da linguagem Java, portanto terá portabilidade a qualquer sistema operacional que tenha suporte a JVM.

3.5.2 Caso de uso

Veremos detalhadamente as documentações dos casos de uso com referência ao diagrama de caso de uso da figura 31.

3.5.2.1 CSU01 - Manipulação de dados de pessoa

3.5.2.1.1 Manipulação de dados de pessoa - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de Pessoa, realizado o cadastro, exclusão, alteração de pessoa.

Fluxo de Eventos – Cadastro

Fluxo Básico

1. Usuário preenche o campo nome;
2. Usuário preenche o campo sobrenome;
3. Usuário seleciona o campo sexo;
4. Usuário preenche o campo rua;
5. Usuário preenche o campo número;

6. Usuário preenche o campo complemento;
7. Usuário preenche o campo número da cidade;
8. Sistema preenche o nome da cidade e ou nome do Estado;
9. Usuário preenche o campo cep;
10. Usuário clica em ‘Cadastrar’;
11. Sistema cadastra nova pessoa na base de dados.

Fluxo Alternativo

a) < Fluxo Alternativo 1>

1. Usuário não sabe o código da cidade, então clica em Consultar;
2. Sistema ativa a tela de pesquisa;
3. Usuário digita o nome da cidade desejada e clica em Pesquisar;
4. Sistema habilita tela com as cidades relacionadas com o nome digitado na pesquisa;
5. Usuário seleciona a cidade e clica em Cadastrar na tela de cadastro de pessoa.

b) < Fluxo Alternativo 2>

1. Usuário clica na opção Limpar;
2. Sistema limpa as informações da tela.

Fluxo de Exceção

a) < Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade;

Interface Gráfica:

Na figura 31 demonstra a interface do caso de uso de cadastro de pessoa.

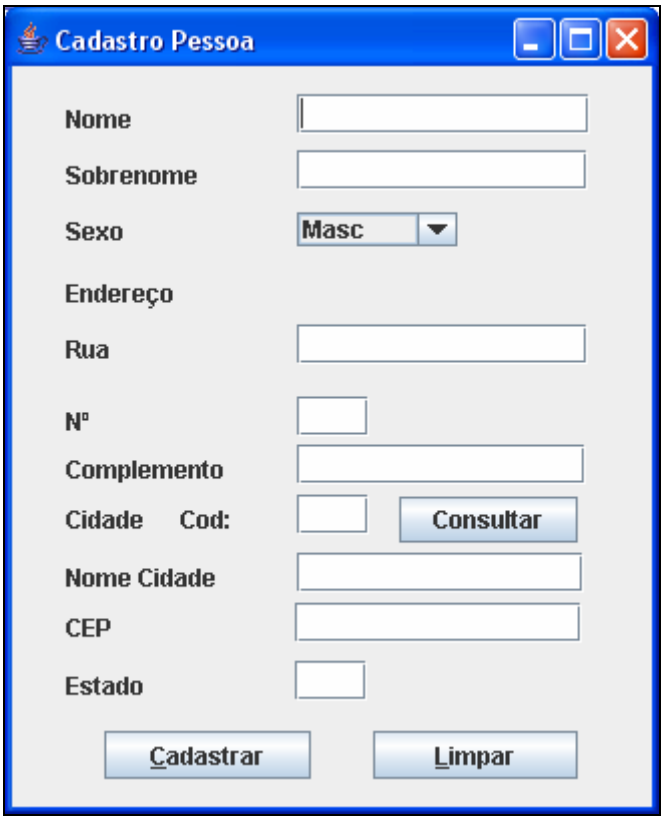


Figura 31 – Interface Desktop – tela de cadastro de pessoa
Fonte: Os autores

3.5.2.1.2 Manipulação de dados de pessoa - Excluir

Nesse caso de uso o usuário realiza a exclusão de Pessoa.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código de pessoa;
2. Usuário clica em Excluir;
3. Sistema exclui a pessoa desejada do banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisar;
2. Sistema habilita tela de pesquisa;
3. Usuário digita o nome da pessoa desejada;

4. Sistema habilita tela com as pessoas relacionadas com que foi digitado no campo de pesquisa.

b) <Fluxo Alternativo 2>

1. Usuário clica em Limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 32 demonstra a interface do caso de uso de exclusão de pessoa.

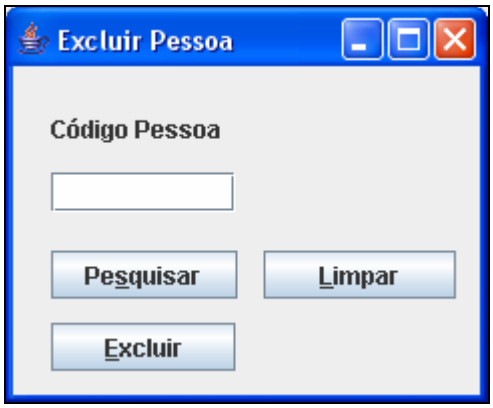


Figura 32 – Interface Desktop – tela de excluir pessoa
Fonte: Os autores.

3.5.2.1.3 Manipulação de dados de pessoa - Alterar

Nesse caso de uso o usuário realiza a alteração de Pessoa.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a alteração dos dados desejados em pessoa;
2. Usuário clica em Alterar;
3. Sistema altera os dados de pessoa no banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

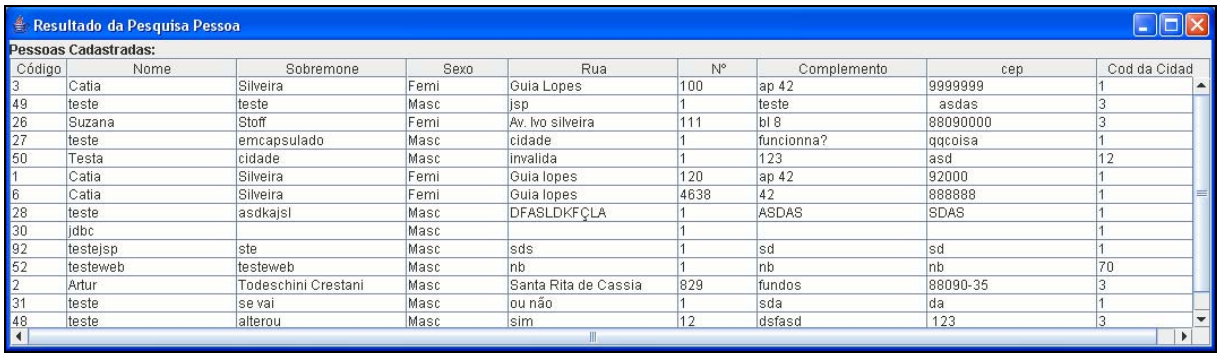
1. Usuário clica em limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

b) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

A figura 33 demonstra a tabela de seleção para o objeto a ser alterado.



Código	Nome	Sobremone	Sexo	Rua	Nº	Complemento	cep	Cod da Cidad
3	Catia	Silveira	Femi	Guia Lopes	100	ap 42	9999999	1
49	teste	teste	Masc	jsp	1	teste	asdas	3
26	Suzana	Stoff	Femi	Av. Ivo silveira	111	bl 8	88090000	3
27	teste	emcapsulado	Masc	cidade	1	funcionna?	qqcoisa	1
50	Testa	cidade	Masc	invalida	1	123	asd	12
1	Catia	Silveira	Femi	Guia lopes	120	ap 42	92000	1
6	Catia	Silveira	Femi	Guia lopes	4638	42	888888	1
28	teste	asdkajsl	Masc	DFASLDKFÇLA	1	ASDAS	SDAS	1
30	jdbc		Masc		1			1
92	testejsp	ste	Masc	sds	1	sd	sd	1
52	testeweb	testeweb	Masc	nb	1	nb	nb	70
2	Artur	Todeschini Crestani	Masc	Santa Rita de Cassia	829	fundos	88090-35	3
31	teste	se vai	Masc	ou não	1	sda	da	1
48	teste	alterou	Masc	sim	12	dsfasd	123	3

Figura 33 – Interface Desktop – tela de retorno da pesquisa de pessoa
Fonte: Os autores

A figura 34 demonstra a interface do caso de uso de alteração de pessoa.

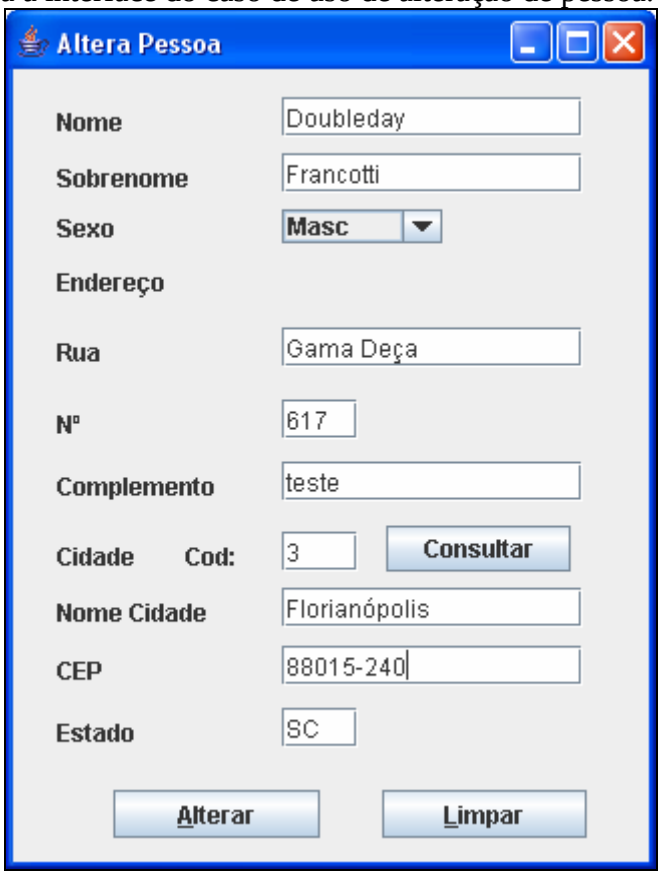


Figura 34 – Interface Desktop – tela de alteração de pessoa
Fonte: Os autores

3.6 DIAGRAMA DE PACOTES

Em muitos casos um único diagrama de classes pode ser exageradamente grande para representar todo o sistema. Assim é conveniente utilizar-se de um elemento para organizar os subsistemas do modelo. Para isto utilizam-se os diagramas de pacote. Um pacote representa um grupo de classes (ou outros elementos) que se relaciona com outros pacotes através de uma relação de dependência.

Na figura 35 temos o diagrama de pacotes do protótipo Desktop.

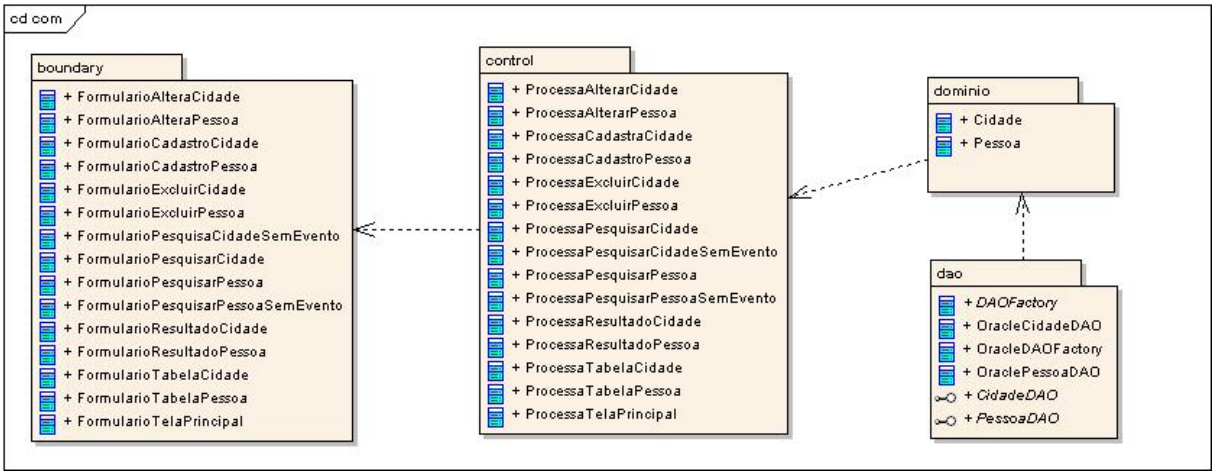


Figura 35 – Diagrama de pacotes do protótipo Desktop
Fonte: Os autores.

3.7 DIAGRAMA DE CLASSES

Diagrama de Classes demonstra um conjunto de classes, interfaces e colaborações que compõem o sistema e as relações entre elas (por exemplo, a herança). Trata de um aspecto estático e estrutural do sistema.

Será demonstrado o diagrama de classe das entidades e dos controles. Para as classes das *boundarys*, não será demonstrada, pois as mesmas não realizam relação entre as interfaces. Todas são executadas pelos controles.

Na figura 36, temos o diagrama de classe do controle utilizado no desenvolvimento do protótipo Desktop.

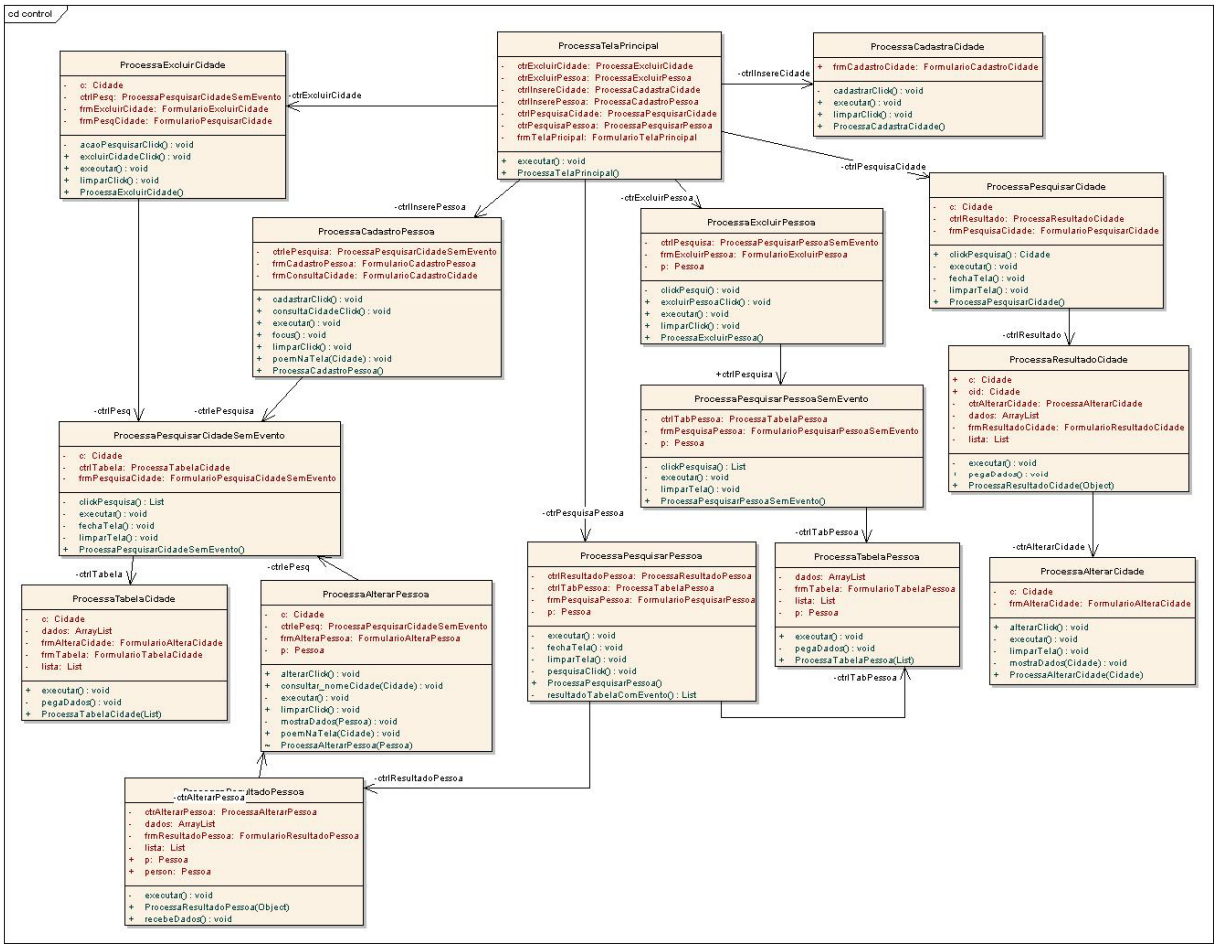


Figura 36 – Diagrama de classes dos controles do protótipo Desktop
Fonte: Os autores.

3.8 DIAGRAMA DE ROBUSTEZ

Nesta etapa serão descritos os diagramas de robustez. São ilustradas graficamente as iterações entre os objetos participantes do caso de uso.

Na figura 37 demonstra a diagrama de robustez do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa - Cadastrar

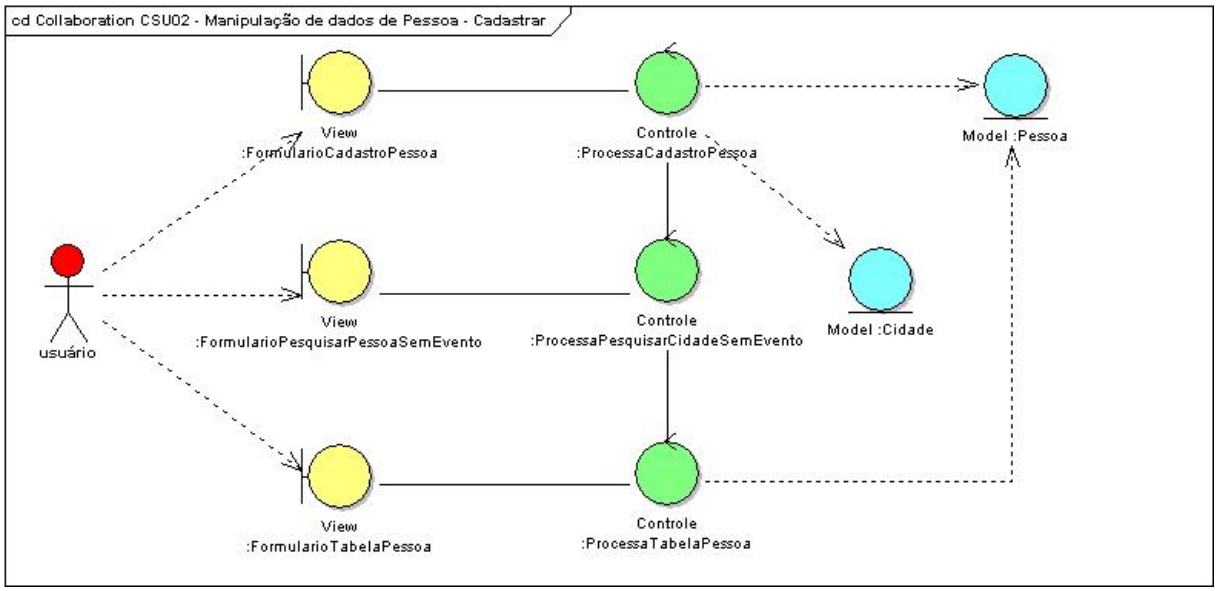


Figura 37 – Diagrama de robustez do CSU01 – Desktop - cadastrar pessoa
Fonte: Os autores.

Na figura 38 demonstra a diagrama de robustez do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

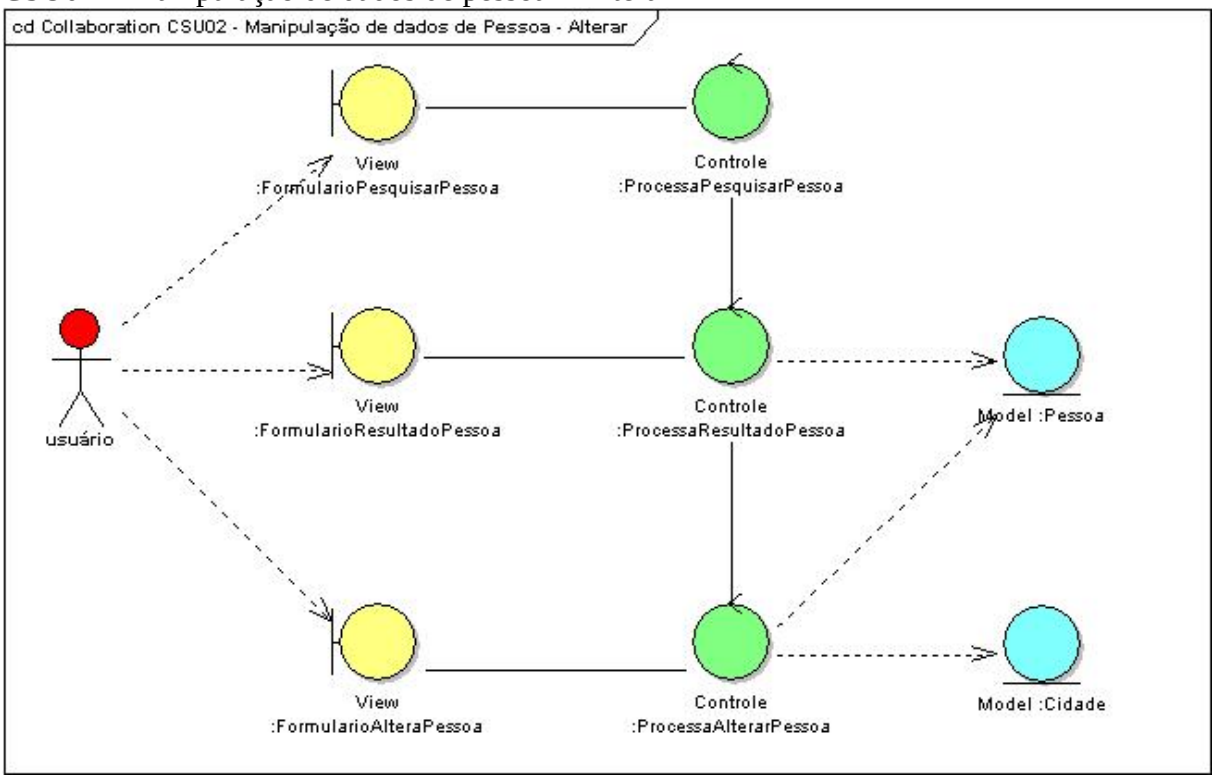


Figura 38 – Diagrama de robustez do CSU01 – Desktop - alterar pessoa
Fonte: Os autores.

Na figura 39 demonstra a diagrama de robustez do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

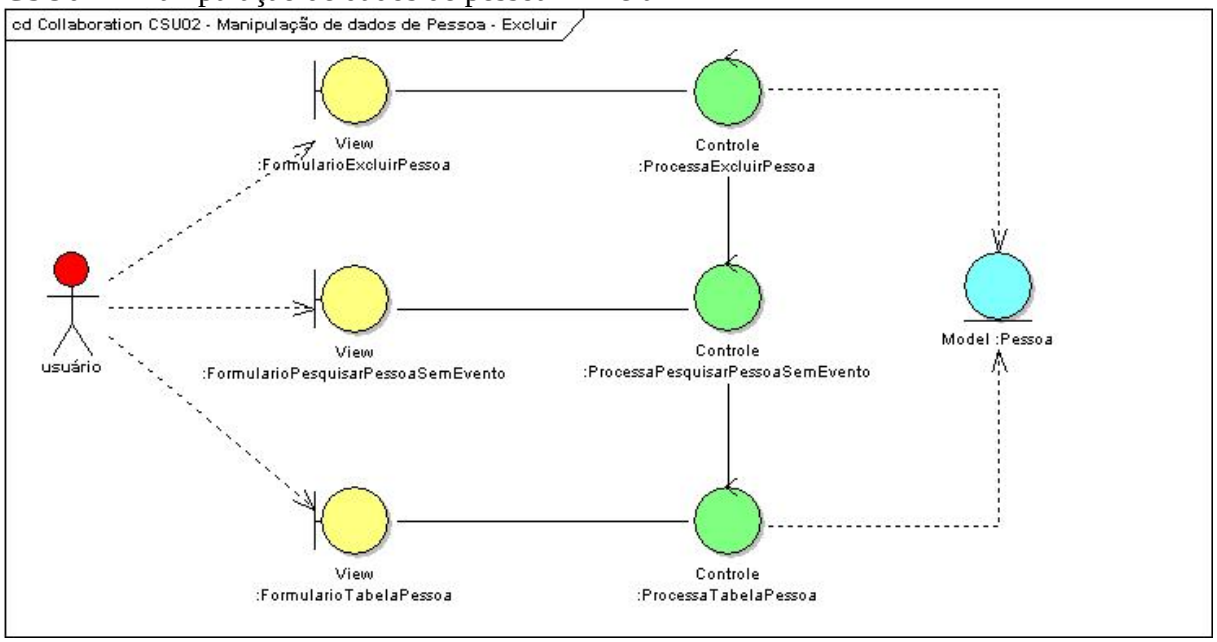


Figura 39 – Diagrama de robustez do CSU01 – Desktop - excluir pessoa
Fonte: Os autores.

3.9 DIAGRAMA DE SEQÜÊNCIA

Nesta etapa são construídos os diagramas de seqüência que tem como finalidade instruir o desenvolvimento.

Na figura 40 demonstra a diagrama de seqüência do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa – Cadastro

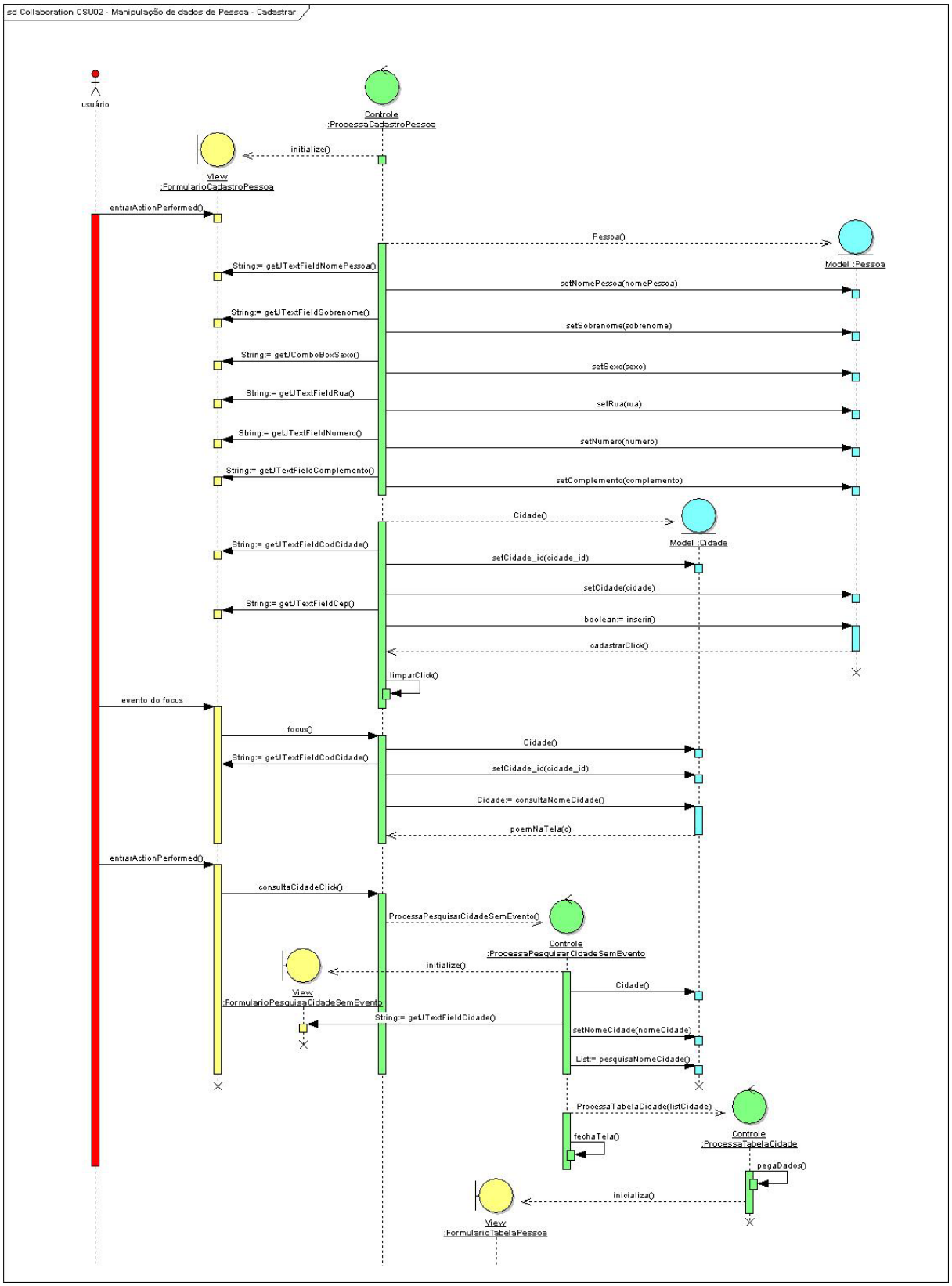


Figura 40 – Diagrama de seqüência do CSU01 – Desktop - cadastrar pessoa
Fonte: Os autores.

Na figura 41 demonstra a diagrama de seqüência do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

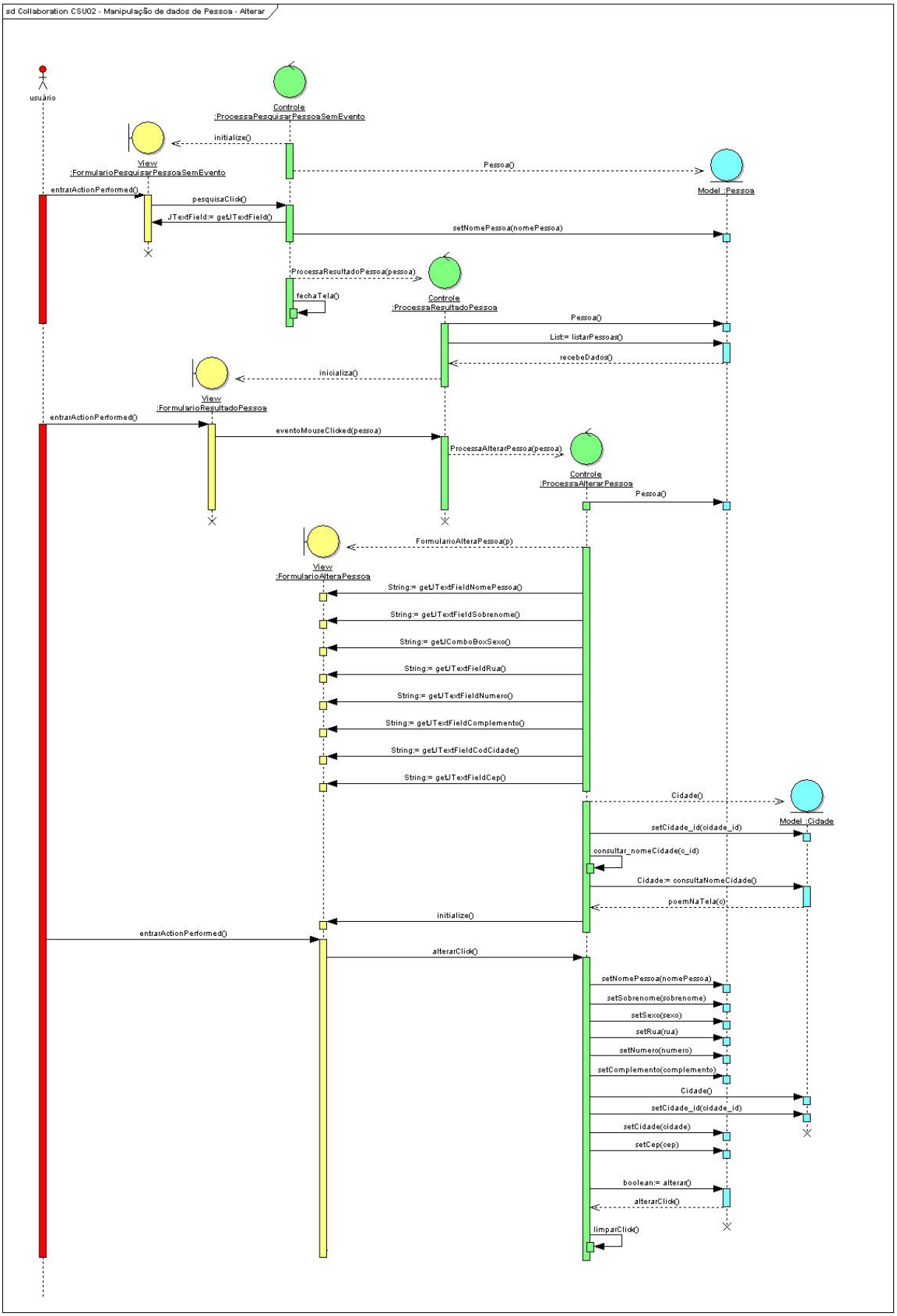


Figura 41 – Diagrama de seqüência do CSU01 – Desktop - alterar pessoa
Fonte: Os autores.

Na figura 42 demonstra a diagrama de seqüência do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

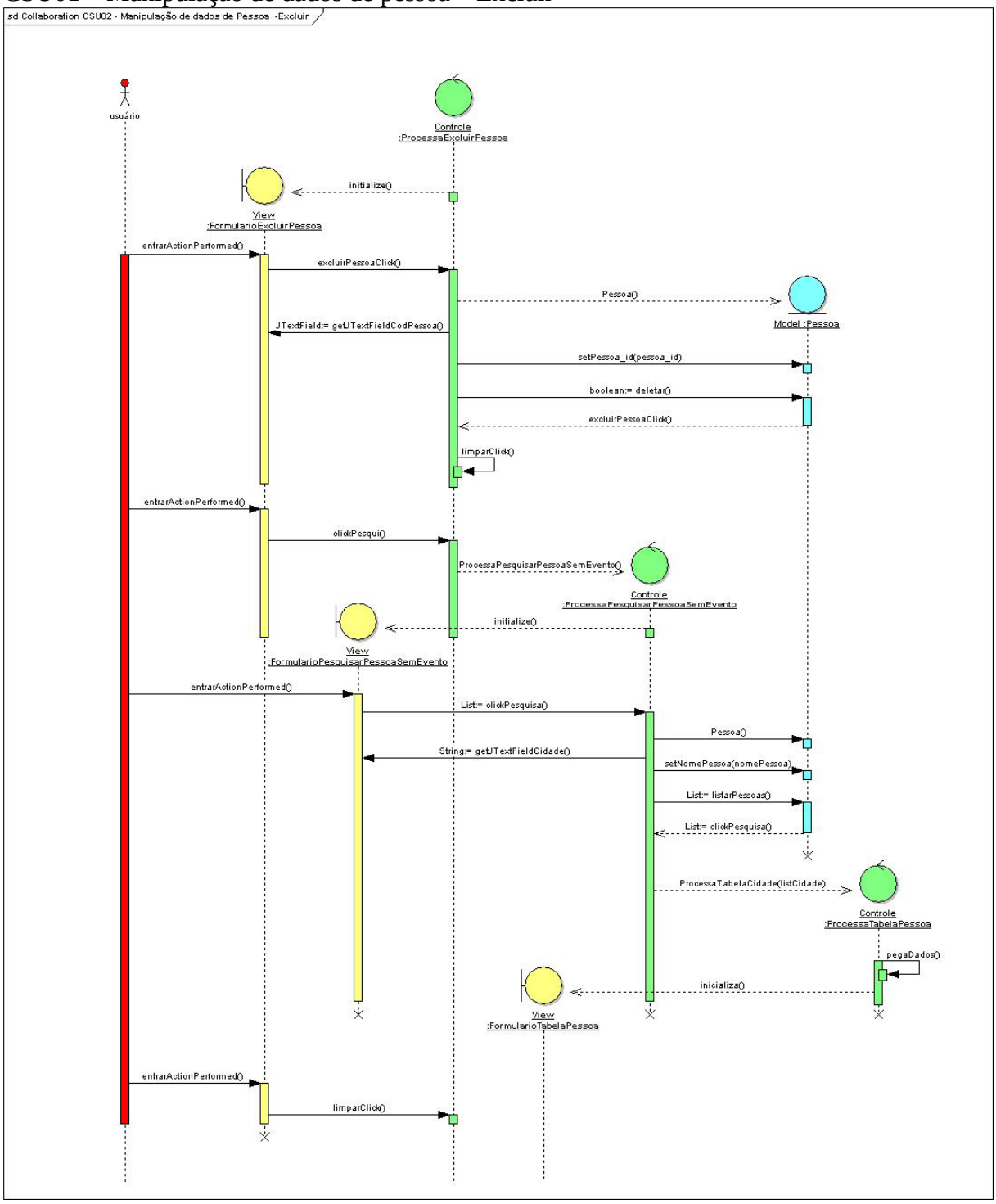


Figura 42 – Diagrama de seqüência do CSU01 – Desktop - excluir pessoa
Fonte: Os autores.

3.10 PROTÓTIPO WEB TRADICIONAL

Na figura 43 temos a demonstração do menu do protótipo Web tradicional.

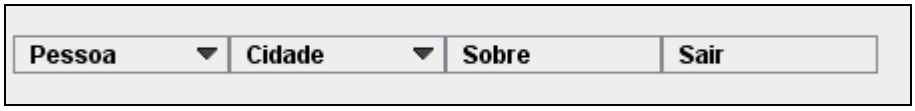


Figura 43 – Menu do protótipo Web tradicional
Fonte: Os autores.

3.10.1 Requisitos Não Funcionais

RNF01 – Uso de falso botão

Descrição: por impossibilidade da plataforma em utilizar botões com link, utilizamos CSS para termos falsos botões como link.

3.10.2 Caso de Uso

Serão detalhados os casos de uso do protótipo Web tradicional.

3.10.2.1 CSU01 - Manipulação de dados de pessoa

3.10.2.1.1 Manipulação de dados de pessoa - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de pessoa, realizado o cadastro, exclusão, alteração de pessoa.

Fluxo de Eventos – Cadastro

Fluxo Básico

1. Usuário preenche o campo nome;
2. Usuário preenche o campo sobrenome;
3. Usuário seleciona o campo sexo;
4. Usuário preenche o campo rua;
5. Usuário preenche o campo número;
6. Usuário preenche o campo complemento;
7. Usuário preenche o campo número da cidade;
8. Usuário preenche o campo cep;
9. Usuário clica em Cadastrar;
10. Sistema cadastra nova pessoa na base de dados.

Fluxo Alternativo

a) < Fluxo Alternativo 1>

1. Usuário não sabe o número da cidade, então clica em Consultar;
2. Sistema ativa a tela de pesquisa;
3. Usuário digita o nome da cidade desejada e clica em Pesquisar;
4. Sistema habilita tela com as cidades relacionadas com o nome digitado na pesquisa;

b) < Fluxo Alternativo 2>

1. Usuário clica na opção Limpar;
2. Sistema limpa as informações da tela.

Fluxo de Exceção

a) < Fluxo Exceção 1>

1. Usuário fecha browser direto, sem executar qualquer tipo de funcionalidade;

Interface Gráfica:

Na figura 44 demonstra a interface do caso de uso de cadastro de pessoa. Conforme processo do fluxo básico.

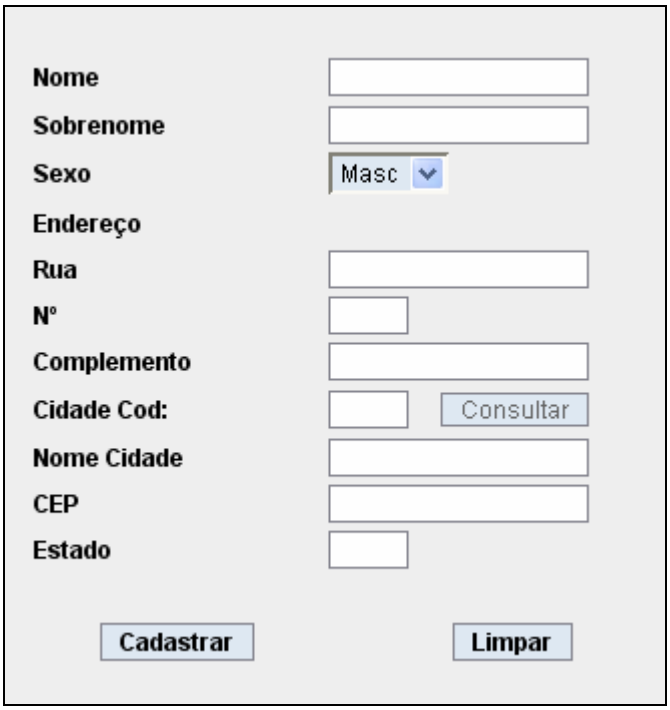


Figura 44 – Interface Web tradicional – tela de cadastro de pessoa
Fonte: Os autores

Na fugira 45 demonstra a pesquisa por cidade, conforme descrito no processo do fluxo alternativo 1.

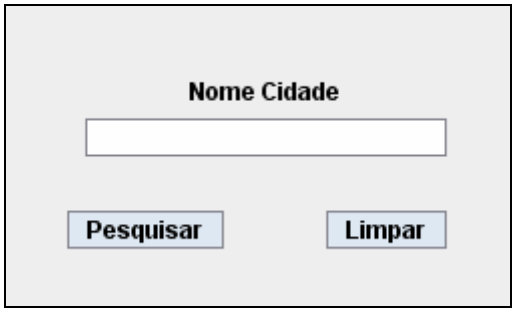
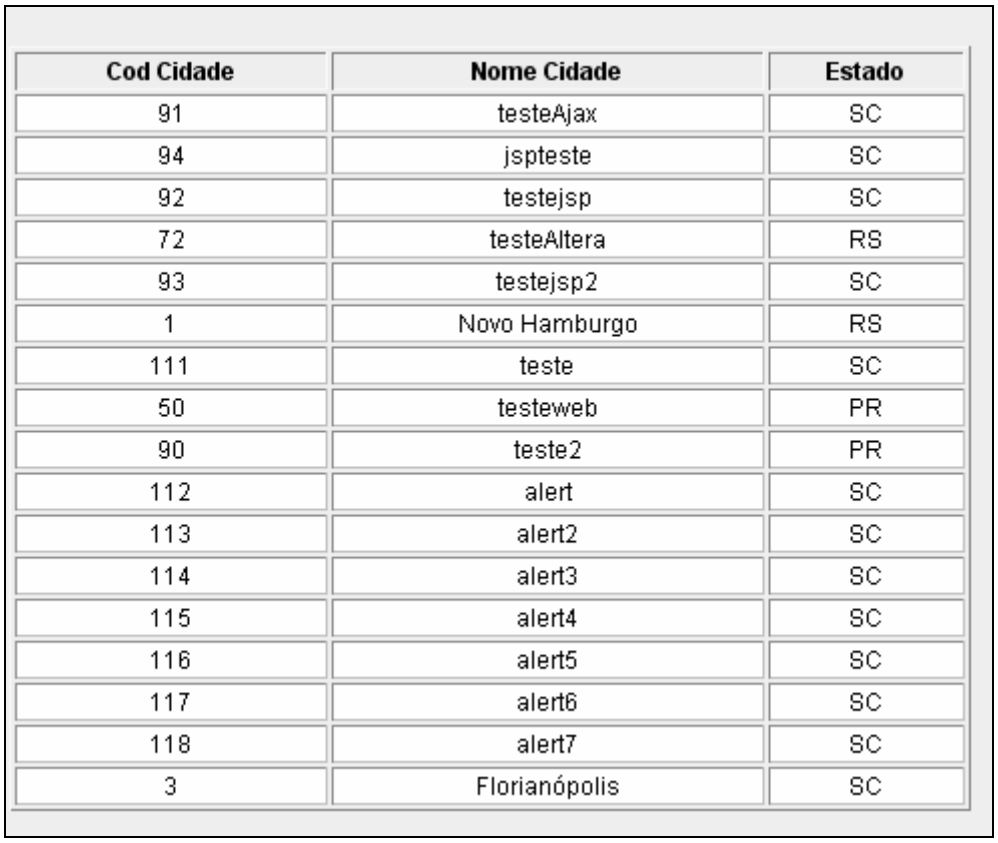


Figura 45 – Interface Web tradicional – tela de pesquisa de cidade
Fonte: Os autores

Caso o usuário não selecione o nome da cidade para a pesquisa, o sistema pegará todos os dados cadastrados na base dados com relação à cidade. Conforme a figura 46.



Cod Cidade	Nome Cidade	Estado
91	testeAjax	SC
94	jspteste	SC
92	testejsp	SC
72	testeAltera	RS
93	testejsp2	SC
1	Novo Hamburgo	RS
111	teste	SC
50	testeweb	PR
90	teste2	PR
112	alert	SC
113	alert2	SC
114	alert3	SC
115	alert4	SC
116	alert5	SC
117	alert6	SC
118	alert7	SC
3	Florianópolis	SC

Figura 46 – Interface Web tradicional – tela do resultado da pesquisa de cidade
Fonte: Os autores

3.10.2.1.2 Manipulação de dados de pessoa - Excluir

Nesse caso de uso o usuário realiza a exclusão de Pessoa.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código de pessoa;
2. Usuário clica em Excluir;
3. Sistema exclui a pessoa desejada do banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisar;
2. Sistema habilita tela de pesquisa;
3. Usuário digita o nome da pessoa desejada;
4. Sistema habilita tela com as pessoas relacionadas.
5. Execute o fluxo básico com o número do registro encontrado.

b) <Fluxo Alternativo 2>

1. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha o browser direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 47 demonstra a interface do caso de uso de exclusão de pessoa.

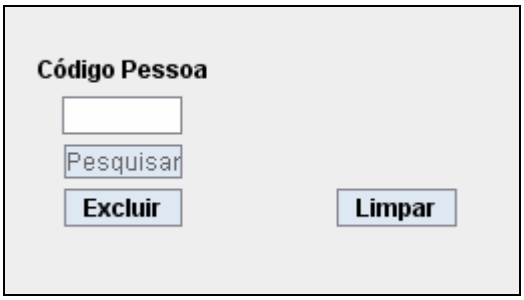


Figura 47 – Interface Web tradicional – tela de excluir pessoa
Fonte: Os autores.

3.10.2.1.3 Manipulação de dados de pessoa - Alterar

Nesse caso de uso o usuário realiza a alteração de Pessoa.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a pesquisa
2. Usuário clica em Editar
3. Usuário realiza a alteração dos dados
4. Usuário clica em alterar
5. Sistema altera os dados de pessoa no banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

A figura 48 demonstra a interface do caso de uso de alteração de pessoa.

Figura 48 – Interface Web tradicional – tela de alterar pessoa
Fonte: Os autores.

3.11 DIAGRAMA DE PACOTES

Em muitos casos um único diagrama de classes pode ser exageradamente grande para representar todo o sistema. Assim é conveniente utilizar-se de um elemento para organizar os subsistemas do modelo. Para isto utilizam-se os diagramas de pacote. Um pacote representa um grupo de classes (ou outros elementos) que se relaciona com outros pacotes através de uma relação de dependência.

Na figura 49 temos o diagrama de pacotes do protótipo Web tradicional.

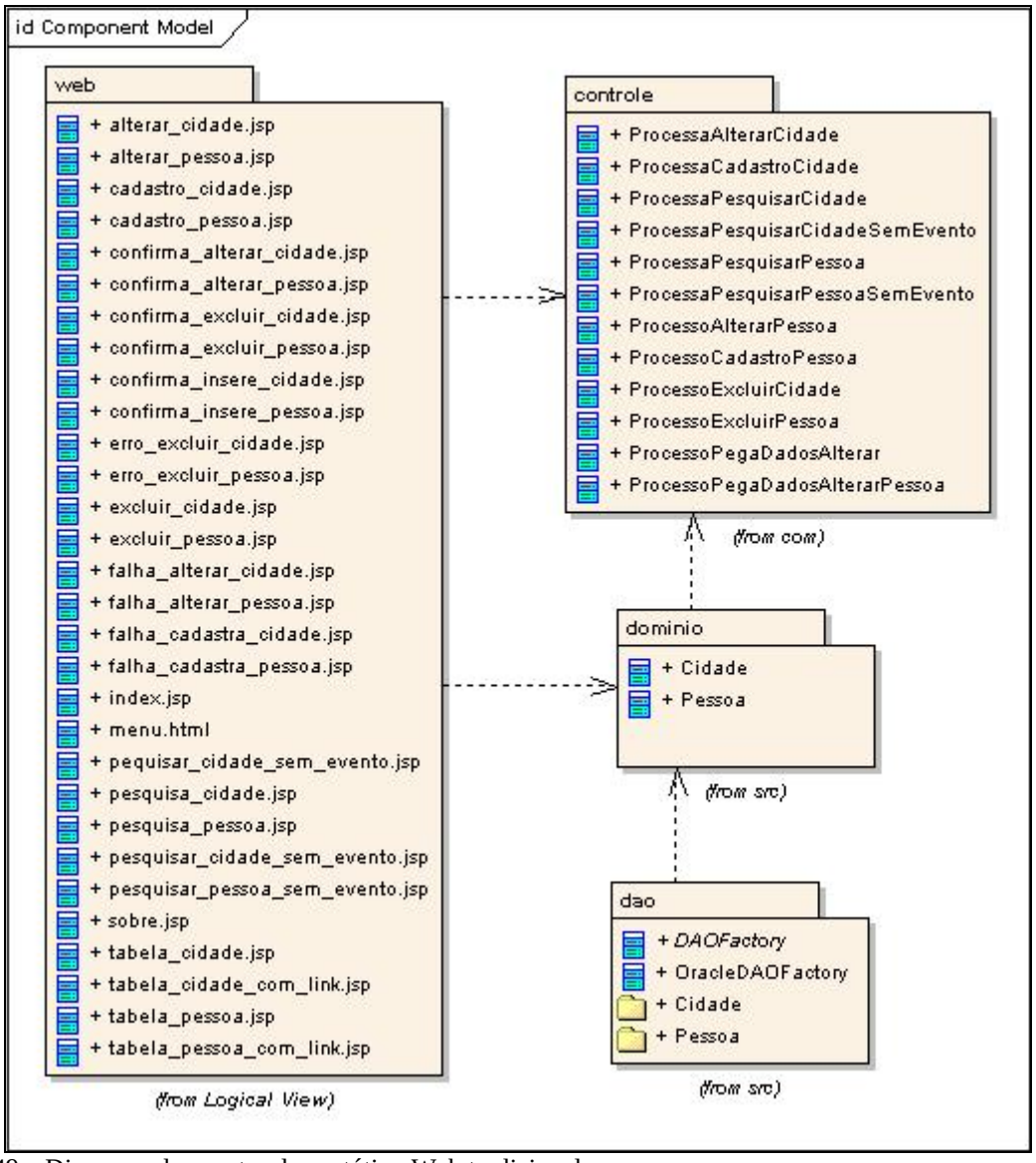


Figura 49 – Diagrama de pacotes do protótipo Web tradicional
Fonte: Os autores.

3.12 DIAGRAMA DE CLASSES

Diagrama de Classes demonstra um conjunto de classes, interfaces e colaborações que compõem o sistema e as relações entre elas (por exemplo, a herança). Trata de um aspecto estático e estrutural do sistema.

Na figura 50 temos demonstrado o diagrama de classes do protótipo Web tradicional. Como verificado no código fonte, as controle não apresentam ligações entre sim como no protótipo Desktop.

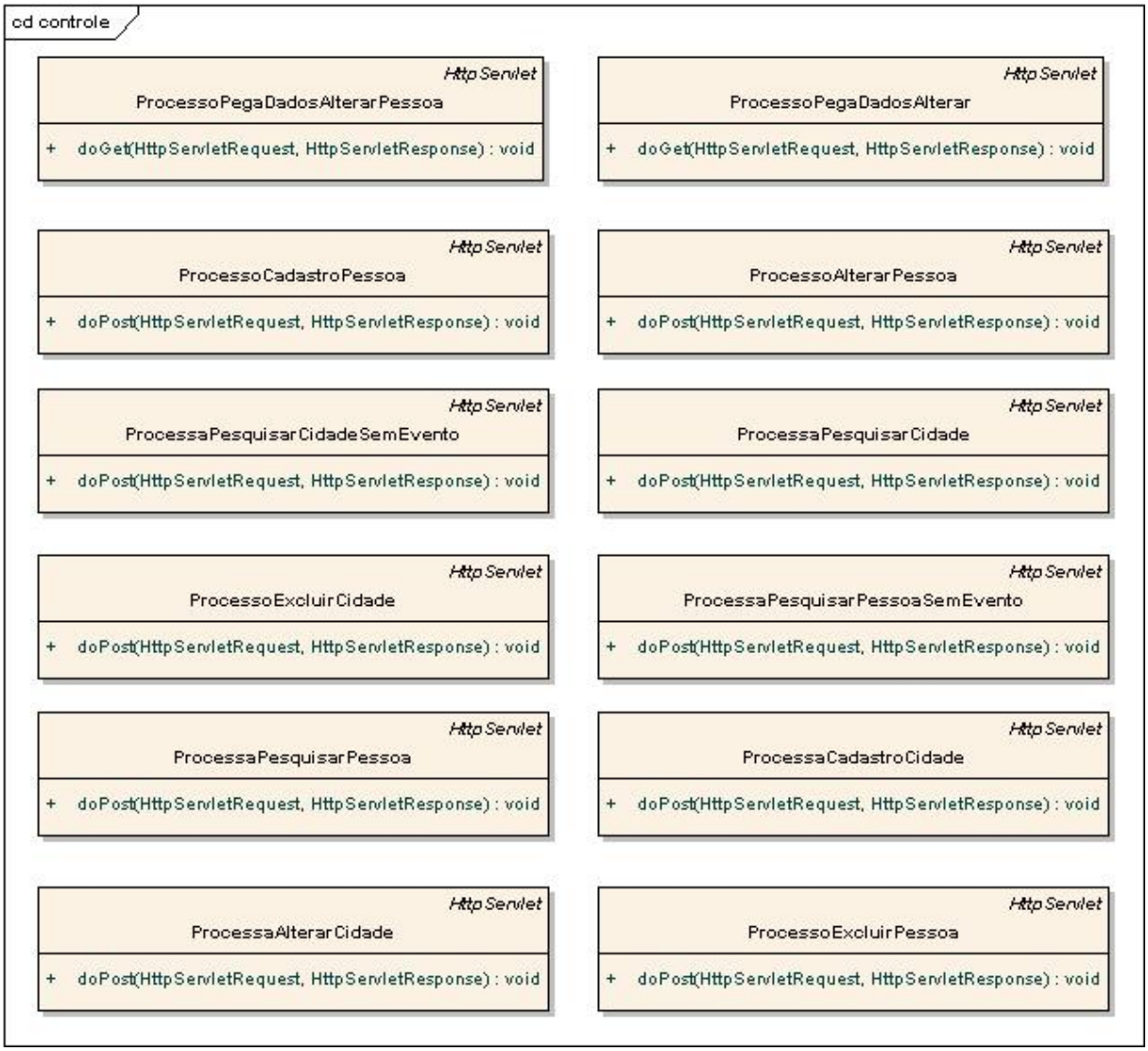


Figura 50 – Diagrama de classe do protótipo Web tradicional
Fonte: Os autores.

3.13 DIAGRAMA DE ROBUSTEZ

Nesta etapa serão descritos os diagramas de robustez. São ilustradas graficamente as iterações entre os objetos participantes do caso de uso.

Na figura 51 demonstra a diagrama de robustez do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa - Cadastrar

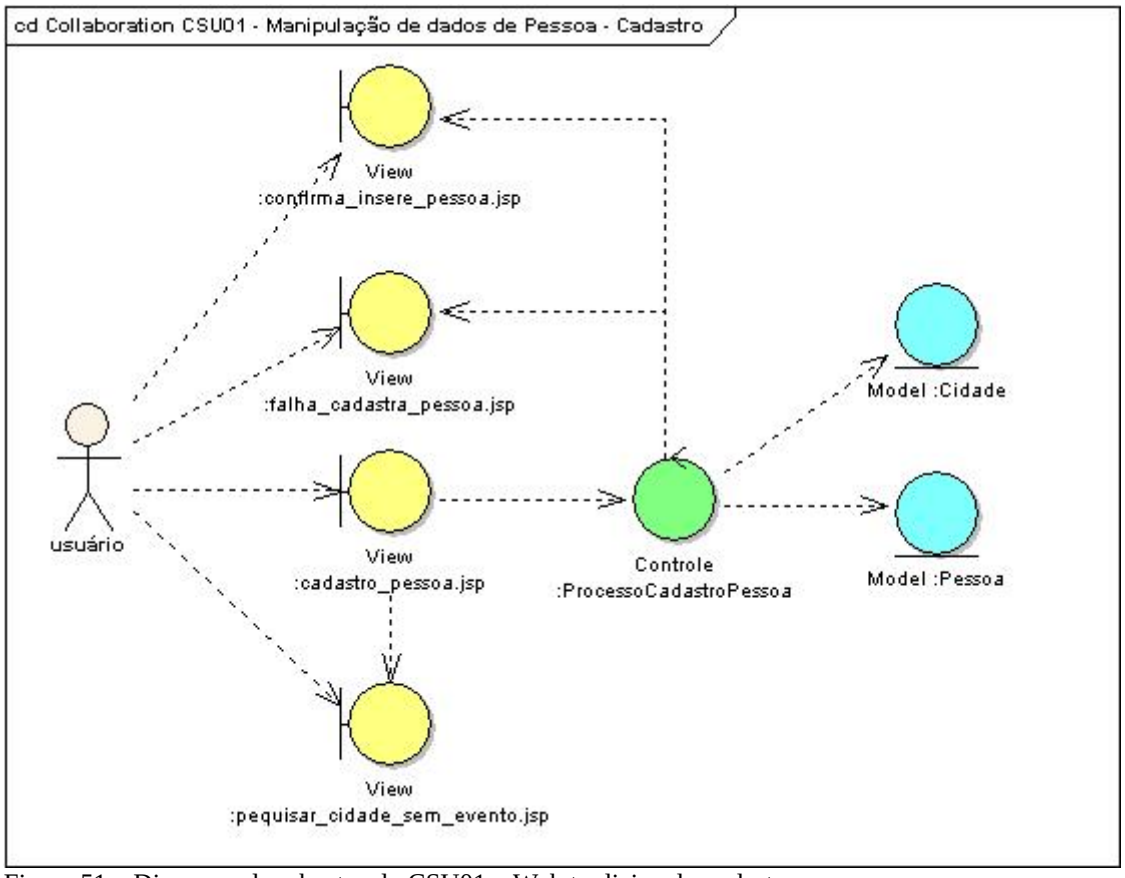


Figura 51 – Diagrama de robustez do CSU01 – Web tradicional - cadastrar pessoa
Fonte: Os autores.

Na figura 52 demonstra a diagrama de robustez do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

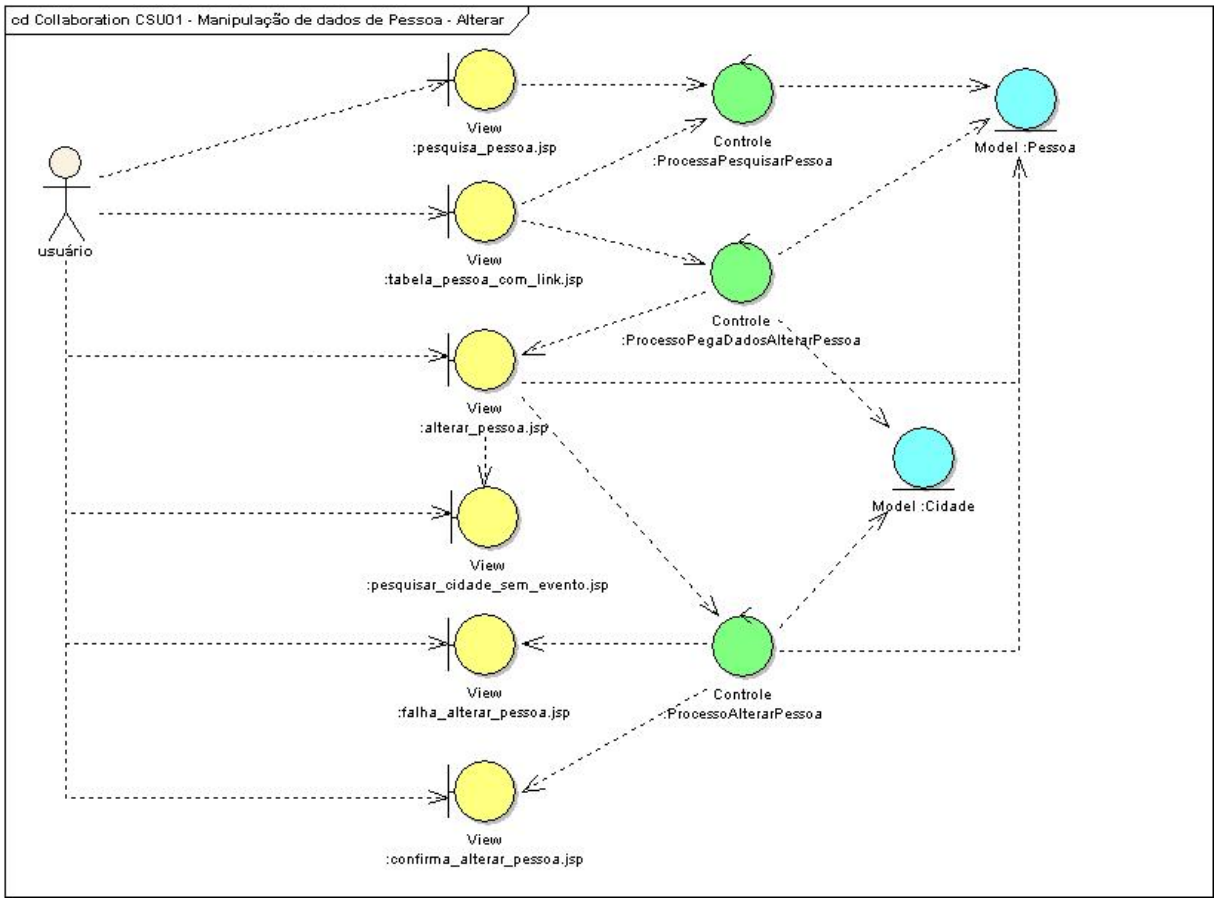


Figura 52 – Diagrama de robustez do CSU01 – Web tradicional - alterar pessoa
Fonte: Os autores.

Na figura 53 demonstra a diagrama de robustez do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

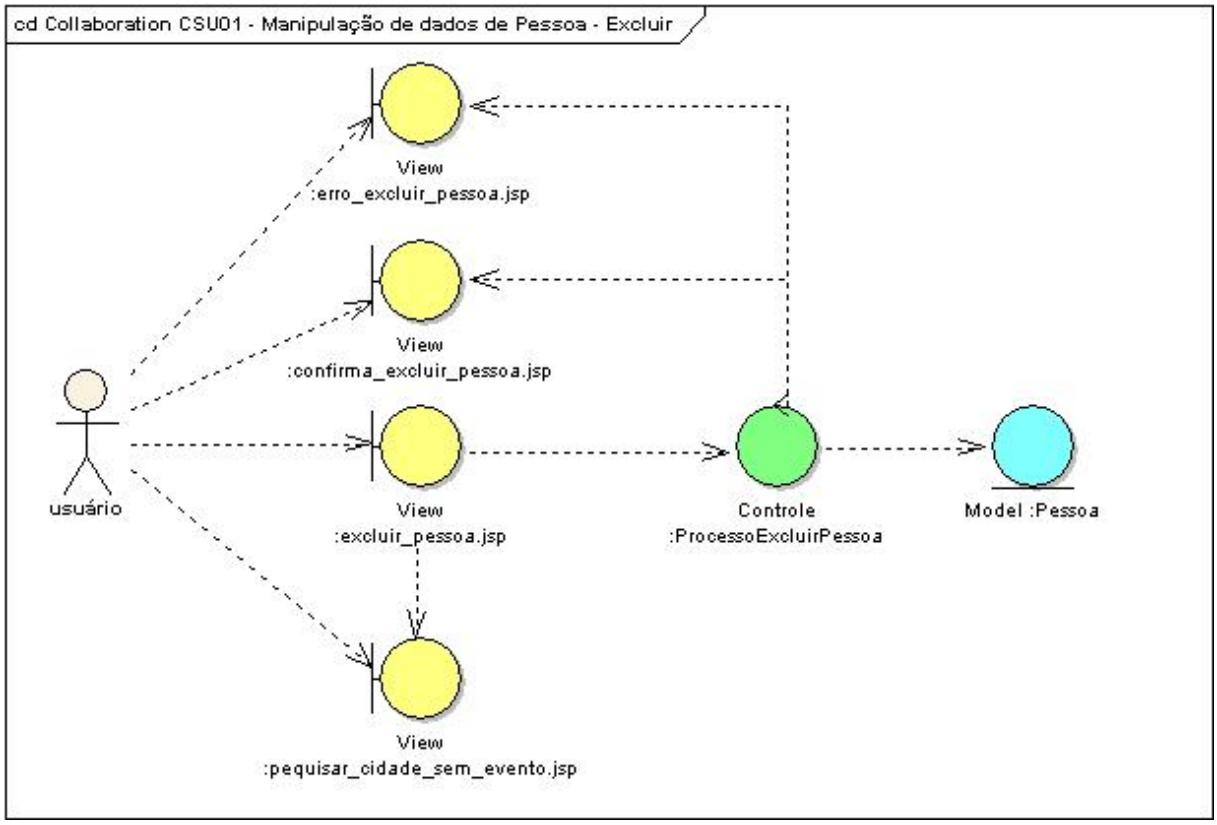


Figura 53 – Diagrama de robustez do CSU01 – Web tradicional - excluir pessoa
Fonte: Os autores.

3.13.1 DIAGRAMA DE SEQUÊNCIA

Nesta etapa são construídos os diagramas de seqüência que tem como finalidade instruir o desenvolvimento.

Na figura 54 demonstra a diagrama de seqüência do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa – Cadastro

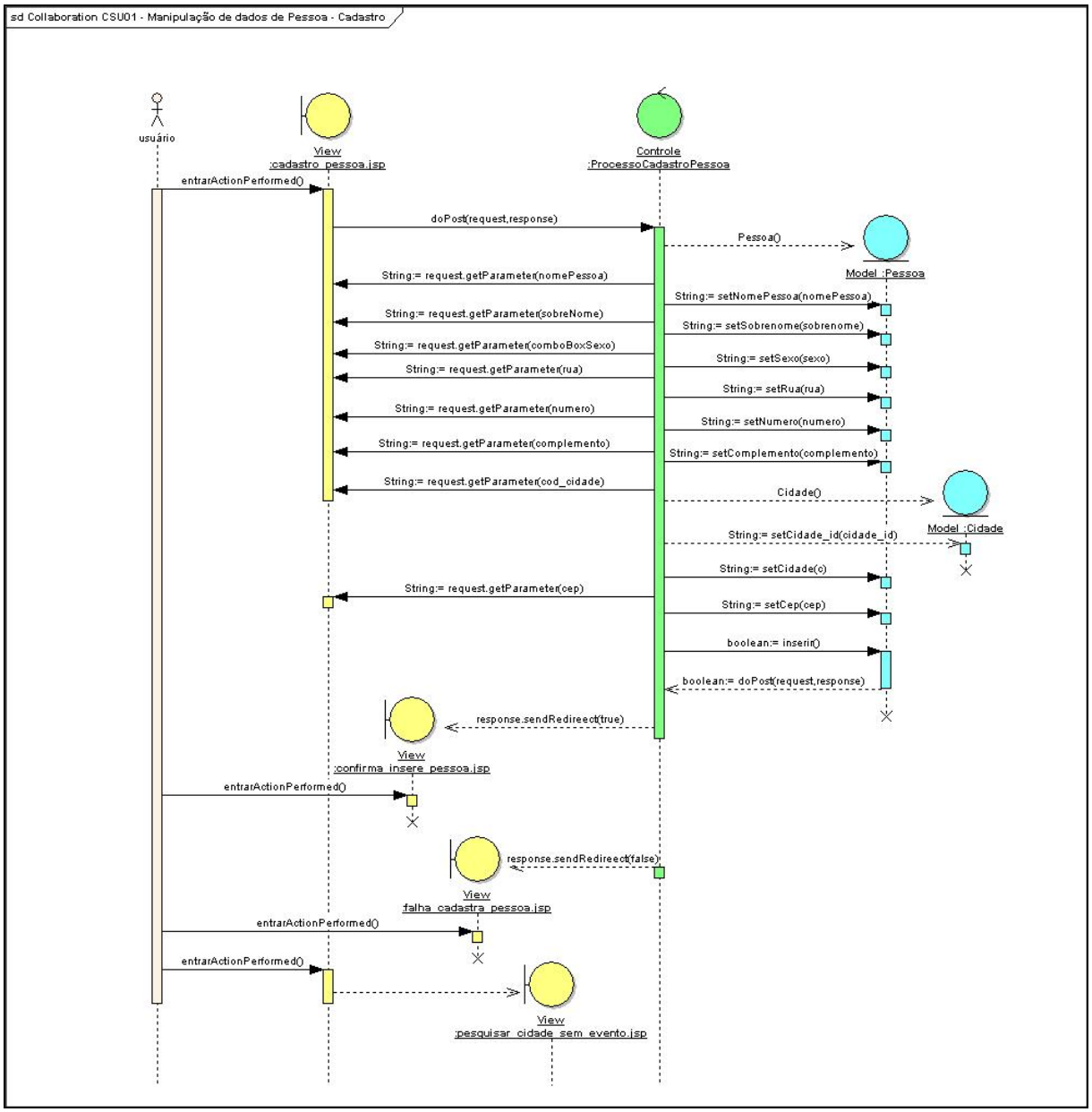


Figura 54 – Diagrama de seqüência do CSU01 – Web tradicional - cadastrar pessoa
Fonte: Os autores.

Na figura 55 demonstra a diagrama de seqüência do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

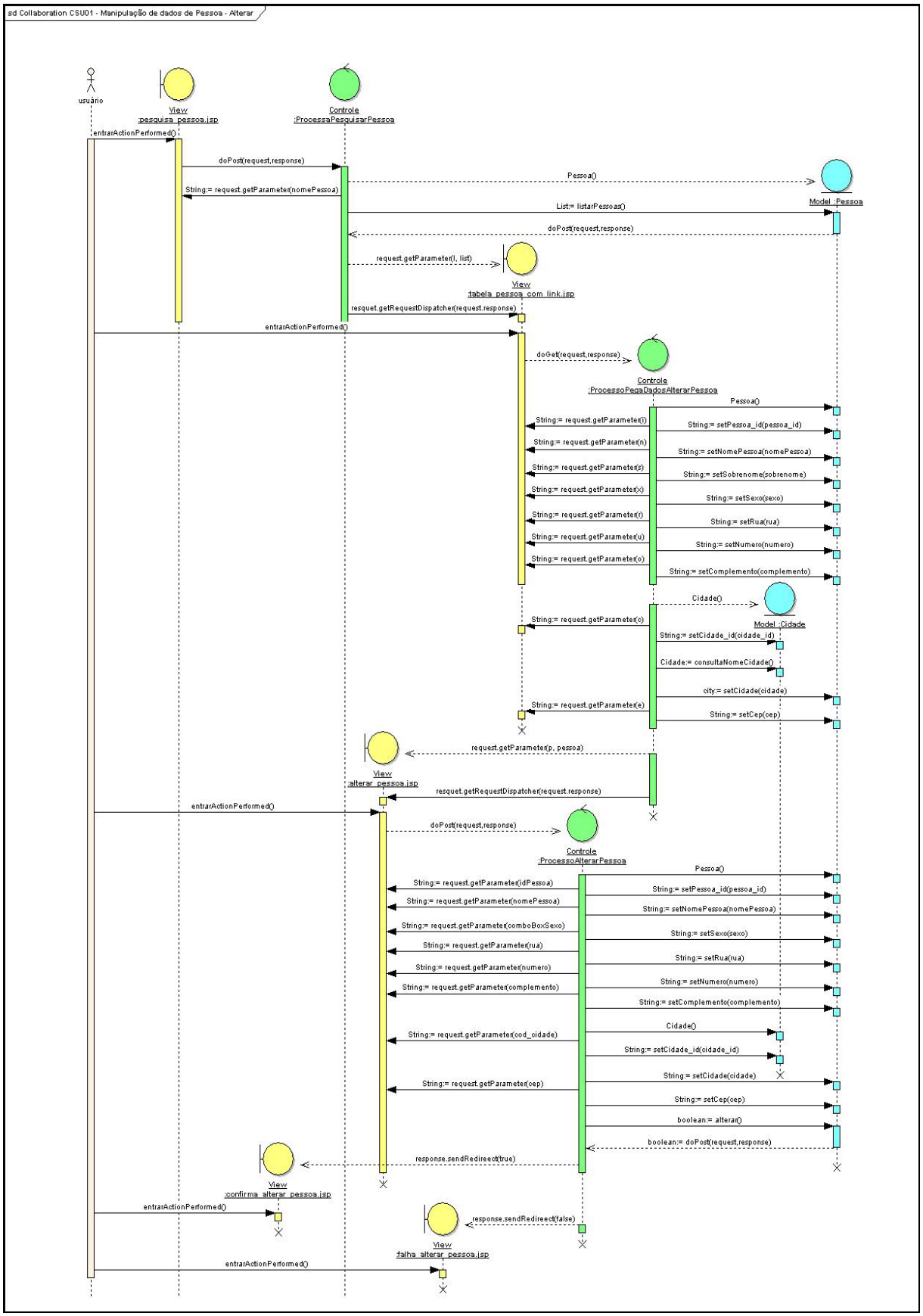


Figura 55 – Diagrama de seqüência do CSU01 – Web tradicional - alterar pessoa
Fonte: Os autores.

Na figura 56 demonstra a diagrama de seqüência do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

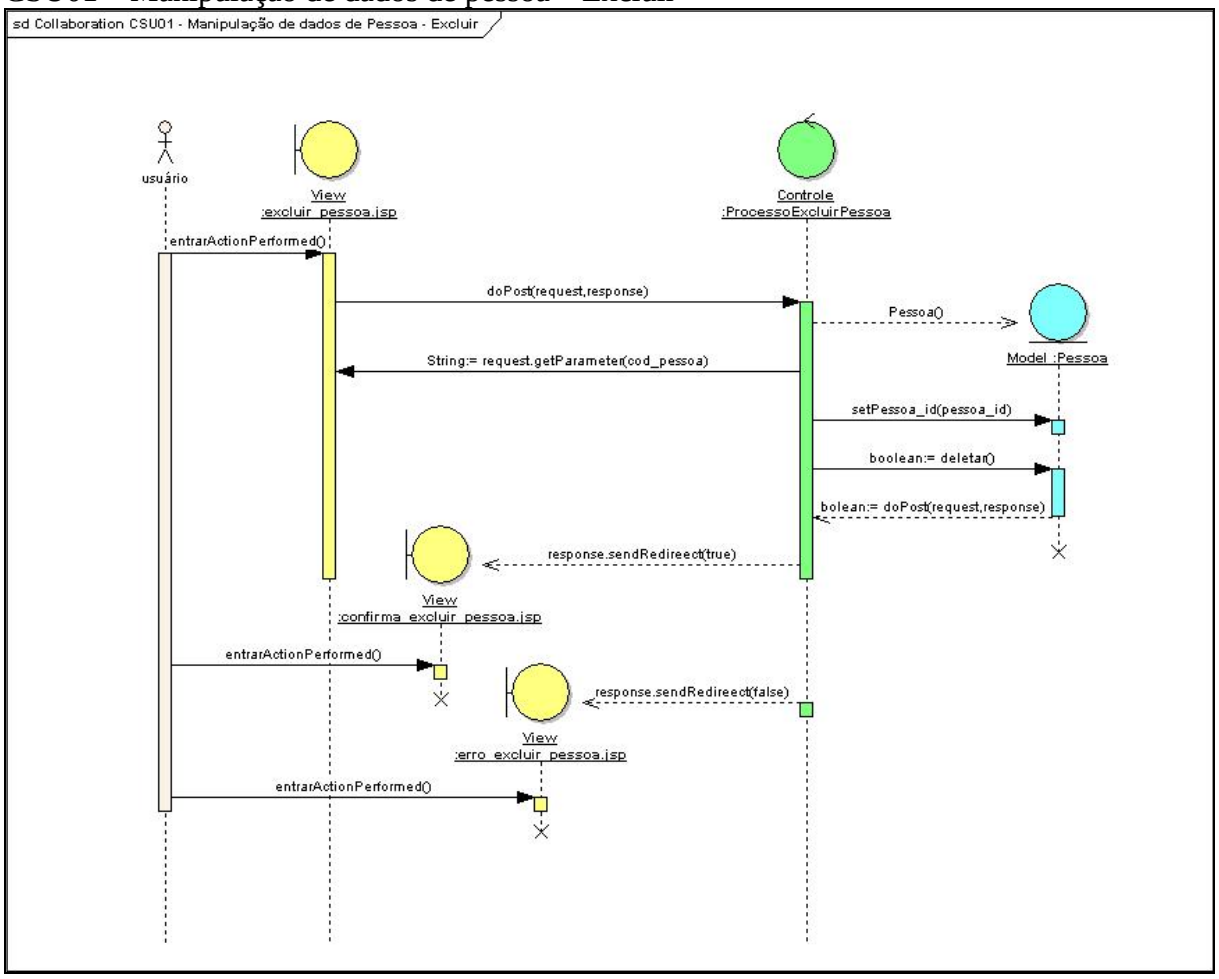


Figura 56 – Diagrama de seqüência do CSU01 – Web tradicional - excluir pessoa
Fonte: Os autores.

3.14 PROTÓTIPO WEB COM AJAX

Veremos a seguir a modelagem do protótipo Web com AJAX.

Na figura 57 temos a demonstração do menu do protótipo Web com AJAX.

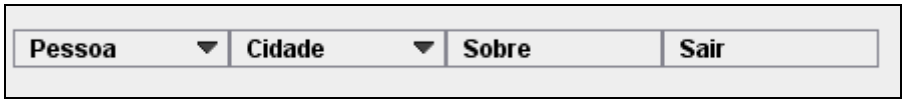


Figura 57 – Menu do protótipo Web com AJAX
Fonte: Os autores.

3.14.1 Requisitos Funcionais

RNF01 – Uso de falso botão

Descrição: por impossibilidade da plataforma em utilizar botões com link, utilizamos a técnica de falsos botões com link.

3.14.2 Caso de Uso

3.14.2.1 CSU01 - Manipulação de dados de pessoa

3.14.2.1.1 Manipulação de dados de pessoa - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de pessoa, realizado o cadastro, exclusão, alteração de pessoa.

Fluxo de Eventos – Cadastro

Fluxo Básico

1. Usuário preenche o campo nome;
2. Usuário preenche o campo sobrenome;
3. Usuário seleciona o campo sexo;
4. Usuário preenche o campo rua;
5. Usuário preenche o campo número;
6. Usuário preenche o campo complemento;
7. Usuário preenche o campo número da cidade;
8. Sistema preenche o nome da cidade e o estado;
9. Usuário preenche o campo cep;
10. Usuário clica em ‘Cadastrar’;
11. Sistema cadastra nova pessoa na base de dados.

Fluxo Alternativo

a) < Fluxo Alternativo 1>

1. Usuário não sabe o número da cidade, então clica em Consulta Cidade;
2. Sistema ativa a tela de pesquisa;
3. Usuário digita o nome da cidade desejada e clica em Pesquisar;
4. Sistema habilita tela com as cidades relacionadas com o nome digitado na pesquisa;

b) < Fluxo Alternativo 2>

1. Usuário clica na opção Limpar;
2. Sistema limpa as informações da tela.

Fluxo de Exceção

a) < Fluxo Exceção 1>

1. Usuário fecha *browser* direto, sem executar qualquer tipo de funcionalidade;

Interface Gráfica:

Na figura 58 demonstra a interface do caso de uso de cadastro de pessoa. Conforme o processo de do fluxo básico.

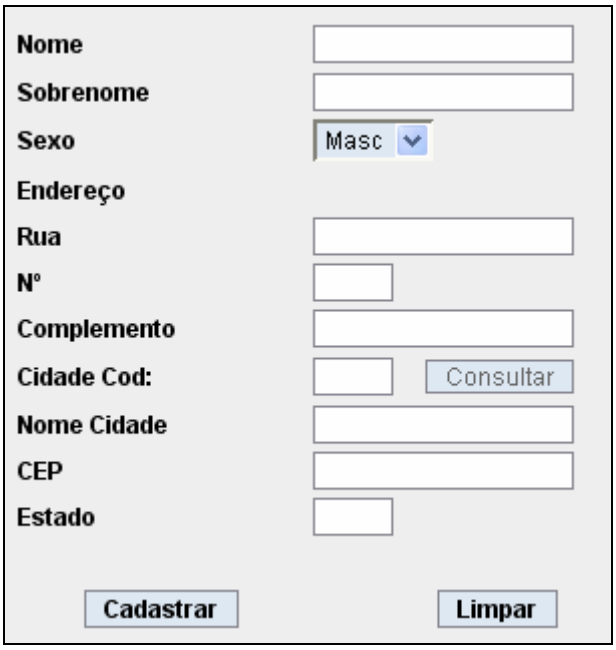


Figura 58 – Interface Web com AJAX – tela de cadastro de pessoa
Fonte: Os autores

Na figura 59 demonstra o resultado da pesquisa de cidade no campo de cadastro de pessoa. Processo descrito no fluxo alternativo 1.

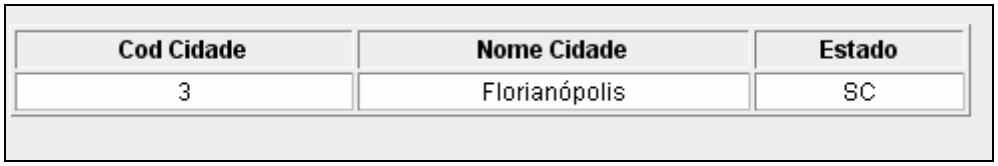


Figura 59 – Interface Web com AJAX – tela de resultado da pesquisa de cidade
Fonte: Os autores

3.14.2.1.2 Manipulação de dados de pessoa - Excluir

Nesse caso de uso o usuário realiza a exclusão de Pessoa.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código de pessoa;
2. Usuário clica em Excluir;
3. Sistema exclui a pessoa desejada do banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisar;
2. Sistema habilita tela de pesquisa;
3. Usuário digita o nome da pessoa desejada;
4. Sistema habilita tela com as pessoas relacionadas.
5. Execute o fluxo básico com o número do registro encontrado.

b) <Fluxo Alternativo 2>

1. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

c) <Fluxo de Exceção 1>

1. Usuário fecha o *browser* direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 60 demonstra a interface do caso de uso de exclusão de pessoa.

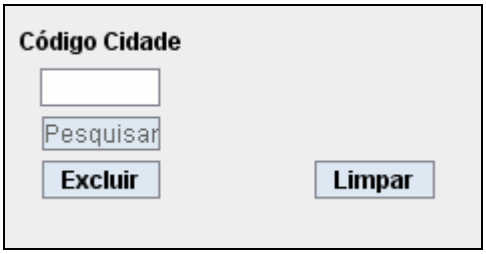
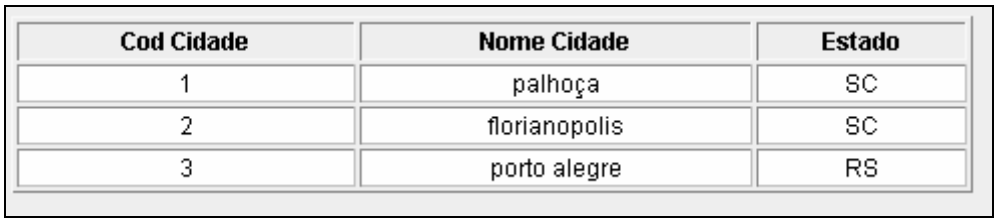


Figura 60 – Interface Web com AJAX – tela de excluir pessoa
Fonte: Os autores.

Na figura 61 demonstra o resultado da pesquisa da cidade para descobrir qual o código da cidade para ser excluída. Conforme fluxo alternativo 01.



Cod Cidade	Nome Cidade	Estado
1	palhoça	SC
2	florianopolis	SC
3	porto alegre	RS

Figura 61 – Interface Web com AJAX – tela de resultado de pesquisa cidade
Fonte: Os autores.

3.14.2.1.3 Manipulação de dados de pessoa - Alterar

Nesse caso de uso o usuário realiza a alteração de Pessoa.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a pesquisa
2. Usuário clica em EDITAR
3. Usuário realiza a alteração dos dados
4. Usuário clica em alterar
5. Sistema altera os dados de pessoa no banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

A figura 62 demonstra a interface do caso de uso de alteração de pessoa.

Figura 62 – Interface Web com AJAX – tela de alteração de pessoa
Fonte: Os autores.

3.15 DIAGRAMA DE PACOTES

Em muitos casos um único diagrama de classes pode ser exageradamente grande para representar todo o sistema. Assim é conveniente utilizar-se de um elemento para organizar os subsistemas do modelo. Para isto utilizam-se os diagramas de pacote. Um pacote representa um grupo de classes (ou outros elementos) que se relaciona com outros pacotes através de uma relação de dependência.

Na figura 63 temos o diagrama de pacotes do protótipo Web com AJAX.

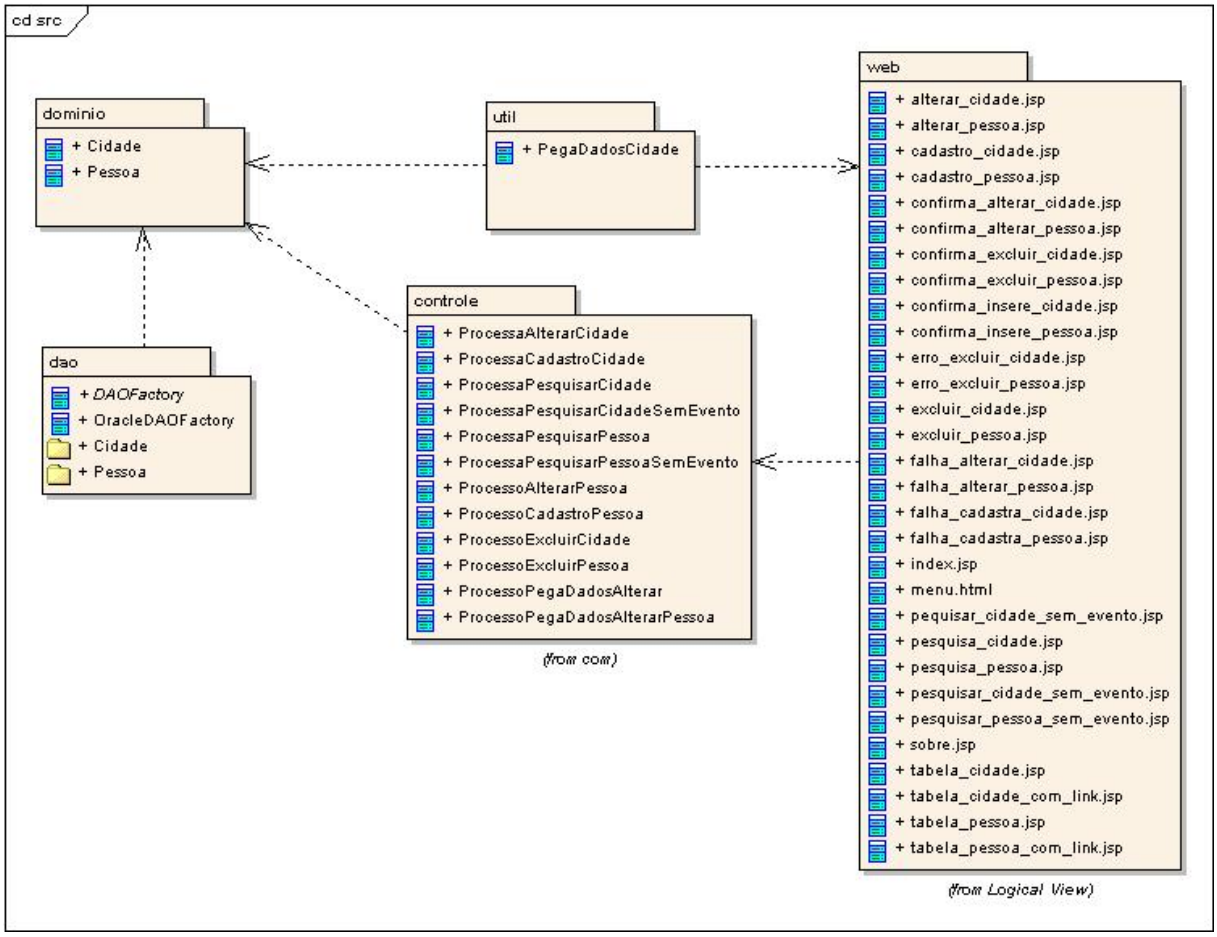


Figura 63 – Diagrama de pacotes do protótipo Web com AJAX
Fonte: Os autores.

3.16 DIAGRAMA DE CLASSES

Diagrama de Classes demonstra um conjunto de classes, interfaces e colaborações que compõem o sistema e as relações entre elas (por exemplo, a herança). Trata de um aspecto estático e estrutural do sistema.

Na figura 64 temos demonstrado o diagrama de classes do protótipo Web com AJAX. Como verificado no código fonte, as controle não apresentam ligações entre sim.

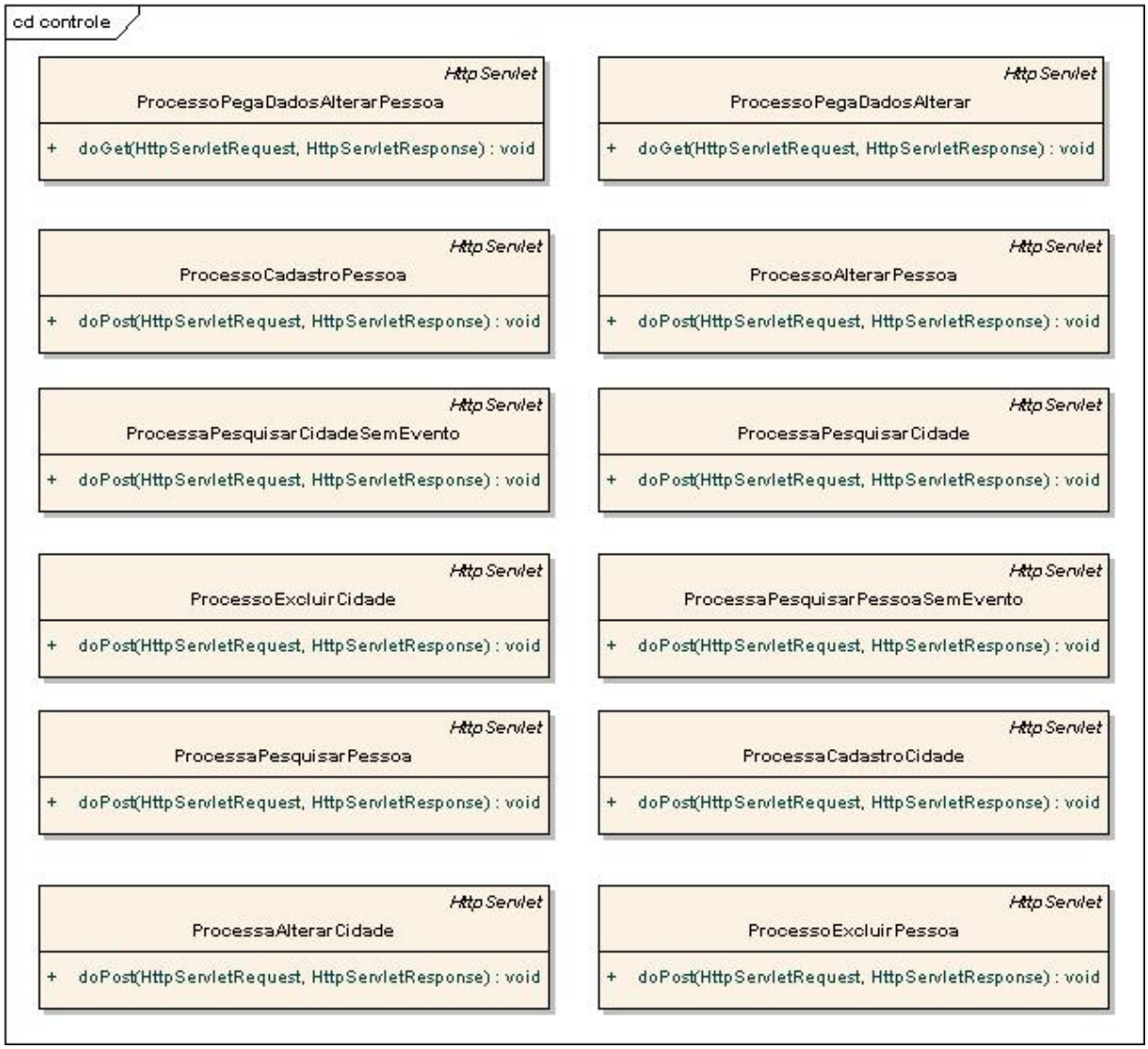


Figura 64 – Diagrama de classe do protótipo Web com AJAX
Fonte: Os autores.

3.17 DIAGRAMA DE ROBUSTEZ

Nesta etapa serão descritos os diagramas de robustez. São ilustradas graficamente as iterações entre os objetos participantes do caso de uso.

Na figura 65 demonstra a diagrama de robustez do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa - Cadastrar

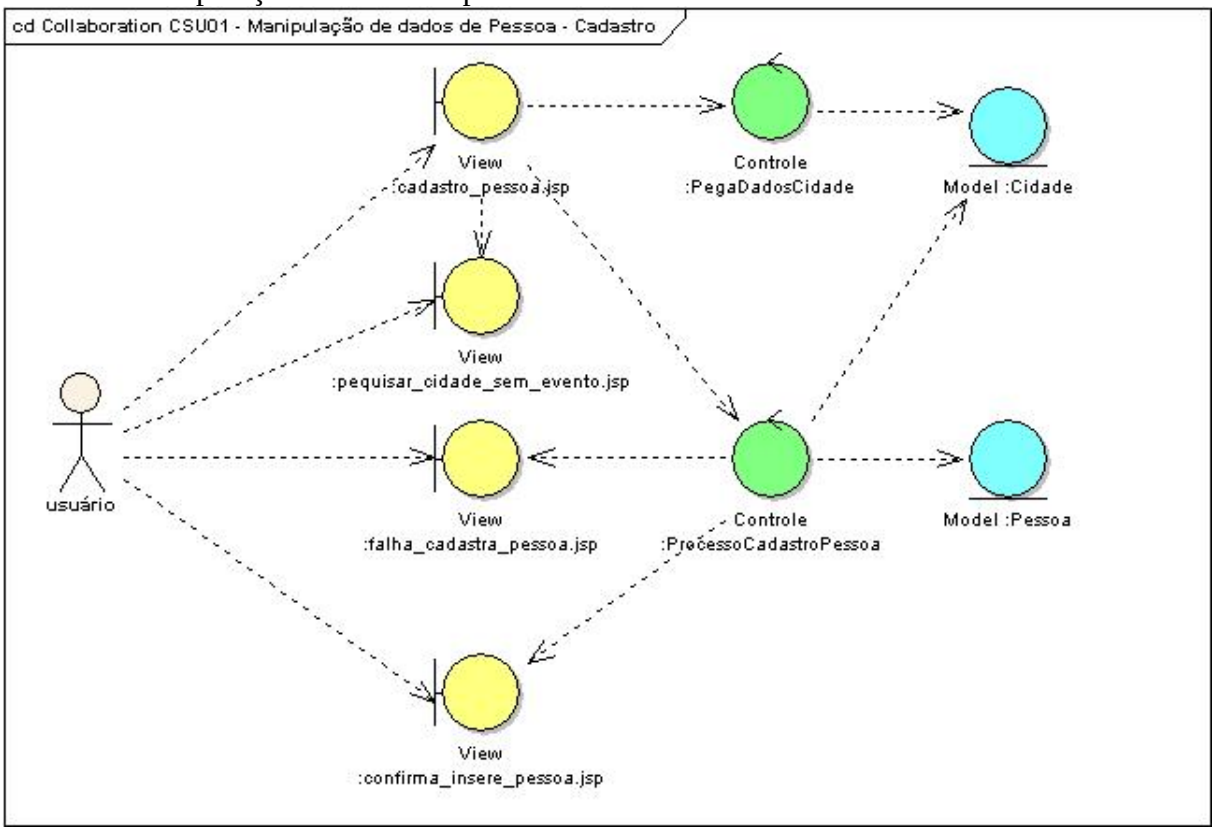


Figura 65 – Diagrama de robustez do CSU01 – Web com AJAX - cadastrar pessoa
Fonte: Os autores.

Na figura 66 demonstra a diagrama de robustez do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

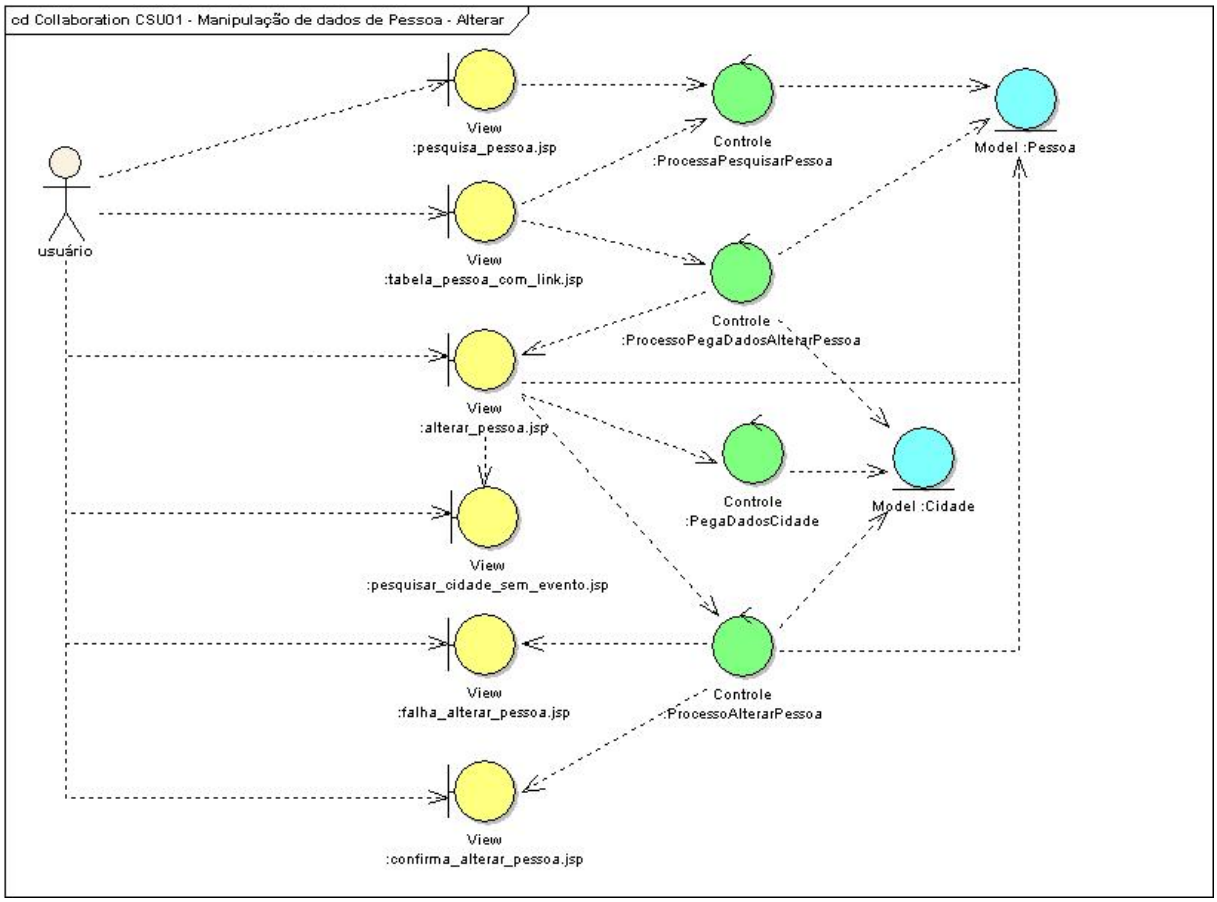


Figura 66 – Diagrama de robustez do CSU01 – Web com AJAX - alterar pessoa
Fonte: Os autores.

Na figura 67 demonstra a diagrama de robustez do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

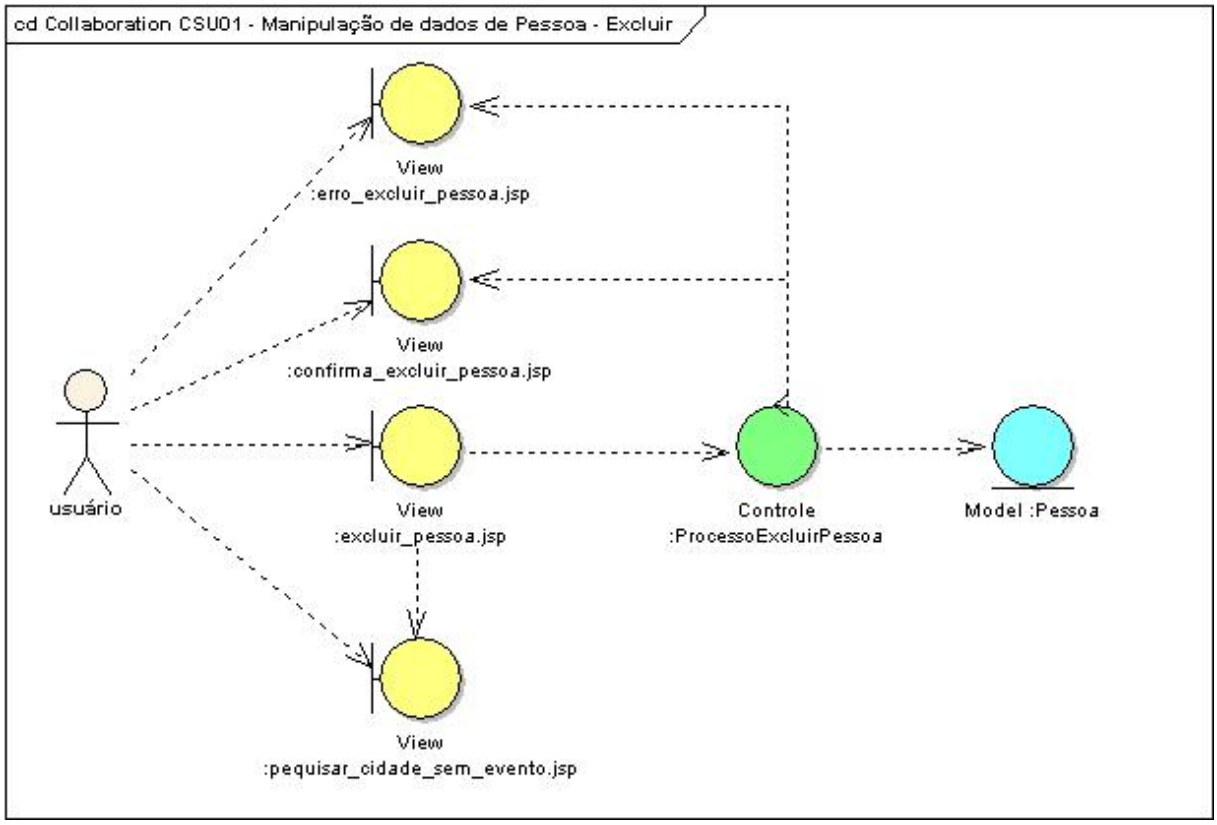


Figura 67 – Diagrama de robustez do CSU01 – Web com AJAX - excluir pessoa
Fonte: Os autores.

3.18 DIAGRAMA DE SEQUÊNCIA

Nesta etapa são construídos os diagramas de seqüência que tem como finalidade instruir o desenvolvimento do protótipo Web com AJAX.

Na figura 68 demonstra o diagrama de seqüência do caso de uso cadastrar pessoa.

CSU01 – Manipulação de dados de pessoa – Cadastro

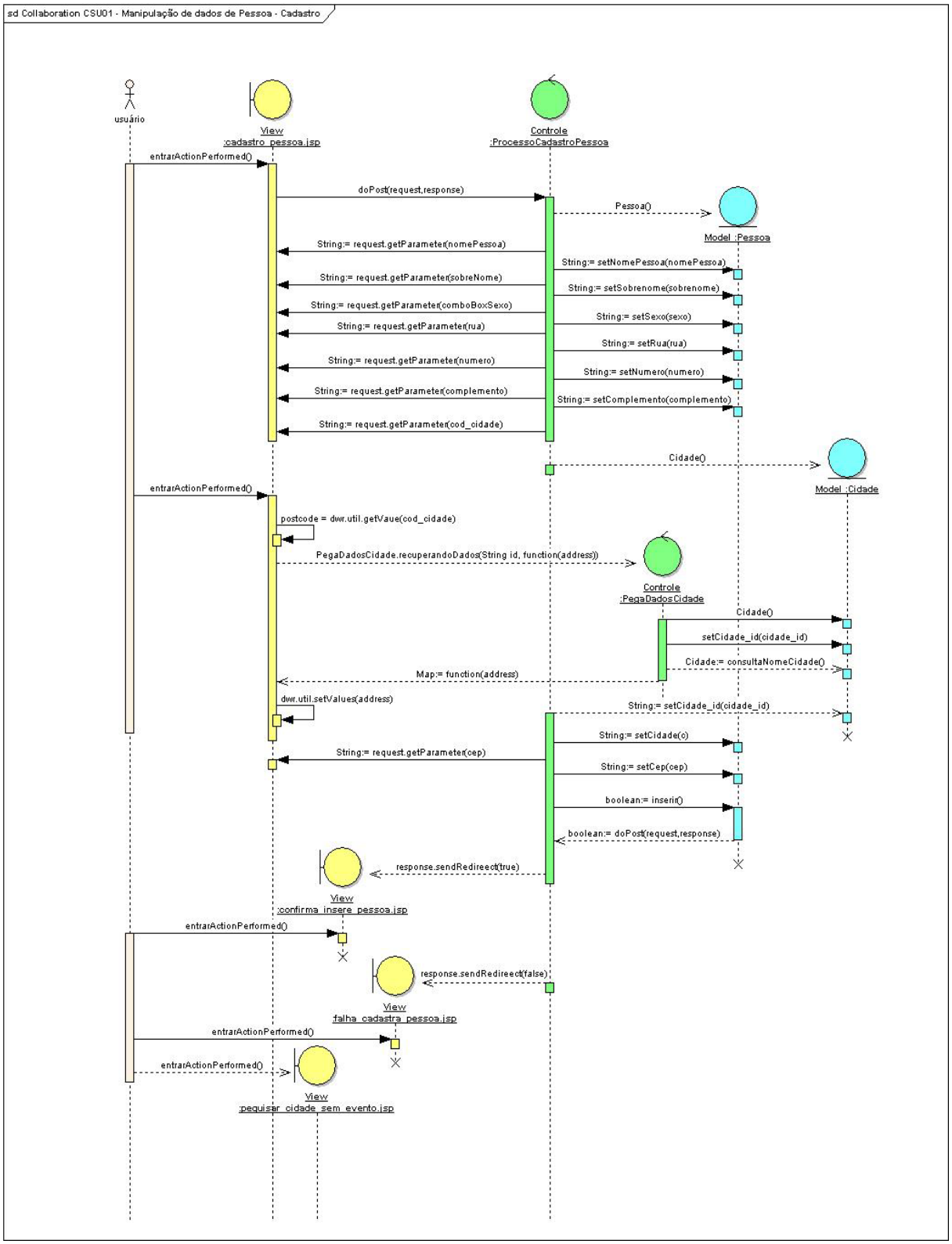


Figura 68 – Diagrama de seqüência do CSU01 – Web com AJAX - cadastrar pessoa
Fonte: Os autores.

Na figura 69 demonstra a diagrama de seqüência do caso de uso alterar pessoa.

CSU01 – Manipulação de dados de pessoa – Alterar

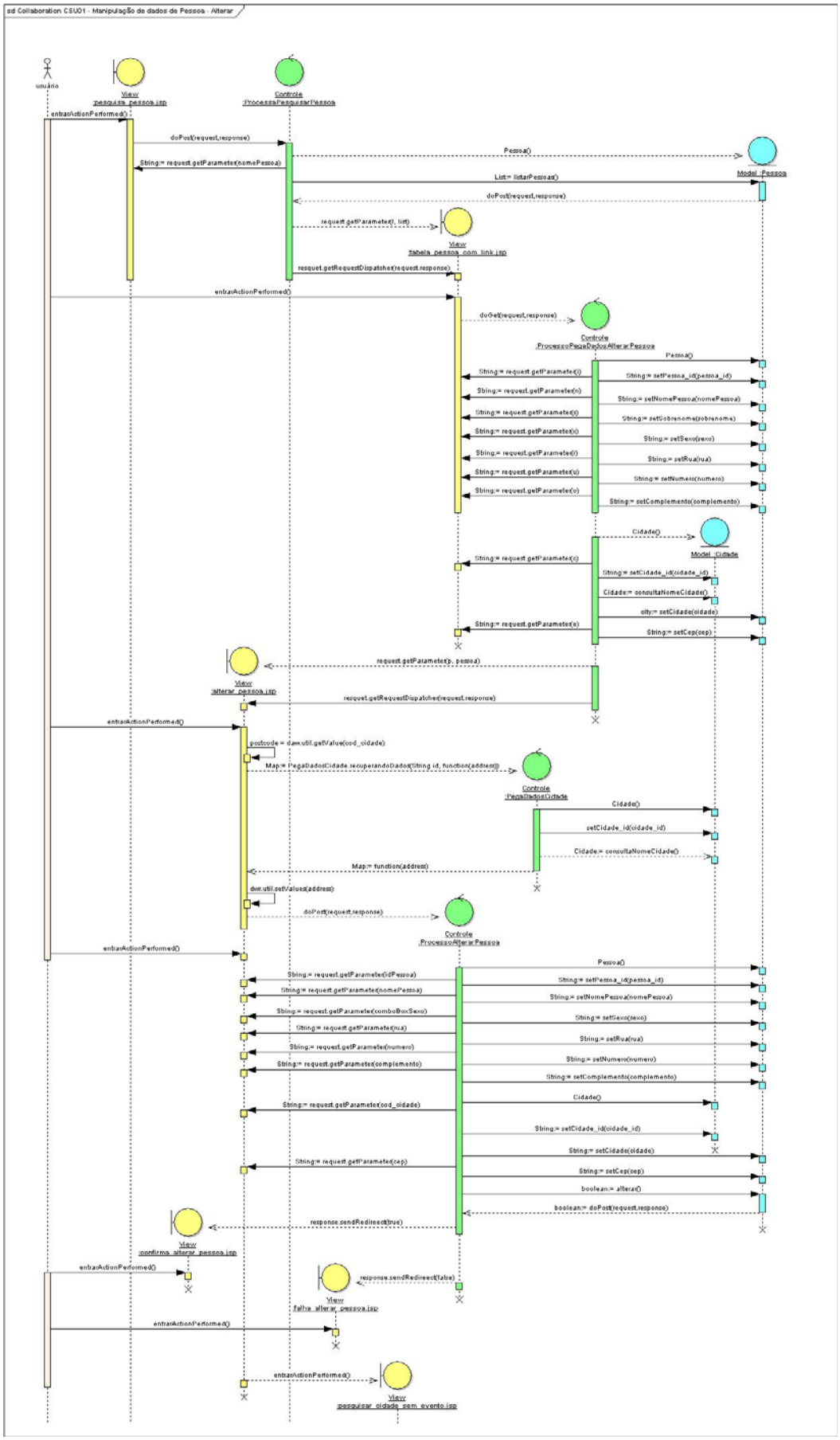


Figura 69 – Diagrama de seqüência do CSU01 – Web com AJAX - alterar pessoa
Fonte: Os autores.

Na figura 70 demonstra a diagrama de seqüência do caso de uso excluir pessoa.

CSU01 – Manipulação de dados de pessoa – Excluir

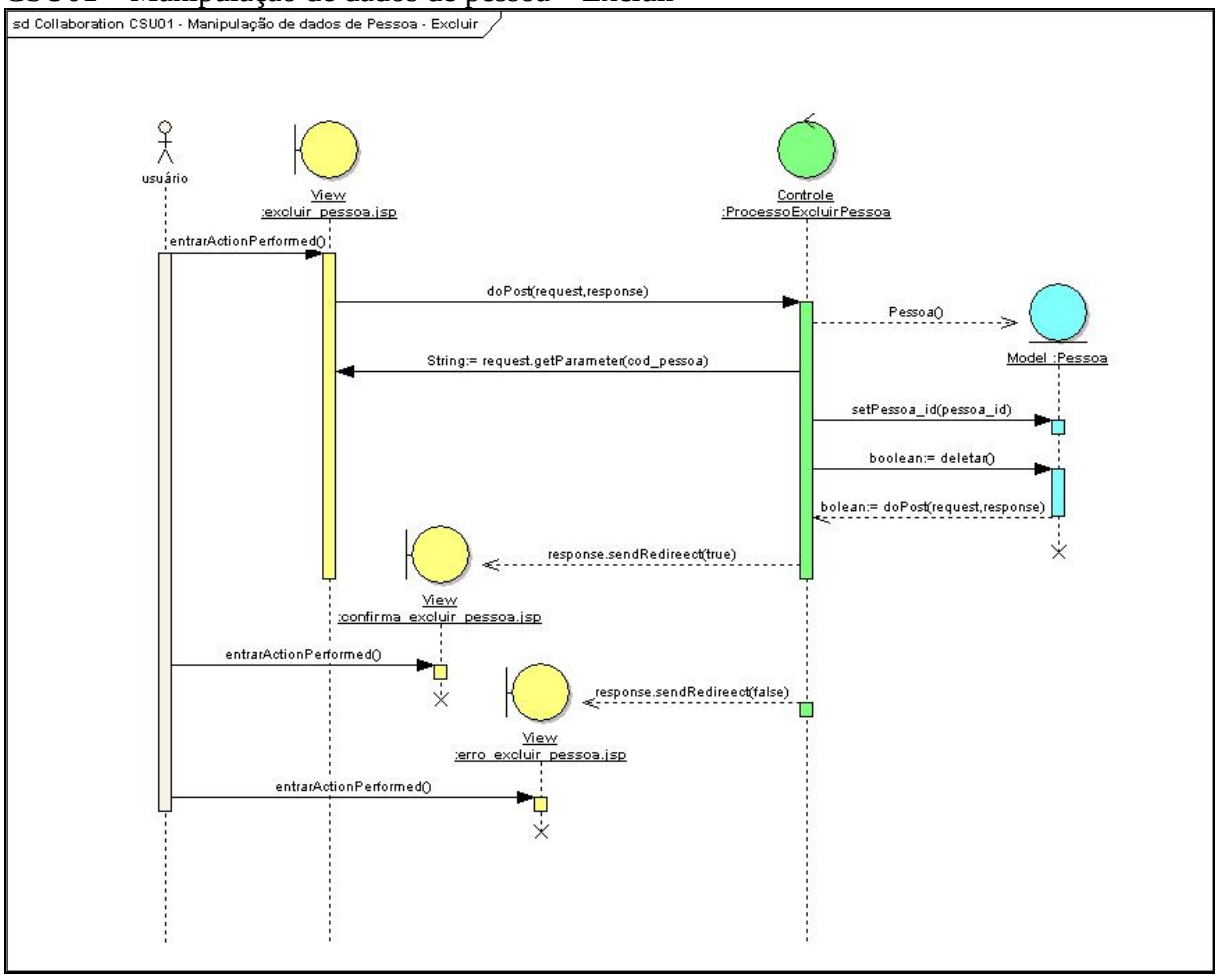


Figura 70 – Diagrama de seqüência do CSU01 – Web com AJAX - excluir pessoa
Fonte: Os autores.

4 FERRAMENTAS COMPUTACIONAIS

Na figura 71 demonstra as ferramentas computacionais utilizada para a criação da monografia e seus desenvolvimentos.

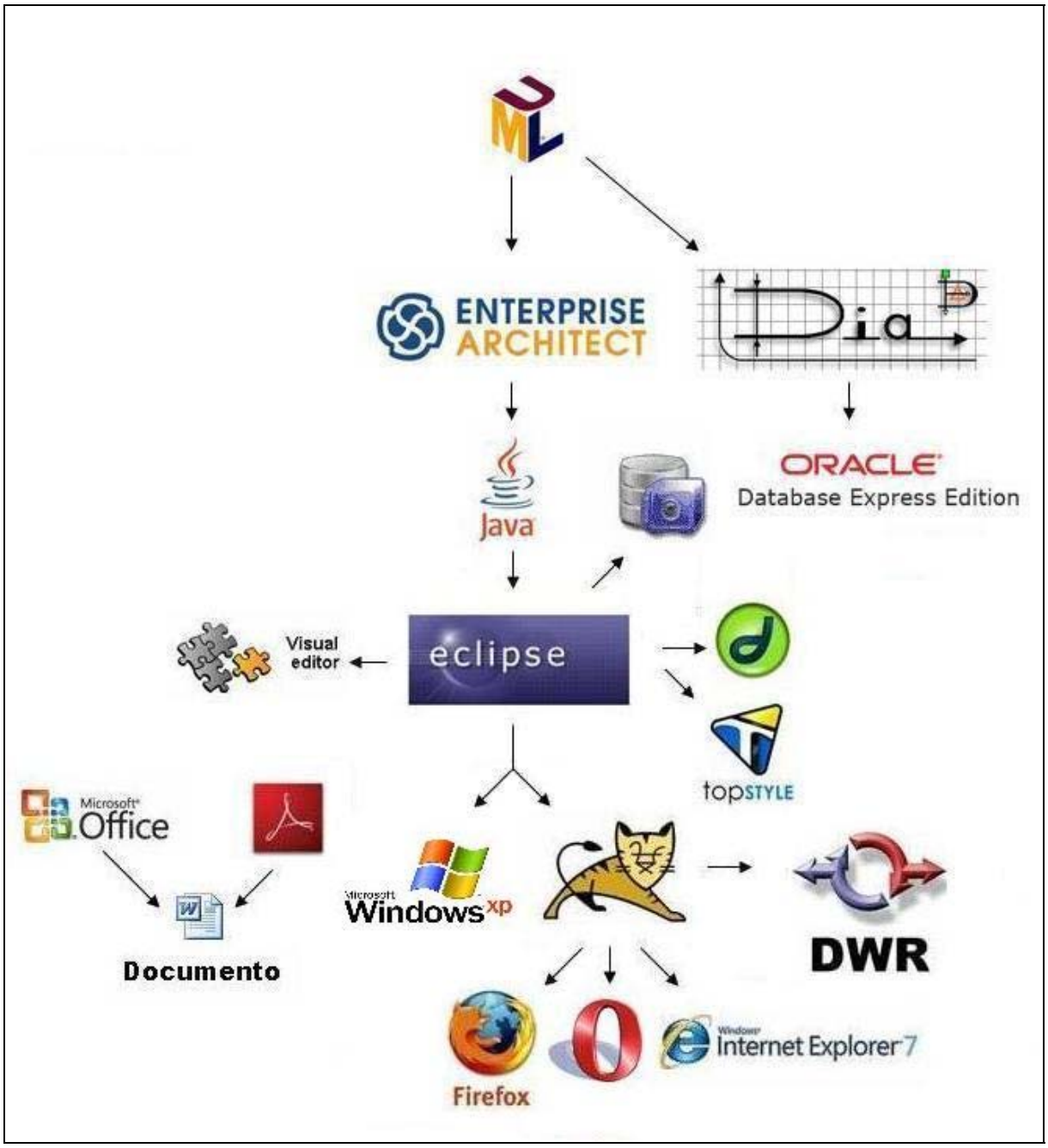


Figura 71 – Ferramentas computacionais utilizadas.
Fonte: Os autores.

Ferramentas utilizadas

Eclipse: utilizado a IDE eclipse 3.x para o desenvolvimento do protótipo Desktop, protótipo Web e Web com AJAX.
Enterprise Architech: Utilizado a versão 6.0 para elaboração das modelagens dos protótipos.
Oracle 10g XE: utilizado para servir como banco de dados para as três modelagens.
Apache Tomcat: utilizado como servidor Web.
Dia: Utilizado para gerar o desenho do modelo ER.
TopSyle: Utiliza para criação dos arquivos scc para os protótipos Webs.
Microsft Office 2003: Pacote utilizado para elaboração do documento da monografia, criação da apresentação e planilha de controle de horas.
Firefox: *Browser* utilizado para rodar os protótipos Web tradicional e Web AJAX.
Opera: *Browser* utilizado para rodar os protótipos Web tradicional e Web AJAX.
Internet Explorer versões 6 e 7: *Browser* utilizado para rodar os protótipos Web tradicional e Web AJAX.
DWR: *Framework* AJAX utilizado para o desenvolvimento do protótipo Web com AJAX.

5 HISTÓRICO DO DESENVOLVIMENTO

Para a realização desse trabalho tivemos inúmeras dificuldades, nas diferentes etapas do desenvolvimento do projeto, as quais nós superamos uma a uma com muita coragem e determinação.

Durante os levantamentos de requisitos para a realização desta monografia, tivermos muita dificuldade para encontrar modelos de modelagem de alto nível com o uso do AJAX em três camadas.

Nosso curso não proporciona nenhum conhecimento para desenvolvimento Web com: JavaScript, XML e CSS. Tivemos que ter muita coragem em aceitar todos esses desafios, em aprender todas estas linguagens para dominar a técnica AJAX.

Para que fosse possível comprovar que era possível a migração de uma aplicação Desktop para Web e Web com AJAX, usando modelos compartilhados; Era necessários desenvolver três modelagens e o desenvolvimento de três aplicações, num curto período de tempo.

Realizamos a primeira etapa do projeto com os capítulos uma (contextualização de pesquisa deste trabalho) e dois (revisão bibliográfica), em seguida começamos a modelagem e implementações para cada protótipo;

Nos baseamos nas melhores praticas da metodologia ICONIX para gerar a documentação da modelagens.

Não tínhamos conhecimento de desenvolvimento, nem de modelagem orientado a objeto utilizando três camadas, MVC.

Não havíamos utilização JDBC;

Mesmo assim aceitamos todas essas dificuldades e fizemos à modelagem e as implementações dos três protótipos usando o paradigma de orientação a objetos e padrão de arquitetura de software MVC e a conexão com um driver JDBC para o banco Oracle.

A modelagem foi feita utilizando-se a ferramenta EA. Onde nossa maior dificuldade com essa ferramenta foi em aprender a utilizá-la adequadamente. Dentre as dificuldades encontradas a importação das classes do JAVA para dentro da modelagem, facilitando o trabalho e dando mais credibilidade as modelagens dos protótipos.

A partir do desenvolvimento das modelagens com a ferramenta EA, gerávamos a estrutura básica das fontes do projeto, bastando desenvolver o corpo dos métodos necessários para cada classe criada a partir do modelo do protótipo.

Concentramos nossos esforços para que tudo o que queríamos implementar estivesse bem modelado, para assim analisar as três modelagens e gerar nossos resultados com qualidade e credibilidade.

Começamos pelo protótipo Desktop, onde desenvolvemos a camada View na API Swing da Linguagem JAVA, a camada Control, que foram únicas para esse protótipo. A camada Model e de persistência juntamente com as tabelas do banco de dados foi desenvolvida no primeiro protótipo e foram reutilizadas ou compartilhadas com os outros dois protótipos.

Reaproveitado as tabelas do banco de dados juntamente com as camadas Model e de persistência, desenvolvemos o protótipo Web utilizando JSP na camada View e Servlet na camada Control. Como já era esperado nesse protótipo conseguimos fazer somente processos síncronos e percebeu-se a falta da interatividade que era presente no protótipo Desktop com processos assíncronos.

Para o terceiro protótipo, Web com AJAX, foi reutilizado todas as camadas do segundo, onde aproveitamos todas as camadas fazendo algumas pequenas modificações que não tiveram muito impactos no desenvolvimento, adicionando a esse protótipo o Framework DWR para que o protótipo AJAX pudesse realizar as mesmas funcionalidades assíncronas como o protótipo Desktop.

Realizamos a comparação do diagramas criados nas modelagens, para que fosse possível ser realizada uma análise criteriosa, onde demonstramos o que havia sido compartilhado, o que era semelhante e o que era diferente, na modelagem e nas implementações dos três protótipos. Através da análise chegamos as nossas conclusões.

Tivemos todas estas dificuldades entre outras, para chegarmos até o fim do projeto, mas a luta para que chegássemos ao fim nós ajudou a melhorar nossos conhecimentos e a sermos melhores profissionais.

6 ANÁLISE DOS RESULTADOS

6.1 Análise da Modelagem

No processo da análise de resultados desta monografia, iremos analisar a funcionalidade e os processos desenvolvidos.

No quadro 5 detalhamos os itens que foram diferentes e semelhantes, com relação à análise da modelagem.

Não iremos comentar os itens que são iguais.

Quadro de resultados para modelagem			
	Desktop	**Web Tradicional**	**Web AJAX**
Prototipação de interface	=	=	=
Requisitos Não Funcionais	~	=	=
Requisitos Funcionais	=	=	=
Diagrama Casos de Usos	=	=	=
Diagrama de Robustez	<>	~	~
Diagrama de Seqüência	~	<>	~
Diagrama de Classes (Domínio)	=	=	=
Diagrama de Classes (DAO)	=	=	=
Diagrama de Classes (Controle)	<>	=	=
Diagrama de Pacotes	<>	~	~

Quadro 5 – de resultados para a modelagem
Fonte: Os autores.

Legenda:
= igual.
<> diferente.
~ semelhante.

Requisitos Não funcionais:

O protótipo Desktop foi implementado utilizando a API *Swing* da linguagem Java, para descrever a portabilidade do protótipo Desktop em quaisquer sistemas operacionais que tenha suporte para JVM.

Diagrama de Robustez:

No protótipo Desktop as classes de controles fazem as instâncias com os novos objetos das *Views* e tornam-se visíveis através método *setVisible(true).* Por tanto as *views* são associadas às controles. Para haver uma *View* faz-se necessário uma controle.

No protótipo Web tradicional e Web com AJAX, as classes controles não fazem associação com as *View*, mas dependem das informações que são passadas da *View* para as controles, onde a controle redirecionará para uma nova *View* de resposta para o usuário.

No protótipo AJAX além das diferenças encontradas na Web tradicional encontra-se uma controle a mais que fará a atualização dos dados na mesma *View* que passou o parâmetro. Tornando semelhante e assíncrona como no protótipo Desktop. Analisaremos essa controle posteriormente no item 5.2 Análise do desenvolvimento.

Diagrama de Seqüência

No diagrama de seqüência dos protótipos Desktop e o Web com AJAX, antes do usuário entrar com a ação do processo Cadastrar e Alterar. O usuário pode realizar várias vezes as alterações do campo código cidade onde os protótipos alteram o nome dos campos cidade e estado. Esse procedimento da troca é realizado assicronamente, onde o processo assíncrono no protótipo Desktop é realizado pela chamada do método *focus()* conforme demonstrado na figura 72.

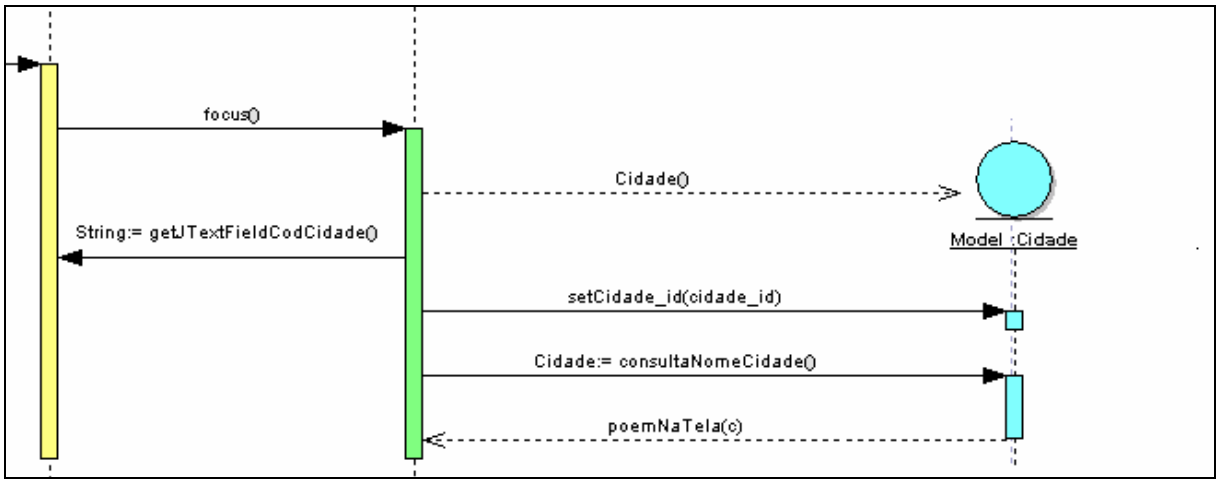


Figura 72 – Parte da modelagem do protótipo Desktop
Fonte: Os autores.

No protótipo AJAX, o processo realizado é por um conjunto de elementos, um da camada *view* o JavaScript *dados_cidade.js* e o outro da classe *PegaDadosCidade* através do método *recuperandoDados.* Conforme descreve a figura 73.

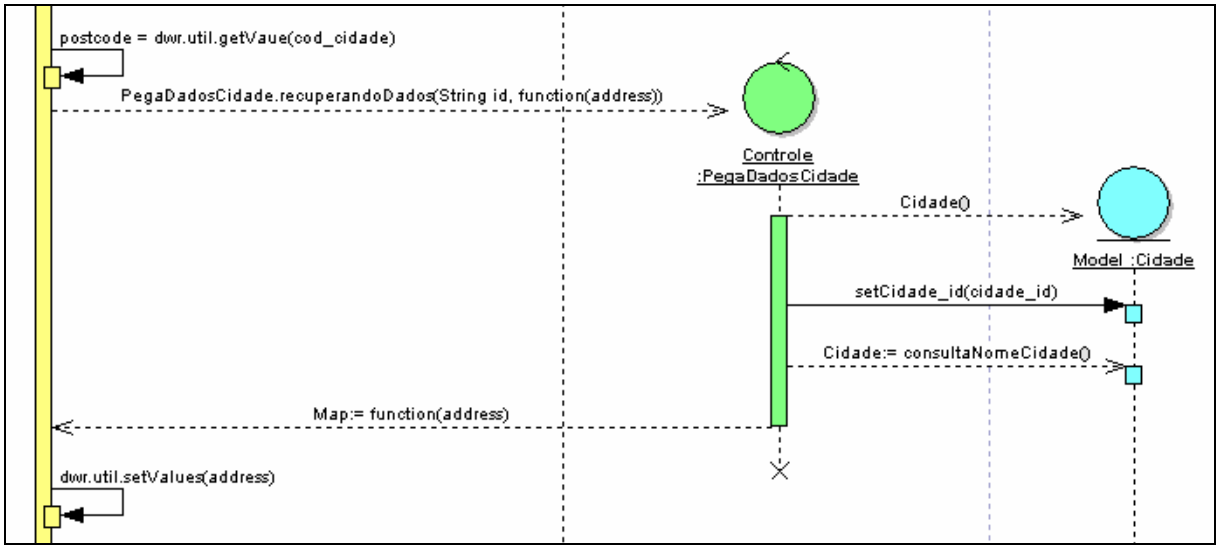


Figura 73 – Parte da modelagem do protótipo Web com AJAX
Fonte: Os autores.

Sem os recursos do AJAX não é possível atualizar os dados no mesmo formulário.

O processo de cadastrar, alterar e excluir os Objetos Pessoa e Cidade são diferentes, pois são sincronizados. O ato de realizar uma operação em conjugação com o entrosamento do seu controle específico, fará o efeito selecionado uma única vez. Conforme ilustrado nas figuras 74 para os protótipos para Web na *alterar_pessoa.jsp* e na figura 75 do protótipo Desktop *FormularioAlterarPessoa*.

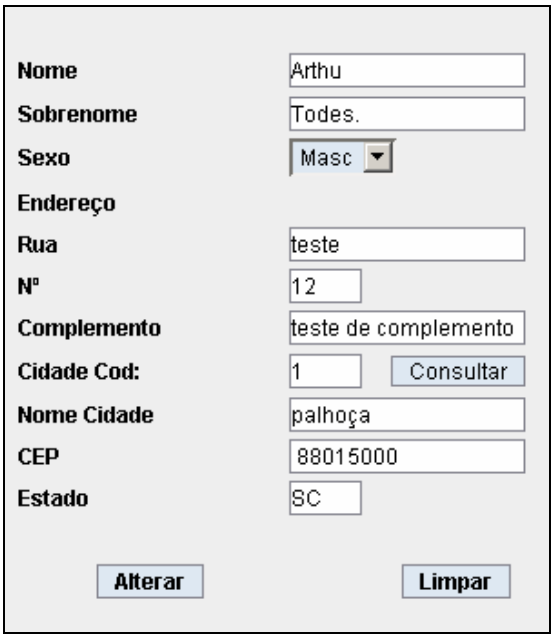


Figura 74 – tela de alterar dados Pessoa para os protótipos para Web
Fonte: Os autores.

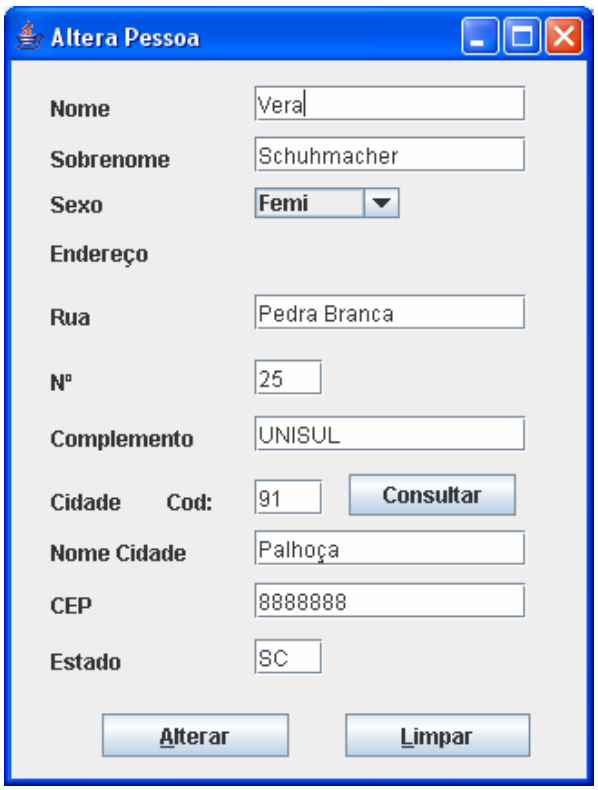


Figura 75 – tela de alterar dados Pessoa para o protótipo Desktop
Fonte: Os autores.

Diagrama de Pacotes

No diagrama de pacote dos protótipos Web tradicional e Web com AJAX, os controles não apresentam ligações entre si. Enquanto no protótipo Desktop, um controle depende de outro. A diferença entre os controles no protótipo Desktop é por causa das associações entre as controles para a chamada dos novos controles ou formulários. Enquanto nos outros dois protótipos Web não há ligações entre eles. Conforme a figura 76 dos protótipos para Web e figura 77 para o protótipo Desktop.

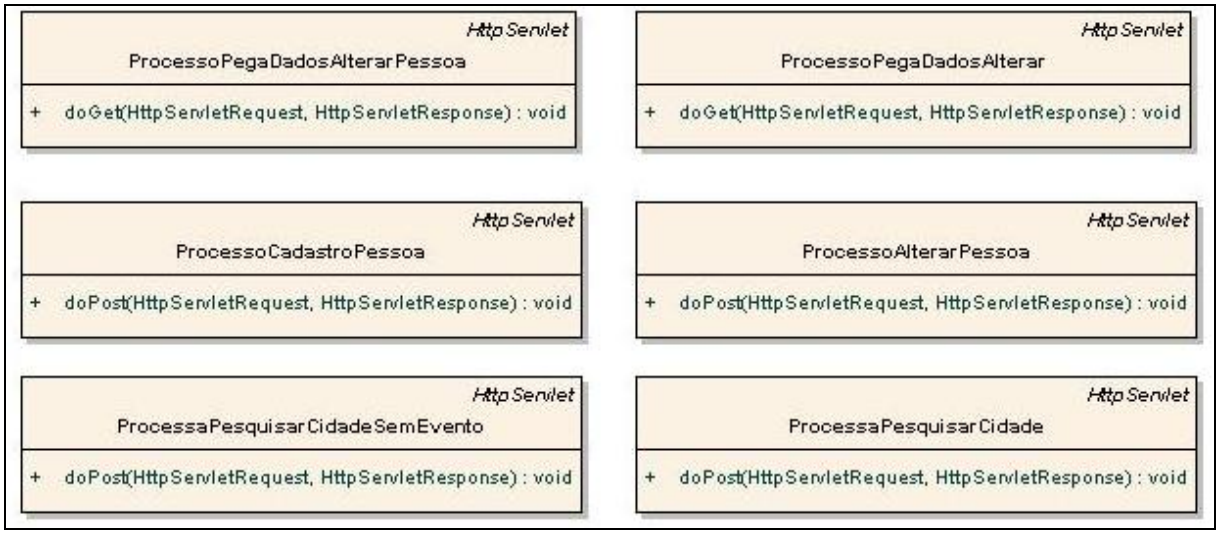


Figura 76 – Parte do diagrama de pacote do controle Web tradicional e Web com AJAX
Fonte: Os autores.

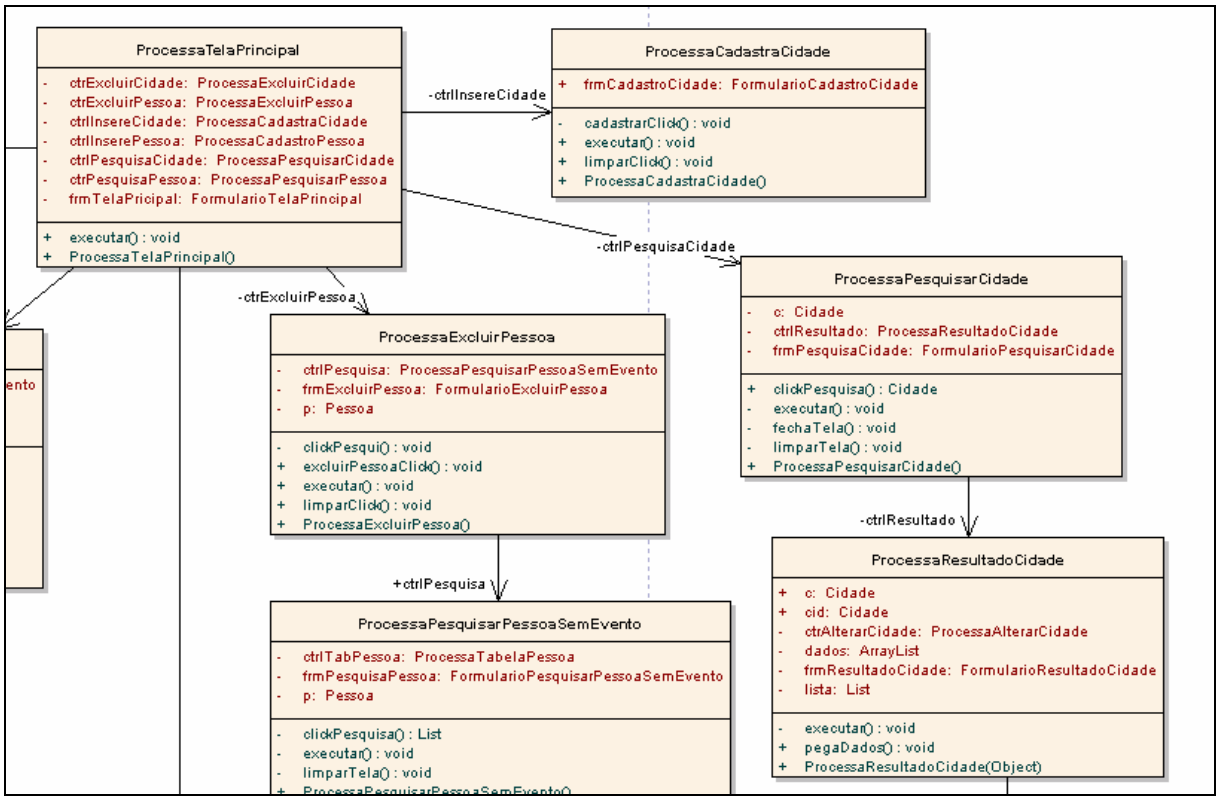


Figura 77 – Parte do diagrama de pacote do controle Desktop
Fonte: Os autores.

Analise dos diagramas de robustez:

Quadro de comparação dos diagramas de robustez			
	Desktop	**Web Tradicional**	**Web AJAX**
Cadastro de pessoa	<>	~	~
Cadastro de cidade	<>	=	=
Alterar pessoa	<>	~	~
Alterar cidade	<>	=	=
Pesquisar pessoa	<>	=	=
Pesquisar cidade	<>	=	=
Excluir pessoa	<>	=	=
Excluir cidade	<>	=	=

Quadro 6 – de resultados para a modelagem
Fonte: Os autores.

Legenda:
= igual.
<> diferente.
~ semelhante.

Cadastro de pessoa

Diferença na quantidade de controles apresentadas entre os diagramas de robustez dos protótipos.

No Desktop contêm 3 (três) controles.

No Web AJAX contêm 2 (duas) controles.

No Web tradiconal contêm apenas 1 (uma) controle.

A necessidade de mais controles no Desktop é por causa das associações entre as controles para a chamada dos novos formulários. Enquanto nos outros dois protótipos Web necessita apenas de um link de formulário para formulário. Conforme as figuras 78 do protótipo Desktop, figura 79 protótipo AJAX e figura 80 protótipo Web.

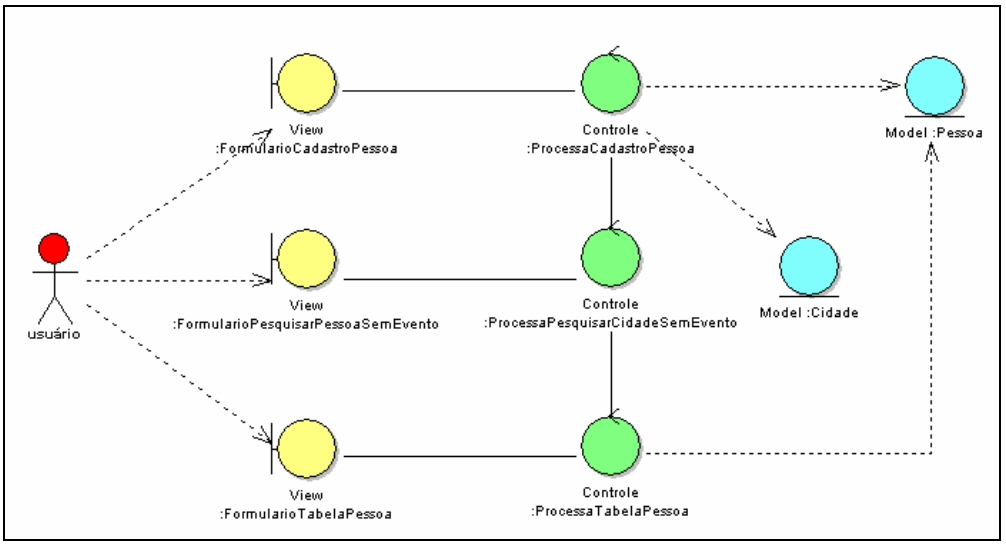


Figura 78 – Diagrama de robustez cadastra pessoa - Desktop
Fonte: Os autores.

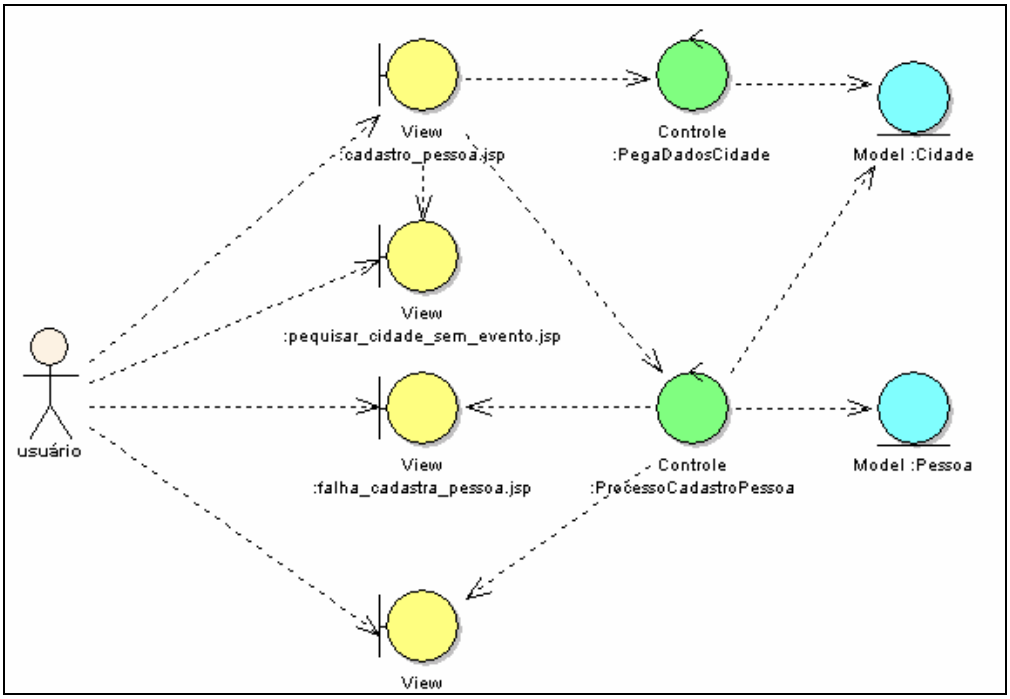


Figura 79 – Diagrama de robustez cadastra pessoa - AJAX
Fonte: Os autores.

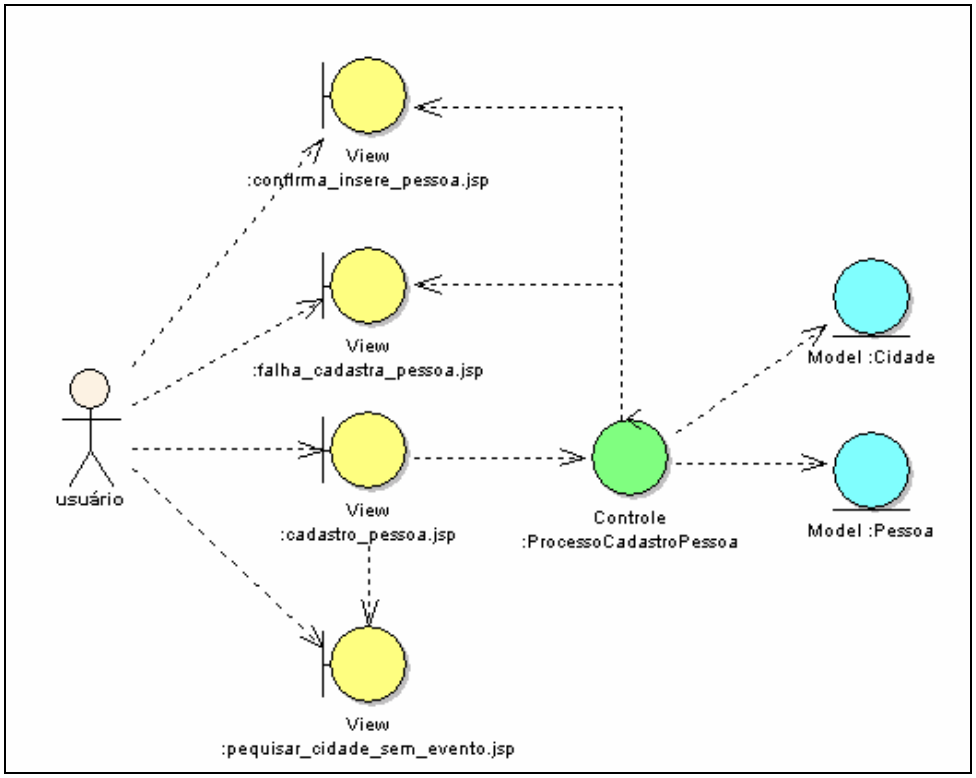


Figura 80 – Diagrama robustez cadastra pessoa – Web tradicional
Fonte: Os autores.

Alterar Pessoa

Diferença na quantidade de controles apresentadas entre os diagramas de robustez dos protótipos.

No Desktop contêm 3 (três) controles.

No Web AJAX contêm 4 (quatro) controles.

No Web tradicional contêm apenas 3 (três) controles.

A necessidade de mais controles no Web com AJAX é por causa da dependência com formulário altera pessoa com a controle *PegaDadosCidade* (evento AJAX).

Os demais itens do quadro 6 não serão abordados por diferirem no número de controles, que já foi analisado. Onde maior número de controles é apresentado no protótipo Desktop devido a associações entre as controles e o formulários.

Análise dos diagramas de seqüência

Percebeu-se que há sempre semelhanças na seqüência dos processos na utilização de modelos compartilhados, mesmo utilizando diferentes métodos para cada plataforma. Conforme veremos nas figuras 81 do protótipo Desktop e figura 82 dos protótipos para Web.

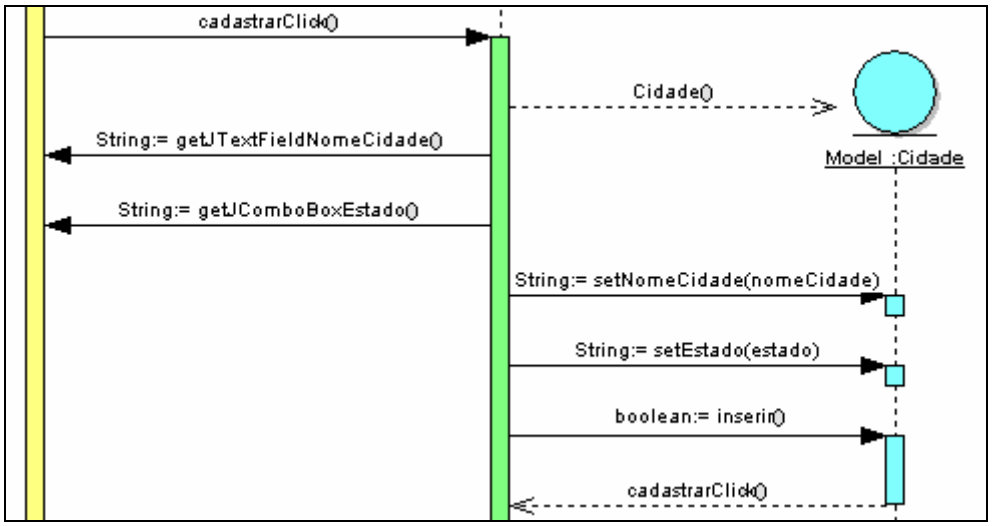


Figura 81 – Parte da modelagem de Cadastro Cidade do protótipo Desktop.
Fonte: Os autores.

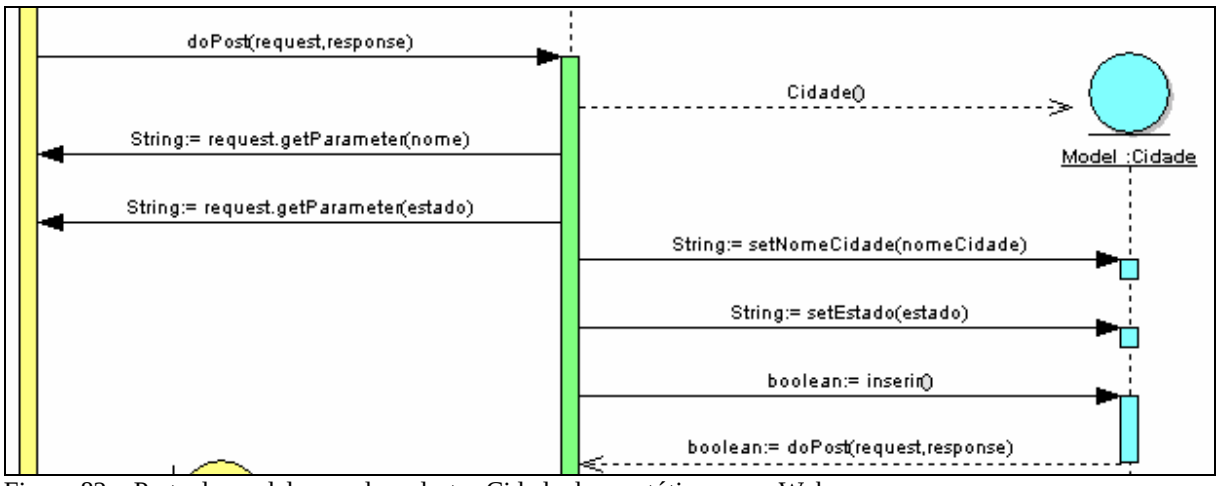


Figura 82 – Parte da modelagem de cadastro Cidade dos protótipos para Web.
Fonte: Os autores.

Quadro de comparação dos diagramas de seqüência			
	Desktop	**Web Tradicional**	**Web AJAX**
Cadastro de pessoa	~	<>	~
Cadastro de cidade	=	=	=
Alterar pessoa	~	<>	~
Alterar cidade	<>	~	~
Pesquisar pessoa	~	=	=
Pesquisar cidade	~	=	=
Excluir pessoa	~	=	=
Excluir cidade	~	=	=

Quadro 7 – Comparação dos digramas de seqüência
Fonte: Os autores.

Legenda:
= igual.
<> diferente.
~ semelhante.

.

Excluir Pessoa/Cidade

Nos protótipos para Web são iguais sendo que o protótipo Desktop é semelhante, pois necessita de outro controle para realizar a chamada do formulário para executar a pesquisa pelo objeto.

Conforme as figuras 53 excluir pessoa na página 106, do capítulo 3 e figura 114 excluir cidade na página 169 apresentadas no anexo B.

Pesquisa Pessoa/Cidade

Nos protótipos para Web possui um controle a menos que no protótipo Desktop. Pois no Desktop necessita do controle *ProcessaTabelaCidade* para montar o *FormuláriotabelaCidade*.

Conforme as figuras 116 pesquisa pessoa na página 169 e figura 115 pesquisar cidade na página 170, apresentadas no anexo B.

Alterar Cidade

Nos protótipos para Web é necessário iniciar no formulário pesquisa, com isso, enviará o parâmetro de busca para a controle *ProcessoPesquisaCidade*, que montará a lista e a enviará a um novo *FormulárioTabelaCidade*. A controle ProcessaPegaDadosAlterar monta um objeto cidade que enviará para o formulário *alterar_cidade.jsp.*

No protótipo Desktop a controle *PrecessaResultadoCidade* já recebe como parâmetro a lista de objetos cidade, tornando visível o *FormulárioResultadoCidade*. Com o evento acionado pelo usuário este passa o objeto cidade para o controle *ProcessoAlteraCidade*, apresentando todos os dados no *FormulárioAlterarCidade*. Confome a figura 83 que mostra parte da modelagem Desktop, onde o processo começa na controle *ProcessoResultadoCidade*.

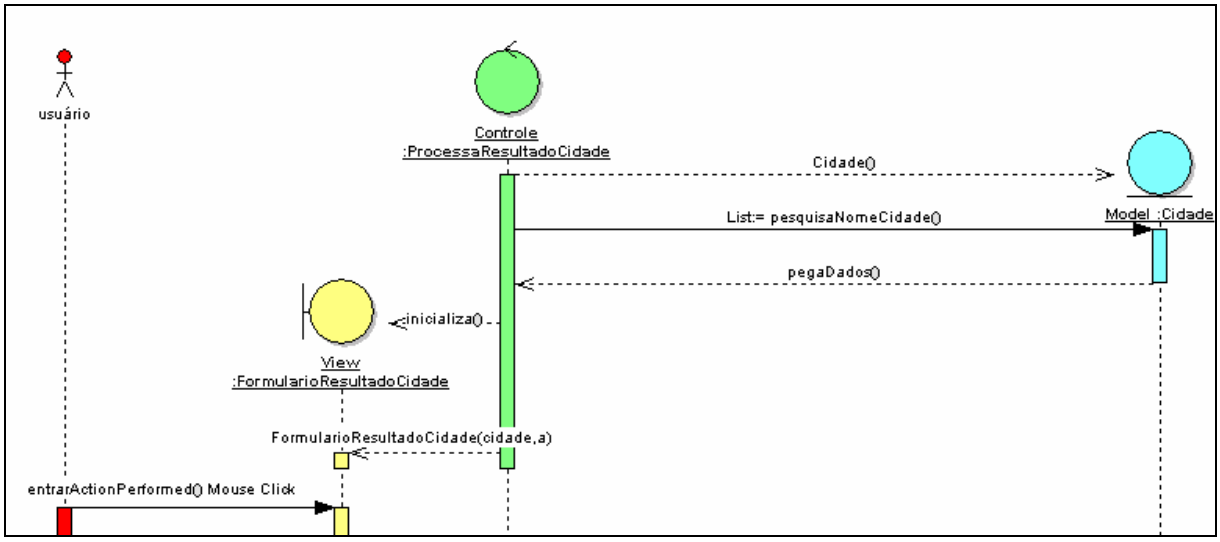


Figura 83 – Parte da modelagem de Alterar cidade protótipo Desktop
Fonte: Os autores.

A figura 84 demonstra o processo dos protótipos para Web onde o mesmo tem o início no formulário *pesquisa_cidade.jsp.*

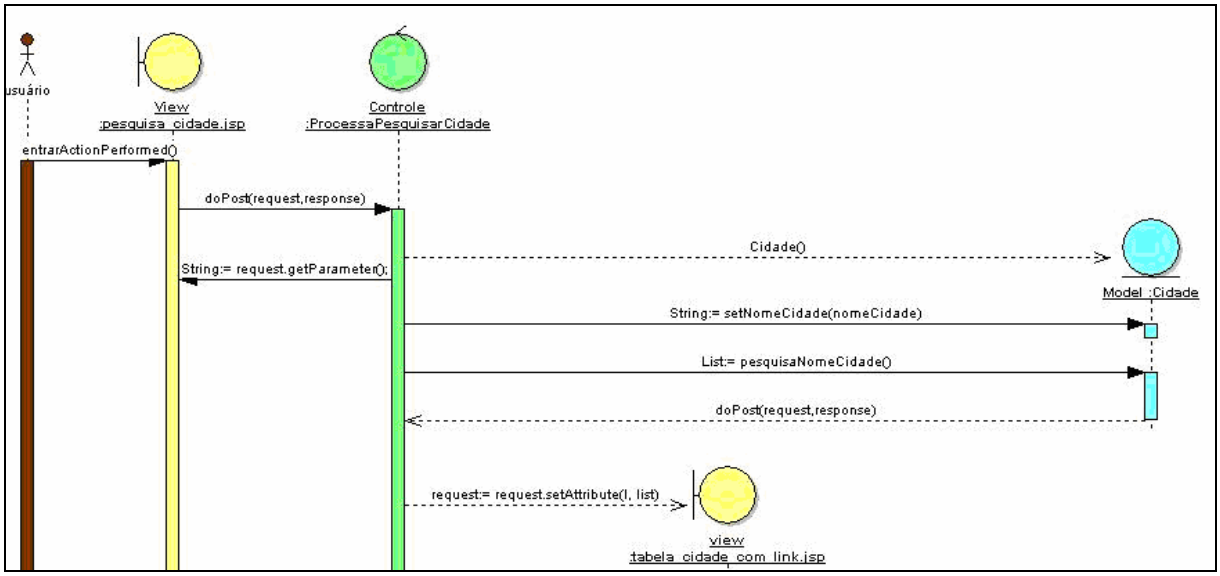


Figura 84 – Parte da modelagem alterar cidade para os protótipos Web

Fonte: Os autores.

Pesquisa Pessoa/Cidade

Nos protótipos para Web o processo de pesquisa e excluir são iguais.

No protótipo Desktop é semelhante, pois há uma controle a mais que realiza o processo de montagem da tabela para o *FormularioTabela*<Objeto>.

Excluir Pessoa/Cidade

Nos protótipos para Web a somente link no *formulario_excluir*<objeto> para o *formulario_ pesquisa*<objeto>.

No protótipo Desktop a mais uma controle para tornar visível o formulário *pesquisa*<objeto> ao usuário.

6.2 Análise do Desenvolvimento

No desenvolvimento dos três protótipos utilizamos o paradigma de OO e o padrão MVC de arquitetura de software.

Separamos as nossas regras de negócios na camada *Control* que está presente nos protótipos como pacote de controle, para o auxílio da camada *Model* desenvolvemos o pacote DAO para realizar a persistência dos nossos objetos criados no banco de dados.

Os pacotes de domínio e de persistência foram utilizados os mesmos nos três protótipos.

Na persistência utilizamos o *design pattern AbstractDAOFactory* que permite utilizar diversos bancos de dados relacionais utilizando a API JDBC. O *AbstractFactory* permite persistência em focos de dados distintos, para nossas implementações no presente trabalho foi implementado somente para uma base dados, Oracle 10g XE. O padrão permitirá a troca do banco de dados relacional dos protótipos facilmente com poucas alterações no código.

O pacote de domínio: Criar nossos objetos de domínio com construtores *default* sem argumentos, implementando a interface *Serializable* da linguagem Java, encapsulando-os com os métodos *getters* e *setters* de cada atributo.

A camada *Control* é composta pelos pacotes de controle dos protótipos.

O pacote de controle: Na camada *Control* do protótipo Desktop há a existência de associações entre eles. Pois uma classe controle chama outra classe (instancia outra classe) que realiza uma chamada de um método. A classe chamada pode ser um controle ou uma classe de formulário. Para os controles Desktop pegarem as informações nos campos da classe de formulário bastava instanciá-la dentro da controle e utilizar os métodos existentes na API *Swing* do Java.

Os pacotes de controle dos protótipos para Web são diferentes do pacote de controle do Desktop, mas são rigorosamente iguais entre os protótipos Web e AJAX. Não existem associações entre as classes de controle, sendo elas totalmente independentes uns dos outros. Os controles implementados pelos autores, para os protótipos Web e AJAX são *Servlets*.

Os *Servlets* recebem objetos *reques e response* que utilizamos para capturar e capturar dados dos formulários para as controles.

Ressaltamos que no protótipo AJAX há um controle a mais no pacote útil. Separamos essa classe controle das demais por se tratar de um controle que não é um *Servlet*.

A figura 85 mostra representação da classe Controle *PegaDadosCidade* do protótipo AJAX.

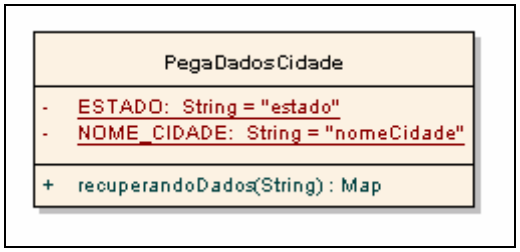


Figura 85 – Classe PegaDadosCidade do protótipo AJAX
Fonte: Os autores.

O elemento *dados_cidade.js* presente nas *views cadastrar_pessoa.jsp e alterar_pessoa.jsp* do protótipo AJAX, com o auxílio da classe *PegaDadosCidade* que retorna um *Map* para o JavaScript atualizando os dados do Objeto Cidade.

Na camada de *View* (formulários): O pacote do protótipo Desktop foi implementado separadamente dos protótipos para Web.

Nos protótipos para Web as camadas de visão são idênticas. Utilizamos à extensão JAVA de geração de conteúdo dinâmico JSP.

Para intermediar a camada *Control* com a *View*, nos protótipos para Web e AJAX utilizamos arquivos XML. O arquivo *web.xml* é presente nos dois protótipos para Web. O arquivo w*eb.xml* faz o mapeamento das classes controle em Java com a camada *view* presente (nossos JSP). Através dos métodos *get* e *post*, do protocolo HTTP os dados são enviados dados para o método *service* da classe *HttpServlet* que implementam os controles.

Trecho de do arquivo *web.xml*.que realiza o mapeamento das classes escritas em JAVA.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2eehttp://java.sun.com/xml/
ns/j2ee/web-app_2_4.xsd">
      <servlet>
            <servlet-name>ProcessaCadastroCidade</servlet-name>
            <servlet-class>
                  com.controle.ProcessaCadastroCidade
            </servlet-class>

            <!—continua  -->

      <servlet-mapping>
            <servlet-name>ProcessaCadastroCidade</servlet-name>
            <url-pattern>/processa_cadastro_cidade</url-pattern>
      </servlet-mapping>

      <!—continua -->
```

A diferença do arquivo *web.xml* do protótipos Web para o arquivo *web.xml* do protótipo AJAX é que o arquivo neste protótipo tem um mapeamento do *framework* do arquivo *dwr.jar*.

O protótipo AJAX tem um arquivo XML a mais, o *dwr.xml* é responsável por mapear as classes queiram acessar o AJAX.

6.3 Elementos Interativos

Com base nos desenvolvimentos dos protótipos, analisamos a perda de elementos interativos na migração do protótipo Desktop para Web tradicional (dinâmica). Estes elementos foram desconsiderados no desenvolvimento dos protótipos Web tradicional por limitação da plataforma que trabalha somente com requisições e respostas. Conforme visto nas figuras 11 - fluxo de atualização Web sem o AJAX e figura 12 - fluxo de atualização Web com o AJAX no capítulo 2 página 42.

Na figura 86 demonstra o resultado da funcionalidade interativa de auto-preenchimento dos campos de nome cidade e estado. Constituindo um elemento assíncrono e interativo com API *Swing*.

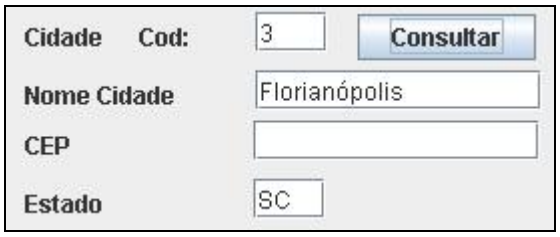


Figura 86 – Exemplo Desktop
Fonte: Os autores.

Na figura 87 demonstra o resultado da funcionalidade interativa de auto-preenchimento dos campos de nome cidade e estado. Constituindo um elemento assíncrono e interativo com o uso da técnica AJAX.

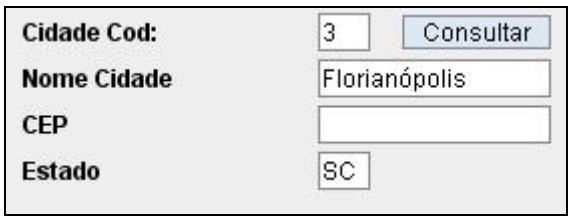


Figura 87 – Exemplo Web AJAX
Fonte: Os autores.

Elementos interativos como os apresentados nas figuras 86 e 87 evitam que o usuário final tenha que realizar um *submit* para preencher o formulário com os campos Nome Cidade e Estado referentes ao campo Cidade Cod:.

7 TRABALHOS FUTUROS

Ao encerrar este trabalho, foi constatada que poderia ser realizada a análise com mais uma outra plataforma a plataforma para dispositivos móveis. Sugerimos para trabalhos futuros possa ser desenvolvida a análise com um protótipo Wap, conforme a figura 88.

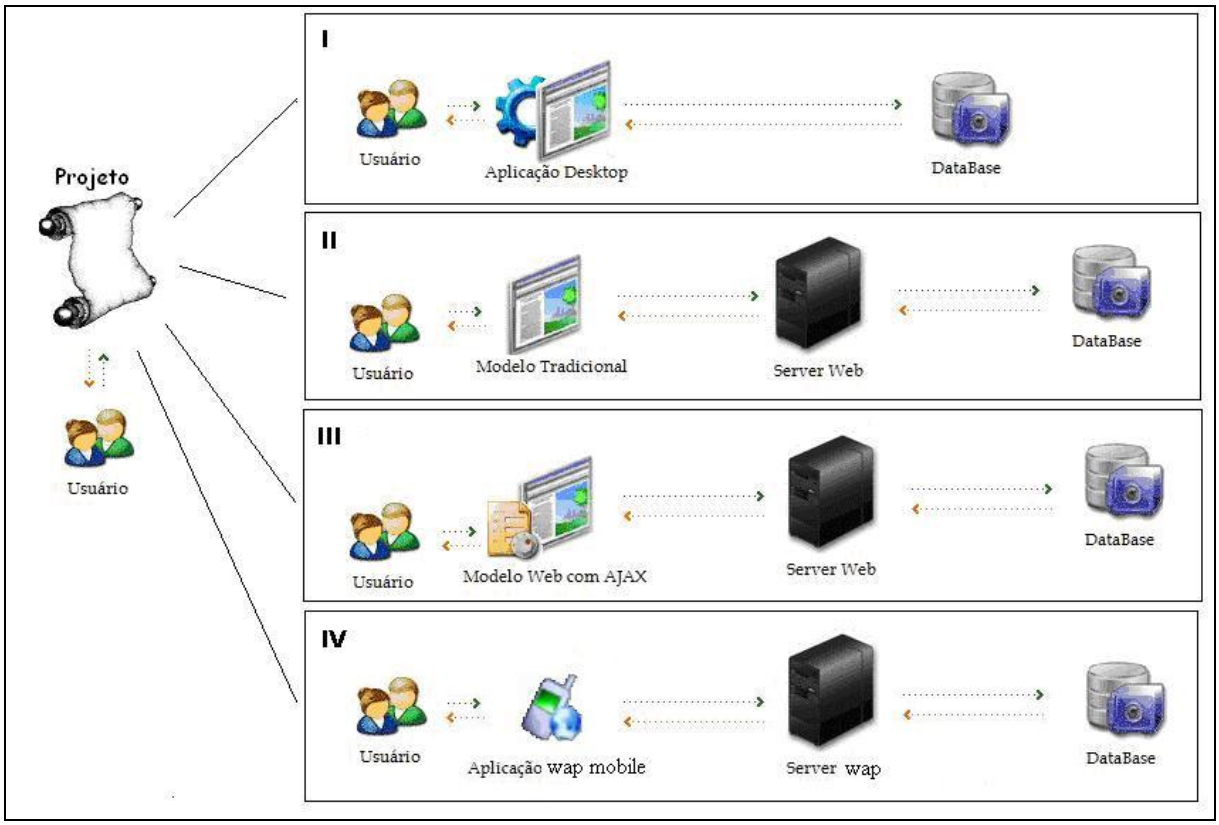


Figura 88 – Trabalhos futuros
Fonte: Os autores.

8 CONCLUSÃO

Através da explanação bibliográfica das tecnologias de apresentação e geração de conteúdos estudados, obteve-se material necessário para a avaliação e comparação dos processos dos três protótipos. A comparação foi feita por meio de estudo do desenvolvimento de requisitos funcionais, não funcionais, diagrama de robustez, diagrama de seqüência e os desenvolvimentos.

Com a concretização deste projeto, têm se a percepção do alcance do objetivo de utilizar modelos compartilhados para diferentes plataformas. No presente estudo, analisamos três modelagens utilizando duas plataformas de desenvolvimento distintas; Desktop com a *API Swing* do JAVA e a extensão JAVA para Web. A análise foi feita com base dos requisitos funcionais, requisitos não funcionais, casos de usos, diagramas de robustez, diagrama de seqüência, diagrama de classes e diagrama de pacotes.

Comprovou-se o reaproveitamento das classes na troca de plataforma de desenvolvimento, onde se conseguiu reaproveitar os objetos de domínio e a sua persistência no banco de dados.

Foram observadas semelhanças, diferenças e igualdades entre as modelagens. Constatou-se que há sempre semelhanças na seqüência dos processos na utilização de modelos compartilhados, mesmo utilizando diferentes métodos para cada plataforma.

O reaproveitamento foi total para os protótipos para Web. Para realizar o protótipo AJAX foram aproveitadas todas as camadas do protótipo Web tradicional. Para que o protótipo AJAX tivesse as mesmas funcionalidades do protótipo Desktop, foram baixas as alterações na modelagem em relação ao protótipo Web tradicional.

Com base nas modelagens e desenvolvimentos dos protótipos, demonstramos a perda de elementos interativos na camada *View* sem o uso da técnica AJAX.

Demonstramos uma vantagem para o usuário final que foi possível com o uso da técnica AJAX evitando que o mesmo tenha que fazer uma consulta por código de cidade para preencher o campo Cidade Cód: no formulário.

Realizamos o estudo dos componentes funcionais do AJAX demonstrados no desenvolvimento e no capítulo 2 (revisão bibliográfica) desta.

Demonstramos que é possível a migração de um protótipo Desktop para a Web mantendo a mesma interatividade e funcionalidade presente no protótipo Desktop.

.

REFERÊNCIAS

ADOBE, **Rich Internet applications**. Disponível em: <http://www.adobe.com/resources/business/rich_internet_apps/>. Acesso 20 abr. 2007.

ASLESON, Ryan; SCHUTTA, Nathaniel. **Fundamentos do AJAX.** Rio de Janeiro: Alta Books, 2006.

BELLO, José Luiz de Paiva, **Metodologia Científica,** Ano 2004. Disponível em: <http://www.pedagogiaemfoco.pro.br/met01.htm>. Acesso em agosto de 2007.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. São Paulo: Campus, 2002.

BOEHM, B. W. **A spiral model of software development and enhancement**. ACM Software Engineering Notes, v. 11, n. 4, p. 14–24, August 1986.

BORBA, Fernando Emmanuel. **AJAX guia de programação**: introdução. São Paulo: Erica, 2006.

BUYENS, Jim. **Aprendendo MySql e PHP**. São Paulo: Pearson Education do Brasil, 2002.

CAMARGO, Fernando Meyer; STEIL, Rafael. AJAX: Desenvolvendo uma Web mais interativa. **Mundo Java**. A revista do desenvolvedor Java Curitiba: PR, v. 1, n. 14. 2005, p. 36 - 48, nov. 2005.

CASTRO. Maria Alice Soares de, Instituto de Ciências Matemáticas e de Computação, da Universidade de São Paulo em São Carlos, SP. **O que é World-Wide Web**. Disponível em: <http://www.icmc.usp.br/ensino/material/html/www.html>. Acesso em 30 mar. 2007.

CUNHA, Livar Correia de Oliveira Cavalcanti. **Timeline Web browser e arquitetura de software para aplicações Web 2.0**. 2006. 48 f. Dissertação de Graduação de Ciência da Computação – Universidade Federal de Pernambuco, Recife, 2006.

CHISHOLN, Wendy; VANDERHEIDEN, Gregg; JACOBS, Ian. **Web Content Accessibility Guidelines 1.0**: W3C recommendation 5-Maio-1999. Disponível em: <http://www.w3.org/TR/WCAG10/ >. Acesso 20 abr. 2007.

CRANE, Dave; PASCARELLO, Eric. **AJAX in Action**. United States of America: Manning, 2006.

CROCKFORD, D. **The application/json Media Type for JavaScript Object Notation (JSON).** Disponível em: <http://www.ietf.org/rfc/rfc4627.txt>. Acesso em 4 de jun. 2007.

DEHAAN, Jen. Flash MX 2004. **Guia autorizado macromedia**. São Paulo: Elsevier, 2004.

DOUG, Rosenberg; MATT Stephens. **Use Case Driven Object Modeling with UML: Theory and Practice**. United States of América: Apress, 2007. p.34.

FERREIRA, Mariana Baird. **Dicionário Novo Aurélio**. O dicionário da língua portuguesa. Rio de Janeiro: Nova Fronteira, c1999.

GLUZ, Marcelo. 8p.com.br beta: Web 2.0 com o gostinho brasileiro. **Webdesign**. Rio de Janeiro: RJ, v. 1, n. 36. p. 48 - 53, dez. 2006. Entrevista concedida a revista.

GUERRA, Eduardo. Mantendo a segunraça em uma aplicação AJAX. **Revista Mundo Java.** A revista do desenvolvedor Java. Curitiba, PR. v. 4, n. 22, p. 36 – 43, mar. 2007.

HUSTED, Ted; DUMOULIN, Cedric; FRANCISCUS, George; WINTERFELDT, David. **Struts em ação**.Rio de Janeiro: Ciência Moderna Ltda, 2004.

International Organization for Standardization, 9241-11: **Ergonomic requirementes for office work witch visual display terminals (VDTs) – Part 1**: Guidance on usability. Switzerland 1998.

JEVEAUX, Paulo César M. Laszlo. Desenvolvendo Aplicações com recursos poderosos na Web. **Revista mundo Java**. A revista do desenvolvedor Java Curitiba: PR, v. 1, n. 14. 2005, p. 24 – 35, nov. 2005.

KENT, Peter; KENT, John. **Guia oficial JavaScripts para Netscape**. Makron books Ltda. São Paulo SP c1997.

KIRK, Cherly; MOULTIS, Nataya Pittys. **XML black book**: solução e poder. Makron Books São Paulo SP c2000.

LEE, T. Berners, CONNOLLY, **Internet Engineering Task Force**: hypertext markup language - 2.0. Disponível em <http://www.ietf.org/rfc/rfc1866.txt> Acesso em 19 abr. 2007.

LIMEIRA, José Luiz Silveira. Utilização **de AJAX no desenvolvimento de sistemas Web** UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, Porto Alegre, novembro de 2006. disponível em: <http://www.limeira.eti.br/monografia_AJAX.pdf>. Acesso 25/5/2007.

MARTINEZ, Maria Laura Martinez, **Usabilidade no design gráfico de Web sites**. 1997. Universidade de São Paulo SP. Disponível em: <http://www.lsi.usp.br/~martinez/works/_artigos/martinez00a.pdf> Acesso em 20 abr. 2007.

MCCARTHY, Philip. **AJAX for Java Developers**: Build dynamic Java applications 2005. Disponível em: < http://win.videotutoriales.com/pagina/AJAX-Java.htm>. Acesso em 23 de jun de 2007.

MCCLANAHAN, **Craig R. The state of Web frameworks**: Sun Microsystems. Inc. <http://kr.sun.com/developers/PDFs/preso/Craig_JCO2006.pdf> Disponível em: Acesso 13 jun. 2007.

MCMILLAN, Sally J. **Exploring models of interactivity from multiple research traditions:** users, documents, and systems, University of Tennessee 2000. Disponível em: <http://Web.utk.edu/~sjmcmill/Research/interactivity2.doc>.
Acesso 17 abr. 2007.

MORO, César Fernando. DWR do básico ao avançado. **Revista Mundo Java.** A revista do desenvolvedor Java. Curitiba PR. v.4, n 22, p. 26 - 35, mar. 2007.

NEXTAPP. **ECHO2**. Disponível em: <http://www.nextapp.com/platform/echo2/echo/>. Acesso 25/5/2007.

OLIVEIRA, José Palazzo M. de; RIGO, Sandro José. **Aquisição automática de classes de usuários integrando mineração de uso da Web e ontologias**. In: II WORKSHOP EM ALGORITMOS E APLICAÇÕES DE MINERAÇÃO DE DADOS, WAAMD; SBBD/SBES, 2006, Florianópolis, Sociedade Brasileira de Computação, 2006. p. 65-72.

PAGE-JONES, Meilir. **Fundamentos do desenho orientado a objeto com UML**. 1 ed. São Paulo: Makron Books, 2001.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

PYNE, Sandra; TUCK, Allene. **Oxford Dictionary of Computing**. New York. Oxford University Press, c 1996.

RAY, Erik T. **Aprendendo XML**. Campus c2001, Rio de Janeiro RJ.

ROCHA, Helder da. **Curso J93I: J2EE Disign Batterns. Versão 1.0** Disponível em <http://www.argonavis.com.br>. Acesso em 7 de set 2007.

RODRIGUES, Douglas Jose Soares. AJAX com Google Web Toolkit. Escrevendo Aplicações Web Altamente Interativas em Java. **Revista Java Magazine** ed. N° 38 ano 5 jul, 2006. volume único.

ROYCE, W. W. **Managing the development of large software systems**. In: Proceedings of IEEE WESCON, 1970. p. 1 -9.

SENGER, Yara; VILLELA, Melissa. Uma Aplicação Java EE Completa: Parte 3: Aumentando a interatividade com AJAX. **Revista Java Magazine** ed. Nº46 ano 7 , 2007. volume único.

SHARMA, Vivek; SHARMA Raijik. **Desenvolvendo Sites de e-Commerce**: como criar um eficaz e lucrativo site de e-commerce passo a passo. ed. Markon books ltda São Paulo SP c2000.

SOURCEFORGE **AJAXLib**. disponível em: <http://sourceforge.net/projects/AJAXlib>. Acesso 25 de maio 2007.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo, ed.6, p. 83 – 86. 2003.

SUN MICROSYSTEMS. Code Samples and Apps. **Applets**. Disponível em: <http://Java.sun.com/applets/>. Acesso 20 abr. 2007a.

SUN MICROSYSTEMS. Desktop Java. **Java Runtime Environment (JRE).** Disponível em: <http://Java.sun.com/j2se/DesktopJava/jre/index.jsp>. Acesso 20 abr. 2007b.

SUN MICROSYSTEMS. Desktop Java. **Java Plug-in Technology**. Disponível em: <http://Java.sun.com/products/plugin/index.jsp>. Acesso 20 abr. 2007c.

SUN MICROSYSTEMS. J2EE. **JavaServer Pages Overview**. Disponível em: <http://Java.sun.com/products/jsp/over*View*.html>. Acesso 20 abr. 2007d.

SUN MICROSYSTEMS. J2EE. **JavaServer Pages Technology**. Disponível em: <http://Java.sun.com/products/jsp/>. Acesso 20 abr. 2007e.

STEPHEN, H. Kan. **Metrics and Models in Software Quality Engineering, Second Edition**. 2ed. Boston: Pearson Education, 2006.

WEISSINGER, A. Keyton. **ASP O**: guia essencial. ed.Campus Rio de Janeiro RJ c2000.

World Wide Web Consortium, **Cascading Style Sheets**. Disponível em: <http://www.w3.org/Style/CSS/>. Acesso 16 abr. de 2007a.

World Wide Web Consortium, **Document Object Model**. Disponível em: <http://www.w3.org/DOM/>. Acesso 16 abr. de 2007b.

ANEXO A – Protótipo Desktop

CSU02 - Manipulação de dados de cidade

Descritos implementado no caso de uso “CSU02” as três operações (cadastro, exclusão e alteração) para o item cidade.

Manipulação de dados de cidade - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de cidade. Será realizado o cadastro, exclusão e alteração de cidade.

Fluxo de Eventos - Cadastro

Fluxo Básico:

1. Usuário preenche nome da cidade;
2. Usuário seleciona o estado;
3. Usuário clica em no botão Cadastrar.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

Não aplica;

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 89 demonstra a interface do caso de uso de cadastro de cidade.

Figura 89 – Interface Desktop – tela de cadastro de cidade
Fonte: Os autores.

Manipulação de dados de cidade - Excluir

Nesse caso de uso o usuário realizar a manipulação (exclusão) de dados de cidade.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código da cidade;
2. Usuário clica em Excluir;
3. Sistema excluir cidade desejada.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisa;
2. Sistema habilita tela com as cidades cadastradas no banco de dados.

b) <Fluxo Alternativo 2>

1. Usuário clica em limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 90 demonstra a interface do caso de uso de exclusão de cidade.

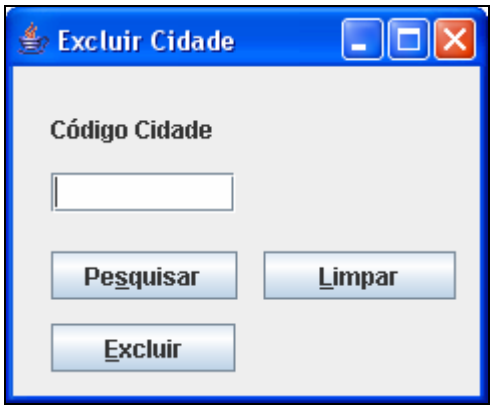


Figura 90 – Interface Desktop – tela de excluir cidade
Fonte: Os autores.

Manipulação de dados de cidade - Alterar

Nesse caso de uso o usuário realiza a alteração de Cidade.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a alteração dos dados desejados em Cidade;

2. Usuário clica em alterar;
3. Sistema altera os dados de cidade no banco de dados.

Fluxo Alternativo:

b) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

c) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 91 demonstra o resultado da pesquisa para a alteração do objeto cidade.

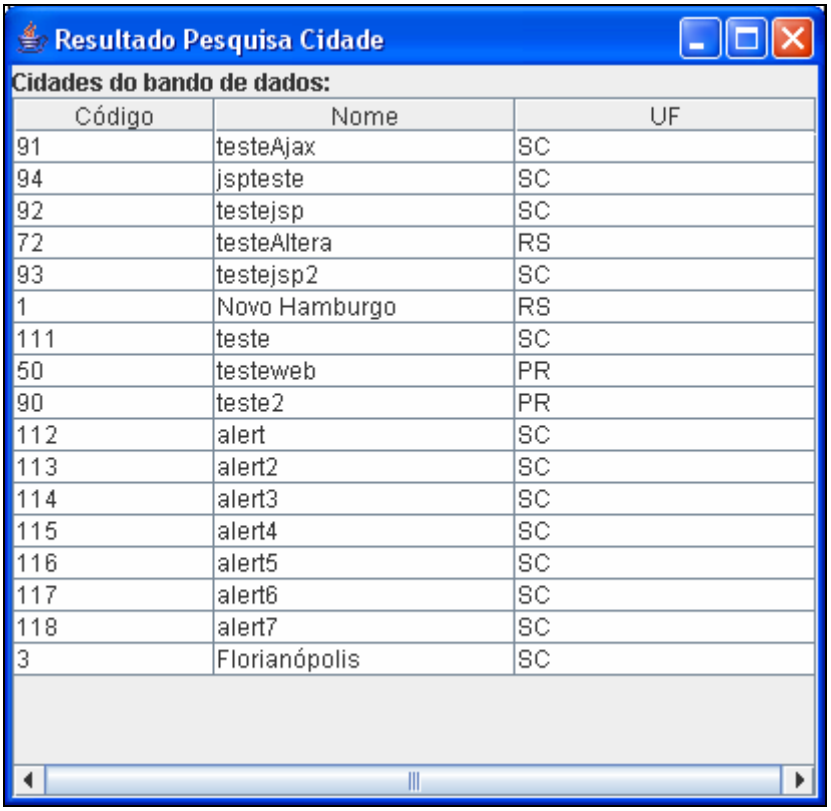


Figura 91 – Interface Desktop – tela de alterar cidade
Fonte: Os autores.

Na figura 92 demonstra a interface do caso de uso de alterar dados de cidade.

Figura 92 – Interface Desktop – tela de alterar cidade
Fonte: Os autores.

CSU03 - Pesquisa por cidade

Nesse caso de uso o usuário realiza a pesquisa por nome de cidade.

Fluxo de Eventos – Pesquisa

Fluxo Básico:

1. Usuário digita o nome da cidade;
2. Usuário clica em Pesquisar;
3. O sistema habilita tela com as informações das cidades cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 93 demonstra a interface do caso de uso de pesquisa de cidade.

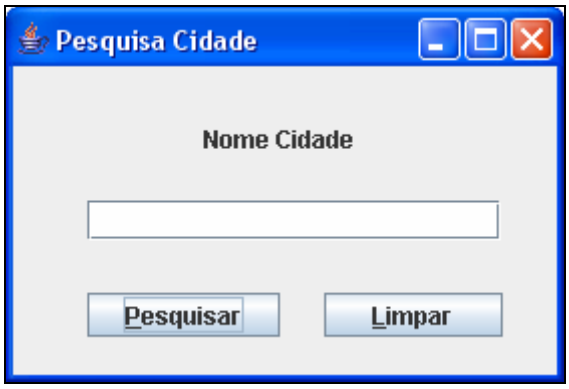


Figura 93 – Interface Desktop – tela de pesquisa cidade
Fonte: Os autores.

CSU04 - Pesquisa por pessoa

Nesse caso de uso o usuário realiza a pesquisa por nome de pessoa.

Fluxo de Eventos - Pesquisa

Fluxo Básico:

1. Usuário digita o nome da pessoa;
2. Usuário clica em Pesquisar;
3. O sistema habilita uma tabela com as informações das pessoas cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

A figura 94 demonstra a interface do caso de uso de pesquisa de pessoa.

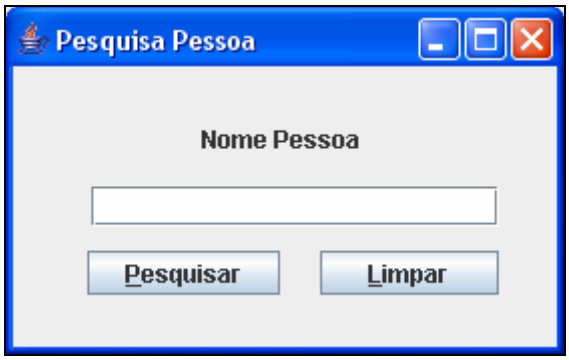


Figura 94 – Interface Desktop – tela de pesquisa pessoa
Fonte: Os autores.

Diagrama de Seqüência

Na figura 95 demonstra a diagrama de robustez do caso de uso cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Cadastrar

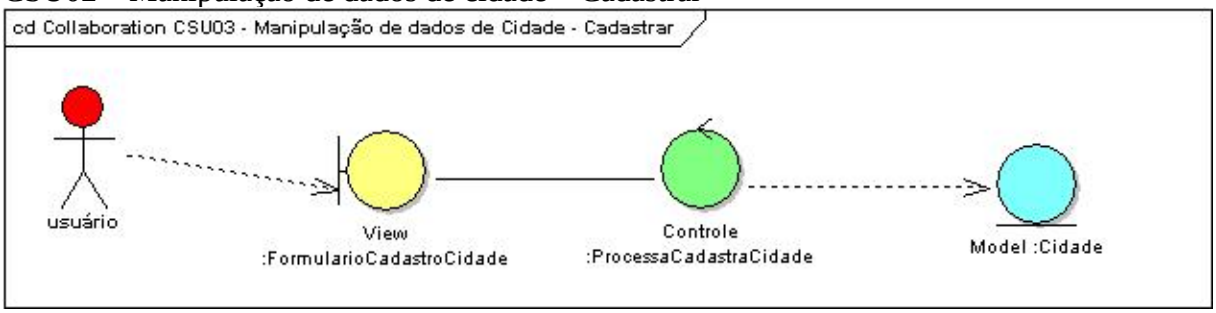


Figura 95 – Diagrama de robustez do CSU02 – Desktop - cadastra cidade
Fonte: Os autores.

Na figura 96 demonstra a diagrama de robustez do caso de uso alterar cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Alterar

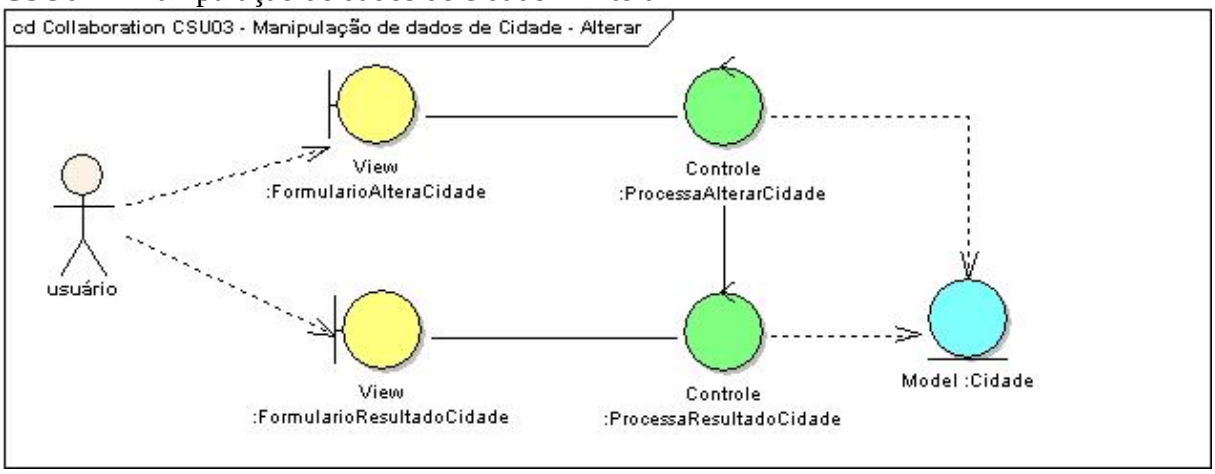


Figura 96 – Diagrama de robustez do CSU02 – Desktop - alterar cidade
Fonte: Os autores.

Na figura 97 demonstra a diagrama de robustez do caso de uso excluir cidade.

CSU02 – Manipulação de dados de cidade – Excluir

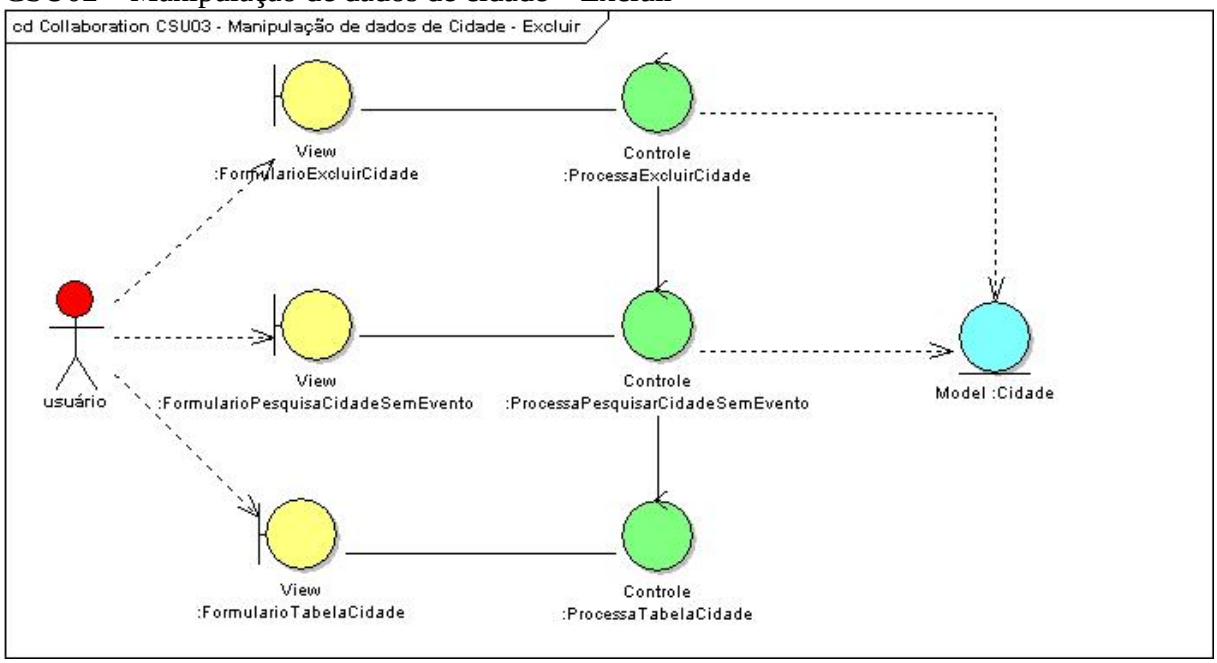


Figura 97 – Diagrama de robustez do CSU02 – Desktop - excluir cidade
Fonte: Os autores.

Na figura 98 demonstra a diagrama de robustez do caso de uso pesquisa de cidade.

CSU03 – Pesquisa por cidade

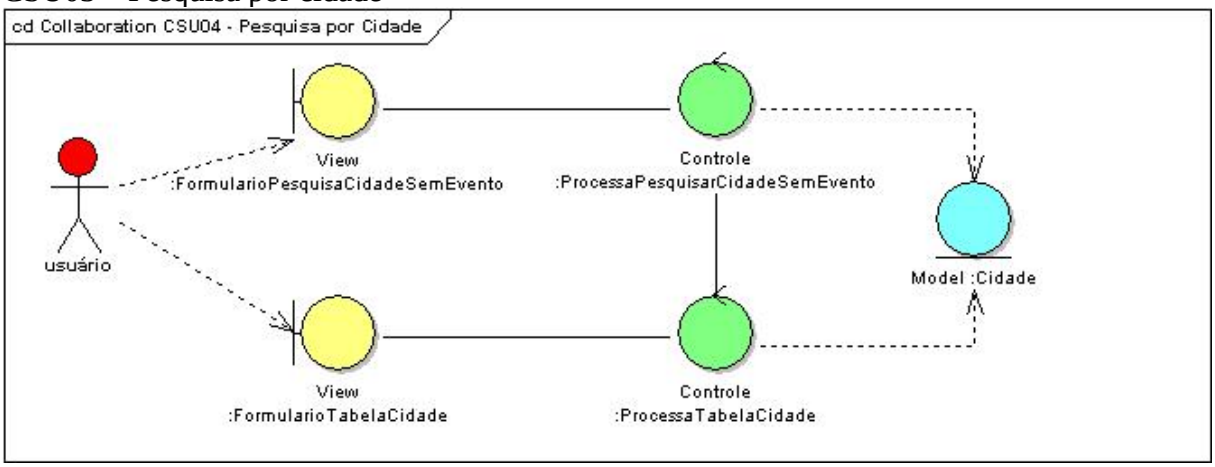


Figura 98 – Diagrama de robustez do CSU03 – Desktop - pesquisa por cidade
Fonte: Os autores.

Na figura 99 demonstra a diagrama de robustez do caso de uso pesquisa de pessoa.

CSU04 – Pesquisa por pessoa

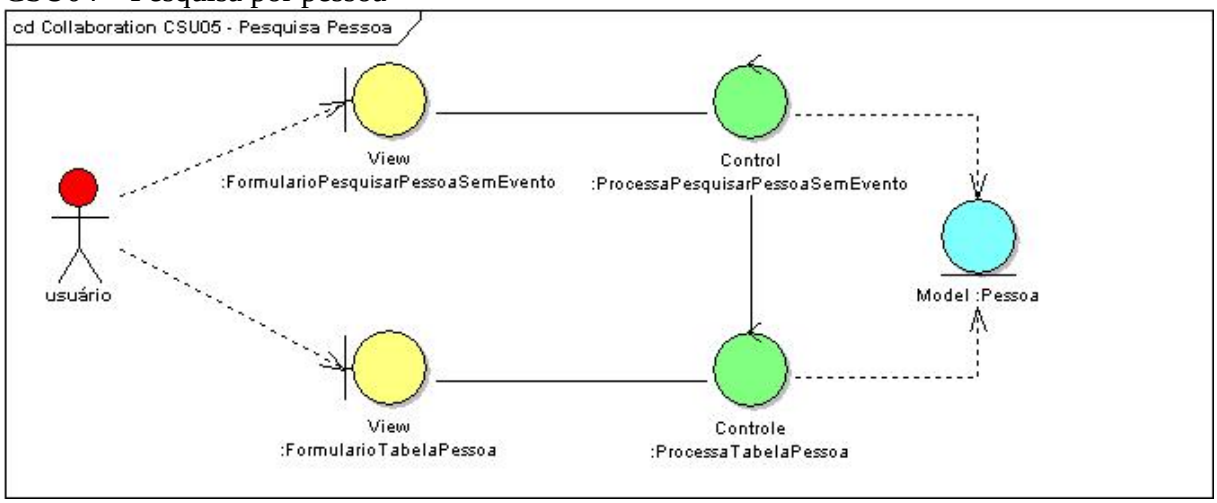


Figura 99 – Diagrama de robustez do CSU04 – Desktop - pesquisa por pessoa
Fonte: Os autores.

Na figura 100 demonstra a diagrama de seqüência do caso de uso cadastro de cidade.
CSU02 – Manipulação de dados de cidade – Cadastro

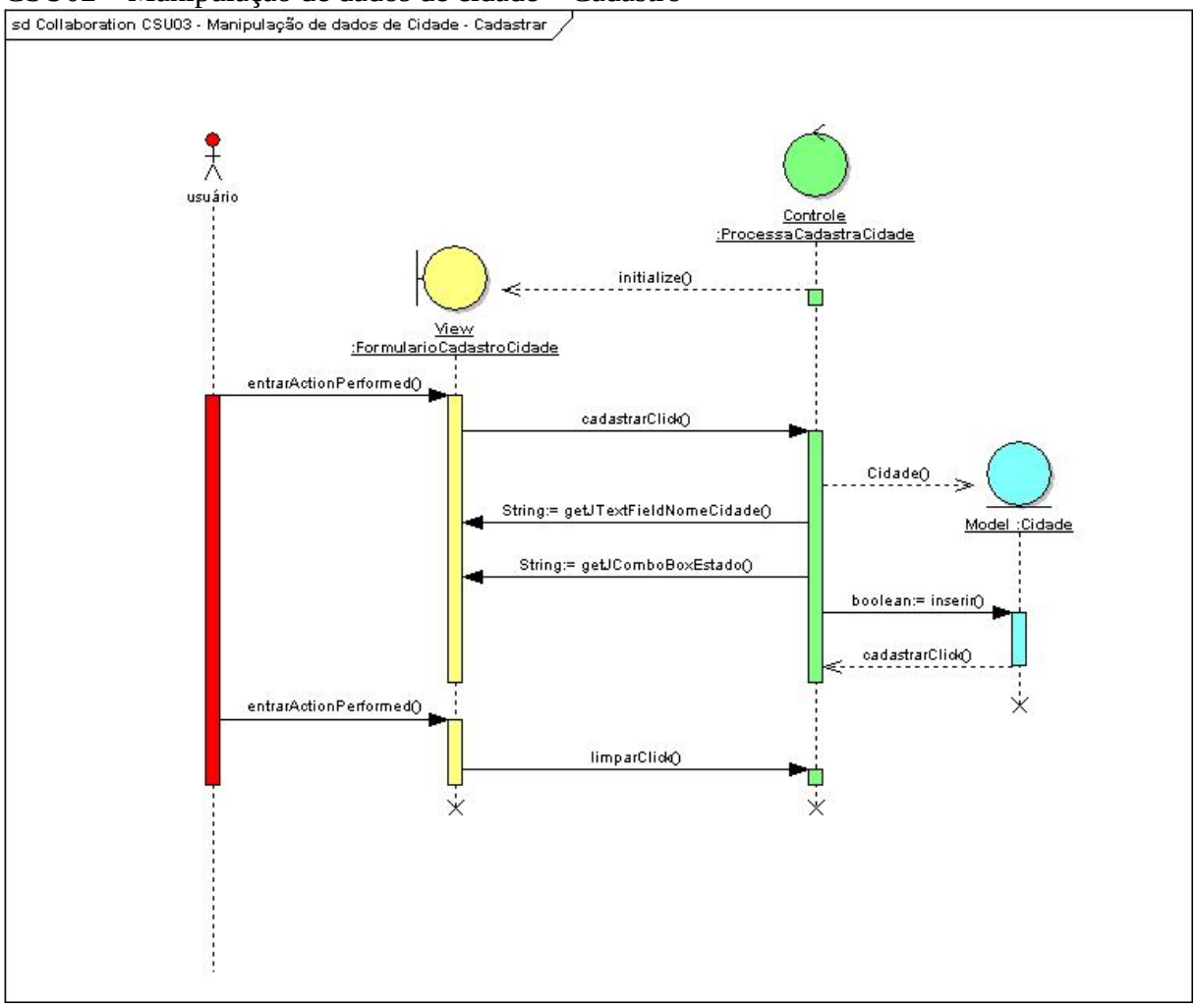


Figura 100 – Diagrama de seqüência do CSU02 – Desktop - cadastro de cidade
Fonte: Os autores.

Na figura 101 demonstra a diagrama de seqüência do caso de uso alterar de cidade.
CSU02 – Manipulação de dados de cidade – Alterar

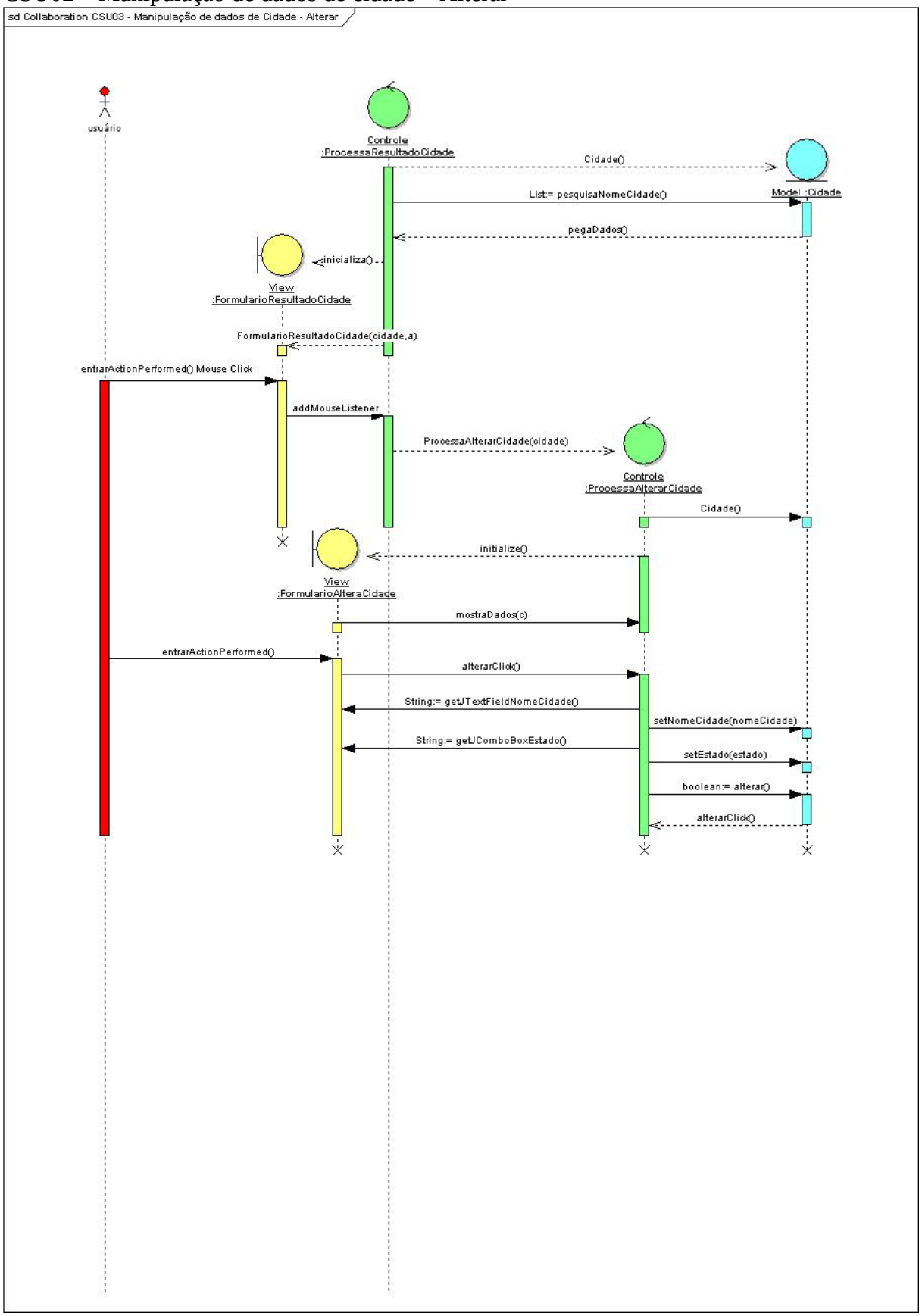


Figura 101 – Diagrama de seqüência do CSU02 – Desktop - alterar cidade
Fonte: Os autores.

Na figura 102 demonstra a diagrama de seqüência do caso de uso excluir cidade.

CSU02 – Manipulação de dados de cidade – Excluir

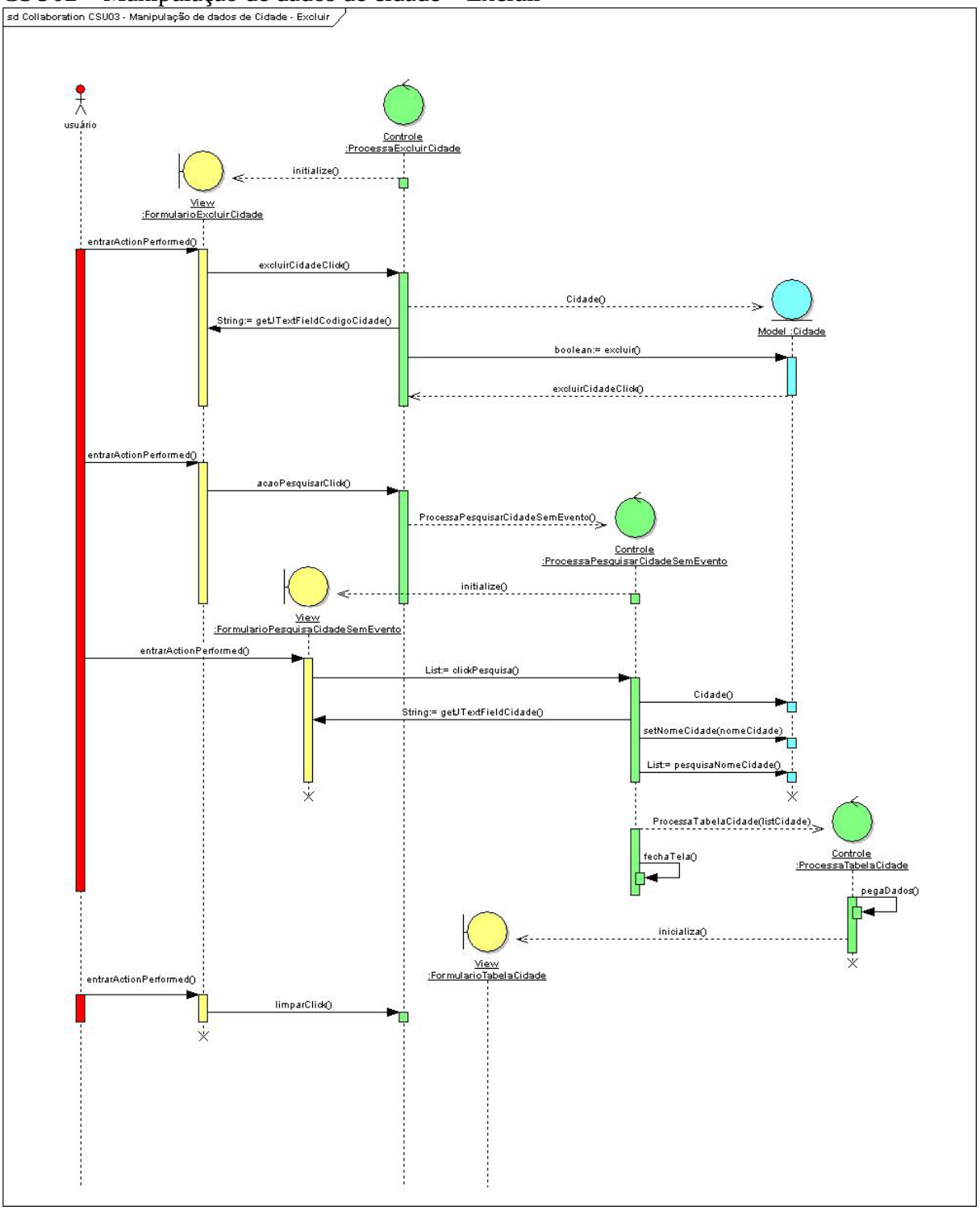


Figura 102 – Diagrama de seqüência do CSU02 – Desktop - excluir cidade

Fonte: Os autores.

Na figura 103 demonstra a diagrama de seqüência do caso de uso pesquisa de pessoa.
CSU04 – Pesquisa por pessoa

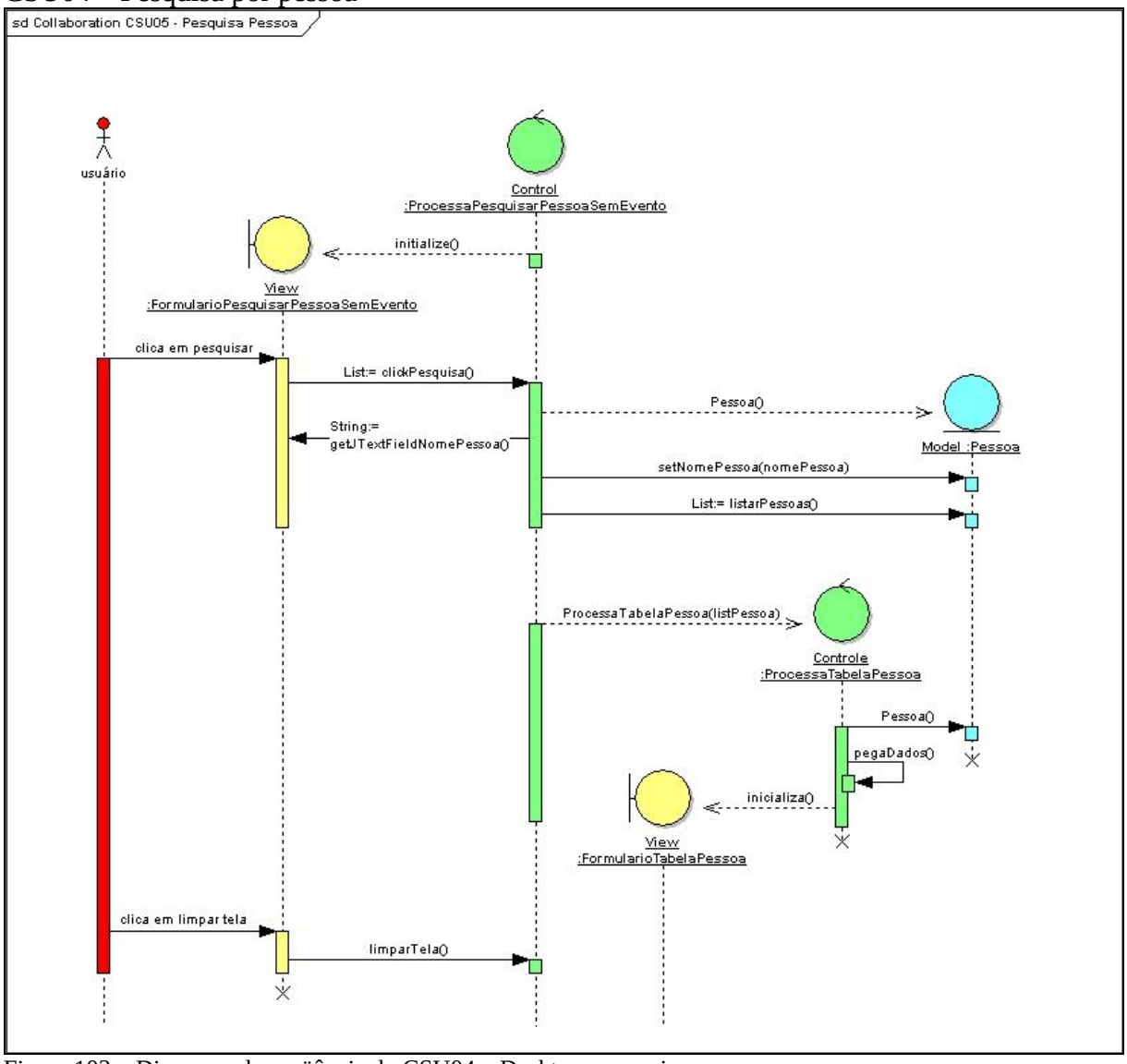


Figura 103 – Diagrama de seqüência do CSU04 – Desktop - pesquisa por pessoa
Fonte: Os autores.

Na figura 104 demonstra a diagrama de seqüência do caso de uso pesquisa de cidade.
CSU03 – Pesquisa por cidade

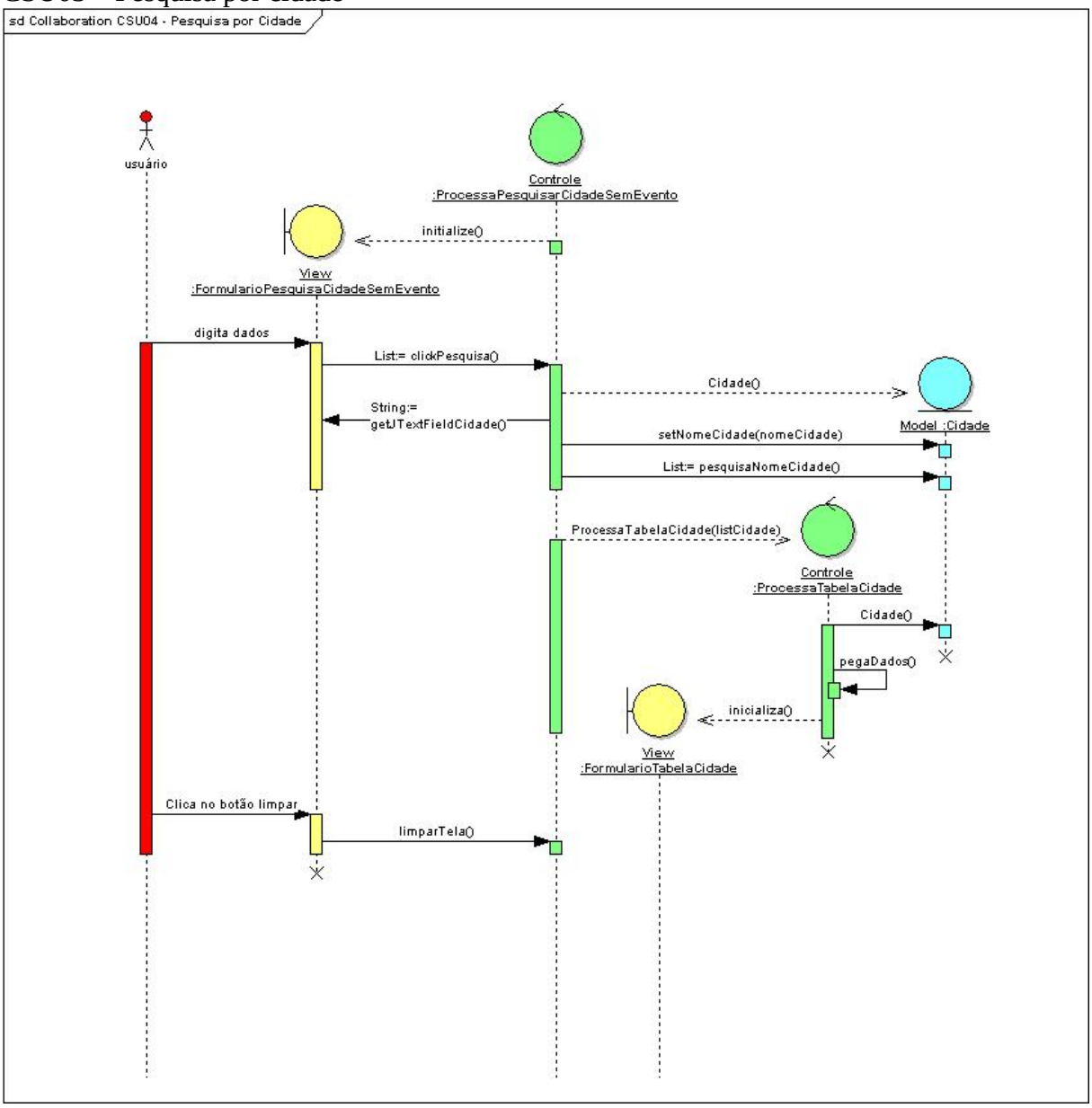


Figura 104 – Diagrama de seqüência do CSU05 – Desktop - pesquisa por cidade
Fonte: Os autores.

ANEXO B – Protótipo WEB

CSU02 - Manipulação de dados de cidade

Descritos implementado no caso de uso “CSU02” as três operações (cadastro, exclusão e alteração) para o item pessoa. Pós se treta da manipulação de dados.

Manipulação de dados de cidade - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de cidade. Será realizado o cadastro, exclusão e alteração de cidade.

Fluxo de Eventos - Cadastro

Fluxo Básico:

1. Usuário preenche nome da cidade;
2. Usuário seleciona o estado;
3. Usuário clica em no botão Cadastrar.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

Não aplica;

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 105 demonstra a interface do caso de uso de cadastro de cidade.

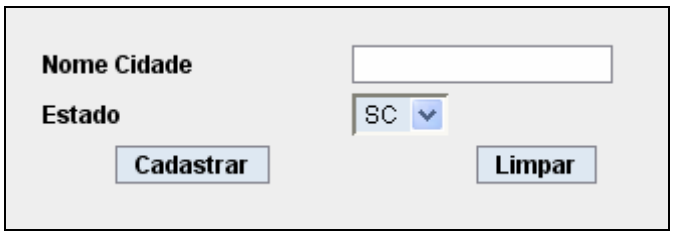


Figura 105 – Interface Web tradicional – tela de cadastro de cidade
Fonte: Os autores.

Manipulação de dados de cidade - Excluir

Nesse caso de uso o usuário realizar a manipulação (exclusão) de dados de cidade.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código da cidade;
2. Usuário clica em excluir;
3. Sistema excluir cidade desejada.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisar Cidade;
2. Sistema habilita tela com as cidades cadastradas no banco de dados.

b) <Fluxo Alternativo 2>

1. Usuário clica em Limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha o browser direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 106 demonstra a interface do caso de uso de exclusão de cidade.

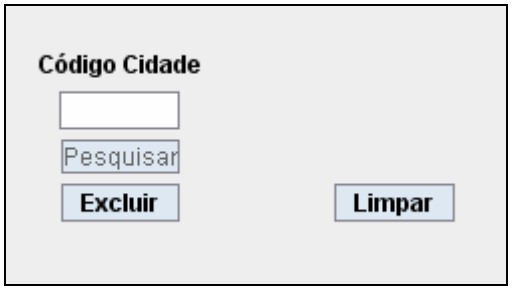


Figura 106 – Interface Web tradicional – tela de excluir cidade
Fonte: Os autores.

Manipulação de dados de cidade - Alterar

Nesse caso de uso o usuário realiza a alteração de Cidade.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a alteração dos dados desejados em Cidade;
2. Usuário clica em alterar;

3. Sistema altera os dados de cidade no banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha o browser direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

A figura 107 demonstra a interface do caso de uso de alteração de cidade.

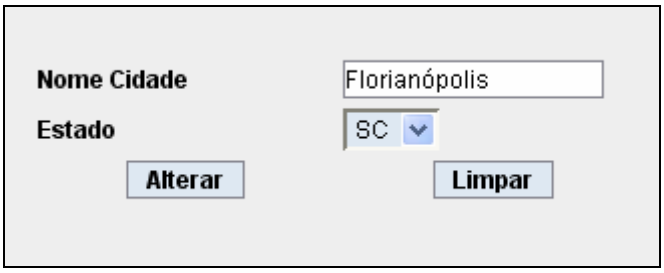


Figura 107 – Interface Web tradicional – tela de alterar de cidade
Fonte: Os autores.

CSU03 - Pesquisa por cidade

Nesse caso de uso o usuário realizar a pesquisa por nome de cidade.

Fluxo de Eventos – Pesquisa

Fluxo Básico:

1. Usuário digita o nome da cidade;
2. Usuário clica em Pesquisa;
3. O sistema habilita tela com as informações das cidades cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha o *browser* direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 108 demonstra a interface do caso de uso de pesquisa de cidade.

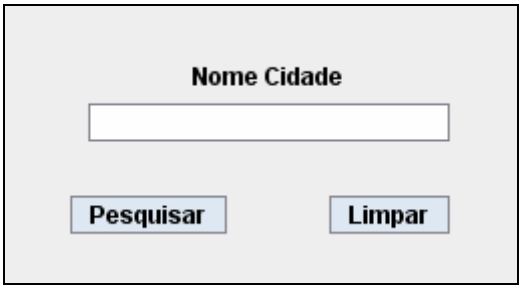


Figura 108 – Interface Web tradicional – tela de pesquisa cidade
Fonte: Os autores.

Resultado da pesquisa de cidade para a opção de alteração dos dados de cidade. Conforme a figura 109.

Cod Cidade	Nome Cidade	Estado	Editar
91	testeAjax	SC	Editar
94	jspteste	SC	Editar
92	testejsp	SC	Editar
72	testeAltera	RS	Editar
93	testejsp2	SC	Editar
1	Novo Hamburgo	RS	Editar
111	teste	SC	Editar
50	testeweb	PR	Editar
90	teste2	PR	Editar
112	alert	SC	Editar
113	alert2	SC	Editar
114	alert3	SC	Editar
115	alert4	SC	Editar
116	alert5	SC	Editar
117	alert6	SC	Editar
118	alert7	SC	Editar
3	Florianópolis	SC	Editar

Figura 109 – Interface Web tradicional – tela de pesquisa cidade
Fonte: Os autores.

CSU04 - Pesquisa por pessoa

Nesse caso de uso o usuário realizar a pesquisa por nome de pessoa.

Fluxo de Eventos - Pesquisa

Fluxo Básico:

1. Usuário digita o nome da pessoa;
2. Usuário clica em Pesquisa;
3. O sistema habilita uma tabela com as informações das pessoas cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha o browser direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

A figura 110 demonstra a interface do caso de uso de pesquisa de pessoa.

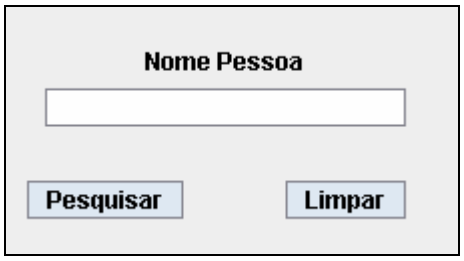


Figura 110 – Interface Web tradicional – tela de pesquisa pessoa
Fonte: Os autores.

Resultado da pesquisa por pessoa, com a opção de selecionar a opção de Editar. Confirme figura 111.

Cod Pessoa	Nome Pessoa	Sobrenome	Sexo	RUA	Nº	Complemento	Cod Cidade	CEP	Editar
3	Catia	Silveira	Femi	Guia Lopes	100	ap 42	1	9999999	Editar
49	teste	teste	Masc	jsp	1	teste	3	asdas	Editar
26	Suzana	Stoff	Femi	Av. Ivo silveira	111	bl 8	3	88090000	Editar
27	teste	emcapsulado	Masc	cidade	1	funcionna?	1	qqcoisa	Editar
50	Testa	cidade	Masc	invalida	1	123	12	asd	Editar
1	Catia	Silveira	Femi	Guia lopes	120	ap 42	1	92000	Editar
6	Catia	Silveira	Femi	Guia lopes	4638	42	1	888888	Editar
28	teste	asdkajsl	Masc	DFASLDKFÇLA	1	ASDAS	1	SDAS	Editar
30	jdbc	null	Masc	null	1	null	1	null	Editar
92	testejsp	ste	Masc	sds	1	sd	1	sd	Editar
52	testeweb	testeweb	Masc	nb	1	nb	70	nb	Editar
2	Artur	Todeschini Crestani	Masc	Santa Rita de Cassia	829	fundos	3	88090-35	Editar
31	teste	se vai	Masc	ou não	1	sda	1	da	Editar
48	teste	alterou	Masc	sim	12	dsfasd	3	123	Editar
72	Cristiano	Lagoia	Masc	Irmã bona vita	11	ap nº0	3	888888	Editar

Figura 111 – Interface Web tradicional – resultado da pesquisa
Fonte: Os autores.

Na figura 112 demonstra a diagrama de robustez do caso de uso cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Cadastrar

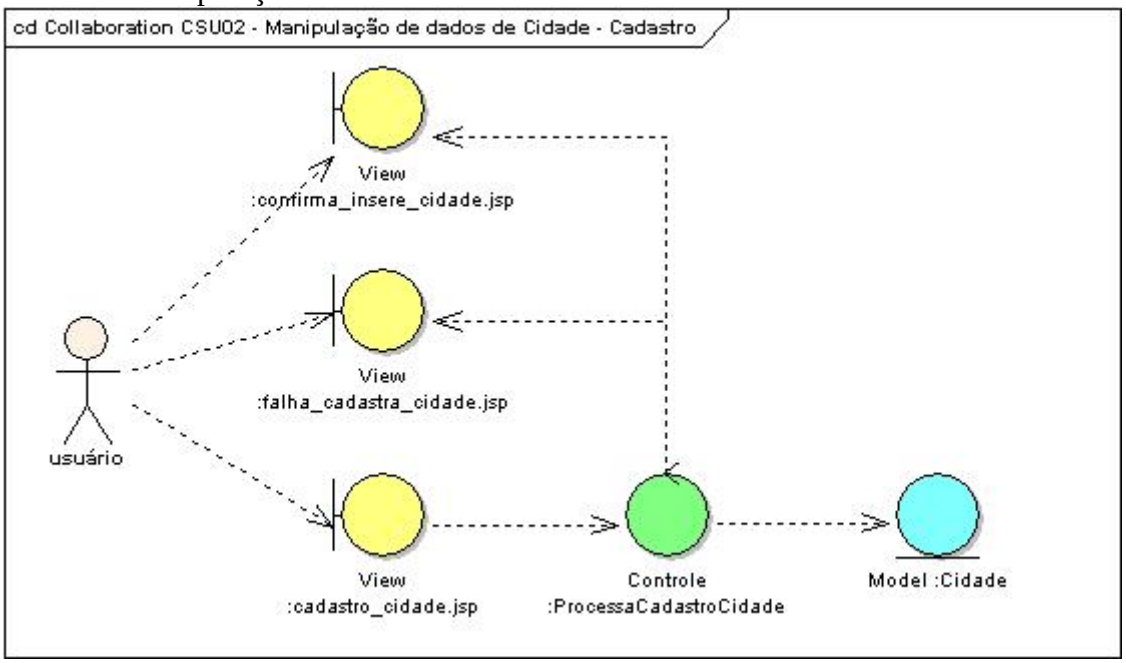


Figura 112 – Diagrama de robustez do CSU02 – Web tradicional - cadastrar cidade
Fonte: Os autores.

Na figura 113 demonstra a diagrama de robustez do caso de uso alterar cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Alterar

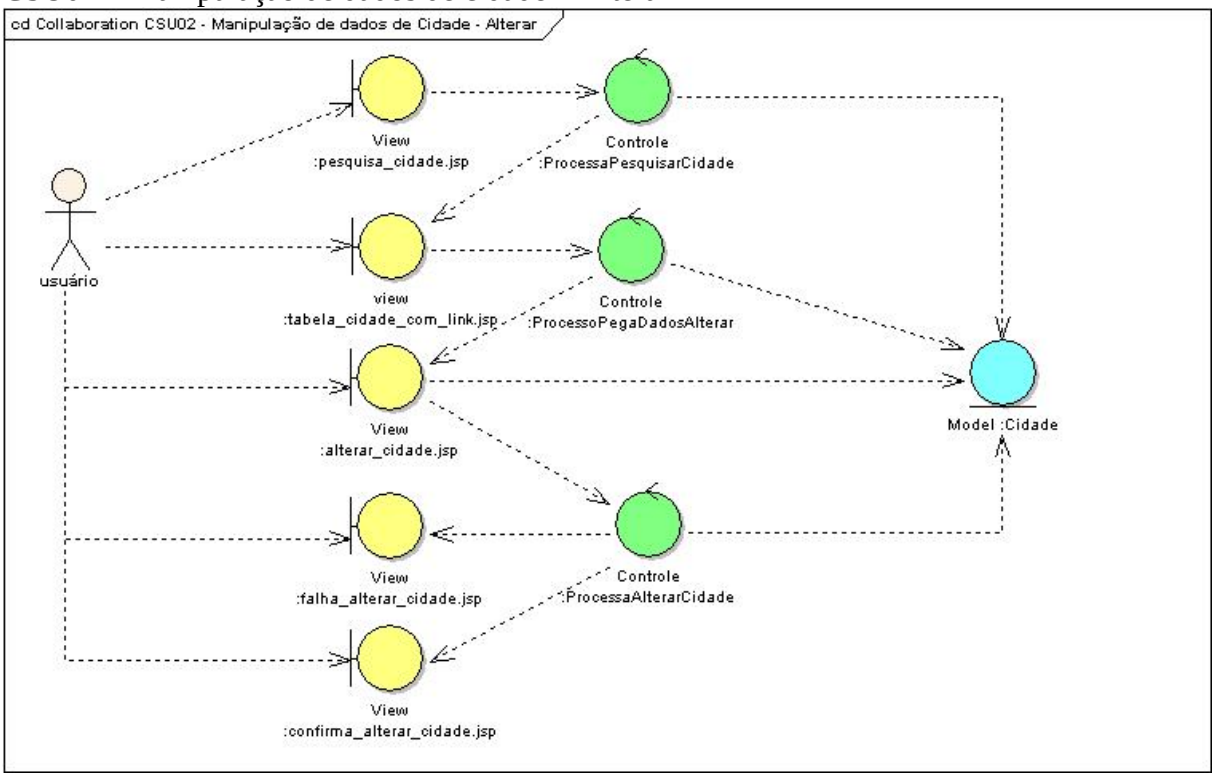


Figura 113 – Diagrama de robustez do CSU02 – Web tradicional - alterar cidade
Fonte: Os autores.

Na figura 114 demonstra a diagrama de robustez do caso de uso excluir cidade.

CSU02 – Manipulação de dados de cidade – Excluir

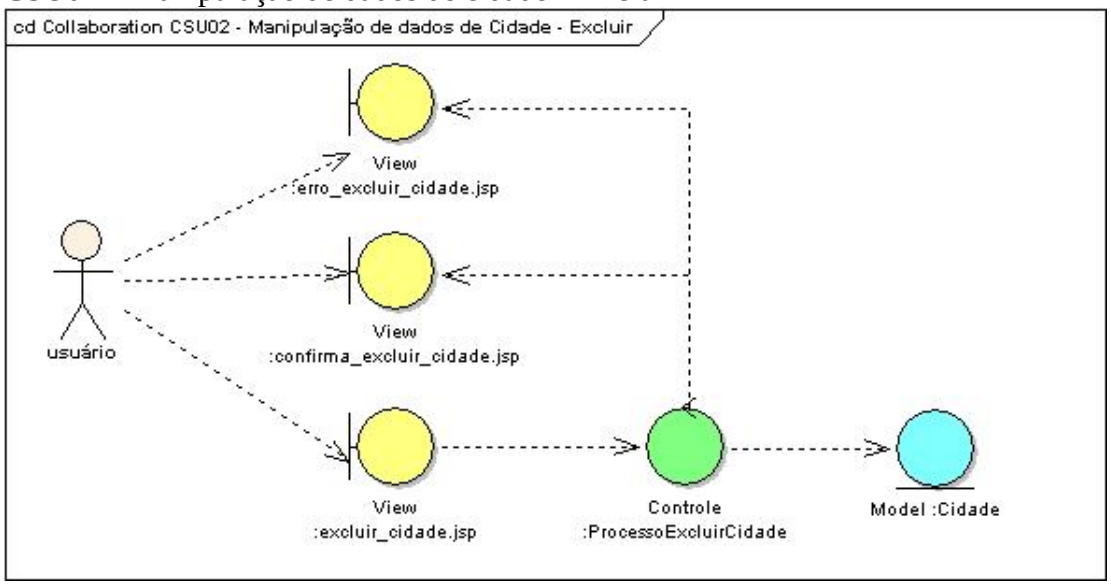


Figura 114 – Diagrama de robustez do CSU02 – Web tradicional - excluir cidade
Fonte: Os autores.

Na figura 115 demonstra a diagrama de robustez do caso de uso pesquisa de cidade.

CSU03 – Pesquisa por cidade

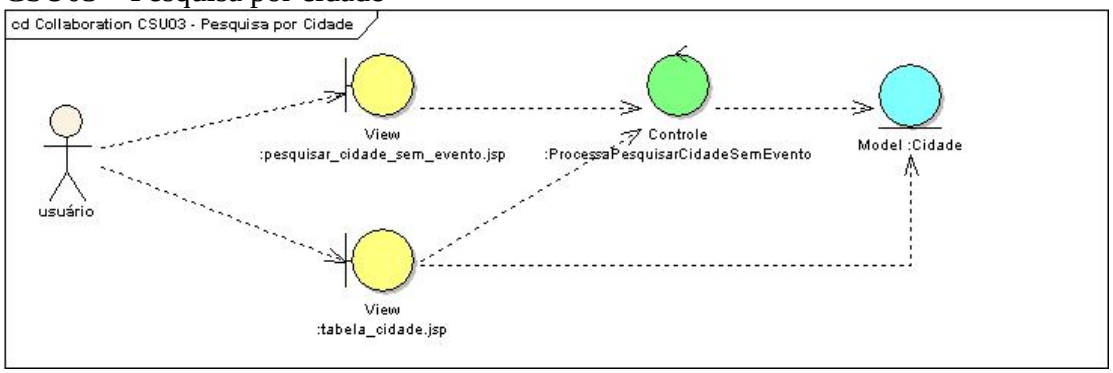


Figura 115 – Diagrama de robustez do CSU03 – Web tradicional - pesquisa por cidade
Fonte: Os autores.

Na figura 116 demonstra a diagrama de robustez do caso de uso pesquisa de pessoa.

CSU04 – Pesquisa por pessoa

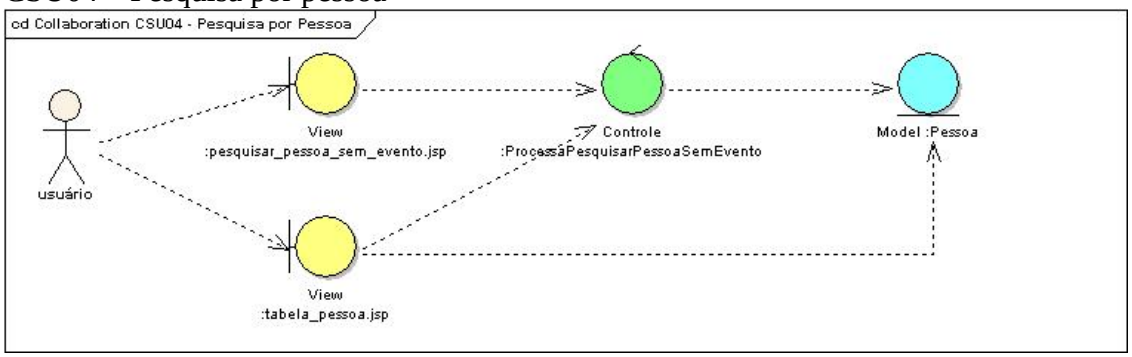


Figura 116 – Diagrama de robustez do CSU04 – Web tradicional - pesquisa por pessoa
Fonte: Os autores.

Na figura 117 demonstra a diagrama de seqüência do caso de uso cadastro de cidade.

CSU02 – Manipulação de dados de cidade – Cadastro

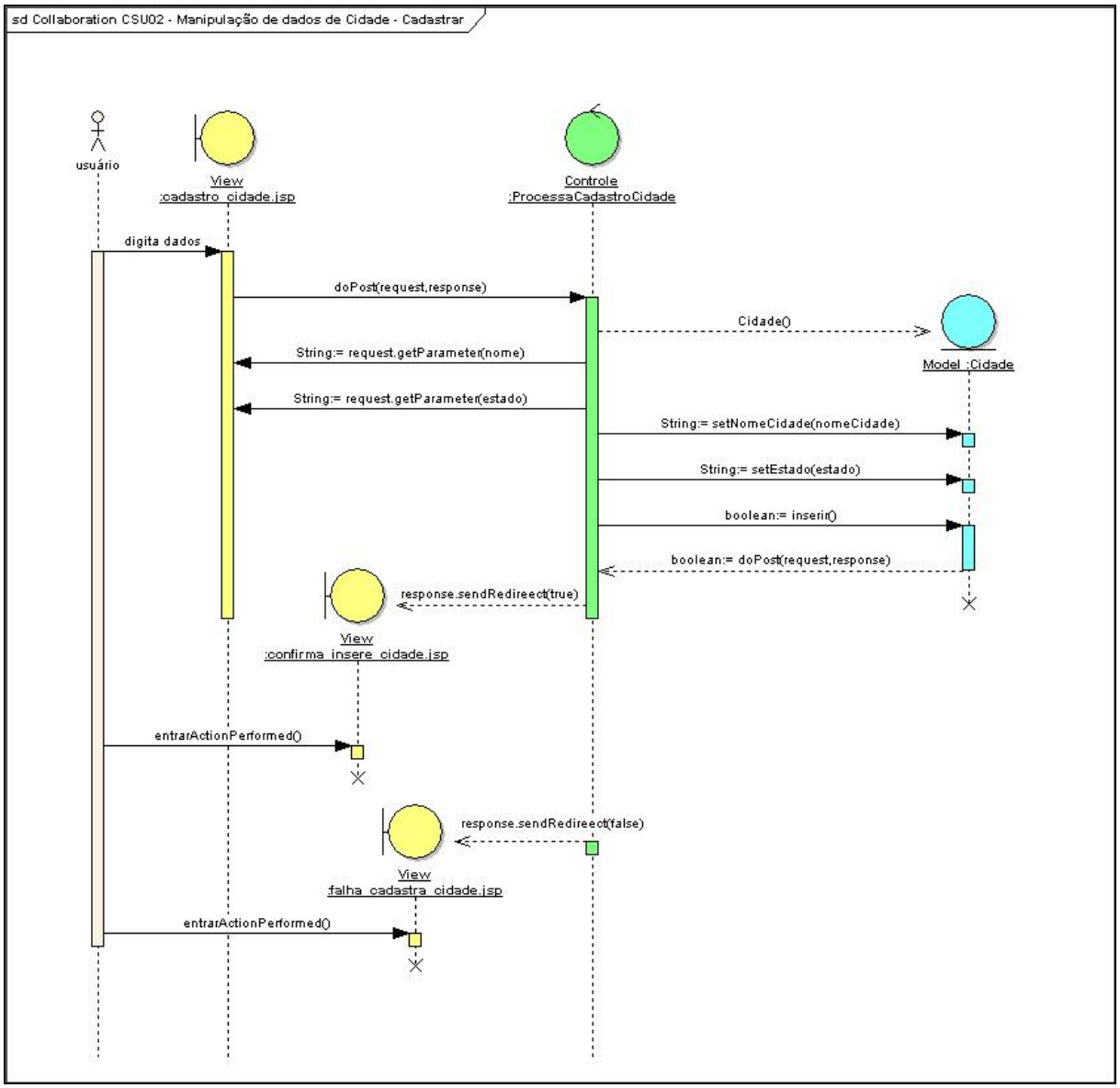


Figura 117 – Diagrama de seqüência do CSU02 – Web tradicional - cadastro de cidade
Fonte: Os autores.

Diagrama de Seqüência

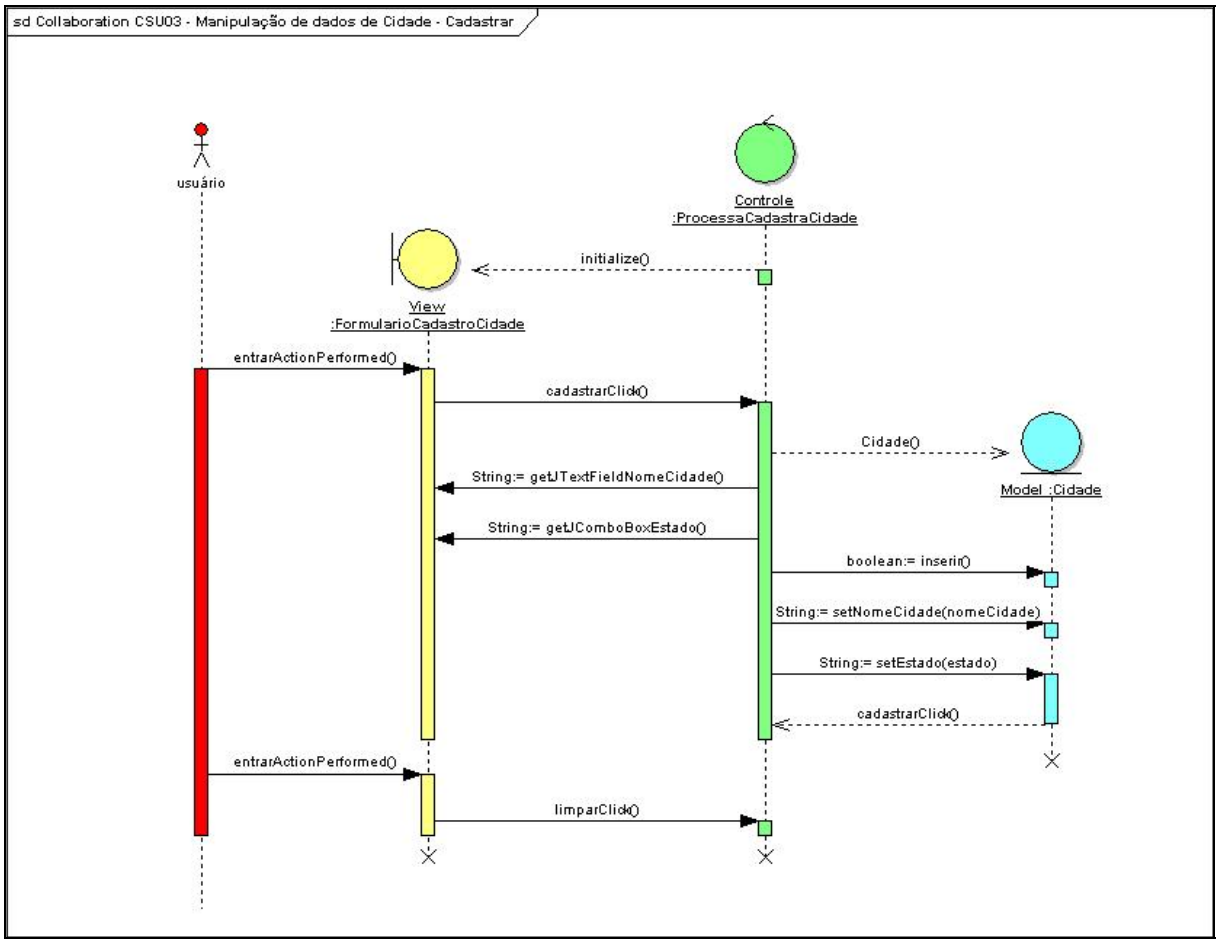


Figura 118 – Diagrama de seqüência do CSU02 – Web tradicional - cadastrar cidade
Fonte: Os autores.

Na figura 119 demonstra a diagrama de seqüência do caso de uso alterar de cidade.

CSU02 – Manipulação de dados de cidade – Alterar

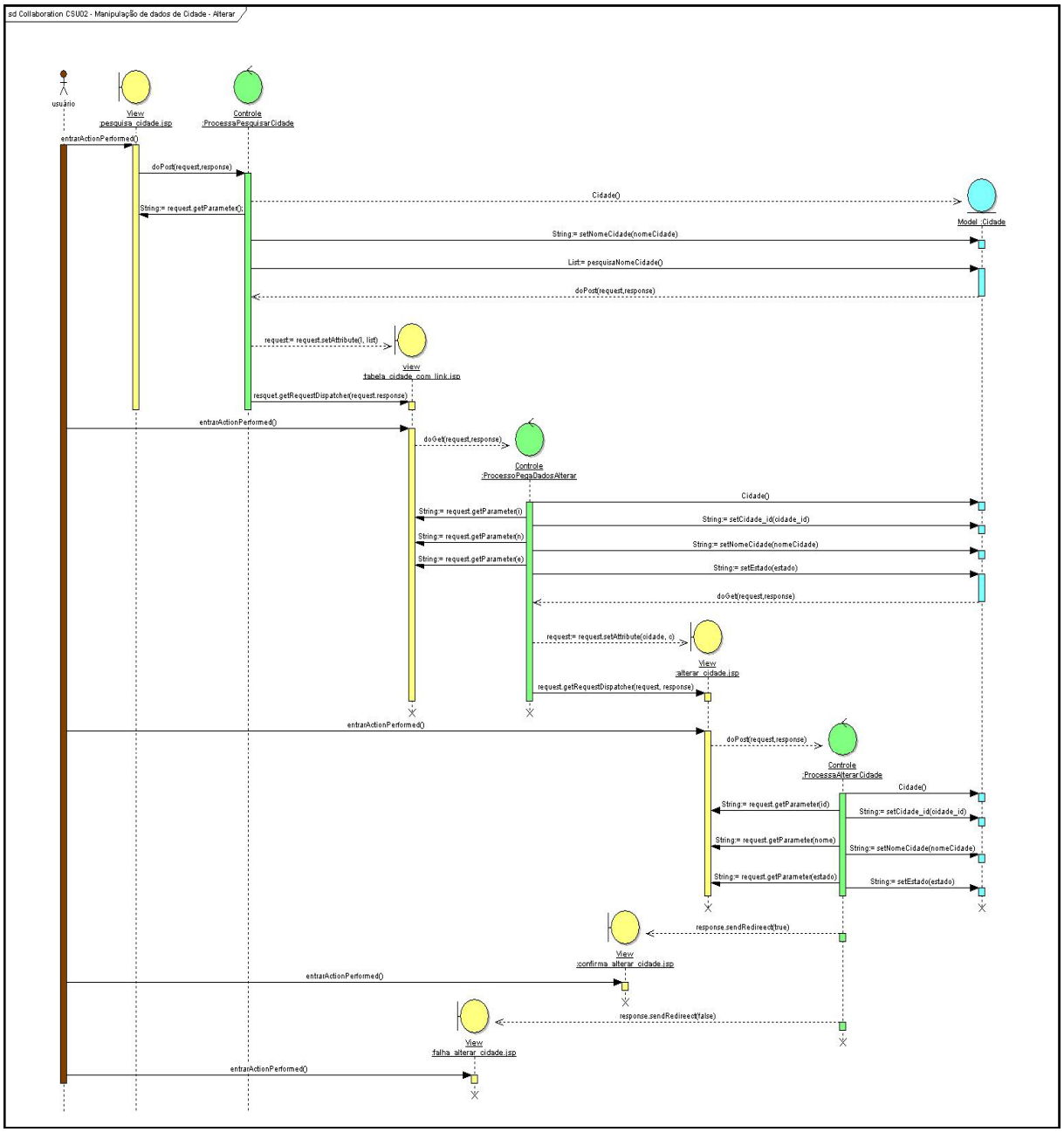


Figura 119 – Diagrama de seqüência do CSU02 – Web tradicional - alterar cidade
Fonte: Os autores.

Na figura 120 demonstra a diagrama de seqüência do caso de uso excluir cidade.
CSU02 – Manipulação de dados de cidade – Excluir

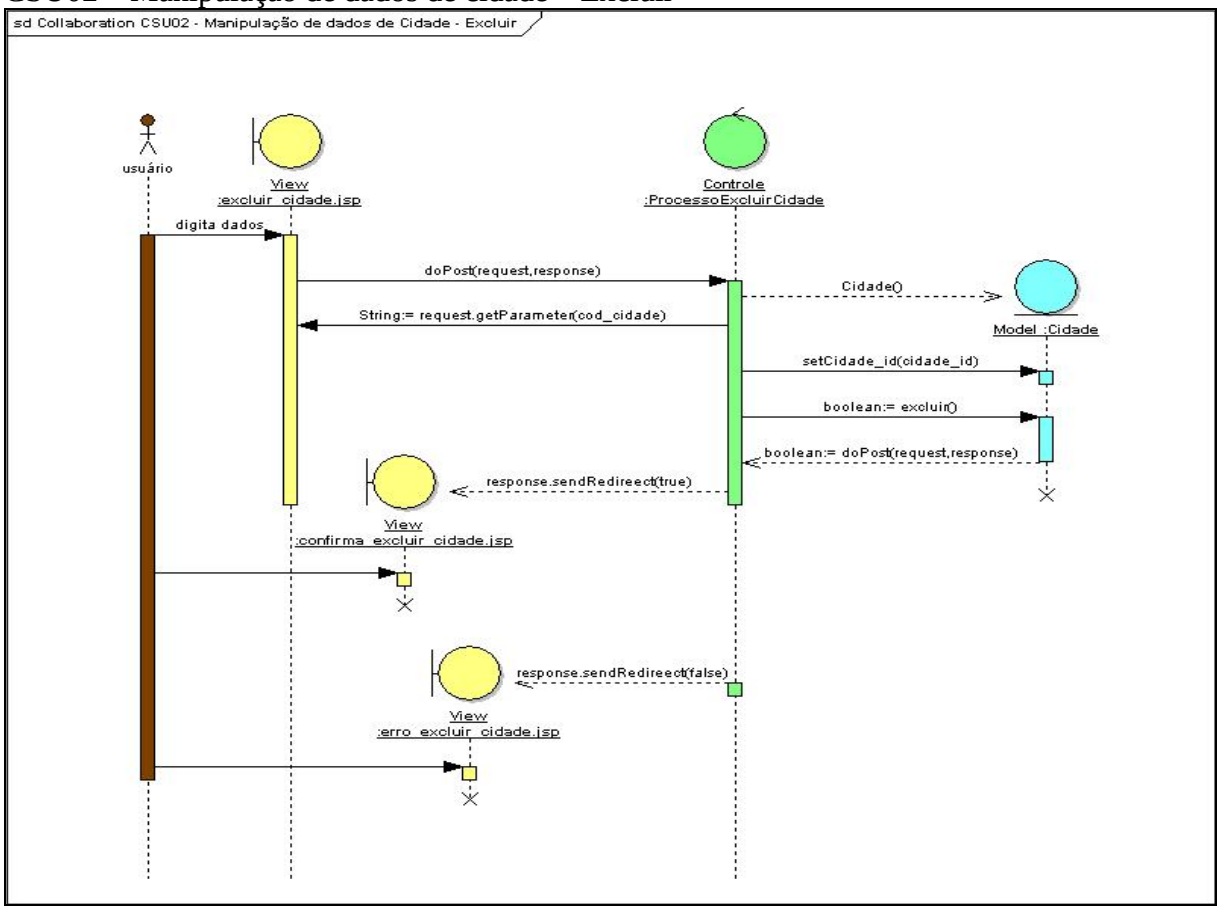


Figura 120 – Diagrama de seqüência do CSU02 – Web tradicional - excluir cidade
Fonte: Os autores.

Na figura 121 demonstra a diagrama de seqüência do caso de uso pesquisa de pessoa.
CSU05 – Pesquisa por pessoa

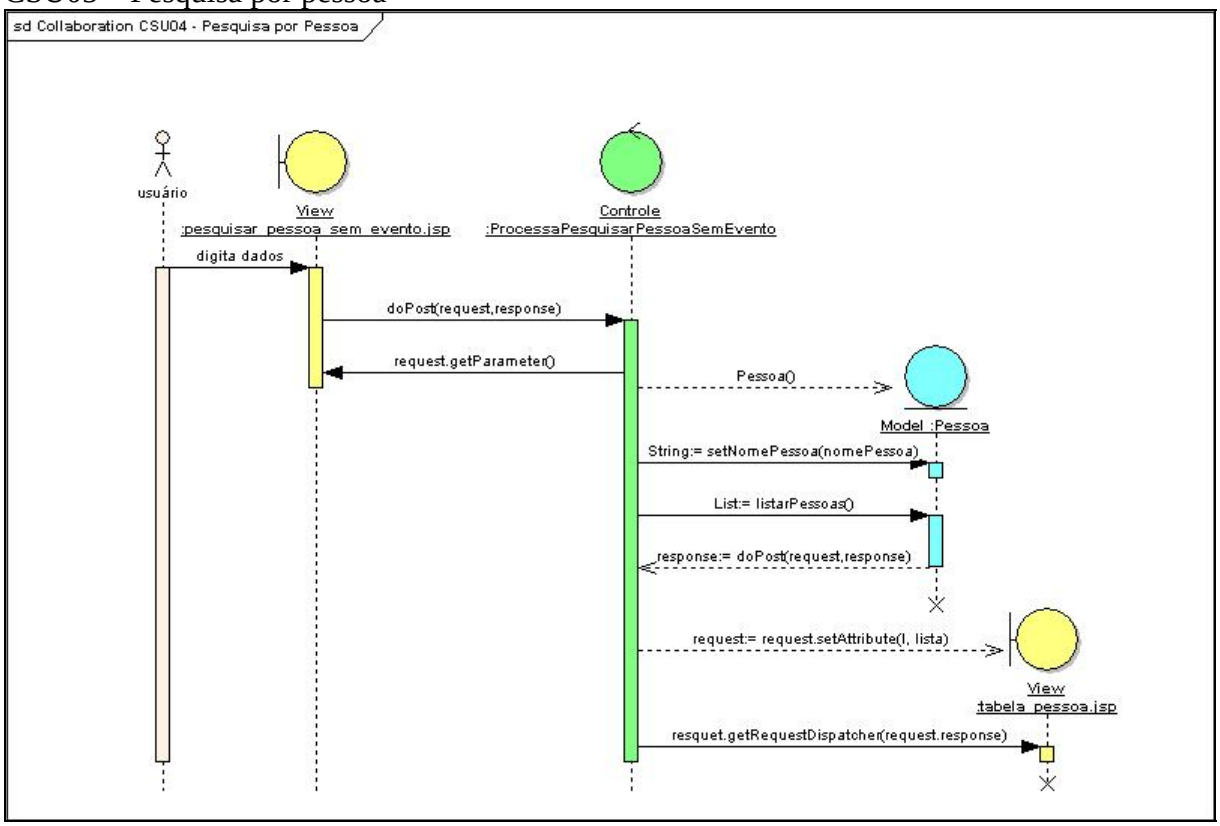


Figura 121 – Diagrama de seqüência do CSU05 – Web tradicional - pesquisa por pessoa
Fonte: Os autores.

ANEXO C – Protótipo WEB com AJAX

CSU02 - Manipulação de dados de cidade

Descritos implementado no caso de uso “CSU02” as três operações (cadastro, exclusão e alteração) para o item cidade.

Manipulação de dados de cidade - Cadastro

Nesse caso de uso o usuário realiza a manipulação de dados de cidade. Será realizado o cadastro, exclusão e alteração de cidade.

Fluxo de Eventos - Cadastro

Fluxo Básico:

1. Usuário preenche nome da cidade;
2. Usuário seleciona o estado;
3. Usuário clica em no botão Cadastrar.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

Não aplica;

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha à aplicação direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 122 demonstra a interface do caso de uso do cadastro de cidade.

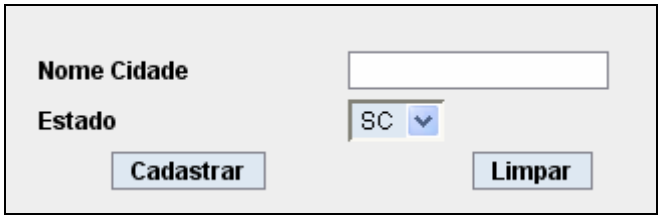


Figura 122 – Interface Web com AJAX – tela de cadastro de cidade
Fonte: Os autores.

Manipulação de dados de cidade - Excluir

Nesse caso de uso o usuário realizar a manipulação (exclusão) de dados de cidade.

Fluxo de Eventos - Excluir

Fluxo Básico:

1. Usuário preenche o código da cidade;
2. Usuário clica em Excluir;
3. Sistema excluir cidade desejada.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Pesquisar Cidade;
2. Sistema habilita tela com as cidades cadastradas no banco de dados.

a) <Fluxo Alternativo 2>

1. Usuário clica em Limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha o browser direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 123 demonstra a interface do caso de uso de exclusão de cidade.

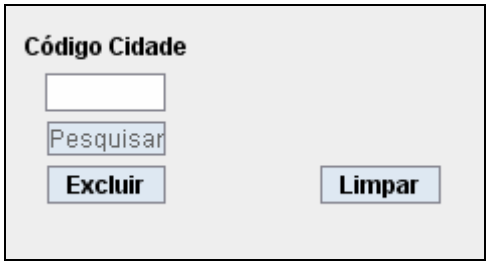


Figura 123 – Interface Web com AJAX – tela de excluir cidade
Fonte: Os autores.

Manipulação de dados de cidade - Alterar

Nesse caso de uso o usuário realiza a alteração de Cidade.

Fluxo de Eventos - Alterar

Fluxo Básico:

1. Usuário realiza a alteração dos dados desejados em Cidade;
2. Usuário clica em alterar;
3. Sistema altera os dados de cidade no banco de dados.

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar;
2. O sistema limpa os dados digitados na tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha o *browser* direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

A figura 124 demonstra a interface do caso de uso de alteração de cidade.

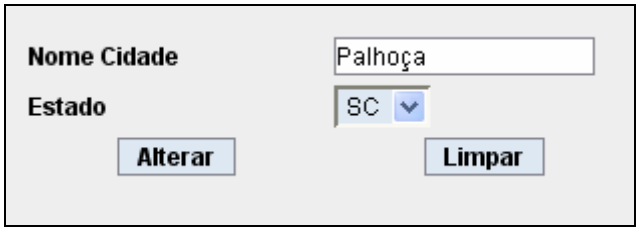


Figura 124 – Interface Web com AJAX – tela de excluir cidade
Fonte: Os autores.

Para realizar o processo de alteração de cidade o usuário deve selecionar qual a cidade será alterada, conforme a figura 125.

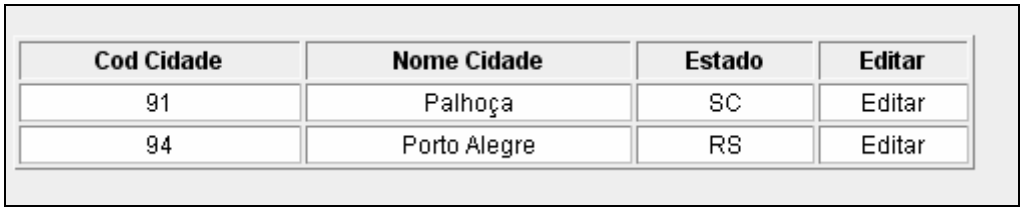


Figura 125 – Interface Web com AJAX – resultado da pesquisa pra alteração de cidade
Fonte: Os autores.

CSU03 - Pesquisa por cidade

Nesse caso de uso o usuário realizar a pesquisa por nome de cidade.

Fluxo de Eventos – Pesquisa

Fluxo Básico:

1. Usuário digita o nome da cidade;
2. Usuário clica em Pesquisa;
3. O sistema habilita tela com as informações das cidades cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo Exceção 1>

1. Usuário fecha o *browser* direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

Na figura 126 demonstra a interface do caso de uso de pesquisa de cidade.

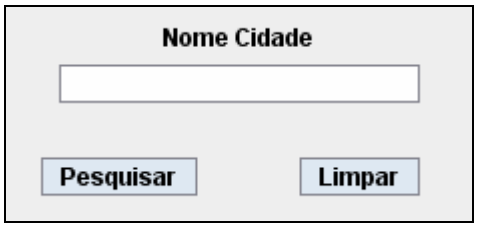


Figura 126 – Interface Web com AJAX – tela de pesquisa cidade
Fonte: Os autores.

CSU04 - Pesquisa por pessoa

Nesse caso de uso o usuário realizar a pesquisa por nome de pessoa.

Fluxo de Eventos - Pesquisa

Fluxo Básico:

1. Usuário digita o nome da pessoa;
2. Usuário clica em Pesquisa;
3. O sistema habilita uma tabela com as informações das pessoas cadastradas no banco de dados;

Fluxo Alternativo:

a) <Fluxo Alternativo 1>

1. Usuário clica em Limpar para limpar a informação da tela.

Fluxo de Exceção:

a) <Fluxo de Exceção 1>

1. Usuário fecha o *browser* direto, sem executar qualquer tipo de funcionalidade.

Interface Gráfica:

A figura 127 demonstra a interface do caso de uso de pesquisa de pessoa.

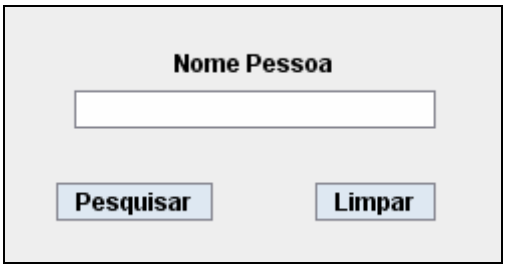


Figura 127 – Interface Web com AJAX – tela de pesquisa pessoa
Fonte: Os autores.

Diagrama de Robustez

Na figura 128 demonstra o diagrama de robustez do caso de uso cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Cadastrar

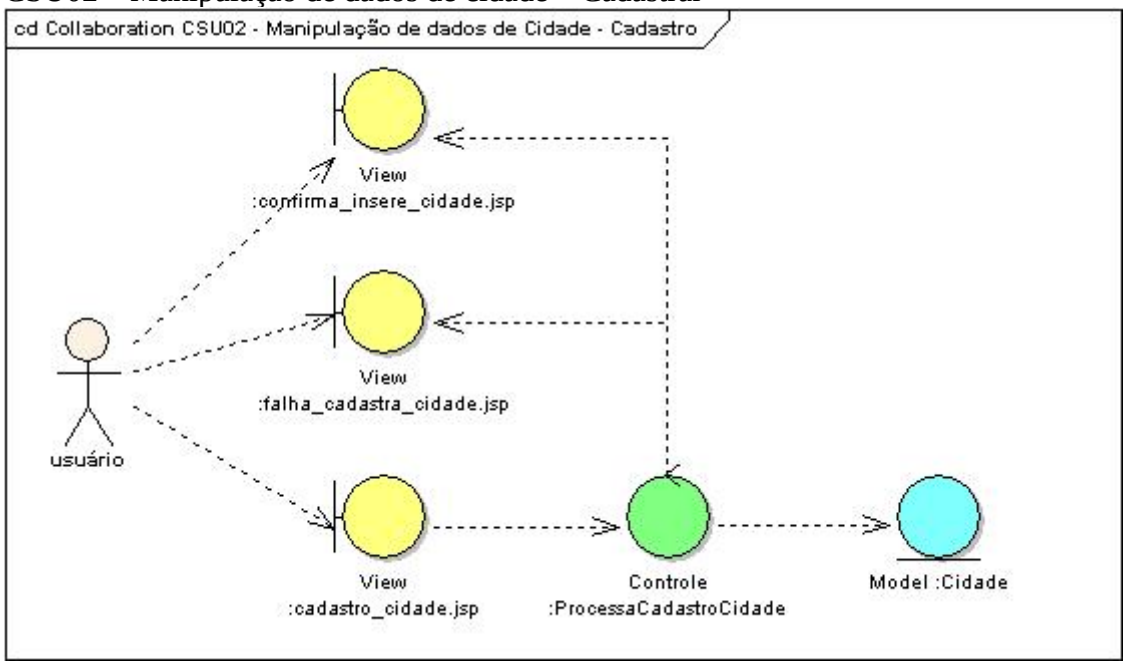


Figura 128 – Diagrama de robustez do CSU02 – Web com AJAX - cadastrar cidade
Fonte: Os autores.

Na figura 129 demonstra a diagrama de robustez do caso de uso alterar cadastrar cidade.

CSU02 – Manipulação de dados de cidade – Alterar

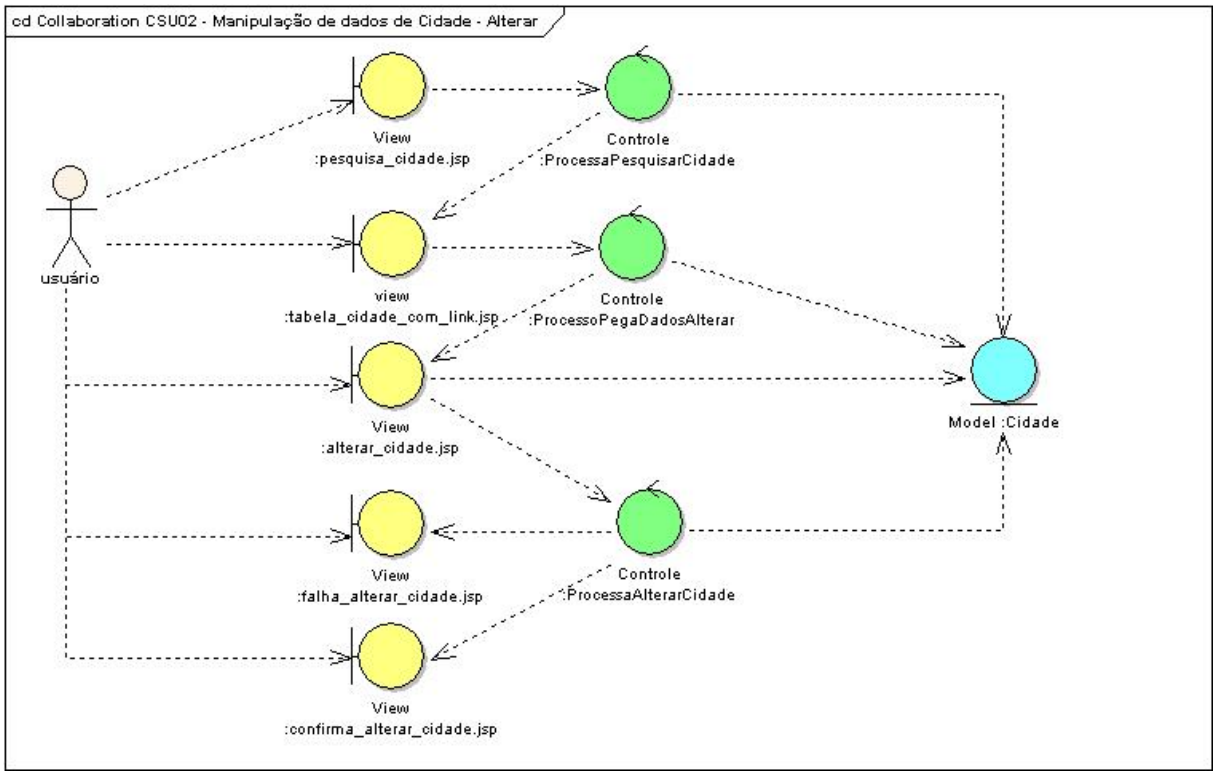


Figura 129 – Diagrama de robustez do CSU02 – Web com AJAX - alterar cidade
Fonte: Os autores.

Na figura 130 demonstra a diagrama de robustez do caso de uso excluir cidade.

CSU02 – Manipulação de dados de cidade – Excluir

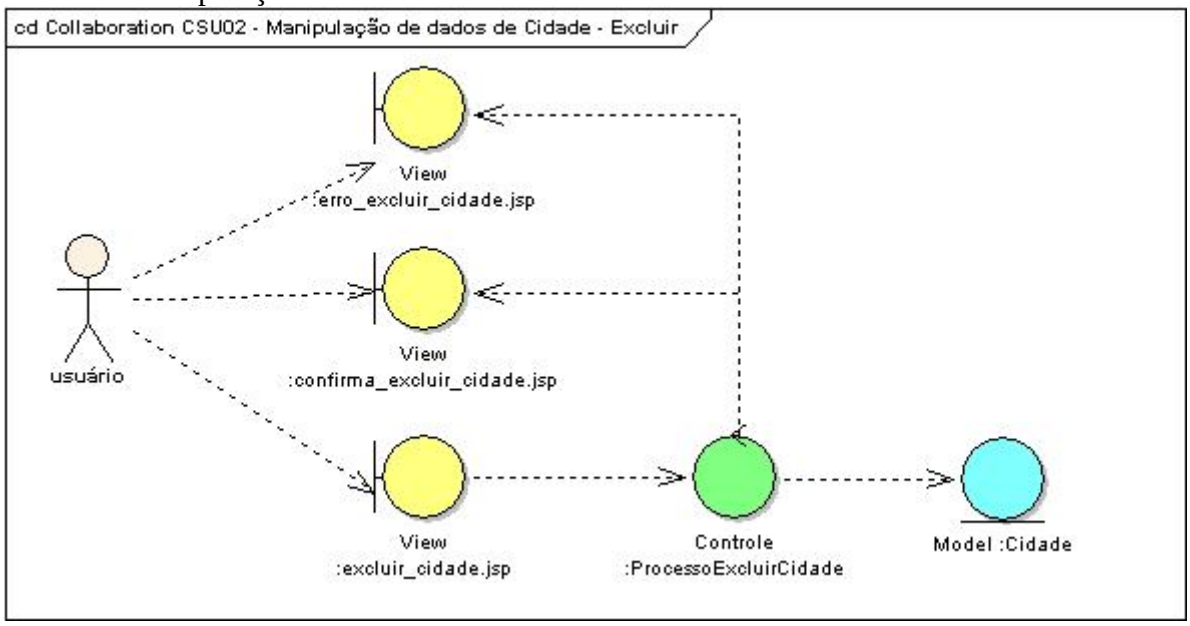


Figura 130 – Diagrama de robustez do CSU02 – Web com AJAX - excluir cidade
Fonte: Os autores.

Na figura 131 demonstra o diagrama de robustez do caso de uso pesquisa de cidade.

CSU03 – Pesquisa por cidade

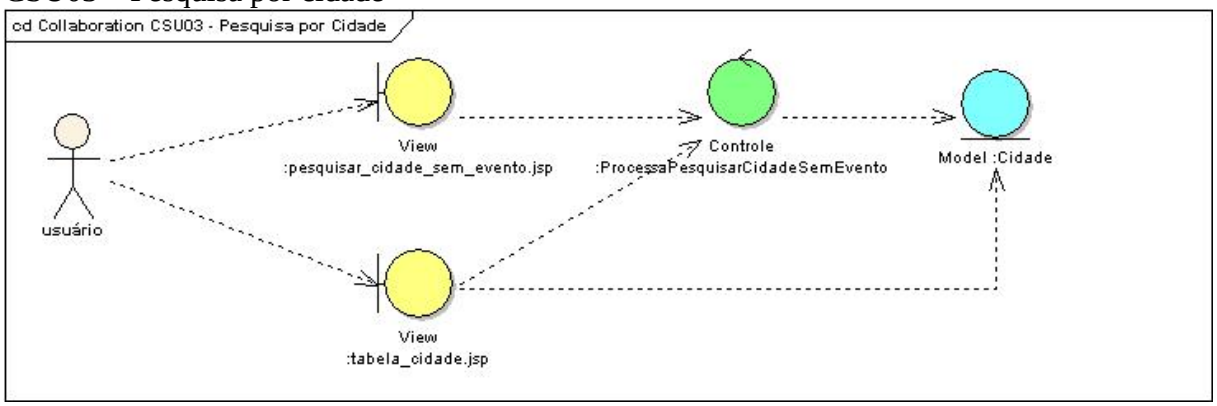


Figura 131 – Diagrama de robustez do CSU03 – Web com AJAX - pesquisa por cidade
Fonte: Os autores.

Na figura132 demonstra o diagrama de robustez do caso de uso pesquisa de pessoa.

CSU04 – Pesquisa por pessoa

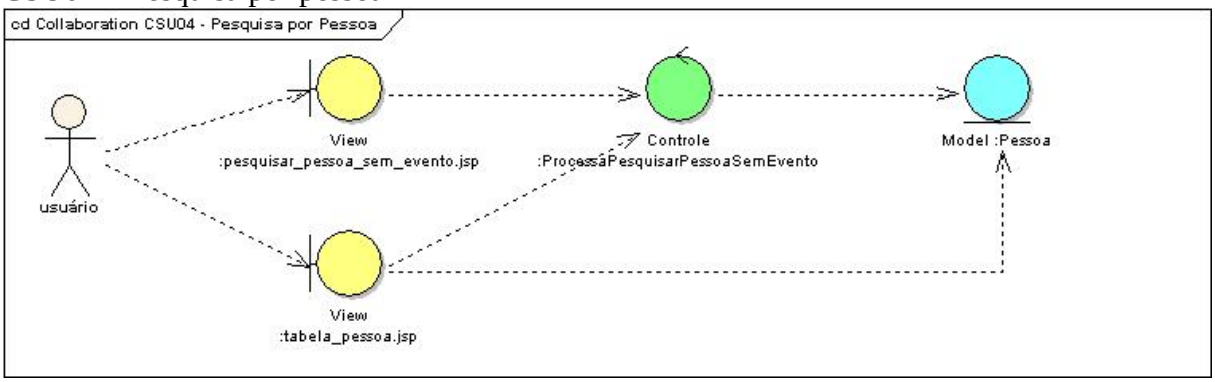


Figura 132 – Diagrama de robustez do CSU04 – Web com AJAX - pesquisa por pessoa
Fonte: Os autores.

Diagrama de Seqüência

Na figura 133 demonstra a diagrama de seqüência do caso de uso cadastro de cidade.
CSU02 – Manipulação de dados de cidade – Cadastro

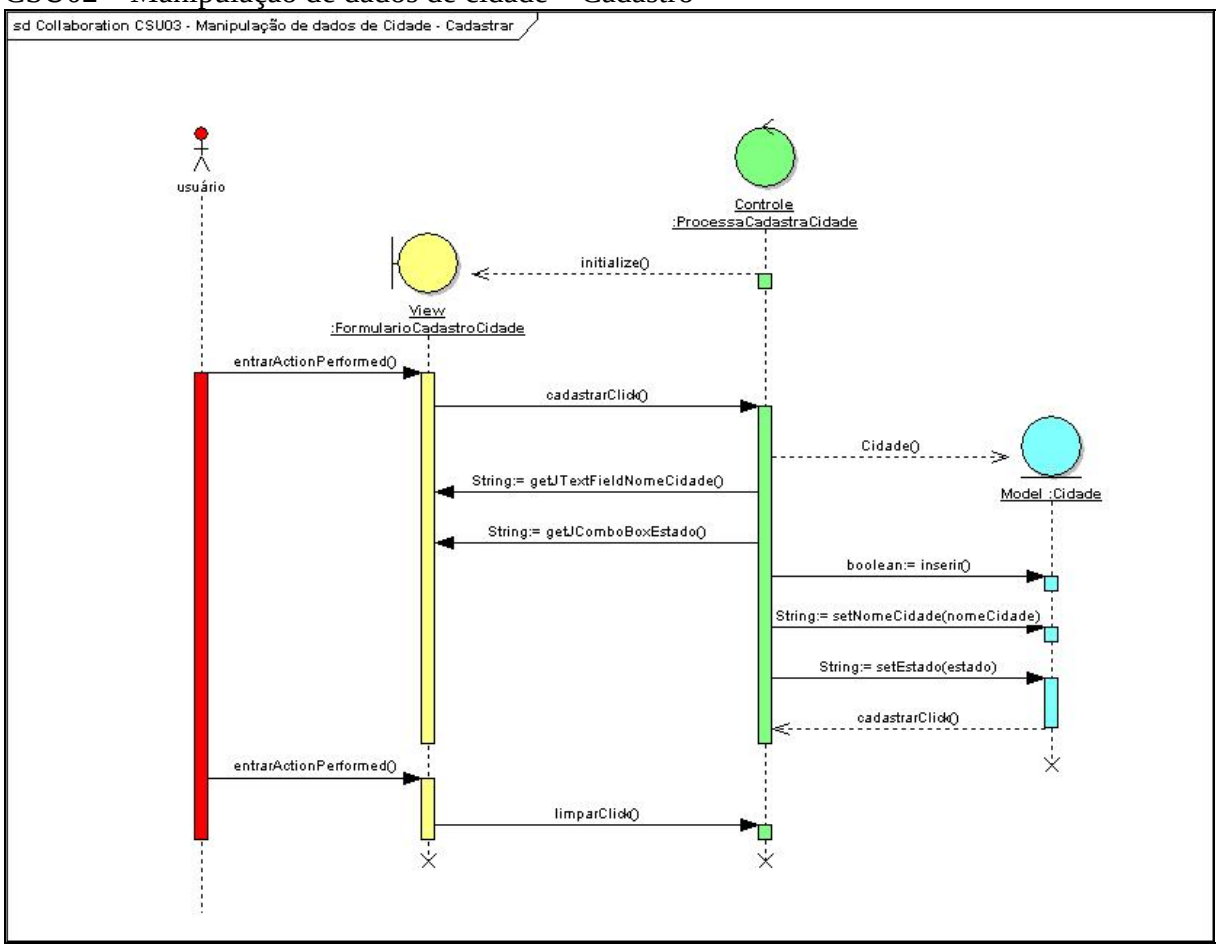


Figura 133 – Diagrama de seqüência do CSU02 – Web com AJAX - cadastro de cidade
Fonte: Os autores.

Na figura 134 demonstra a diagrama de seqüência do caso de uso alterar de cidade.
CSU02 – Manipulação de dados de cidade – Alterar

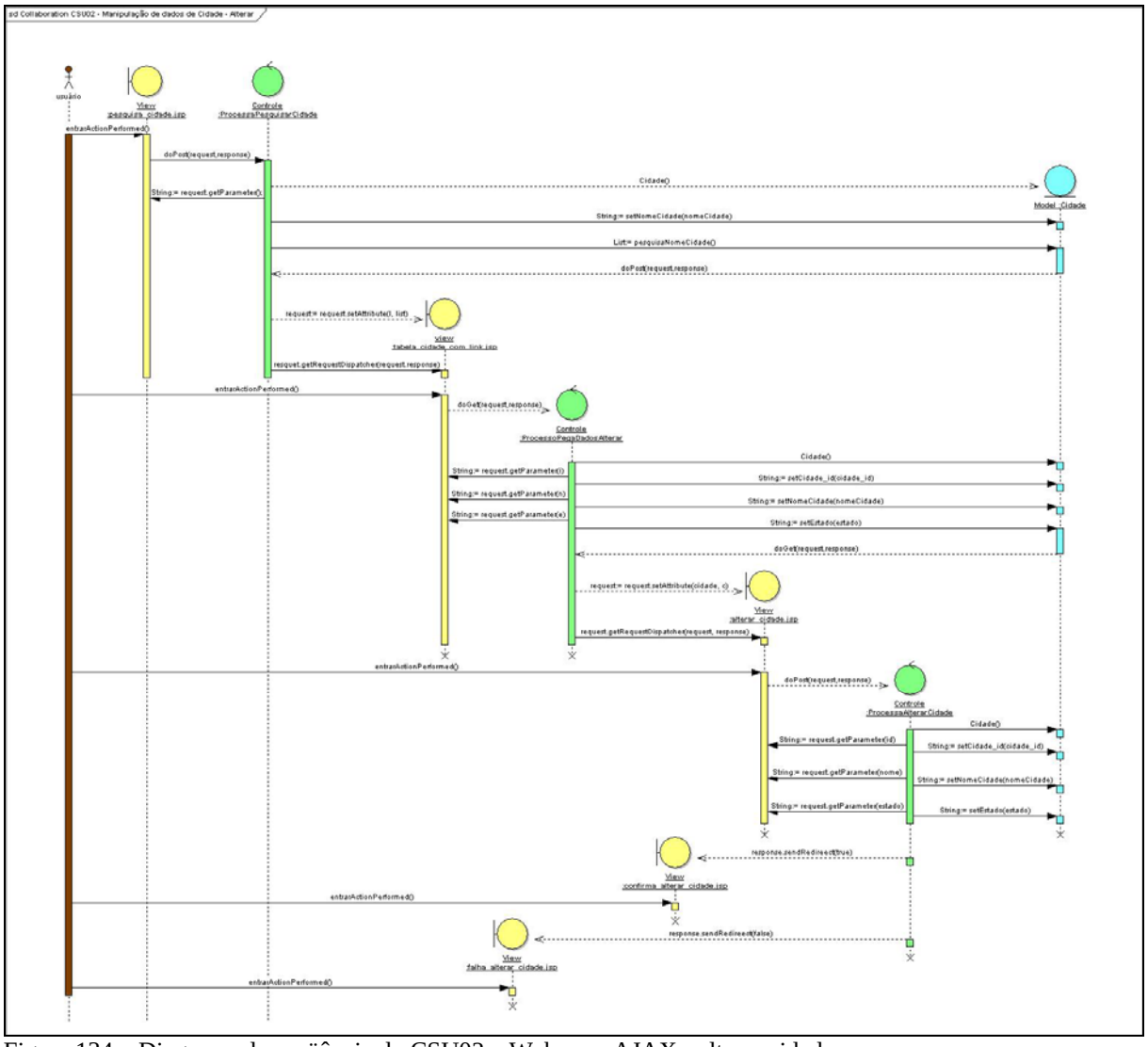


Figura 134 – Diagrama de seqüência do CSU02 – Web com AJAX - alterar cidade
Fonte: Os autores.

Na figura 135 demonstra a diagrama de seqüência do caso de uso excluir cidade.
CSU02 – Manipulação de dados de cidade – Excluir

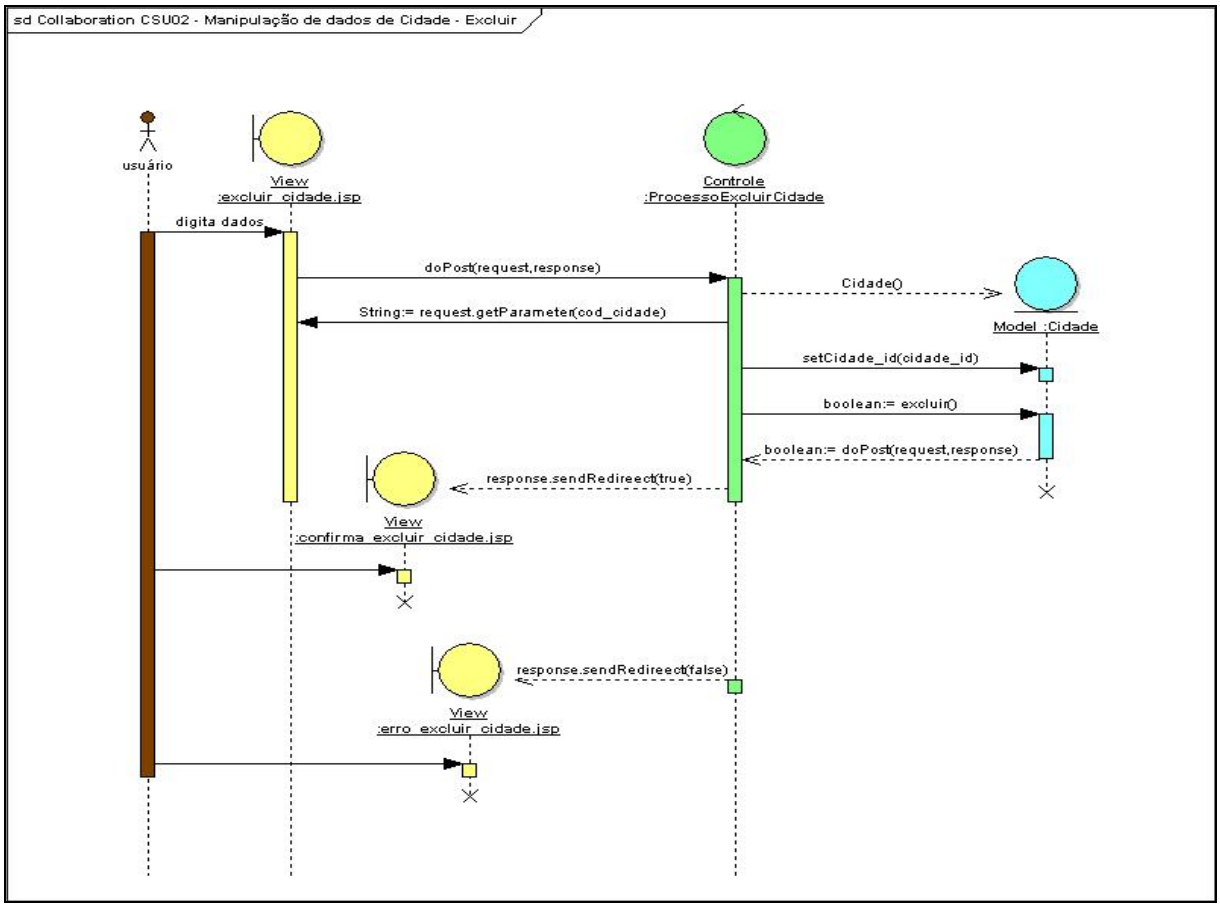


Figura 135 – Diagrama de seqüência do CSU02 – Web com AJAX - excluir cidade
Fonte: Os autores.

Na figura 136 demonstra a diagrama de seqüência do caso de uso pesquisa de pessoa.
CSU05 – Pesquisa por pessoa

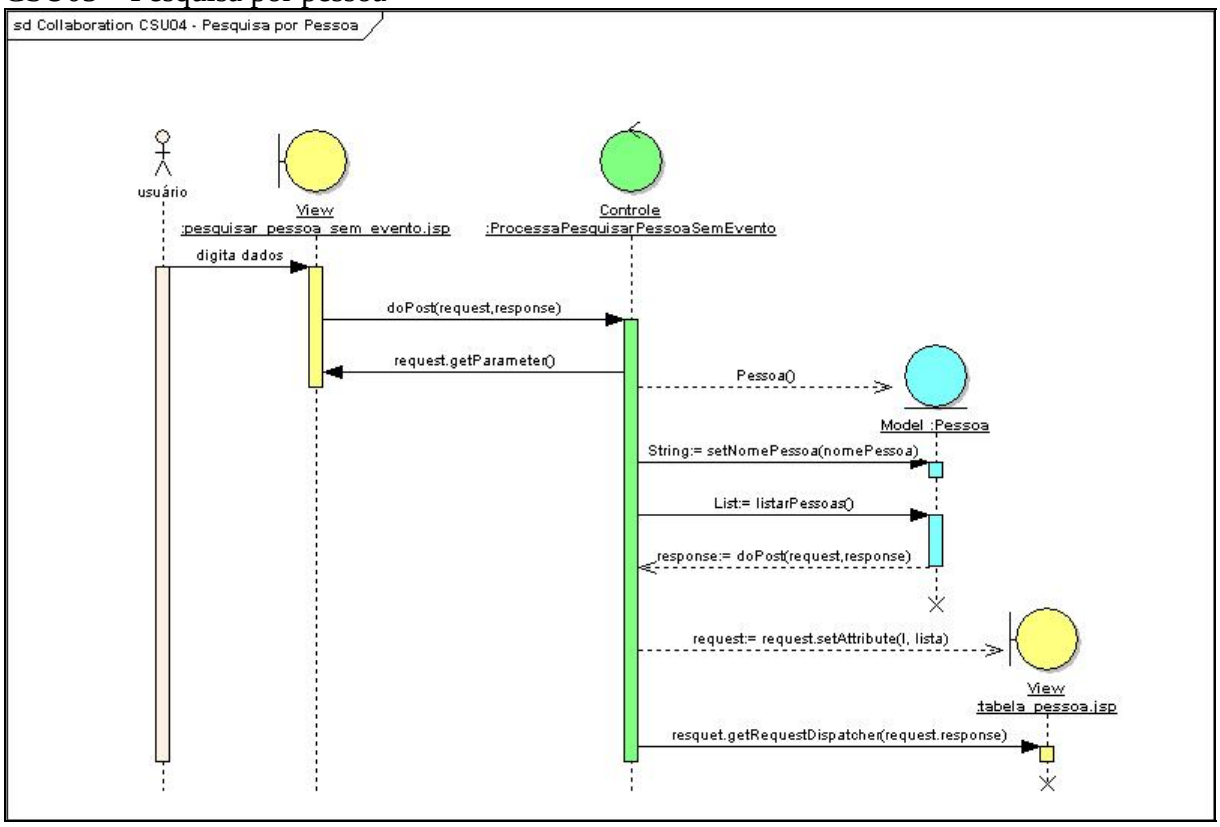


Figura 136 – Diagrama de seqüência do CSU05 – Web com AJAX - pesquisa por pessoa
Fonte: Os autores.