Fundação Educacional do Município de Assis
IMESA - Instituto Municipal de Ensino Superior de Assis

SAMUEL BREDA DE LIMA

SISTEMA CONTROLE DE VENCIMENTO PARA SUPERMERCADOS

Assis
2016

SAMUEL BREDA DE LIMA

SISTEMA CONTROLE DE VENCIMENTO PARA SUPERMERCADOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas.

Orientador: Esp. Domingos de Carvalho Villela Junior
Área de Concentração: Desenvolvimento de Sistemas

Assis
2016

FICHA CATALOGRÁFICA

BREDA, Samuel Breda.
Sistema de controle vencimentos para supermercados /
Samuel Breda De Lima. Fundação Educacional do Município de Assis, 2016.
77 p.

Orientador: Esp. Domingos de Carvalho Villela Junior
Trabalho de Conclusão de Curso
Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Sistema Controle de Vencimento para Supermercados.
2. Programação.
3. Linguagem de programação Java.

SISTEMA CONTROLE DE VENCIMENTO PARA SUPERMERCADOS

SAMUEL BREDA DE LIMA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas.

Orientador: Esp. Domingos de Carvalho Villela Junior
Analisador: Dr. Almir Rogério Camolesi

Assis
2016

DEDICATÓRIA

Dedico este trabalho à minha família, amigos e todas as pessoas que acreditaram e apoiaram em meu sonhos e desejos, deles tirei forças necessária para que pudesse realizá-los.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela vida, saúde e por sempre estar em meu caminho, pois sem Ele, nada sou e nada em minha vida seria possível. Dele vem minha sabedoria e tudo que sou. Aos meus pais, Luiz Antonio da Silva e Maria de Jesus Breda da Silva, pelo amor incondicional e por sempre me apoiarem emocionalmente, financeiramente e em meus sonhos, sem eles não chegaria aqui. Ao meu orientador Domingos de Carvalho Villela Junior, e o Prof. Felipe Pazinatto pela orientação, durante todo o período deste trabalho e também por toda a caminhada acadêmica. E a todos meus amigos e demais familiares que sempre estão me apoiando.

Você tem que aprender a engatinhar antes de aprender a andar.

(Steven Tyler - Aerosmith 1994)

RESUMO

A tecnologia atual está em constante evolução, e a medida que evolui, aumenta a a capacidade de transmissão usando meios mais eficientes de comunicação, de coleta e segurança de dados.

Hoje em dia todos os tipos de empresas buscam o melhor e mais confiável software, mas particularmente nos últimos anos, a busca por um softwares especializados tem aumentado bastante devido à necessidade de modelar o comportamento do consumidor, recolhendo dados que as vezes parecem menos importantes.

Neste trabalho será apresentada a especificação e o desenvolvimento de um aplicativo desktop voltado para lojas de supermercado. O software irá permitir que a empresa evite a perda de lucratividade por conta de muitos produtos que tenham seus prazos de validade muito curto, o que em muitas das vezes dificulta os trabalho de verificação dos funcionários que analisam o vencimento desses produtos, devido a grande quantidade e variedade. A aplicação fará a informatização da loja, permitindo um melhor controle da data de validade de seus produtos.

Para o desenvolvimento do trabalho, foi feito um levantamento de todas as necessidades dos encarregados de loja e repositores de mercadorias, o que a loja espera do software. *T*ambém foi realizado um estudo das tecnologias utilizadas, a linguagem de programação Java, Eclipse Mars, e o banco de dados MariaDB.

ABSTRACT

Currently the technology lives in constant evolution, and as the evolution of tecnology grown up, grown up the demand for most efficient ways of communication, data collection and data security.

Nowadays all kinds of companies seeking for the best and most reliable software, more particularly in recent years, the search for a specificialized software has greatly increased due to the requirement to collect more and more data, even the one that seem less important.

This work presents the specification and development of a desktop application aimed at grocery stores. The software will allow the company to avoid the loss of profitability due to many products that have their very short expiration date, allowing the employees avoid the expiration of these products due the large quantities and varieties of it, allowing better control of the expiration date of their products.

For the development work was done a survey of all the needs of those in charge of store and shelf stockers of goods. Was also carried out a study of the technologies used, as the programming language Java, Eclipse Mars, and MariaDB database.

LISTA DE ILUSTRAÇÕES

LISTAS DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Produtos vencidos na loja, considerado um problema recorrente e grave, é hoje ainda mais sério. Além de significar prejuízo para a imagem da empresa e perda de vendas (o que não é pouco), representa também multas que variam de "R$ 400 mil a R$ 6 milhões e risco de detenção. Proprietários e gerentes de lojas flagrados só podem pagar fiança perante licença de uma autoridade judicial e não mais de autoridades policiais. Ou seja, podem ficar até 48 horas detidos à espera de liberação. A medida vale para todo o Brasil e prevista na Lei nº. 12.403/11.

Para evitar que os produtos encalhem no estoque e nas prateleiras, o que coloca em risco o prazo de validade, a primeira coisa a ser feita é programar bem as compras. Os pedidos não devem ser definidos exclusivamente a partir das vantagens comerciais oferecidas pelo fornecedor. "Se não for preciso adquirir grandes volumes, fuja das negociações de fim de mês, quando a indústria está louca para bater metas. Estoques muito altos colocam em risco o prazo de validade dos produtos.

O mais importante é levar em conta o giro dos produtos, o histórico de vendas, tirando do cálculo períodos de sazonalidade e promoções. Segundo o consultor, estimar a demanda hoje é um processo simples, que pode ser calculado por meio de softwares, exigindo apenas que os parâmetros sejam claros e os dados, confiáveis.

Uma proposta criada pela APAS (Associação Paulista de Supermercados)
prevê que, a partir de outubro deste ano, o consumidor que encontrar um produto vencido na gôndola que tenha ultrapassado a validade, tem direito de receber gratuitamente igual produto, dentro do prazo de validade. A medida vale para o Estado de São Paulo.

1.1 OBJETIVO

Alguns sistemas possuem recursos que avisam automaticamente quando o prazo de validade dos produtos está se esgotando. O controle de produtos próximos do prazo de validade não é uma tarefa fácil para supermercados e hipermercados. O objetivo é estimular a compra dos clientes de produtos alimentícios, garantir que o mesmo está levando para casa um produto com a data de validade dentro do prazo estimulado e evitar o constrangimento, o aborrecimento e insatisfação de levar um produto vencido para casa e, depois, ter de voltar ao estabelecimento para fazer a troca, que é um problema ainda pertinente em supermercados de nossa região.
Os principais objetivos do sistema proposto, é implantar um software que ofereça uma maior organização ao trabalho de repositores no que diz respeito o abastecimento dos produtos em relação a validade dessas mercadorias, O sistema também irá ajudar a princípio os supermercados a oferecerem produtos com a melhor qualidade e dentro dos prazos.

1.2 PÚBLICO-ALVO

O sistema será disponível para os profissionais de supermercados tais como: Gerentes, Enc. de Loja, além dos repositores de mercadorias. Também a profissionais de mercearias, e miní mercados em geral, com a finalidade de facilitar o controle de datas de vencimentos de produtos em estoque e principalmente os mais próximos do consumidor, que são os produtos abastecidos nas prateleiras.

1.3 JUSTIFICATIVA

Espera-se que este sistema contribua de forma efetiva para os futuros estudos e decisões tomadas pelos profissionais do mercado alimentício, com o fornecimento ágil de informações valiosas como a validade dos alimentos, que irá com certeza acarretar em maiores lucros a empresa.

O sistema proposto também é simples e de fácil manuseio para os funcionários, além de gerar relatórios eficientes e de fácil leitura, para que a tarefas realizadas diariamente pelos profissionais sejam mais produtivas.

Com o desenvolvimento de um trabalho que ainda é bem pouco disponível nesse segmento na região, torna-se imprescindível para uma empresa do ramo alimentício mais especificamente supermercados, adotar o uso de um software que lhe forneça de maneira eficiente e organizada a validade de seus produtos vendidos.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido em 7 capítulos que serão apresentados a seguir. O primeiro capítulo apresenta o objetivo, justificativa e o público-alvo para o desenvolvimento da proposta de trabalho.

No capítulo **1.5** são abordadas as metodologias de análise do projeto, e as ferramentas utilizadas no sistema.

O capítulo **2.1** apresenta as etapas e especificações do software, levantamento de requisitos, lista de eventos, caso de uso e os principais diagramas UML (classe, sequência, atividade e entidade e relacionamento).

O capítulo **2.9** descreve a WBS – *Work Breakdown Structure*, sequenciamento das atividades e o orçamento do software.

O capítulo **3.2** apresenta a implementação do sistema, exibindo um detalhamento sobre a aplicação desenvolvida assim como a distribuição dos pacotes, e organização do projeto.

O capítulo **4.2** apresenta a implementação da interface criada para interagir com o usuário final.

No ultimo capítulo são apresentados a conclusão do trabalho, cronograma atual do trabalho e as referências utilizadas.

1.5 MÉTODO DE DESENVOLVIMENTO

1.6 METODOLOGIAS DE ANÁLISE

Na análise foram utilizados os conceitos de *Unified Modeling Language UML(UML). UML* é uma linguagem de modelagem, que proporciona todos os ícones de desenho necessários para capturar a maioria dos conceitos ou mecanismo que são valiosos para a resolução dos problemas reais de negócios(regras de negócio). Ela provê todos os diagramas que são vitais para a documentação dos modelos do sistema. Basicamente, a UML permite que desenvolvedores visualizem os produtos em diagramas padronizados. (LEE, Tepfenhart, 2001).

A ferramenta utilizada para criar os diagramas de UML foi o *Violet UML Editor*[1], que é um editor UML (GPL License, 2014). E para modelagem do MER (Modelagem de Entidade e Relacionamento) foi utilizado a ferramenta Web Vertabello.[2]

1.7 DESENVOLVIMENTO DO SISTEMA

A ferramenta escolhida para o desenvolvimento do sistema foi o Eclipse Mars[3], que utilizará a linguagem Java[4].

O Eclipse é uma plataforma de desenvolvimento de software livre extensível, baseada em Java. Por si só, é simplesmente uma estrutura e um conjunto de serviços para desenvolvimento de aplicativos de componentes de plug-in. Felizmente, o Eclipse vem com um conjunto padrão de plug-ins, incluindo as amplamente conhecidas Ferramentas de Desenvolvimento Java (JDT). (O QUE É O ECLIPSE?, IBM, 2012).

1 http://alexdp.free.fr/violetumleditor/page.php - Acesso em 29, jan.2016
2 http://www.vertabelo.com/ - Acesso em 29, jan.2016
3 https://eclipse.org/mars/ - Acesso em 27, jul.2016
4 http://www.oracle.com/br/java/overview/index.html/ - Acesso em 27, jul.2016

Java é uma linguagem de programação orientada a objetos feita na Sun Microsystems, hoje Oracle Corporation, lançada em 1995.
A semelhança da sintaxe do Java com C e C++ não é coincidência, derivou dessas linguagens mesmo. Porém, programar em Java é mais simples, pois é alto nível. Isso quer dizer que não nos preocupamos tanto com detalhes baixo nível, como memória, processamento, ponteiros, lixo etc. O Java já provém um gerenciamento automático de memória e um coletor de lixo, que facilitam a vida do desenvolvedor, mas consomem mais processamento. A diferença do Java é que os programas não são compilados diretamente na arquitetura do computadores. Ao invés disso, roda na JVM - Java Virtual Machine, uma máquina virtual, e esta é implementada nos mais diversos dispositivos, o que torna o Java referência quando o assunto é portabilidade. O Java é bastante flexível por conta da possibilidade de expansão através das bibliotecas, ou APIs, além das extensões do Java, voltadas especificamente para desenvolvimento de aplicações para desktop, para celulares, para empresas, para áudio, para gráicos 3D, banco de dados, para aplicações de Internet, criptografia, computação/sistemas distribuídos, linguagem de marcação, infra estrutura peer-to-peer e várias outras (COMECE A PROGRAMAR: A LINGUAGEM DE PROGRAMAÇÃO JAVA, Programação Progressiva, 2012).

1.8 PERSISTÊNCIA DOS DADOS

Hibernate[5] é um framework para realizar o mapeamento objeto relacional(ORM) escrito na linguagem java, onde seu principal objetivo é diminuir a complexidade envolvido no desenvolvimento de aplicações que necessitam trabalhar com banco de dados relacional, onde ele realiza a intermediação entre o banco de dados e sua aplicação, poupando o desenvolvedor de ter que se preocupar com instruções SQL para recuperar ou persistir os dados do seu software. O hibernate realiza o mapeamento do objeto relacional, ou seja, as tabelas do seu banco de dados são representadas através de classes na sua aplicação e as operações de recuperação e persistência dos dados são realizadas através de métodos do hibernate, sendo assim, o programador não precisa de se preocupar com instruções SQL como selects, join e etc, sendo o framework capaz até de resolver as peculiaridades que cada SGDB impõe (O QUE É HIBERNATE?, Naison Souza, 2012).

5 http://hibernate.org/orm/ - Acesso em 27, jul.2016

1.9 ARMAZENAMENTO E VISUALIZAÇÃO DOS DADOS

Para o armazenamento dos dados, foi utilizado o banco de dados MariaDB 10[6]
MariaDB é um banco de dados que surgiu como fork do MySQL, criado pelo próprio fundador do projeto após sua aquisição pela Oracle. A intenção principal do projeto é manter uma alta fidelidade com o MySQL. O líder do MariaDB é Michael 'Monty' Widenius, o fundador do MySQL e da Monty Program AB. Para conseguir isso, o Programa Monty trabalha para contratar os melhores e mais brilhantes desenvolvedores do setor, trabalhar em estreita cooperação com a maior comunidade de usuários e desenvolvedores no verdadeiro espírito do software livre e open source. MariaDB é um avançado substituto para o MySQL e está disponível sob os termos da licença GPL v2[7] (HISTÓRIA MARIADB, Elderstroparo, 2015).

2 GERAÇÃO DE RELATÓRIOS

Para a geração de relatórios foi utilizado o JasperReports em conjunto com o IReport 5.6[8] O iReport é uma ferramenta desenvolvida pela mesma empresa do JasperReports, a JasperForge, e por isso é muito comum ver os dois sendo usados em conjunto. Uma das dificuldades ao trabalhar com os relatórios, está na definição do layout. É complicado escrever o layout totalmente em XML[9], sem ter que se aprofundar em todas as *tags* e atributos possíveis, e além disso posicionar todos os elementos corretamente. Na prática, é muito raro alguém editar o JRXML manualmente, e sim apenas para fazer alguns pequenos ajustes quando necessários. O processo normal é utilizar alguma ferramenta para gerar o JRXML automaticamente, e o iReport é utilizado com esse propósito.

6 https://mariadb.org/ - Acesso em 29, jan.2016
7 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html - Acesso em 05, fev.2016
8 http://community.jaspersoft.com/project/ireport-designer - Acesso em 29, jan.2016
9 http://www.w3schools.com/xml/ - Acesso em 05, fev.2016

2.1 ESTRUTURA DE DESENVOLVIMENTO DO SISTEMA

2.2 MAPA MENTAL

Mapa Mental é uma ferramenta que permite a memorização, organização e representação da informação com o propósito de facilitar os processos de aprendizagem, administração e planejamento organizacional, assim como, a tomada de decisão.

A técnica dos Mapas Mentais foi desenvolvida pelo britânico Tony Buzan com o objeto de fortalecer as conexões sinápticas que têm lugar entre os neurônios do córtex cerebral e que fazem praticamente todas as atividades intelectuais do ser humano.

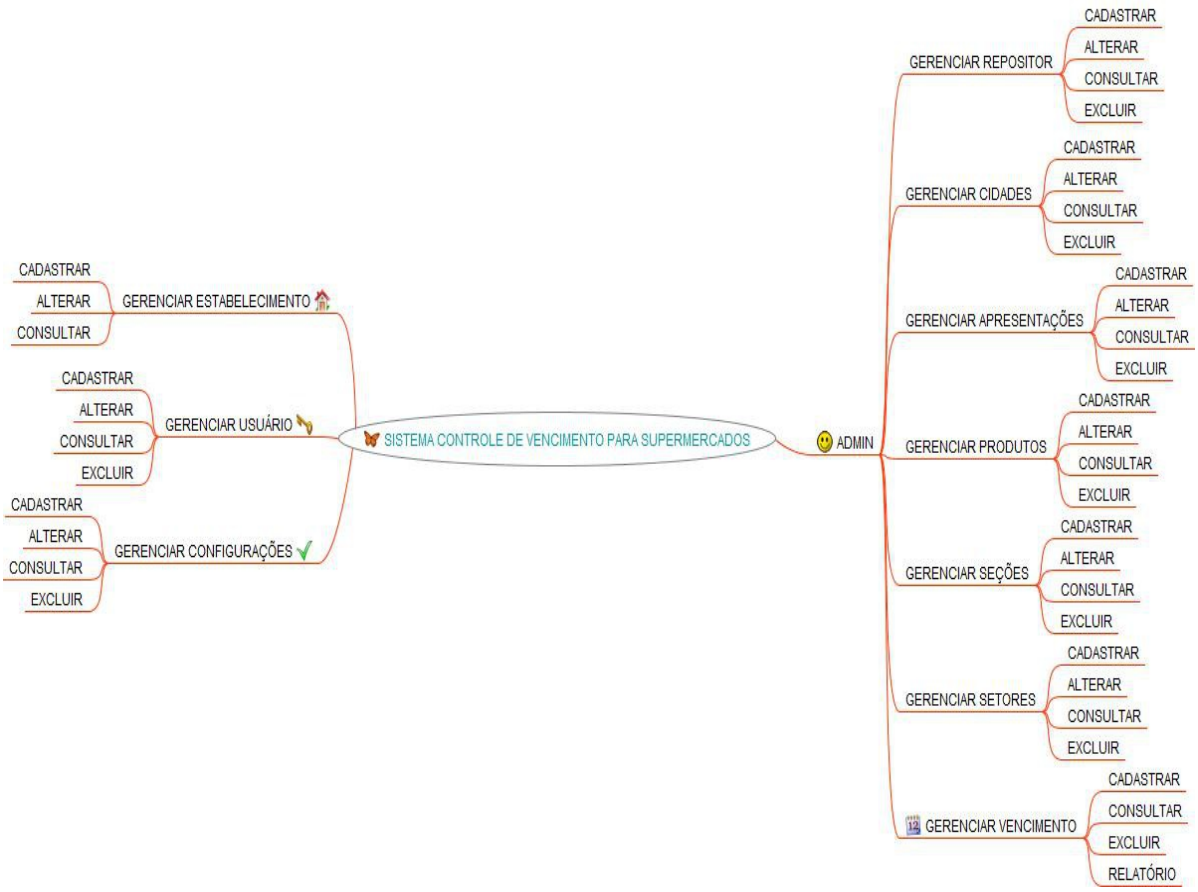

Figura 1- Mapa Mental

2.3 LISTA DE EVENTOS

Baseado no mapa mental, os seguintes eventos foram encontrados para melhorar o sistema.

1. Efetuar Login (Controle de acesso)
2. Cadastrar Estabelecimento
3. Cadastrar Usuário
4. Cadastrar Configuração
5. Cadastrar Repositor
6. Cadastrar Cidade
7. Cadastrar Apresentação
8. Cadastrar Produto
9. Cadastrar Seção
10. Cadastrar Setor
11. Cadastrar Produtos a vencer
12. Consultar Estabelecimento
13. Consultar Usuário
14. Consultar Configuração
15. Consultar Repositor
16. Consultar Cidade
17. Consultar Apresentação
18. Consultar Produto
19. Consultar Seção
20. Consultar Setor
21. Consultar Produtos a vencer

22. Excluir Repositor
23. Excluir Cidade
24. Excluir Apresentação
25. Excluir Produto
26. Excluir Seção
27. Excluir Setor
28. Excluir Produtos a vencer
29. Gerar Relatório de Produtos a vencer
30. Gerar Relatório de Repositores cadastrados
31. Gerar Reatório de Produtos a vencer por periodo
32. Definir a senha de usuário para utilização do sistema

2.4 DIAGRAMAS DE CASO DE USO

Casos de uso especificam o comportamento do sistema ou parte(s) dele e escrevem a funcionalidade do sistema desempenhada pelos atores. Você pode imaginar um caso de uso como um conjunto de cenários, onde cada cenário é uma sequência de passos a qual descreve uma interação entre um usuário e o sistema. Os casos de uso são representados em forma de elipse.

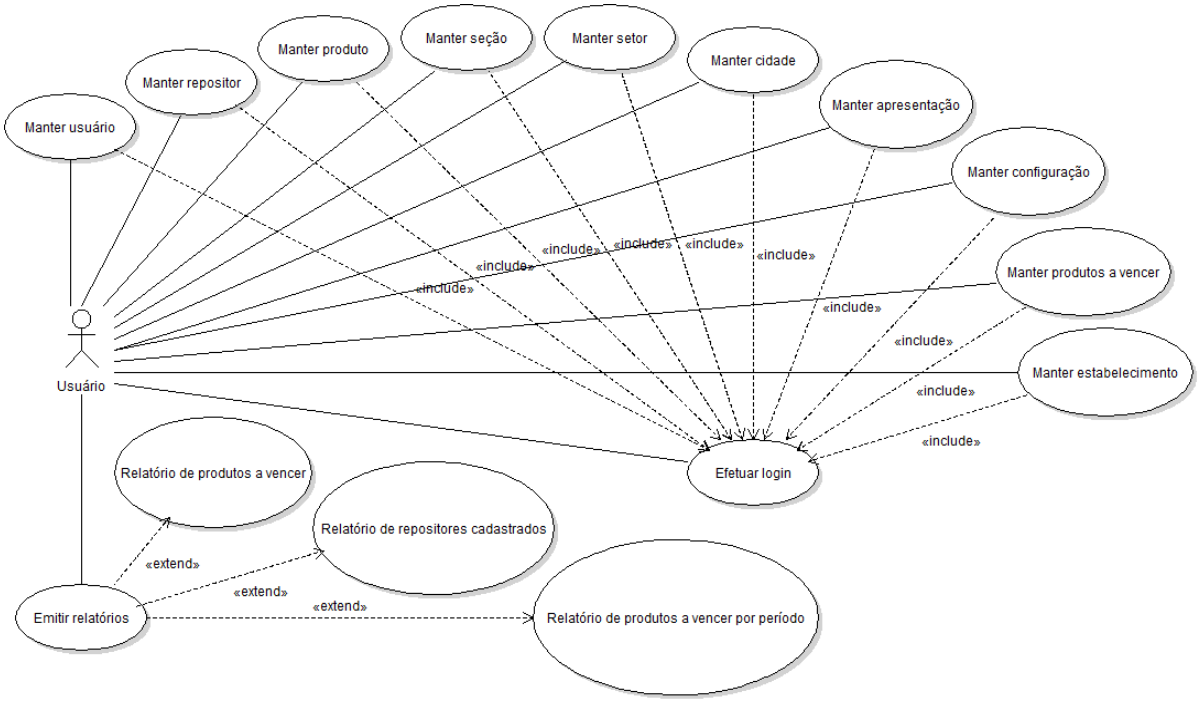

Figura 2 caso de uso geral – Administrador / usuário

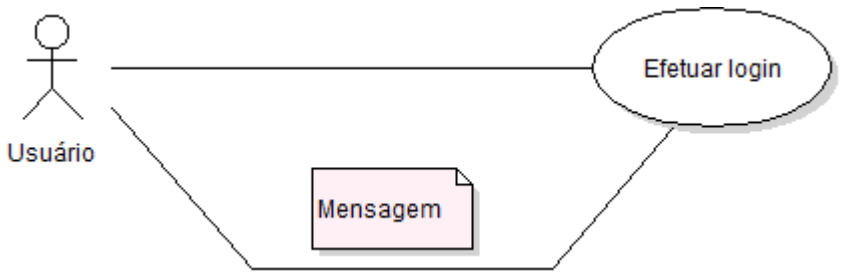

Figura 3 diagrama de caso de uso 1 Efetuar Login

Nome do Caso de Uso 1	Efetuar Login
Atores	Administrador / usuário.
Pré-condições	Não existe.
Cenário Principal	1 - O sistema solicita os dados para efetuar o controle de acesso. 2 - O usuário informa os dados. 3 - O usuário confirma os dados de controle de acesso. 4 - O sistema recupera os dados informados pelo usuário. 5 - O sistema valida os dados. 6 - Caso os dados sejam válidos o usuário se conecta ao sistema.
Cenários Alternativos	Não existe.
Casos de Testes	1 - Se os dados informados estiverem corretos, executa a operação solicitada. 2 - Se os dados informados estiverem incorretos, cancela a operação e exibe uma mensagem de alerta.

Tabela 1 Efetuar Login

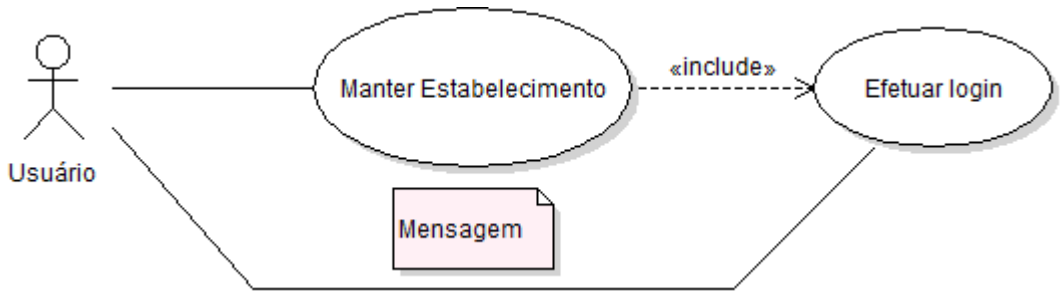

Figura 4 diagrama de caso uso 2 de Manter Estabelecimento

Nome do Caso de Uso 2	Manter Estabelecimento.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados do estabelecimento. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados do estabelecimento.
Cenários Alternativos	1 - O usuário pode excluir os dados do estabelecimento. 2 - O usuário pode alterar os dados do estabelecimento. 3 - O usuário pode consultar os dados do estabelecimento cadastrado.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 2 Manter Estabelecimento

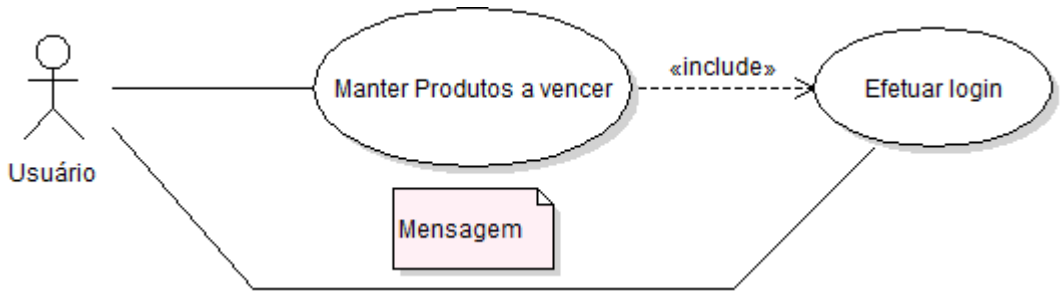

Figura 5 diagrama de caso uso 3 de Manter Produtos a Vencer

Nome do Caso de Uso 3	Manter Produtos a Vencer.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados dos Produtos a vencer 2 – O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados dos Produtos a vencer.
Cenários Alternativos	1 - O usuário pode excluir os dados dos Produtos a vencer. 2 - O usuário pode alterar os dados dos Produtos a vencer. 3 - O usuário pode pesquisar os Produtos a vencer cadastrados.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 3 Manter Produtos a Vencer

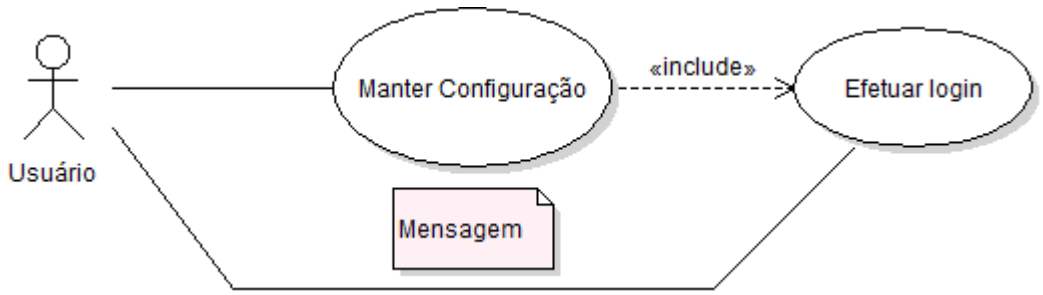

Figura 6 diagrama de caso uso 4 de Manter Configuração

Nome do Caso de Uso 4	Manter Configuração.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados da configuração. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados da apresentação.
Cenários Alternativos	1 - O usuário pode excluir os dados da apresentação. 2 - O usuário pode alterar os dados da configuração. 3 - O usuário pode consultar os dados da configuração cadastrada.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 4 Manter Configuração

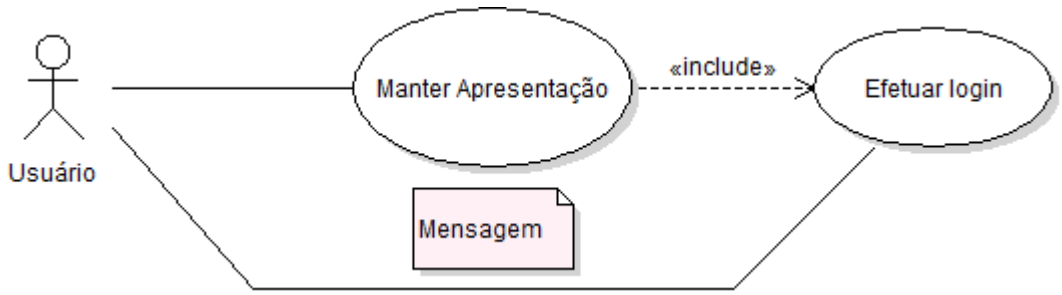

Figura 7 diagrama de caso uso 5 de Manter Apresentação

Nome do Caso de Uso 5	Manter Apresentação.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados da apresentação. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados da apresentação.
Cenários Alternativos	1 - O usuário pode excluir os dados da apresentação. 2 - O usuário pode alterar os dados da apresentação. 3 - O usuário pode pesquisar as apresentações cadastradas.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 5 Manter Apresentação

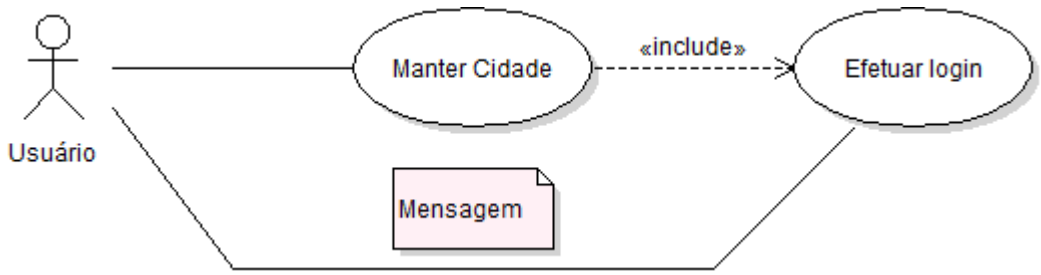

Figura 8 diagrama de caso uso 6 de Manter Cidade

Nome do Caso de Uso 6	Manter Cidade.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados da cidade. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados da apresentação.
Cenários Alternativos	1 - O usuário pode excluir os dados da cidade. 2 - O usuário pode alterar os dados da cidade. 3 - O usuário pode pesquisar as cidades cadastradas.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 6 Manter Cidade

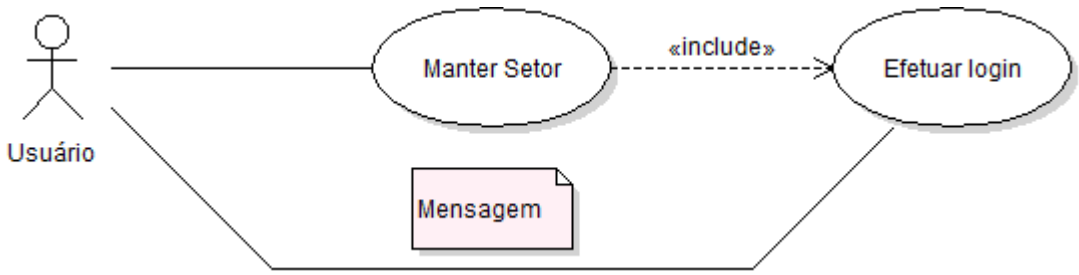

Figura 9 diagrama de caso uso 7 de Manter Setor

Nome do Caso de Uso 7	Manter Setor.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login
Cenário Principal	1 - O usuário informa os dados da seção. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados da seção.
Cenários Alternativos	1 - O usuário pode excluir os dados da seção. 2 - O usuário pode alterar os dados da seção. 3 - O usuário pode pesquisar as seções cadastradas.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 7 Manter Setor

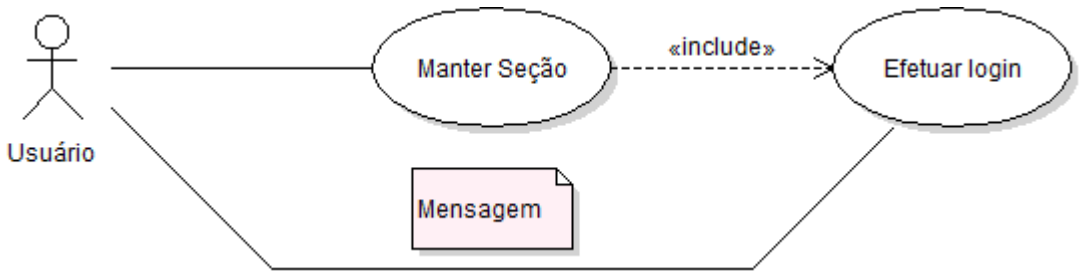

Figura 10 diagrama de caso uso 8 de Manter Seção

Nome do Caso de Uso 8	Manter Seção.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login
Cenário Principal	1 - O usuário informa os dados da seção. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados da seção.
Cenários Alternativos	1 - O usuário pode excluir os dados da seção. 2 - O usuário pode alterar os dados da seção. 3 - O usuário pode pesquisar as seções cadastradas.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 8 Manter Seção

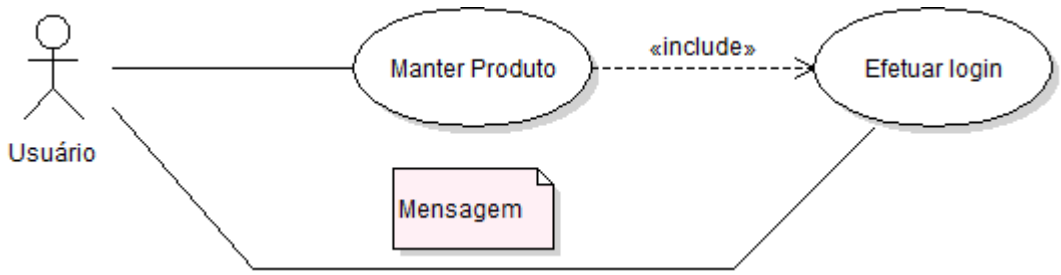

Figura 11 diagrama de caso uso 9 de Manter Produto

Nome do Caso de Uso 9	Manter Produto.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login
Cenário Principal	1 - O usuário informa os dados do produto. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados do produto.
Cenários Alternativos	1 - O usuário pode excluir os dados do produto. 2 - O usuário pode alterar os dados do produto. 3 - O usuário pode pesquisar os produtos cadastrados.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 9 Manter Produto

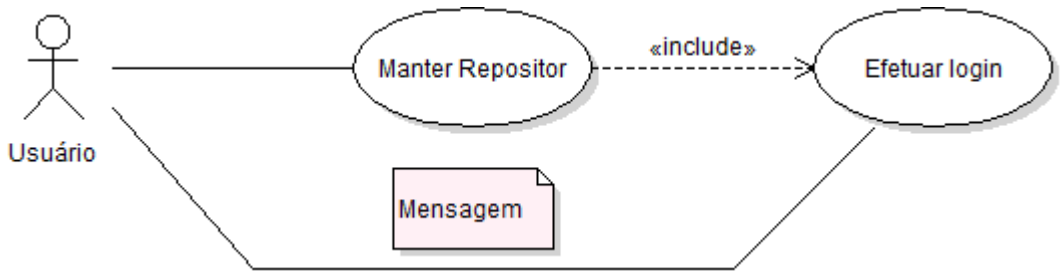

Figura 12 diagrama de caso uso 10 de Manter Repositor

Nome do Caso de Uso 10	Manter Repositor.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O usuário informa os dados do repositor. 2 - O usuário seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados do repositor.
Cenários Alternativos	1 - O usuário pode excluir os dados do repositor. 2 - O usuário pode alterar os dados do repositor. 3 - O usuário pode pesquisar os repositores cadastrados.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 10 Manter Repositor

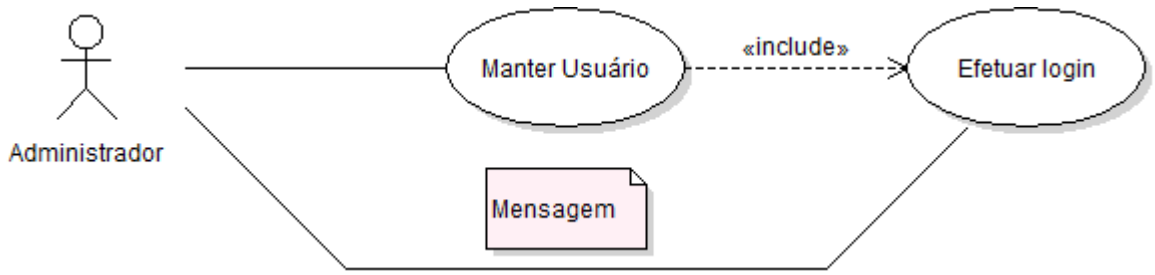

Figura 13 diagrama de caso uso 11 de Manter Usuário

Nome do Caso de Uso 11	Manter Usuário.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O administrador informa os dados do usuário. 2 - O administrador seleciona a opção “Salvar”. 3 - O sistema valida os dados informados. 4 - O sistema emite mensagem de sucesso. 5 - O sistema cadastra os dados do usuário.
Cenários Alternativos	1 - O administrador pode excluir os dados do repositor. 2 - O administrador pode alterar os dados do usuário. 3 - O administrador pode pesquisar os usuários cadastrados.
Casos de Testes	O sistema verifica se todos os campos obrigatórios foram preenchidos.

Tabela 11 Manter Usuário

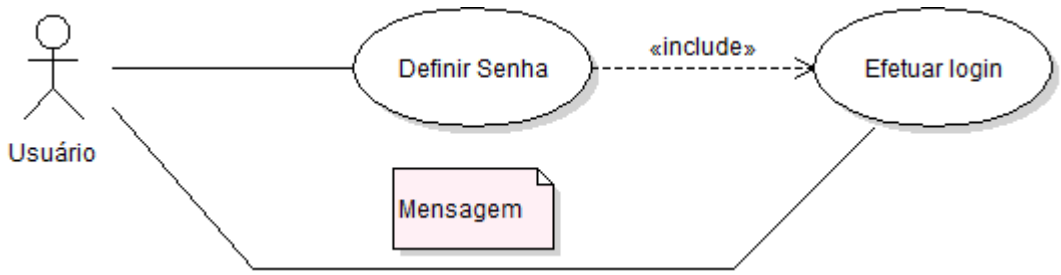

Figura 14 diagrama de caso uso 12 Definir Senha

Nome do Caso de Uso 12	Definir Senha.
Atores	Administrador / usuário.
Pré-condições	Efetuar controle de acesso.
Cenário Principal	1 - O sistema solicita os dados para efetuar a alteração. 2 - O usuário informa os dados. 3 - O usuário confirma os dados. 4 - O usuário seleciona a opção “Salvar”. 5 - O sistema altera os dados do usuário
Cenários Alternativos	Não existe.
Casos de Testes	1 - Se os dados informados estiverem corretos, executa a operação solicitada. 2 - Se os dados informados estiverem incorretos, cancela a operação e exibe uma mensagem de alerta.

Tabela 12 Definir Senha

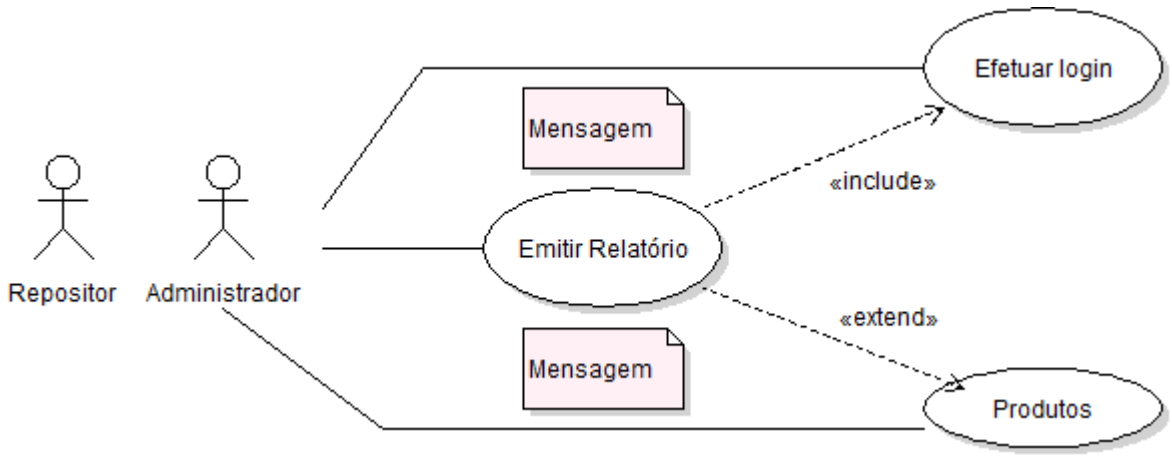

Figura 15 diagrama de caso de uso 13 Emitir Relatório de Produtos a Vencer

Nome do Caso de Uso 13	Emitir Relatório de Produtos a Vencer.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O sistema disponibiliza os dados necessários para gerar o relatório. 2 - O usuário seleciona a opção “Visualizar” relatório. 3 - O usuário seleciona a opção “Imprimir” relatório. 4 - O sistema imprime o relatório com sucesso.
Cenários Alternativos	O usuário pode visualizar o relatório e não salvar.
Casos de Testes	O usuário cancela a opção.

Tabela 13 Emitir Relatório de Produtos a Vencer

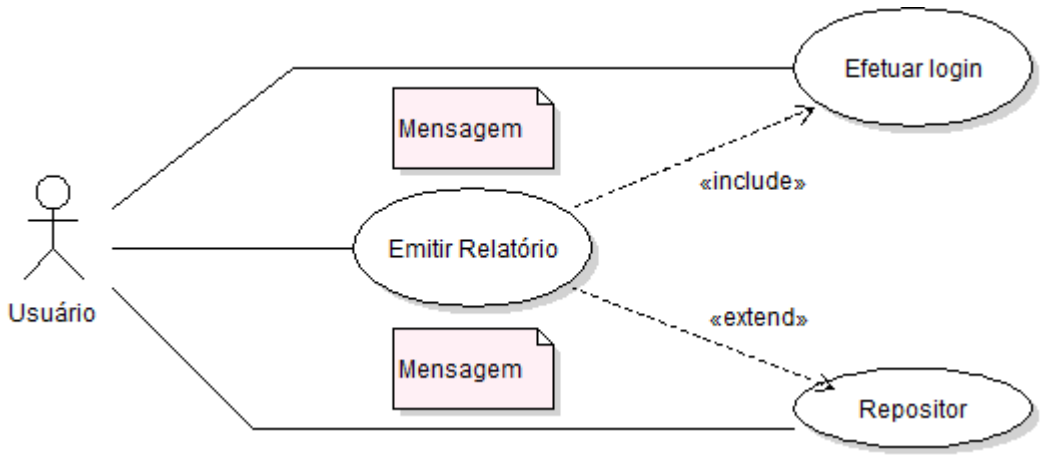

Figura 16 diagrama de caso de uso 14 Emitir Relatório de Repositores

Nome do Caso de Uso 14	Emitir Relatório de Repositores.
Atores	Administrador / usuário.
Pré-condições	Efetuar controle de acesso.
Cenário Principal	1 - O sistema disponibiliza os dados necessários para gerar o relatório. 2 - O administrador seleciona a opção “Visualizar” relatório. 3 - O administrador seleciona a opção “Imprimir” relatório. 4 - O sistema imprime o relatório com sucesso.
Cenários Alternativos	O administrador pode visualizar o relatório e não salvar.
Casos de Testes	O administrador cancela a opção.

Tabela 14 Emitir Relatório de Repositores

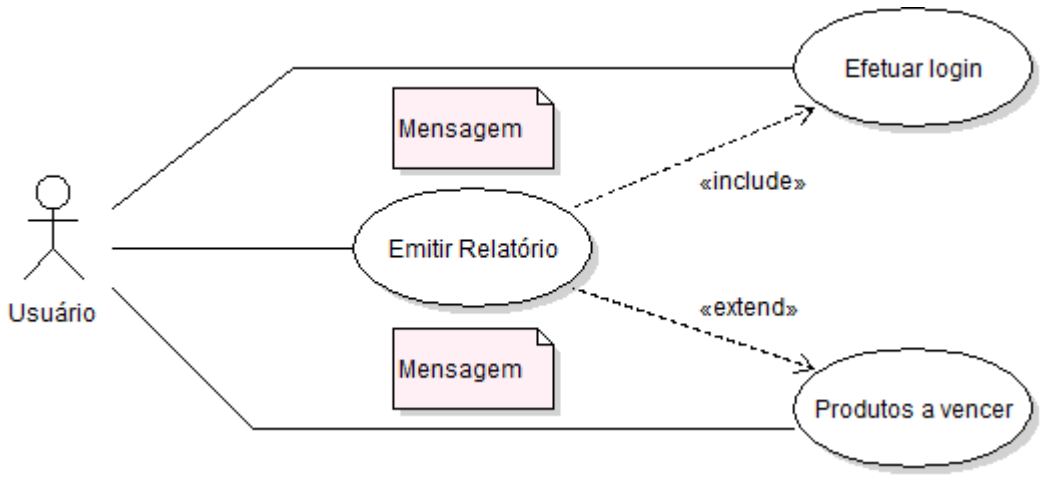

Figura 17 diagrama de caso de uso 15 Emitir Relatório de Produtos a Vencer por período

Nome do Caso de Uso 15	Emitir Relatório de Produtos a Vencer por período.
Atores	Administrador / usuário.
Pré-condições	Efetuar Login.
Cenário Principal	1 - O sistema disponibiliza os dados necessários para gerar o relatório. 2 - O usuário seleciona a opção “Visualizar” relatório. 3 - O usuário seleciona a opção “Imprimir” relatório. 4 - O sistema imprime o relatório com sucesso.
Cenários Alternativos	O usuário pode visualizar o relatório e não salvar.
Casos de Testes	O usuário cancela a opção.

Tabela 15 Emitir Relatório de Produtos a Vencer por período

2.5 DIAGRAMA DE ATIVIDADES

O diagrama de Atividades se preocupa em registrar os passos percorridos para a conclusão de uma atividade especifica, podendo esta ser representada por um método com certo grau de complexidade, um algoritmo, ou mesmo por um processo completo. Ele concentra-se na representação do fluxo de controle de uma atividade (GUEDES, 2011).

Abaixo, a Figura 18 ilustra o Diagrama de Atividades relativo a Consulta de Produtos a Vencer.

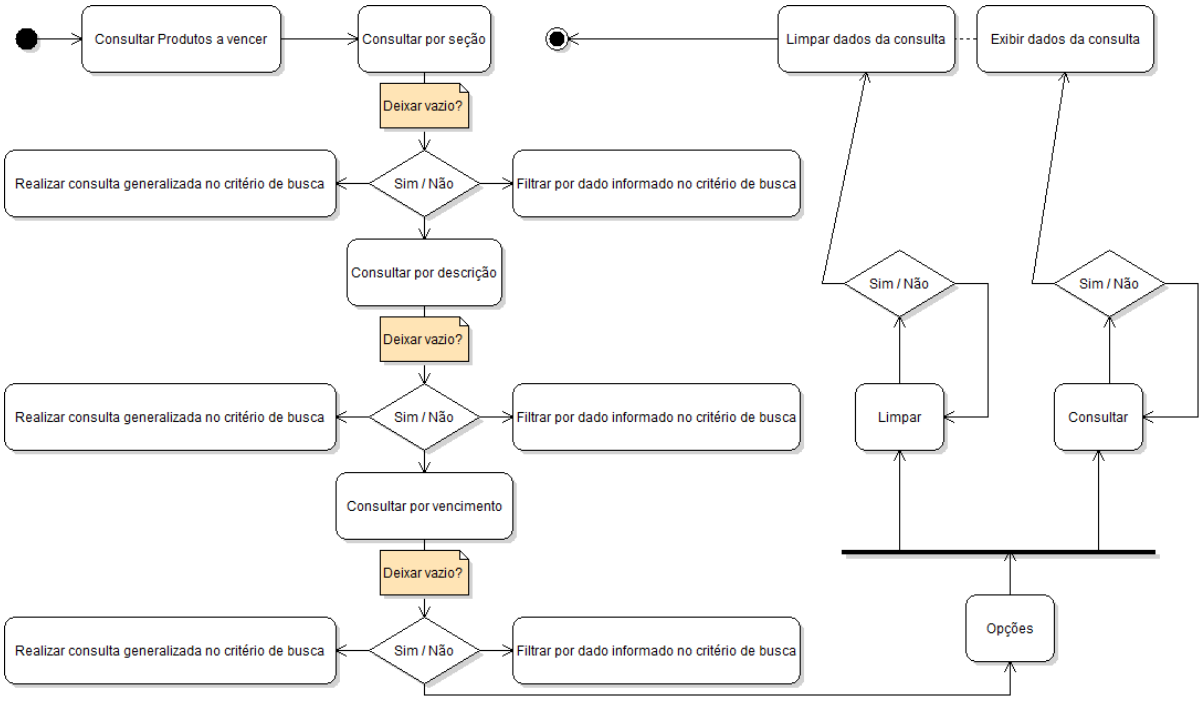

Figura 18 Diagrama de atividades

2.6 DIAGRAMA DE SEQUÊNCIA

Diagrama de sequência é um diagrama que se preocupa em registrar a ordem em que as mensagens são trocadas entre os processos envolvidos. Ele determina como o processo deve se desenrolar e ser concluído por meio de chamada de métodos disparados por mensagens enviadas entre os objetos (GUEDES, 2011).

Abaixo, as Figuras 19, 20, 21 Mostram os Diagramas de Sequência.

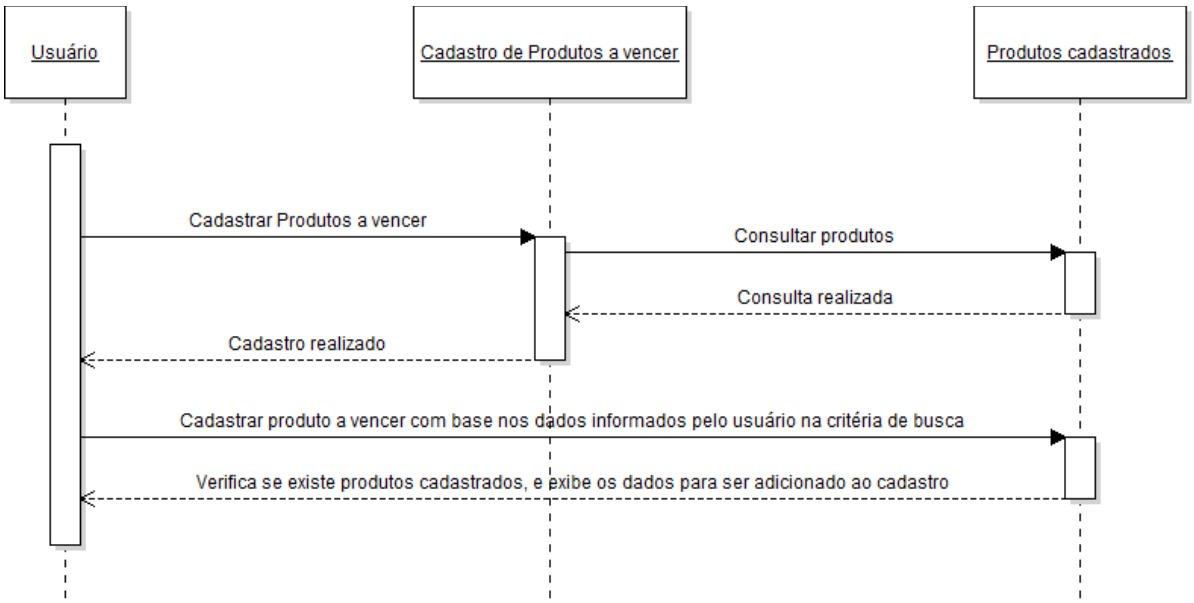

Figura 19 Diagrama de Sequência Cadastro de Produtos a Vencer

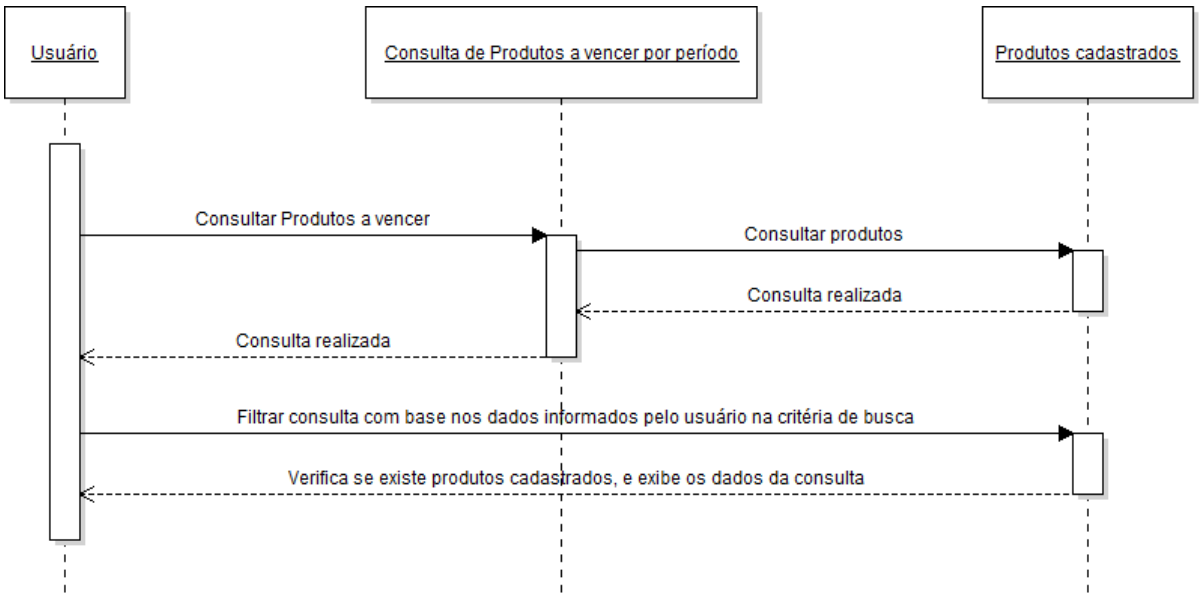

Figura 20 Diagrama de Sequência Consulta de Produtos a Vencer por período

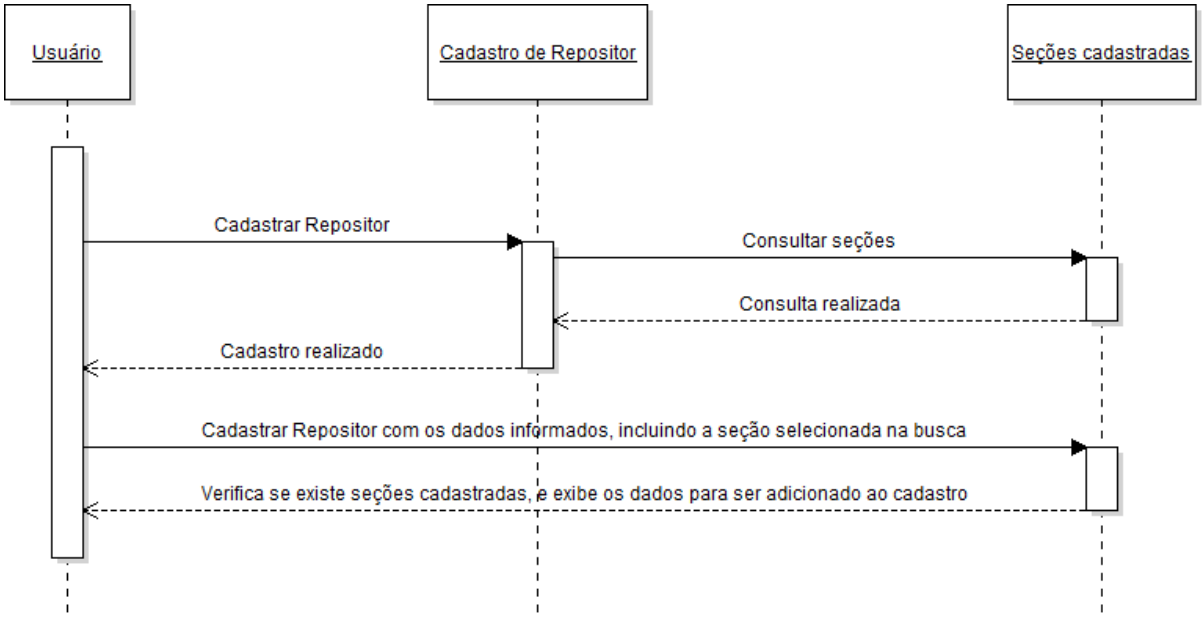

Figura 21 Diagrama de Sequência Cadastro de Repositor

2.7 DIAGRAMA DE CLASSES

Esse diagrama é um dos mais importantes e mais utilizados da UML. Sua principal característica e permitir a visualização das classes do sistema com seus respectivos atributos e métodos, bem como demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si (GUEDES, 2011).

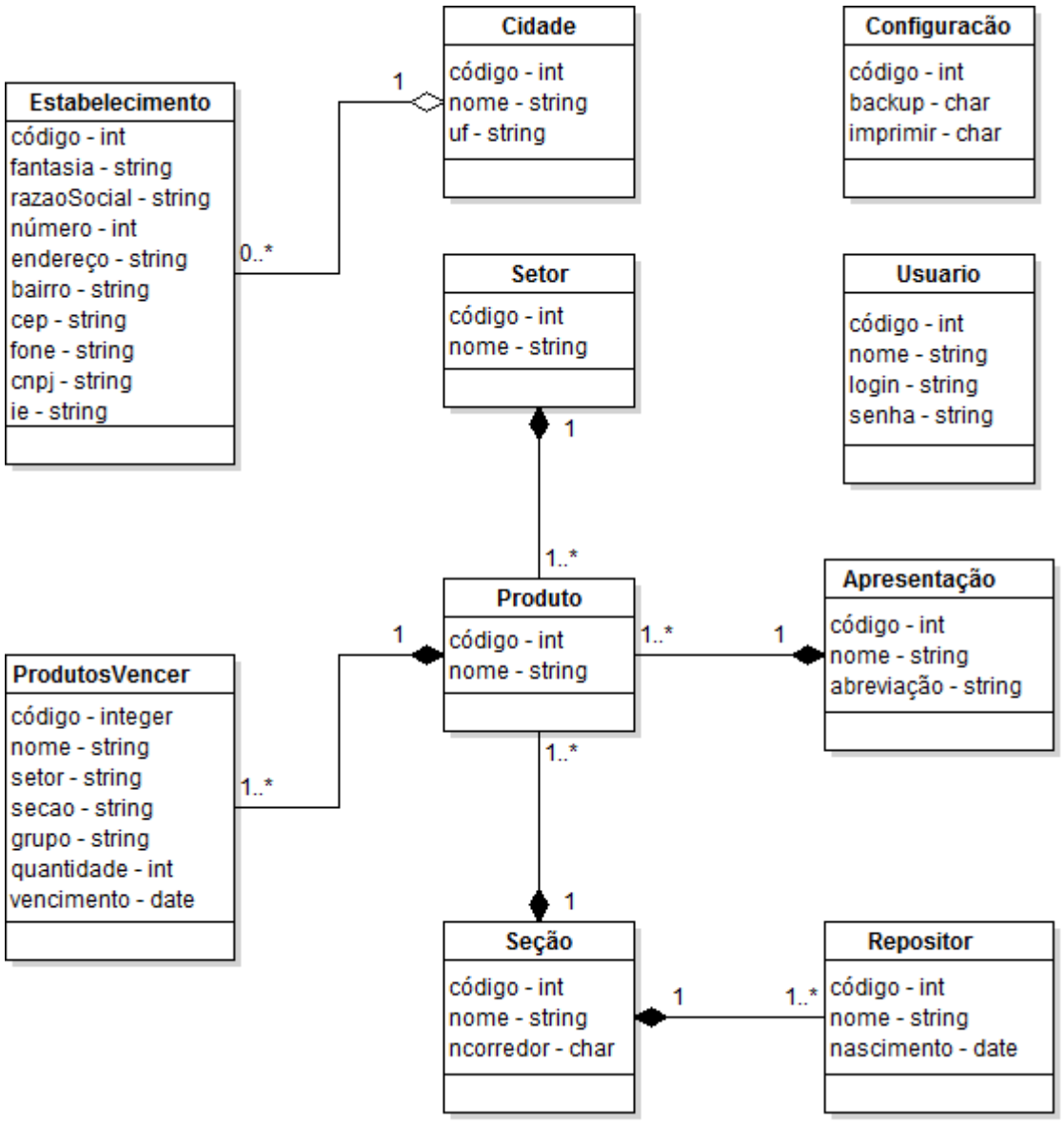

Figura 22 Diagrama de Classe - Beans

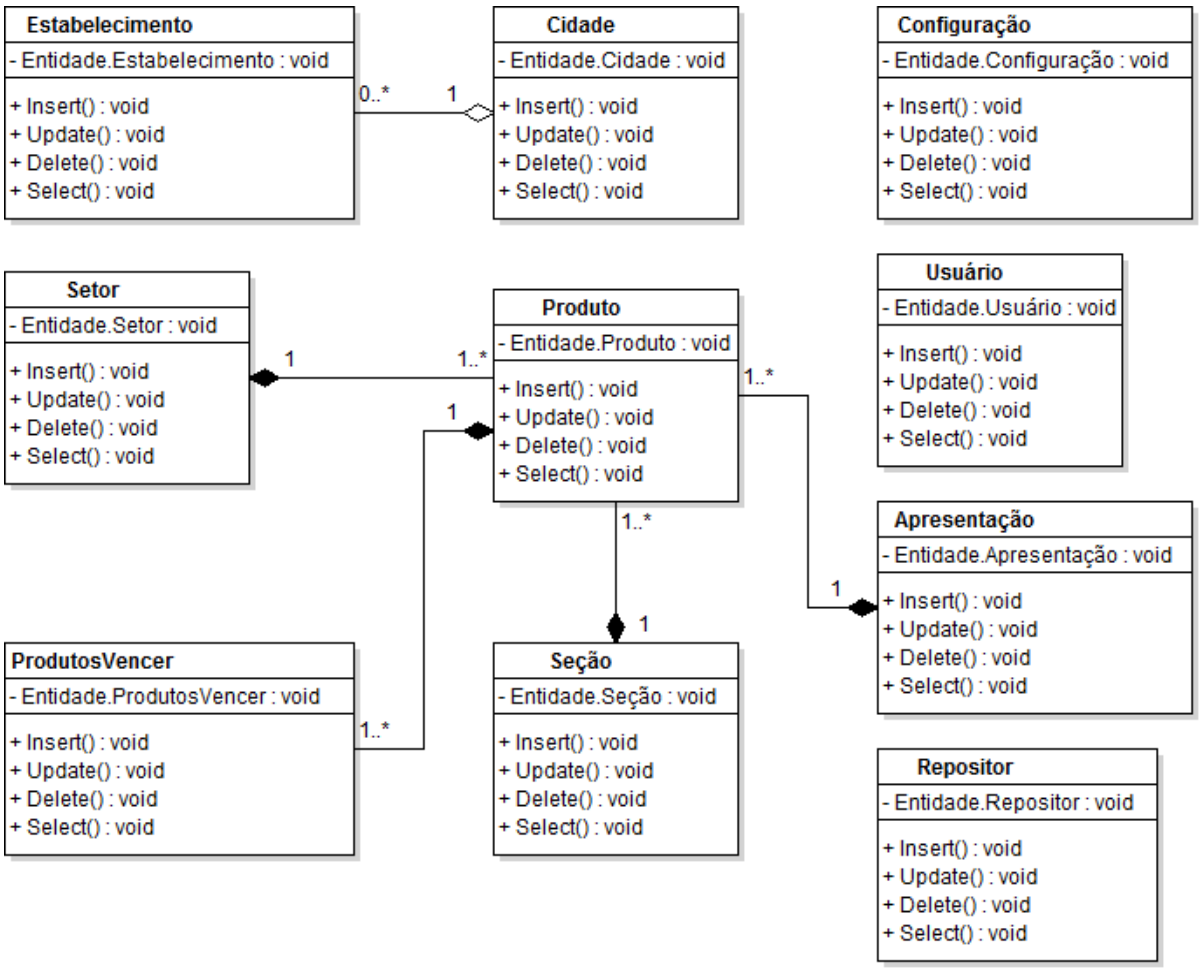

Figura 23 Diagrama de Classes - Dao

2.8 MODELAGEM DE ENTIDADE E RELACIONAMENTO

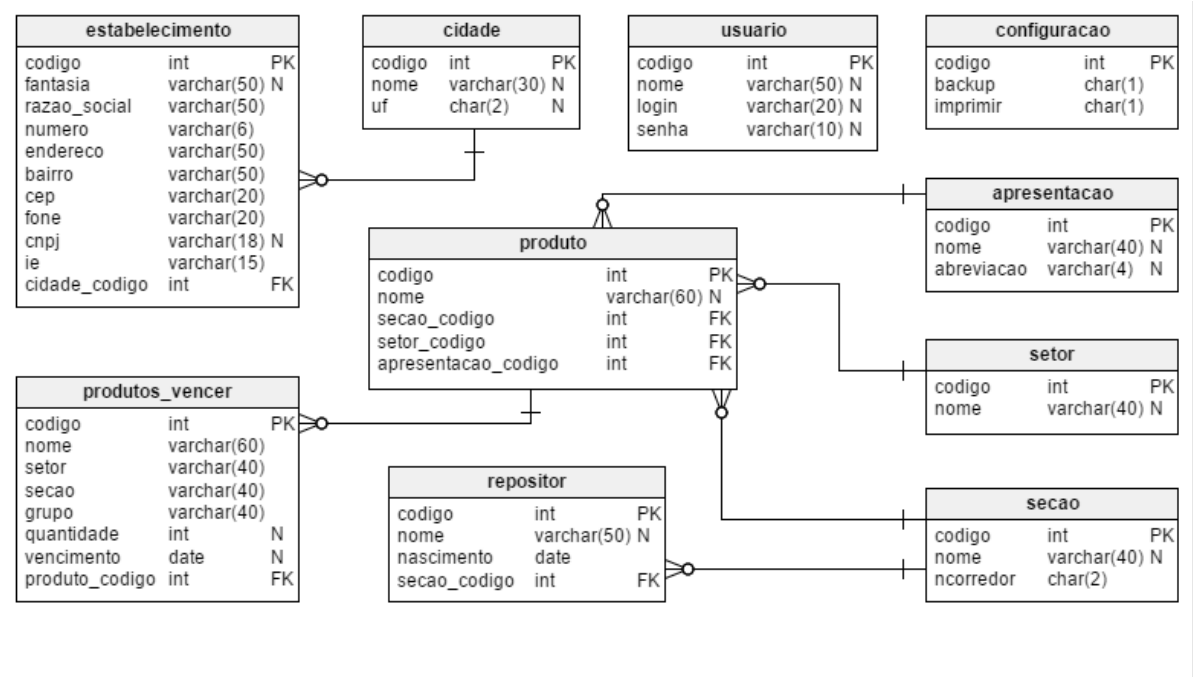

Figura 24 Modelo Entidade-Relacionamento

2.9 ESTRUTURA DO PROJETO

A EAP é um processo para subdividir os trabalhos em um projeto. Dessa forma, os trabalhos tornam-se componentes menores e mais simples de serem gerenciados. Ela é uma representação hierárquica das entregas de um projeto.

3 *WORK BREAKDOWN STRUCTURE* – WBS

Segue abaixo um diagrama que descreve as atividades que serão realizadas durante o desenvolvimento do sistema.

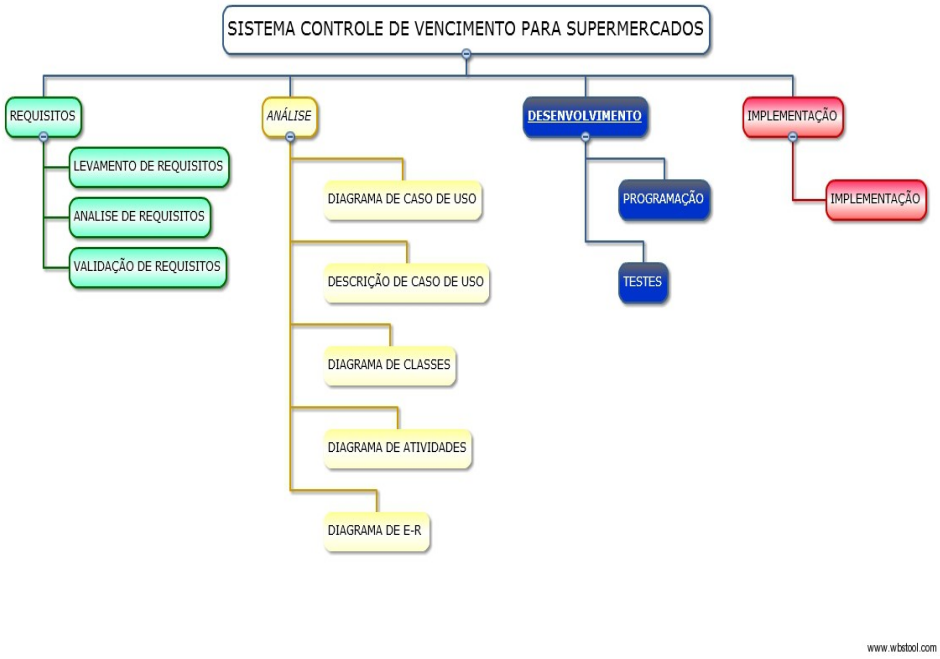

Figura 25 Atividades que serão desenvolvidas

3.1 ORÇAMENTO

Os recursos necessários para a análise e desenvolvimento de software Sistema controle de vencimento para supermercados são:

- 01 Notebook;
- 01 Impressora Multifuncional;

Equipamentos		
Equipamento	Valor	Modelo
Notebook	R$ 1.700,00	Acer
Impressora	R$ 340,00	HP
Total:	R$ 2.040,00	

Figura 26 Orçamento

3.2 IMPLEMENTAÇÃO DO APLICATIVO

Para a implementação do sistema foi utilizado o ambiente de desenvolvimento Eclipse Mars com a linguagem de programação Java. As classes foram divididas em cinco pacotes principais com sub-pacotes que será mostrado mais adiante, abaixo temos a imagem da projeto SCVS no Package Explorer do eclipse e também para melhor visualização foi configurado a apresentação deles em formato de hierarquia.

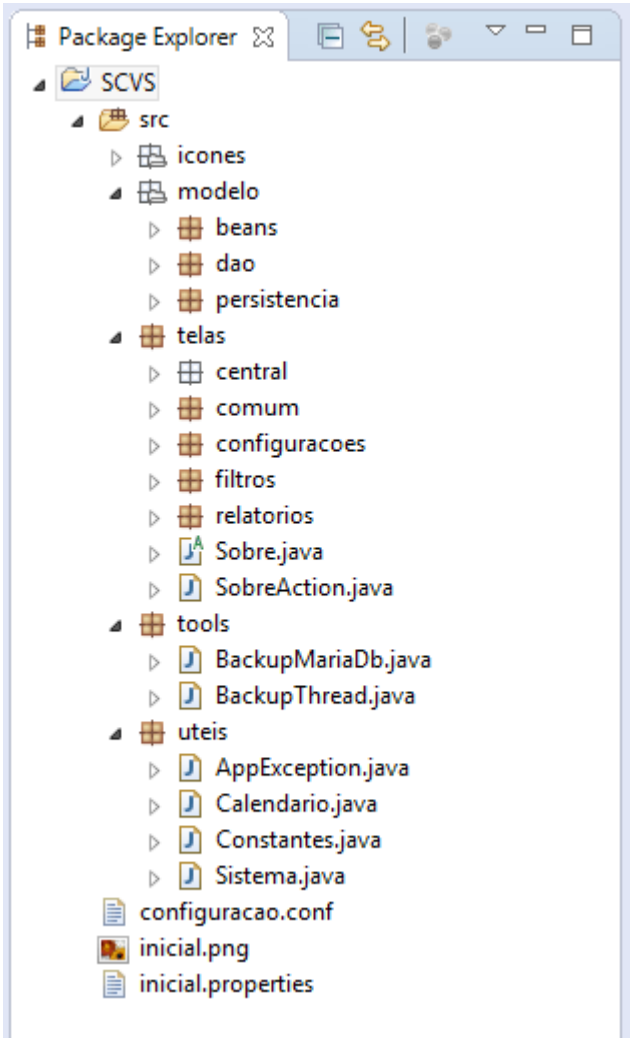

Figura 27 Package Explorer

Package icones – Pacote onde se encontra todos icones usados na interface do sistema

Package modelo – Pacote que faz a separação das camadas do projeto como o pacote Beans onde fica as classes responsáveis por encapsular e abstrair uma entidade, o pacote Dao que permite separar regras de negócio das regras de acesso a banco de dados e o pacote Persistência onde se encontra a classe de configuração do Hibernate.

Package telas – Pacote onde fica as janelas da interface do sistema em geral e também outros pacotes auxiliares importantes.

Package tools – Pacote que contém classes responsáveis para os parâmetros de backup do banco.

Package uteis – Pacote que contém classes responsáveis para o tratamento de exceções, modelagem do calenário, constantes e tratamento de erros verificados.

3.3 ORGANIZAÇÃO DO PROJETO SCVS

Um dos objetivos principais no desenvolvimento do sistema foi fazer com que sua implementação tivesse uma organização impecável no que diz respeito a estrutura do projeto, assim como a nomenclatura dos seus pacotes e classes, desta forma torna-lo o mais claro possível e consequentemente fazer com que o projeto tenha uma excelente visualização de sua estrutura de pacotes no que beneficia a alteração do sistema e também agiliza manutenções que podem ser realizadas por qualquer programador com total facilidade.

Os arquivos dentro da pasta do projeto 'SCVS' são arquivos de configuração do sistema como cor de fundo da tela principal, e a imagem inicial.

3.4 MAPEAMENTO XML – BEANS

Um mapeamento simples pode ser usado para simultaneamente mapear propriedades da classe e nós de um documento XML para um banco de dados ou, se não houver classe para mapear, pode ser usado simplesmente para mapear o XML.

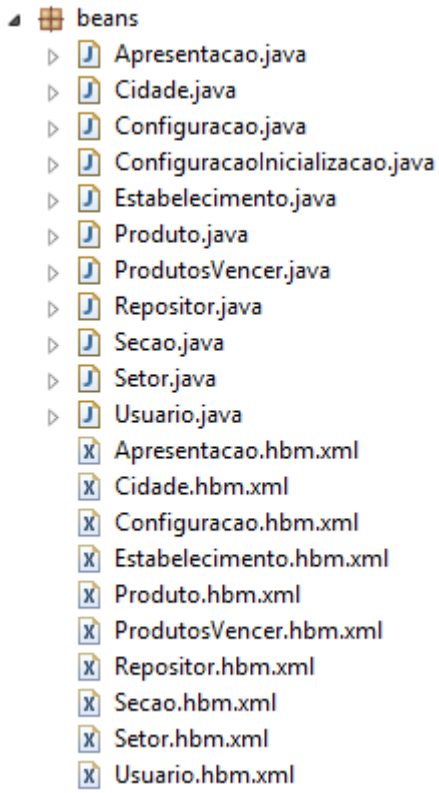

Figura 28 Package Beans

Pode se observar na imagem acima que para cada classe irá existir seu respectivo arquivo de mapeamento, no caso um arquivo do tipo HBM.XML[10]

10 http://www.tutorialspoint.com/hibernate/hibernate_mapping_files.htm - Acesso em 28, jul.2016

Apresentacao.java

```java
package modelo.beans;

public class Apresentacao {

    private Integer codigo;
    private String nome;
    private String abreviacao;

    public Integer getCodigo() {
        return codigo;
    }
    public void setCodigo(Integer codigo) {
        this.codigo = codigo;
```

Apresentacao.hbm.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hiber
<hibernate-mapping package="modelo.beans">
    <class name="Apresentacao" table="apresentacao">
        <id column="codigo" name="codigo">
            <generator class="increment" />
        </id>
        <property name="nome" type="string" />
        <property name="abreviacao" type="string" />
    </class>
</hibernate-mapping>
```

Figura 29 Mapeamento da classe modelo via XML

A imagem acima mostra claramente como é simples realizar o mapeamento da classe por meio de XML.

3.5 ORGANIZAÇÃO DO PACOTE DAO

Para acessar os dados usamos o padrão DAO(Data Access Object), assim encapsulamos todo o trabalho com o banco, e nossas classes que querer usar e manipular os dados simplesmente devem conhecer nossa classe DAO, sem se preocupar com abrir conexão, fechar e inserir comandos.

CidadeDao.java

```java
package modelo.dao;

import modelo.beans.Cidade;

public class CidadeDao extends Dao<Cidade> {

    public CidadeDao() {
        super(Cidade.class);
    }

    public String simularPromixoCodigo() throws Exception {
        return super.simularPromixoCodigo("'cidade'");
    }
}
```

Dao.java

```java
public void cadastrar(E object) throws Exception {
    HibernateUtil.getSession().save(object);
}

public void alterar(E object) throws Exception {
    HibernateUtil.getSession().update(object);
}

public void excluir(Integer id) throws Exception {
    excluir(carregarPorId(id));
}

public void excluir(E object) throws Exception {
    HibernateUtil.getSession().delete(object);
}

public E carregarPorId(String id) throws Exception {
    try {
        return carregarPorId(Integer.parseInt(id));
    } catch (NumberFormatException nfe) {
        return null;
    }
}

@SuppressWarnings("unchecked")
public E carregarPorId(Integer id) throws Exception {
    if (id == null)
        return null;
    Object object = HibernateUtil.getSession().get(clazz, id);
```

Figura 30 Package Dao

3.6 CLASSE HIBERNATE UTIL

Nesta classe foi feita a configuração do Hibernate através da programação e não pelo arquivo XML, isso fica a sua escolha. Na classe 'ConfiguracaoInicializacao' esta presente os atributos que contém por definição as devidas configurações (Dialeto, Driver, Usuário e Senha...) e retorna-mos o mesmo através do método 'getInstance()' na classe 'HibernateUtil' como pode ser observado na imagem abaixo, a partir dai esses valores são setados nas propriedades de configuração do Hibernate pelo método 'setProperty()'.

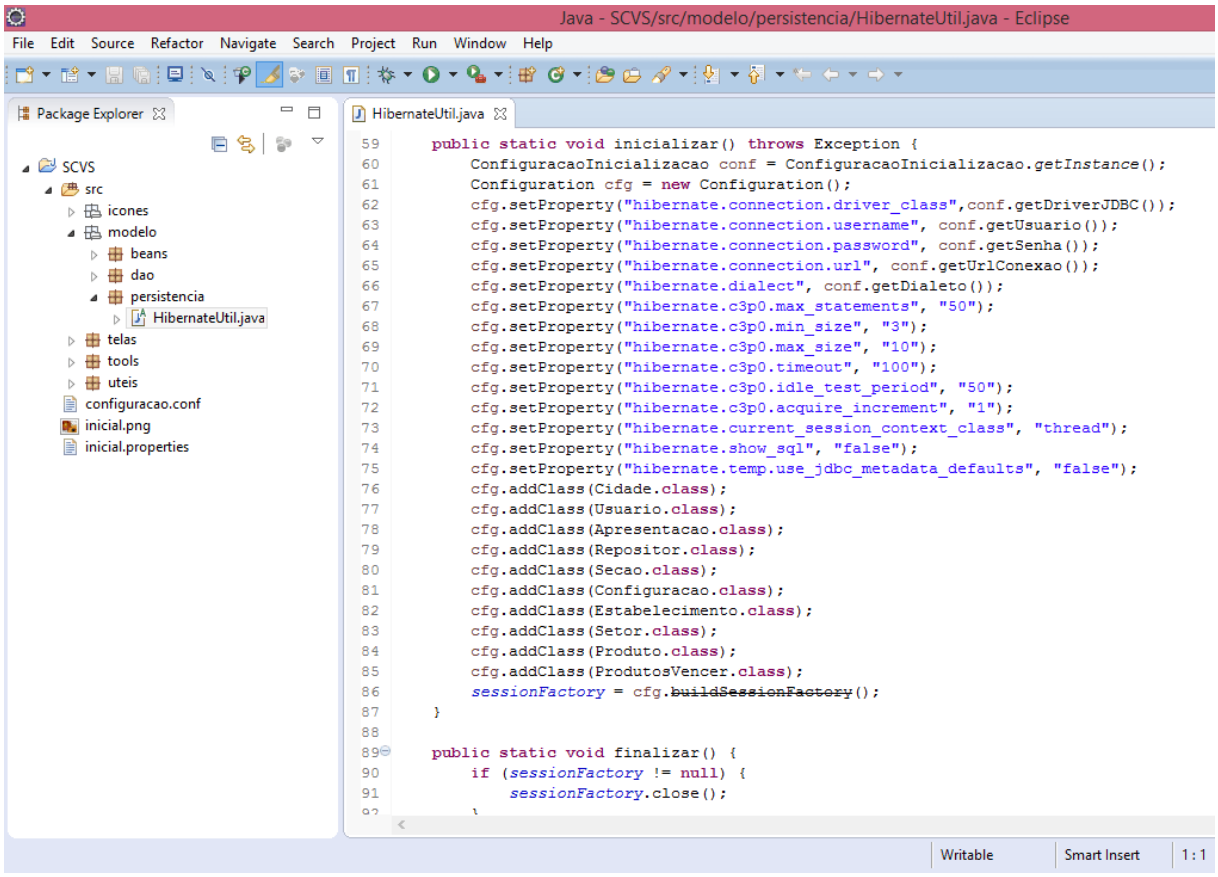

Figura 31 Hibernate Util

HibernateUtil.java

```
24    public static void begin() {
25        sessionFactory.getCurrentSession().beginTransaction();
26    }
27
28    public static void commit() {
29        Transaction transaction = sessionFactory.getCurrentSession()
30                .getTransaction();
31        if (transaction.isActive()) {
32            transaction.commit();
33        }
34    }
35
36    public static void rollback() {
37        Transaction transaction = sessionFactory.getCurrentSession()
38                .getTransaction();
39        if (transaction.isActive()) {
40            transaction.rollback();
41        }
42    }
43
44    public static void refresh(Object o) {
45        if (o == null)
46            return;
47        sessionFactory.getCurrentSession().refresh(o);
48    }
49
50    public static Session getSession() {
51        return sessionFactory.getCurrentSession();
52    }
53
54    public static Session fechar() {
55        return (Session) sessionFactory.getCurrentSession().close();
56    }
```

Figura 32 – Sessão Hibernate Util

Setor da classe 'HibernateUtil' que contém os métodos responsáveis pela abertura da sessão(begin), conclusão da sessão(commit), estorno da sessão(rollback) e fechamento.

3.7 ORGANIZAÇÃO DO PACOTE TELAS

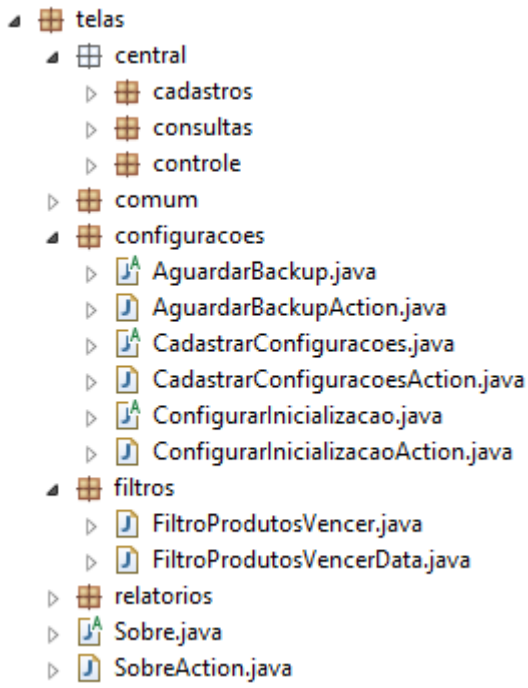

Figura 33 – Package telas

Package central – Pacote que contem classes de telas voltadas para o 'Crud' do sistema como interfaces de cadastros, consultas e controle.

Package comum – Pacote que contem classes de telas responsáveis pelo carregamento do sistema, login e tela inicial.

Package configurações – Pacote responsável por conter telas de ajustes simples do sistema, como backup automático e inicialização do sistema.

Package filtros – Pacote que contém classes com atributos para filtragem de dados dos relatórios.

Package relatórios – Pacote que contém classes de telas dos relatórios.

3.8 ORGANIZAÇÃO DO PACOTE RELATÓRIOS

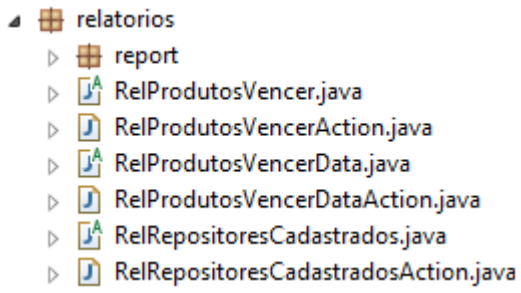

Figura 34 – Package relatórios

Package report – Pacote que contém os arquivos JRXML; Este XML possui todas as informações de formatação do relatório, e além disso, possui os campos que serão preenchidos posteriormente, de acordo com a fonte de dados utilizada (*data source).*

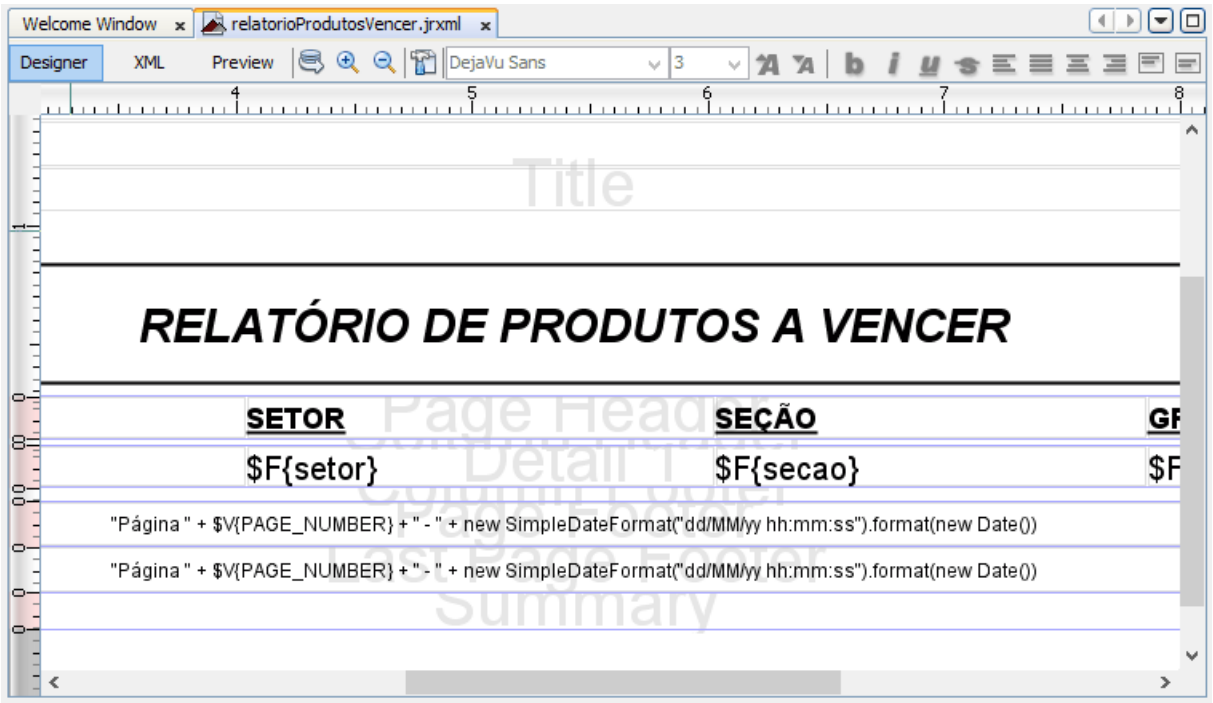

Figura 35 – iReport

ProdutosVencerDao.java

```java
    @SuppressWarnings("unchecked")
    public List<ProdutosVencer> pesquisaProdutosVencer(FiltroProdutosVencer fpv) {
        Criteria criteria = HibernateUtil.getSession().createCriteria(ProdutosVencer.class);
        if (fpv.getNome() != null) {
            criteria.add(Expression.eq("nome", fpv.getNome()));
        }
        if (fpv.getSetor() != null) {
            criteria.add(Expression.eq("setor", fpv.getSetor()));
        }
        if (fpv.getSecao() != null) {
            criteria.add(Expression.eq("secao", fpv.getSecao()));
        }
        if (fpv.getGrupo() != null) {
            criteria.add(Expression.eq("grupo", fpv.getGrupo()));
        }
        if (fpv.getVencimento() != null) {
            criteria.add(Expression.eq("vencimento", fpv.getVencimento()));
        }
        criteria.addOrder(Order.asc("nome"));
        return criteria.list();
    }

    @SuppressWarnings("unchecked")
    public List<ProdutosVencer> consultaProdutosVencer(FiltroProdutosVencer fpv) {
        Criteria criteria = HibernateUtil.getSession().createCriteria(ProdutosVencer.class);
        if (fpv.getSetor() != null) {
            criteria.add(Expression.eq("setor", fpv.getSetor()));
        }
        if (fpv.getSecao() != null) {
            criteria.add(Expression.eq("secao", fpv.getSecao()));
        }
        if (fpv.getGrupo() != null) {
            criteria.add(Expression.eq("grupo", fpv.getGrupo()));
```

Figura 36 – Criteria Hibernate

A imagem acima mostra um trecho do código da classe 'ProdutosVencerDao' no qual é responsável pela criteria que realiza o filtro dos dados do objeto que são passados por parâmetro pelos métodos de consulta da classe.

A API Criteria nos permite construir consultas estruturadas utilizando Java, e ainda provê uma checagem de sintaxe em tempo de compilação que não é possível com uma linguagem de consulta como HQL ou SQL.

3.9 REGRAS DE NEGÓCIO

A regra de negócio do projeto foi estruturada pela separação da tela de ações e da tela de desing por meio de herança, ou seja a tela na qual terá apenas código de desing será a classe pai, consequentemente a classe filha será responsável pelos métodos definidos na regra de negócio e não terá código de tela, ou seja ela terá apenas métodos voltados para as regras de negócio do sistema, desta forma ambas classes terá visivelmente seus códigos extremamente limpos.

A imagem abaixo mostra com mais detalhe a separação da regra de negocio da interface do usuário.

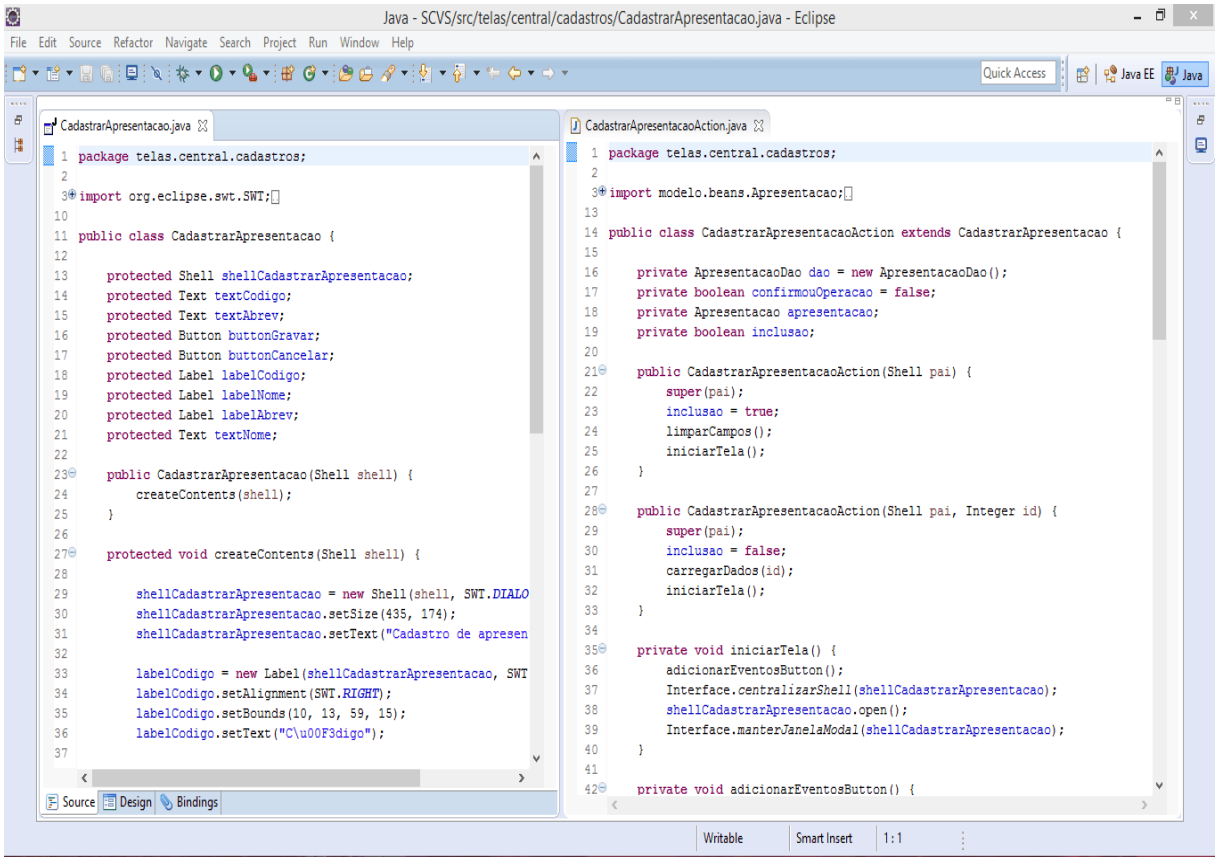

Figura 37 – Regras de negócio

4 CLASSE DE CONEXÃO COM O BANCO

ConfiguracaoInicializacao.java

```
14 public class ConfiguracaoInicializacao implements Serializable {
15
16     private static final long serialVersionUID = 1L;
17     private String usuario = "root";
18     private String senha = "123";
19     private String dialeto = "org.hibernate.dialect.MySQLDialect";
20     private String driverJDBC = "org.mariadb.jdbc.Driver";
21     private String servidor = "localhost", database = "banco", porta = "3306";
22     private String urlConexao = "jdbc:mariadb://" + servidor + ":" + porta + "/" + database;
23     private String gmt = TimeZone.getDefault().getID();
```

ConfigurarInicializacaoAction.java

```
24     private void carregarConfiguracao() {
25         ConfiguracaoInicializacao config = ConfiguracaoInicializacao.getInstance();
26         textUrlConexao.setText(config.getUrlConexao());
27         textDriver.setText(config.getDriverJDBC());
28         textUsuario.setText(config.getUsuario());
29         textSenha.setText(config.getSenha());
30         textGMT.setText(config.getGmt());
31     }
32
33     private void salvarArquivoConfiguracao() throws Exception {
34         ConfiguracaoInicializacao config = ConfiguracaoInicializacao.getInstance();
35         config.setUrlConexao(textUrlConexao.getText());
36         config.setDriverJDBC(textDriver.getText());
37         config.setUsuario(textUsuario.getText());
38         config.setSenha(textSenha.getText());
39         config.setGmt(textGMT.getText());
40         Class.forName(config.getDriverJDBC());
41         DriverManager.getConnection(config.getUrlConexao(), config.getUsuario(), config.getSenha());
42         ConfiguracaoInicializacao.atualizar();
```

Figura 38 – Classe de conexão

4.1 CLASSE DE BACKUP DO BANCO

A imagem abaixo mostra a classe responsável pela parametrização de uma execução de dump do banco de dados desta forma realizando o backup do mesmo.

BackupMariaDb.java

```
1  package tools;
2
3  import java.io.IOException;
6
7  public class BackupMariaDb {
8
9      public void realizarBackup(String caminhoSalvar) throws Exception {
10         ConfiguracaoInicializacao conf = ConfiguracaoInicializacao.getInstance();
11         String comando = "mysqldump --host=" + conf.getServidor() + " --user="
12                 + conf.getUsuario() + " --password=" + conf.getSenha() + " "
13                 + conf.getDatabase() + " > " + caminhoSalvar;
14         try {
15             Runtime.getRuntime().exec(comando);
16         } catch (IOException e) {
17             throw new Exception("Não foi possível realizar o backup.");
18         }
19     }
20 }
```

Figura 39 – Classe de backup do banco

4.2 INTERFACES DO SISTEMA

O Eclipse tornou-se famoso não apenas por ser uma IDE de primeira qualidade, desenvolvido com a extensibilidade em mente, mas também porque ele tinha uma interface gráfica levíssima e que era igual a do sistema operacional onde ele estivesse executando, diferentemente do conjunto de componentes gráficos do Java, o Swing, que tinha uma “cara” própria e algumas opções que fazem com que ele “imite” a aparência de um sistema operacional qualquer. Mas essa possibilidade de imitar ainda não era a melhor escolha, porque vários sistemas operacionais tem a opção para aplicar-se temas gráficos e as aplicações Swing não tinham como acompanhar estas mudanças.

Esta interface gráfica do Eclipse foi “separada” do código principal da IDE e tornou-se o que se conhece hoje como SWT, o Standard Widget Toolkit. Ele é uma camada de componentes sobre os componentes padrão do sistema operacional, um botão SWT é na verdade um botão do seu sistema operacional, então, se você aplicar um tema que mude a cor ou a forma do seu botão, o botão SWT vai se modificar conforme o novo tema.

A interface do sistema desenvolvido provem de um padrão de design impecável no que diz respeito o tamanho das janelas do aplicativo e a organização de seus componentes, tornando-se um sistema leve, versátil e tecnicamente com um visual simples, elegante, e o mais importante de tudo de fácil manipulação pelo ao usuário.

4.3 BANCO DE DADOS

A imagem abaixo mostra o ambiente de desenvolvimento SQL do projeto no caso utilizando o banco dados MariaDB_10.0.20[11] que mantém o HeidiSQL como o seu SGDB(Sistema Gerenciador de Banco de Dados) que é instalado automaticamente junto com o banco.

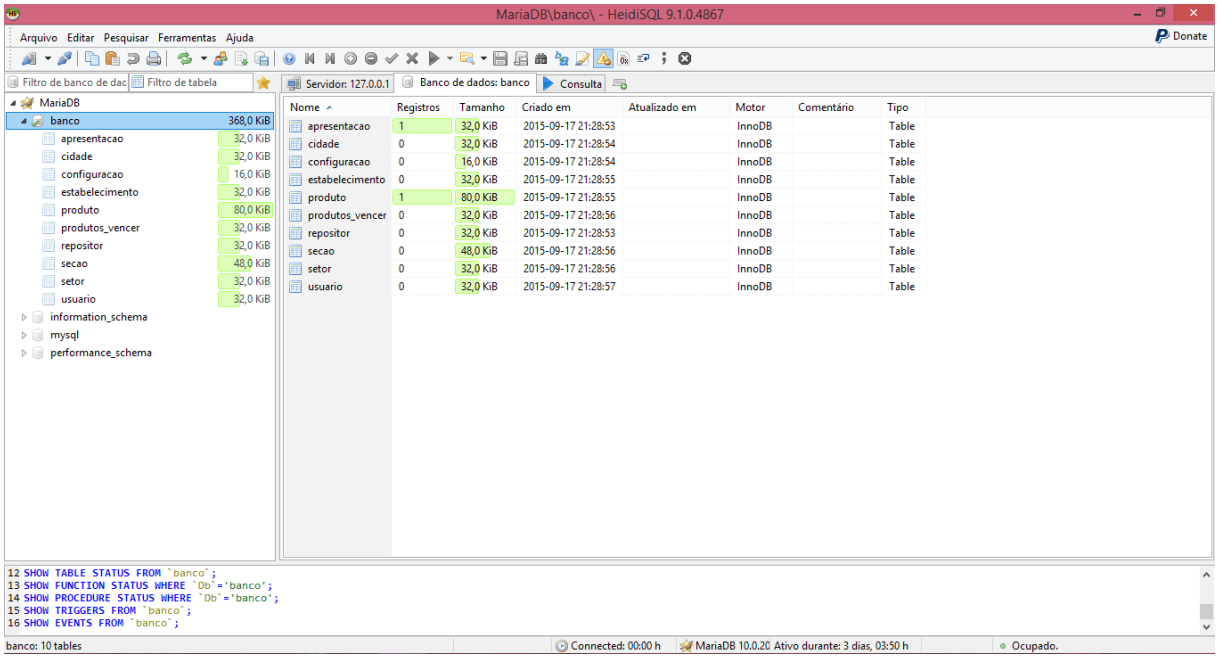

Figura 40 – Banco de dados

11 https://downloads.mariadb.org/mariadb/10.0.20/ - Acesso em 29, jul.2016

4.4 SWT DESIGNER

O SWT é uma incrível biblioteca de componentes visuais para aplicações desktop que garantem ter a mesma aparência dos outros componentes visuais do sistema operacional, diferentemente do Swing que mesmo parecendo ser igual, não consegue ter o mesmo pressentimento dos componentes padrão do sistema.

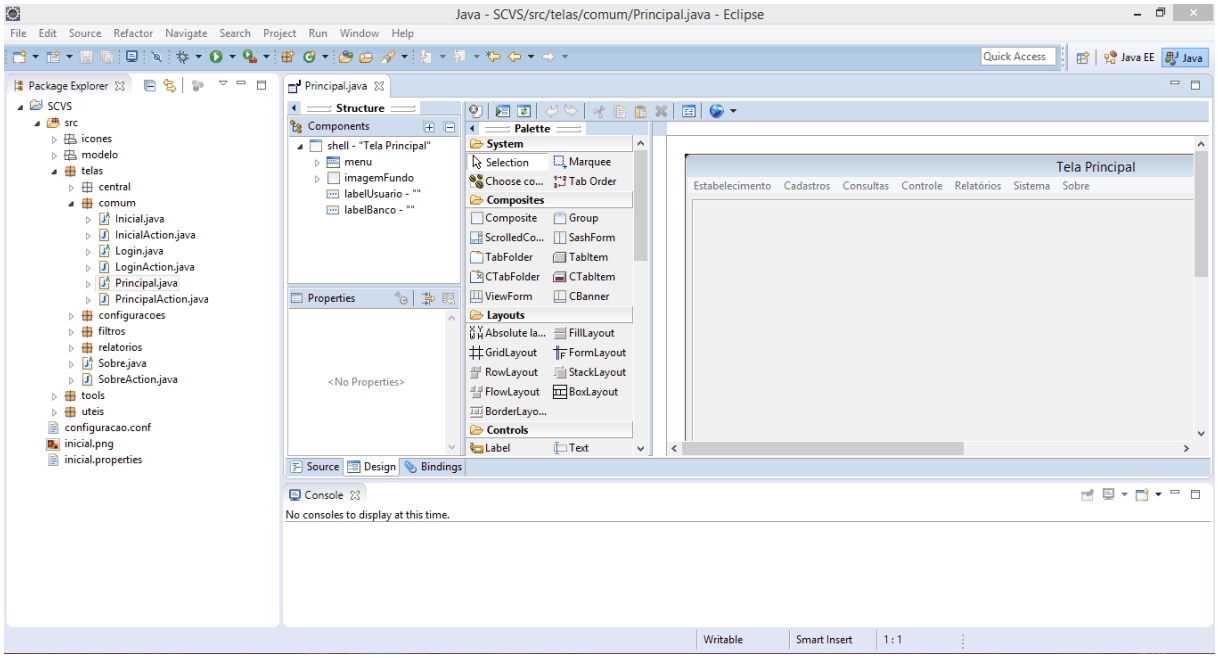

Figura 41 – SWT Designer

4.5 LINHA DE ARGUMENTO

Antes da inicialização do sistema no servidor, ou seja a maquina na qual será instalada a aplicação devemos configurar o caminho do banco, driver, senha etc.. desta forma para abrir a tela de configuração do banco devemos ir antes na opção 'Run Configurations' do eclipse e em seguida inserir 'config' na linha de argumentos da aba 'Arguments', no eclipse, logo após isso devemos aplicar e rodar o sistema, e automaticamente irá abrir a tela de configuração do banco para o administrador inserir os parâmetros necessários para comunicação com o banco de dados. Se os parâmetros inseridos estiverem corretos a tela dará prosseguimento e o administrador poderá assim dar continuidade a inicialização normal do sistema, para que isso ocorra basta fazer o processo da linha de argumentos no eclipse novamente e remover a linha 'config', aplicar, e o sistema abrirá normalmente ao ser executado.

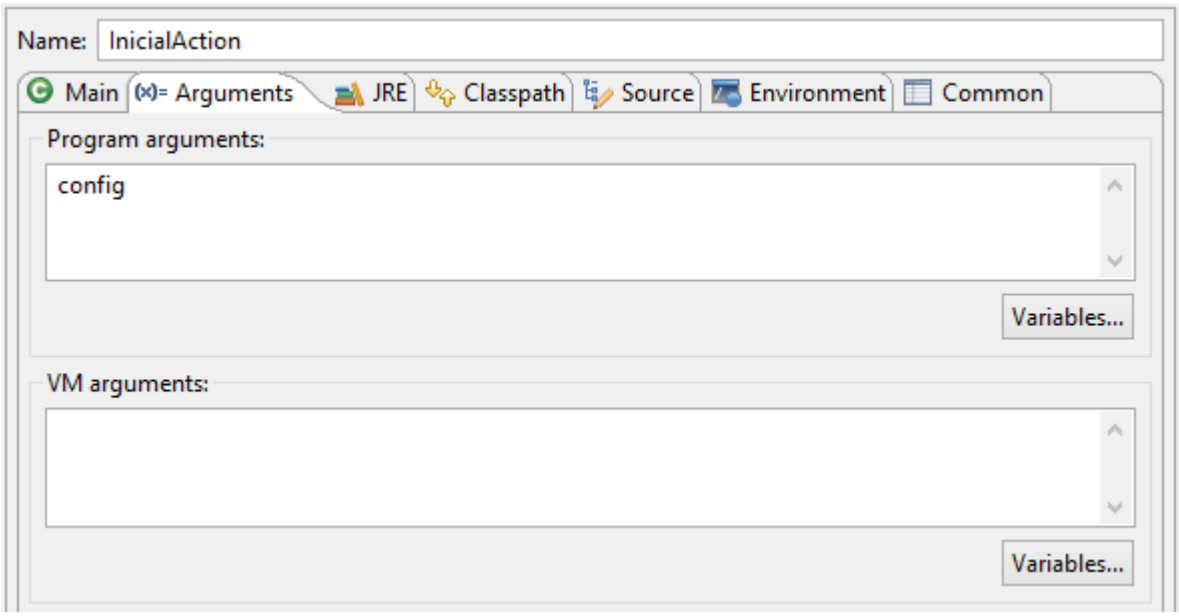

Figura 42 – Program arguments

4.6 CONFIGURAÇÃO DE INICIALIZAÇÃO

Tela responsável pela configuração de inicialização do sistema, citada no Subcapítulo anterior, essa tela contém basicamente os campos que serão preenchidos pelos parâmetros necessários para comunicação com o banco.

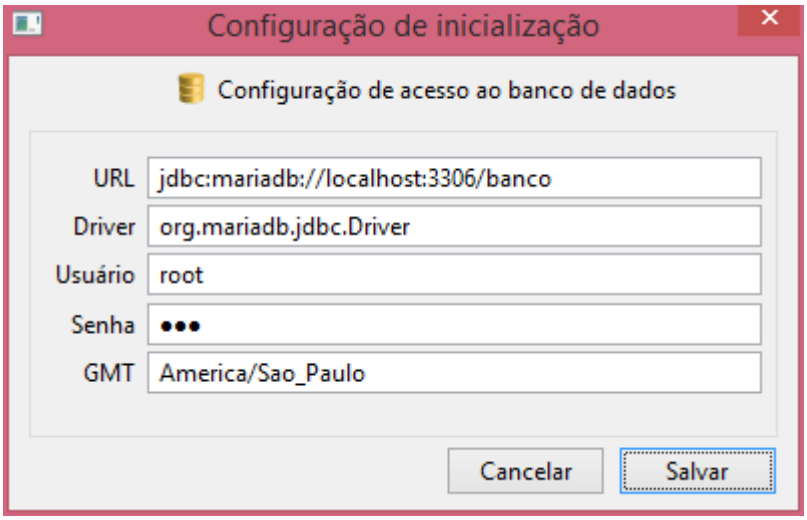

Figura 43 – Configuração de inicialização

4.7 TELAS DO SISTEMA

– Tela de Login

Ao acessar o sistema, está é a primeira tela a se abrir, local onde se faz o login dos usuários do sistema, informando um usuário e login válido, o sistema será habilitado.

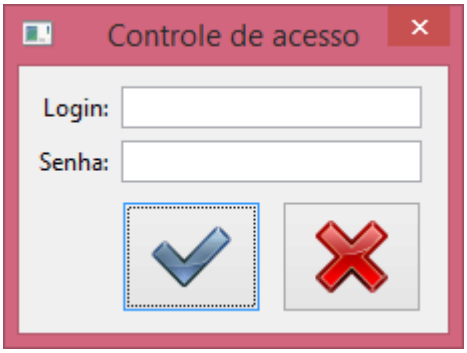

Figura 44 – Tela de login

O usuário administrador sendo o usuário master do sistema tem um registro padrão no banco desta forma qualquer usuário poderá logar no sistema utilizando o login '1' e senha '1'. Após a abertura do sistema o administrador poderá cadastrar novos usuários e também fazer alteração de seus próprios dados de administrador.

– Tela Principal

Tela principal do sistema simplificada, a tela apresenta menus limpos, com perfeita organização e de fácil acesso, nela a alguns detalhes como mostrar em sua barra o nome do usuário logado no sistema e também é possível visualizar qual o banco a aplicação esta conectada.

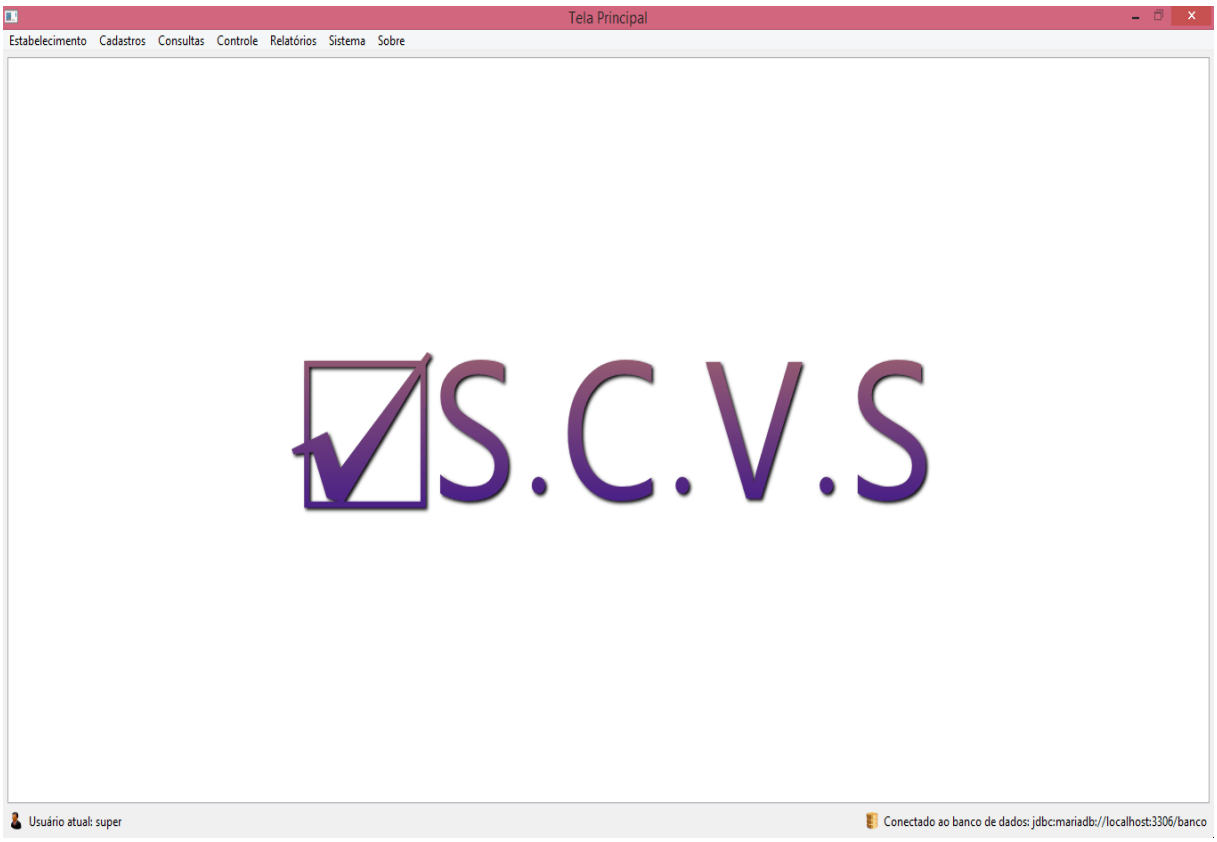

Figura 45 – Tela principal

– Tela Estabelecimento

Tela de cadastro e consulta do estabelecimento, basicamente é uma tela unificada, tornando-se uma tela de edição, ela provem de uma 'ComboBox' na qual será carregada pelas cidades cadastradas no sistema.

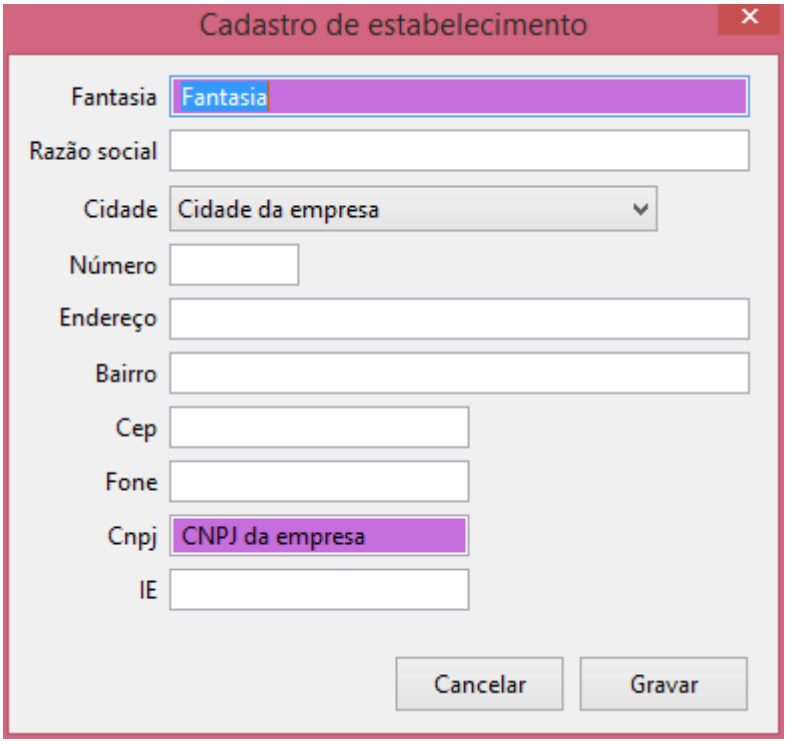

Figura 46 – Tela estabelecimento

Todas telas do sistema terá seus campos obrigatórios sinalizados pela cor roxa.

– Telas de cadastro

Telas simplificadas de cadastro, com a diferença das telas de cadastro dos repositores e a de produtos que possuem algumas 'ComboBox' que facilitam a busca de informações já cadastradas no sistema, e que se relacionam com os cadastros em questão.

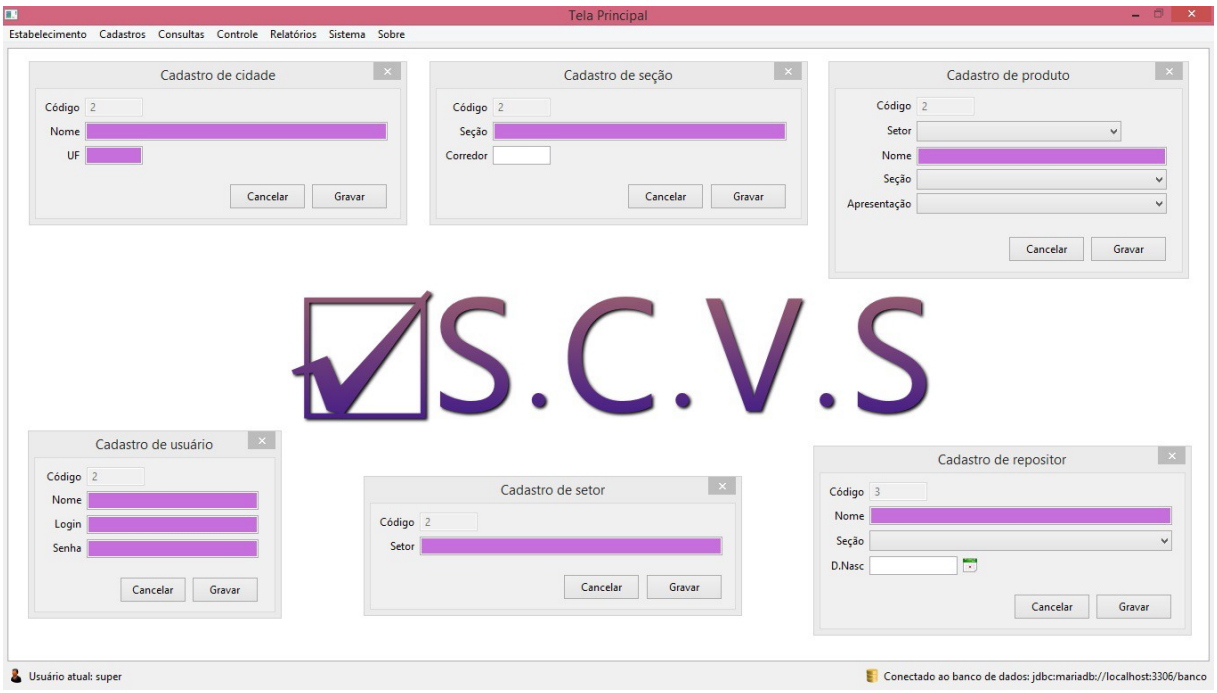

Figura 47 – Telas cadastros

– Telas de consulta

Telas simplificadas de consulta, todas as telas de consultas são praticamente iguais, apenas mudando os campos disponíveis na 'ComboBox', para o filtro da busca e as informações da 'Table' que seriam as colunas na tabela.

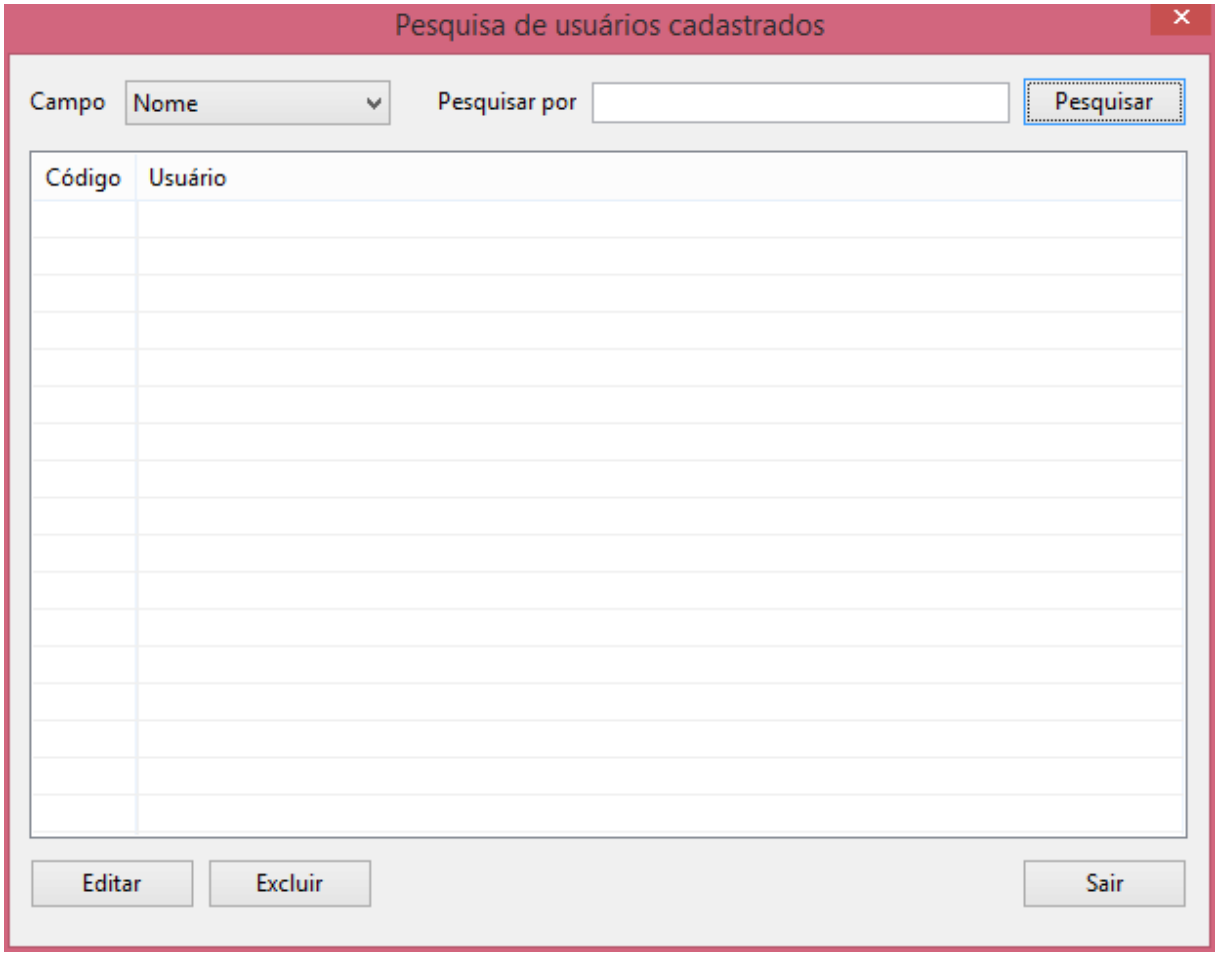

Figura 48 – Telas consultas

Quando o usuário não informar nenhuma opção de busca, ao executar a pesquisa a tela trará todos os dados cadastrados. Esse comportamento ocorre em todas telas de busca e pesquisa.

– Tela cadastrar produtos a vencer

Tela que tem a funcionalidade de adicionar um produto já cadastrado no sistema na lista de vencimento. A lupa na tela leva o usuário a tela de pesquisa de produtos cadastrados, desta forma facilitando a busca do produto desejado para inserir no cadastro de vencimento, e também oferece a opção de busca pelo código do produto informando o código diretamente no primeiro campo com fundo branco na tela.

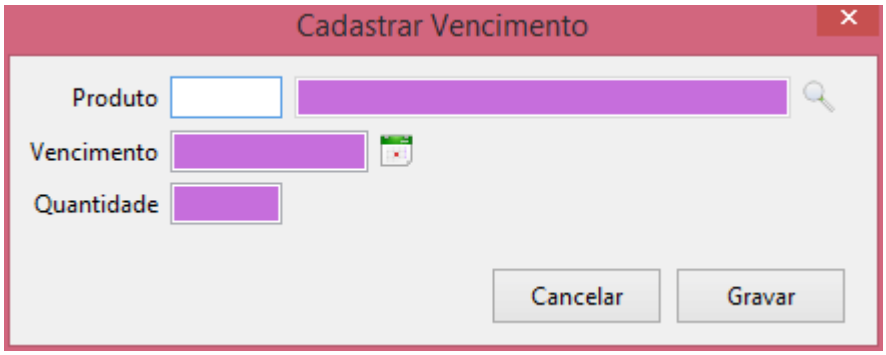

Figura 49 – Tela cadastrar produtos a vencer

– Tela consultar produtos a vencer

Tela que tem a funcionalidade de realizar a pesquisa dos produtos que estão na lista de vencimento, utilizando filtros de busca pelo campo desejado que são oferecidos na 'ComboBox' da tela como; Nome, código, seção, setor e grupo. Além de filtrar a pesquisa pela data de validade do produto cadastrado.

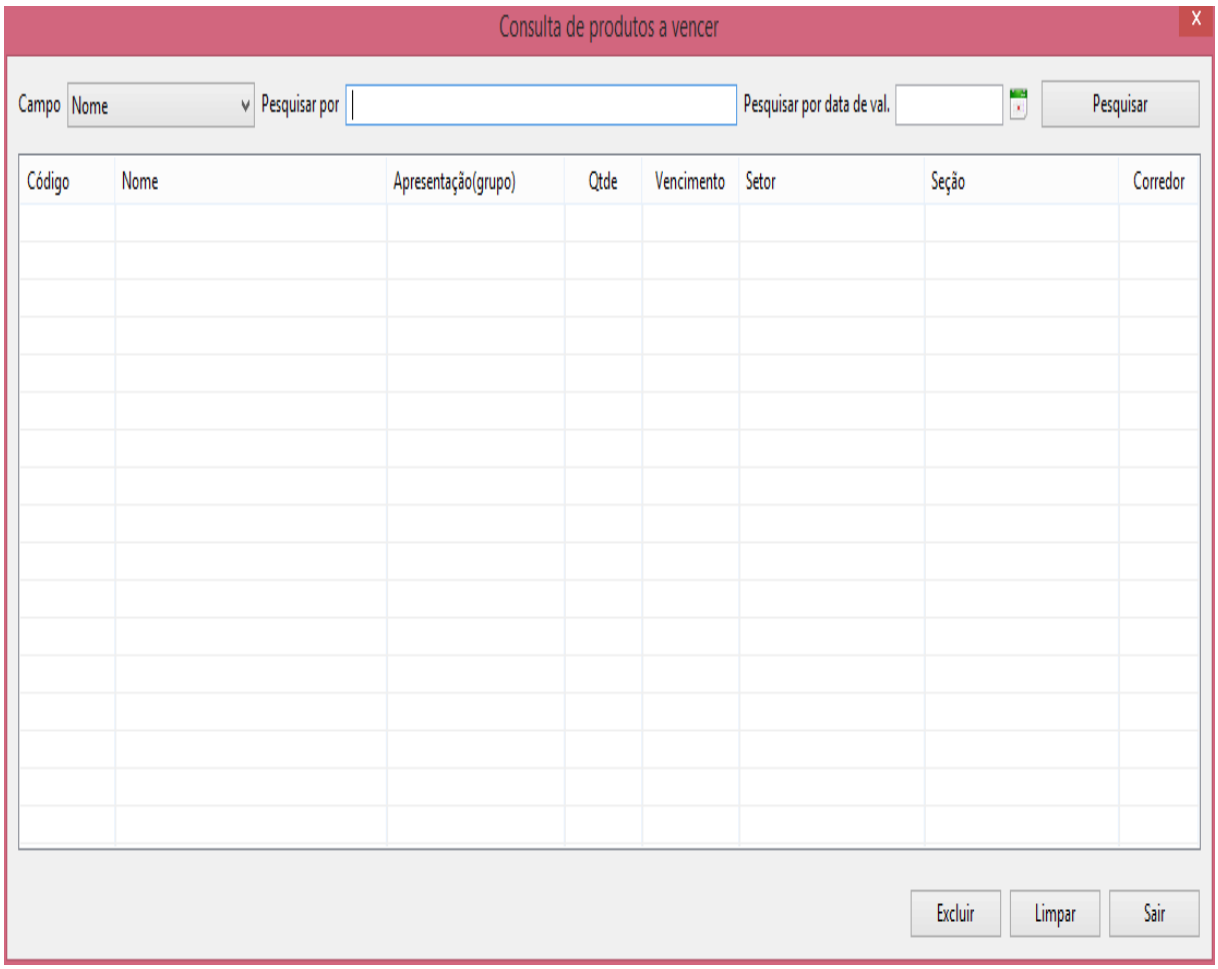

Figura 50 – Tela consultar produtos a vencer

– Tela relatórios

Telas que tem a funcionalidade de emitir relatórios a partir de filtros buscas como intervalo de datas, dando como exemplo o relatório de 'Produtos a vencer por data' e outros filtros comums, e no que tange internamente o sistema também foi usado a API criteria do Hibernate para tornar-se mais pratica a elaboração destes filtros e tratamento de filtragem de datas, dando também como exemplo o relatório de repositores cadastrados.

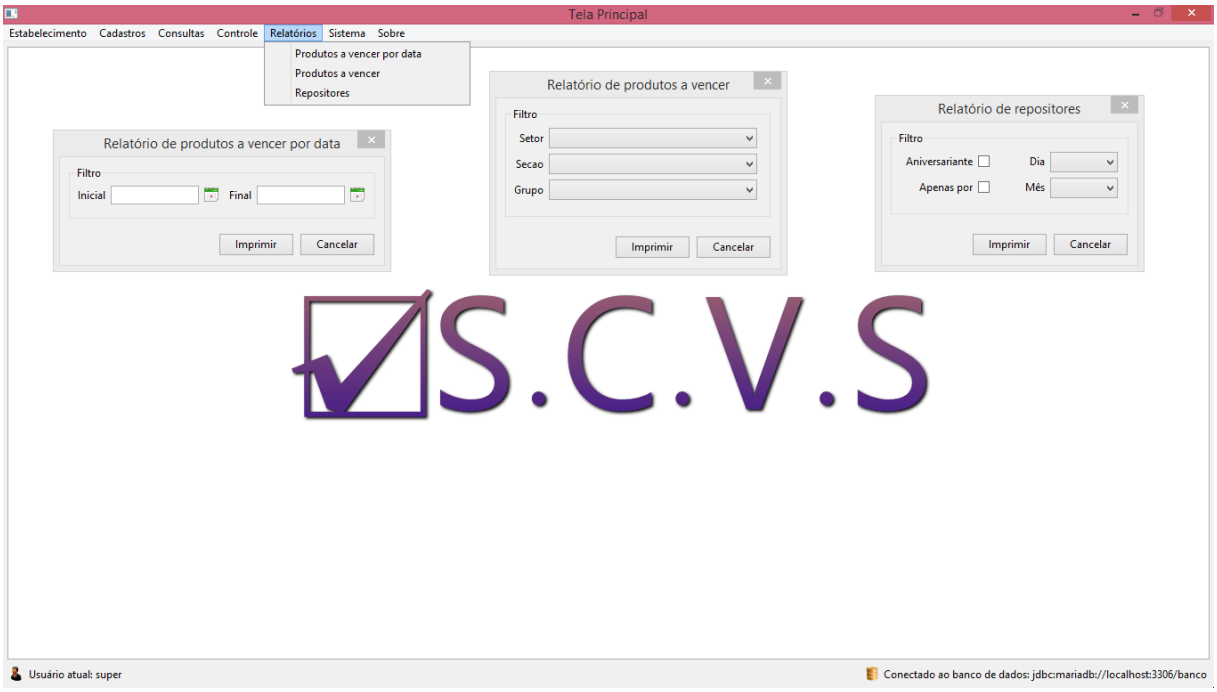

Figura 51 – Telas relatórios

– Outras Telas

Telas finais do sistema, nas quais fornecem a opção de ajustar para que o aplicativo pergunte para o usuário logo após sua saída do sistema se ele deseja salvar o backup do banco, como também pode optar por não receber essa mensagem, e também temos a opção de abrir os relatórios que serão emitidos no sistema em PDF, ou mandar o relatório diretamente para a fila da impressora.

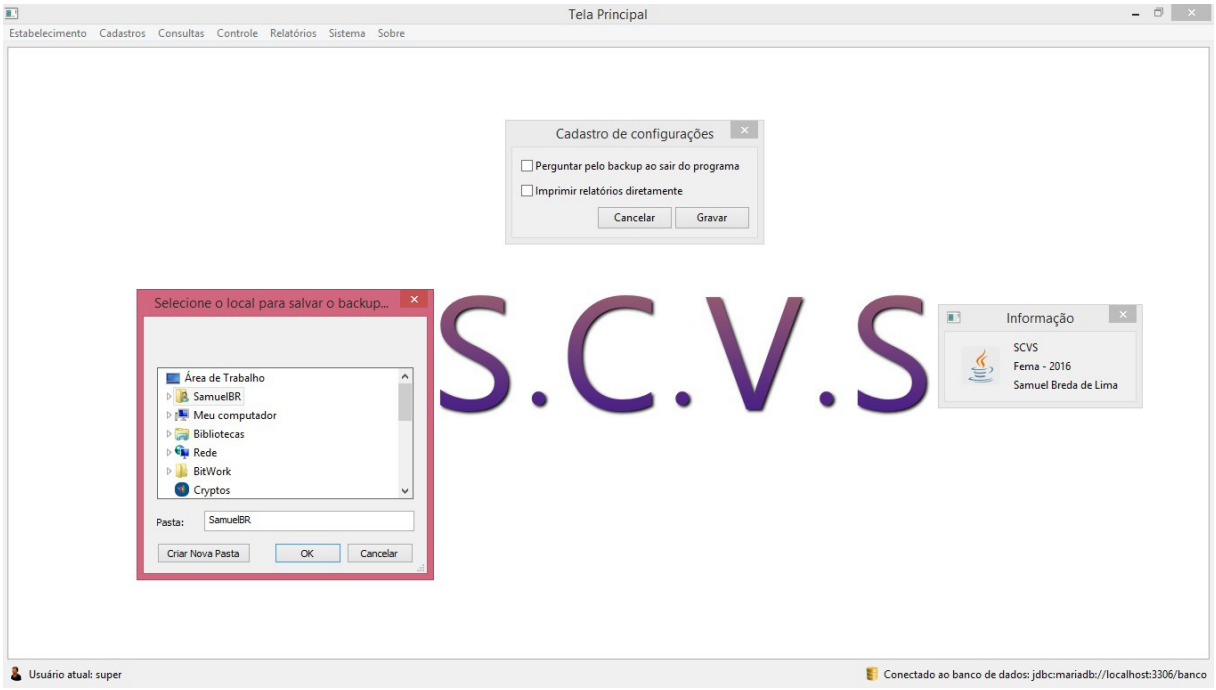

Figura 52 – Outras Telas

4.8 CONCLUSÃO

O software “Sistema Controle De Vencimento Para Supermercados” foi desenvolvido, com o intuito de informatizar e gerenciar supermercados e mercearias de pequeno e médio porte, facilitando o controle da data de vencimento de seus produtos, evitando prejuízos e constrangimento com clientes, para melhorar o acompanhamento do giro dessas mercadorias, serão registrando os produtos vencidos em um histórico mensal.
Para implementação do sistema, foi necessário primeiro realizar o levantamento de requisitos e das funcionalidades necessárias. Na análise foram utilizados os conceitos de UML, com os diagramas de Caso de Uso, Diagrama de Classes, Diagrama de Sequência e Diagrama de Entidade de Relacionamento. Tais diagramas foram de muita importância para um melhor entendimento das funcionalidades que o sistema deveria ter.

CRONOGRAMA

Abaixo, é apresentado o cronograma das atividades que foram desenvolvidas para o trabalho de conclusão do curso:

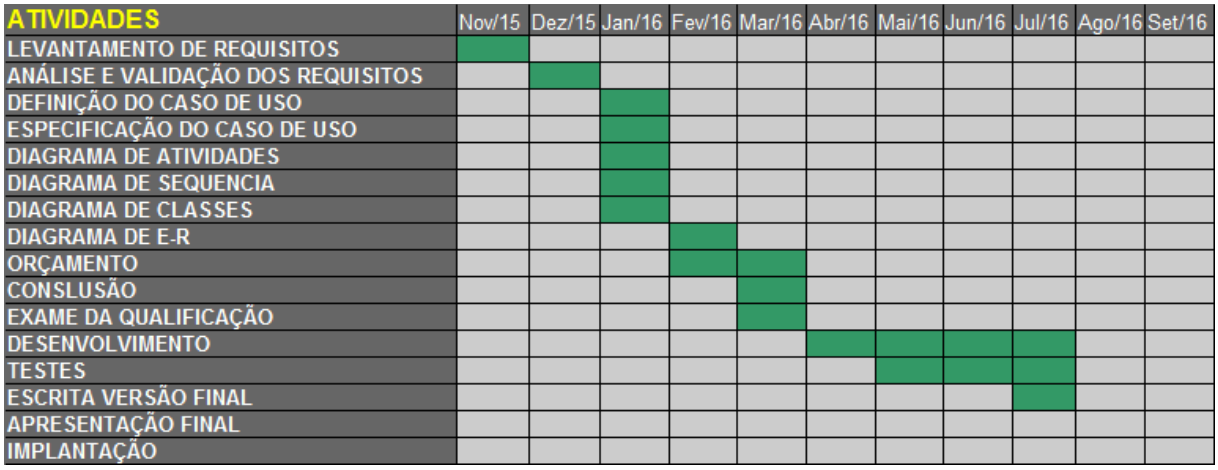

Figura 53 Cronograma

REFERÊNCIAS BIBLIOGRÁFICAS

Sousa, Viviane Sousa. **5 regras para evitar produto vencido.**
Disponível em:
<http://www.sm.com.br/Editorias/Gestao/5-regras-para-evitar-produto-vencido-16180.html>. Acesso em: 17 fev. 2016.

Prazo de validade dos alimentos deve estar claro aos consumidores.
Disponível em:
<http://www.idec.org.br/consultas/dicas-e-direitos/prazo-de-validade-dos-alimentos-deve-estar-claro-aos-consumidores>.
Acesso em: 17 fev. 2016.

Martins, Marcelo Martins. **Relatórios em Java – JasperReports e iReport**
Disponível em: <http://www.k19.com.br/artigos/relatorios-em-java-jasperreports-e-irepor/>. Acesso em: 17 fev. 2016.

Marins, Antomar Marins. **O Que são Mapas Mentais?**
Disponível em:
<http://www.administradores.com.br/artigos/economia-e-financas/o-que-sao-mapas-mentais/28259/>. Acesso em: 17 fev. 2016.

The Hibernate Team. **Hibernate Reference Documentation**
Disponível em:
<http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html_single/>.
Acesso em: 28 jul. 2016.

DEVMEDIA. **Conhecendo o SWT**
Disponível em: <http://www.devmedia.com.br/conhecendo-o-swt/3093/>.
Acesso em: 30 ago. 2016.

GUEDES, Gilleanes T. A. UML2 Uma Abordagem Prática, São Paulo: Novatec 2011.