UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

ENGENHARIA DE SOFTWARE APLICADA À WEB – ORIENTAÇÃO A OBJETOS

Autores: Lucas Lopes Alenquer

Orientador: Prof. Antônio Cláudio Gómez de Sousa

Examinador: Prof. Marcelo Luiz Drumond Lanza

Examinador: Prof. Sérgio Palma da Justa Medeiros

DEL

Dezembro de 2005

Resumo

O trabalho desenvolvido realizou uma iteração sobre um processo de desenvolvimento de uma aplicação Web já em fase de produção, adicionando novos requisitos e considerando a orientação a objetos.

Ao longo do trabalho são discutidas boas práticas no processo de Engenharia de Software e problemas, limitações e vantagens ao adotar a orientação a objetos para uma aplicação Web.

Os resultados obtidos foram um conjunto de telas e classes de negócio e auxiliares que implementam os novos requisitos e estão prontos para serem incorporados ao restante do sistema.

Palavras-chave

Engenharia de Software, UML, World Wide Web, Orientação a Objetos, Aplicação Web

Índice

1 Introdução

O trabalho desenvolvido a seguir tem como objetivo discorrer a respeito de como aplicar os preceitos da Engenharia de Software com o paradigma orientado a objetos no desenvolvimento de aplicações Web.

Atualmente existe uma grande corrente que conduz a indústria de software para repensar a maneira como os programas são feitos e distribuídos aos seus clientes. Cada vez mais se escuta falar sobre "software como um serviço", "hospedagem de aplicações", etc. Por mais que estas definições sirvam para denominar diferentes abordagens de utilização do software, todas elas possuem em comum o fato de utilizarem a Web como canal de comunicação entre um sistema e seus usuários.

Esta maior demanda por sistemas na Web implica em softwares que atendam à requisitos cada vez mais complexos e que se comuniquem com uma série de outros sistemas externos, dependendo do contexto da aplicação.

Percebendo toda esta movimentação, a pergunta que surge é: como desenvolver e gerenciar o desenvolvimento de aplicações Web utilizando as ferramentas de que dispõe-se hoje? A UML permite a modelagem de software em níveis de abstração condizentes com cada fase do projeto. Porém, como a UML trataria as especificidades que uma aplicação Web possui? Cada tela da aplicação representaria um subsistema próprio? E caso esta mesma tela abra uma outra cujo estado depende da primeira? Como realizar a interação entre o usuário que utiliza o navegador e o servidor que executa a aplicação? E as interações do usuário com o sistema que não passam pelo servidor?

Por conta destas questões, foi utilizada uma extensão da UML chamada WAE (Web Application Extension) para conseguir modelar de forma adequada uma aplicação Web com todas as suas características próprias. A WAE representa um conjunto de protótipos de classes e relacionamentos UML específicos do ambiente Web, tais como a página do servidor e scripts do cliente.

Como exemplo de sua utilização, foi realizada uma iteração sobre um projeto baseado em uma Intranet com o objetivo de inserir o paradigma orientado a objetos em alguns requisitos de uma aplicação Web existente e sua eficácia será discutida.

Além desta introdução, o projeto pode ser dividido em quatro partes principais:

- Capítulo II, onde são apresentados os conceitos básicos que cercam uma aplicação Web;
- Capítulos III e IV, que explica o processo de desenvolvimento de uma aplicação Web, seus fluxos de trabalho e principais artefatos gerados;
- Capítulos V e VI, que exemplifica o processo descrito anteriormente durante uma iteração de desenvolvimento sobre um sistema de exemplo;
- Capítulos VII e VIII, que resumem as principais conclusões extraídas ao longo de todo o projeto

Neste capítulo será apresentada uma série de conceitos e definições sobre o ambiente Web e suas especificidades. De caráter introdutório, o seu conteúdo serve para melhor compreensão dos capítulos subseqüentes.

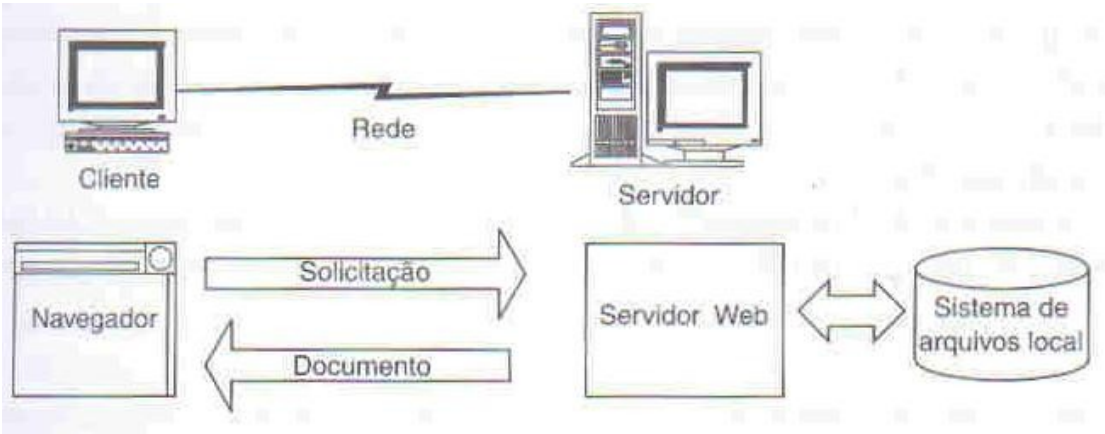
2 CAPÍTULO II - Conceitos básicos

2.1 Sistema Web básico

Os documentos são acessados e visualizados por um software navegador (browser), que é uma aplicação executada pelo computador cliente. O cliente pode através do navegador solicitar documentos de outros computadores da rede e apresentá-los na tela.

O navegador envia uma solicitação do documento para o computador host. Esta solicitação é tratada por um software de aplicação denominado servidor Web, que é uma aplicação que normalmente é executada como um serviço, geralmente utilizando a porta 80.

Ao receber a solicitação, o servidor Web localiza o documento no seu sistema de arquivos locais e o envia para o navegador:



2.1.1 Aplicação Web x Site Web

Existe uma sutil diferença entre Aplicação Web e site Web. Uma Aplicação Web é um sistema que permite que seus usuários executem as regras de negócio através de um navegador Web, com uma interface de um site Web. Desta forma, os usuários da aplicação podem afetar, via entrada de dados, o estado do negócio. E é justamente isto – alterar o estado do negócio – que um site Web não realiza.

2.1.2 HTTP (HyperText Transfer Protocol)

É o protocolo principal de comunicação utilizado entre os navegadores e os servidores Web. Ele especifica como um navegador deve formatar e enviar uma solicitação para o servidor Web.

O HTTP é executado sobre o TCP (Transmission Control Protocol), mas poderá ser executado sobre qualquer serviço orientado para conexão.
O HTTP é um protocolo sem conexão, ou seja, cada vez que o cliente solicita um documento a um servidor Web uma nova conexão deve ser estabelecida com o servidor.

A Internet também possui outros protocolos comumente utilizados, tais como FTP, news, file e Gopher.

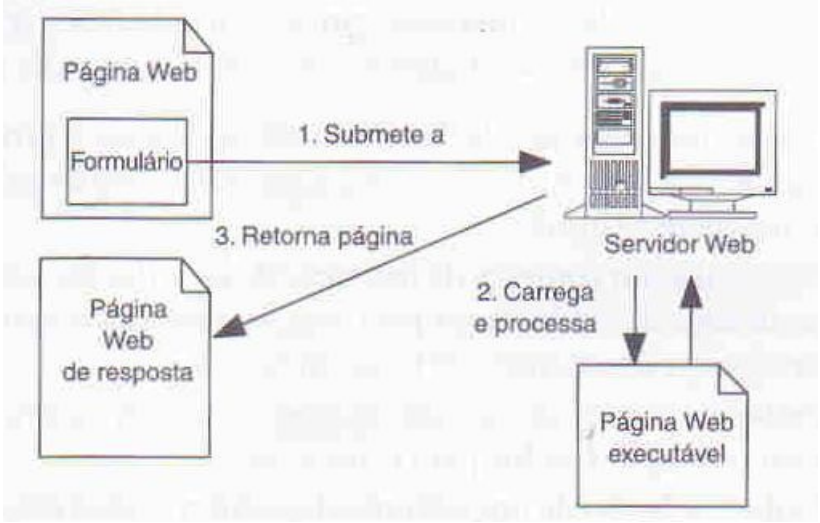
2.1.3 HTML (HyperText Markup Language)

Além de estabelecer conexões com a rede e protocolos para intercâmbio de documentos, os navegadores necessitam apresentar o documento em uma tela. A apresentação do conteúdo é gerenciada pelo navegador.
Para isso existe a HTML, que é uma linguagem baseada em tags para apresentação de documentos de texto. Ela define a formatação do texto, através de fonte, cor, tamanho, etc. Ela também utiliza tags para apontar para imagens a serem incluídas na tela, bem como para definir links para outras páginas Web.

2.1.4 Formulários

Os elementos de formulário da HTML distinguem um site Web de uma aplicação Web. Será através dos campos do formulário que o usuário poderá digitar e/ou alterar valores.

O formulário e seu conteúdo são submetidos ao servidor Web como parte de outra solicitação de página Web. O servidor Web recebe a solicitação para uma página Web especial ou um módulo executável, que é capaz de ler os valores dos campos do formulário e processá-los no servidor. O resultado final é a geração de uma nova página HTML enviada de volta para o navegador.



Geralmente, este processamento dos dados dos campos do formulário envolve comunicação com objetos no servidor ou com bancos de dados.

Os dados do formulário podem ser enviados ao servidor por dois métodos: GET e POST. Quando GET é usado, os valores de todos os campos são anexados ao URL como parâmetros. Quando o método POST é utilizado, o navegador irá empacotar os valores de cada campo em uma seção especial da solicitação chamada corpo de dados (data body).

O parâmetro mais comum é o <input>, que possui como tipos mais comuns os seguintes elementos:

Checkbox, Hidden, Password, Radio, Submit, Text

2.1.5 Aplicações Web

As Aplicações Web usam as chamadas tecnologias de ativação (serão discutidas posteriormente) para tornar o conteúdo dinâmico e para permitir que os usuários alterem a regra de negócio no servidor.

Se não houver nenhuma regra de negócio em um servidor, o sistema não deve ser considerado uma aplicação Web.

Normalmente, os usuários de uma aplicação Web inserem diversos tipos de dados, desde textos simples até opções de uma caixa de seleção ou arquivos.

Para distingüir melhor uma aplicação Web de um site Web, observemos as ferramentas de busca na Internet. Elas simplesmente aceitam os critérios de pesquisa passados pelo usuário e retornam os resultados, sem alteração perceptível no estado da ferramenta de pesquisa.

2.2 Gerenciamento do estado do cliente

O fato das comunicações entre cliente e servidor serem sem conexão não torna fácil o controle do servidor das solicitações dos clientes e associá-las entre si, uma vez que toda solicitação estabelece e logo em seguida interrompe um conjunto novo de conexões.

Imagine um cenário de utilização do sistema. É de se esperar que o usuário navegue por algumas páginas Web. Se não existir um mecanismo de gerenciamento de estado do cliente, todas as informações inseridas anteriormente deverão ser fornecidas para cada nova página, tornando o processo cansativo e custoso. Um exemplo deste provável transtorno seria a obrigação de se inserir seu login e senha para cada página do seu serviço de WebMail.

O W3C propôs um mecanismo de gerenciamento de estado http, conhecido como “cookie”

2.2.1 Cookies

Um cookie é um fragmento de dado que um servidor Web pode pedir a um navegador para manter e retornar cada vez que ele fizer uma solicitação subseqüente de um recurso http para esse servidor. O tamanho do dado pode alcançar até 4K.

Inicialmente, o cookie é enviado de um servidor para o browser adicionando uma linha ao cabeçalho HTTP. Se o browser estiver configurado para aceitar cookies, a linha será aceita e armazenada em algum lugar na máquina do cliente (isto varia de navegador para navegador).

Quando um cookie é enviado ao cliente, ele pode ter até 6 parâmetros preenchidos, que são:

Parâmetro	Obrigatório?	Restrições	Definição
Name	Sim	Não deve conter blank, “,” e “;”	Nome do cookie
Value	Sim	Não deve conter blank, “,” e “;”	Valor do cookie
Expiration date	Não	-	Informa ao browser por quanto tempo ele manterá a informação
Path	Não	-	Utilizado para organizar os cookies dentro do domínio
Domain	Não	-	Utilizado para indicar para quais servidores ou domínios os cookies serão enviados de volta
Require a secure connection	Não	-	Indica se os dados serão enviados via conexão segura

Além do servidor , o JavaScript também pode configurar o valor de um cookie.

2.2.2 Sessão

Uma sessão representa um uso coesivo e único do sistema. Normalmente, uma sessão envolve muitas páginas Web executáveis e bastante interação com a regra de negócio no servidor de aplicação.

O estado da sessão em uma aplicação Web pode ser mantido de quatro maneiras comuns:

1. Colocar todos os valores do estado em cookies;

2. Colocar uma chave única no cookie e usar um dicionário ou mapa gerenciado pelo servidor;

3. Incluir todos os valores de estado como parâmetros em cada URL do sistema;

4. Incluir chave única como um parâmetro em cada URL do sistema e usar um dicionário ou mapa gerenciado pelo servidor.

Os dois primeiros métodos sugeridos, baseados em cookies, têm o inconveniente da dependência do cliente estar com o browser aceitando cookies.

Os dois últimos métodos sugeridos se baseiam na idéia que todos os URLs do sistema serão dinamicamente contruídos para incluir parâmetros que contenham todo o estado da sessão ou apenas uma chave para um dicionário no servidor.

Porém, esta solução poderá se tornar cara se a manutenção do dicionário na memória nunca expirar. Para contornar este problema, a maioria dos dicionários de sessão são removidos quando o usuário da aplicação Web termina o processo ou pára de utilizar o sistema por um determinado período de tempo.

Um valor de timeout da sessão geralmente utilizado é o de 15 minutos.

2.2.3 Tecnologias de Ativação

As tecnologias de ativação são, em parte, o mecanismo pelo qual as páginas Web se tornam dinâmicas e respondem à entrada do usuário.

A mais antiga tecnologia de ativação envolve a execução de um módulo separado por um servidor Web. Em vez de solicitar uma página HTML formatada do sistema de arquivos, os navegadores deviam solicitar o módulo, que o servidor Web interpreta como uma solicitação para carregar e executar o módulo. A saída do módulo é normalmente uma página HTML formatada adequadamente, mas poderia ser qualquer outro dado.

O mecanismo original para processar a entrada do usuário em um sistema Web é o Common Gateway Interface (CGI). Os módulos CGI podem ser escritos em qualquer linguagem, podendo ser ainda em forma de script.

Os dois grandes problemas ao utilizar CGI são que ele não proporciona automaticamente serviços de gerenciamento de sessão e que cada execução do módulo CGI requer um processo novo e separado.

Hoje já existem soluções que resolvem o problema do multiprocesso do CGI. Ao adicionar plug-ins ao servidor Web, permite-se que o servidor web se preocupe apenas com o serviço de solicitações HTTP padrão e transfira as páginas executáveis para outro, já executando o processo.

Os dois principais enfoques para as tecnologias que ativam a aplicação Web são: módulos compilados e scripts interpretados.

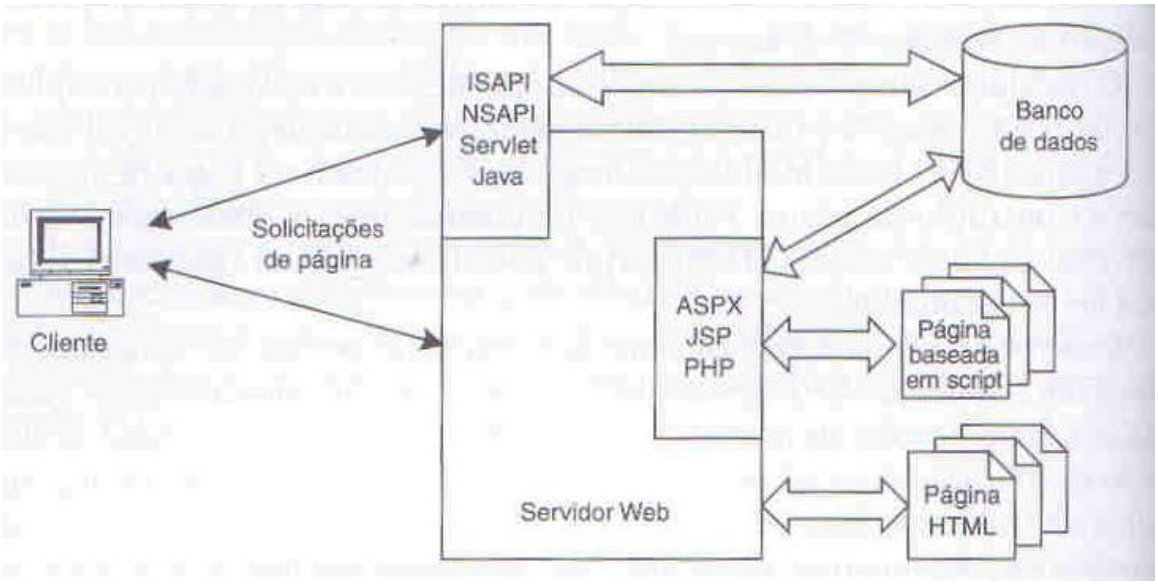
Os módulos compilados são uma solução eficiente e adequada para aplicações de alto volume.

As desvantagens recaem sobre o desenvolvimento e manutenção, uma vez que geralmente estes módulos combinam regra de negócio com a construção de página HTML, tornando a leitura difícil para o programador. Além disso, a cada nova atualização a aplicação Web tem de ser fechada e o módulo descarregado.

Enquanto o módulo compilado parece um programa de lógica do negócio que ocorre para emitir HTML, as páginas baseadas em script parecem uma página HTML que ocorre para processar uma lógica do negócio.

Estas páginas possuem scripts que serão interpretados e irão interagir com objetos no servidor. Feito isto, finalmente irão produzir uma saída HTML.

Normalmente, a extensão do arquivo determina ao servidor Web qual servidor de aplicação deverá ser usado para pré-processar a página. Exemplos: JavaServerPages, Microsoft's Active Server Pages e o PHP.



A figura acima demonstra o relacionamento entre os componentes das tecnologias de ativação e o servidor Web.

A solução de módulo compilado quase intercepta as solicitações de página Web do servidor.

A solução de páginas baseadas em script é chamada pelo servidor Web apenas após ele ter determinado que a página realmente tem scripts para interpretar. Quando ele recebe uma solicitação para essa página, ele localiza a página no diretório especificado e a entrega para o servidor de aplicação. O servidor de aplicação pré-processa a página, interpretando todos os

scripts de servidor existentes e interagindo com os recursos de servidor, se necessário. O resultado é uma página HTML formatada adequadamente que é enviada para o navegador do cliente.

O sucesso desta última solução se deve ao fato de facilitar o desenvolvimento e a distribuição, funcionando como uma "cola" que liga os aspectos da interface de usuário HTML do sistema com os componentes da regra de negócio.

2.3 Clientes Dinâmicos

Até o momento discutimos as aplicações Web e suas tecnologias de ativação imaginando que toda a execução da lógica do negócio estando centrada no servidor. Sob este ponto de vista, o browser se torna apenas um dispositivo de interface generalizado e simples.

Porém, os desenvolvedores de sistema passaram a imaginar que uma partilha de responsabilidades entre cliente e servidor poderia ser bastante proveitosa para ambos. A idéia era simples, mas alguns problemas como a distribuição e independência de plataforma deveriam ser solucionados.

A lógica do negócio é composta de regras e processos que formam o estado do negócio do sistema. Ela é composta de validações e sistemas de dados.

Já a lógica de apresentação se concentra em tornar a interface com o usuário mais fácil. Ela pode aceitar qualquer dado que contribua para o estado do negócio do sistema, mas não é responsável por ele.

Por exemplo, imaginemos uma tela com um controle de seleção de data de nascimento de um usuário do sistema, que se exibe como um minicalendário no navegador. Este artefato captura a entrada do usuário (data de nascimento), não constituindo, assim, uma lógica do negócio. A utilização desta data para o cálculo de um período de validade, por exemplo, entretanto, é uma lógica do negócio.

Os exemplos mais comuns da lógica do negócio no cliente são validações de campo e formulário. Por exemplo, é plausível que o sistema não aceite que o usuário entre com a data de 30 de Fevereiro. Ainda assim, ele pode digitá-la e enviar este dado incorreto para o servidor, que apenas após o recebimento irá verificar o erro e reenviar o formulário inteiro de volta ao cliente. Esta solução exige um tempo demasiado grande, uma vez que a duração entre a submissão do formulário ao servidor, o recebimento do mesmo e o seu reenvio pode levar segundos.

Imaginando uma solução com um cliente dinâmico, o valor da data preenchido seria verificado pela máquina do cliente, possivelmente com o usuário, já que a ação não requer recursos do servidor.

Vários mecanismos e tecnologias trazem a lógica do negócio para o cliente. Todas elas podem ser utilizadas para melhorar a interface com o usuário, e podem ser divididas em duas categorias:

- baseadas em script
- compiladas

Entretanto, todas compartilham alguns pontos em comum, que são:

- Estão associadas à uma página Web e podem acessar e modificar seu conteúdo;

- São automaticamente distribuídas, ou descarregadas no computador cliente conforme necessário;

- Tratam de questões específicas de segurança.

A escolha de qual solução usar na máquina do cliente – se baseada em script ou compilada – irá depender do grau de complexidade da lógica de negócio a ser executada. Quando ela é mínima, o script é o meio mais simples. Quando se deseja executar uma lógica mais sofisticada, geralmente são construídos applets Java, controles JavaBeans ou ActiveX. Haverá uma discussão posterior sobre estas tecnologias.

Vale ressaltar que a utilização da lógica de negócio por parte do cliente deve ser incentivada sempre que ela acesse apenas recursos no cliente. Uma lógica de negócio que requer recursos no servidor não é uma candidata a migrar para o cliente.

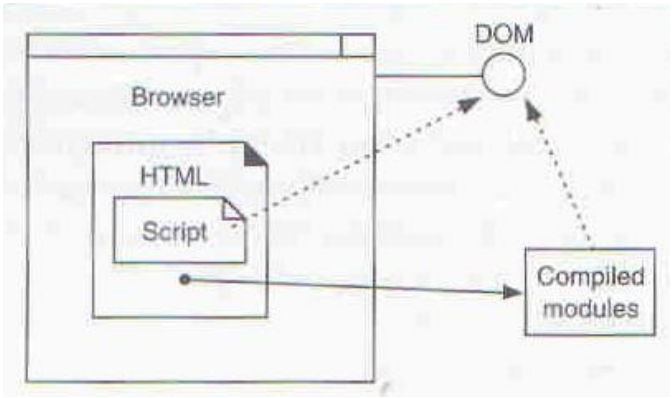
A lógica de negócio que é implementada no cliente precisa se concentrar no tratamento de dados na página Web propriamente dita.

2.3.1 Document Object Model (DOM)

O DOM representa uma plataforma neutra para o navegador e o documento HTML desenvolvida pelo W3C e está implementada na grande maioria dos navegadores atualmente.

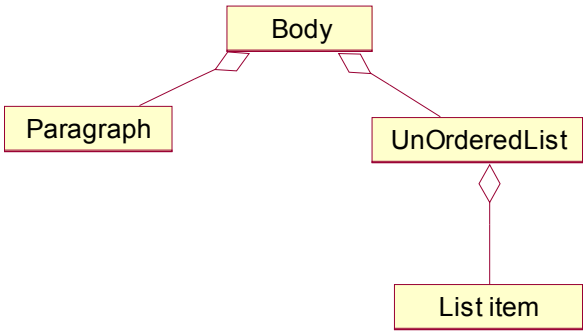
A idéia básica é possuir uma API comum com a qual os desenvolvedores Web possam tratar o conteúdo do documento HTML e XML, assim como os recursos do próprio navegador.

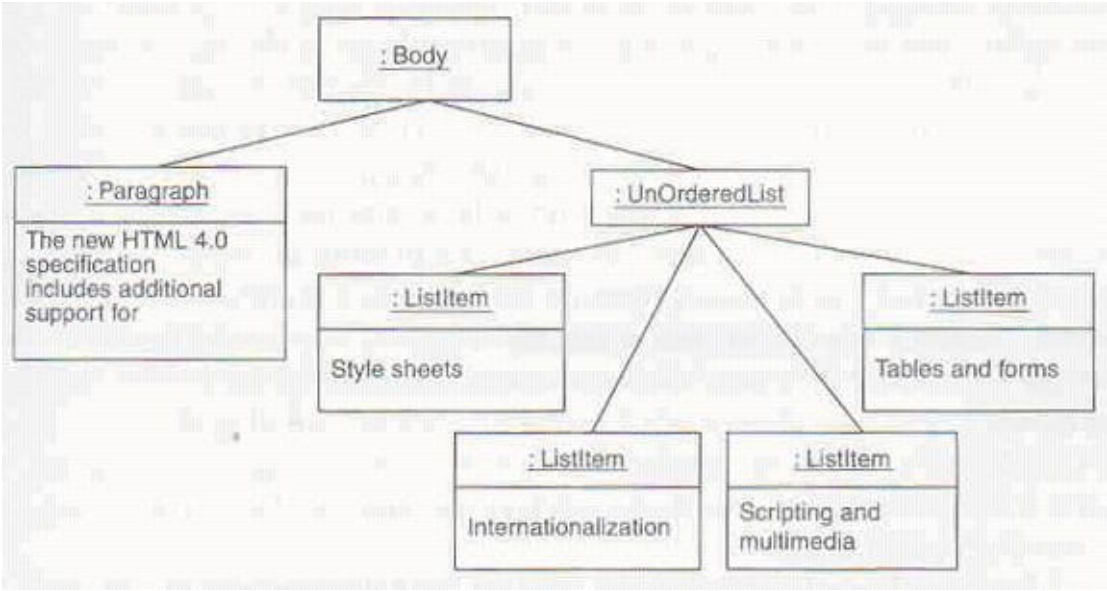
A figura abaixo exibe os relacionamentos entre o navegador, o documento HTML, os scripts, os módulos compilados e o DOM.



O documento contém os scripts que utilizam a interface DOM. Ele também possui referências aos módulos compilados, que também utilizam interface DOM.
Já o navegador é responsável pela apresentação do documento HTML dinâmico e pela implementação da interface DOM dos scripts e programas para modificar o documento HTML.

Como a sigla DOM demonstra, os documentos são tratados como objetos possuindo dados e comportamento. A colaboração entre esses objetos constitui a estrutura do documento. Como um documento é uma coleção de objetos, ele pode ser representado por um modelo de objetos:





Os três principais objetivos do DOM são especificar:

- As interfaces e objetos usados para representar e tratar um documento;
- As semânticas destas interfaces e objetos;
- Os relacionamentos e colaborações entre estas interfaces e objetos

2.3.2 Script

Dentre as tecnologias de script mais comuns destaca-se o JavaScript. O JavaScript é uma linguagem baseada em objeto, e não orientada a objeto. Isto significa que ele utiliza objetos extensíveis internos, mas não suporta classes definidas pelo usuário ou herdadas.

O JavaScript não é compilado. Ele utiliza binding dinâmico e todas as referências a objeto são verificadas em tempo de execução. O seu código vem incorparado ao HTML e é delimitado pela tag <script>:

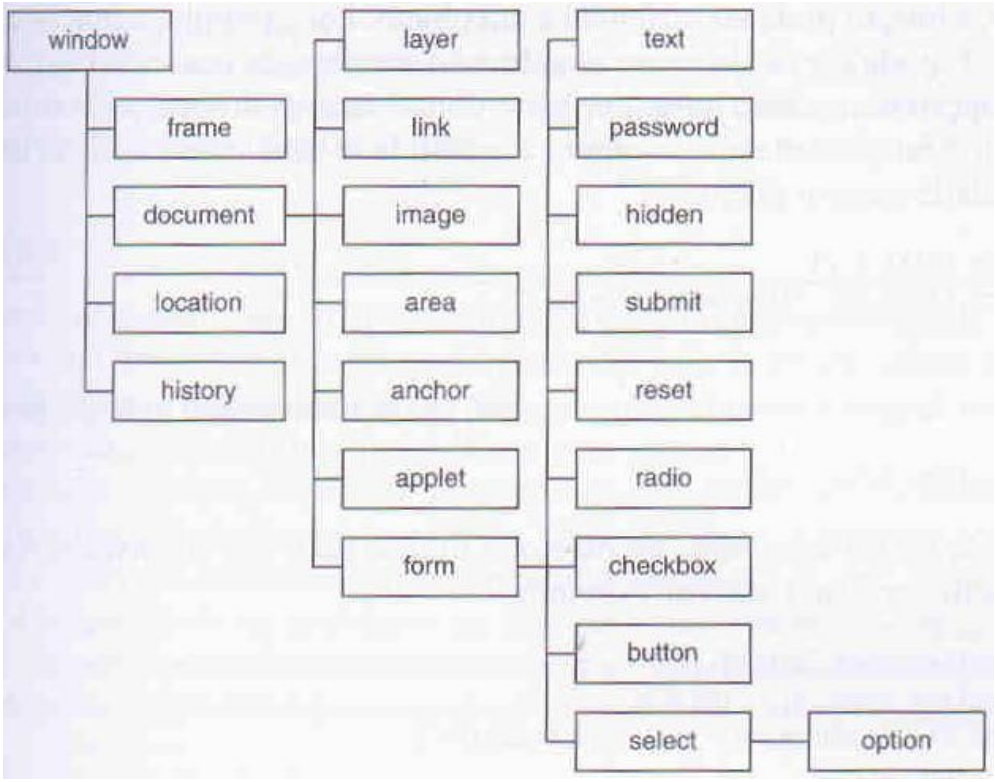
```
<script language = ”javascript”>
        alert(“Hello World”);
</script>
```

2.3.3 Objetos JavaScript

O DOM, que é a fonte principal de objetos para JavaScript, coloca uma interface de objeto tanto no documento HTML quanto no browser. O JavaScript pode interagir com o browser - e dele extrair informações como o seu histórico – ou com outras páginas de um frameset.

O objeto principal utilizado no trabalho sobre o conteúdo de um documento é o objeto document, cuja referência pode ser alcançada pela propriedade document do objeto window, existente em qualquer função JavaScript.

A figura abaixo mostra a hierarquia DOM disponível para scripts em uma página:



2.3.4 Eventos

O JavaScript incorporado ao HTML também é lido e interpretado para que apareça no documento. Por isso, alguns problemas podem ocorrer. Imagine o caso em que um trecho em JavaScript faça chamada a duas funções, uma seguida da outra, e que a segunda dependa do resultado da primeira para funcionar corretamente. O navegador, tão logo seja chamada a primeira função, irá para a próxima linha sem aguardar o encerramento da primeira.

Para contornar este tipo de problema e organizar o documento se criam funções que encapsulem os comportamentos desejados para que sejam executadas conforme necessário.

Desse modo, uma função irá responder a um evento. Como por exemplo, uma função irá responder ao evento onLoad do elemento <body> da página HTML quando o documento for carregado pela primeira vez.

A tabela a seguir mostra um resumo dos eventos a que um documento HTML pode responder, valendo ressaltar que nem todos os elementos do documento HTML podem responder a todos os eventos:

Evento	Aplica-se a	Ocorre quando	Manipulador de evento
Abort	Imagens	O usuário aborta a carga de uma imagem, clicando, por exemplo, em um link ou no botão Stop.	OnAbort
Blur	Janelas, quadros e todos os elementos de formulário	O usuário remove o foco de entrada da janela, quadro ou elemento do formulário	OnBlur
Click	Botões, botões de opção, caixas de seleção, botões Submit, botões Reset, links	O usuário clica no elemento do formulário ou link	OnClick
Change	Campos de texto, áreas de texto, listas de seleção	O usuário altera o valor do elemento	OnChange
Error	Imagens, janelas	A caraga de um documento ou imagem causa um erro	OnError
Focus	Janelas, quadros e todos os elementos de formulário	O usuário dá o foco para a janela, quadro ou elemento de formulário	OnFocus
Load	Corpo do documento	O usuário carrega a página no navegador	OnLoad
Mouse Out	Áreas, links	O usuário move o ponteiro do mouse para fora de uma área – mapa de imagem do cliente – ou link	OnMouseOut
Mouse Over	Links	O usuário move o ponteiro do mouse sobre um link	OnMouseOver
Reset	Formulários	O usuário redefine um formulário: clica em um botão Reset	OnReset
Select	Campos de texto, áreas de texto	O usuário seleciona	OnSelect

		um campo de entrada do elemento do formulário	
Submit	Botão Submit	O usuário submete um formulário	OnSubmit
UnLoad	Corpo do documento	O usuário sai da página	OnUnLoad

Desta forma, a programação no cliente é voltada para o evento. Quando esta programação em JavaScript é usada para executar a lógica de negócio ela deve ser idealizada durante a fase de design, e possivelmente, nos modelos de análise. Quando ela visa apenas melhorar a apresentação ela deve ser pensada durante a modelagem dos modelos de interface e na criação dos protótipos. Estes tópicos serão melhor abordados e esclarecidos no decorrer do trabalho.

2.3.5 Alternativas: Applets Java e ActiveX/COM

As tecnologias de cliente dinâmico não se restrigem apenas ao JavaScript. Os applets JavaScript ou Java também conferem à página um maior dinamismo por parte do cliente.

O applet é uma espécie de controle de interface com o usuário que é colocado em uma página Web, e é referenciado por uma tag <object>. Dentro desta tag são preenchidos parâmetros que identificam o applet e indicam onde ele pode ser obtido.

Além desses parâmetros, a tag <param> no interior da tag <object> permite que o applet acesse parâmetros definidos pelo programador. Um exemplo de utilização de um applet pode ser visto abaixo:

<OBJECT codetype = ”application/java”
classid = ”java:DatePicker.class”
codebase = http://www.meusite.org/javaclasses/>
<PARAM name = “format” value = “mm-dd-yyyy” type = ”data”>
<PARAM name = “style” value = “DropDown” type = ”data”>

</OBJECT>

Um ponto positivo é o fato de os applets Java, após terem sido descarregados para o cliente, não serão carregados novamente quando forem chamados posteriormente. Cada vez que a página Web que contém o applets é revisitada, o browser verifica se o applet já foi descarregado e se certifica que possui a versão correta. Se houve alteração de versão ou se não existir mais no cliente o applet em cache, o applet é descarregado novamente.

Já os controles ActiveX, ou objetos COM, são a solução da Microsoft para dar dinamismo ao cliente, da mesma maneira que os applets Java. Trata-se de um módulo executável completo que é acionado pelo navegador.

Neste capítulo será sugerido um processo de desenvolvimento de uma aplicação Web, que será dividido em diversas etapas e irão gerar uma série de artefatos que irão organizar e facilitar o gerenciamento do trabalho da equipe ao longo do tempo.

3 CAPÍTULO III - O processo de desenvolvimento de uma aplicação Web

Neste momento será enfatizada a importância em se construir um processo para desenvolver uma aplicação Web.
Uma equipe de desenvolvimento de software pode através de muito esforço e dedicação concluir um projeto em sua forma completa. Porém, esta mesma equipe, através de muito esforço e dedicação, pode, auxiliada por um processo robusto fazer do desenvolvimento de uma aplicação Web uma tarefa repetitiva e confiável.

Será apresentado a seguir um processo de desenvolvimento de software, com seus termos e conceitos que serão largamente utilizados no decorrer do texto.

Independente do processo utilizado, qualquer processo de desenvolvimento de software possui quatro finalidades básicas:

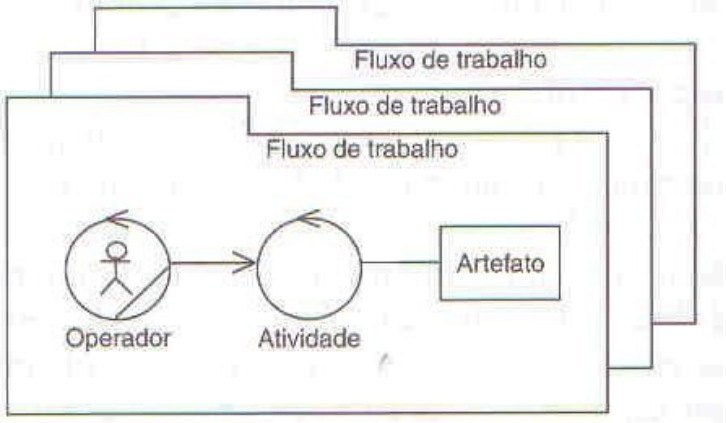
1. Fornecer uma ordenação sobre a seqüência de atividades de uma equipe de desenvolvimento. Desta forma a equipe possui em mãos um roteiro de atividades a serem executadas em uma determinada ordem pré-estabelecida;

2. Especificar os artefatos que serão desenvolvidos

3. Delegar tarefas para cada desenvolvedor individualmente e também para a equipe como um todo

4. Criar critérios para monitorar e comparar produtos e atividades do projeto

O processo de desenvolvimento de software geralmente fica disposto sob a forma de um conjunto de documentos ou um sistema de hipertexto on-line. A figura abaixo demonstra como o processo de desenvolvimento de software implica na criação de fluxos de trabalho que envolvem a definição de atividades executadas por operadores para gerar artefatos pré-determinados.



Vale ressaltar que será utilizada a notação dos diagramas UML para sintetizar graficamente alguns dos fluxos de trabalho, como generalizado acima. Não haverá, neste momento, nenhum compromisso em se manter fiel às determinações que a linguagem UML impõe.

Fluxo de trabalho – é o conjunto de atividades – levantamento de requisitos, análise, projeto, codificação, teste e manutenção – que ao final produz um resultado tangível e observável.

Atividade – é o conjunto de procedimentos que os operadores executam para gerar os artefatos de saída do fluxo de trabalho.

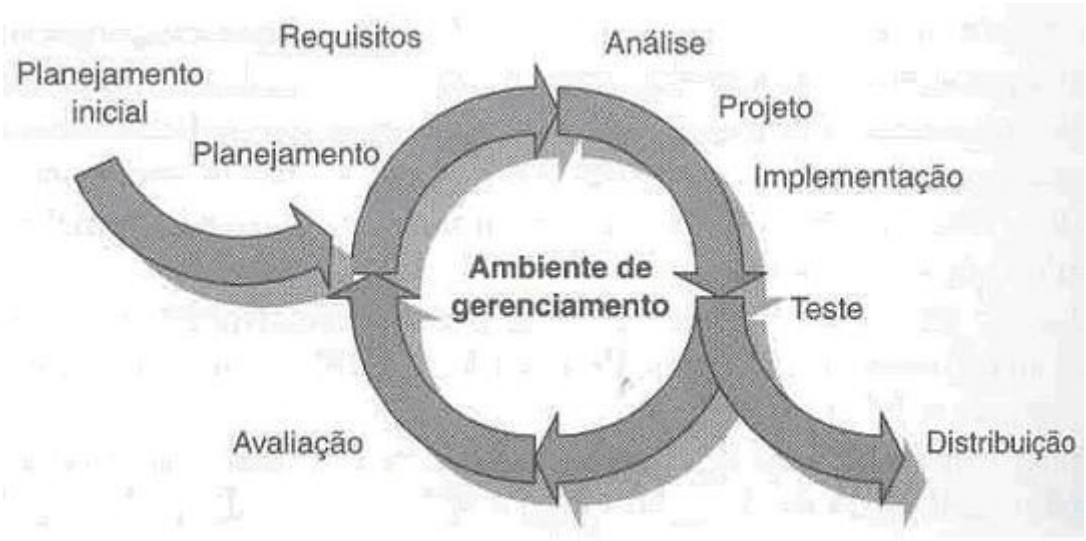
Artefato – é qualquer parte de informação persistente ao projeto produzida pelos operadores durante suas atividades. Podem envolver modelos, elementos de modelo, documentos e código-fonte.

Operador – é um papel executado por uma pessoa durante o processo de desenvolvimento de software

As principais características do processo descrito a seguir são sua orientação a caso de uso, sua centralização na arquitetura e sua natureza iterativa e incremental.

Por se tratar de um processo iterativo, cada fluxo de trabalho (que envolve as diversas atividades que geram artefatos na saída) é repetido e redefinido até que se atenda a todos os requisitos do sistema e seja implantado.

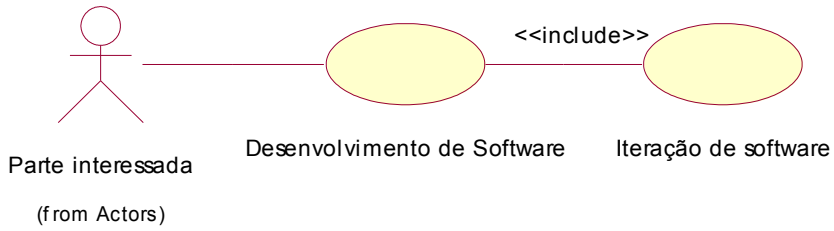
Geralmente, cada atividade do desenvolvimento de software descobre problemas e circunstâncias que põem em cheque ou podem até alterar decisões tomadas anteriormente. A figura abaixo esquematiza a idéia de processo iterativo que se deseja alcançar:



3.1 Representação do processo de desenvolvimento de software

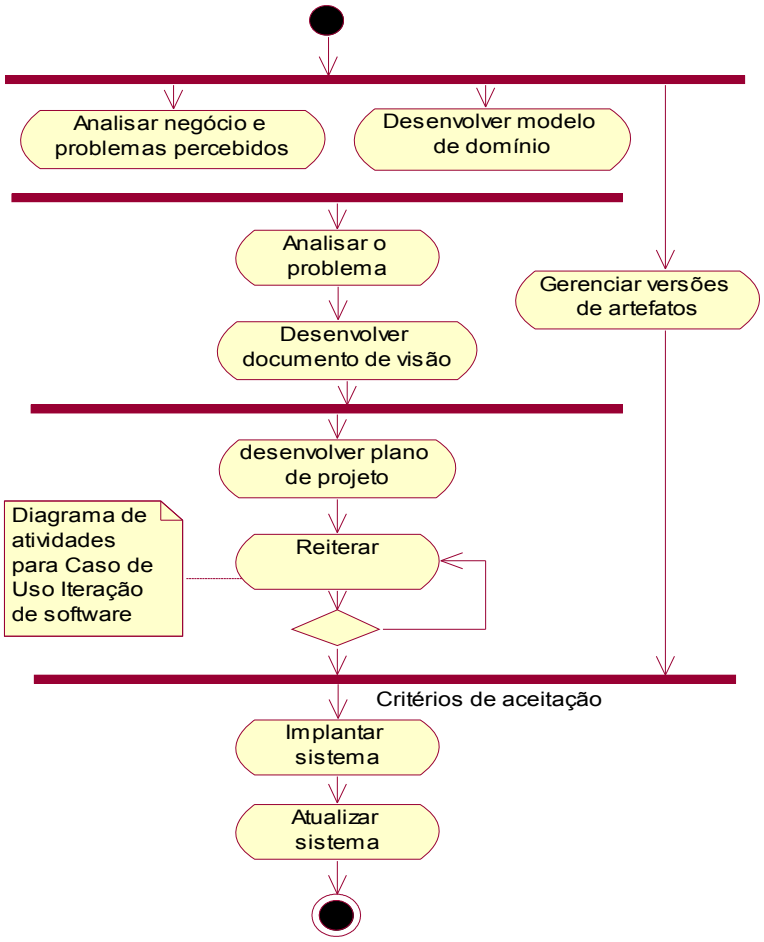
Como usarei bastante os recursos da UML durante todas as etapas do desenvolvimento de software voltado para a Web, também irei representar o processo de desenvolvimento de software através de elementos UML.

No caso do processo em si, temos um caso de uso superior que chamarei de "Desenvolvimento de Software", que irá interagir com as partes interessadas neste processo: clientes, usuários, donos de empresa, etc. Este caso de uso se utiliza de um outro caso de uso crítico para este processo, que darei o nome de "Iteração de Software".



3.1.1 Atividades do Caso de Uso "Desenvolvimento de Software"

As atividades de "Desenvolvimento de Software" podem ser expressas através de um diagrama de atividades. Inicialmente ocorre uma análise do negócio e a identificação de seus problemas mais evidentes (É bastante comum que outros problemas surjam durante todo o processo, daí minha defesa sobre o ciclo iterativo). Será representada, de maneira ilustrativa, a seqüência de atividades de uma iteração apenas:



Simultaneamente, ocorre a atividade de desenvolvimento de um modelo de domínio que irá expressar as entidades e os processos principais do negócio. O documento mais importante a ser gerado nesta atividade será o glossário, que irá definir os termos e conceitos principais do contexto em que o sistema está inserido.

Concluídas as duas primeiras atividades, a equipe pode se dedicar à análise do problema em si e desenvolver o documento de visão, que irá expressar o escopo e a finalidade de todo o projeto de software. Geralmente este documento de visão tem como principal objetivo fornecer os recursos principais e critérios de aceitação para o sistema em desenvolvimento.
Tendo o documento de visão em mãos, a equipe está apta para desenvolver o plano de projeto, que irá esboçar as atividades de desenvolvimento do software como um todo, determinando e dispondo eventos em ordem de prioridade e também gerando plano de gerenciamento de alteração e de configuração.

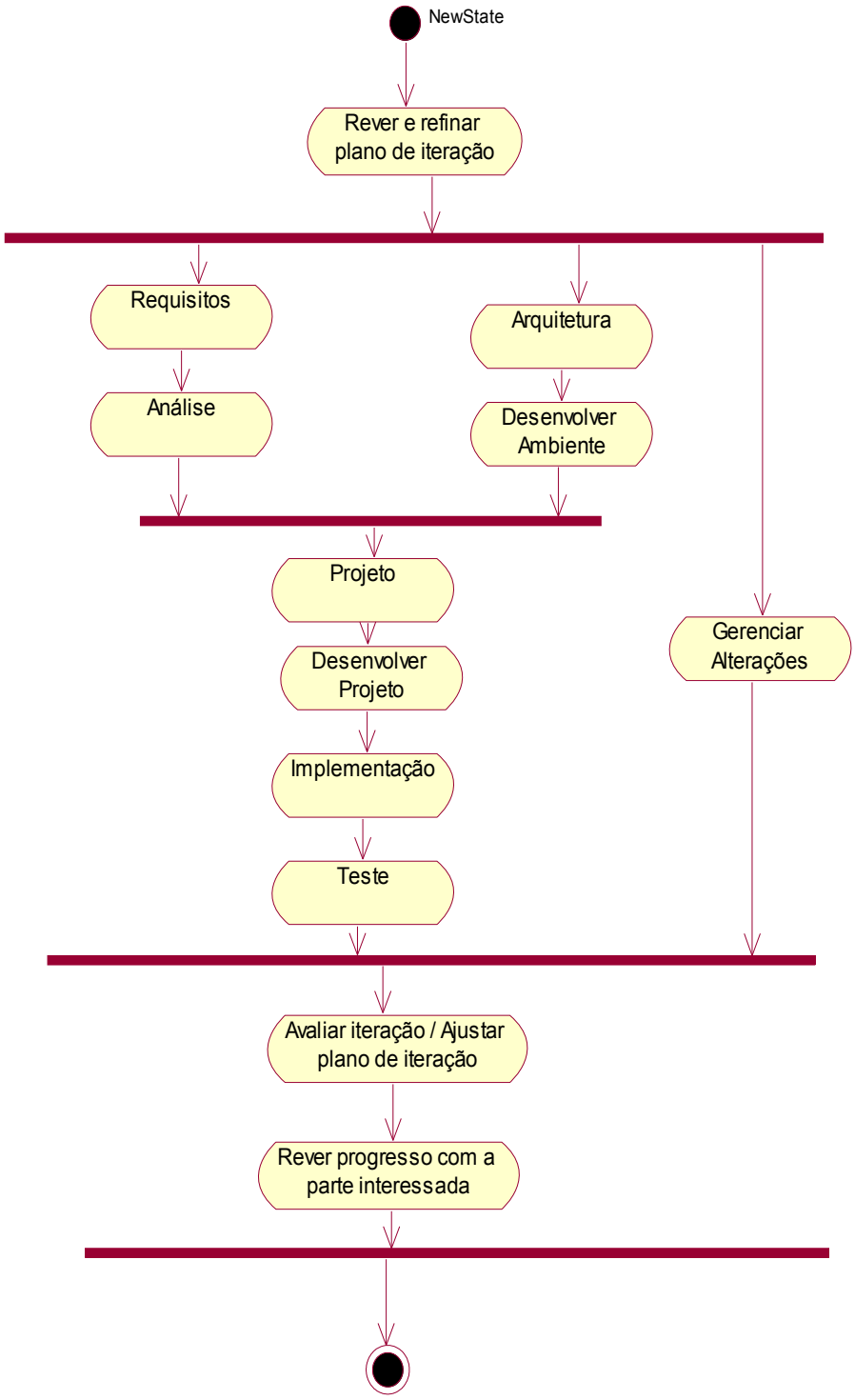
É importante que o plano de projeto contenha o primeiro esboço do plano de iteração, que é a razão do sucesso de todo o processo. Este plano deverá conter as atividades que serão executadas durante cada iteração, os artefatos que deverão ser gerados e, o mais importante, os critérios de avaliação para cada um.

É de se esperar, também, que o plano de iteração sofra mudanças significativas até a conclusão do projeto de desenvolvimento. Porém, o ponto mais importante desta atividade é que as iterações devem ser pensadas antecipadamente.

A atividade de reiterar nos leva ao caso de uso “Iteração de Software”, descrito a seguir.

3.1.2 Atividades do Caso de Uso “Iteração de Software”

O caso de uso “Iteração de Software” corresponde a todas as atividades do processo que envolve a iteração de todos os artefatos que formam o ponto de partida do documento de visão para a formação da coleção de artefatos que irão compor o produto final.


NewState
Rever e refinar plano de iteração
Requisitos
Análise
Arquitetura
Desenvolver Ambiente
Projeto
Desenvolver Projeto
Implementação
Teste
Gerenciar Alterações
Avaliar iteração / Ajustar plano de iteração
Rever progresso com a parte interessada

Como descrito na figura anterior, a característica mais marcante deste caso de uso é que vários os fluxos de trabalho ocorrem em paralelo. Além disso, é recomendável que o gerente da equipe consiga ditar um ritmo constante ao processo, seja através de reuniões de status semanais, relatórios de testes da manhã ou qualquer outra. Isto irá manter a equipe concentrada no desenvolvimento dos artefatos pretendidos e diminuirá a sensação de pânico quando algum problema ocorrer.

Ao longo da iteração sobre todos os artefatos, devem ocorrer as atividades de rever e refinar o plano de iteração, rever o progresso com as partes interessadas (atividade importante para manter investimentos e mostrar o progresso ou estagnação do status do projeto) e também avaliar a iteração ao seu final e ajustar o próximo plano.

3.2 Artefatos Gerados

3.2.1 Modelos

Durante a confecção dos diferentes artefatos que irão compor o sistema, os modelos de sistema se situam entre os principais, por serem um excelente canal de comunicação entre os membros da equipe, e também por auxiliarem no gerenciamento da complexidade do sistema em construção.

Os modelos recomendados para desenvolvimento são os seguintes:

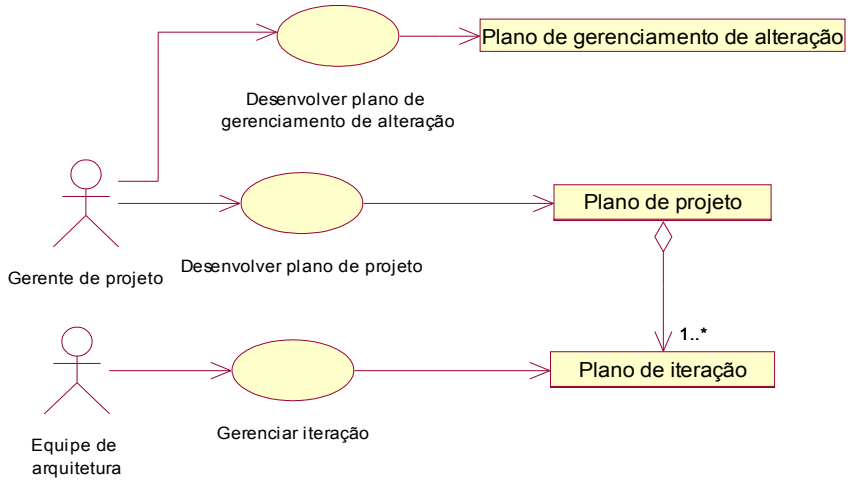
- Modelo de domínio
- Modelo de caso de uso
- Modelo de análise/projeto
- Modelo de implementação
- Modelo de processo
- Modelo de segurança
- Modelo de experiência de usuário ou interface com o usuário

Cada um destes modelos, ainda que possuam um enfoque particular, devem ser consistentes entre si. Será esta consistência que permitirá que a equipe consiga visualizar a interação entre os diversos componentes e também prever o impacto que uma alteração em algum deles teria sobre todo o sistema ou parte dele.

Todos os artefatos principais podem ser agrupados em conjuntos que irão interagir uns com os outros. Neste momento definem-se estes conjuntos, seus atores e fluxos de trabalho necessários para a construção de cada um dos artefatos pretendidos.

3.2.2 Conjunto de gerenciamento de projeto

Os principais artefatos deste conjunto são os planos de projeto, de iteração e de gerenciamento de alteração (veja figura abaixo).

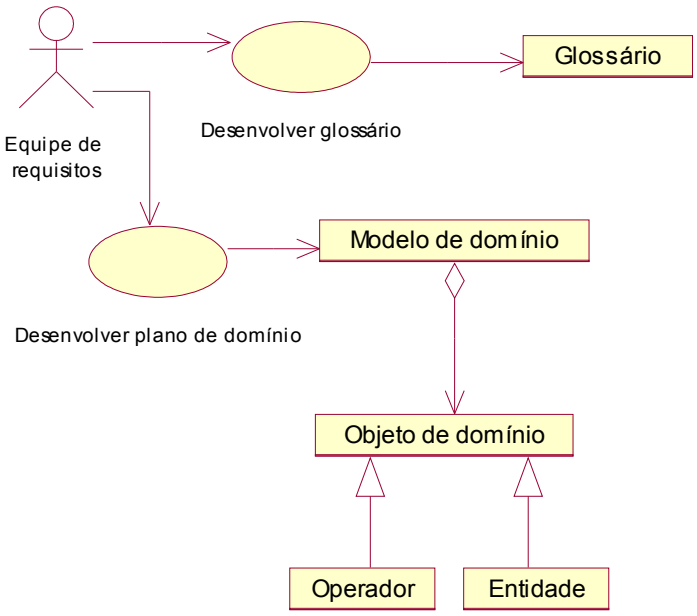


Serão estes documentos que esboçam o cronograma de desenvolvimento, procedimentos de seleção de pessoal, áreas de responsabilidade e eventos principais. Neste momento deverá se buscar metas realistas para cada iteração e também testar as áreas de maior risco da aplicação logo nas primeiras iterações e não nas últimas.

3.2.3 Conjunto de domínio

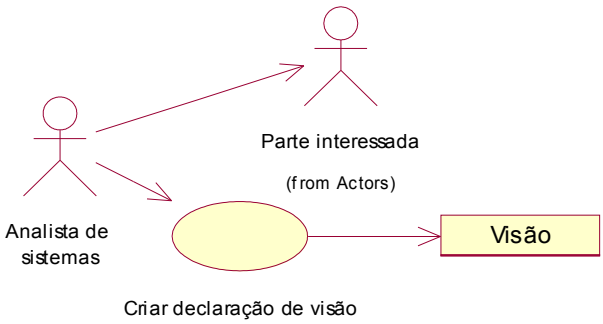
Os principais artefatos deste conjunto são o modelo de domínio e o glossário. O modelo de domínio corresponde a um modelo UML que captura a essência do contexto do negócio e do domínio no qual o sistema irá residir. Este modelo é descrito através de objetos de domínio possuidores de operadores e entidades.

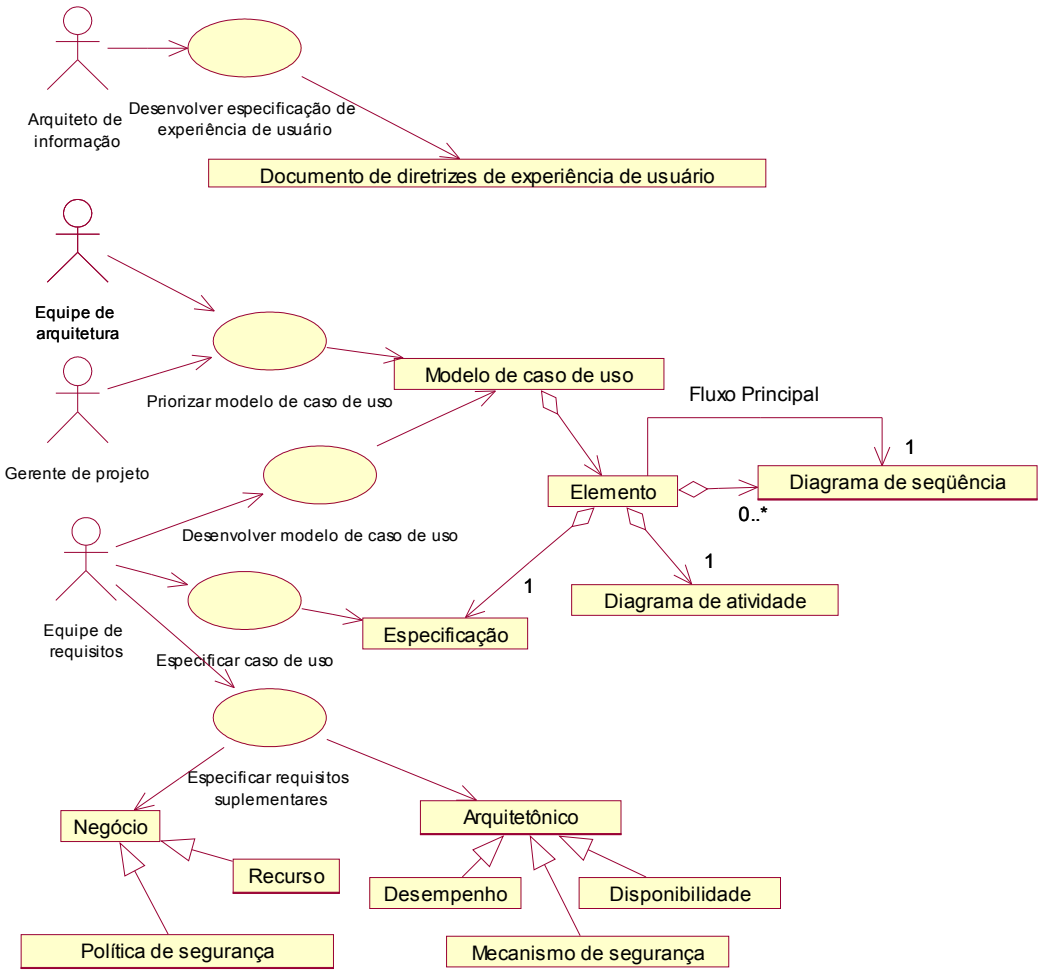
Já o glossário irá capturar definições de conceitos e termos-chave para as discussões do negócio, e deverá estar facilmente acessível a todas as partes interessadas no desenvolvimento. Será ele que irá evitar interpretações dúbias ou incompletas pelos membros da equipe.



3.2.4 Conjunto de requisitos

Os principais artefatos deste conjunto são as visões, o documento de diretrizes de experiência do usuário, o modelo de casos de uso(e seus respectivos diagramas de atividade e seqüência), e especificar os requisitos suplementares (de negócio e arquitetura).





Um requisito é uma declaração do que o sistema deve executar. Ele deverá especificar uma solução de sistema independente da arquitetura do sistema, apenas do ponto de vista de como gostaríamos que o sistema se comportasse.

O documento mais importante deste conjunto é o documento de visão. Será ele que irá determinar a meta global do sistema a ser desenvolvido e o contexto em que está definido. Geralmente ele é criado antes do projeto mas deve permanecer estável ao longo de todo o ciclo de vida do projeto.

Já os casos de uso possuem um modelo UML, meramente simbólico, e sua especificação com termos do domínio para que possam ser bem compreendidos por todas as partes interessadas. Serão eles os principais elos entre o domínio e o mundo técnico do sistema.

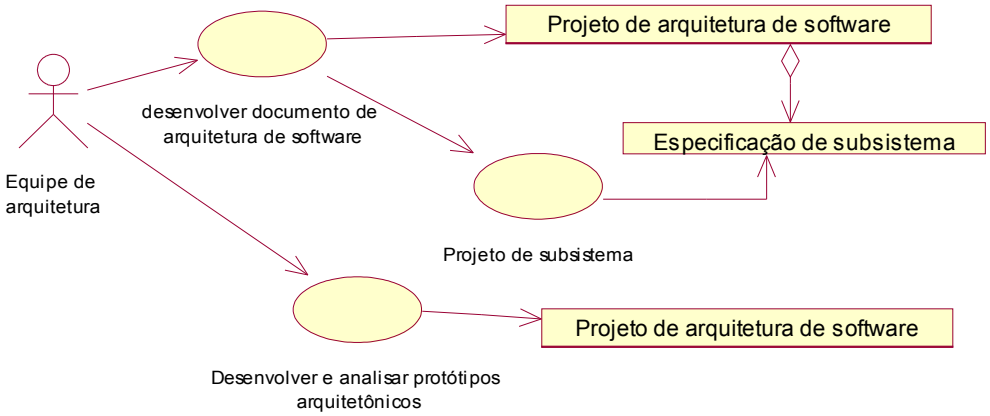
No caso específico de aplicações Web, o documento de experiência do usuário ganhou notável importância. Este documento irá definir a aparência destinada para a aplicação. Ele faz uma combinação entre o técnico e o criativo, na medida em que descreve mecanismos de navegação ao mesmo tempo que folhas de estilo e esquemas de cores. Maiores detalhes serão apresentados na seção “Experiência do Usuário – UX”.

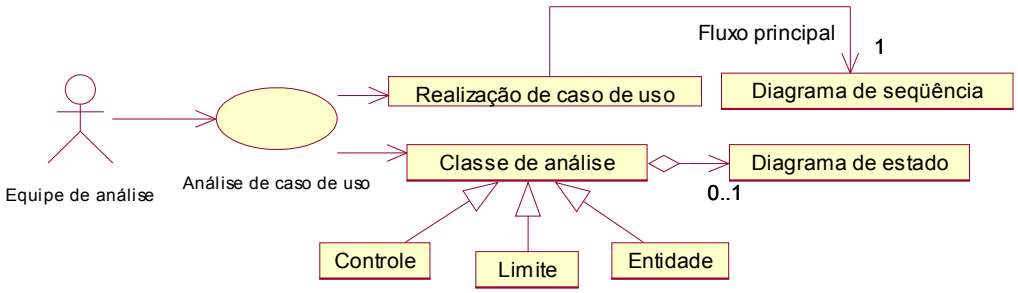
3.2.5 Conjunto de análise

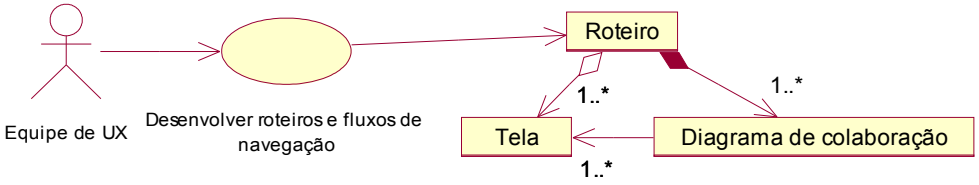
O principal artefato deste conjunto é o documento de arquitetura de software. Será a partir dele que a equipe poderá ser dividido em subsistemas e respectivas áreas de responsabilidade, e assim permitir o seu desenvolvimento assíncrono.

O documento de arquitetura será melhor confeccionado a partir do desenvolvimento e análise de protótipos arquitetônicos, o que permite testar diversas tecnologias existentes e precaver a equipe de riscos maiores no futuro.

A análise dos casos de uso irá levar a um modelo conceitual do sistema o mais independente possível da arquitetura. Este modelo irá corresponder a criação de diversas classes de análise, que provavelmente irão possuir os mesmos nomes das classes de domínio.

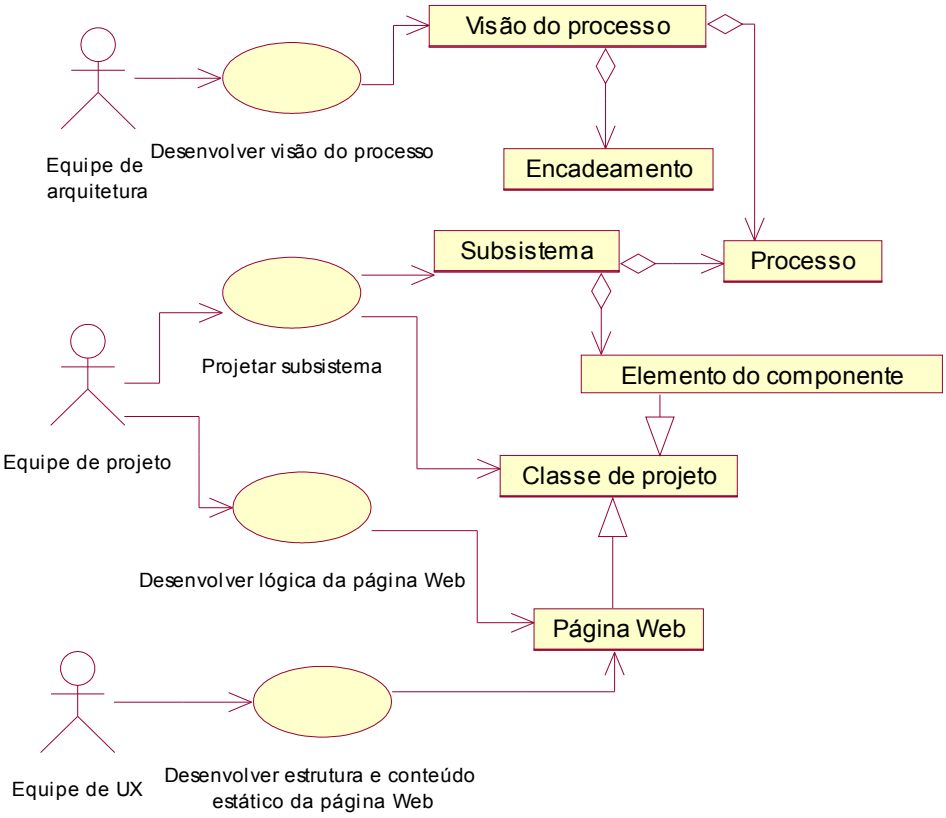

Projeto de arquitetura de software
Especificação de subsistema
desenvolver documento de arquitetura de software
Equipe de arquitetura
Projeto de subsistema
Projeto de arquitetura de software
Desenvolver e analisar protótipos arquitetônicos


Fluxo principal
1
Realização de caso de uso
Diagrama de seqüência
Equipe de análise
Análise de caso de uso
Classe de análise
Diagrama de estado
0..1
Controle
Limite
Entidade


Roteiro
1..*
1..*
Equipe de UX
Desenvolver roteiros e fluxos de navegação
Tela
Diagrama de colaboração
1..*

3.2.6 Conjunto de projeto

Este fluxo de trabalho se encarrega de aplicar a arquitetura escolhida sobre os artefatos gerados durante o fluxo de trabalho da análise. O objetivo principal do projeto é tornar o modelo de análise do sistema realizável em software.



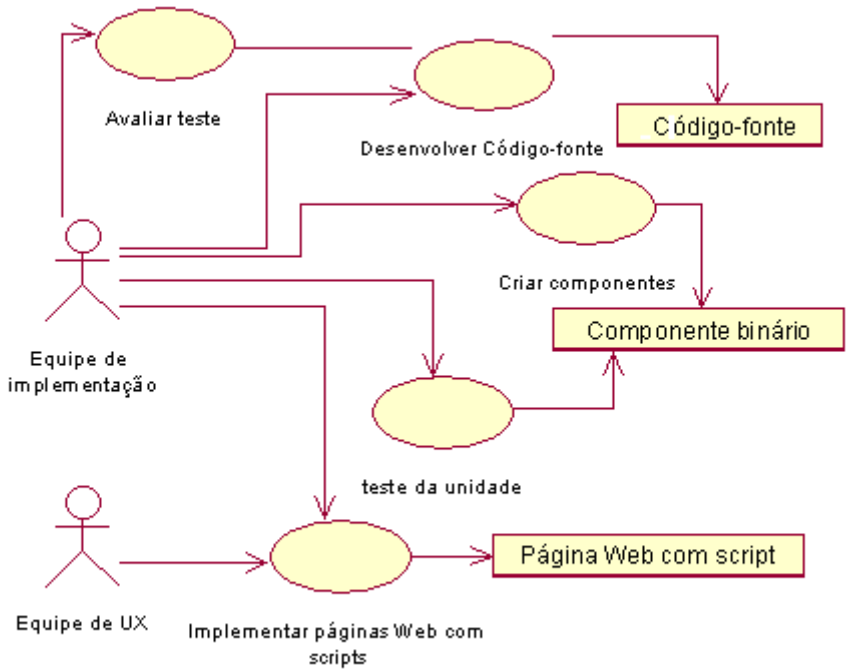
Também serão gerados os modelos de processo, que irão definir em quais camadas da aplicação – Web, cliente, dados, etc – os ciclos de vida do objeto irão residir.

Também há preocupação com a definição de classes ao nível de projeto, as assinaturas de seus métodos e as associações entre elas para satisfazer o comportamento do negócio desejado.

Além disso, ocorre a colaboração entre a equipe de experiência do usuário com a equipe de projeto para se definir como estarão dispostas as páginas Web

3.2.7 Conjunto de implementação

Durante o fluxo de trabalho de implementação os artefatos gerados durante a fase de projeto serão aplicados através das ferramentas de desenvolvimento (geralmente editores e compiladores). Chegamos à fase da codificação e do teste, que irá gerar os componentes binários do sistema.



3.2.8 Conjunto de teste

O fluxo de trabalho do conjunto de teste se baseia na avaliação dos artefatos executáveis gerados durante o fluxo de trabalho de implementação.

Muitas vezes desprezado, este fluxo de trabalho requer bastante esforço da equipe para que seja de fato eficiente. Basta lembrar que realizar testes sobre os artefatos executáveis sem o auxílio de uma ferramenta de teste automática fatalmente fará a equipe identificar apenas os mais óbvios erros.

Isto ainda pode piorar, na medida que a equipe esteja com um ciclo de desenvolvimento pequeno – o que tornará as alterações uma constante.

Cabe aqui mencionar os tipos de teste que podemos realizar sobre um artefato desenvolvido, sendo cada um possuidor de um enfoque particular. São eles:

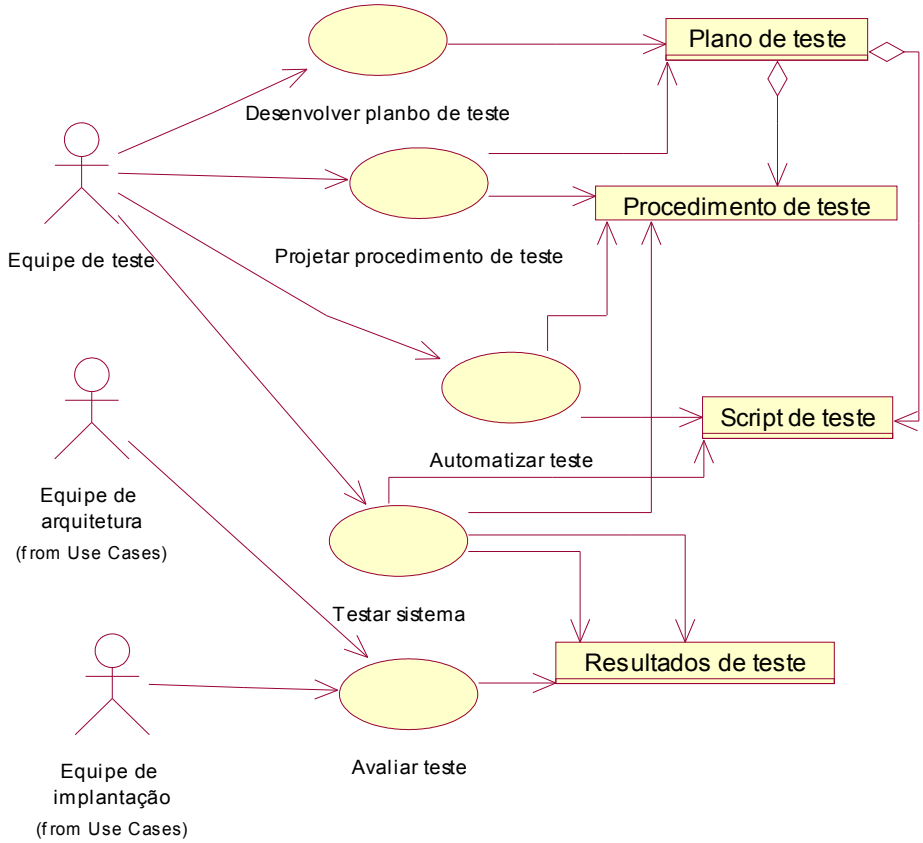
- **Teste de desempenho** – avalia a capacidade do sistema de funcionar rapidamente e sob intensa demanda.

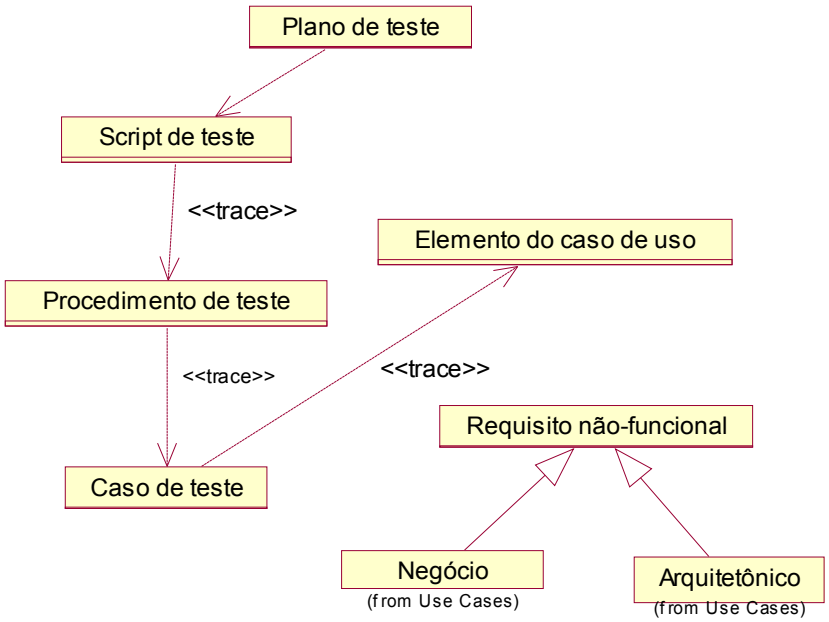
- **Teste de demanda** – estabelece o ponto de equilíbrio do sistema ou apenas as curvas de desempenho sob demanda

- **Teste funcional** – avaliam se funções específicas, definidas na especificação de requisitos, estão implementadas corretamente (ele geralmente parte da análise dos casos de uso).

- **Teste de regressão** – testa novamente o sistema em busca de alterações que pudessem ter sido efetuadas. Ele tenta evitar que alterações realizadas posteriormente não adicione falhas a outros pontos do sistema não detectados através do rastreamento dos casos de uso.

As figuras abaixo demonstram os fluxos de trabalho que envolve o conjunto de testes e também suas associações e rastreamentos:

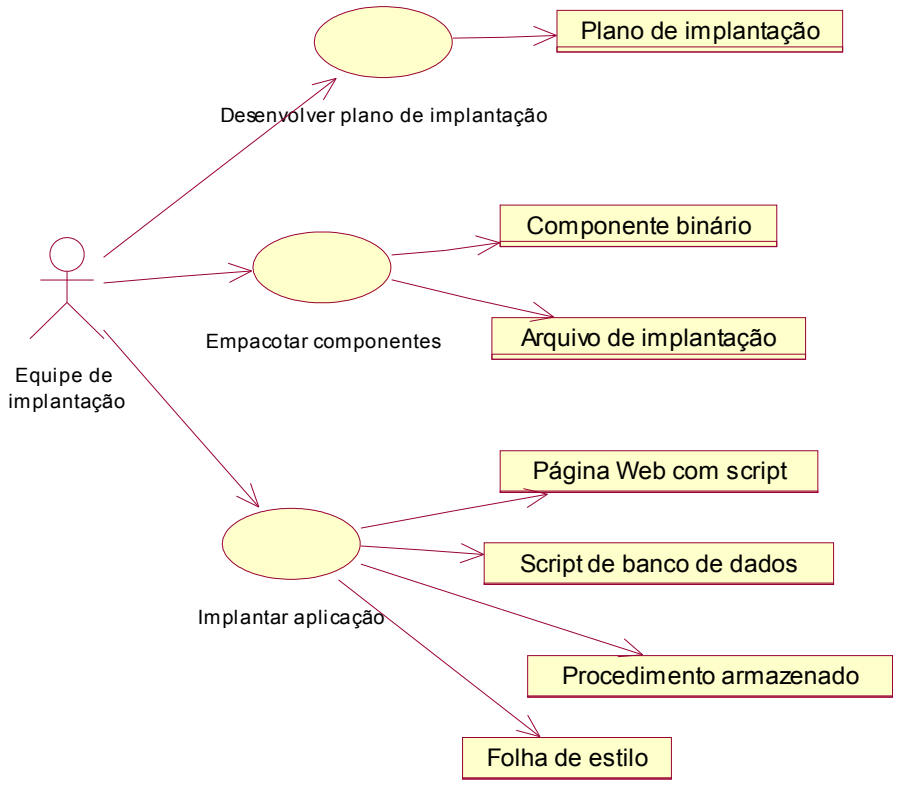

Plano de teste
Desenvolver planbo de teste
Procedimento de teste
Equipe de teste
Projetar procedimento de teste
Script de teste
Automatizar teste
Equipe de
arquitetura
(from Use Cases)
Testar sistema
Resultados de teste
Equipe de
implantação
(from Use Cases)
Avaliar teste


Plano de teste
Script de teste
<<trace>>
Elemento do caso de uso
Procedimento de teste
<<trace>>
<<trace>>
Requisito não-funcional
Caso de teste
Negócio
(from Use Cases)
Arquitetônico
(from Use Cases)

3.2.9 Conjunto de implantação

O fluxo de trabalho do conjunto de implantação tem como objetivo maior definir o plano de desenvolvimento, que deverá ser bem estudado e construído muito antes da produção do sistema.

Dependendo do contexto, o plano de desenvolvimento pode ser simples (uma aplicação Intranet) ou bastante complexo (aplicação Internet com alta demanda e segurança imprescindível). Tratar situações como migração de aplicações (um sistema herdado sendo gradualmente substituído), componentes comprados de terceiros, conexões de Internet redundantes são alguns dos exemplos que demonstram a importância de um estudo prévio e cauteloso sobre o desenvolvimento do sistema em si.



4 CAPÍTULO IV – Principais Documentos Gerados

4.1 Documento de Visão

O documento de visão representa a visão geral de todo o projeto e funciona como a origem de toda a análise do que conduzirá a equipe de desenvolvimento durante todo o processo.

O documento de visão possui alguns elementos principais. São eles:

4.1.1 Descrição do problema

O analista de sistema responsável pelo documento de visão deverá capturar a natureza do problema e seu contexto. Toda a linguagem utilizada no documento deverá ser a do domínio do problema, uma vez que, normalmente, as partes interessadas não possuem conhecimento técnico suficiente. Além disso, é essencial que elas entendam da forma mais clara possível as visões do problema.

4.1.2 Lista das partes interessadas envolvidas no projeto e suas visões do sistema

Este elemento é fundamental para que a equipe de desenvolvimento possa compreender para quem o sistema está sendo criado e quais são as suas reais necessidades e expectativas individuais. As partes interessadas podem compreender odesde os usuários finais do sistema até departamentos ou organizações inteiras.

4.1.3 Esboço do escopo do projeto

A grande parte dos projetos bem-sucedidos são aqueles que possuem escopo limitado. Nunca será possível que todos os problemas do negócio sejam resolvidos por um sistema de software. A equipe deve elencar o menor conjunto de necessidades das partes interessadas que deve ser plenamente atendido, e nunca as necessidades da equipe de desenvolvimento.

Uma idéia inicialmente discutida sobre recursos disponíveis e intervalo de tempo também contribuem bastante para um bom documento de visão.

4.1.4 Esboço da solução de software

A solução de software contida no documento de visão deverá conter somente requisitos de alto nível, que irão descrever, de modo amplo, os fragmentos de funcionalidade que se espera que o sistema possua.

Serão estes requisitos principais que irão dar origem posteriormente aos diversos requisitos detalhados que a solução de software exigirá. A análise e documentação de requisitos será discutida mais adiante.

Quanto à arquitetura, o documento de visão deverá ser completamente independente. Porém, no caso de plicações Web, é comum encontrarmos termos associados a elas, e que inseridos em alto nível de abstração são de fácil entendimento para toda a equipe e partes interessadas.

No caso específico de aplicações Web, o documento de visão deverá incluir seções que tratem de:

4.1.5 Segurança

Deve-se esboçar o nível de segurança do sistema de modo geral e apontar a(s) área(s) onde ela assume um papel mais crítico. Pode-se tentar estabelecer o impacto que a segurança deverá ter sobre funcionalidades e desempenho do sistema.

4.1.6 Ambientes de cliente suportados

Deve-se descrever o ambiente do cliente de destino, mencionando a versão do navegador suportada, o sistema operacional, recursos de tela e a importância de dar suporte a outros clientes na primeira e futuras versões. Todas estas informações deverão ser escrutinadas no documento de requisitos.

Ao fim, vale ressaltar que o documento de visão, por ser confeccionado com grande facilidade de compreensão, é bastante usado para se obter financiamento. Por isso, e para garantir finaciamenos futuros, o documento deve conter visões realistas, e nunca futuristas ao ponto de não se concluirem dentro do prazo e modo estipulados e causar frustração. Estipular critérios de sucesso efetivos e tangíveis de fato também em muito contribuem para sua boa aceitação.

4.2 Documento de Requisitos

Uma especificação de requisitos compreende uma coleção de artefatos (documentos, registros de banco de dados, modelos) que irão descrever sem ambigüidade um sistema de software a ser criado.

O documento de requisitos contém informações sobre sua finalidade, versão, seus colaboradores e finalmente uma lista de requisitos específicos do sistema. Ele geralmente deve estar acessível a todas as pessoas envolvidas no projeto, sendo que para uma aplicação Web a infraestrutura local já permite sua publicação em rede.

Um requisito representa uma restrição que o sistema deve observar. Ele possui como finalidade expôr da maneira mais objetiva possível um comportamento ou uma propriedade que o sistema irá possuir.

Um bom requisito é aquele que possa ser verificado pela equipe de teste quando produzido. Para tal, é necessário criar um requisito objetivo e com critério de aprovação claramente definido.

De modo geral, os requisitos podem ser divididos em funcionais e não-funcionais:

- **Requisito Funcionais** : são aqueles que descrevem uma ação que o sistema deve executar

- **Requisitos Não-Funcionais** : são aqueles que descrevem fatores como usabilidade, desempenho, rotina, segurança, hardware e implantação.

Os requisitos de usabilidade se referem aos aspectos de interface entre usuário e sistema, como a aparência dos padrões de interface, número máximo de cliques que o usuário pode fazer para concluir uma função do sistema, restrições a elementos HTML, etc.

Os requisitos de desempenho geralmente estão relacionados ao tempo que o sistema leva para executar suas funções. Normalmente eles são declarados restringindo o tempo máximo de carregamento de página no navegador do cliente em situações normais de uso do sistema.

Os requisitos de rotina tentam precaver e organizar tarefas que fatalmente a equipe irá enfrentar, como a manutenção, a correção e o backup do sistema. Eles podem vir declarados com sentenças que definam, por exemplo, o tempo máximo por semana que a equipe terá para executar estar tarefas.

Os requisitos de segurança irão especificar os níveis de acesso ao sistema, sejam eles atores humanos ou sistemas externos. Eles também podem definir controles de autenticação, criptografia, detecção de intrusão e auditoria.

Os requisitos de hardware geralmente definem o hardware mínimo exigido para que se implemente o sistema, tendo a preocupação de definir um conjunto para cada camada do sistema: cliente, apresentação, entidade e dados.

Os requisitos de implantação servem para restringir o modo como o sistema deve ser instalado, atualizado e acessado pela equipe. Determinar um processo único para execução destas tarefas é importante para evitar erros de versões ou paralisações inesperadas de partes do sistema.

O documento de requisitos, como todos os demais criados durante o processo de desenvolvimento de software interativo e incremental, deverá sofrer alterações ao longo do tempo. Quanto melhor for o trabalho da equipe de requisitos, menos freqüentes serão as modificações.

4.2.1 Coletando requisitos

A equipe de requisitos será a responsável por coletar os requisitos exigidos pelo sistema. A participação de todos os seus membros é fundamental neste momento, pois neste momento ocorre a udança de foco entre o mundo do problema (o seu domínio) e o campo da solução (o mundo do sistema de software). Será ao realizar esta transição que algumas informações podem ser negligenciadas ou até mesmo erroneamente transformadas.

É importante que a equipe de requisitos possua como membro pelo menos um representante da comunidade de usuários ou das partes interessadas e um membro técnico da equipe de desenvolvimento.

Cada requisito deverá possuir uma identificação única dentro do documento, sendo o seu conteúdo auto-explicativo. Ao longo do projeto, cada elemento do modelo deverá atender a pelo menos um desses requisitos, para que sua existência seja justificada.

4.2.2 Priorizando requisitos

Antes que os requisitos possam ser utilizados em qualquer fase do processo de desenvolvimento, eles devem ser priorizados. Esta tarefa é atribuída às equipes de gerenciamento do projeto e de arquitetura.

Neste momento, o engenheiro de software deve assumir compromissos entre funcionalidade e tempo para produzir. Os requisitos classificados como os de maior prioridade seriam aqueles que viabilizam a funcionalidade do sistema como um todo. Os de prioridade média englobariam os requisitos que são altamente desejáveis mas podem ser retardados para um segundo momento. Já os requisitos de baixa prioridade representariam aqueles que são desejáveis, mas opcionais.

Uma boa fonte de orientação no momento de priorizar requisitos é consultar o documento de visão e indagar a importância de sua funcionalidade perante a visão original do sistema.

Como visto anteriormente, o documento de requisitos é bastante útil para se capturar e priorizar requisitos de desempenho, de hardware, de implementação e de usabilidade – os chamados requistos não funcionais.

Porém, esta técnica ainda é um pouco deficiente para capturar os requisitos funcionais em sua totalidade, uma vez que eles demandam uma dinâmica entre atores – usuários e/ou sistemas externos - e o sistema. Para que estes requisitos sejam capturados com uma riqueza maior de detalhes devemos utilizar os casos de uso.

Um caso de uso consiste em uma declaração textual da interação entre um ator e o sistema. Todo caso de uso possui uma descrição textual de um cenário de utilização do sistema, contendo a entrada de dados por parte do ator e o sistema, em resposta, exibindo uma saída observável.

Todo caso de uso possui um cenário denominado cenário principal. Ele poderá possuir diversos cenários alternativos, que geralmente irão capturar a interação ator / sistema quando ocorrem exceções durante o cenário principal.

Um caso de uso deve expressar o que o sistema deve fazer, sem se preocupar em como irá fazer. Ele deve descrever o comportamento do sistema quando observado pelo lado externo. O esforço da equipe neste momento será focar no comportamento de entrada e saída do sistema, deixando os detalhes de arquitetura e projeto para um momento posterior.

Para uma correta declaração de caso de uso, além de sua descrição textual de cenário e de um nome que reflita objetivamente sua finalidade, ele também deverá conter:

- **Identificação Única**: fundamental para se manter o rastreamento do caso de uso ao longo do processo de desenvolvimento

- **Declaração de Meta**: uma declaração simples e concisa que resuma o objetivo do caso de uso. Bastante útil durante o rastreamento de casos de uso e durante a sua própria confecção, uma vez que a equipe poderá questionar se as ações contidas no caso de uso estão de fato contribuindo para suportarem o objetivo pré determinado.

- **Autores**: útil para entrar em contato caso alguns pontos exigirem maiores explicações e também caso se verifique algum erro.

- **Suposições**: uma declaração do estado sistema quando da entrada do caso de uso. Consiste em procedimentos que a equipe julgam verdadeiros enquanto escrevem o caso de uso

- **Condições prévias**: Diferente das suposições, elas devem descrever as pré-condições que devem ser satisfeitas antes do caso de uso se iniciar

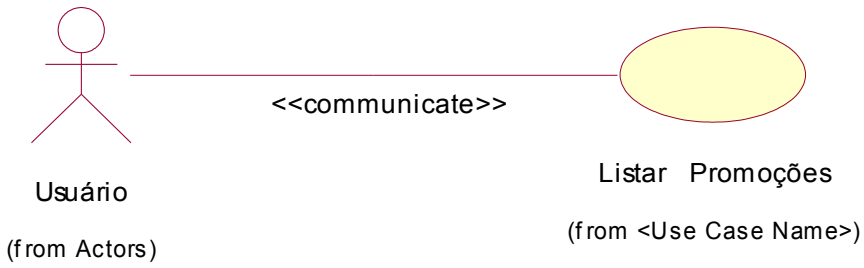
- **Condições posteriores**: devem descrever as pós-condições que devem ser satisfeitas antes do caso de uso se encerrar. Geralmente elas representam um estado do sistema

- **Questões importantes / Observações** : Indica uma coleção de itens que precisam ser definidos antes do caso de uso ser utilizado nas etapas de análise e projeto.

Como o caso de uso é utilizado para melhor descrever os requisitos funcionais, eles deverão estar escritos - assim como os requisitos – na linguagem do domínio. Como será visto a seguir, o modelo de caso de uso da UML é uma ferramenta bastante útil não só para a comunicação entre os membros da equipe, mas também entre a equipe e as partes interessadas. E neste momento a linguagem do domínio se faz imprescindível.

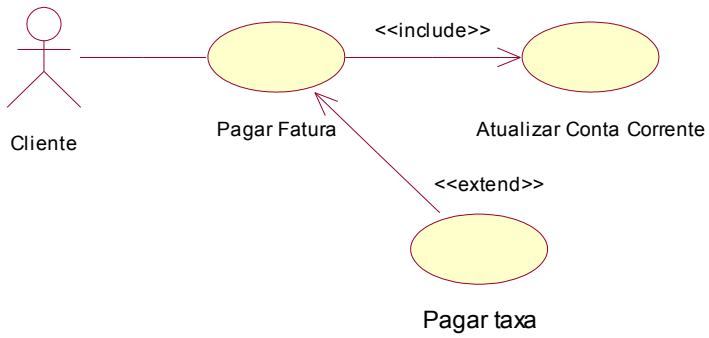
4.3 Documento de Caso de Uso

A UML nos auxilia a dar uma interpretaão gráfica dos casos de uso e dos atores que interagem com o sistema:



Os relacionamentos entre ator (no exemplo Usuário) e caso de uso (“Listar Promoções”) indicam que o ator poderá requisitar o caso de uso especificado.

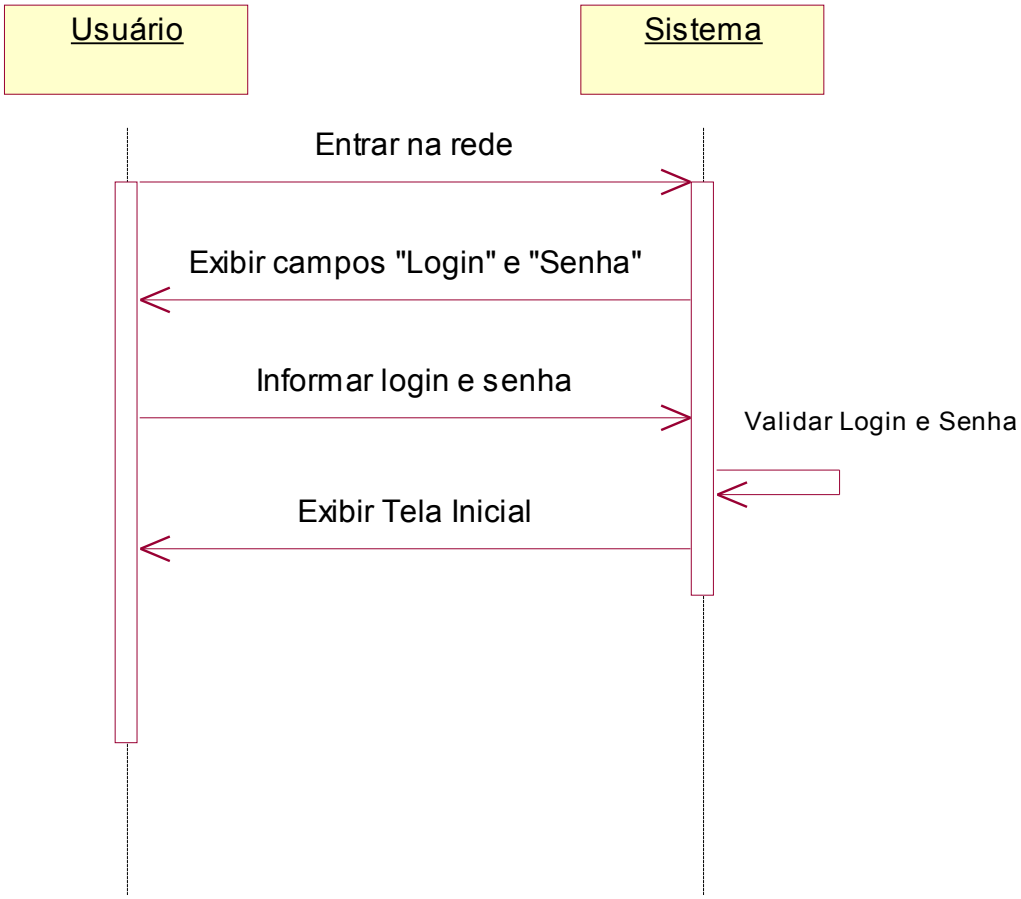
Os principais relacionamentos entre casos de uso são os do tipo <<include>> e <<extend>>. Ambos representam uma relação de dependência entre casos de uso. O relacionamento <<include>> entre casos de uso indica que um irá chamar o outro pelo menos uma vez quando requisitado. Já o relacionamento <<extend>> indica que o caso de uso pode - ou não – estender o diálogo entre com o sistema ao chamar o outro caso de uso.



O modelo de caso de uso mostra um relacionamento estrutural entre os casos de uso, sem mostrar os relacionamentos dinâmicos ou fluxos de trabalho. Para tal, a equie deve utilizar deveremos os diagramas de seqüência, atividade e colaboração.

Serão os diagramas de atividade que irão mostrar claramente todos os cenários – o principal e os alternativos – durante cada caso de uso. Eles também esclarecem a natureza dos relacionamentos <<include>> e <<extend>>. Basta notar no diagrama de atividades se é possível que o ator vá da atividade inicial à final sem chamar o caso de uso extendido.

Também é útil gerar diagramas de seqüência para cada cenário de caso de uso. Eles irão auxiliar a equipe a identificar as declarações mais importantes no texto do caso de uso. Ao se deparar com casos de uso prolixos esta atividade será muito bem-vinda.



Vale ressaltar que estes diagramas de seqüência serão constituídos apenas pelo ator e uma "instância" do sistema, cabendo às etapas de mais baixo nível a tarefa de expandir as mensagens entre os dois em mensagens concretas entre instâncias de classes determinadas.

4.4 Documento de Análise

Tendo em mãos os documentos de requisitos e de casos de uso - cada um focado, respectivamente, em requisitos estáticos e dinâmicos das regras de negócio -, a equipe de análise já se encontra em condições de desenvolver um modelo de análise do sistema.

O modelo de análise do sistema será composto por classes e colaborações de classes que irão exibir os comportamentos dinâmicos detalhados nos casos de uso e nos requisitos.

As classes irão representar objetos do domínio do negócio ou do campo do problema, tais como carrinho de compras, pedido, artigo, produto, etc.

É recomendável que o nível de abstração do modelo de análise seja tal que as mesmas classes possam ser definidas independente da arquitetura do sistema. Na prática, porém, sabemos que a arquitetura começa a surgir com importância neste momento do processo e algumas observações serão consideradas.

A análise deve enfocar os requisitos funcionais do sistema, ignorando temporariamente suas restrições. Desta forma, iremos garantir que todos os requisitos funcionais estejam implementados em algum lugar do sistema.

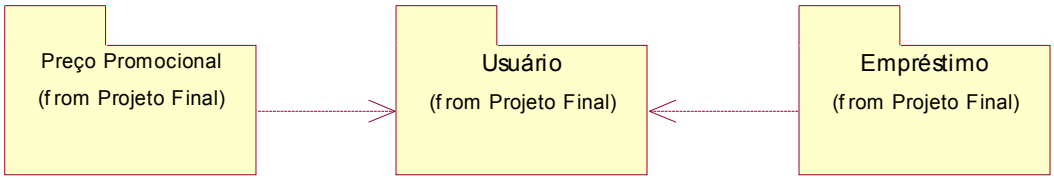
4.4.1 Estrutura

A atividade inicial da equipe de análise será criar a hierarquia de pacotes do modelo de análise. Um pacote em UML representa uma amostra do modelo, de tamanho pequeno suficiente para que uma pessoa possa entender seu significado e e finalidade no modelo.

Os pacotes irão conter elementos do modelo, como classes, diagramas, componentes, etc. As classes do interior do pacote poderão ser públicas ou privadas. Serão as classes públicas as responsáveis pela interface pública do pacote.

Definidas as classes públicas que irão se comunicar com o restante do sistema, o analista ganha a liberdade de criar quantas classes privadas quiser sem afetar o restante da equipe. Já possíveis alterações em classes públicas deverão ser combinadas com todos.

O modelo de análise de nível superior deve considerar a divisão de pacotes realizada durante a atividade de definição dos casos de uso:



Mesmo sendo um bom ponto de partida, este modelo inicial pode falhar ao tentar definir a visão estrutural do sistema, que são as classes. Isto ocorre porque é provável que alguns objetos participem de muitos casos de uso e pacotes e logicamente não podem ser atribuídos a um único pacote de caso de uso.

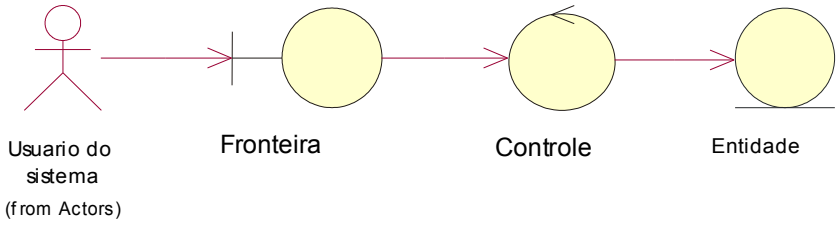
Por isso, é aconselhável que, ao partir do modelo de análise de nível superior, se vá a níveis mais baixos e consigamos dividir cada pacote principal em menores. Desta maneira, a equipe conseguirá gerenciar melhor o modelo de análise e agrupar os futuros objetos semelhantes, e não os comportamentos.

Em resumo, ao definir pacotes a equipe de análise deverá se preocupar em garantir que eles sejam:

- **Compreensíveis**;
- **Coesivos** – as classes do pacote devem se agrupar naturalmente;
- **Conjugados livremente** – as classes se relacionam mais com classes do próprio pacote do que com classes externas ao pacote;
- **Hierarquicamente superficiais** – poucos níveis hierárquicos reduzem a complexidade. O limite costuma ser de dois ou três níveis.

4.4.2 Elementos estruturais

Neste momento, a equipe já identificou todos os objetos ao analisar os casos de uso. As classes podem ser classificadas como classes estereotipadas “boundary” (fronteira), “control” (controle), e “entity” (entidade):



As classes de fronteira conectam os usuários ao sistema. Em uma aplicação Web, elas são as páginas ou telas do sistema.

As classes de controle fazem o mapeamento para os processos que distribuem a funcionalidade do sistema.

As classes de entidade representam tudo o que é persistente no sistema, como o banco de dados. As entidades são compartilhadas e possuem um cliclo de vida fora de qualquer uso do sistema, considerando que as instâncias de controle possuem um ciclo de vida bem definido.

As classes surgidas a partir da análise dos casos de uso de uma aplicação Web serão classificados como classes de entidade e de controle. As classes de fronteira serão identificados durante a análise de experiência do usuário, nosso próximo tópico.

4.5 Experiência do Usuário – UX

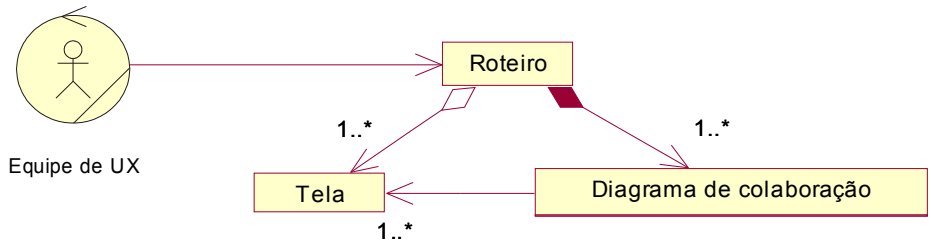
O modelo de experiência do usuário – user experience (UX) – representa um novo conceito na modelagem de sistemas e ganha especial importância nas aplicações Web.

O termo experiência do usuário descreve todas as atividades responsáveis por manter a interface com o usuário adequada ao contexto no qual se espera que o sistema seja executado. A equipe de UX é responsável por criar a aparência da aplicação, determinar as rotas de navegação principais das páginas Web do sistema e gerenciar a estrutura do conteúdo das páginas.

A equipe de UX irá desenvolver artefatos que incluirão desde a definição de cores e fontes até o posicionamento de informações na tela durante o fluxo de navegação. Algumas dessa informações serão arquitetonicamente significativas, enquanto outras terão efeito puramente cosmético.

4.5.1 Artefatos do modelo UX

O modelo de UX é um modelo separado dos demais porque é uma visualização completa do sistema do ponto de vista de suas telas. As propriedades arquitetonicamente significativas das telas e seus relacionamentos de navegação são os elementos principais do modelo UX e merecem maior atenção.



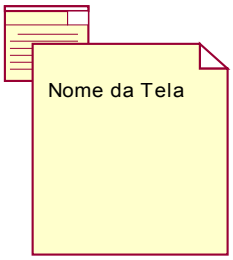
O conceito de tela não pode ser confundido com página Web, que são os mecanismos que criam e produzem as telas. Desta forma, a tela representa estritamente o que é apresentado ao usuário.

Cada tela de uma aplicação Web deverá possuir um conteúdo estático – geralmente fornecido pelo sistema de arquivos do servidor - e um conteúdo dinâmico – construído a partir da interação com os componentes do lado do servidor.

Além destas propriedades, a tela também possui:

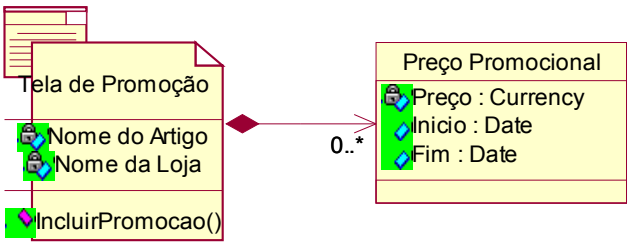
- Nome e descrição;
- Campos de entrada e controles que aceitem a entrada de dados do usuário;

A representação em UML de uma tela se dá através de uma classe estereotipada “screen” definida pela WAE (a ser definida adiante):



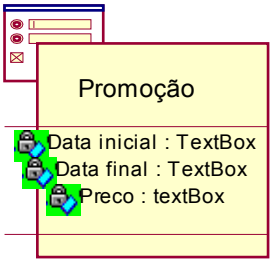
Como em qualquer modelo, nem todas as propriedades de uma tela são adequadas para serem capturadas por ele, que representa uma simplificação, uma abstração da realidade.

O conteúdo estático da tela e a sua estrutura (“layout”) não devem ser capturados pelo modelo. Já o conteúdo dinâmico é arquitetonicamente significativo e deve ser representado pelo modelo da equipe de UX. A representação da associação entre a tela e o conteúdo dinâmico é igual a todas as classes UML. No exemplo abaixo, a tela de promoção possui como conteúdo dinâmico o nome do artigo, da loja e uma lista contendo linhas de preço promocional:



4.5.2 Entrada de dados do usuário

Os formulários de entrada de dados são muito importantes para o modelo de UX. Eles são modelados como como uma classe estereotipada “<<input form >>”. Os campos de entrada são capturados como atributos e podem ser classificados como caixas de texto, checkbox, lista de seleção, etc:



Através destas classes estereotipadas, os artefatos de UX – os cenários de roteiro e os caminhos de navegação - podem ser construídos de maneira apropriada.

Um cenário de roteiro tem como objetivo expressar um uso típico do sistema através dos olhos do usuário. As telas utilizadas para cada cenário de roteiro podem ser acessadas mais de uma vez e com um conjunto de dados dinâmicos. Um diagrama de seqüência envolvendo as telas somado a pequenas descrições textuais executam bem esta tarefa.

Já o mapa de caminhos de navegação, que irá expressar todos os caminhos legais e esperados ao longo do sistema, irá conectar as telas a seus formulários de entrada e outras telas através de associações UML.

4.6 Documento de Arquitetura

O planejamento da arquitetura do sistema visa definir um conjunto de restrições que serão aplicados nas equipes de projeto e implementação durante a transformação dos modelos de requisitos e análise em sistema executável.

Desse modo, é compreensível que o documento de arquitetura tente abranger diversos pontos de vista do sistema. Cada ponto de vista – ou visão – será uma projeção dos modelos da aplicação Web.

Num primeiro momento esta atividade pode parecer redundante, uma vez que cada modelo é criado a partir de uma abstração diferente do sistema. Porém, o documento de arquitetura irá focar apenas no que é “arquitetonicamente significativo”. Um requisito arquitetonicamente significativo é aquele que possui um profundo impacto no desenvolvimento do resto do sistema.

As visões irão compreender, assim, os requisitos, os modelos de projeto UML –visão lógica e de processos – a realização do sistema (quais serão os mecanismos de comunicação? Há limites de desempenho?) e as estratégias e estruturas de teste.

4.6.1 Atividades da Arquitetura

As principais atividades da arquitetura são:

- Priorizar os casos de uso arquitetonicamente significativos;
- Definir e documentar uma arquitetura candidata;
- Definir a estratégia de reutilização

Ao priorizar os casos de uso, deve-se buscar aqueles que apresentam maior risco, que envolvam uma tecnologia nova e/ou desconhecida para a equipe ou que exija um alto grau de desempenho. Desta maneira, a equipe estará evitando surpresas desagradáveis durante o desenvolvimento da aplicação.

Ao definir uma arquitetura candidata para uma aplicação Web, deve levar em consideração os requisitos significativos e também os aspectos culturais da empresa destinada a realizar o sistema. Assim, as experiências anteriores e as habilidades dos profissionais envolvidos são determinantes para a decisão da arquitetura candidata.

4.6.2 Padrões arquitetônicos de alto nível

Um padrão arquitetônico tem como objetivo nos ajudar a compreender as formas básicas de camadas de apresentação em uma aplicação Web.

Os três padrões mais comuns são:

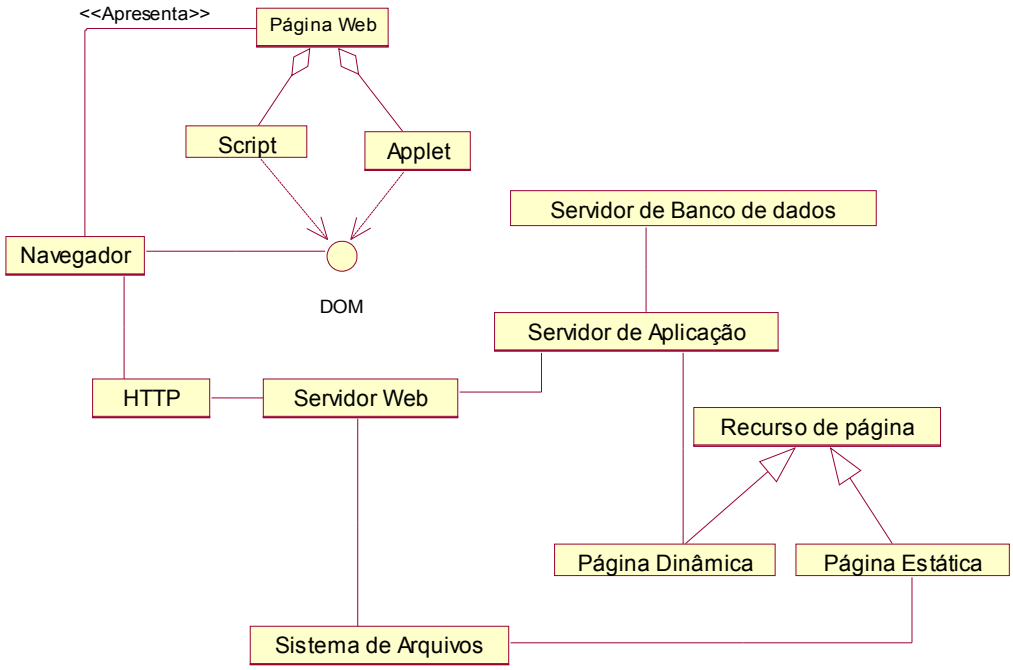
- **O cliente Web magro** – utilizado basicamente para aplicações baseadas na Internet. Se caracteriza por um pouco controle sobre a configuração do cliente. Toda a lógica do negócio é executada no servidor.
- **O cliente Web gordo** – uma parte significativa da lógica do negócio é executada no cliente, utilizando HTML, applets e controles ActiveX. A aplicação ainda utiliza o protocolo HTTP para comunicação com o servidor.

- **Produção Web** – Caracterizado quando o browser age como um dispositivo de produção e de contêiner para um de sistema de objetos distribuídos. Além do HTTP, a aplicação pode utilizar os protocolos IIOP e DCOM.

Considerando os padrões mais comuns – Clientes Web magro e gordo -, os principais componentes de ambas arquiteturas são:

- Navegador: realiza a interface com usuário, solicita páginas Web, HTML, scripts e applets;
- Servidor Web: recebe as solicitações por páginas Web, estáticas ou do servidor. Responsável por formatar a página HTML e enviá-la ao navegador do cliente;
- Protocolo HTTP: realiza a conexão entre cliente e servidor;
- Página estática: página Web que não processa nada do lado do servidor;
- Página dinâmica: páginas Web que processam algo do lado do servidor, sendo implementadas como páginas com script (ASP, JSP, PHP, etc)e processadas por um servidor de aplicação;
- Servidor de aplicação: responsável por executar o código das páginas do servidor;
- Servidor de banco de dados: mantém o estado do sistema persistente;
- Sistemas de arquivos: responsável por gerenciar arquivos HTML e arquivos Web com scripts

A figura abaixo mostra os relacionamentos entre os componentes básicos dos padrões de arquitetura Cliente Web magro e gordo:



4.7 Projeto

A etapa do projeto é aquela onde a abstração do negócio começa a se materializar na realidade de um software.

Com base no modelo de análise, no modelo de UX e no documento de arquitetura, a equipe de projeto terá como finalidade refinar o modelo de análise, de forma que ele possa ser implementado com os componentes que obedecem às regras da arquitetura.

Desse modo, as classes se tornam mais definidas, com atributos totalmente qualificados – nome e tipo – e métodos contendo assinaturas perfeitas. Nesta fase de projeto surgem as classes auxiliares e de implementação. Ao final, temos um modelo que pode ser mapeado diretamente para o código, que é de fato o vínculo entre as regras de negócio abstratas e a realidade do software.

Além disso, durante a fase de projeto a equipe também deverá separar os objetos entre camadas - cliente, servidor e outras que possam ser identificadas – e definir interfaces com o usuário por meio das páginas Web.

Ao particionar os objetos em suas camadas, precisamos conhecer quais camadas estão disponíveis para os objetos. Estas e outras respostas serão dadas pela arquitetura adotada para o sistema. Por exemplo: caso o padrão arquitetônico de cliente da aplicação Web seja o de cliente Web magro, este não é capaz de suportar objetos definidos na camada do cliente. Desta forma, todos os objetos deverão estar presentes na camada de servidor.

Neste momento, nota-se a importância das página Web. A página Web atua como um contêiner generalizado de interface com o usuário, unindo o navegador com o restante do sistema. Para a modelagem de aplicações Web, então, é vital capturar as páginas e representá-las com a mesma preocupação concedida às classes e componentes do restante do sistema.

Partindo deste conceito – a página Web é um objeto como outro qualquer – a modelagem de páginas Web levanta questões referentes a como representar a dinâmica de scripts que rodam no servidor e outros scripts que rodam no cliente. Ou seja, temos comportamentos completamente distintos dependendo da camada onde a página estará sendo processada.

A simples utilização dos conceitos da UML não consegue representar de forma adequada esta nova realidade. Por isso, os criadores da UML criaram uma maneira de estender a linguagem de forma controlável: a extensão de aplicação Web (WAE).

4.7.1 WAE – Web Application Extension

A extensão de aplicação Web permite que se represente páginas Web e outros elementos significativos do ponto de vista arquitetônico do modelo, juntamente às classes normalmente utilizadas para todos os sistemas de software.

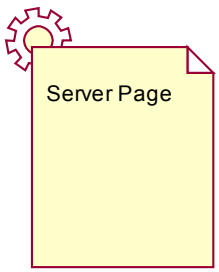
A seguir, serão apresentados os principais estereótipos, valores com tags e restrições que compõem esta extensão da UML.

4.7.1.1 Visão Lógica

A WAE define três estereótipos de classe principais:

- Página do Servidor

Representa uma página Web dinâmica, que contém scripts que são executados pelo servidor, interagindo com recursos do lado do servidor, como bancos de dados, componentes de lógica do negócio, sistemas externos, etc.

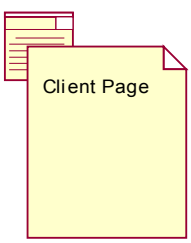


As páginas de servidor só se relacionam com objetos no servidor

- Página do Cliente

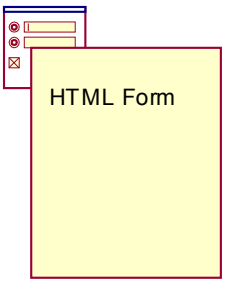
Representa uma página Web formatada em HTML com uma mistura de dados, apresentação e lógica. Elas são apresentadas pelos navegadores, podendo conter scripts interpretados também pelo navegador.

As páginas do cliente podem se associar a outras páginas do cliente e do servidor.

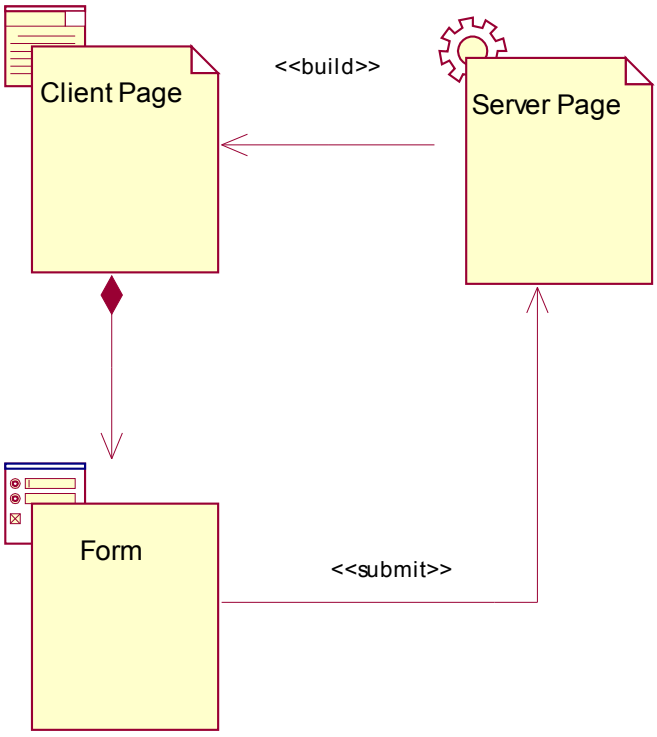


- Formulário HTML

Representa uma coleção de campos de entrada que são parte de uma página do cliente, sendo mapeada diretamente para a tag <form> da HTML.



As relações básicas entre os elementos estereotipados WAE podem ser observadas abaixo:



Os principais estereótipos de associação da WAE são:

<<link>>	Relacionamento entre uma página no cliente e outra página no cliente ou página no servidor. Representa o elemento <a> da HTML. Possui um valor com tag de nome “parameters”, que transmite seus valores junto com a solicitação HTTP.
<<build>>	Relacionamento entre uma página do servidor e uma página do cliente. Representa a construção da saída HTML por parte da página do servidor.
<<submit>>	Relacionamento entre um formulário HTML e uma página do servidor. Todos os atributos de campo do formulário são enviados ao servidor, junto com a solicitação HTTP.
<<redirect>>	Representa um comando de uma página do cliente para solicitar outro recurso, podendo ser páginas do cliente ou servidor
<<object>>	Relacionamento entre uma página do cliente e uma classe lógica, geralmente um applet ou controle ActiveX. Representa os elementos HTML <object> e <applet>
<<include>>	Associação entre uma página do servidor e páginas do servidor ou cliente. Indica que a página incluída é processada e seu conteúdo é utilizado pela página pai.
<<forward>>	Relacionamento entre uma página do servidor e outra página do servidor ou cliente. Representa a delegação de processamento de uma solicitação do cliente de um recurso para outra página do lado do servidor.

4.7.1.2 Visão de Componente

A visão de componente de um modelo UML irá representar os arquivos distribuídos que compõem o sistema durante a sua execução. A WAE define dois estereótipos de componentes abstratos, que serão representados por estereótipos dependentes da linguagem utilizada (<<JSP>>, <<ASPX>>, <<ASCX>>, <<XML>>, etc).

Os dois estereótipos definidos são:

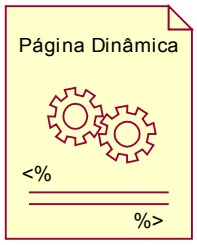
- Página Estática

Representa um recurso que pode ser solicitado diretamente pelo navegador do cliente. Ela é produzida diretamente do sistema de arquivos para o cliente.

Ela não pode implementar componentes lógicos que serão executados no servidor.

- Página Dinâmica

Representa um recurso que pode ser solicitado pelo navegador do cliente. Quando estiver apontado pela associação <<forward>>, ocorre o processamento do lado do servidor. Os resultados podem alterar o estado do servidor para construir algum HTML a ser transmitido.

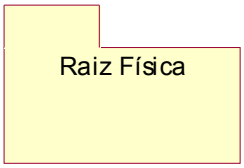


A página dinâmica deve implementar apenas uma página do servidor

A WAE também define um pacote de componentes:

- Raiz Física

Representa uma abstração de uma hierarquia de arquivos que contém recursos disponíveis à solicitação. Ele nos remete diretamente para um diretório do sistema de arquivos do servidor Web. Para isto, seus valores de tag “Web Location” (ex: “www.projetofinal.com.br”) e “Project Directory” (ex: “www.projetofinal.com.br/LucasAlenquer”) são utilizados.

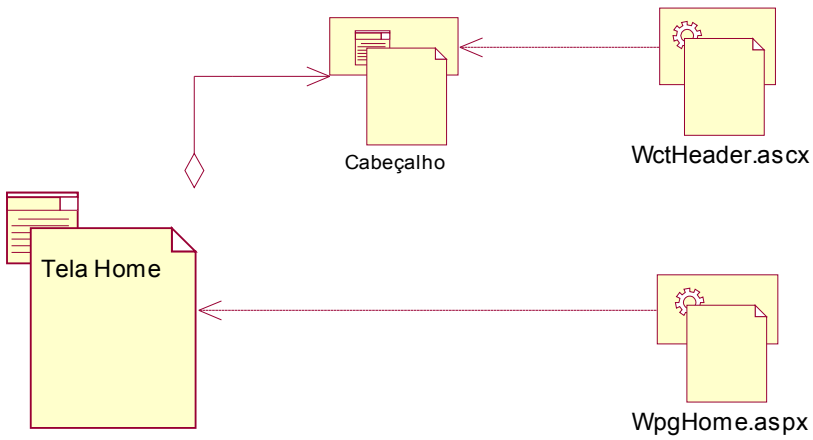


4.7.2 Mapeando o projeto para o modelo UX

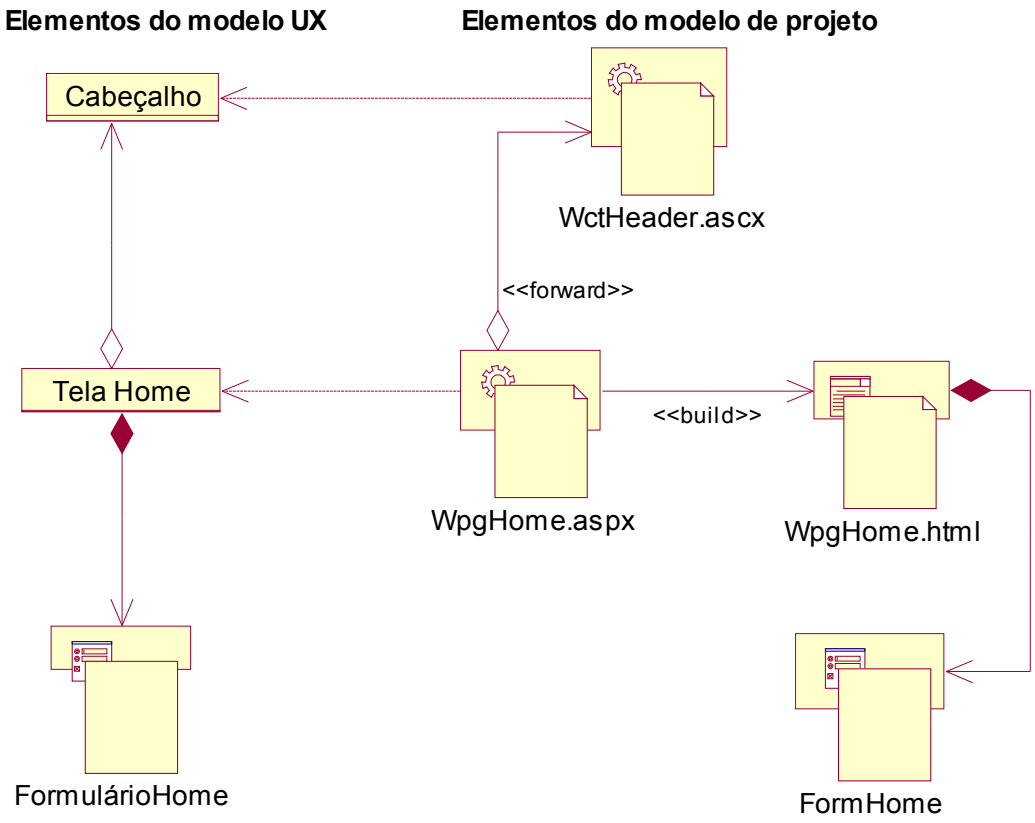
Formalizar a experiência do usuário em um modelo UML facilita o entendimento do que se quer realizar por parte da equipe criativa responsável pelo “layout” das telas e também permite que a equipe de engenharia de software mantenha um mapeamento direto entre os casos de uso e os modelos de projeto para o modelo UX.

Este mapeamento se dá a partir de diagramas de classe contendo telas de UX e classes do modelo de projeto ligadas através de relacionamentos de dependência.

No exemplo abaixo, a tela Home estabelece um relacionamento de agregação com a tela de cabeçalho, e sua plena visualização depende das páginas no servidor WpgHome.aspx e WctHeader.ascx:



Refinando a modelagem teremos o seguinte cenário:



Desta forma permitiremos que as duas equipe possam trabalhar de forma relativamente independente. A equipe de UX pode desenvolver o HTML e supor que o acesso a determinado conteúdo estará disponível. Já a equipe de engenharia irá trabalhar com esboços das páginas Web bastante simples, apenas para testar o seu funcionamento.

Neste capítulo estão reunidos os artefatos gerados durante a realização de uma iteração sobre o sistema Livraria, onde novos requistos são introduzidos e o projeto será orientado a objetos.

5 CAPÍTULO V – Caso - Sistema Livraria - Artefatos Gerados

5.1 Documento de Visão - Sistema Livraria -

5.1.1 Descrição do problema

Uma livraria com lojas espalhadas pela cidade necessita realizar um correto gerenciamento sobre seus processos, principalmente os que interferem no preço praticado dos artigos à venda e, conseqüentemente, nos seus estoques.

Neste contexto, é necessário armazenar e acessar dados sobre as transações que os funcionários das lojas estão envolvidos, como a definição/alteração de promoções nos preços praticados e empréstimo de artigos para clientes, bem como limitar o acesso de alguns funcionários a partes do sistema cujas tarefas não lhe competem.

5.1.2 Partes interessadas envolvidas:

Funcionários: serão eles os reais utilizadores do sistema.

Visão do Sistema : O sistema deverá se comportar como um automatizador de procedimentos antes executados manualmente e armazenados sob forma de papel.

5.1.3 Esboço da solução de software:

É importante ressaltar que a solução de software a ser desenvolvida **compreende uma nova iteração** sobre um sistema já existente e em fase de evolução. Desta forma, alguns requisitos e seus posteriores casos de uso já foram implementados e os requisitos abaixo mencionados serão adicionados ao sistema.

Os requisitos básicos os quais o sistema se destina são:

- Gerência de Preço Promocional

O usuário poderá escolher um determinado artigo e programar - ou cancelar - o período e o valor de um preço promocional a ser praticado em uma determinada loja.

- Gerência de Empréstimo

O usuário escolherá um determinado artigo e serão armazenados dados referentes a um empréstimo a um cliente em uma determinada loja, bem como os dados de sua posterior devolução.

O sistema deverá informar o saldo da quantidade e do valor dos artigos que um cliente possui sob empréstimo no momento.

- Cadastro de Usuário

O sistema deverá tratar da inserção e edição de informações dos usuários do sistema, que permitirão a sua entrada através de um login identificador e uma senha pessoal.

5.1.4 Esboço do escopo do projeto:

O sistema será uma aplicação ASP.NET que irá permitir que o usuário edite preços promocionais, gerencie empréstimos de artigos e realize cadastro e alteração de dados dos usuários do próprio sistema.

Para realizar tais tarefas, serão utilizados os seguintes recursos:

1. Classes ASP.NET com implementação em Visual Basic .NET;
2. Páginas Web com extensão .aspx;
3. Scripts do lado do cliente;
4. Banco de Dados SQL Server;
5. Stored Procedures , Funções e Views armazenadas no Banco de Dados determinado

O sistema será composto por uma página central, que irá conter links para as páginas que irão realizar os casos de uso desejados.

Cada página terá a responsabilidade de garantir a consistência dos dados visualizados pelo usuário através da conexão com o Banco de Dados, se utilizando de artefatos previamente armazenados (Stored Procedures , Funções e Views) para capturar as informações que desejar.
Os scripts do lado do cliente terão como finalidade básica poupar o servidor da aplicação de requisições desnecessárias, realizando tarefas tais como verificação de dados de entrada do usuário e outras que não necessitem de interação com o servidor.

- Gerência de Preço Promocional

Haverá uma página que receberá uma identificação do artigo especificado. Conhecendo o artigo, surgirá uma listagem com os valores e os períodos de validade dos preços promocionais anteriormente registrados.

Haverá um formulário para entrada de dados referentes a um novo preço promocional ou a um já existente (neste caso uma modificação de informações).

- Gerência de Empréstimo

O usuário acessará uma página contendo uma listagem com os clientes que possuam empréstimos realizados em cada loja. Cada nome de cliente, quando clicado, funcionará como um link para uma nova página que mostrará os detalhes do empréstimo do cliente e na loja especificados.

- Cadastro de Usuário

O usuário acessará uma página contendo uma listagem com os usuários do sistema cadastrados. Haverá um link para uma nova página que irá conter um formulário para entrada de dados de cadastro de um novo usuário. Caso o usuário clique sobre o nome de um usuário da listagem, a nova página que contem o formulário irá surgir com o mesmo já preenchido com os dados mais recentes do usuário escolhido.

5.2 Glossário - Sistema Livraria –

Artigo – Qualquer item que esteja à venda em qualquer uma das lojas existentes.

Cliente – Pessoa Física ou Jurídica que se envolve em qualquer tipo de operação com a empresa

Devolução – Movimento de entrada de estoque que se caracteriza quando o cliente ou funcionário retorna o(s) artigos(s) emprestado(s) à loja de origem.

Empréstimo - Movimento de saída de estoque que se caracteriza quando o cliente ou funcionário obtém um ou mais artigos sob empréstimo junto à loja de origem.

Funcionário – Qualquer pessoa que esteja contratada pela livraria em qualquer uma das lojas existentes.

Loja – Qualquer estabelecimento comercial que esteja em funcionamento sob a marca da livraria.

Preço Promocional – Preço de venda praticado para um determinado artigo em uma determinada loja, tendo um valor expresso em moeda local e um período de validade pré-definido.

Produtor – Pessoa jurídica responsável pela produção de um artigo.

Usuário – Qualquer pessoa física ou jurídica que possua acesso ao sistema da livraria e dele se valha para executar tarefas ou obter informações sobre a livraria.

5.3 Documento de Requisitos - Sistema Livraria -

Finalidade – Determinar todas as funcionalidades e restrições que o sistema deverá obedecer. É importante ressaltar que os requisitos **compreendem uma nova iteração** sobre um sistema já existente e em fase de evolução. Desta forma, alguns requisitos e seus posteriores casos de uso já foram implementados e os requisitos abaixo mencionados serão adicionados ao sistema.

Versão – 1.0

Colaboradores – Lucas Lopes Alenquer

5.3.1 Requisitos Funcionais

5.3.1.1 Gerência de Preço Promocional

Finalidade: O sistema deverá permitir que o usuário escolha um determinado artigo e possa programar - ou cancelar – a data de início, a data de término e o valor de um preço promocional a ser praticado em uma determinada loja.

Cada artigo poderá possuir um preço promocional diferente em cada loja da livraria.

As datas de início e de término deverão ser formatadas na ordem dia / mês / ano.

O preço promocional deverá ser formatado como número real com duas casas decimais e com a parte decimal separada por vírgula.

5.3.1.1.1 Requisito

O sistema deverá permitir que o usuário visualize, dentre uma lista de artigos, aquele que deseja programar um preço promocional.

Critério de sucesso: O sistema deve mostrar o nome e o produtor do artigo escolhido.

5.3.1.1.2 Requisito

O sistema deverá permitir que o usuário escolha a loja para qual deseja programar o preço promocional do artigo selecionado.

Critério de sucesso: O sistema deve mostrar uma lista com os dados das últimas programações do artigo na loja selecionada: datas de início e de término, preço de venda programado e o nome do autor da programação

5.3.1.1.3 Requisito

O sistema deverá permitir somente a entrada de preços promocionais cuja data de início(dia mês e ano) seja posterior à data corrente. Caso contrário, o usuário deverá ser informado.

Critério de sucesso: O sistema mostrará mensagem de confirmação de entrada do preço promocional e mostrará a lista com os dados das programações (datas, preço de venda programado e o nome do autor da programação) já incluindo a nova entrada.

5.3.1.1.4 Requisito

O sistema deverá permitir a entrada de quantos forem os preços promocionais programados, desde que um período não se sobreponha a outro existente.

Critério de sucesso: O sistema mostrará mensagem de confirmação de entrada do preço promocional e mostrará a lista com os dados das programações (datas, preço de venda programado e o nome do autor da programação) já incluindo a nova entrada. Caso contrário, o usuário deverá ser informado.

5.3.1.1.5 Requisito

O sistema deverá permitir que se altere o valor de um preço promocional somente enquanto sua data de entrada em vigor não tiver sido alcançada.

Critério de sucesso: O sistema deverá atualizar a lista com os dados dos preços promocionais programados (datas, preço de venda programado e o nome do autor da programação). Caso contrário o usuário deverá ser informado.

5.3.1.1.6 Requisito

O sistema deverá permitir que o usuário altere as datas de início e de término do preço promocional programado, desde que o novo perído não seja anterior à data corrente e também não seja posterior à data de início de um eventual próximo preço promocional.

Critério de sucesso: O sistema deverá atualizar a lista com os dados dos preços promocionais programados (datas, preço de venda programado e o nome do autor da programação). Caso contrário o usuário deverá ser informado.

5.3.1.1.7 Requisito

O sistema deverá permitir que o usuário cancele uma programação de preço promocional cujo período seja posterior à data corrente.

Critério de sucesso: O sistema deverá atualizar a lista com os dados dos preços promocionais programados (datas, preço de venda programado e o nome do autor da programação). Caso contrário o usuário deverá ser informado.

5.3.1.2 Gerência de Empréstimo

O sistema escolherá um determinado artigo e permitirá o armazenamento de dados referentes a um empréstimo a um funcionário em uma determinada loja, bem como o de sua posterior devolução.

O sistema deverá informar o saldo da quantidade e valor dos artigos que o cliente possui sob empréstimo no momento.

Somente funcionários da livraria poderão realizar empréstimos.

O saldo corresponde ao número de itens tomados emprestados pelo funcionário subtraído do número de itens devolvidos até o momento.
Um artigo jamais poderá ser devolvido à loja sem antes ter sido emprestado.

O movimento de empréstimo representa um decréscimo do estoque físico, enquanto a devolução representa um acréscimo ao estoque físico.

Um funcionário poderá realizar um empréstimo somente nas lojas as quais possui relação.

5.3.1.2.1 Requisito

O sistema deverá permitir que o usuário visualize uma lista contendo os nomes dos funcionários e das lojas em que realizaram empréstimos, seu respectivo saldo e o valor em moeda corrente dos itens ainda por devolver.

Critério de sucesso: As informações contidas na tela devem corresponder fielmente às contidas na base de dados.

5.3.1.2.2 Requisito

O sistema deverá permitir que o usuário escolha visualizar os detalhes de cada empréstimo separadamente, composto pelo nome do funcionário, da loja, de cada um dos artigos emprestados, o seu preço de venda e a data de cada movimento de empréstimo e devolução.

Critério de sucesso: As informações contidas na tela devem corresponder fielmente às contidas na base de dados.

5.3.1.2.3 Requisito

O sistema deverá permitir que o usuário insira o empréstimo de mais um item por ele especificado, bem como sua quantidade.

Critério de sucesso: O sistema mostrará mensagem de confirmação de entrada do novo empréstimo e mostrará a lista atualizada com os dados dos movimentos, já incluindo o novo empréstimo.

5.3.1.2.4 Requisito

O sistema deverá permitir que o usuário insira a devolução de um item por ele especificado, bem como sua quantidade.

Critério de sucesso: O sistema mostrará mensagem de confirmação de entrada da nova devolução e mostrará a lista atualizada com os dados dos movimentos, já incluindo a nova devolução.

5.3.1.2.5 Requisito

O sistema deverá permitir que o usuário visualize apenas os artigos que o funcionário ainda está por devolver.

Critério de sucesso: O sistema mostrará uma lista contendo apenas os artigos cujo saldo não sejam iguais à zero.

5.3.1.3 Gerência de Usuário

O sistema deverá tratar da inserção e edição de informações dos usuários do sistema, que permitirão a sua entrada através de um login identificador e uma senha pessoal.

Cada usuário possuirá um login composto por caracteres alfanuméricos e senha de até 7 caracteres alfanuméricos.

A senha do usuário deverá ser criptografada e então armazenada no sistema.

Cada usuário poderá informar, além de login e senha, seu nome, e-mail para contato e telefone.

Cada usuário deverá estar ligado a no mínimo uma loja.

Cada usuário deverá estar ligado a no mínimo um departamento de uma loja.

Cada usuário deverá possuir ao menos um privilégio de acesso ao sistema.

5.3.1.3.1 Requisito

O sistema deverá permitir que o usuário visualize uma lista contendo todos os usuários do sistema, indicando seus respectivos privilégios, lojas a que estão ligados e os departamentos a que pertencem.

Critério de sucesso: As informações contidas na tela devem corresponder fielmente às contidas na base de dados.

5.3.1.3.2 Requisito

O sistema deverá permitir que o usuário altere as informações de um usuário especificado.

Critério de sucesso: O sistema mostrará uma lista atualizada com os usuários do sistema, já contendo os dados alterados do usuário especificado.

5.3.1.3.3 Requisito

O sistema deverá permitir que o usuário insira as informações de um novo usuário.

Critério de sucesso: O sistema mostrará uma lista atualizada com os usuários do sistema, já contendo os dados do novo usuário.

5.3.2 Requisitos Não Funcionais

5.3.2.1 Usabilidade

O sistema não deverá se utilizar de elementos frame de HTML.

O sistema deve permitir que o usuário execute uma função do sistema em no máximo quatro cliques.

5.3.2.2 Desempenho

O sistema deverá levar no máximo 8 segundos para executar suas funções e carregar suas páginas no navegador do cliente.

5.3.2.3 Rotina

A manutenção do sistema deverá ocorrer uma vez por semana quando em funcionamento normal.

A correção de possíveis falhas deverá ocorrer uma vez por semana, porém nunca às sextas-feiras, por se tratar da véspera de fim de semana – período de maior movimento nas lojas da livraria.

O backup do sistema deverá ocorrer todos os dias, ao término do horário comercial.

A equipe deverá possuir no máximo 12 horas para executar todas as tarefas mencionadas anteriormente.

5.3.2.4 Segurança

Além do controle de acesso ao sistema através de login e senha pessoal criptografada, o sistema não irá exigir outras medidas preventivas, por se tratar de um sistema de Intranet que não estará exposto a outros sistemas externos.

5.3.2.5 Hardware

As máquinas dos clientes deverão possuir no mínimo 256MB de memória RAM e o navegador Internet Explorer instalado. O servidor de banco de dados deverá possuir no mínimo 8GB de espaço em disco.

5.3.2.6 Implantação

Todos os membros da equipe deverão seguir o mesmo processo de alteração do sistema. As atividades ocorrerão na seguinte ordem:

1- Verificar se há cópia do sistema no dia. Caso não haja, o integrante da equipe deverá criá-la no diretório de backup;

2- Copiar para o diretório de pré-alteração todos os arquivos do sistema que foram criados / modificados após a data da última cópia anterior ao dia de hoje;

3- Gerar um script com todos os objetos da base de dados que foram criados / modificados após a data da última cópia anterior ao dia de hoje;

4- Filtrar os arquivos do sistema de modo a obter apenas aqueles que compõem a apresentação do sistema;

5- Incluir todos os arquivos criados / alterados do sistema no diretório de teste e gerar o novo executável;

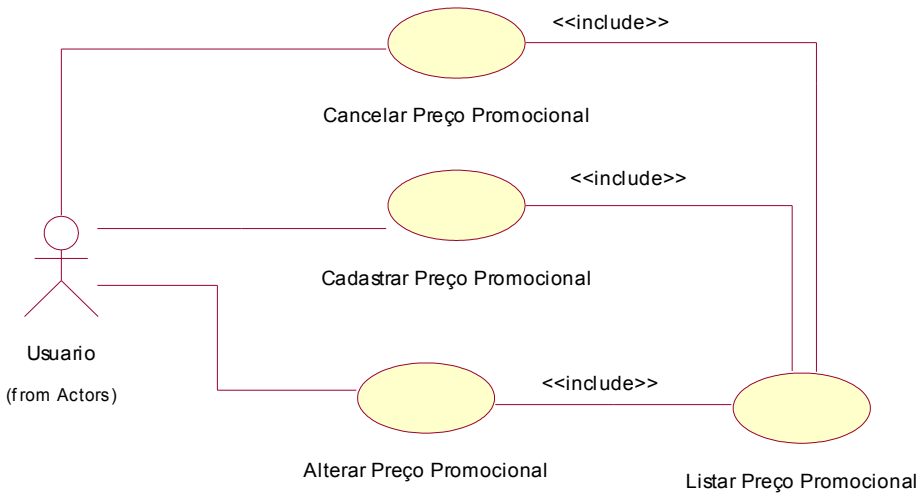
6- Enviar e-mail para a equipe contendo todos os arquivos do sistema e objetos da base de dados criados / alterados e suas respectivas funcionalidades;

7- Durante a fase de testes: caso surja algum problema, retornar com o sistema armazenado como backup e reiniciar o processo até obter o resultado desejado.

5.4 Documento de casos de uso – Sistema Livraria -

Neste documento estarão relacionados todos os casos de uso gerados para implementar os requisitos descritos para a iteração de desenvolvimento de software do sistema Livraria.

5.4.1 Gerência de preço promocional



5.4.1.1 Casos de Uso

5.4.1.1.1 Listar Preço Promocional

- **Declaração de Meta**: listar todos os preços promocionais cadastrados de um artigo numa loja especificada.

- **Autores**: Lucas Lopes Alenquer

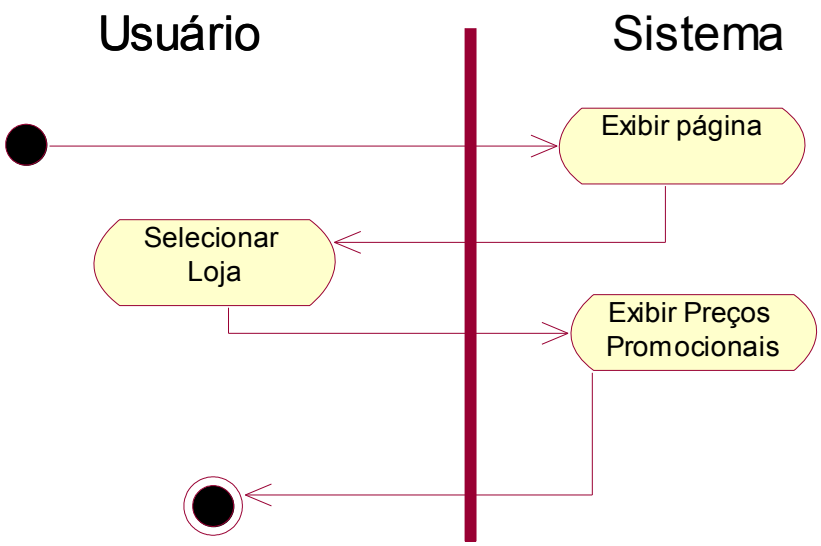
- **Suposições**: Nenhuma

- **Condições prévias**: O código identificador do artigo deve ser conhecido.

- **Condições posteriores**: Nenhuma

- **Questões importantes / Observações** : Nenhuma

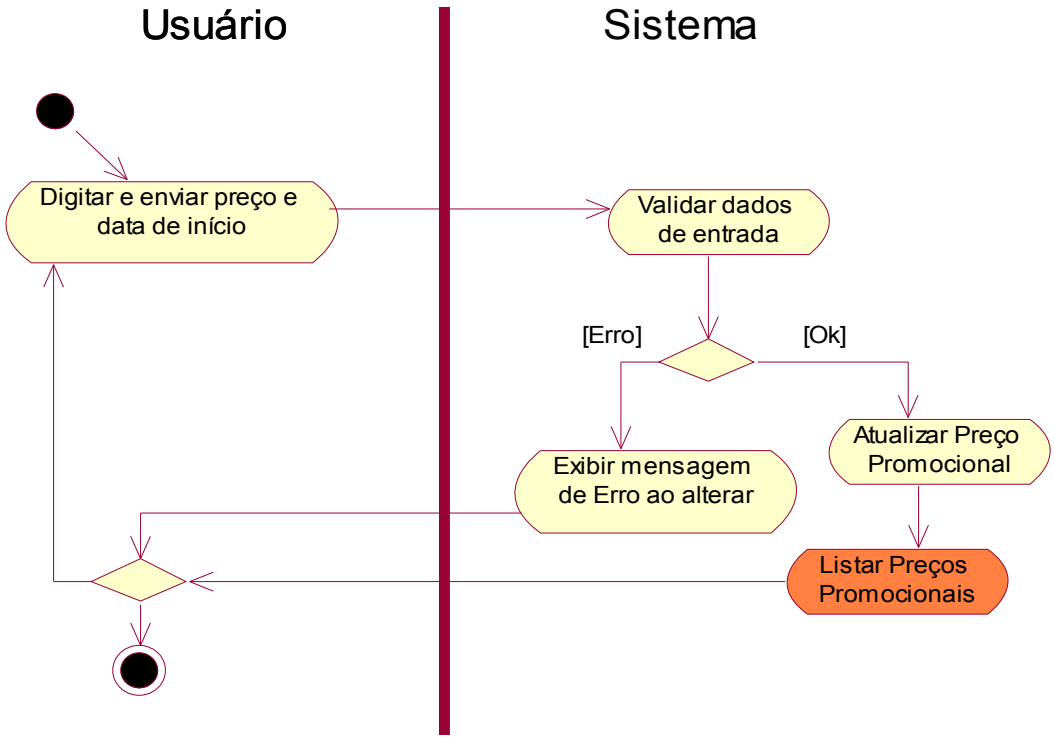
1 – O sistema exibe a página;
2 – O usuário seleciona a loja;
3 – O sistema exibe os preços promocionais do artigo na loja



5.4.1.1.2 Alterar Preço Promocional

- **Declaração de Meta**: Editar preços promocionais cadastrados de um artigo numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Deve existir ao menos um preço promocional cadastrado passível de alteração.
- **Condições prévias**: O código identificador do artigo deve ser conhecido.
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

1 – O usuário digita e envia o preço e a data inicial da promoção;
2 – O sistema valida os dados de entrada. Em caso afirmativo, o sistema irá alterar a promoção e chamar o caso de uso "Listar Preços Promocionais". Do contrário, ir para atividade 2.1
2.1 - O sistema exibirá uma mensagem de erro.



5.4.1.1.3 Cancelar Preço Promocional

- **Declaração de Meta**: Excluir preços promocionais cadastrados de um artigo numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Deve existir ao menos um preço promocional cadastrado passível de exclusão.
- **Condições prévias**: O código identificador do artigo deve ser conhecido.
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

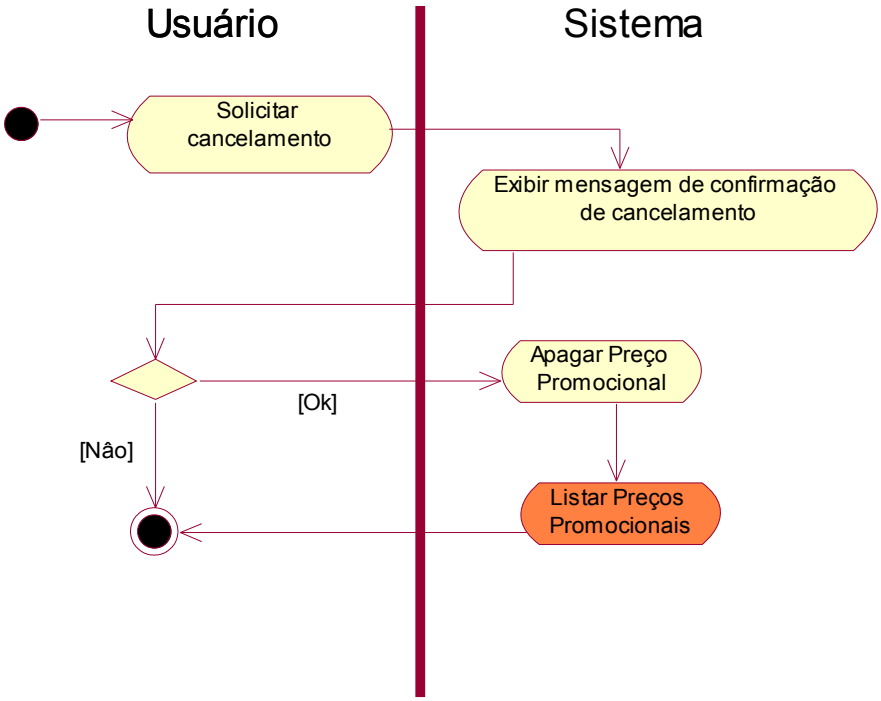
1 – O usuário solicita o cancelamento;

2 – O sistema exibe mensagem do tipo “Você deseja cancelar realmente...?”;

3 – Caso o usuário confirme, faça a atividade 3.1; do contrário, faça a atividade 3.2

3.1 – O sistema irá excluir o preço promocional.

3.2 - O sistema nada faz



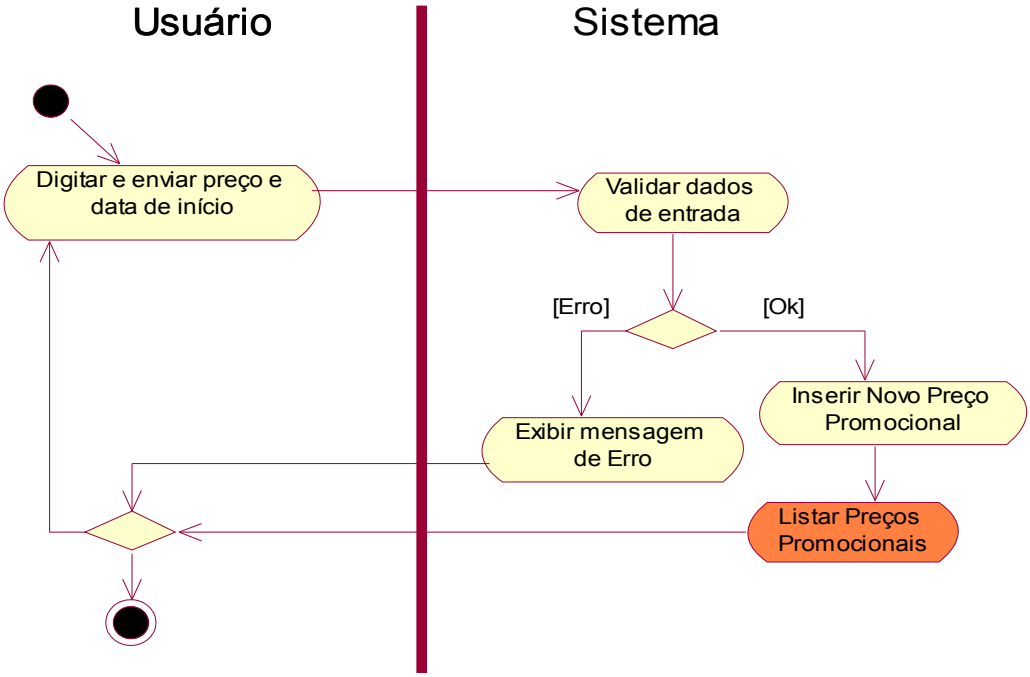
5.4.1.1.4 Cadastrar Preço Promocional

- **Declaração de Meta**: Inserir preços promocionais de um artigo numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: O código identificador do artigo deve ser conhecido.
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

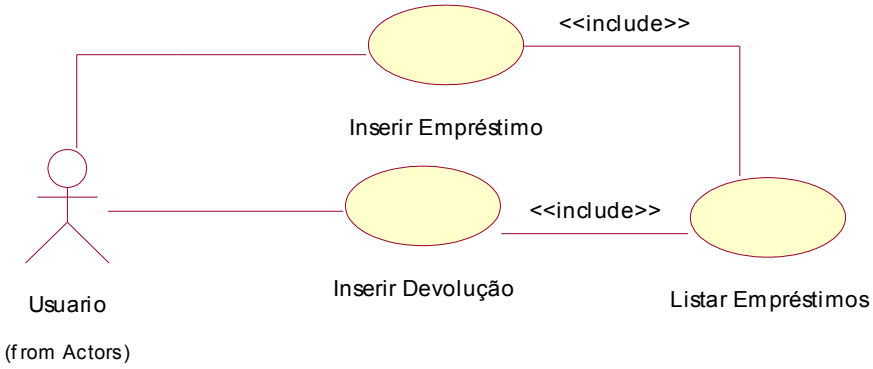
1 – O usuário digita e envia o preço e a data inicial da promoção;

2 – O sistema valida os dados de entrada. Em caso afirmativo, o sistema irá alterar a promoção e chamar o caso de uso "Listar Preços Promocionais". Do contrário, ir para a atividade 2.1

2.1 – O sistema exibirá uma mensagem de erro.



5.4.2 Gerência de empréstimo

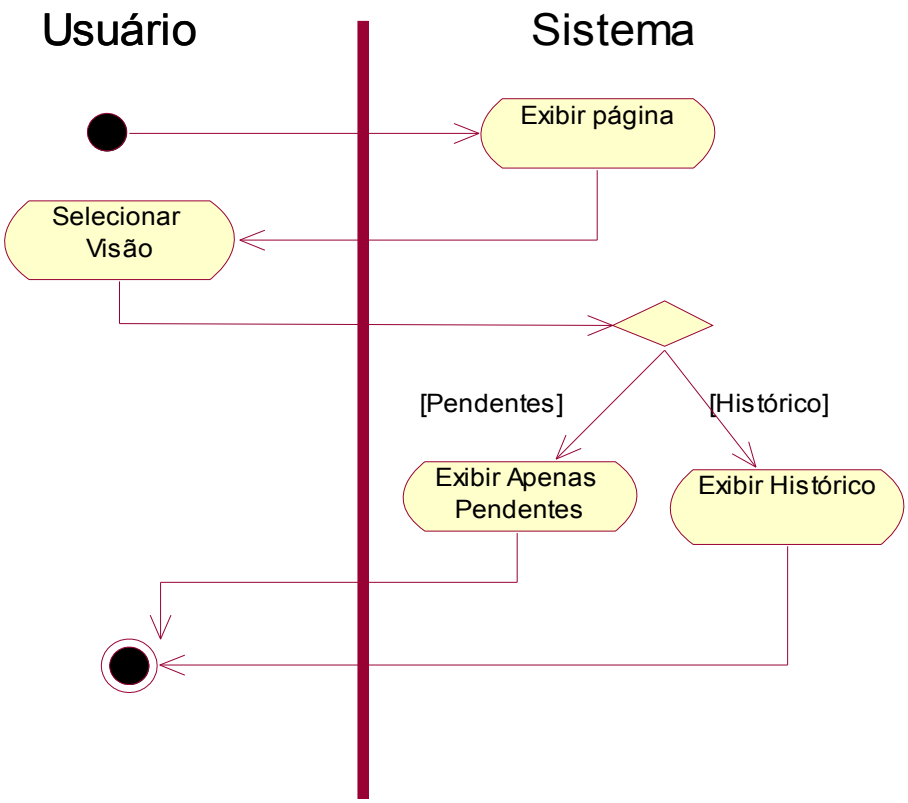


5.4.2.1 Casos de Uso

5.4.2.1.1 Listar Empréstimos

- **Declaração de Meta**: listar todos os empréstimos de artigos realizados por um funcionário numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: O código identificador do funcionário e da loja devem ser conhecidos.
- **Condições posteriores**: Nenhuma
- **Questões importantes / Observações** : Nenhuma

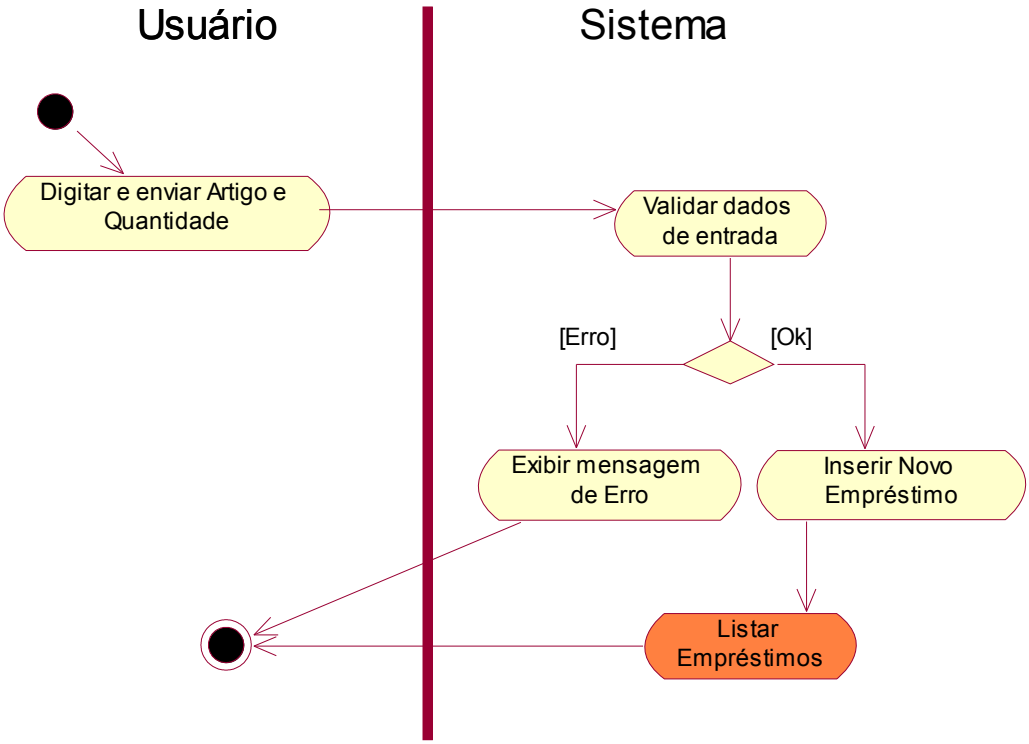
1 – O sistema exibe a página;
2 – O usuário escolhe a visão que deseja: dos empréstimos pendentes (atividade 2.1) ou do histórico com todos os empréstimos e devoluções(atividade 2.2)
2.1 – O sistema exibe lista apenas com os artigos emprestados e ainda não devolvidos;
2.2 – O sistema exibe lista com todos os empréstimos e devoluções realizados pelo funcionário na loja;


Usuário
Sistema
Exibir página
Selecionar Visão
[Pendentes]
[Histórico]
Exibir Apenas Pendentes
Exibir Histórico

5.4.2.1.2 Inserir Empréstimos

- **Declaração de Meta**: Cadastrar um empréstimo de um artigo realizado por um funcionário numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: O código identificador do funcionário e da loja devem ser conhecidos.
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

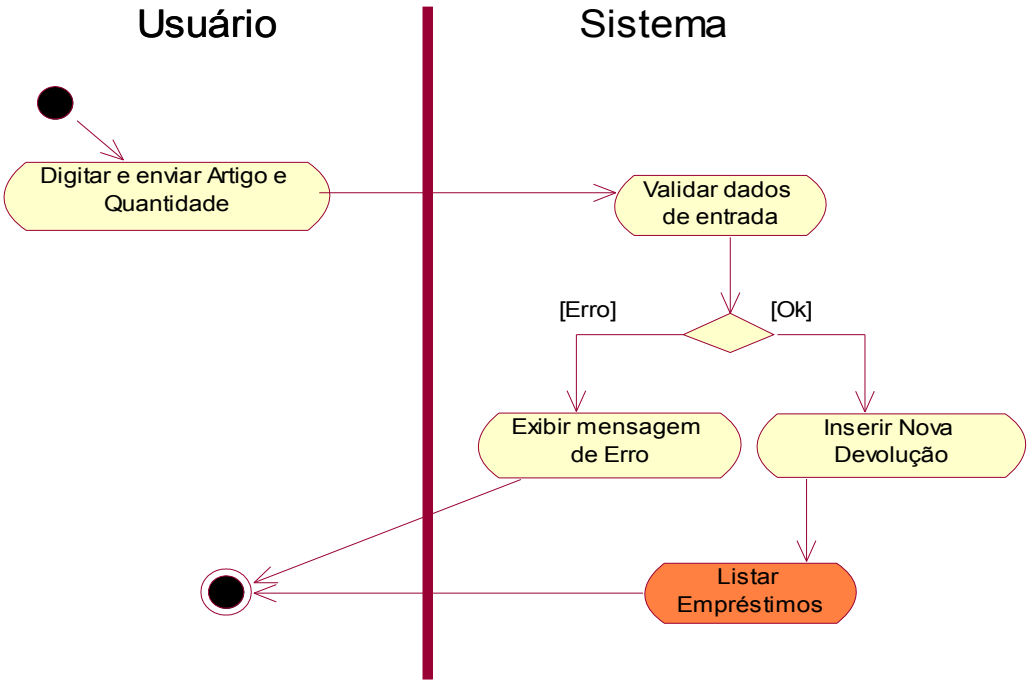
1 – O usuário digita e envia o artigo e a quantidade;
2 – O sistema valida os dados de entrada. Em caso afirmativo, o sistema irá inserir o empréstimo e chamar o caso de uso "Listar Empréstimos". Do contrário, ir para a atividade 2.1;
2.1 – O sistema exibirá uma mensagem de erro.



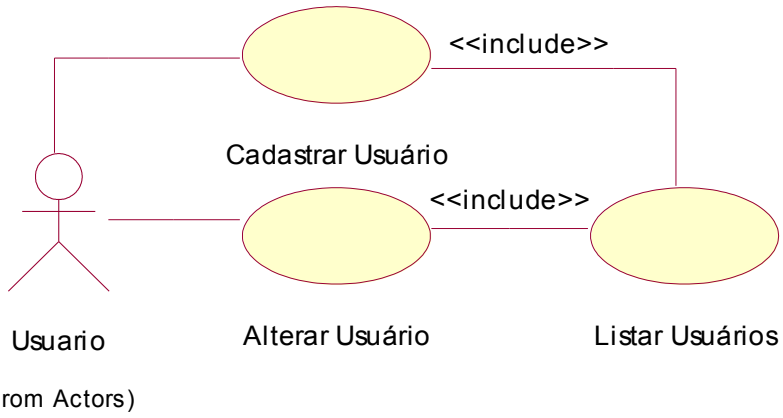
5.4.2.1.3 Inserir Devolução

- **Declaração de Meta**: Cadastrar uma devolução de artigo realizada por um funcionário numa loja especificada.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: O código identificador do funcionário e da loja devem ser conhecidos. Deve existir ao menos um empréstimo do mesmo artigo
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

1 – O usuário digita e envia o artigo e a quantidade;
2 – O sistema valida os dados de entrada. Em caso afirmativo, o sistema irá inserir a devolução e chamar o caso de uso "Listar Empréstimos". Do contrário, ir para a atividade 2.1
2.1 - O sistema exibirá uma mensagem de erro.



5.4.3 Gerência de Usuários



5.4.3.1 Casos de Uso

5.4.3.1.1 Listar Usuários

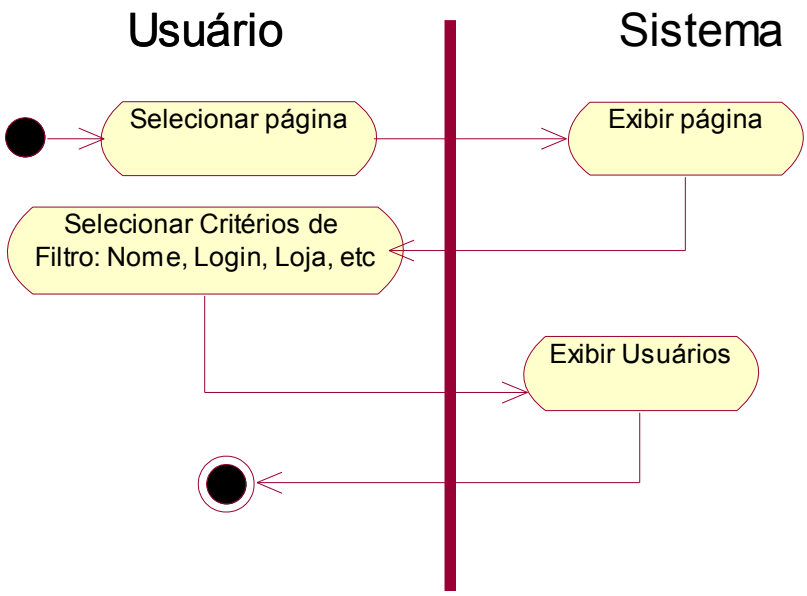
- **Declaração de Meta**: listar todos os usuários do sistema que estão cadastrados.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: Nenhuma
- **Condições posteriores**: Nenhuma
- **Questões importantes / Observações** : Nenhuma

1 – O usuário seleciona a página;

2 – O sistema exibe a página;

3 – O usuário define critérios de filtro, tais como nome, login, loja, etc;

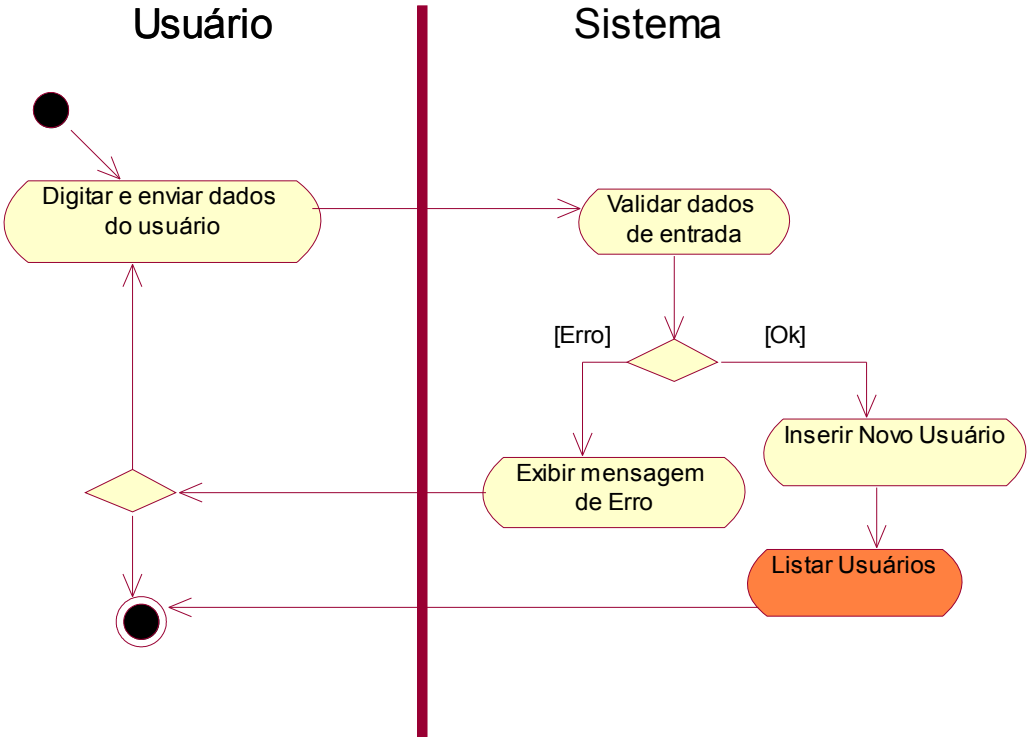
4 – O sistema exibe os usuários que se encaixam nos critérios especificados


Usuário
Sistema
Selecionar página
Exibir página
Selecionar Critérios de Filtro: Nome, Login, Loja, etc
Exibir Usuários

5.4.3.1.2 Cadastrar Usuários

- **Declaração de Meta**: Inserir um novo usuário do sistema.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: Nenhuma
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

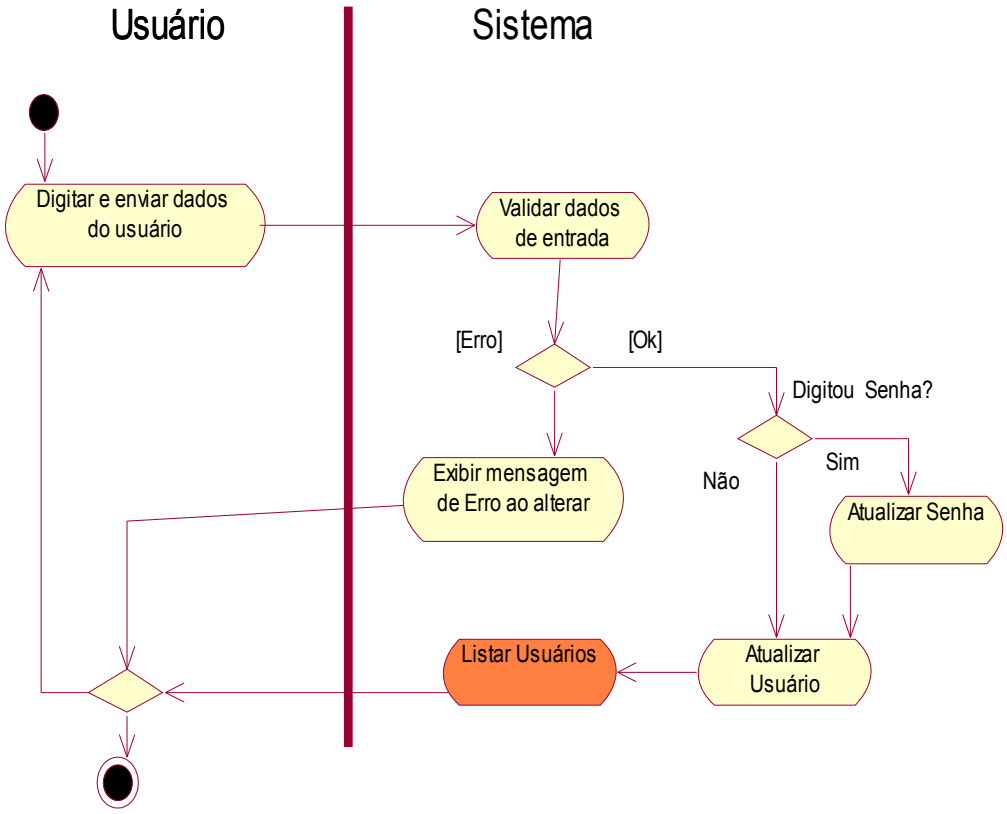
1 – O usuário digita e envia os dados cadastrais do usuário (nome login, senha, lojas, etc);
2 – O sistema valida os dados de entrada. Em caso afirmativo, o sistema irá inserir o usuário e chamar o caso de uso "Listar Usuários". Do contrário, ir para a atividade 2.1
2.1 – O sistema exibirá uma mensagem de erro.



5.4.3.1.3 Alterar Usuário

- **Declaração de Meta**: Alterar dados de um usuário cadastrado do sistema.
- **Autores**: Lucas Lopes Alenquer
- **Suposições**: Nenhuma
- **Condições prévias**: Deve existir ao menos um usuário cadastrado passível de alteração
- **Condições posteriores**: Atualização
- **Questões importantes / Observações** : Nenhuma

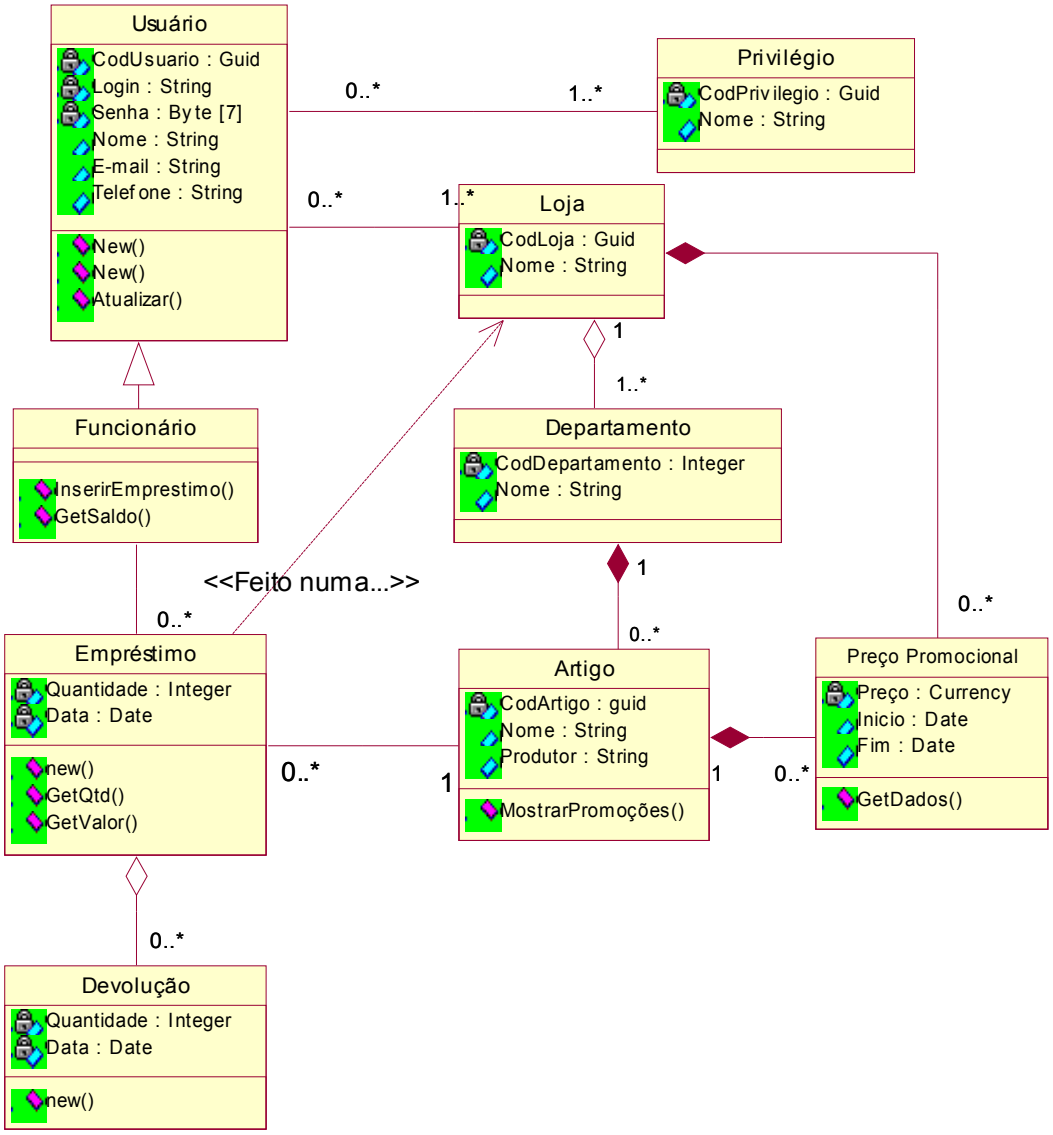
1 – O usuário digita e envia os dados do usuário;
2 – O sistema valida os dados de entrada. Se estiverem certos, irá atualizar os dados do usuário (inclusiva senha caso tenha sido digitada) e chamar o caso de uso “Listar Usuários”; do contrário, faça a atividade 2.1;
2.1 – O sistema exibirá uma mensagem de erro.



5.5 Documento de Análise

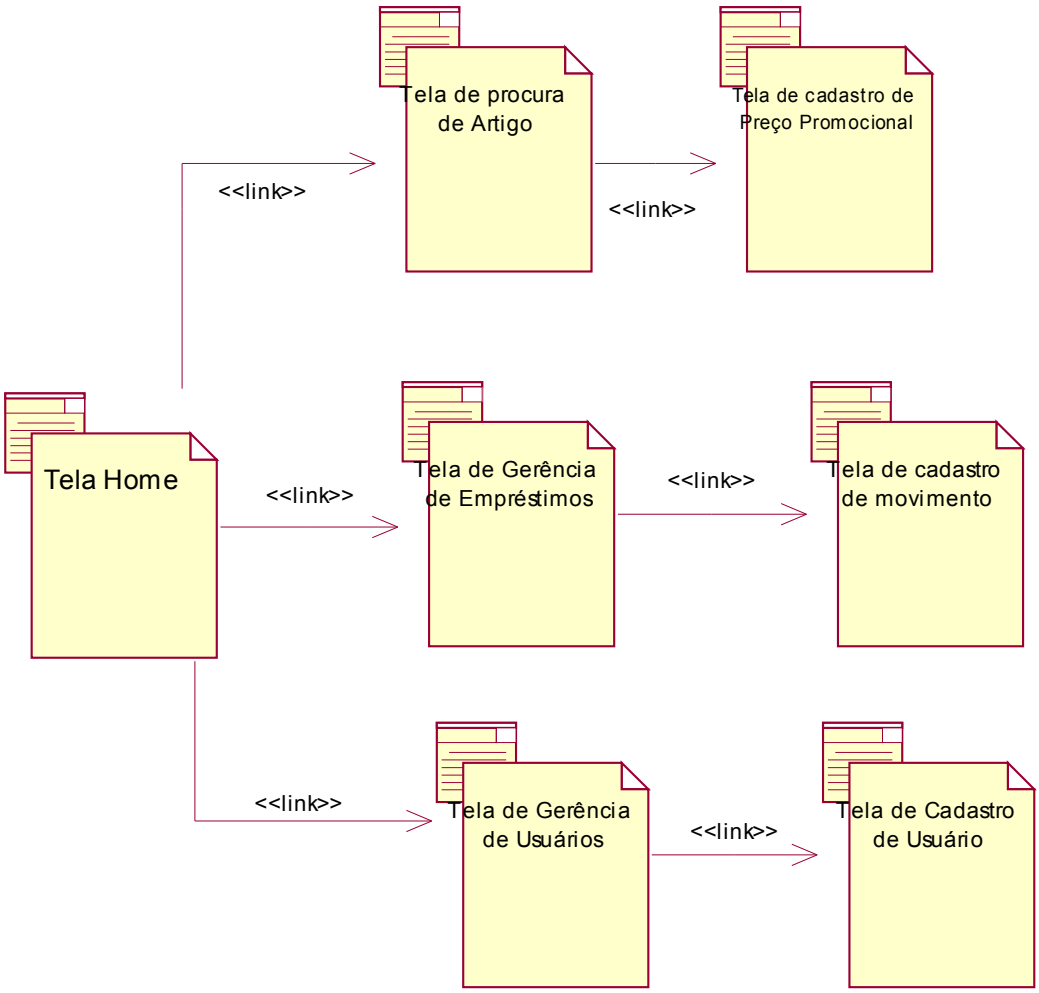
Este documento irá apresentar o esboço do diagrama de relacionamentos entre as classes do negócio envolvidas na iteração do desenvolvimento do sistema Livraria

5.5.1 Diagrama de classes



5.5.2 Caminhos de navegação

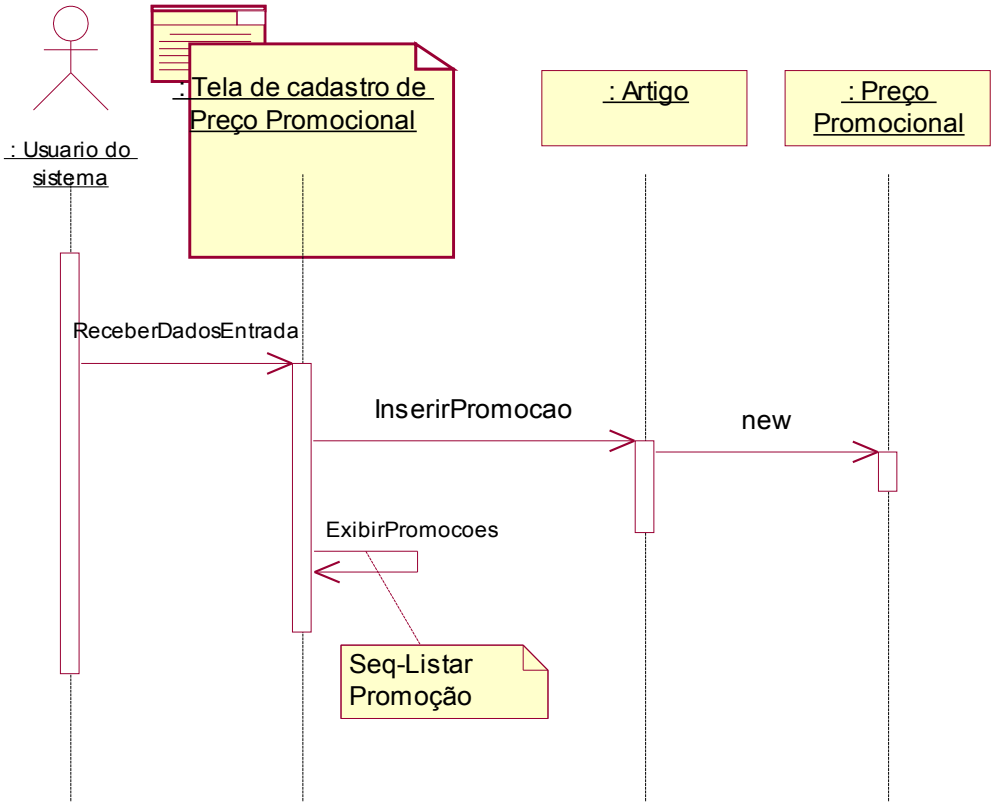
Este documento visa ilustrar os possíveis caminhos que o usuário pode tomar ao navegar para executar as funcionalidades do sistema descritas durante a iteração do desenvolvimento do sistema Livraria.



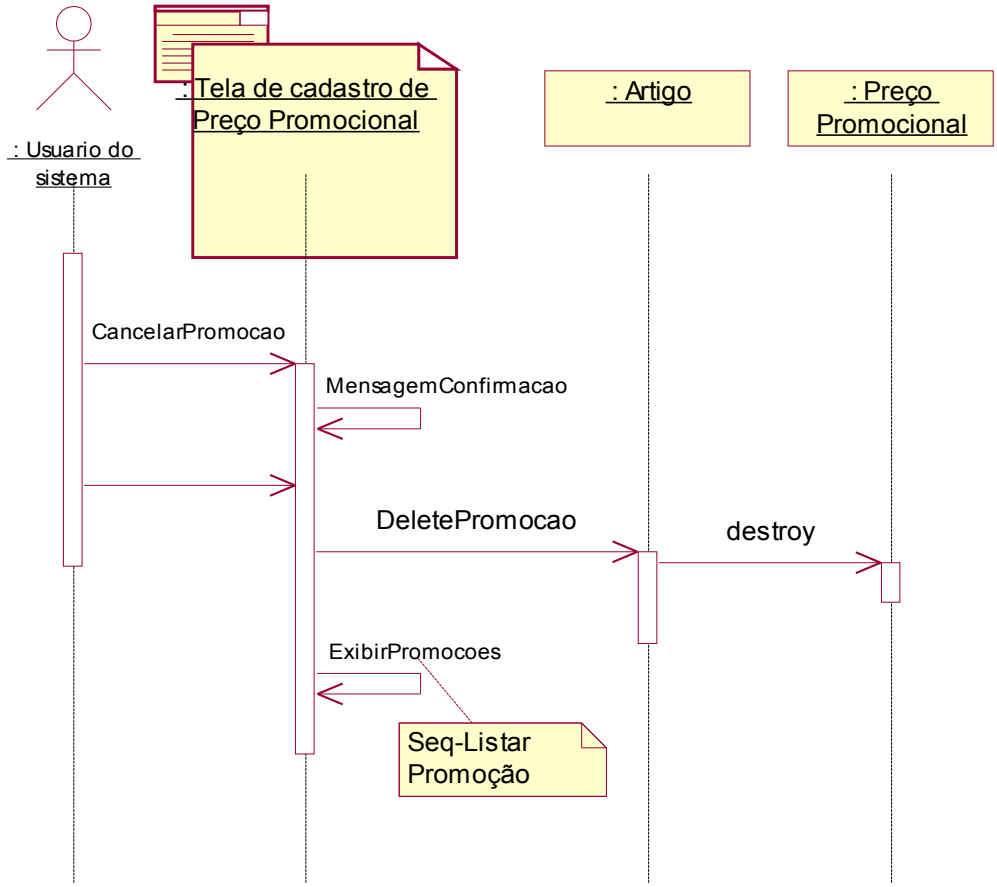
A seguir serão listados os diagramas de seqüência envolvendo as telas e os objetos de negócio especificados durante a iteração do desenvolvimento do sistema Livraria. Cada diagrama de seqüência possui o seu respectivo caso de uso associado.

5.5.3 Diagramas de Seqüência

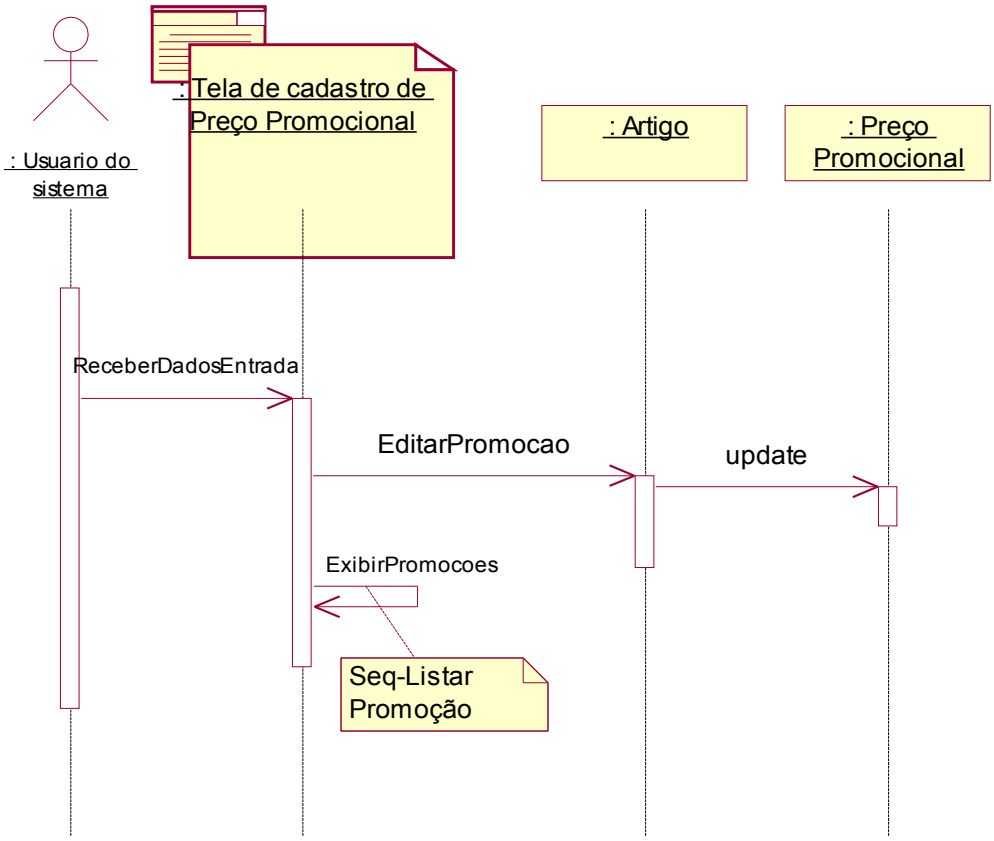
5.5.3.1 Cadastrar Preço Promocional



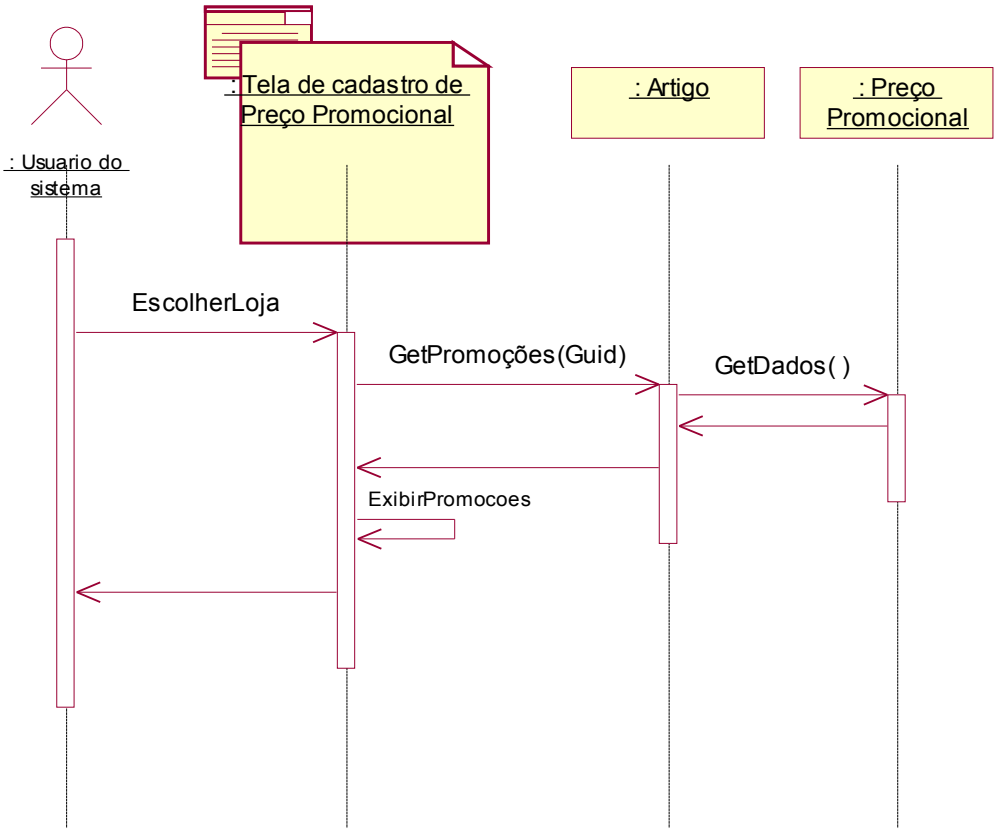
5.5.3.2 Cancelar Preço Promocional



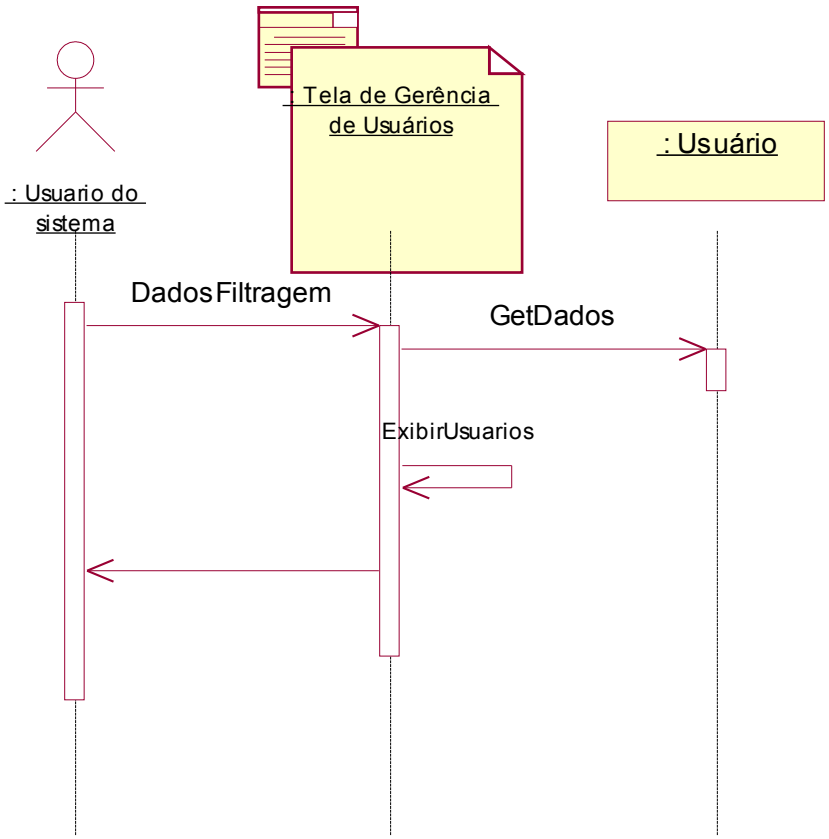
5.5.3.3 Alterar Preço Promocional


: Usuario do sistema
: Tela de cadastro de Preço Promocional
: Artigo
: Preço Promocional
ReceberDadosEntrada
EditarPromocao
update
ExibirPromocoes
Seq-Listar Promoção

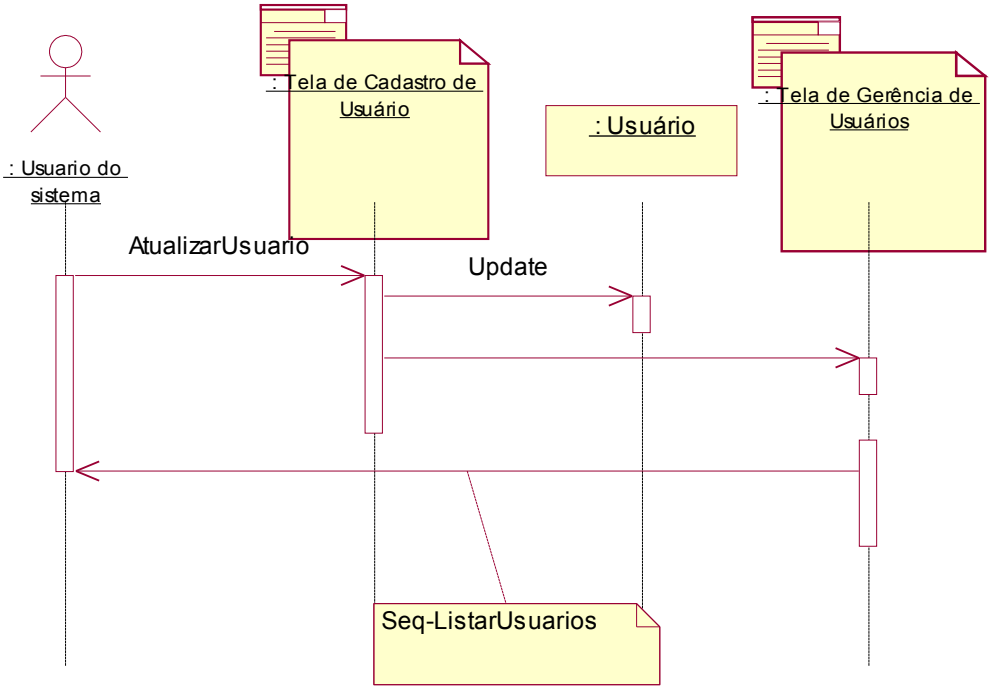
5.5.3.4 Listar Preços Promocionais



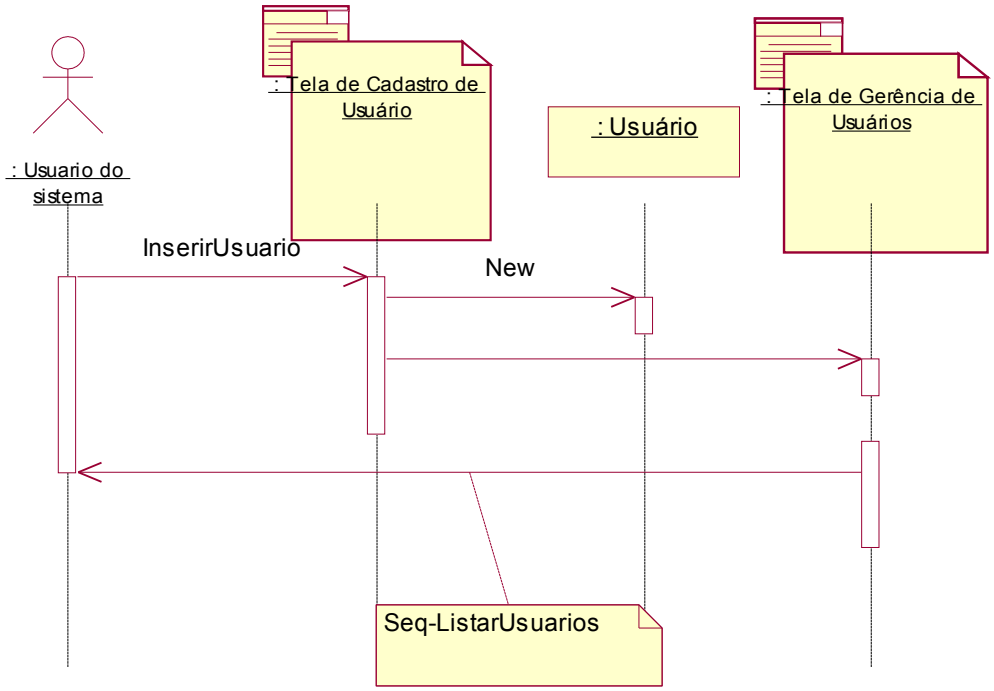
5.5.3.5 Listar Usuários



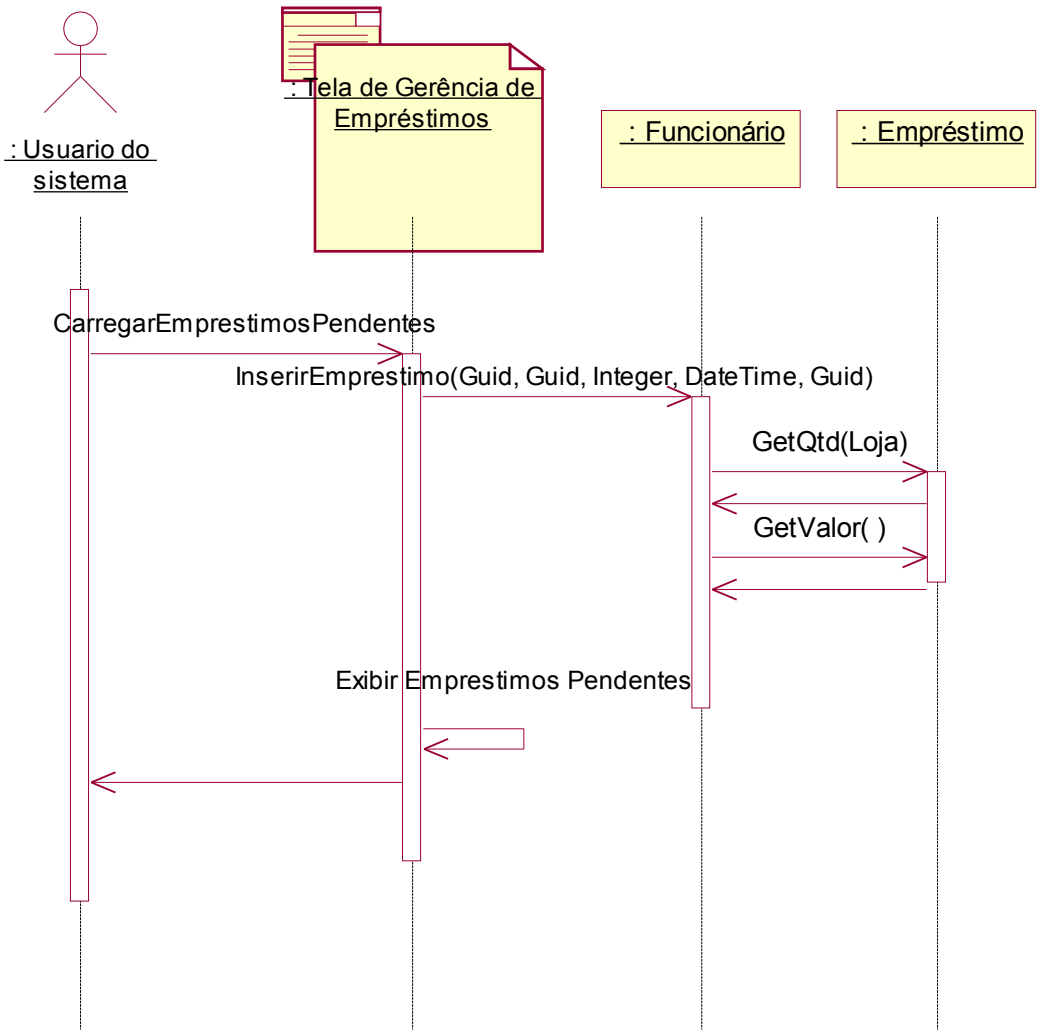
5.5.3.6 Atualizar Usuário



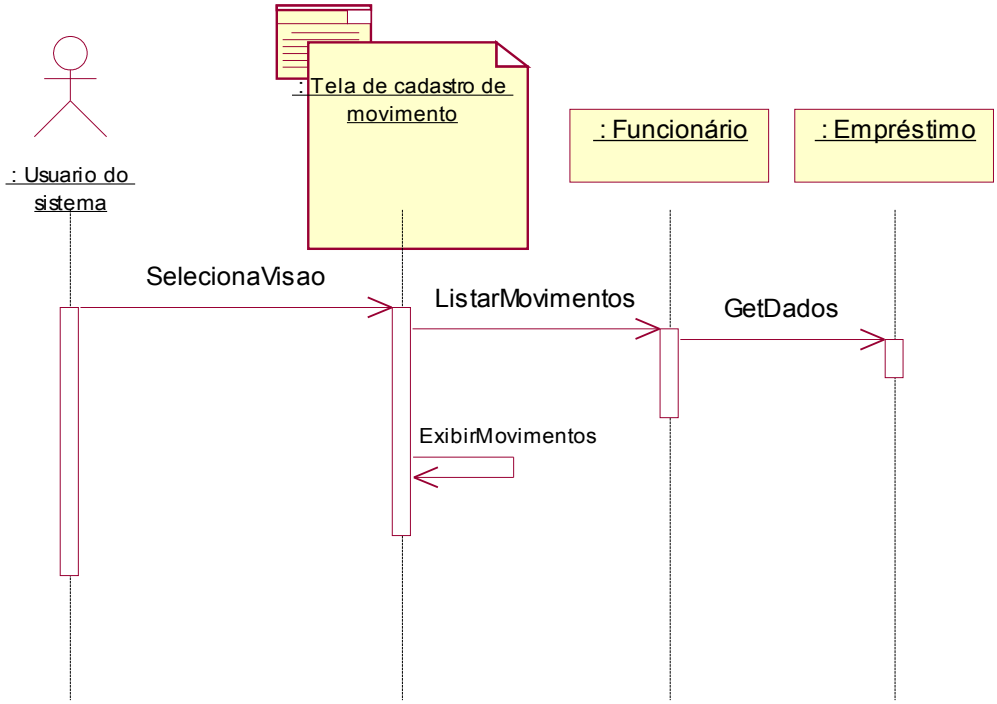
5.5.3.7 Inserir Usuário



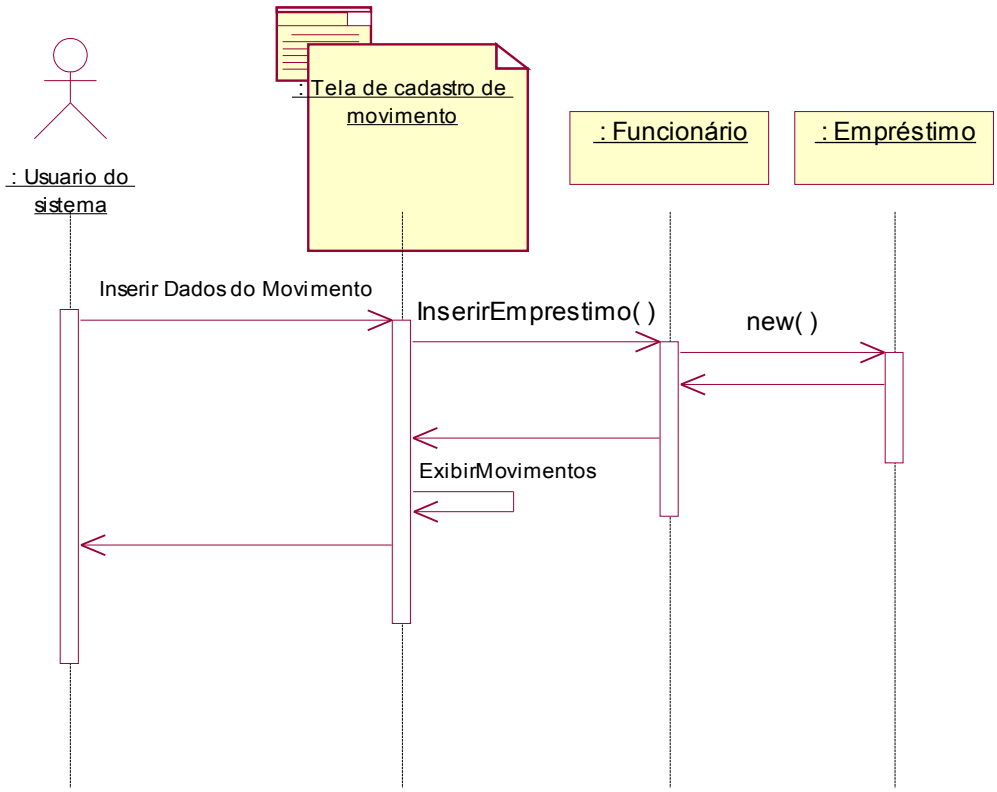
5.5.3.8 Listar Empréstimos



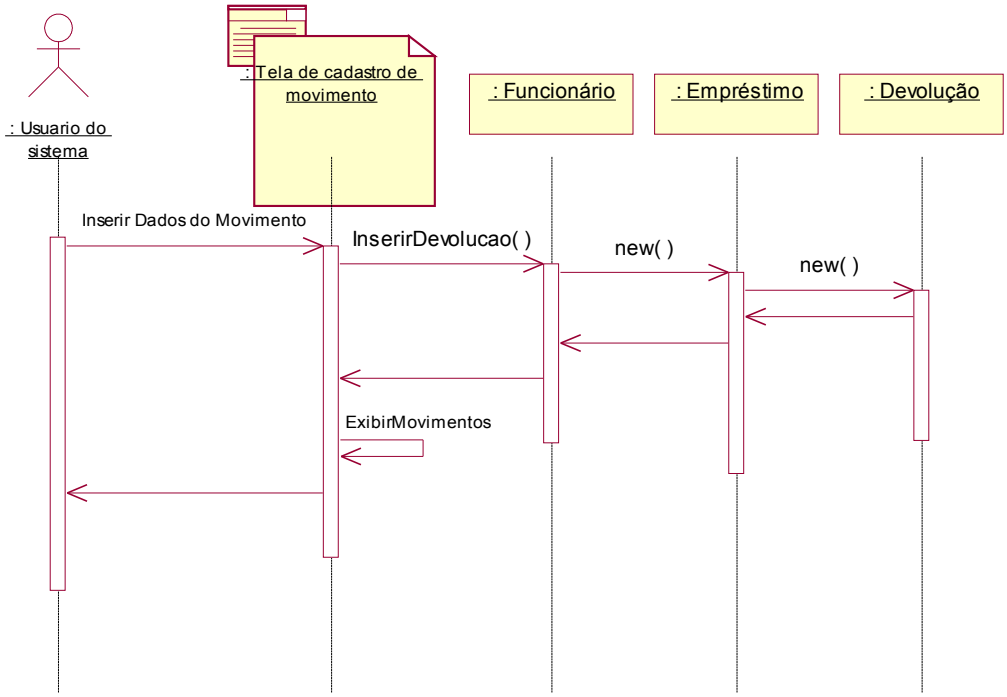
5.5.3.9 Listar Movimentos



5.5.3.10 Inserir Empréstimo



5.5.3.11 Inserir Devolução



5.6 Documento de arquitetura

Este documento visa especificar a arquitetura sob a qual será implementada as funcionalidades da iteração do desenvolvimento do sistema Livraria.

- **Casos de uso arquitetonicamente significativos**

Por se tratar de um projeto onde o foco é estudar a engenharia de software aplicada a aplicações Web, nenhum dos casos de uso extraídos para demonstração são considerados arquitetonicamente significativos.

- **Arquitetura candidata**

A arquitetura candidata a ser utilizada na aplicação é o .NET, da Microsoft. A linguagem a ser utilizada será o ASP.NET, que habilita a criação e o funcionamento de uma aplicação Web. A linguagem escolhida para se conjugar ao ASP.NET para a codificação é o Visual Basic.NET.

A escolha da arquitetura e da linguagem foi feita considerando a experiência anterior da equipe com o ambiente Microsoft.

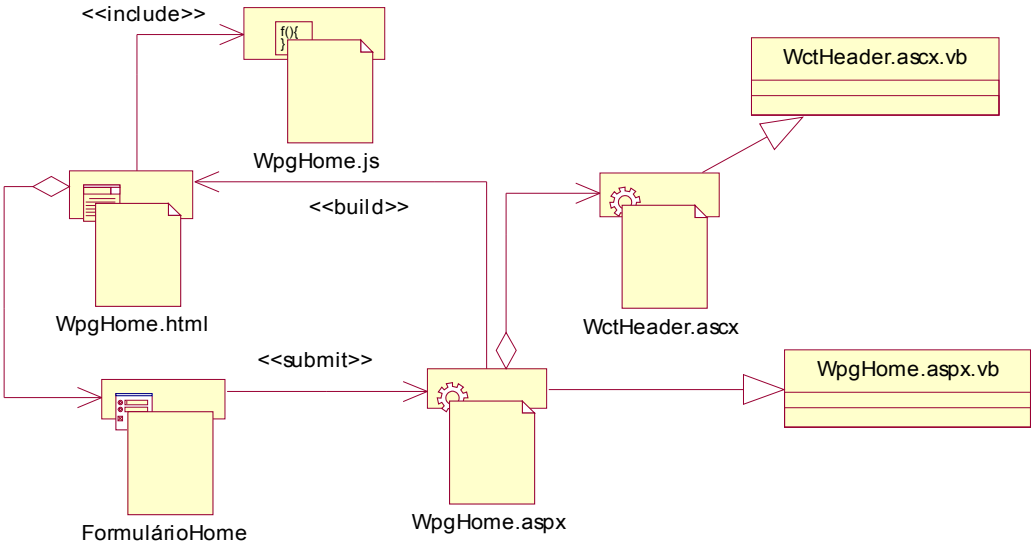
Os outros componentes a serem utilizados serão:

- **Navegador**: Internet Explorer, por se tratar de um browser já existente nas máquinas do cliente que irá utilizar a aplicação Intranet;
- **Protocolo de comunicação**: HTTP
- **Página dinâmica**: páginas Web ASP.NET (.aspx)
- **Servidor de aplicação**: o IIS, pela empresa desenvolvedora ser Microsoft Partner
- **Servidor de banco de dados**: o SQL Server, pela empresa desenvolvedora ser Microsoft Partner e pelo banco comportar aplicação de tal porte.

A aplicação Web .NET define a página Web como a responsável por manipular todos os eventos do lado do cliente. A página MinhaPagina.aspx será modelada como uma classe <<server page>> associada a uma classe oculta MinhaPagina.vb, que irá possuir os códigos que manipulam os eventos e interagem com os objetos do negócio. Desta forma, o arquivo MinhaPagina.aspx é escrito quase todo em HTML, se preocupando apenas com a criação da tela para o cliente.
A manipulação dos eventos do lado cliente poderia ser implementada no próprio arquivo aspx, porém não recomendo esta prática. No projeto, prefiro associar através do estereótipo <<include>> o arquivo MinhaPagina.aspx a um arquivo Javascript MinhaPagina.js, cabendo ao aspx fazer apenas as chamadas às funções do script.

A aplicação Web .NET também permite que se crie arquivos definidos como controles do usuário – user control. Estes arquivos (com a extensão ascx) são basicamente "trechos" de páginas aspx, possuindo seus próprios controles e elementos HTML no cliente, bem como seu código oculto que manipula seus eventos. Desta forma, o arquivo ascx possui uma interface com o cliente tanto quanto a página aspx. Porém, ele necessita da página aspx para ser utilizado pelo usuário, bastando que no arquivo aspx exista a referência ao "user control", quantas vezes o desenvolvedor julgar necessário.

O esquema básico de modelagem de páginas Web .NET a ser utilizado é mostrado a seguir. Nele, temos os relacionamentos existentes entre a página Web WpgHome.aspx com um "user control" chamado WctHeader.ascx:



5.6.1 Estratégia de reutilização

Para usar da melhor forma a reutilização no .NET, deve-se levar em conta que não pode haver herança entre os arquivos .ascx e os .js. Porém, haverá funções de script no lado do cliente que serão utilizadas por grande parte das telas e serão, portanto, candidatas a serem agrupadas em um próprio arquivo Javascript.

A reutilização dos "user controls" se dará na medida em que a equipe identifique um trecho de interface com funções bem definidas que se repetirá na tela do usuário com freqüência. Este trecho de interface terá, obviamente, uma representação HTML própria e seus objetos de interface irão disparar eventos também. Desta forma, este trecho será candidato a ser encapsulado em um arquivo ascx.

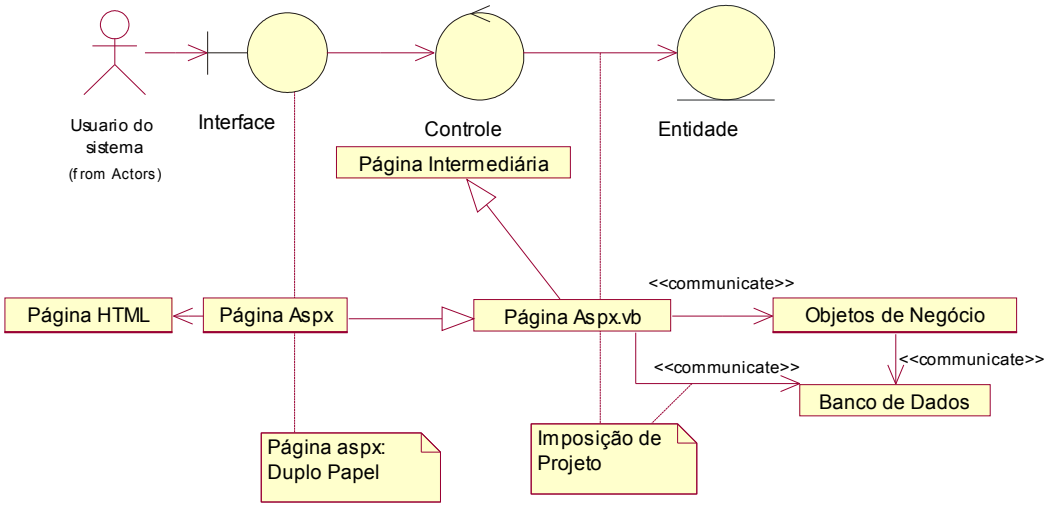
Já as páginas Web .NET geradas pela ferramenta herdam de uma classe do framework denominada "Page", que já possui encapsulados diversos métodos que facilitam a troca de informações entre o seu formulário e o servidor. Porém, a página aspx não pode

simplesmente herdar de uma outra página aspx, pois cada elemento inserido na página aspx possui uma representação única em HTML, e no momento da construção do HTML da tela as tags dos objetos da classe página herdada não serão encontradas.

Para que haja a reutilização de páginas aspx, a equipe deverá criar uma classe intermediária que herde da classe "Page". Esta classe intermediária deverá ser herdada por todas as classes das páginas Web aspx criadas durante o desenvolvimento. As páginas aspx deverão conter apenas seus elementos - de servidor e HTML – e seus "user controls", sem a presença de tags <html>, <form>, etc.

A classe intermediária deverá apenas conter métodos, sem representação HTML. Estes métodos serão responsáveis por construir o documento HTML a partir dos elementos inseridos na página filha, retirando assim uma função que caberia normalmente a própria página Web aspx.

A classe da página aspx, apesar de parecer "acéfala" durante sua construção, permanecerá realizando a comunicação com os objetos do negócio, como se pode observar na figura abaixo:



Note a presença da página aspx na fronteira entre a interface e o controle do sistema. Isto se deve ao fato da página aspx exercer o duplo papel de agregar os elementos de servidor e

HTML e ao mesmo tempo controlar a manipulação de eventos no servidor e dialogar com os objetos de negócio através de sua superclasse aspx.vb.

Por sua vez, a classe aspx.vb também foi inserida na fronteira entre a camada de controle e de entidade, quando na teoria ela deveria apenas se comunicar com os objetos de negócio.

5.6.2 Considerações Finais

A aplicação Web nos impõe alguns limites. E um deles nos faz considerar o trafégo de rede e uma utilização de banda eficiente que não prejudique os demais usuários da aplicação. Ao requerer um objeto de negócio, a página aspx.vb irá fazer com que a classe vá até a base de dados e instancie o objeto. Isto deverá ocorrer inúmeras vezes durante a interação entre usuário e sistema. Até este momento nada de novo.

Dependendo do contexto da página Web, porém, o simples fato de instanciar um objeto pode ser muito custoso. Imagine, por exemplo, uma tela de busca que apresente como resultado os dados dos artigos da livraria que possuam a letra “a” no seu título. Uma query bem formulada ao banco de dados nos dará a resposta em pouco tempo. Em contrapartida, aguardar que o sistema identifique os artigos retornados e os instancie - um a um - para então exibi-los, além de exigir inúmeras conexões ao banco retardará demais o tempo de carregamento da página HTML resultante.

Levando tudo isto em consideração, a equipe deverá identificar em quais momentos a página deverá conhecer ou informar ao usuário informações que remetam a um “todo” (ou conjunto) e a uma “parte” (elemento específico). Provavelmente, as telas que necessitem de muitos dados – como a de resultado de busca – serão fundamentalmente de consulta.

Desta maneira, o projeto não se enquadrará completamente no paradigma orientado a objetos porque:

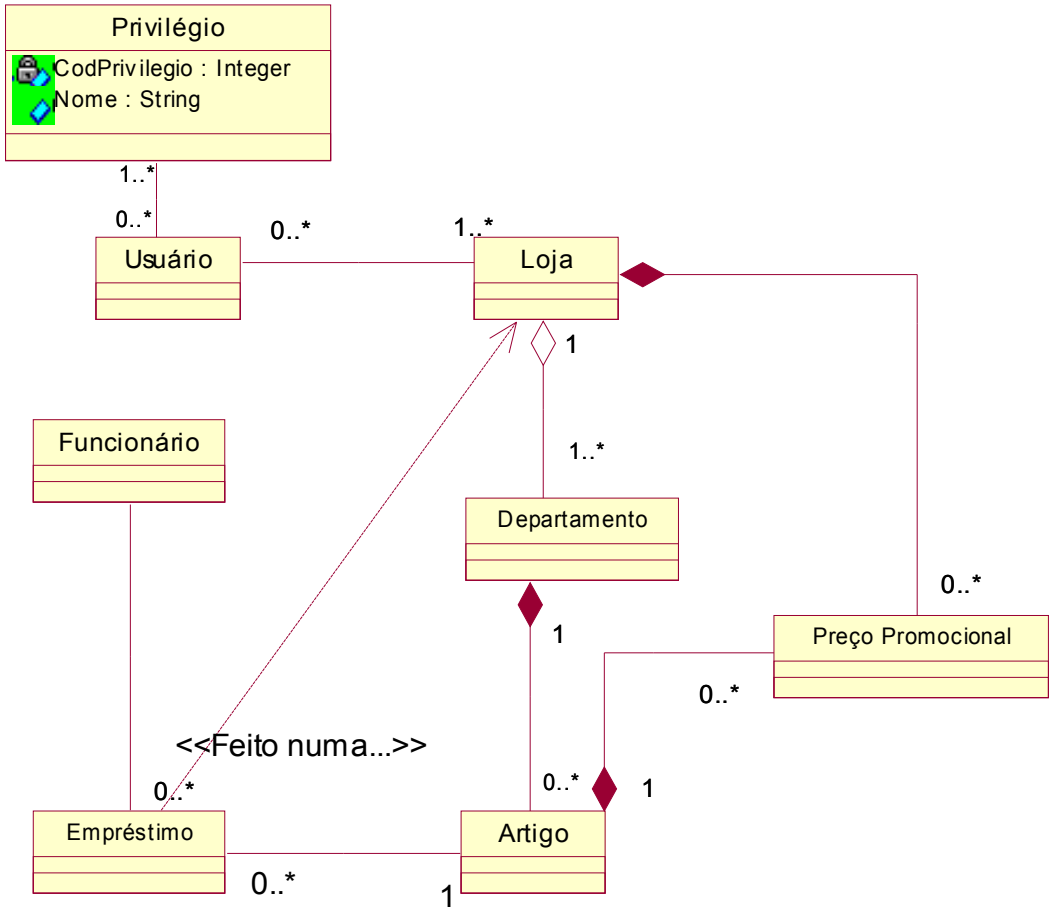
- O comportamento dos objetos não estará encapsulado em seu interior, e sim nas funções e procedures no banco de dados;

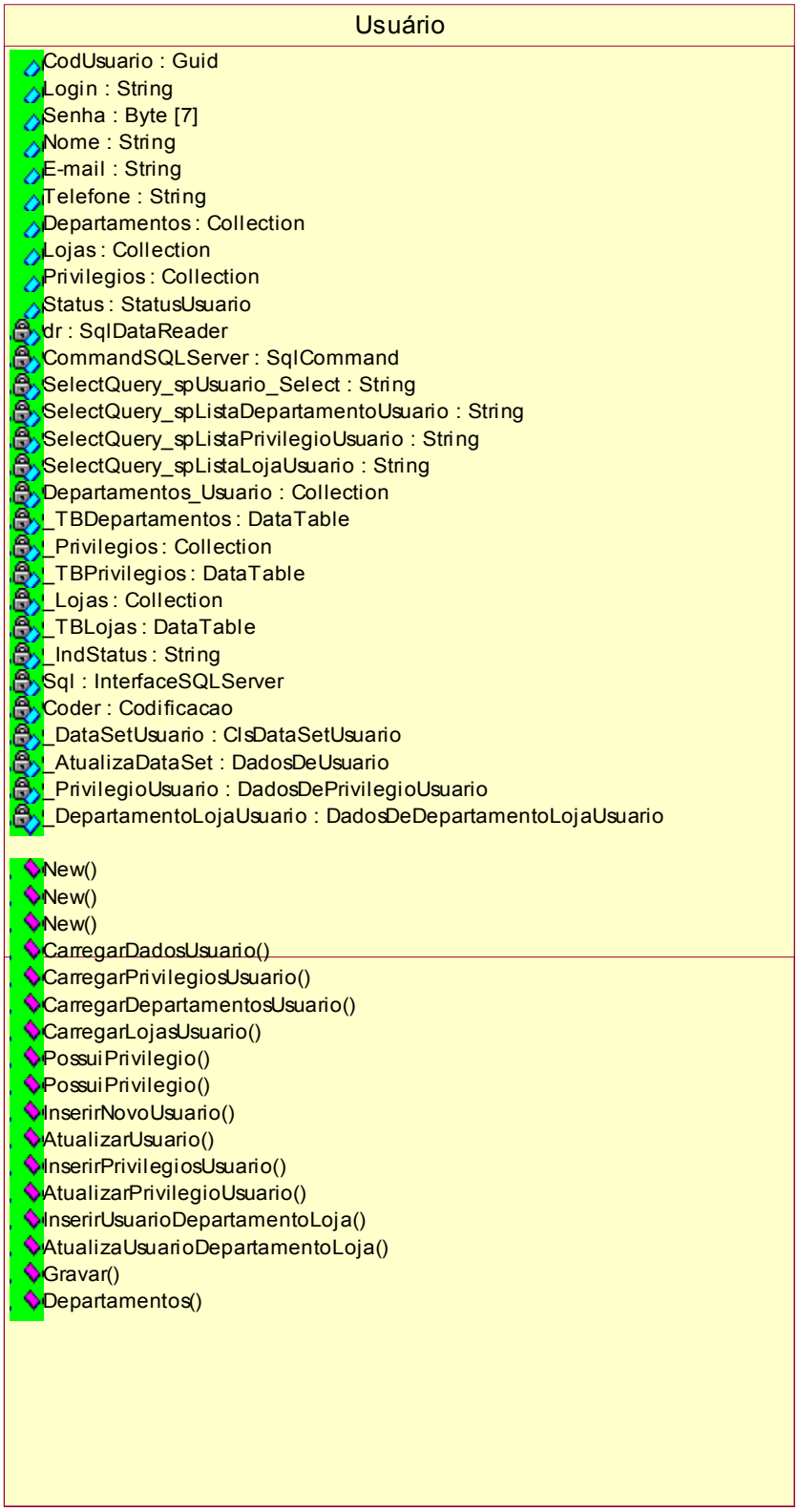
- Os objetos não serão responsáveis, exclusivamente, por fornecer seus dados durante toda a execução do sistema. Haverá classes de suporte que irão fornecer dados que remetem a um “todo”.

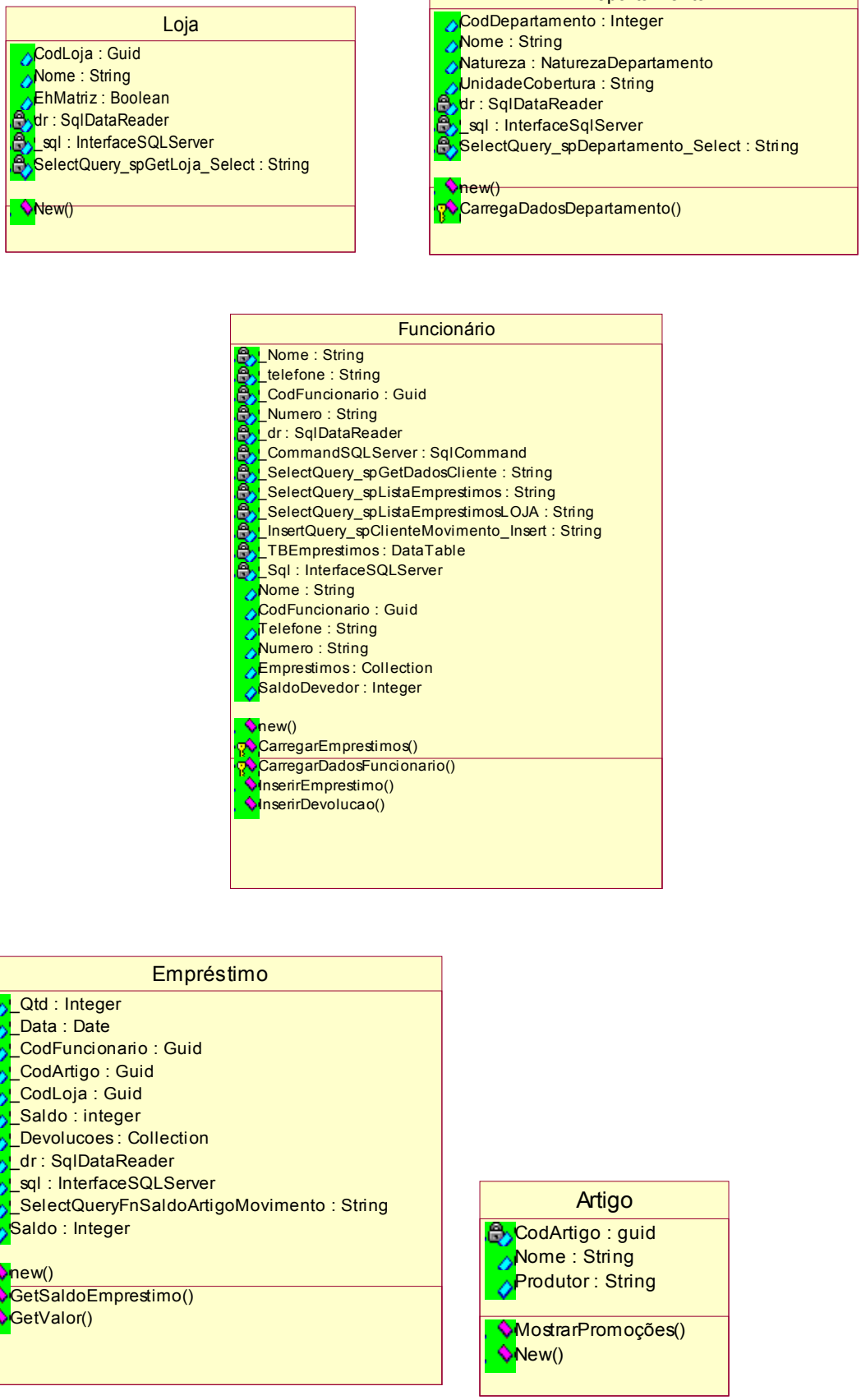
6 Capítulo VI - Etapa Projeto

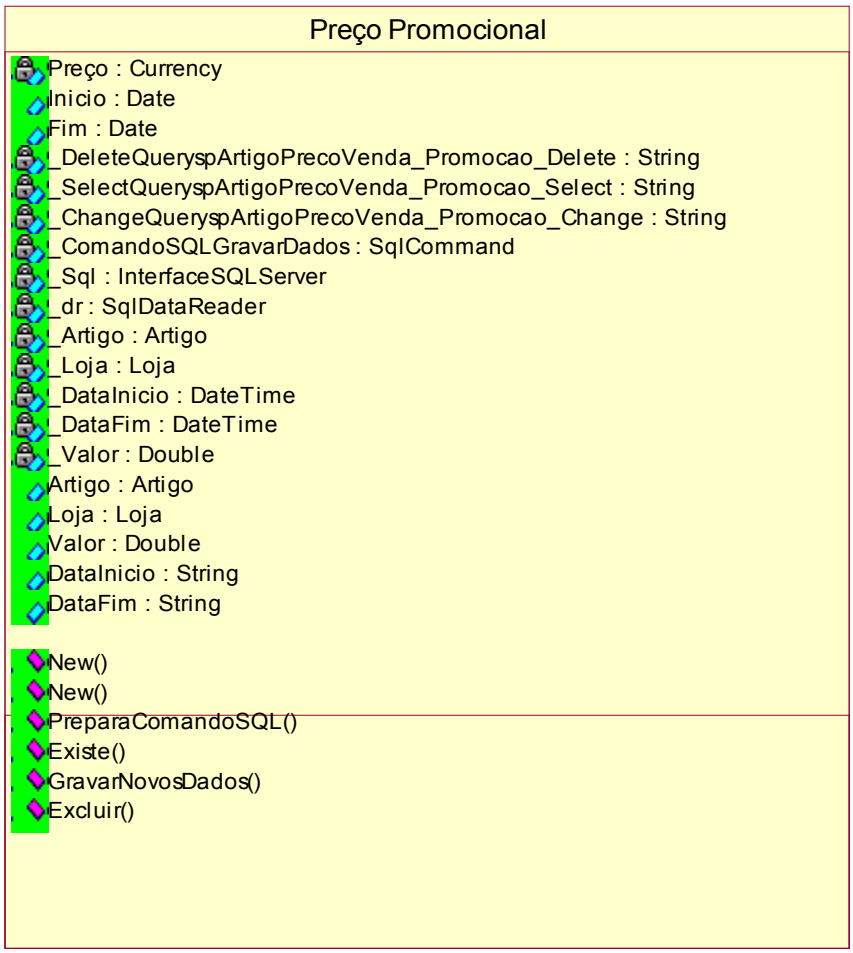
Neste capítulo serão apresentados os modelos de classe e seus relacionamentos com todos os detalhes.

6.1 Diagramas de Classes - Aperfeiçoado

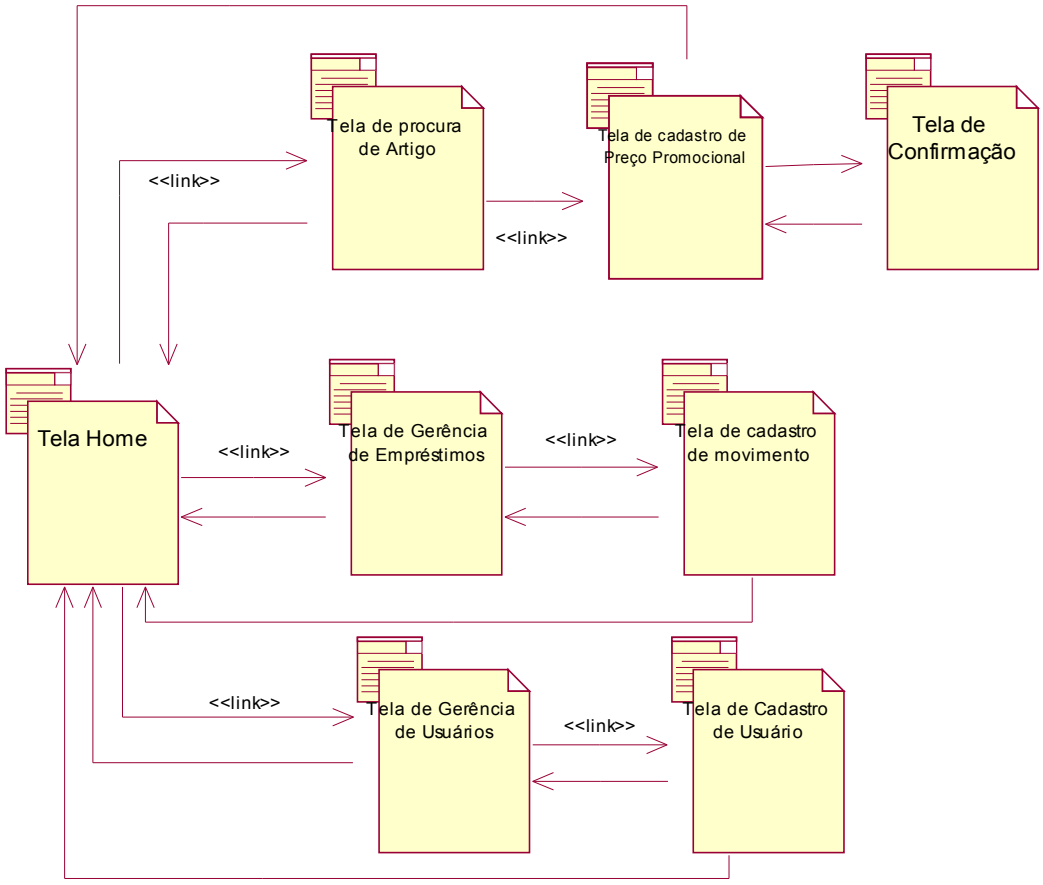



Usuário
CodUsuario : Guid
Login : String
Senha : Byte [7]
Nome : String
E-mail : String
Telefone : String
Departamentos : Collection
Lojas : Collection
Privilegios : Collection
Status : StatusUsuario
dr : SqlDataReader
CommandSQLServer : SqlCommand
SelectQuery_spUsuario_Select : String
SelectQuery_spListaDepartamentoUsuario : String
SelectQuery_spListaPrivilegioUsuario : String
SelectQuery_spListaLojaUsuario : String
Departamentos_Usuario : Collection
_TBDepartamentos : DataTable
_Privilegios : Collection
_TBPrivilegios : DataTable
_Lojas : Collection
_TBLojas : DataTable
_IndStatus : String
Sql : InterfaceSQLServer
Coder : Codificacao
_DataSetUsuario : ClsDataSetUsuario
_AtualizaDataSet : DadosDeUsuario
_PrivilegioUsuario : DadosDePrivilegioUsuario
_DepartamentoLojaUsuario : DadosDeDepartamentoLojaUsuario
New()
New()
New()
CarregarDadosUsuario()
CarregarPrivilegiosUsuario()
CarregarDepartamentosUsuario()
CarregarLojasUsuario()
PossuiPrivilegio()
PossuiPrivilegio()
InserirNovoUsuario()
AtualizarUsuario()
InserirPrivilegiosUsuario()
AtualizarPrivilegioUsuario()
InserirUsuarioDepartamentoLoja()
AtualizaUsuarioDepartamentoLoja()
Gravar()
Departamentos()


Loja
CodLoja : Guid
Nome : String
EhMatriz : Boolean
dr : SqlDataReader
_sql : InterfaceSQLServer
SelectQuery_spGetLoja_Select : String
New()
Departamento
CodDepartamento : Integer
Nome : String
Natureza : NaturezaDepartamento
UnidadeCobertura : String
dr : SqlDataReader
_sql : InterfaceSqlServer
SelectQuery_spDepartamento_Select : String
new()
CarregaDadosDepartamento()
Funcionário
_Nome : String
_telefone : String
_CodFuncionario : Guid
_Numero : String
_dr : SqlDataReader
_CommandSQLServer : SqlCommand
_SelectQuery_spGetDadosCliente : String
_SelectQuery_spListaEmprestimos : String
_SelectQuery_spListaEmprestimosLOJA : String
_InsertQuery_spClienteMovimento_Insert : String
_TBEmprestimos : DataTable
_Sql : InterfaceSQLServer
Nome : String
CodFuncionario : Guid
Telefone : String
Numero : String
Emprestimos : Collection
SaldoDevedor : Integer
new()
CarregarEmprestimos()
CarregarDadosFuncionario()
InserirEmprestimo()
InserirDevolucao()
Empréstimo
_Qtd : Integer
_Data : Date
_CodFuncionario : Guid
_CodArtigo : Guid
_CodLoja : Guid
_Saldo : integer
_Devolucoes : Collection
_dr : SqlDataReader
_sql : InterfaceSQLServer
_SelectQueryFnSaldoArtigoMovimento : String
Saldo : Integer
new()
GetSaldoEmprestimo()
GetValor()
Artigo
CodArtigo : guid
Nome : String
Produtor : String
MostrarPromoções()
New()

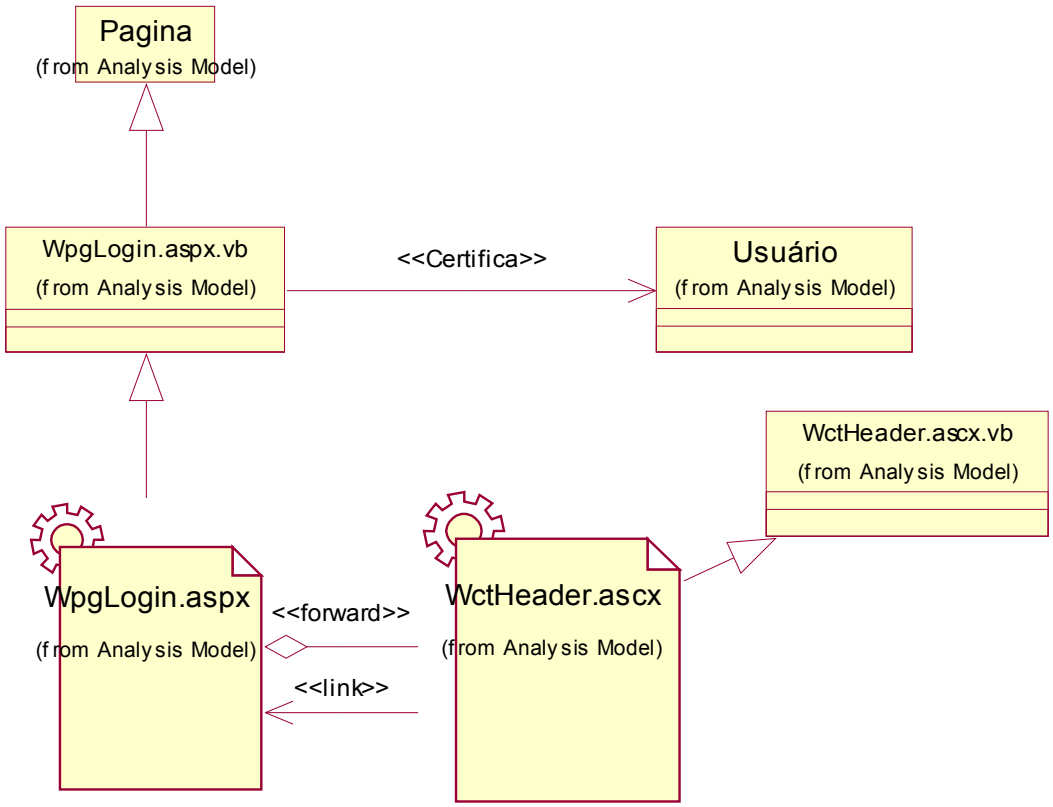

Preço Promocional
_Preço : Currency
Inicio : Date
Fim : Date
_DeleteQueryspArtigoPrecoVenda_Promocao_Delete : String
_SelectQueryspArtigoPrecoVenda_Promocao_Select : String
_ChangeQueryspArtigoPrecoVenda_Promocao_Change : String
_ComandoSQLGravarDados : SqlCommand
_Sql : InterfaceSQLServer
_dr : SqlDataReader
_Artigo : Artigo
_Loja : Loja
_DataInicio : DateTime
_DataFim : DateTime
_Valor : Double
Artigo : Artigo
Loja : Loja
Valor : Double
DataInicio : String
DataFim : String
New()
New()
PreparaComandoSQL()
Existe()
GravarNovosDados()
Excluir()

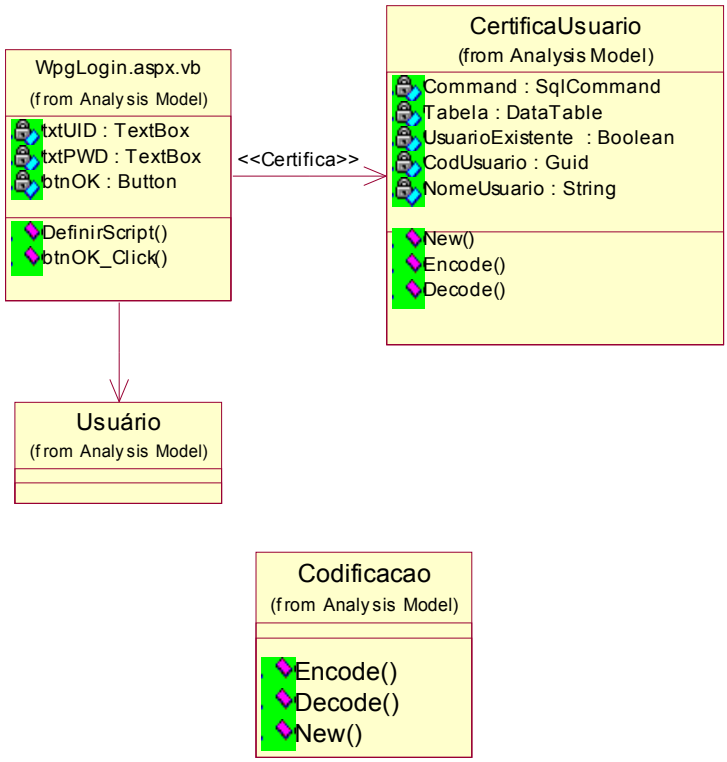
6.2 Caminhos de navegação



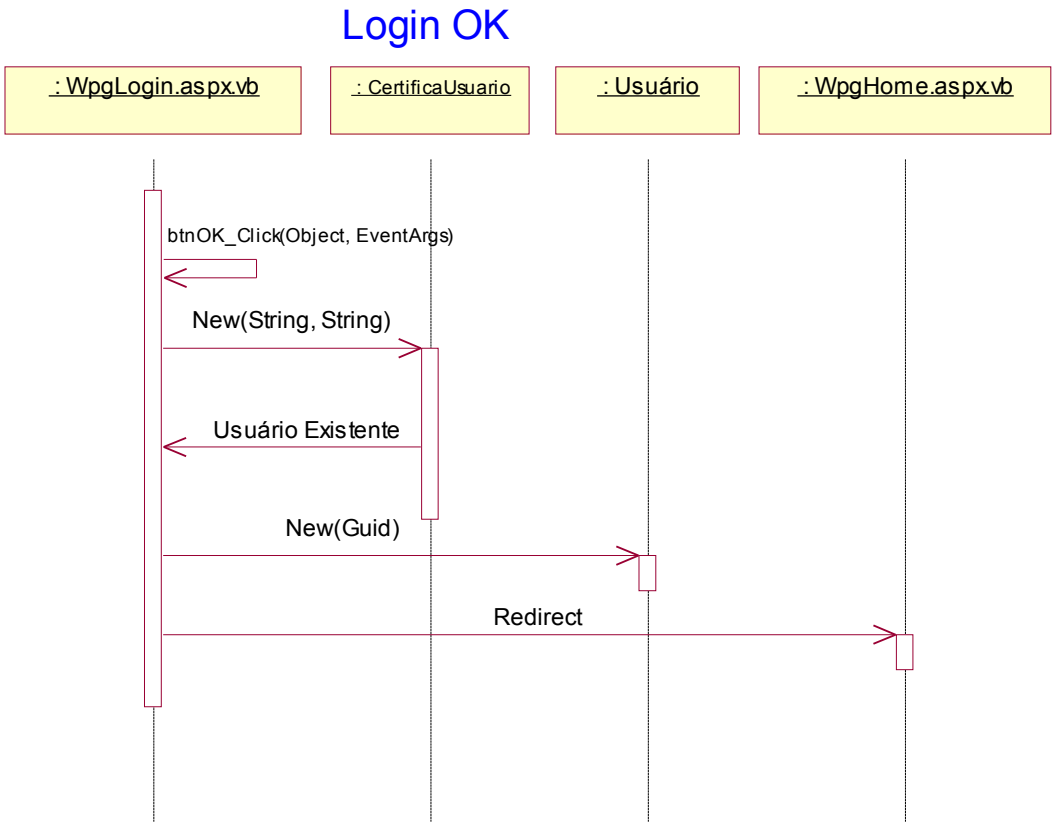
6.3 Pacote Login

6.3.1 Relacionamentos - Tela de Login

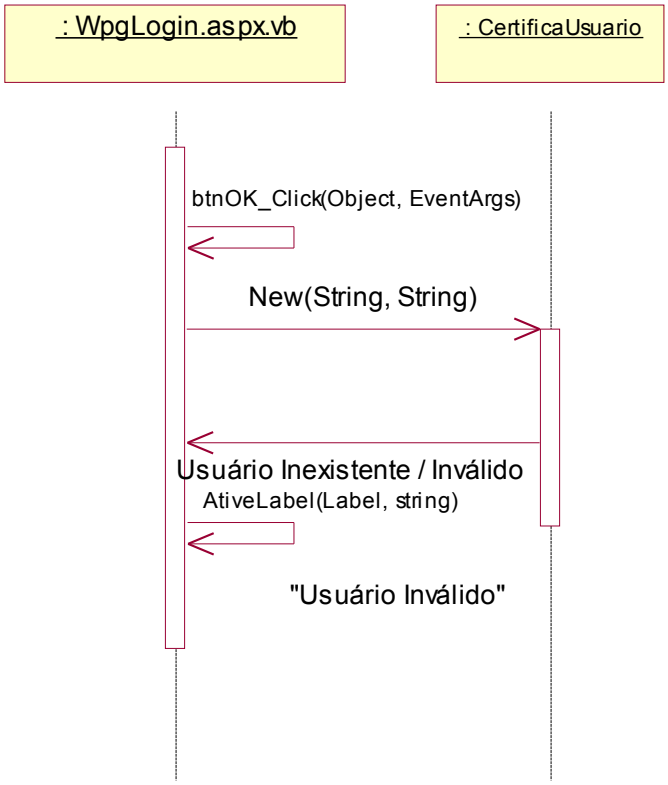




6.3.2 Diagramas de Seqüência

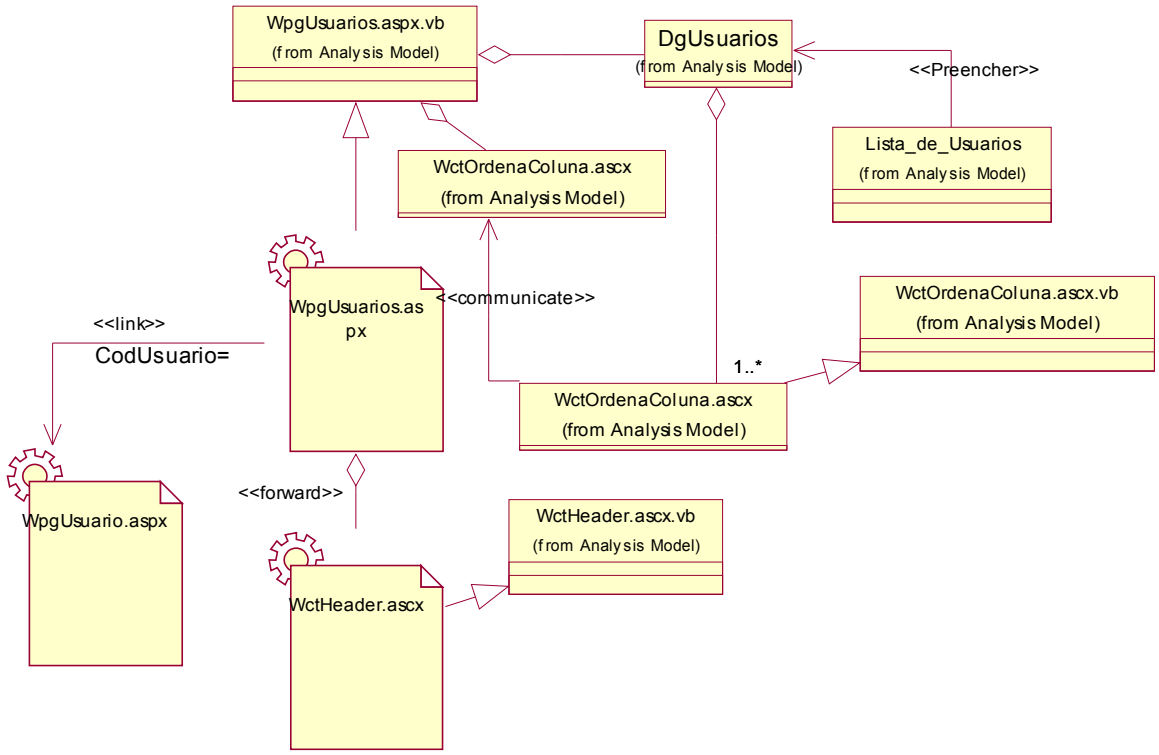

Login OK
: WpgLogin.aspx.vb
: CertificaUsuario
: Usuário
: WpgHome.aspx.vb
btnOK_Click(Object, EventArgs)
New(String, String)
Usuário Existente
New(Guid)
Redirect

Login Falho

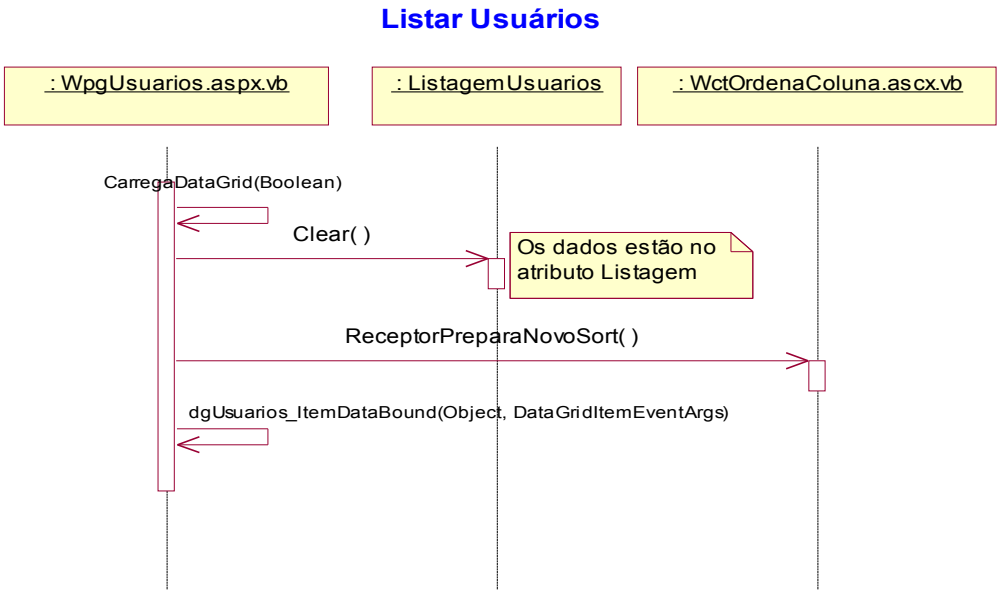

: WpgLogin.aspx.vb
: CertificaUsuario
btnOK_Click(Object, EventArgs)
New(String, String)
Usuário Inexistente / Inválido
AtiveLabel(Label, string)
"Usuário Inválido"

6.4 Pacote Usuário

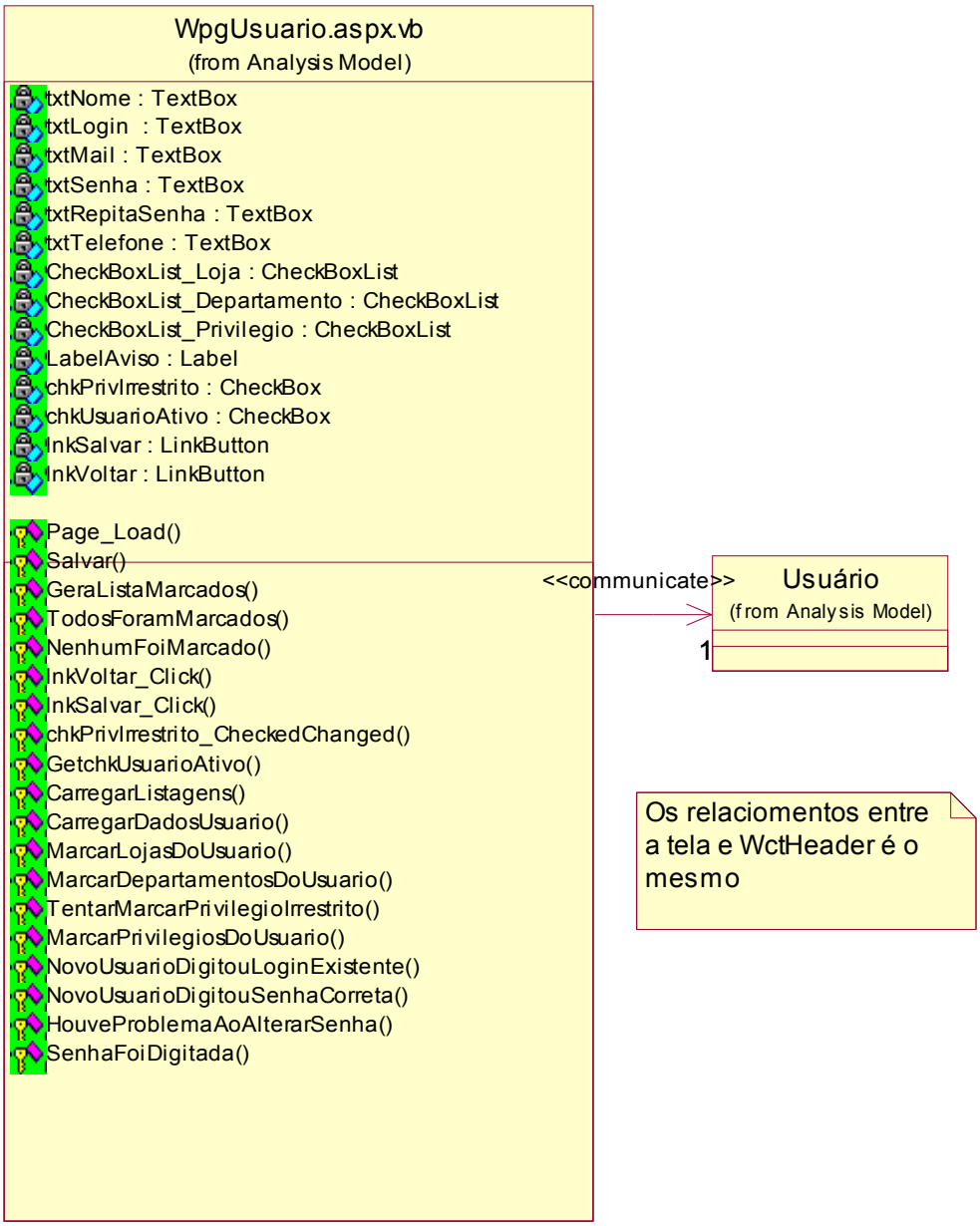
6.4.1 Relacionamentos – Tela de Gerência de Usuários

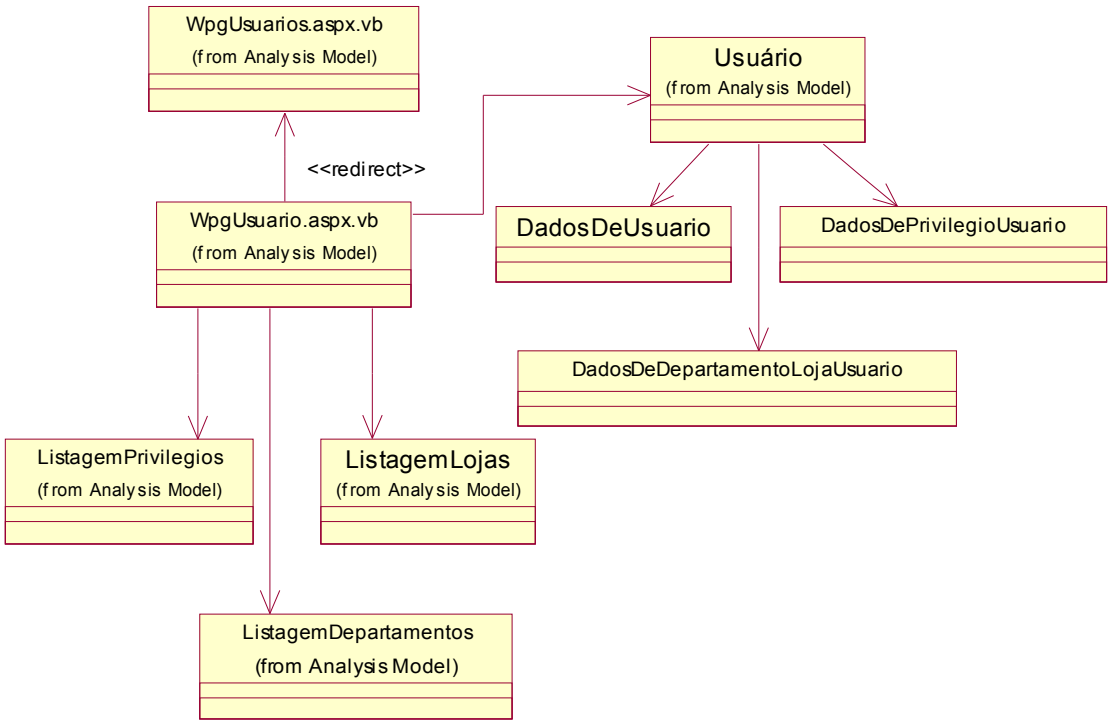


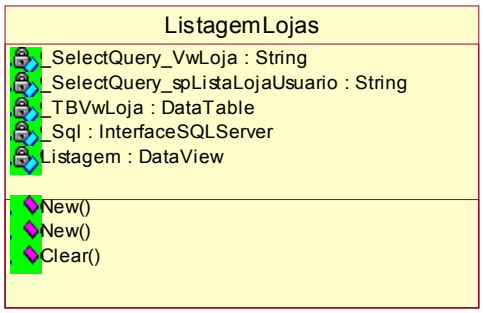
6.4.1.1 Diagrama de Seqüência

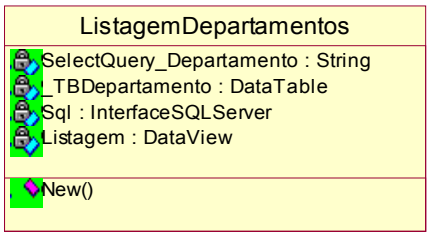


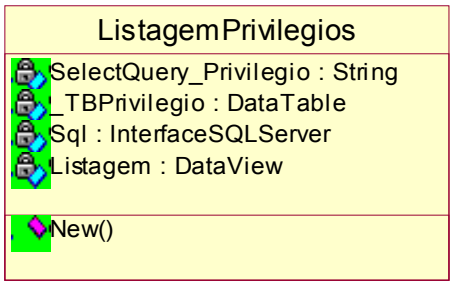
6.4.2 Relacionamentos – Tela de Cadastro de Usuário

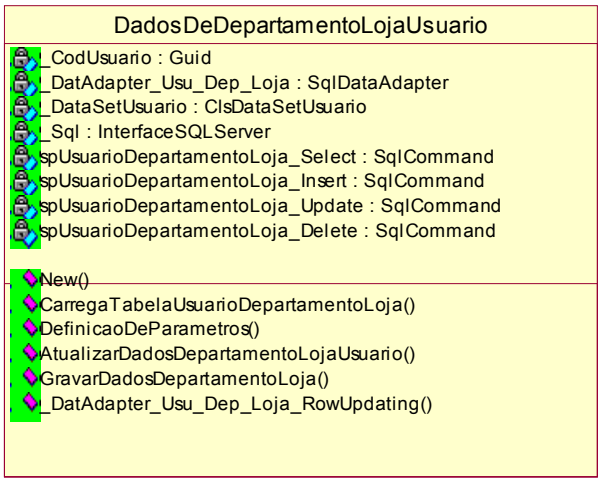

WpgUsuario.aspx.vb
(from Analysis Model)
txtNome : TextBox
txtLogin : TextBox
txtMail : TextBox
txtSenha : TextBox
txtRepitaSenha : TextBox
txtTelefone : TextBox
CheckBoxList_Loja : CheckBoxList
CheckBoxList_Departamento : CheckBoxList
CheckBoxList_Privilegio : CheckBoxList
LabelAviso : Label
chkPrivIrrestrito : CheckBox
chkUsuarioAtivo : CheckBox
lnkSalvar : LinkButton
lnkVoltar : LinkButton
Page_Load()
Salvar()
GeraListaMarcados()
TodosForamMarcados()
NenhumFoiMarcado()
lnkVoltar_Click()
lnkSalvar_Click()
chkPrivIrrestrito_CheckedChanged()
GetchkUsuarioAtivo()
CarregarListagens()
CarregarDadosUsuario()
MarcarLojasDoUsuario()
MarcarDdepartamentosDoUsuario()
TentarMarcarPrivilegioIrrestrito()
MarcarPrivilegiosDoUsuario()
NovoUsuarioDigitouLoginExistente()
NovoUsuarioDigitouSenhaCorreta()
HouveProblemaAoAlterarSenha()
SenhaFoiDigitada()
<<communicate>>
Usuário
(from Analysis Model)
1
Os relaciomentos entre a tela e WctHeader é o mesmo

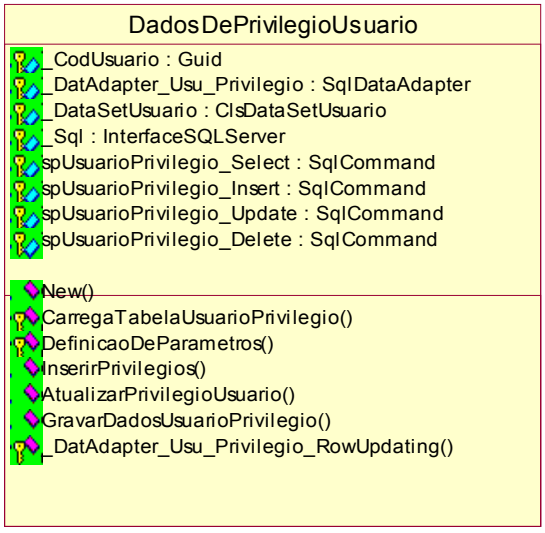

WpgUsuarios.aspx.vb
(from Analysis Model)
Usuário
(from Analysis Model)
<<redirect>>
WpgUsuario.aspx.vb
(from Analysis Model)
DadosDeUsuario
DadosDePrivilegioUsuario
DadosDeDepartamentoLojaUsuario
ListagemPrivilegios
(from Analysis Model)
ListagemLojas
(from Analysis Model)
ListagemDepartamentos
(from Analysis Model)

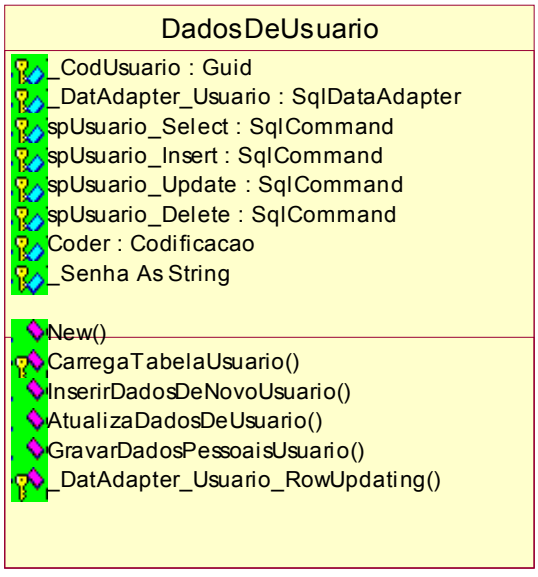

ListagemLojas
_SelectQuery_VwLoja : String
_SelectQuery_spListaLojaUsuario : String
_TBVwLoja : DataTable
_Sql : InterfaceSQLServer
Listagem : DataView
New()
New()
Clear()


ListagemDepartamentos
SelectQuery_Departamento : String
_TBDepartamento : DataTable
Sql : InterfaceSQLServer
Listagem : DataView
New()

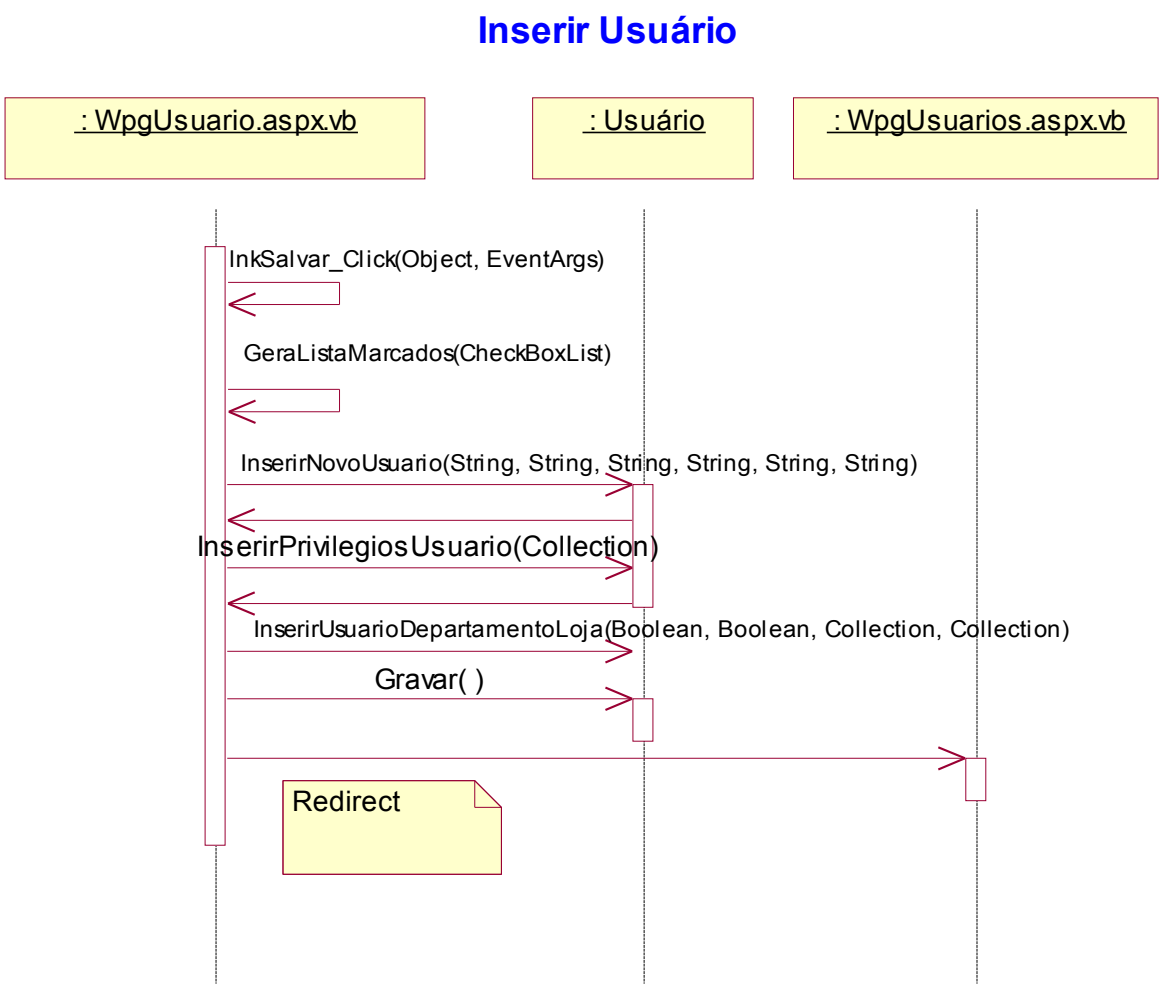

ListagemPrivilegios
SelectQuery_Privilegio : String
_TBPrivilegio : DataTable
Sql : InterfaceSQLServer
Listagem : DataView
New()


DadosDeDepartamentoLojaUsuario
_CodUsuario : Guid
_DatAdapter_Usu_Dep_Loja : SqlDataAdapter
_DataSetUsuario : ClsDataSetUsuario
_Sql : InterfaceSQLServer
spUsuarioDepartamentoLoja_Select : SqlCommand
spUsuarioDepartamentoLoja_Insert : SqlCommand
spUsuarioDepartamentoLoja_Update : SqlCommand
spUsuarioDepartamentoLoja_Delete : SqlCommand
New()
CarregaTabelaUsuarioDepartamentoLoja()
DefinicaoDeParametros()
AtualizarDadosDepartamentoLojaUsuario()
GravarDadosDepartamentoLoja()
_DatAdapter_Usu_Dep_Loja_RowUpdating()

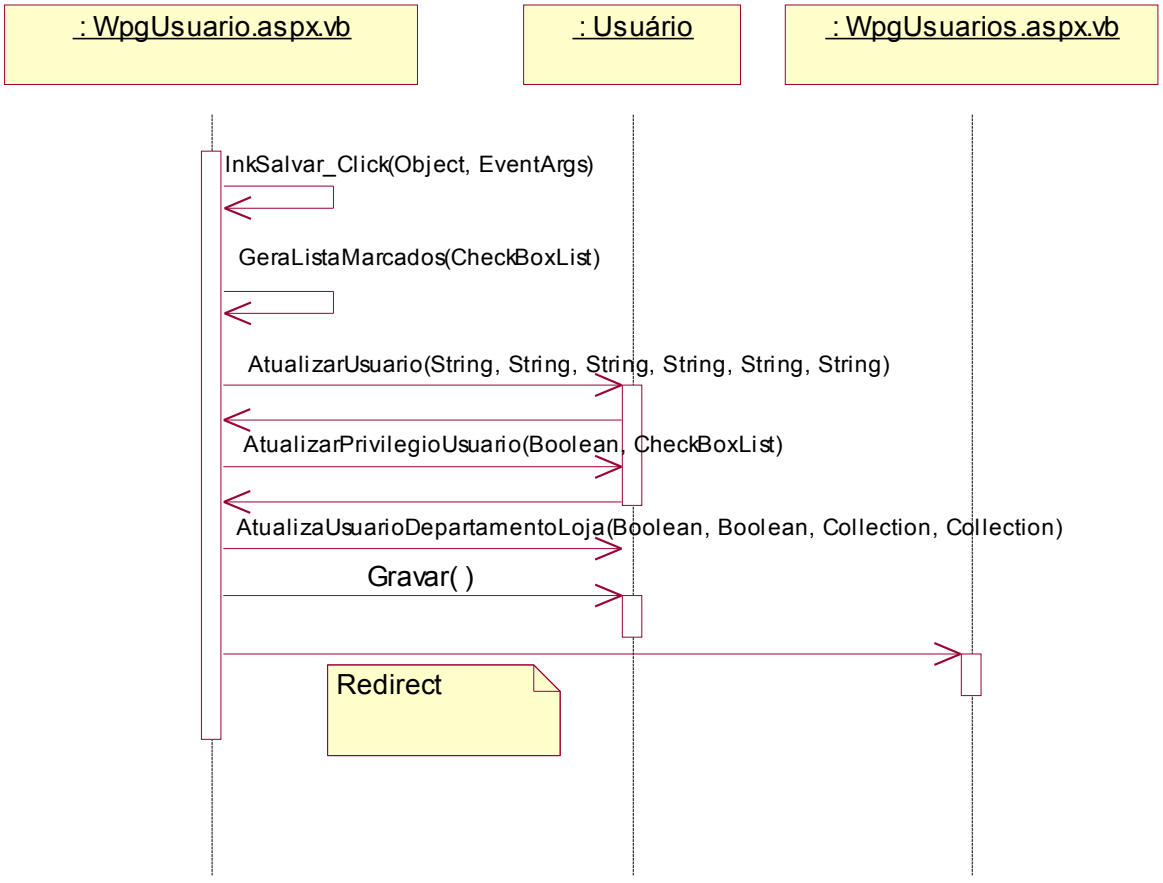

DadosDePrivilegioUsuario
_CodUsuario : Guid
_DatAdapter_Usu_Privilegio : SqlDataAdapter
_DataSetUsuario : ClsDataSetUsuario
_Sql : InterfaceSQLServer
spUsuarioPrivilegio_Select : SqlCommand
spUsuarioPrivilegio_Insert : SqlCommand
spUsuarioPrivilegio_Update : SqlCommand
spUsuarioPrivilegio_Delete : SqlCommand
New()
CarregaTabelaUsuarioPrivilegio()
DefinicaoDeParametros()
InserirPrivilegios()
AtualizarPrivilegioUsuario()
GravarDadosUsuarioPrivilegio()
_DatAdapter_Usu_Privilegio_RowUpdating()


DadosDeUsuario
_CodUsuario : Guid
_DatAdapter_Usuario : SqlDataAdapter
spUsuario_Select : SqlCommand
spUsuario_Insert : SqlCommand
spUsuario_Update : SqlCommand
spUsuario_Delete : SqlCommand
Coder : Codificacao
_Senha As String
New()
CarregaTabelaUsuario()
InserirDadosDeNovoUsuario()
AtualizaDadosDeUsuario()
GravarDadosPessoaisUsuario()
_DatAdapter_Usuario_RowUpdating()

6.4.2.1 Diagramas de Seqüência

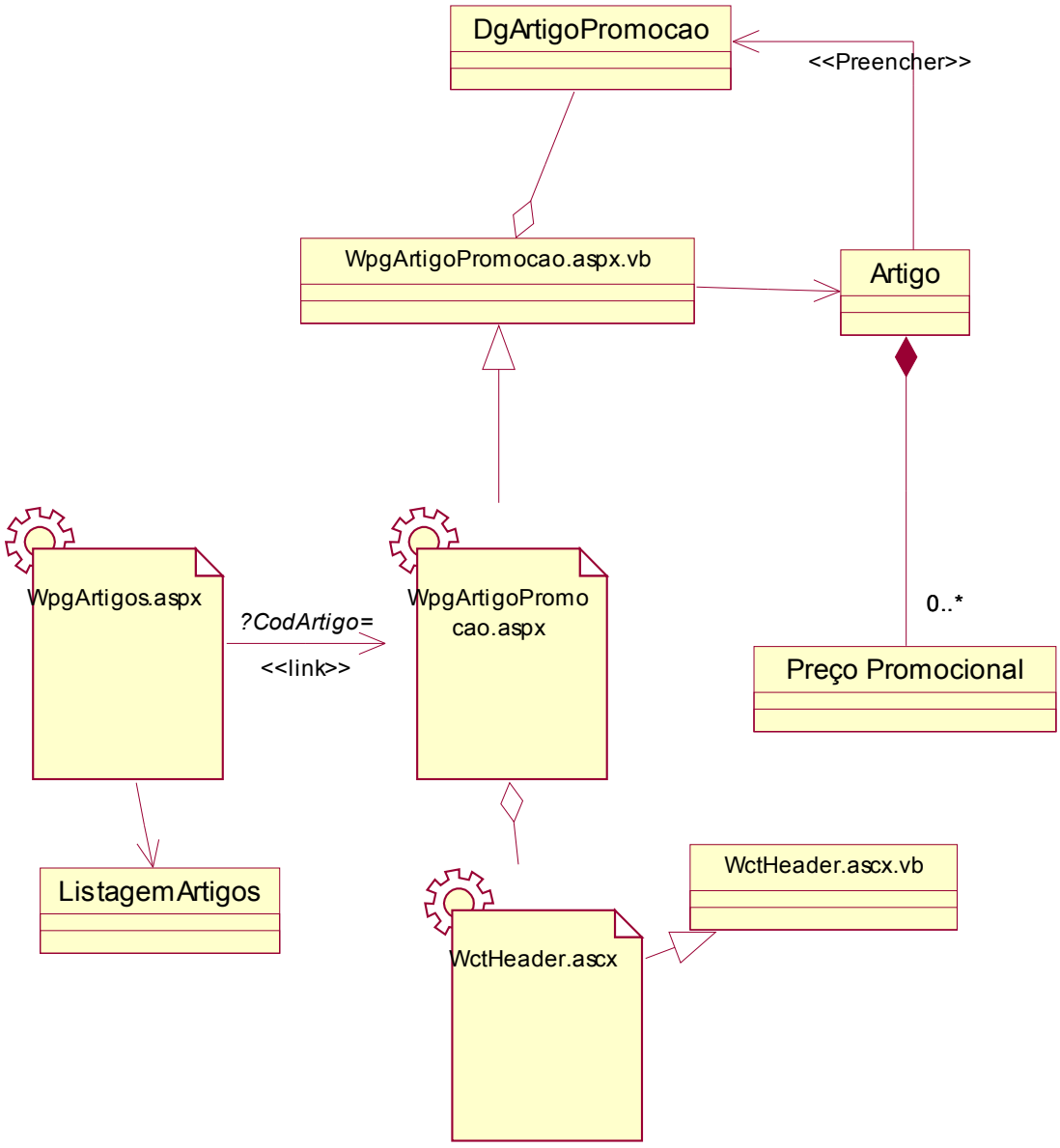


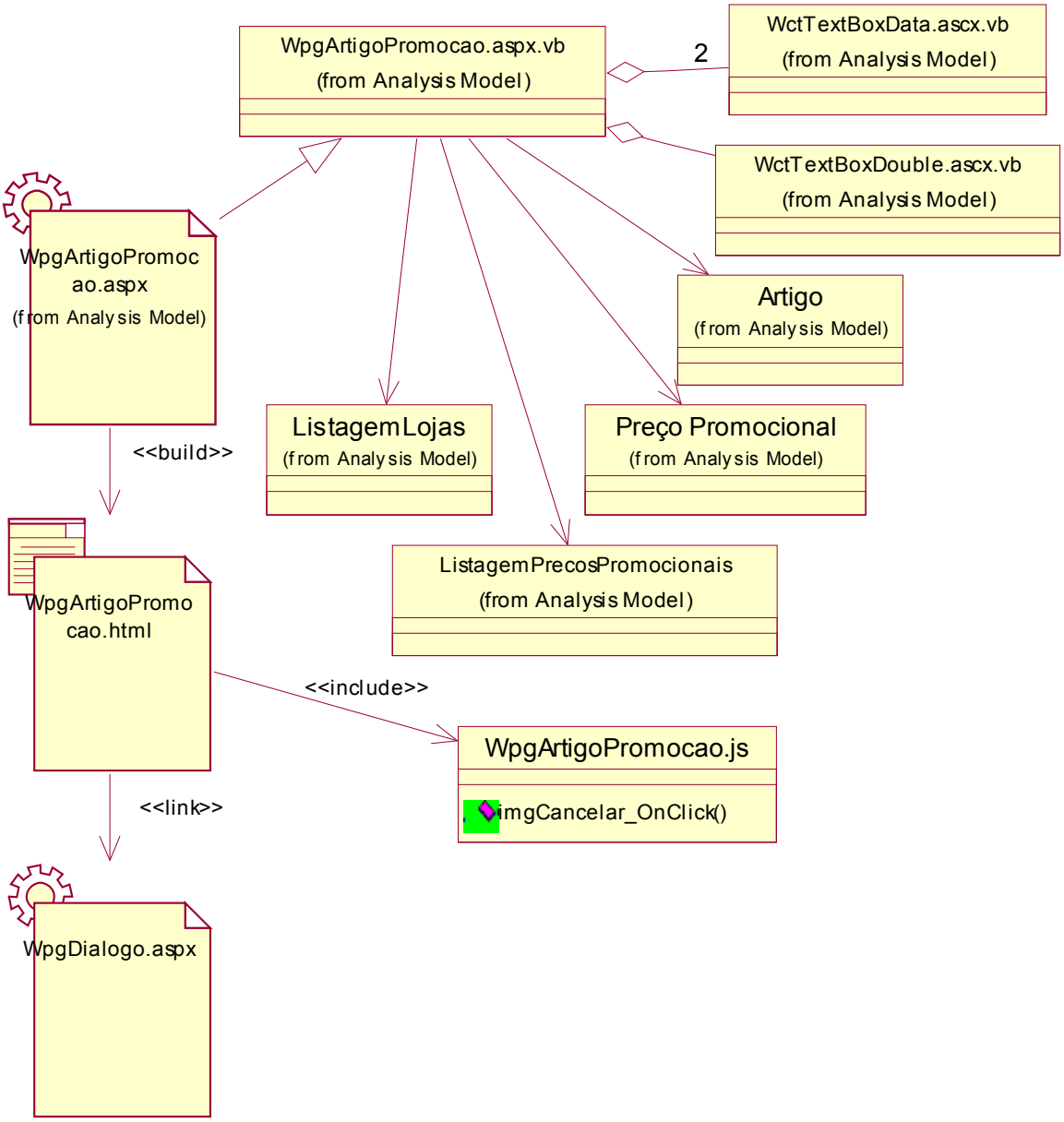
Atualizar Usuário

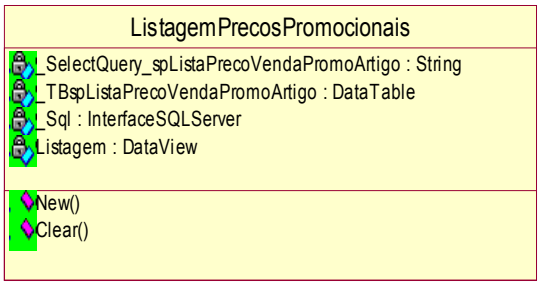


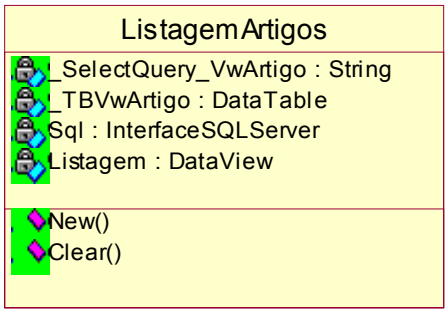
6.5 Pacote Promoção

6.5.1 Relacionamentos – Tela de Promoções

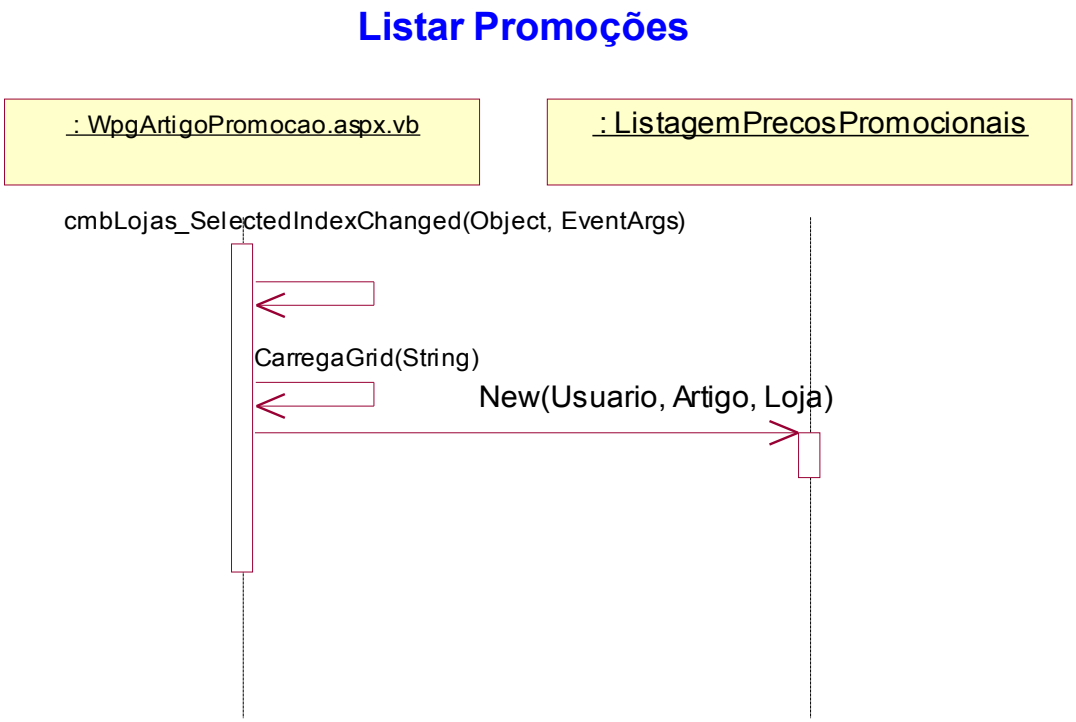

DgArtigoPromocao
<<Preencher>>
WpgArtigoPromocao.aspx.vb
Artigo
WpgArtigos.aspx
?CodArtigo=
<<link>>
WpgArtigoPromo
cao.aspx
0..*
Preço Promocional
ListagemArtigos
WctHeader.ascx.vb
WctHeader.ascx


WpgArtigoPromocao.aspx.vb
(from Analysis Model)
2
WctTextBoxData.ascx.vb
(from Analysis Model)
WctTextBoxDouble.ascx.vb
(from Analysis Model)
WpgArtigoPromocao.aspx
(from Analysis Model)
Artigo
(from Analysis Model)
ListagemLojas
(from Analysis Model)
Preço Promocional
(from Analysis Model)
<<build>>
ListagemPrecosPromocionais
(from Analysis Model)
WpgArtigoPromocao.html
<<include>>
WpgArtigoPromocao.js
imgCancelar_OnClick()
<<link>>
WpgDialogo.aspx

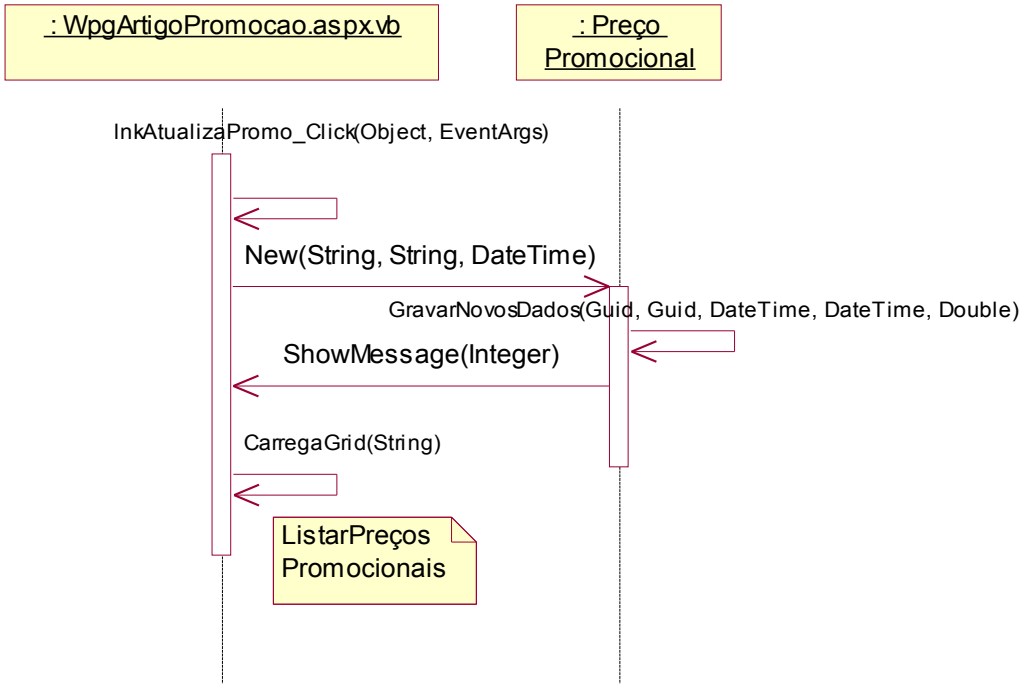

ListagemPrecosPromocionais
_SelectQuery_spListaPrecoVendaPromoArtigo : String
_TBspListaPrecoVendaPromoArtigo : DataTable
_Sql : InterfaceSQLServer
Listagem : DataView
New()
Clear()


ListagemArtigos
_SelectQuery_VwArtigo : String
_TBVwArtigo : DataTable
Sql : InterfaceSQLServer
Listagem : DataView
New()
Clear()

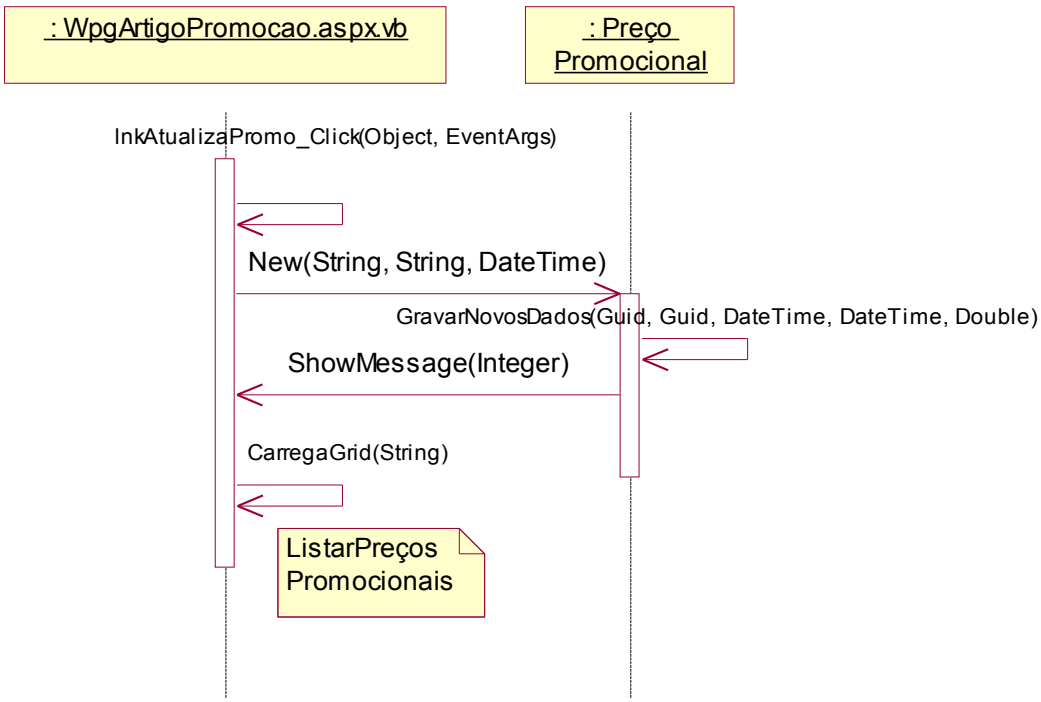
6.5.2 Diagramas de Seqüência

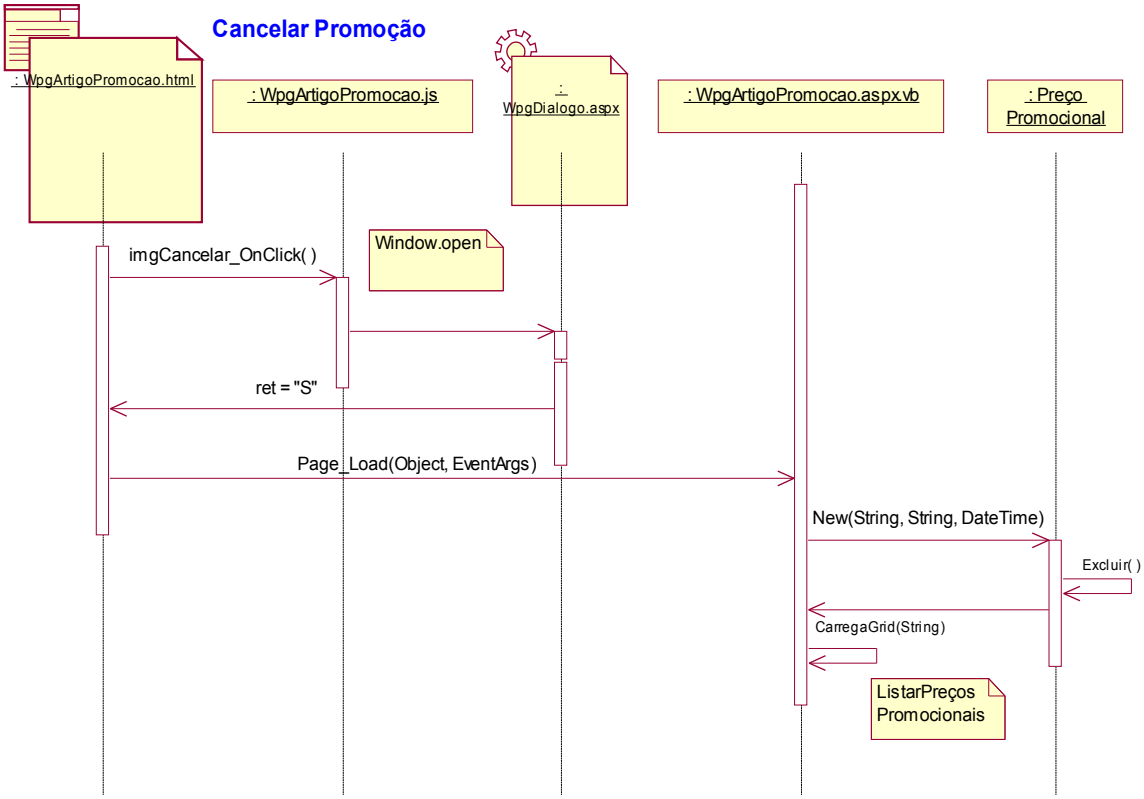


Cadastrar Promoção



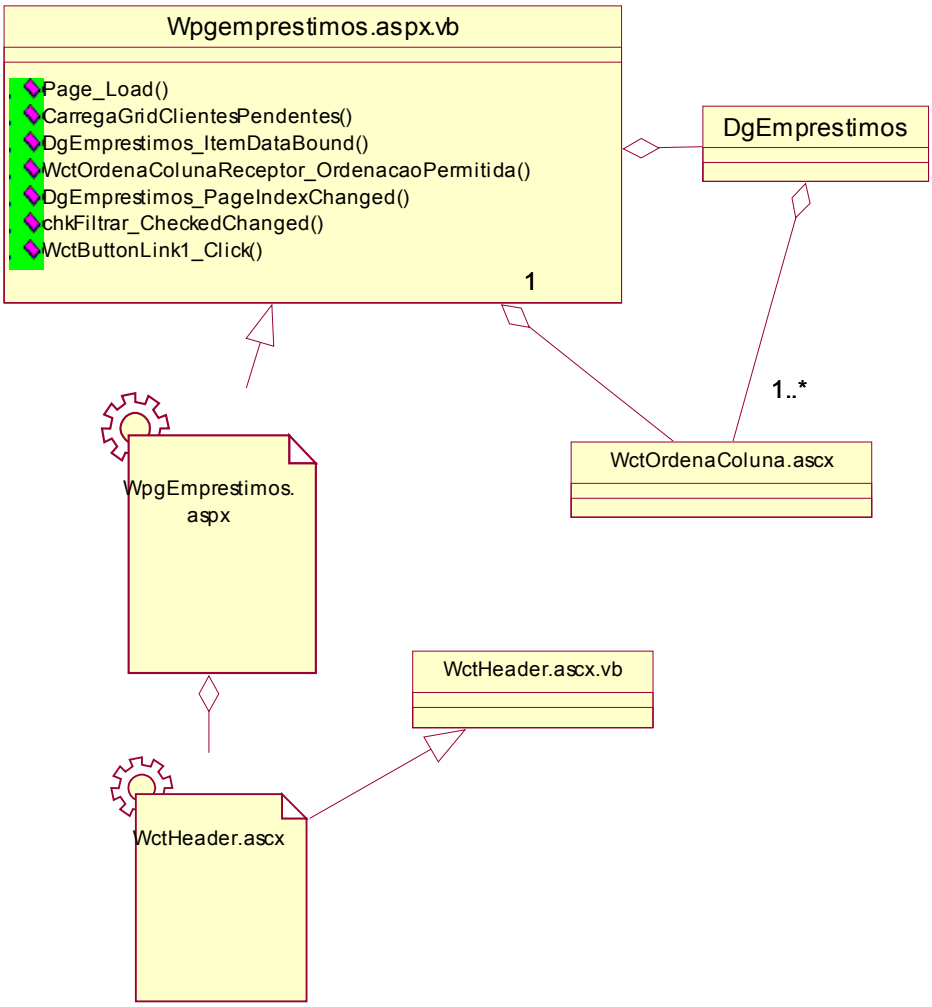
Alterar Promoção

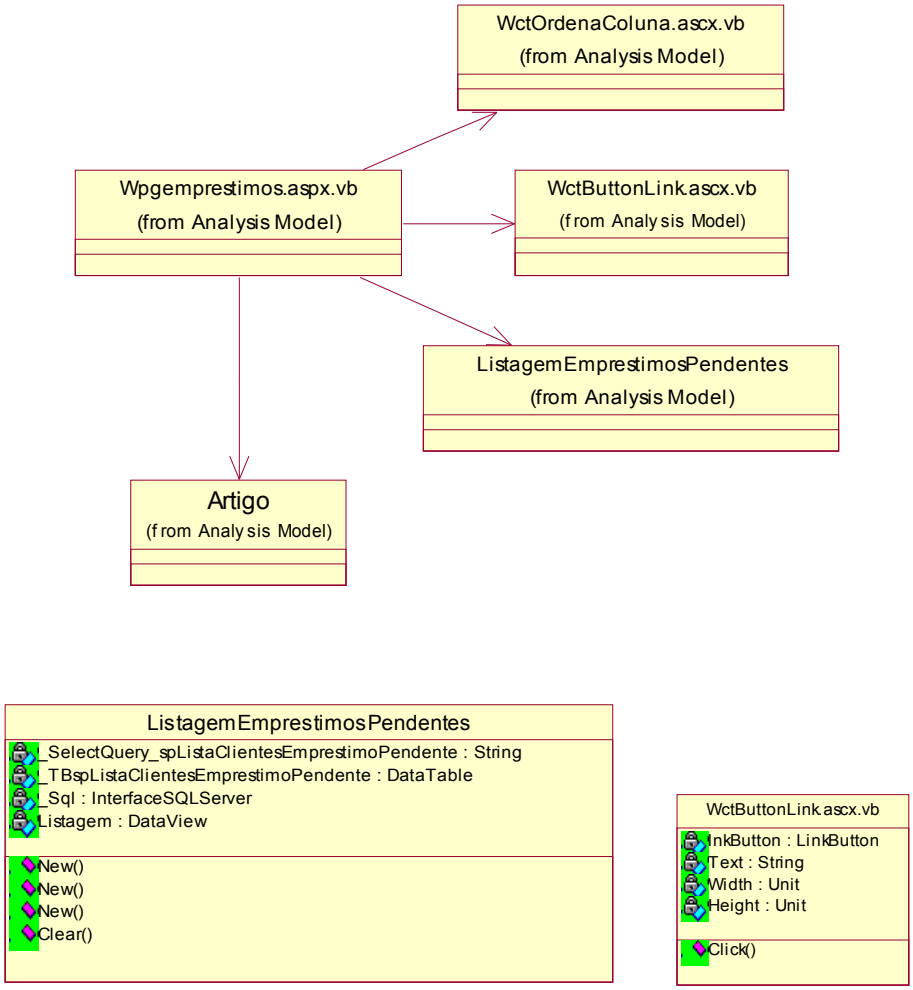




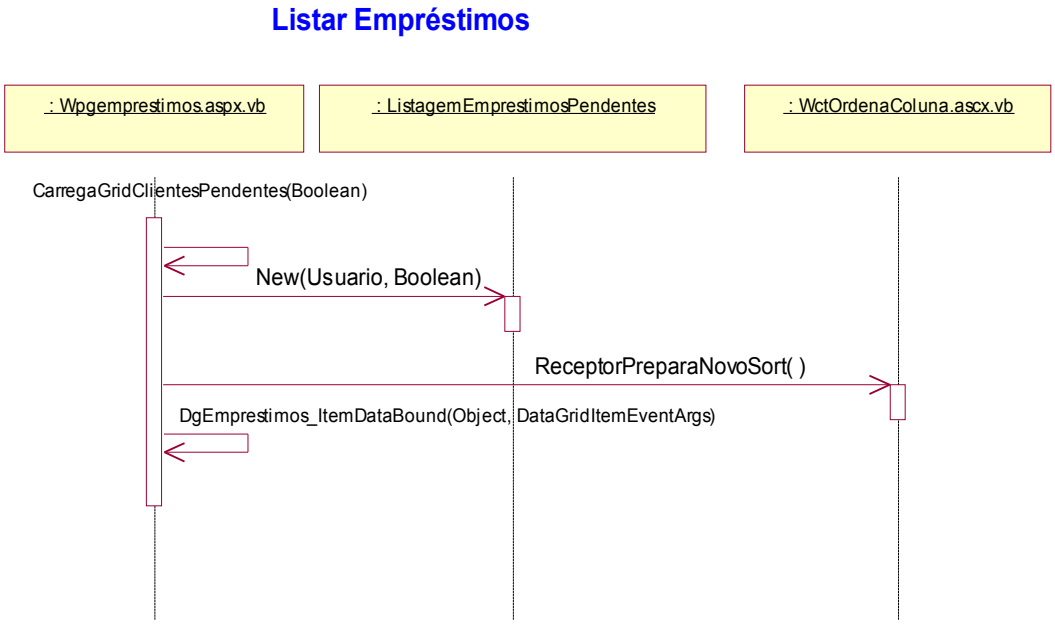
6.6 Pacote Empréstimo

6.6.1 Relacionamentos – Tela de Gerência de Empréstimos

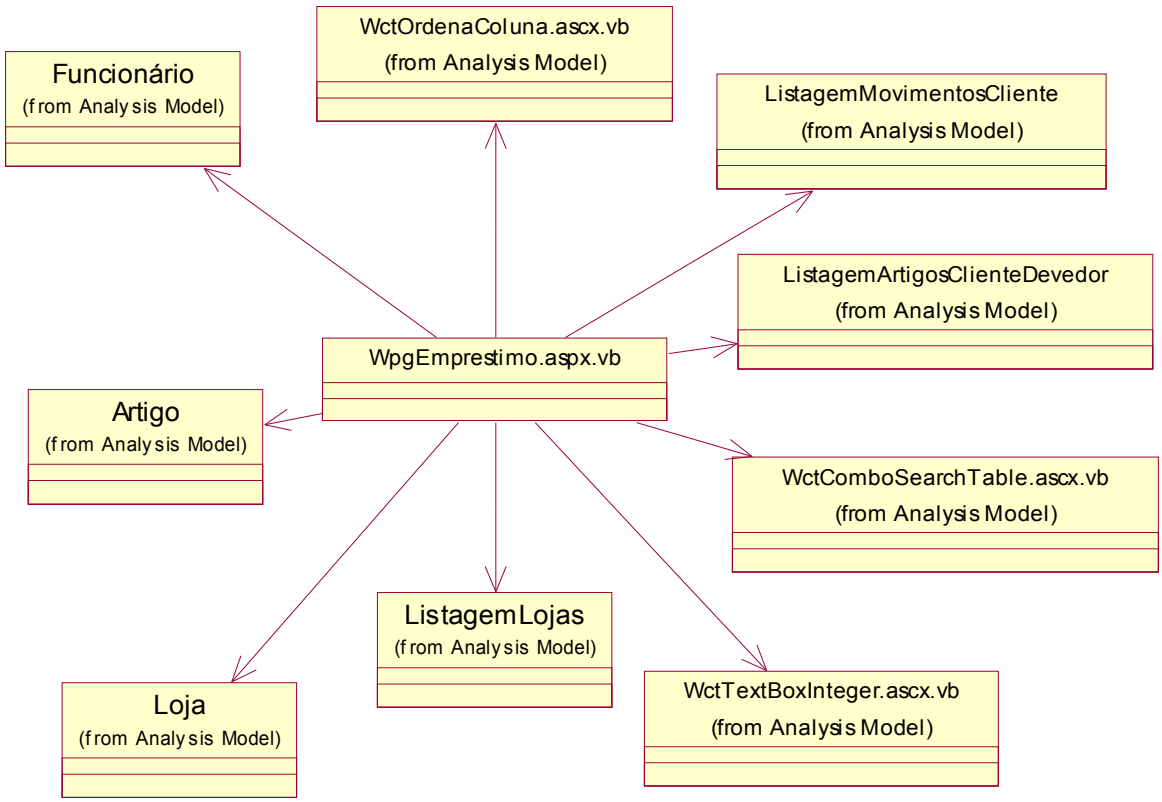

Wpgemprestimos.aspx.vb
Page_Load()
CarregaGridClientesPendentes()
DgEmprestimos_ItemDataBound()
WctOrdenaColunaReceptor_OrdenacaoPermitida()
DgEmprestimos_PageIndexChanged()
chkFiltrar_CheckedChanged()
WctButtonLink1_Click()
1
DgEmprestimos
1..*
WctOrdenaColuna.ascx
WpgEmprestimos.
aspx
WctHeader.ascx.vb
WctHeader.ascx

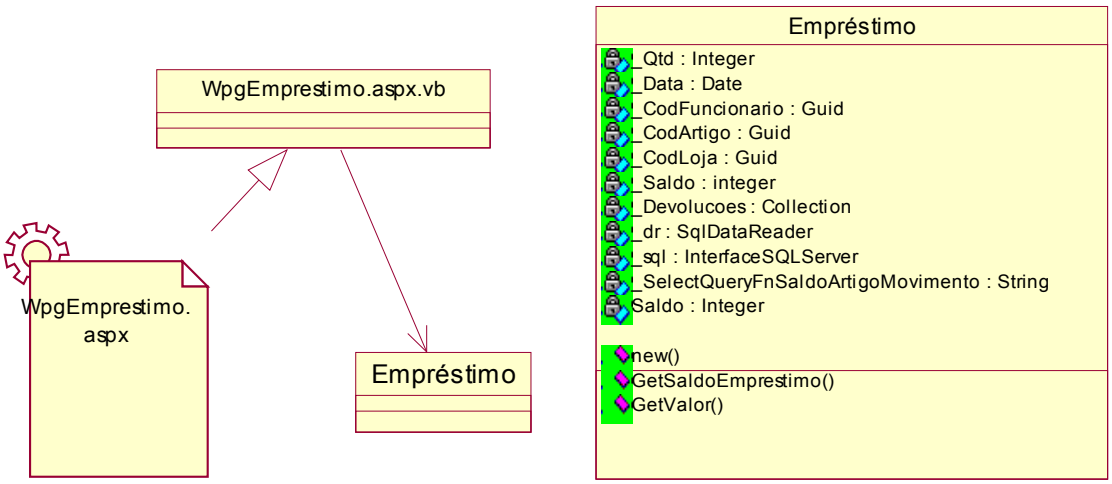


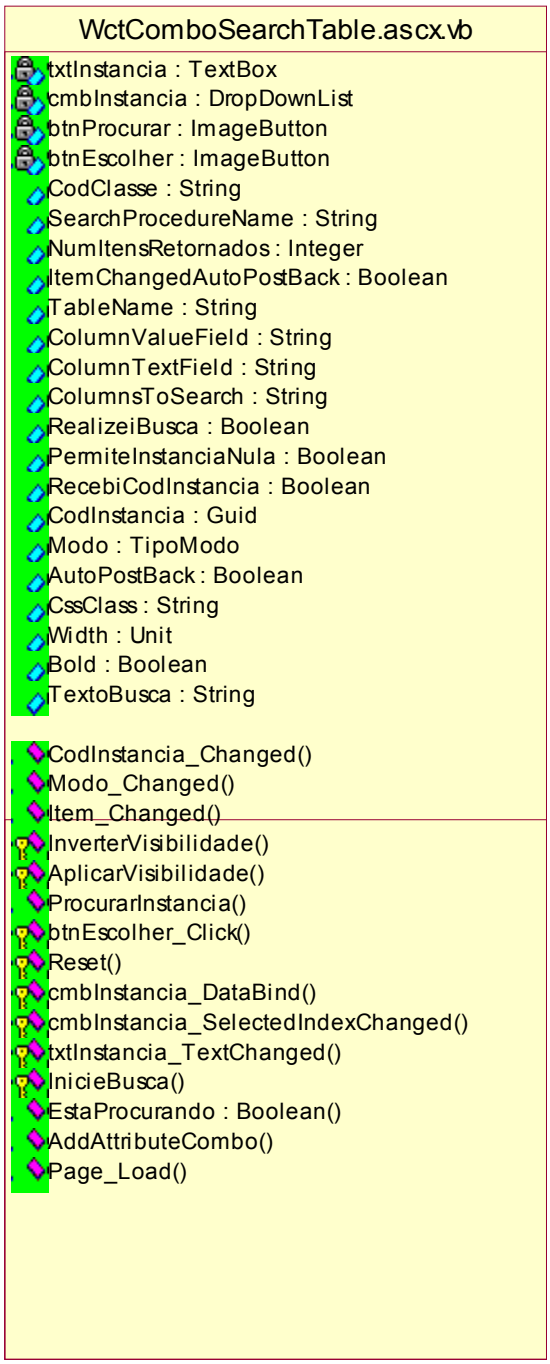
6.6.1.1 Diagramas de Seqüência

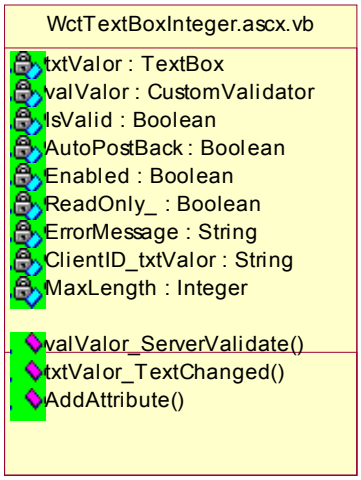


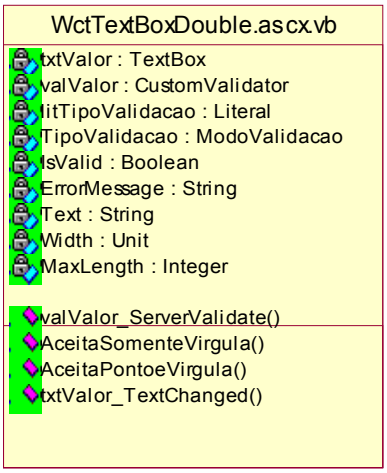
6.6.2 Relacionamentos – Tela de Cadastro de Empréstimo

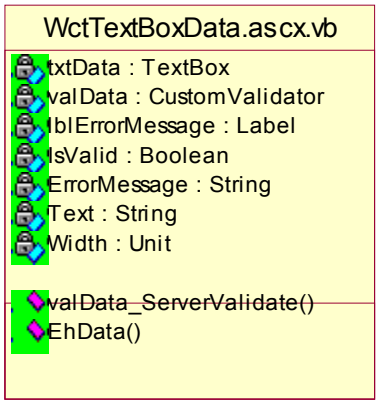


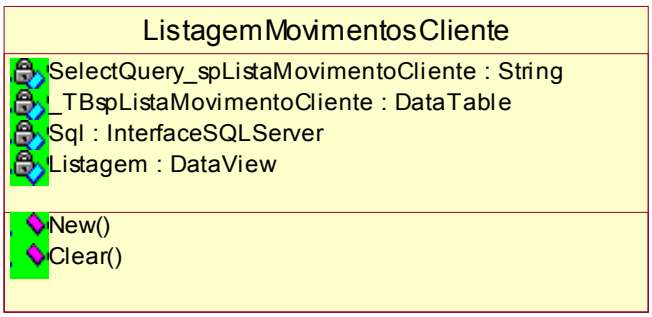

WpgEmprestimo.aspx.vb
WpgEmprestimo.aspx
Empréstimo
Empréstimo
_Qtd : Integer
_Data : Date
_CodFuncionario : Guid
_CodArtigo : Guid
_CodLoja : Guid
_Saldo : integer
_Devolucoes : Collection
_dr : SqlDataReader
_sql : InterfaceSQLServer
_SelectQueryFnSaldoArtigoMovimento : String
Saldo : Integer
new()
GetSaldoEmprestimo()
GetValor()

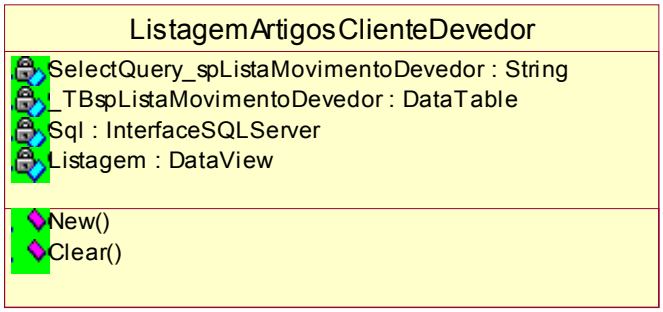

WctComboSearchTable.ascx.vb
txtInstancia : TextBox
cmbInstancia : DropDownList
btnProcurar : ImageButton
btnEscolher : ImageButton
CodClasse : String
SearchProcedureName : String
NumItensRetornados : Integer
ItemChangedAutoPostBack : Boolean
TableName : String
ColumnValueField : String
ColumnTextField : String
ColumnsToSearch : String
RealizeiBusca : Boolean
PermiteInstanciaNula : Boolean
RecebiCodInstancia : Boolean
CodInstancia : Guid
Modo : TipoModo
AutoPostBack : Boolean
CssClass : String
Width : Unit
Bold : Boolean
TextoBusca : String
CodInstancia_Changed()
Modo_Changed()
Item_Changed()
InverterVisibilidade()
AplicarVisibilidade()
ProcurarInstancia()
btnEscolher_Click()
Reset()
cmbInstancia_DataBind()
cmbInstancia_SelectedIndexChanged()
txtInstancia_TextChanged()
InicieBusca()
EstaProcurando : Boolean()
AddAttributeCombo()
Page_Load()


WctTextBoxInteger.ascx.vb
txtValor : TextBox
valValor : CustomValidator
IsValid : Boolean
AutoPostBack : Boolean
Enabled : Boolean
ReadOnly_ : Boolean
ErrorMessage : String
ClientID_txtValor : String
MaxLength : Integer
valValor_ServerValidate()
txtValor_TextChanged()
AddAttribute()

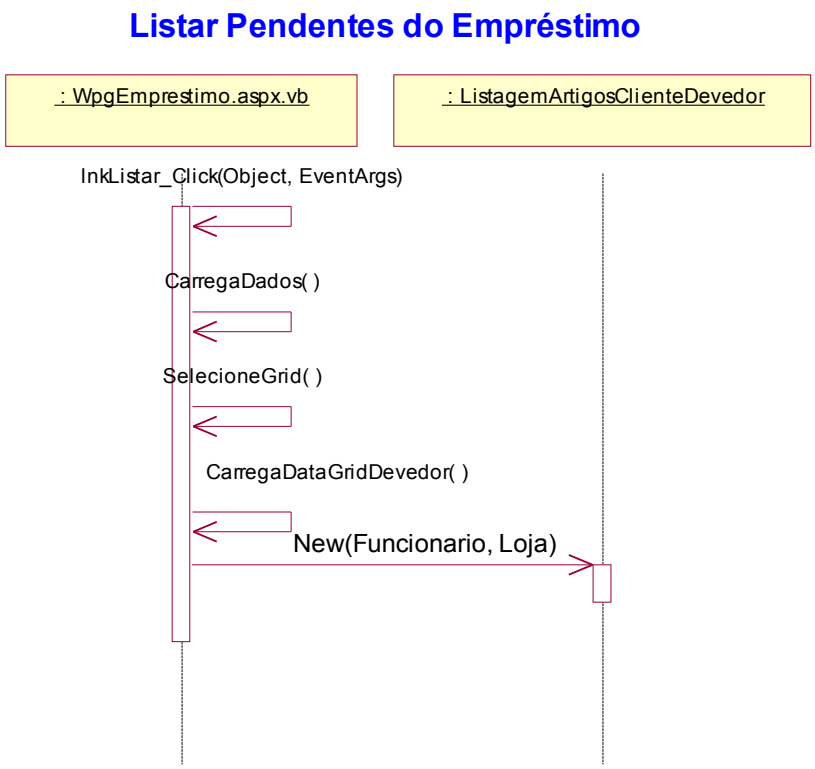

WctTextBoxDouble.ascx.vb
txtValor : TextBox
valValor : CustomValidator
litTipoValidacao : Literal
TipoValidacao : ModoValidacao
IsValid : Boolean
ErrorMessage : String
Text : String
Width : Unit
MaxLength : Integer
valValor_ServerValidate()
AceitaSomenteVirgula()
AceitaPontoeVirgula()
txtValor_TextChanged()


WctTextBoxData.ascx.vb
txtData : TextBox
valData : CustomValidator
lblErrorMessage : Label
IsValid : Boolean
ErrorMessage : String
Text : String
Width : Unit
valData_ServerValidate()
EhData()

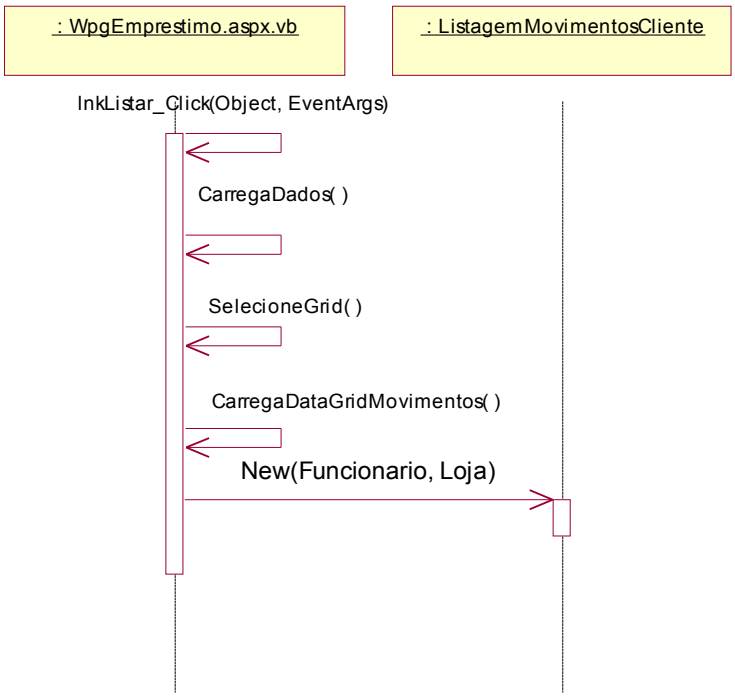

ListagemMovimentosCliente
SelectQuery_spListaMovimentoCliente : String
_TBspListaMovimentoCliente : DataTable
Sql : InterfaceSQLServer
Listagem : DataView
New()
Clear()


ListagemArtigosClienteDevedor
SelectQuery_spListaMovimentoDevedor : String
_TBspListaMovimentoDevedor : DataTable
Sql : InterfaceSQLServer
Listagem : DataView
New()
Clear()

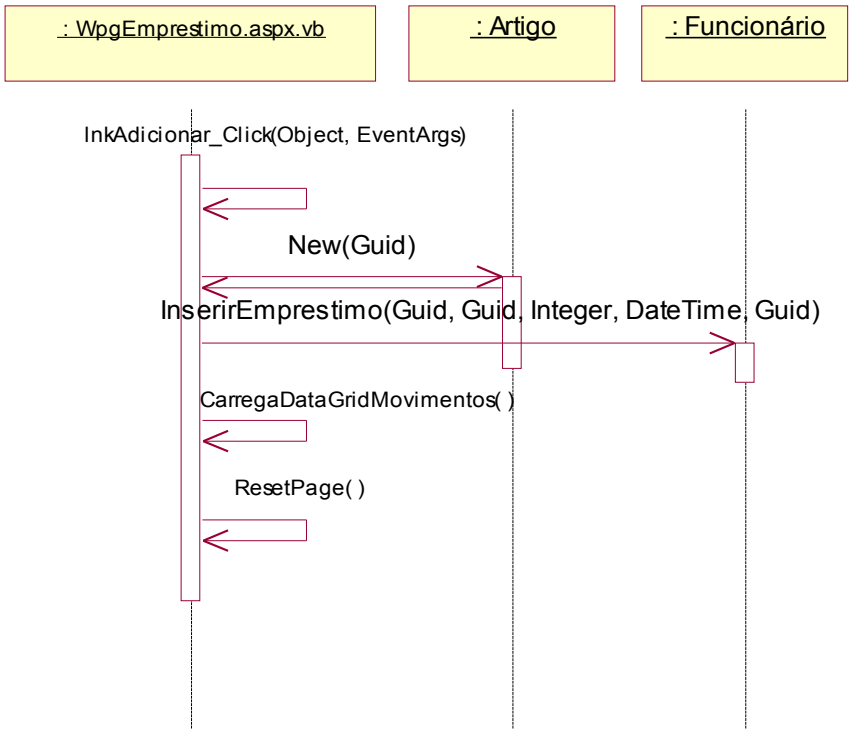
6.6.2.1 Diagramas de Seqüência

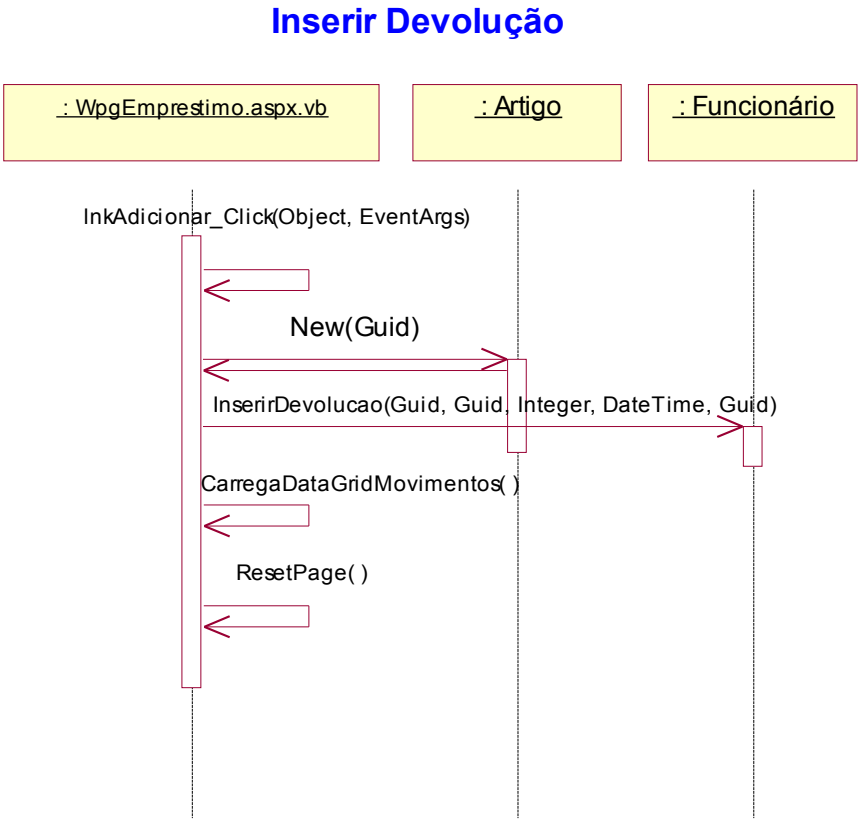


Listar Histórico do Empréstimo

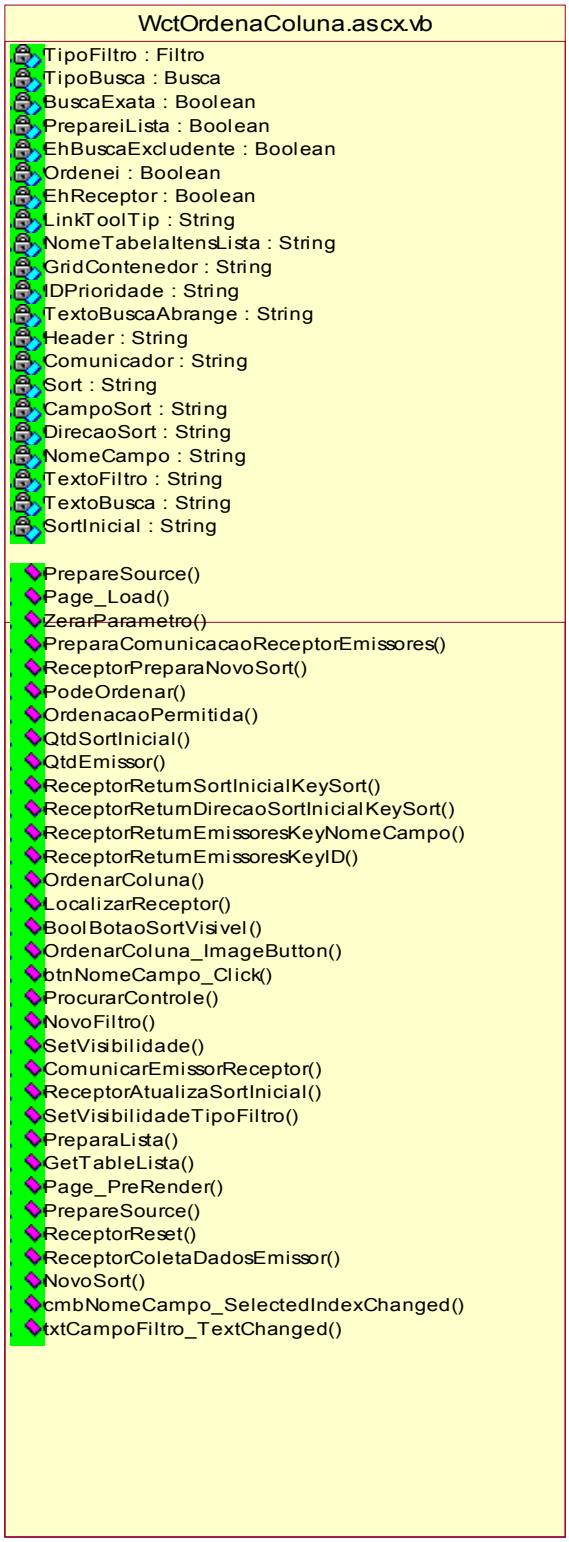


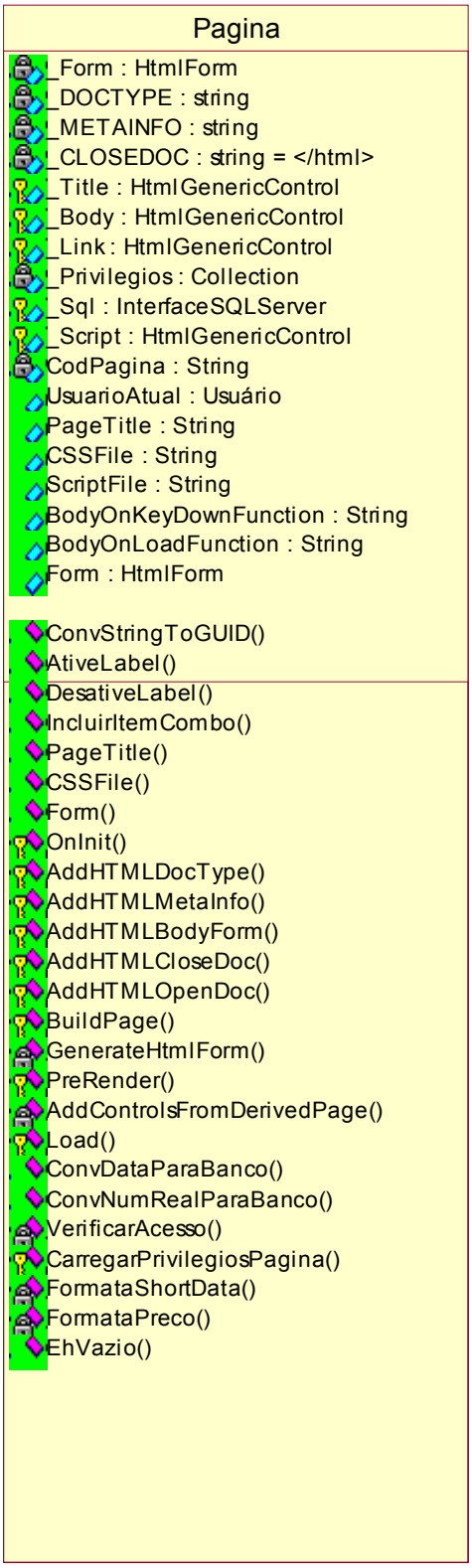
Inserir Empréstimo

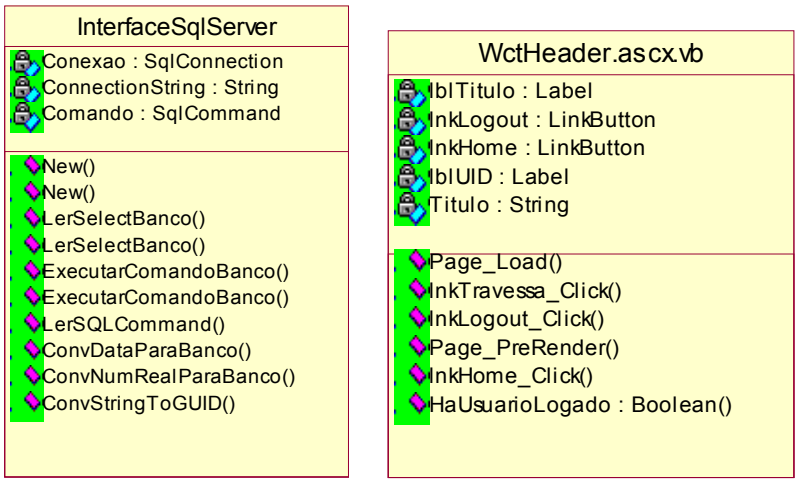


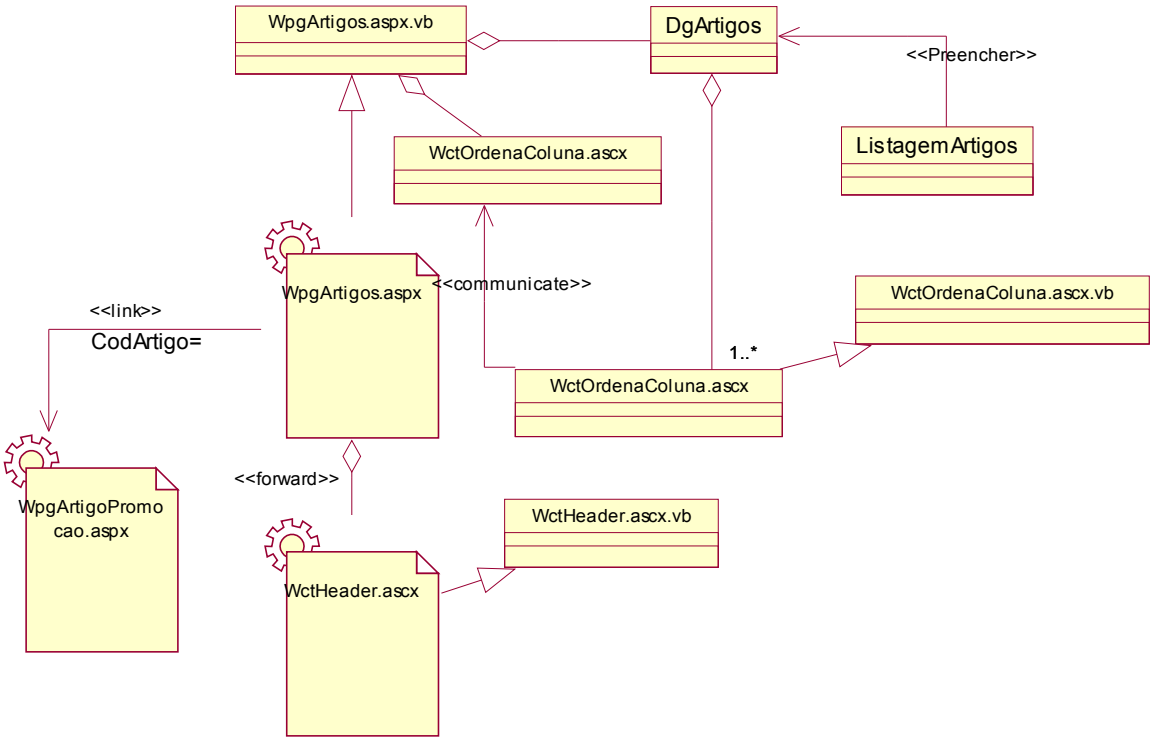


6.7 Classes Auxiliares


WctOrdenaColuna.ascx.vb
TipoFiltro : Filtro
TipoBusca : Busca
BuscaExata : Boolean
PrepareiLista : Boolean
EhBuscaExcludente : Boolean
Ordenei : Boolean
EhReceptor : Boolean
LinkToolTip : String
NomeTabelaItensLista : String
GridContenedor : String
IDPrioridade : String
TextoBuscaAbrange : String
Header : String
Comunicador : String
Sort : String
CampoSort : String
DirecaoSort : String
NomeCampo : String
TextoFiltro : String
TextoBusca : String
SortInicial : String
PrepareSource()
Page_Load()
ZerarParametro()
PreparaComunicacaoReceptorEmissores()
ReceptorPreparaNovoSort()
PodeOrdenar()
OrdenacaoPermitida()
QtdSortInicial()
QtdEmissor()
ReceptorRetumSortInicialKeySort()
ReceptorRetumDirecaoSortInicialKeySort()
ReceptorRetumEmissoresKeyNomeCampo()
ReceptorRetumEmissoresKeyID()
OrdenarColuna()
LocalizarReceptor()
BoolBotaoSortVisivel()
OrdenarColuna_ImageButton()
btnNomeCampo_Click()
ProcurarControle()
NovoFiltro()
SetVisibilidade()
ComunicarEmissorReceptor()
ReceptorAtualizaSortInicial()
SetVisibilidadeTipoFiltro()
PreparaLista()
GetTableLista()
Page_PreRender()
PrepareSource()
ReceptorReset()
ReceptorColetaDadosEmissor()
NovoSort()
cmbNomeCampo_SelectedIndexChanged()
txtCampoFiltro_TextChanged()

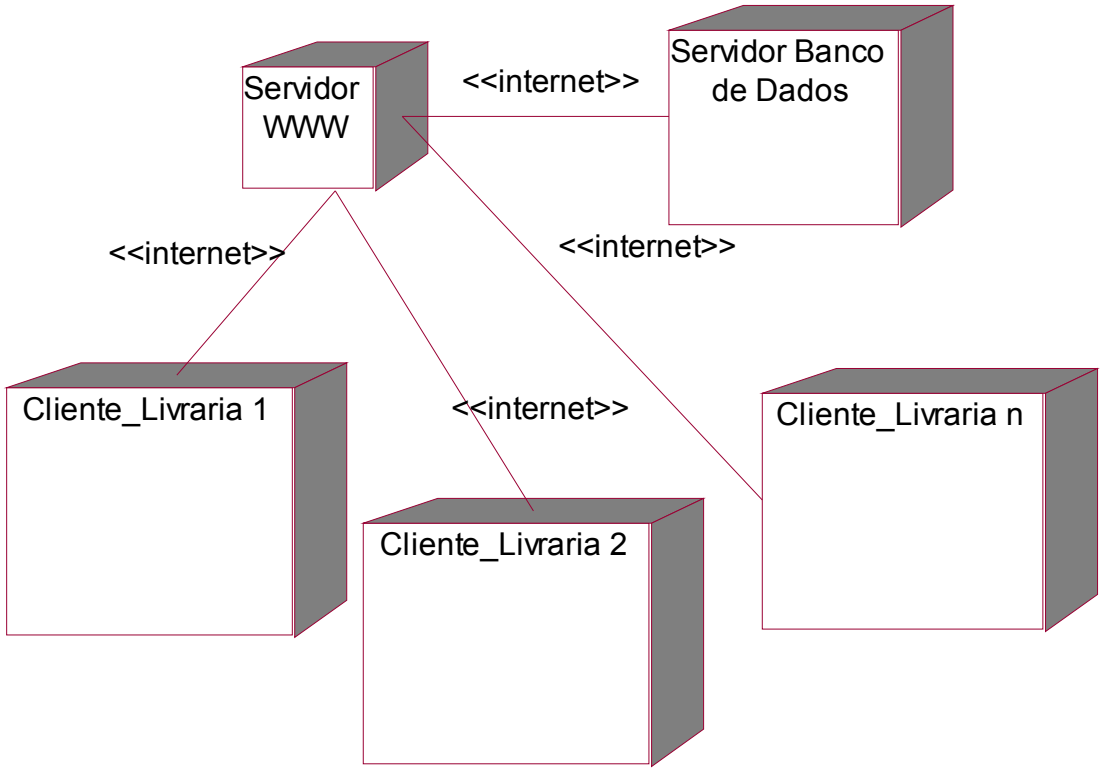

Pagina
_Form : HtmlForm
_DOCTYPE : string
_METAINFO : string
_CLOSEDOC : string = </html>
_Title : HtmlGenericControl
_Body : HtmlGenericControl
_Link : HtmlGenericControl
_Privilegios : Collection
_Sql : InterfaceSQLServer
_Script : HtmlGenericControl
CodPagina : String
UsuarioAtual : Usuário
PageTitle : String
CSSFile : String
ScriptFile : String
BodyOnKeyDownFunction : String
BodyOnLoadFunction : String
Form : HtmlForm
ConvStringToGUID()
AtiveLabel()
DesativeLabel()
IncluirItemCombo()
PageTitle()
CSSFile()
Form()
OnInit()
AddHTMLDocType()
AddHTMLMetaInfo()
AddHTMLBodyForm()
AddHTMLCloseDoc()
AddHTMLOpenDoc()
BuildPage()
GenerateHtmlForm()
PreRender()
AddControlsFromDerivedPage()
Load()
ConvDataParaBanco()
ConvNumRealParaBanco()
VerificarAcesso()
CarregarPrivilegiosPagina()
FormataShortData()
FormataPreco()
EhVazio()


InterfaceSqlServer
Conexao : SqlConnection
ConnectionString : String
Comando : SqlCommand
New()
New()
LerSelectBanco()
LerSelectBanco()
ExecutarComandoBanco()
ExecutarComandoBanco()
LerSQLCommand()
ConvDataParaBanco()
ConvNumRealParaBanco()
ConvStringToGUID()
WctHeader.ascx.vb
lblTitulo : Label
lnkLogout : LinkButton
lnkHome : LinkButton
lblUID : Label
Titulo : String
Page_Load()
lnkTravessa_Click()
lnkLogout_Click()
Page_PreRender()
lnkHome_Click()
HaUsuarioLogado : Boolean()


WpgArtigos.aspx.vb
DgArtigos
<<Preencher>>
WctOrdenaColuna.ascx
ListagemArtigos
WpgArtigos.aspx
<<communicate>>
WctOrdenaColuna.ascx.vb
<<link>>
CodArtigo=
1..*
WctOrdenaColuna.ascx
<<forward>>
WpgArtigoPromocao.aspx
WctHeader.ascx.vb
WctHeader.ascx

6.8 Diagrama de Desdobramento

O diagrama de desdobramento evidencia que o Sistema Livraria funciona como uma aplicação Intranet apenas em teoria, uma vez que todos os clientes se conectam ao servidor WWW através da "Internet". O acesso é restrito pela existência de senha.



7 Resultados

Aplicando uma iteração sobre o processo de desenvolvimento do Sistema Livraria obteve-se o acréscimo de funcionalidades ao sistema já em produção. São elas (resumidamente):

1 – Cadastro e edição de usuários do sistema;
2 - Cadastro e edição de promoções de artigos cadastrados no sistema;
3 - Cadastro e gerenciamento de empréstimos de artigos concedidos a funcionários da livraria

Ao longo do processo de desenvolvimento, foi observada a necessidade de realizar algumas adaptações ao ambiente de desenvolvimento, principalmente ao se constatar que alguns conceitos da orientação a objetos ainda não eram atendidos pela versão da linguagem utilizada. Porém, a implementação não inviabilizou um dos maiores objetivos da iteração que era o de iniciar a construção de uma classe base para todas as telas do sistema, unificando procedimentos e tornando mais fácil sua manutenção futura.

De modo geral, o resultado final correspondeu às expectativas geradas no início da iteração, sendo possível vislumbrar futuras ações que possam dar continuidade e incrementar o processo como um todo, como a incorporação de todos os conceitos da orientação a objetos numa futura versão do ambiente de desenvolvimento (por exemplo a sobrecarga de operadores) e a expansão das classes desenvolvidas para atender a novos requisitos sem afetar os já desenvolvidos.

8 Conclusão

Durante todo o estudo sobre o processo de desenvolvimento de uma aplicação Web e o desenvolvimento de uma iteração sobre um projeto existente sob o paradigma orientado a objetos, foi observado na prática todas as dificuldades que um projeto real implica e algumas conclusões importantes foram extraídas.

A primeira delas foi que o processo iterativo é o que melhor se adapta às condições reais de desenvolvimento de um sistema, que se inicia com o levantamento de requisitos e que ao longo do tempo – até determinado ponto - sofre todo tipo de alterações.

Por causa destas modificações a iteração desenvolvida ao longo do projeto se mostrou bastante satisfatória, sendo capaz de evoluir sem nenhum ônus ao restante do sistema já implementado, e isto se deve muito ao isolamento que as classes e procedimentos conferem à modelagem.

O resultado mais concreto pode ser representado pelo surgimento de uma maior disciplina ao se criar futuras telas para o sistema e a implementação da classe base “Pagina”, que permitirá um desenvolvimento mais acelerado das telas em futuras iterações.

Também foi observado - forçado pelas circunstâncias e por imposições do projeto - que nem sempre seguir à risca o paradigma orientado a objetos resulta na melhor solução para a implementação do sistema. No caso específico do sistema “Livraria”, trabalhar com grandes quantidades de dados e com o servidor de aplicação tendo que acessar via Internet a base de dados implica em algumas restrições de acesso à base de dados e uso de banda, que devem ser consideradas.

A solução implementada se baseou na utilização de classes auxiliares responsáveis exclusivamente por uma grande lista de dados cujo estado não altera o estado do sistema. Isto evitou que se instanciasse um objeto para cada linha retornada pelo banco de dados, abrindo

inúmeros processos quando apenas uma única “query” nos fornece a mesma resposta. Quando o contexto de utilização do sistema oferece a mudança de seu estado, as instâncias das classes surgem normalmente. Assim foram geradas classes como “Artigo”, “Funcionário”, etc.

Alterando outro conceito do paradigma orientado a objetos, optou-se por realizar um esvaziamento das classes, representada na transferência da responsabilidade de alterar o estado do sistema para os procedimentos e funções do banco de dados, cabendo aos métodos das classes apenas invocá-los adequadamente.

Esta decisão foi correta por realizar um isolamento entre o mundo conceitual (representado pelas classes) e o mundo da informação (representado pelas funções e "stored procedures" em SQL), permanecendo as assinaturas previamente acordadas como o elo de ligação. Desta forma, a equipe de desenvolvimento de uma aplicação Web pode ser composta por pessoas especialistas em banco de dados, mas não necessariamente especialistas em Web. Caberia a elas, então, desenvolver as mais eficientes funções e procedimentos que irão instanciar os objetos da modelagem e garantir a segurança das informações. Assim, os desenvolvedores Web podem se preocupar com a melhor utilização dos métodos criados durante a realização dos casos de uso do sistema.

Fica aprendida a lição que o desenvolvimento de sistemas Web com objetos exige um espírito mais conciliador com a visão estruturada, cabendo à equipe perceber em qual momento esta fronteira deve ser enfraquecida. Desta forma, um sistema Web orientado a objetos eficiente e leve na execução, desenvolvido com organização e disciplina tem grandes chances de ser conduzido com sucesso.

9 Bibliografia

1- Conallen, Jim - Desenvolvendo aplicações Web com UML, Rio de Janeiro, Campus, 2003, 476p;
2- Fowler, Martin - UML Essencial – Um breve guia para linguagem padrão de modelagem de objetos, Rio de Janeiro, Campus, 2003, 192p;
3- Pressman, Roger – Engenharia de Software, São Paulo, Makron Books, 1995, 1056;
4- Martin, James - Princípios de Análises e Projetos Baseados em Objetos, Rio de Janeiro, Campus, 1994, 512p.